ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA EN SISTEMAS

"INTEGRACIÓN DE SISTEMAS EN PLATAFORMAS HETEROGÉNEAS MEDIANTE EL USO DE DISPOSITIVOS HARDWARE, APLICADAS EN LA MUNICIPALIDAD DE BAÑOS"

TESIS DE GRADO

Previa la obtención del título de:
INGENIERO EN SISTEMAS INFORMÁTICOS

Presentado por:
WILLIAM OSWALDO PILCO YAMBAY
ROGELIO CRISTIAN GAVILANES MONTOYA

RIOBAMBA – ECUADOR
2009

Nuestros mas sinceros agradecimientos a Dios por guiarnos y vendecirnos día a día y permitirnos culminar este trabajo, a nuestros padres por su apoyo incondicional, a mis amigos por su sincera amistad, ayuda y consejos, a todos los docentes que han contribuido en mi formación académica, de manera especial a los que han colaborado en el desarrollo de esta tesis de grado.

Mil gracias a todos mis amigos, sin su ayuda no hubiese sido posible llegar a hacer realidad este reto.

William - Cristian

Dedico este proyecto y toda mi carrera universitaria a Dios por ser quien a estado a mi lado en todo momento, dándome las fuerzas necesarias para continuar lucahndo día tras día y seguir adelante rompiendo todas las barreras que se me presentan. Le agradezco a mi mamá Leonor Yambay y a mi papá Oswaldo Pilco ya que gracias a ellos soy quien soy hoy en día, fueron los que me dieron todo el cariño y calor humano, para seguir adelante y cumplirmis metas.

William Oswaldo Pilco Yambay

Dedico todo mi esfuerzo y empeño en la realización del presente trabajo, a mi familia de manera especial mis padres Rogelio Gavilanes y Teresita Montoya, quienes con su amor, apoyo constante han sabido impulsarme en cada uno de los retos tomados. A mis hermanos Viviana, Alex y Carolina, por su comprensión en las distintas etapas pasadas, a mi abuelito Vicente Montoya, quien con su sabiduría ha podido enseñarme el verdadero significado de la vida, de manera muy especial dedico este trabajo a mis amigos, Ing. Wladimir Castro, Dr. Antonio Fray quienes han confiado en mi capacidad, y siempre me han inculcado el espiritu de lucha y constancia.

Rogelio Cristian Gavilanes Montoya

FIRMAS DE RESPONSABILIDADES

NOMBRE	**FIRMA**	**FECHA**
Dr. Romeo Rodríguez DECANO FACULTAD DE INFORMATICA Y ELECTRONICA	……	……
Ing. Iván Menes DIRECTOR DE ESCUELA DE INGENIERIA EN SISTEMAS	……..	……
Ing. Wladimir Castro S DIRECTOR DE TESIS	……	……
Ing. Patricio Moreno MIEMBRO DEL TRIBUNAL	……	……
Lcdo. Carlos Rodríguez DIR. CENTRO DE DOCUMENTACIÓN	……	……
NOTA DE LA TESIS	……	

“Yo, William Oswaldo Pilco Yambay, soy responsable de las ideas, doctrinas y resultados expuestos en esta Tesis, y el patrimonio intelectual de la Tesis de Grado pertenece a la ESCUELA SUPERIOR POLITECNICA DE CHIMBORAZO”

William Oswaldo Pilco Yambay

“Yo, Rogelio Cristian Gavilanes Montoya, soy responsable de las ideas, doctrinas y resultados expuestos en esta Tesis, y el patrimonio intelectual de la Tesis de Grado pertenece a la ESCUELA SUPERIOR POLITECNICA DE CHIMBORAZO”

Rogelio Cristian Gavilanes Montoya

INDICE GENERAL

CAPITULO IV

INDICE DE ABREVIATURAS

AME	Asociación de Municipalidades Ecuatorianas
BD	Base de Datos
CLR	Common Language Runtime
DLL	Dynamic Link Library (Bibliotecas de Enlace Dinámico)
E – R	Entidad – Relación
FK	Foreign Key (Llave Foránea)
FK	Foreing Key (Llave Foránea)
GCS	Sistema para la Comercialización de los Servicios Municipales
HTML	Hipertext Markup Lenguaje. (Lenguaje de Marcado Hipertexto).
I/O	Operaciones de Entrada Salida
IDE	Entorno de desarrollo integrado (Integrated Development Environment)
JDBC	Java Database Connectivity (Conectividad de base de datos Java)
JDK	Java Development Kit (Kit de desarrollo de Java)
JIT	Compilación Just-In-Time
LDAP	Lightweight Directory Access Protocol, Protocolo Ligero de Acceso a Directorios
LDB	Base de Datos Legal
ODBC	Origen de Datos de Usuario
PDA	Personal Digital Assistant (Asistente Personal Digital)
PDB	Archivo nativo de PalmOS para escribir los registros
PEINSEB	Proyecto para la Entrega Inmediata y Notificación de los Servicios Básicos
PK	Primary Key (Llave Primaria)
RDBMS	Sistema Manejador de Base de Datos Relacional.
SGBD	Sistemas de Gestión de Bases de Datos
SGBDR	Sistemas Gestores de Bases de Datos Relacionales
SISNYCS	Sistema de Notificación y Cobro de Servicios
SISTRAD	Sistema de Transferencia
SQL	Structured Query Language (Lenguaje Estructurado de Consultas)
SQL CE	SQL Compact Edition
XP	Extreme Programming (Programación Extrema)

INDICE DE TABLAS

INDICE DE FIGURAS

INTRODUCCION

Hace más de veinte años que empezó la revolución del ordenador con el PC de IBM. El PC de IBM llevó la computación lejos de los entornos con aire acondicionado de los mainframe o mini computadores y la puso en el escritorio de todo el mundo. Hoy en día la mayoría de los trabajadores tienen un PC en su mesa, y muchos tienen un PC también en su casa.

Los ordenadores portátiles permiten a los usuarios tener un computador que puede utilizarse en casa y en el trabajo, como también en el camino. Los PCS son dispositivos de computación generales que proporcionan un tremendo poder de computación y flexibilidad, y todos los PCS en el mundo desde ordenadores portátiles hasta ordenadores personales y servidores tienen fundamentalmente la misma arquitectura. La era del PC ha sido divertida, frustrante y excitante. Sin embargo hay una revolución más grande en camino con más potencial e incluso más retos – el movimiento hacia la computación basada en dispositivos móviles.

En los últimos años los dispositivos de computación que están llegando a las tiendas proporcionan una incomparable portabilidad y accesibilidad. Los dispositivos Microsoft Windows CE, como el dispositivo Palm Size, Pocket PC y el Handheld PC, proporcionan una reducción de las capacidades de un ordenador personal en un tamaño realmente pequeño, y el PALM Pilot ha sido un< exitoso PDA (Personal Digital Assistant). Las ventajas que ofrecen los sistemas PALM son que trabajan mejor con el manejo de información básica personal y contactos.

Suelen ser bastante más pequeños y ligeros, además de más baratos y tienen una duración increíblemente superior a los Pocket PC. Los Pocket PC son PDA que suelen venir con pantalla a todo color, y con capacidades multimedia, pero son más caras y la duración de sus baterías es inferior. Microsoft Pocket PC tiene unas tremendas características para proyectos de computación, juegos, y entretenimiento. El sistema operativo Windows CE se ha empotrado

en muchos dispositivos (como bombas de gasolina y sistemas de producción) para monitorizar y controlar. Al contrario que los PC genéricos, estos dispositivos de computación no son iguales y están diseñados para propósitos específicos.

Se ha pensado en los ordenadores portátiles como dispositivos móviles, pero realmente son un modo adecuado de mover un PC de un escritorio a otro. Pensaren una situación en la que voy a una oficina de clientes, y cuando cruzó la puerta quiero chequear los nombres de las personas que están allí. Con un ordenador personal, tengo que encenderlo (asumiendo que la batería está cargada), esperar a que arranque el sistema operativo, entrar, ejecutar mi aplicación, y buscar la Información. La operación completa puede durar cinco minutos durante los cuales yo tengo que sufrir el aspecto intrigante de los recepcionistas. El mismo escenario con un verdadero dispositivo móvil es completamente diferente – con un instante para encenderlo y un clic de acceso a mi aplicación, yo tengo la información en 30 segundos.

Los dispositivos móviles están diseñados para rellenar los intervalos en nuestra vida en los que no tenemos acceso conveniente a la computación. Los ordenadores personales proporcionan capacidades de computación en el escritorio en casa y en el trabajo. Los dispositivos móviles permiten acceder a la computación mientras los viajes, en encuentros con clientes, en vacaciones, y en cualquier sitio. La computación no es solamente sobre trabajo, estos dispositivos pueden también entretener. Se puede escuchar la música favorita, jugar a un juego o leer un libro.

Con estos dispositivos se puede sincronizar la información existente en el ordenador personal como contactos y tareas que pueden descargarse al dispositivo. Consecuentemente, muchas organizaciones están ahora empezando a producir estrategias para adoptar y soportar los dispositivos móviles. Estos dispositivos tienen la capacidad de bajar, actualizar y manipular datos de bases de datos, de Internet y de intranet.

Cuando estos dispositivos se combinan con otras tecnologías, las posibilidades pueden llegar a ser mucho más excitantes. Combinar un dispositivo computador con un teléfono GSM permite computación móvil con acceso a los datos incluso cuando el teléfono no está conectado. Los servidores pueden poner datos en los dispositivos sin la intervención del usuario – el dispositivo incluso puede encenderse el mismo al recibir datos. Incorporando soporte GPS (Global Positioning System), la localización de un dispositivo puede ser determinada con mucha precisión, y esto puede utilizarse para dirigir al usuario a un destino, como una estación de gasolina o una tienda de café.

Últimamente se ha notado una gran evolución en la versatilidad y comercialización de los Asistentes Personales Digitales (PDA, Iniciales en Inglés). Estos diminutos equipos de alta tecnología te ayudan de manera asombrosa en el control y manejo de tus actividades diarias, la mayoría de ellos poseen una serie de funciones básicas, tales como: manejador de correo Electrónico (e-mail), programador de citas, apuntador de notas, manejador de recordatorios, agenda telefónica, administrador de gastos, etc.; convirtiéndolos en la herramienta ideal para cualquier persona que desee administrar su tiempo y recursos eficientemente. Las nuevas tecnologías, tanto en hardware como en software, le han dado una mayor potencialidad a los PDAs, colocándolos a un nivel que muy fácilmente pudiesen desplazar al computador portátil en la mayoría de las aplicaciones de campo, viéndose fuertemente favorecido por su reducido tamaño y bajo costo cuando se compara con este último.

Las tecnologías de la información tienen cada vez una mayor presencia en nuestras vidas. Por otro lado, los computadores se están haciendo cada vez más portables, potentes y asequibles. La aparición de una nueva generación de dispositivos móviles, apoyada en los avances que se están produciendo en el ámbito de las telecomunicaciones, representa una nueva línea de actuación. Todo ello está llevando a su uso en ámbitos para los cual no se pensó inicialmente. Y uno de los que se puede ver más beneficiado de este hecho, es el campo de la educación.
La aparición de la computación móvil permite a los usuarios el acceso a los recursos “en cualquier momento y en cualquier lugar”. Esta nueva forma de operar cambiará, por

consiguiente, el modo en que la información es usada y compartida en entornos de enseñanza. Entre los nuevos dispositivos de reciente aparición cabe destacar las PDAs. Estos elementos permiten una gran variedad de tareas, entre las que destacan el acceso a Internet, la lectura de libros en formato electrónico, la escritura, envío y recepción de e-mails, y el registro de información. Actividades, todas ellas, de gran utilidad dentro del ámbito escolar.

Pero la plena introducción de estos elementos móviles no se apoya tanto en el desarrollo y abaratamiento de los mismos, sino en la creación de un software que de soporte a nuevas aplicaciones.

CAPITULO I

1 MARCO REFERENCIAL

1.1 PROBLEMATIZACIÓN

En la actualidad el uso de dispositivos móviles para el tratamiento de la información dentro de entidades públicas y privadas no es gran auge dentro de nuestro país, administrando la información dentro de proyectos de planificación que integran información alfanumérica relacional.

AME (Asociación de Municipalidades Ecuatorianas), por medio de uno de sus departamentos; UTIC (Unidad de tecnología y Conectividad), firmó un pre-acuerdo con la ESPOCH (Escuela Superior Politécnica de Chimborazo), a inicios del año 2008; en el cual se establece el beneficio mutuo entre tesistas, egresados de la ESPOCH y las Municipalidades del Ecuador; para solventar las necesidades de ambas partes.

En la actualidad muchos de los Municipios pertenecientes a la Regional 3 del AME, en sus Departamentos de Agua potable hacen uso del sistema GCS, SISTEMA DE COMERCIALIZACIÓN DE SERVICIOS MUNICIPALES, el mismo que es utilizado para la facturación y cobro por el uso de los servicios básicos como son, Agua Potable, Alcantarillado, y Recolección de Basura, así como también para la venta a crédito y al contado de medidores de agua potable, a los usuarios registrados y catastrados.

Las lecturas del consumo de Agua potable, es el punto más importante a tener en cuenta, ya que de aquí se derivan los valores a ser cancelados para el resto de servicios básicos; y es en este punto en donde han existido una gran infinidad de inconvenientes, por varios motivos, entre ellos la toma de lecturas erróneas, las mismas que provocan conflictos tanto para los clientes como para el personal que labora en dichos departamentos, ocasionando una serie de reclamos dentro de la institución, y por ende la inconformidad de los clientes con la institución.
Además la pérdida de tiempo generada desde el momento en que sale el personal municipal a la toma de lecturas, hasta el momento de la emisión de facturas del mes a ser cobrado, ocasiona un caos institucional, así como un desperdicio económico y de tiempo por parte del personal que realiza las tareas de facturación y digitación de los datos leídos en los medidores.

Es en vista de aquello se ha llegado a un acuerdo con la Municipalidad de Baños para solucionar una imperiosa necesidad en la automatización informática en el proceso de recolección de datos sobre los servicios básicos prestados por la institución, específicamente con lo que tiene que ver al servicio de recolección de la información del consumo de Agua Potable y otros servicios básicos, para su posterior notificación inmediata; con esto se pretende automatizar de una manera rápida y sencilla todos los problemas para realizar la facturación de las planillas por el concepto del consumo de servicios básicos.

1.2 JUSTIFICACION

Se lo hará en función a una justificación Teórica, Metodológica y Practica.

1.2.1 JUSTIFICACIÓN TEÓRICA

El presente trabajo de investigación se dará debido a que se requiere de una indagación de cómo integrar sistemas con plataformas heterogéneas ya que la administración de grandes volúmenes de mensajes exige una infraestructura flexible, ampliable y

segura en tiempo real, y basada en eventos que aprovecha aplicaciones y dispositivos del mismo tipo, todo esto aplicado para el proceso de recaudación del servicio de agua potable en la Ilustre Municipalidad de Baños, cabe resaltar que además se realizará un análisis del hardware y software empleado para su correcto funcionamiento.

1.2.2 JUSTIFICACIÓN METODOLÓGICA

Investigación Aplicada

Ya que los conocimientos adquiridos en nuestra vida estudiantil serán explotados al máximo con la finalidad de conseguir incrementar nuestros conocimientos científicos o filosóficos, pero sin contrastarlos con ningún aspecto práctico, y de esta manera buscar la aplicación o utilización de los conocimientos que se adquirieron.

Investigación de Campo.

Porque la información requerirá para la realización de este trabajo de investigación se apoya en informaciones que provienen entre otras, de entrevistas, cuestionarios, encuestas y observaciones.

Y con el fin de evitar duplicaciones de trabajo nos apoyaremos en la investigación de carácter documental.

1.2.3 JUSTIFICACIÓN PRÁCTICA

Tomando en consideración que se cuenta con el apoyo logístico por parte del AME, ESPOCH, y de la Municipalidad de Baños en principio, se ha decidido realizar este proyecto ya que es indispensable contar con una aplicación móvil para este fin, debido a los siguientes motivos:

- Facilitará el trabajo del personal municipal al momento de hacer las lecturas de los medidores de agua potable.
- Solucionará la pérdida de tiempo por parte del personal municipal al momento de recolectar información de los medidores de agua y al emitir las facturas para un mes determinado.
- Proveerá de información confiable y actualizada de los clientes.
- Proporcionará información detallada de los clientes notificándoles de los pagos con los principales aspectos a ser tomados en cuenta.
- Evitará conflictos internos entre el personal de los Departamento de Agua Potable, así como también con los clientes al momento de realizar los pagos respectivos.

1.3 OBJETIVOS

1.3.1 OBJETIVO GENERAL

Integrar Sistemas con Plataformas Heterogéneas mediante el uso de dispositivos hardware, aplicadas en la Municipalidad de Baños, y que se puedan ejecutar como un apoyo social mediante la vinculación de la ESPOCH con la sociedad.

1.3.2 OBJETIVOS ESPECIFICOS

- Analizar y Diseñar la Integración de dispositivos Hardware y Software en este caso los dispositivos empleados para la recolección de datos y el software implementable para dispositivos móviles para optimizar el proceso de recolección de información en la Municipalidad de Baños.
- Implementar la conectividad entre dispositivos de diferentes características técnicas en plataformas Heterogéneas, para un mejor desempeño en el funcionamiento de los dispositivos.

- Comprobar que diversos dispositivos Hardware pueden funcionar como un medio de almacenamiento de información mediante una Base de Datos para aumentar el rendimiento.
- Comparar, analizar y establecer las mejores herramientas para el desarrollo en dispositivos móviles, para explorarlas de mejor manera acorde a sus características técnicas.
- Generar una aplicación, mediante la utilización de la herramienta que mejor se ajuste en la implementación de la parte lógica y programable de los dispositivos a utilizar, para una recolección precisa en la información requerida.
- Generar series de códigos de barra, para una mejor organización e identificación de los Clientes Abonados del Servicio.
- Implementación y configuración de un Servidor Centralizado de Base de Datos con SQL SERVER, con el propósito de actualizar constantemente la información recogida periódicamente por parte del personal encargado.

1.4 HIPÓTESIS

"La integración de sistemas en Plataforma Heterogéneas, mediante el uso de dispositivos hardware, mejoraran los procesos tradicionales para la recopilación de la Información manejada por las Municipalidades de Baños".

1.5 MÉTODOS Y TÉCNICAS

La investigación a realizar es de tipo Aplicada puesto que es un trabajo científico que busca obtener conocimientos e informar sobre las herramientas de entorno visual para el desarrollo de Aplicaciones Móviles determinando así cual es la que brinda mejores prestaciones y aplicarla al desarrollo del sistema de cobros de las Ilustre Municipalidad de Baños

1.5.1 MÉTODOS

Método Científico: Se utilizará este método para la recolección de información y desarrollo de la investigación.

Método Analítico Sintético: Se utilizará para descubrir los distintos elementos que componen la naturaleza o esencia del fenómeno investigado, las causas y los efectos y por otro lado con respecto a lo Sintético, integra los elementos en una unidad nueva, en una comprensión total de la esencia de lo que ya se conoce en todos sus elementos y particularidades.

1.5.2 TÉCNICAS

Encuestas: Se usará esta técnica para poder determinar el grado de uso y los beneficios de las herramientas a ser analizadas.

Entrevistas: Usaremos esta técnica para llegar a definir los requerimientos del personal destinado al uso de la aplicación.

Revisión de Documentación: Utilizaremos esta técnica para la recolección de información más importante para nuestro análisis de la herramientas de entorno visual.

Consultas bibliográficas: Se utilizara para la recolección de información.

Pruebas prácticas: Se utilizará para determinar la funcionalidad en tiempo real de las herramientas a estudiarse.

CAPITULO II

2 MARCO TEORICO

2.1 DISPOSITIVOS POCKET PC

Antes de comentar las características de las que disponen hoy en día los dispositivos Pocket Pc, se verá una breve historia de los dispositivos móviles. Tras se expondrán las características hardware y software que actualmente poseen estos dispositivos.

2.1.1 Breve Historia de los Dispositivos Móviles

La primera compañía en desarrollar un dispositivo móvil fue Psion. Esto ocurría en 1984 y el dispositivo lanzado se llamó Psion Organiser y se puede englobar dentro de la categoría de organizadores personales. Este dispositivo trabaja con tecnología de 8 bits, tiene una pantalla LCD de 16 caracteres, almacenamiento en cartuchos y hasta dispone de un lenguaje de programación basado en BASIC llamado POPL (Psion Organiser Programming Language).

Figura 1. Psion Organiser

La Psion Organiser fue superada por la Psion II que salió al mercado en 1986. Este modelo dispone de una pantalla mayor, mayor capacidad de cálculo y almacenamiento. Fue usado mayormente como punto de ventas.

En 1993 fue lanzada la Psion Series 3a. Este modelo aportaba una interesante capacidad: podía conectarse con un ordenador y transferir (y convertir) datos entre los dos entornos. Esto hizo que Psion dominara el mercado de los asistentes personales digitales (PDA) por un par de años. Debido al éxito de la Series 3a, Psion lanzó la Series 3c que era mucho más potente.

En 1997 lanzó la Series 5 que contaba con un procesador de 32 bits. En 2001, y superada por Palm y Windows CE, Psion decidió retirarse del mercado de los PDA. El primero que utilizo el concepto de PDA[1] (Personal Digital Assistant) fue John Sculley, Presidente de Apple, en enero de 1992. Podemos definir un PDA como un “pequeño computador de mano que podemos usar para tomar notas, apuntar citas y otro tipo de cosas que mantengan nuestra vida en orden. Un PDA provee toda la funcionalidad de una pequeña libreta de notas pero usada muchas veces, y con una cierta capacidad de almacenamiento”.

[1] Personal Digital Assistant (Asistente Digital Personal)

El primer dispositivo en ofrecer una funcionalidad parecida a la definición anterior fue el Newton MessagePad de Apple Computers. Salió al mercado en Agosto de 1993. Este dispositivo innovó por la utilización de un lápiz para trabajar con él, el reconocimiento de escritura, la capacidad de almacenamiento de información personal y sus capacidades de comunicación. Fue un éxito instantáneo y pronto surgieron un gran número de competidores.

Figura 2. Apple Newton MessagePad 130

Esta generación de dispositivos tenía un problema: el reconocimiento de escritura no era todavía muy bueno. El reconocimiento de escritura en el Newton era muy sofisticado e intentaba aprender la escritura de su usuario y convertirla a texto. Aunque fue un intento valiente, este método de reconocimiento de escritura se tornó irrealizable. No fue hasta Marzo de 1996 que alguien consiguió realizar un reconocimiento de escritura medianamente razonable. La empresa Palm Computing Inc[2]. Lanzó al mercado su primer PDA Palm Pilot.

Este dispositivo usaba su propio lenguaje de caracteres llamado Grafiti, este se podía aprender en 20 minutos, el cual facilitaba el reconocimiento de escritura y conseguía mejores resultados.

[2] Fabricante de PDAs y smartphone con sede en Sunnyvale

Figura 3. Palm Pilot 5000.

El Palm Pilot comenzó a venderse en mayor cantidad que sus competidores, y al fin de 1999 ya copaba el 70 % del mercado mundial de PDAs, y además, 20.000 desarrolladores trabajaban escribiendo software para este dispositivo. A partir de 2000 y viendo la amenaza que suponía Microsoft, la compañía Palm Computing Inc. comenzó a emitir licencias para que otros fabricantes de hardware fabricaran dispositivos que integraran el sistema operativo Palm OS.
Viendo que esta oportunidad era muy buena para perderla, Microsoft entró en el mercado en otoño de 1996 con su sistema operativo Windows CE, este era soportado por unas 40 compañías que desarrollarían hardware y software compatible. Sin embargo, las primeras dos versiones del sistema operativo no fueron bien acogidas por el mercado: los primeros dispositivos que usaban Windows CE consumían mucha energía (la batería duraba poco) y el interfaz gráfico intentaba imitar el de Windows, resultando un interfaz no muy cómodo para un dispositivo de mano.

Los primeros dispositivos que usaban Windows CE, los denominados Handheld PC, disponían de pantallas de 480 por 240 y 640 por 240, y tenían teclados. Cuando se presentó Windows CE 2 el estándar de la pantalla se quedó en 640 por 240 de forma horizontal y el teclado era un poco mayor.

Figura 4. Philips Velo

En Enero de 1998 Microsoft presentó dos nuevas plataformas que ejecutarían Windows CE: los Palm-size PCs y los Auto PC. El Palm-size PC era un producto para competir directamente con los dispositivos basados en Palm-OS. Los Palm-size PC se basaban en una pantalla vertical de 240 por 320 puntos y el uso de un lápiz para la entrada de datos. Al principio no tuvieron una acogida muy entusiasta.

El año siguiente apareció una nueva clase de máquinas de estilo miniordenador portátil basadas en Windows CE con teclados sensibles al toque y pantallas VGA o Super VGA.
Estos dispositivos también incluían modem interno, una duración de batería de 10 horas y algunos cambiaron la, entonces estándar, pantalla táctil por un trackpad. Estos dispositivos recibieron el nombre de H/PC Professional (H/PC ↔ Handheld PC).

En Abril de 2000 Microsoft lanzó el Pocket PC, una versión bastante mejorada del viejo Palm-size PC. El Pocket PC usaba una pre-versión del Windows CE 3. El interfaz de usuario de la Pocket PC era diferente de sus predecesores: se había sustituido el escritorio por la pantalla “Hoy” que era mucho más limpia y serviría como lugar inicial del sistema. Una de las más importantes mejoras que tuvo el sistema fue la mejora del rendimiento, algo que se complementó con la inclusión en los dispositivos de CPUs más rápidas. Esto permitió que estos dispositivos funcionaran como verdaderos organizadores de bolsillo.

Figura 5. Dell Axim

Los sistemas operativos Palm OS y Pocket PC se han convertido hoy en día en verdaderos sistemas de propósito general que son capaces de ejecutar todo tipo de aplicaciones. Como contrapunto a esto, e intentando mantener los motivos originales que llevaron a la creación de los dispositivos móviles, han surgido dispositivos como los BlackBerry de la compañía Research In Motion.

Figura 6. BlackBerry 7100t

Una BlackBerry es un dispositivo inalámbrico que provee servicios de teléfono agenda, correo electrónico, mensajes de texto y navegador web. Podríamos decir que son un híbrido entre teléfono móvil y PDA. Es un teléfono móvil con capacidades de organizador y sin características extra adicionales. Una de las características más valoradas por los usuarios es la capacidad de enviar y recibir el correo electrónico en cualquier lugar.

Hoy en día el mercado de los dispositivos de mano podemos decir que se reparte de la siguiente forma:

• Dispositivos que ejecutan Palm OS con un 40,5 % de cuota de mercado.
• Dispositivos que ejecutan Pocket PC con un 40,4 % de cuota de mercado.
• Dispositivos que ejecutan BlackBerry OS con un 14,8 % de cuota de mercado.
• Dispositivos basados en el kernel de Linux 1,9 % de cuota de mercado.
• Dispositivos que ejecutan otros sistemas operativos 2,4 % de cuota de mercado.

2.1.2 DISPOSITIVOS POCKET PC

Ahora veremos las diferentes facetas de la plataforma Pocket PC.

2.1.2.1 Hardware

La mayoría de los dispositivos que cumplen con la especificación Pocket PC llevan incluido un procesador de la familia ARM, actualmente un Intel XScale. Este tipo de procesador está basado en la filosofía RISC[3] y, pensado para dispositivos móviles, diseñado para un consumo energético mínimo.

La memoria del sistema nos sirve de memoria RAM y de espacio de almacenamiento. Cuando ejecutamos un programa que está almacenado en el dispositivo, éste al estar ya almacenado

[3] (Del inglés *Reduced Instruction Set Computer*), Computadora con Conjunto de Instrucciones Reducidas.

en la misma memoria, se ejecuta directamente desde su ubicación. Si el programa estuviera ubicado en algún dispositivo externo, como una memoria Compact Flash, tendría que copiarse a la memoria para ser ejecutado. Esta memoria suele ser volátil, es decir, que en el momento que el dispositivo se quede sin batería se perderán los datos.

El sistema operativo se encuentra almacenado en una ROM. Esta ROM es flasheable para poder introducir actualizaciones del sistema operativo.

La pantalla del dispositivo Pocket PC suele ser de una resolución de 240 x 320 píxeles.

En las últimas versiones del sistema operativo se puede conseguir el girar la pantalla para trabajar de forma horizontal. Esta pantalla es además táctil, es decir, sensible al contacto. Con esto se consigue que no haga falta disponer de un dispositivo como un ratón para poder acceder a cualquier parte de la pantalla. Todos los dispositivos Pocket PC se proporcionan con un lápiz que hace las veces de ratón.

Los dispositivos Pocket PC poseen hoy en día muchas alternativas para su comunicación con otros dispositivos. Al principio, sólo tenían una interfaz serie para poder conectarse con un PC.
En la especificación Pocket PC 2000 se les introdujo una interfaz IrDA como obligatoria.
Otra forma de conseguir conectividad con el exterior ha sido a base de tarjetas de expansión. A las primeras versiones de Pocket PC se les podía instalar una tarjeta de red externa.

Hoy en día los dispositivos ya vienen con Bluetooth y WiFi de serie. Lo que no es tan común es que un Pocket PC integre un teléfono móvil. Hoy en día disponemos de muchos modelos que lo hacen.

Esto es debido a que Microsoft ha llevado el sistema operativo hasta los teléfonos móviles, no en vano la última versión de Windows CE se llama ahora Windows Mobile. Los dispositivos que ejecutan Windows Mobile disponen de teléfono móvil y además GPRS.

2.1.2.2 Sistema Operativo

Conocida la capacidad del hardware de los dispositivos Pocket PC, veamos ahora la capacidad de su sistema operativo.

El sistema operativo Windows CE, conocido después como Pocket PC y últimamente como Windows Mobile, es un sistema operativo de 32 bits, orientado a objetos y modular. Esto último es una característica muy importante debido a las importantes diferencias existentes entre el hardware de algunos dispositivos que lo utilizan. Si echamos un vistazo a su arquitectura podremos comprobar como parte del sistema es proporcionada directamente por el fabricante de hardware.

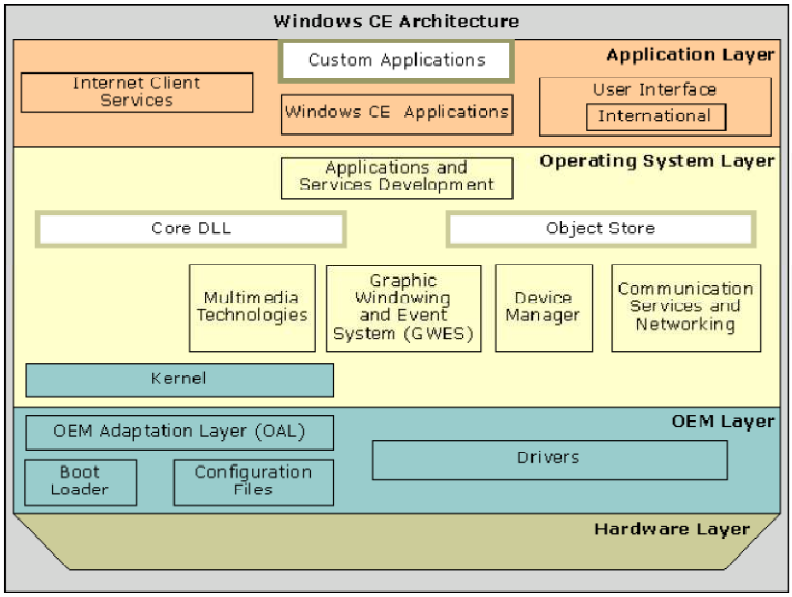


Figura 7. Arquitectura de Windows Ce

Ahora se mencionarán las características más importantes de este sistema operativo relacionadas con la programación de aplicaciones:

• Soporte de las APIs de Win32: debido al reducido tamaño de la memoria no toda la API está soportada, pero sí un subconjunto. Algunas de las funciones que siguen en la API también tienen sus características recortadas.

• Unicode: el formato de texto nativo en los dispositivos Pocket PC es Unicode. Esto nos da las ventajas de poder usar caracteres especiales y hacer aplicaciones multilingües.

• Interfaz de Usuario: la interfaz está pensada para tener una sola ventana en pantalla al mismo tiempo. Otro aspecto que cambia es el redibujado de la pantalla: en Pocket PC la pantalla no se redibuja con tanta frecuencia como en el Windows normal, el redibujo es forzado por la aplicación que se ejecuta.

El sistema operativo también viene acompañado por versiones de bolsillo (Pocket) de muchos programas conocidos de la plataforma Windows, entre ellos: Pocket Word, Pocket Excel, Pocket Internet Explorer, etc.

2.2 Plataformas .Net, .Net Compact Framework y Lenguaje C#

En este tema se conocerá la plataforma y el lenguaje de programación sobre los que vamos a desarrollar la aplicación. Primero veremos la plataforma .NET, sus orígenes y características, tras esto comentaremos las diferencias que tiene con .NET Compact Framework y por último conoceremos, muy por encima, las características del lenguaje de programación C#.

2.2.1 La plataforma .NET

La compatibilidad hacia atrás ha sido una característica importante de la tecnología Windows y una de las responsables de la fortaleza de la plataforma Windows. Cada vez que la tecnología va aportando nuevas características es un poco más complicado mantener dicha compatibilidad hacia atrás. De forma sencilla podemos decir que .NET es un nuevo framework

(una API) para programar en la plataforma Windows que hace que el problema de la compatibilidad hacia atrás por mejoras tecnológicas sea algo más sencillo de solventar.

Según Microsoft, ".NET Framework es una nueva plataforma informática que simplifica el desarrollo de aplicaciones en un entorno altamente distribuido como es Internet". El .NET Framework fue diseñado para cumplir los siguientes objetivos:

- Proporcionar un entorno coherente de programación orientada a objetos, en el que el código de los objetos se pueda almacenar y ejecutar de forma local, ejecutar de forma local pero distribuida en Internet o ejecutar de forma remota.

- Proporcionar un entorno de ejecución de código que reduzca lo máximo posible la implementación de software y los conflictos de versiones.

- Ofrecer un entorno de ejecución de código que garantice la ejecución segura del mismo, incluso del creado por terceras personas desconocidas o que no son de plena confianza.

- Proporcionar un entorno de ejecución de código que elimine los problemas de rendimiento de los entornos en los que se utilizan secuencias de comandos o intérpretes de comandos.

- Ofrecer al programador una experiencia coherente entre tipos de aplicaciones muy diferentes, como las basadas en Windows o en la Web.

- Basar toda la comunicación en estándares del sector para asegurar que el código de .NET Framework se puede integrar con otros tipos de código.

.NET tiene dos componentes principales: Common Language Runtime (CLR) y la biblioteca de clases de .NET Framework (BCL - Basic Class Library).

2.2.1.1 El Common Language Runtime (CLR)

El CLR administra la memoria, ejecución de subprocesos, ejecución de código, comprobación de la seguridad del código, compilación y demás servicios del sistema. Podríamos decir que esta es la máquina virtual de .NET. El código destinado al motor de tiempo de ejecución se denomina código administrado, a diferencia del resto de código, que se conoce como código no administrado.

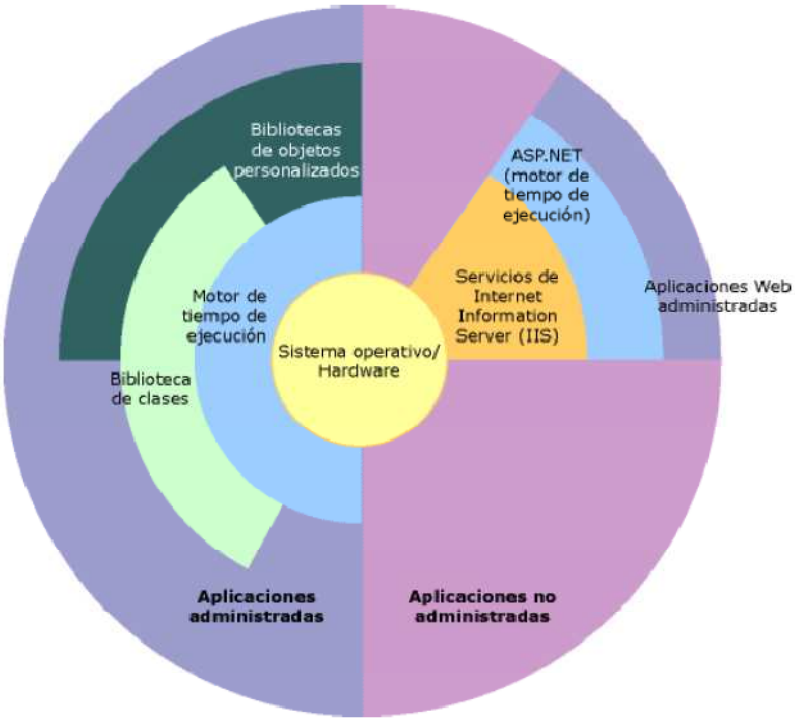


Figura 8. Sistema con .Net

Estas son las características que aporta el CLR:

Seguridad avanzada: En el CLR los componentes administrados reciben niveles de confianza diferentes en función de una serie de factores entre los que se incluye su localización (como

Internet, red empresarial o equipo local), origen (dependiendo del fabricante) y permisos o roles del usuario que lo esté ejecutando. Esto permite asegurar al administrador del sistema que el código que se esté ejecutando no pueda poner en peligro la integridad del sistema donde se ejecute.

Seguridad de tipos: El CLR impone la robustez del código mediante la implementación de un sistema estricto de comprobación de tipos y código denominada CTS (Common Type System, Sistema de Tipos Común). El CTS garantiza que todo el código administrado es autodescriptivo. Los diferentes compiladores de lenguajes de Microsoft y de terceros generan código administrado que se ajusta a CTS. Esto significa que el código administrado puede usar otros tipos e instancias administrados, al tiempo que se aplica inflexiblemente la fidelidad y seguridad de los tipos.

Gestión de memoria: El CLR mantiene el control de la ubicación de los objetos, administra las referencias a éstos y los libera cuando ya no se utilizan. Dispone de un recolector de basura, que es una aplicación que se activa cuando se quiere crear algún objeto nuevo y se detecta que no queda memoria libre para hacerlo. En tal caso, el recolector recorre la memoria dinámica asociada a la aplicación, detecta que objetos hay en ella que no pueda ser accedido por el código de la aplicación, y los elimina para limpiar la memoria de “objetos basura” y permitir la creación de otros nuevos. Esta administración automática de la memoria soluciona los dos errores más comunes de las aplicaciones: la pérdida de memoria y las referencias no válidas a la memoria.

Integración de lenguajes: Los programadores pueden crear aplicaciones en el lenguaje que prefieran y seguir sacando todo el provecho del motor de tiempo de ejecución, la biblioteca de clases y los componentes escritos en otros lenguajes por otros colegas. Los compiladores de lenguajes que se destinan a .NET Framework hacen que las características de .NET Framework estén disponibles para el código existente escrito en dicho lenguaje, lo que facilita enormemente el proceso de migración de las aplicaciones existentes. Microsoft ha desarrollado

varios compiladores, que generan código intermedio, para los lenguajes C#,C++,Visual Basic.NET, J#, Jscript, etc. Se puede hacer una clase en un lenguaje y llamar a esta clase desde otra con otro lenguaje.

Interoperatibilidad con código antiguo: Aunque el CLR está diseñado para el software del futuro, también es compatible con el software actual y el software antiguo. La interoperabilidad entre el código administrado y no administrado permite que los programadores continúen utilizando los componentes COM y las DLL que necesiten.

Soporte multihilo: El CLR es capaz de trabajar con aplicaciones divididas en múltiples hilos de ejecución que pueden ir evolucionando en paralelo o intercalándose, según el número de procesadores de la máquina sobre la que se ejecuten. Las aplicaciones pueden lanzar nuevos hilos, destruirlos, suspenderlos por un tiempo o hasta que les llegue una notificación, enviarles notificadores, sincronizarlos, etc.

Ejecución multiplataforma: El CLR actúa como una máquina virtual, encargándose de ejecutar las aplicaciones diseñadas para la plataforma .NET. Es decir, cualquier plataforma para la que exista una versión del CLR podrá ejecutar cualquier aplicación .NET.

Microsoft ha desarrollado versiones del CLR para la mayoría de las versiones de Windows: Windows 95, Windows 98, Windows ME, Windows NT 4.0, Windows 2000, Windows XP, Windows 2003 y Windows CE. Por otro lado Microsoft ha firmado un acuerdo con Corel para portar el CLR a Linux, aunque existen también terceros que están desarrollando de manera independiente versiones de libre distribución del CLR para Linux (proyecto Mono).

Otras características de .NET sólo podrán ser comprendidas por programadores con experiencia en el entorno Windows:

Modelo de programación sencillo: Con el CLR desaparecen muchos elementos complejos incluidos en los sistemas operativos actuales: registro de Windows, GUIDs, HRESULTS, Iunknown, etc. No es que el CLR abstraiga estos conceptos al programador, sino que son conceptos inexistentes en la plataforma .NET.

Eliminación del "Infierno de las DLLs": Este problema es de los sistemas operativos actuales de la familia Windows y consiste en que al sustituirse versiones viejas de DLLs compartidas por versiones nuevas puede que aplicaciones que fueron diseñadas para ser ejecutadas usando las viejas dejen de funcionar si no son 100% compatibles con las anteriores. En la plataforma .NET las versiones viejas de las DLLs pueden existir con las viejas, de modo que las aplicaciones diseñadas para ejecutarse con las viejas podrán seguir usándolas tras la instalación de las nuevas.

Aislamiento de procesos: El CLR asegura que desde código perteneciente a un determinado proceso no se pueda acceder a código o datos pertenecientes a otro, lo que evita errores de programación muy frecuentes e impide que unos procesos puedan atacar a otros. Esto se consigue gracias al sistema de seguridad de tipos antes comentado, pues evita que se pueda convertir un objeto a un tipo de mayor tamaño que el suyo propio, ya que al tratarlo como un objeto de mayor tamaño podría accederse a espacios en memoria ajenos a él que podrían pertenecer a otro proceso. También se consigue gracias a que no se permite acceder a posiciones arbitrarias de memoria.

Tratamiento de Excepciones: En el CLR todos los errores que se puedan producir durante la ejecución de una aplicación se propagan de igual manera: mediante excepciones. Esto es muy diferente a como se venía haciendo en los sistemas Windows hasta la aparición de la plataforma .NET, donde ciertos errores se transmitían mediante códigos de error en formato Win32, otros mediante HRESULTs[4] y otros mediante excepciones.

[4] Se utilizan como parámetros de la función y los valores de retorno para describir los errores y advertencias en un programa.

Distribución transparente: El CLR ofrece la infraestructura necesaria para crear objetos remotos y acceder a ellos de manera completamente transparente a su localización real, tal y como si se encontrasen en la máquina que los utiliza.

2.2.1.2 La Biblioteca de Clases de .NET

La biblioteca de clases de .NET Framework es una colección de tipos reutilizables que se integran estrechamente con el CLR. La biblioteca de clases está orientada a objetos, lo que proporciona tipos de los que su propio código administrado puede derivar funciones. Esto ocasiona que los tipos de .NET Framework sean sencillos de utilizar y reduce el tiempo asociado con el aprendizaje de las nuevas características de .NET Framework. Además, los componentes de terceros se pueden integrar sin dificultades con las clases de .NET Framework.

Como en cualquier biblioteca de clases orientada a objetos, los tipos de .NET Framework permiten realizar diversas tareas de programación comunes, como son la administración de cadenas, recopilación de datos, conectividad de bases de datos y acceso a archivos. Además de estas tareas habituales, la biblioteca de clases incluye tipos adecuados para diversos escenarios de desarrollo especializados. Por ejemplo, puede utilizar .NET Framework para desarrollar los siguientes tipos de aplicaciones y servicios:

- Aplicaciones de consola
- Aplicaciones GUI de Windows (Windows Forms)
- Aplicaciones de ASP.NET
- Servicios Web XML
- Servicios de Windows

Esta librería está escrita en MSIL (Microsoft Intermediate Language – Lenguaje Intermedio de Microsoft) que es un conjunto de instrucciones independiente de la CPU que se pueden

convertir de forma eficaz en código nativo. Por tanto esta librería puede ser utilizada desde cualquier lenguaje cuyo compilador genere MSIL[5].

Dada la amplitud de esta librería, ha sido necesario organizar las clases en ella incluida en espacios de nombres que agrupen clases con funcionalidades similares. Los espacios de nombres más usados son:

System: Tipos usados de forma frecuente. Por ejemplo tipos básicos, tablas, excepciones, fechas, números aleatorios, recolector de basura, entrada/salida en consola, etc.

System.Collections: Tipos usados como colecciones de datos de uso común. Por ejemplo pilas, colas, diccionarios, etc.

System.Data: Tipos usados para la manipulación de basas de datos. Forman la denominada arquitectura ADO.NET.

System.IO: Tipos usados para la manipulación de ficheros y otros flujos de datos.

System.Net: Tipos usados para las comunicaciones en red.

System.Runtime.Remoting: Tipos usados para el acceso a objetos remotos.

System.Security: Tipos basados para acceder a la política de seguridad del CLR.

System.Threading: Tipos usados para la manipulación de hilos de ejecución.

System.Web.UI.Webcontrols: Tipos usados para la creación de interfaces de usuario basadas en ventanas para aplicaciones Web.

[5] Es el lenguaje de programación legible por humanos de más bajo nivel en el Common Language Infrastructure y en el .NET Framework

System.Windows.Forms: Tipos usados para la creación de interfaces de usuario basadas en ventanas para aplicaciones estándar.

System.XML: Tipos para el acceso y manipulación de datos en formato XML.

2.1.1.1 **El Proceso de Ejecución Administrada**

Ahora se detallará como es el proceso de compilación y ejecución de un programa sobre la plataforma .NET.

Paso 1: Para compilar primero debemos de elegir un compilador. Como antes se ha mencionado, para aprovechar las ventajas que ofrece Common Language Runtime, se deben utilizar uno o varios compiladores de lenguaje orientados al tiempo de ejecución, como Visual Basic, C#, Visual C++, Jscript o uno de los muchos compiladores de otros fabricantes comoun compilador Eiffel, Perl o COBOL.

Paso 2: Tras elegir el compilador, este procederá a convertir nuestro código fuente a lenguaje intermedio de Microsoft (MSIL). MSIL incluye instrucciones para cargar, almacenar, inicializar y llamar a métodos en los objetos, así como instrucciones para operaciones lógicas y aritméticas, flujo de control, acceso directo a la memoria, control de excepciones y otras operaciones.
Cuando el compilador produce MSIL, también genera metadatos. Los metadatos describen los tipos que aparecen en el código, incluidas las definiciones de los tipos, las firmas de los miembros de tipos, los miembros a los que se hace referencia en el código y otros datos que el motor de tiempo de ejecución utiliza en tiempo de ejecución. El lenguaje intermedio de Microsoft (MSIL) y los metadatos se incluyen en un archivo ejecutable portable (PE), que se basa y extiende el PE de Microsoft publicado y el formato Common Object File Format (COFF) utilizado tradicionalmente para contenido ejecutable. Este formato de archivo, que contiene

código MSIL o código nativo así como metadatos, permite al sistema operativo reconocer imágenes de Common Language Runtime.

Paso 3: Ahora se convierte el código MSIL a código nativo. Esto lo realiza mediante un compilador Just-In-Time (JIT) de .NET Framework. Common Language Runtime proporciona un compilador JIT para cada arquitectura de CPU compatible, por lo que los programadores pueden escribir un conjunto de MSIL que se puede compilar mediante un compilador JIT y ejecutar en equipos con diferentes arquitecturas. No obstante, el código administrado sólo se ejecutará en un determinado sistema operativo si llama a las API[6] nativas específicas de la plataforma o a una biblioteca de clases específica de la plataforma.

La compilación JIT tiene en cuenta el hecho de que durante la ejecución nunca se llamará a parte del código. En vez de utilizar tiempo y memoria para convertir todo el MSIL de un archivo ejecutable portable (PE) a código nativo, convierte el MSIL necesario durante la ejecución y almacena el código nativo resultante para que sea accesible en las llamadas posteriores. El cargador crea y asocia un código auxiliar a cada uno de los métodos del tipo cuando éste se carga. En la llamada inicial al método, el código auxiliar pasa el control al compilador JIT, el cual convierte el MSIL del método en código nativo y modifica el código auxiliar para dirigir la ejecución a la ubicación del código nativo. Las llamadas posteriores al método compilado mediante un compilador JIT pasan directamente al código nativo generado anteriormente, reduciendo el tiempo de la compilación JIT y la ejecución del código.

Paso 4: Para poder ejecutar un método, primero se debe compilar a código específico del procesador. Los métodos para los que se genera el Lenguaje intermedio de Microsoft (MSIL) se compilan mediante un compilador JIT cuando se les llama por primera vez y, a continuación, se ejecutan. La próxima vez que se ejecuta el método, se ejecuta el código nativo existente resultante de la compilación JIT. El proceso de compilación JIT y, a continuación, la ejecución de código se repiten hasta completar la ejecución.

---

[6] Interfaz de programación de aplicaciones

2.2.2 .NET Compact Framework

Podemos decir que .NET Compact Framework es un subconjunto de .NET Framework orientado a dispositivos móviles tales como PDAs, teléfonos móviles, etc. Muchas de las funcionalidades de .NET se ven disminuidas en esta versión debido, mayormente, a temas de rendimiento y tamaño del Common Language Runtime.

Las siguientes son algunas de las características que cambian en .NET Framework:

Clases: .NET Compact Framework acepta un subconjunto de la Biblioteca de clases de .NET Framework. Dicho subconjunto es apropiado para las aplicaciones diseñadas para su ejecución en dispositivos con limitaciones de recursos y es semánticamente compatible con las clases del mismo nombre en .NET Framework.

Interoperatibilidad COM y funciones de devolución de llamada: .NET Compact Framework no admite la interoperabilidad con objetos COM, aunque puede utilizar la invocación de la plataforma (PInvoke) para obtener acceso a funciones DLL nativas que sí pueden llamar a objetos COM. Los controles ActiveX, Windows Scripting Host y las funciones de devolución de llamada que utilizan interoperabilidad COM no son compatibles.

Datos: .NET Compact Framework proporciona una implementación mediante un subconjunto de ADO.NET e incluye el proveedor de datos de SQL Server CE .NET. No está soportado ODBC.

Nombres de Archivo y Rutas de acceso: El esquema archivo:// URI se procesa de forma diferente, por lo que se recomienda utilizar nombres de archivo y rutas de acceso absolutos.

Globalización: Por motivos de compatibilidad y eficiencia del tamaño, la compatibilidad con la globalización, como formatos de fecha y tablas de clasificación apropiadas para la configuración regional, remite al sistema operativo subyacente siempre que es posible.

Entrada / Salida: A causa de las diferencias entre los sistemas operativos, existen restricciones y limitaciones en el modelo de E/S.NET Compact Framework no proporciona notificaciones de cambios en los archivos. Dado que la E/S en los dispositivos ocurre en la RAM, no es posible definir ni obtener acceso a los atributos de archivos y directorios.

Memoria: .NET Compact Framework está optimizado para sistemas alimentados por baterías y evita el uso intensivo de ciclos de RAM y de CPU.

Redes: .NET Compact Framework proporciona clases IrDA (Infrared Data Association) para establecer conexiones por infrarrojos y clases de escucha de Web para atender peticiones de HTTP al dispositivo. Las clases siguientes sólo están disponibles en .NET Compact Framework.

Formularios Windows Forms y gráficos: Los controles de formularios Windows Forms están diseñados especialmente para .NET Compact Framework. Hay varias diferencias que deben tenerse en cuenta.

XML: Por consideraciones de tamaño, .NET Compact Framework no admite validación de esquemas de XML o consultas XPath en documentos XML. No se acepta el DOM (Modelo de objetos de documento) de XML.

2.2.3 El Lenguaje C#

C# es el nuevo lenguaje de propósito general diseñado por Microsoft para la plataforma .NET. Sus principales creadores son Scott Wiltamuth y Anders Hejlsberg.

Aunque es posible escribir código para la plataforma .NET en muchos otros lenguajes, C# es el único que ha sido diseñado específicamente para ser utilizado sobre ella, por lo que programar usando C# es mucho más sencillo e intuitivo que hacerlo con cualquiera de los otros lenguajes, disponibles en .NET, ya que C# carece de elementos heredados innecesarios.

Por esta razón, se suele decir que C# es el lenguaje nativo de .NET.
La sintaxis y estructuración de C# es muy parecida a la de C++ o Java, puesto que la intención de Microsoft es facilitar la migración de código y su aprendizaje a los desarrolladores habituados a ellos. Sin embargo, su sencillez y el alto nivel de productividad son comparables a los de Visual Basic.

Un lenguaje que hubiese sido ideal para estos menesteres es Java, pero debido a problemas con la empresa creadora del mismo (Sun Microsystems), Microsoft tuvo que desarrollar un nuevo lenguaje que añadiese, a las ya probadas virtudes de Java, las modificaciones que Microsoft tenía pensado añadirle para mejorarlo aún más y hacerlo un lenguaje orientado a desarrollo de componentes.
En resumen, C# es un lenguaje de programación que toma las mejores características de lenguajes preexistentes como Visual Basic, Java o C++ y las combina en uno solo. El hecho de ser relativamente reciente no implica que sea inmaduro, pues Microsoft ha escrito la mayor parte de la BCL (Basic Class Library) usándolo, por lo que su compilador es el más depurado y optimizado de los incluidos en el .NET Framework SDK.

2.2.3.1 Características de C#

A continuación recogemos, de forma resumida, las principales características de C#:

Sencillez: C# elimina muchos elementos que otros lenguajes incluyen y que son innecesarios en .NET. Como ejemplo: el código escrito en C# es autocontenido, lo que significa que no

necesita de ficheros adicionales al propio fuente tales como ficheros de cabecera o ficheros IDL.

Independencia de Tipos: El tamaño de los tipos de datos básicos es fijo e independiente del compilador, sistema operativo o máquina para quienes se compile (no como en C++), lo que facilita la portabilidad de código.

No se incluyen elementos poco útiles: De lenguajes como C++ tales como macros, herencia múltiple y la necesidad de un operador diferente del punto (.) para acceder a miembros de espacios de nombres (en C++ ::).

Modernidad: C# incorpora en el propio lenguaje elementos que se han mostrado útiles para el desarrollo de aplicaciones y que en otros lenguajes como Java o C++ hay que simular, como un tipo básico decimal que permita realizar operaciones de alta precisión con reales de 128 bits, la inclusión de una instrucción foreach que permita recorrer colecciones con facilidad y es ampliable a tipos definidos por el usuario, la inclusión de un tipo básico string para representar cadenas o la distinción de un tipo bool específico para representar valores lógicos.

Orientación a Objetos: Como todo lenguaje de programación actual de propósito general, C# es un lenguaje orientado a objetos, aunque eso es más bien una característica del CTS que de C#. Una diferencia de este enfoque orientado a objetos respecto al de otros lenguajes como C++, es que el de C# es más puro en tanto que no admite ni funciones ni variables globales sino que todo el código y datos han de definirse dentro de definiciones de tipos de datos, lo que reduce problemas por conflictos de nombres y facilita la legibilidad del código. C# soporta todas las características propias del paradigma de programación orientada a objetos:

Encapsulación, herencia y polimorfismo: En lo referente a la encapsulación es importante señalar que aparte de los típicos modificadores public, private y protected, C# añade un cuarto modificador llamado internal, que puede combinarse con protected e indica que al elemento a

cuya definición precede sólo puede accederse desde su mismo ensamblado. Respecto a la herencia, a diferencia de C++ y al igual que Java, C# sólo admite herencia simple de clases ya que la múltiple provoca más quebraderos de cabeza que facilidades y en la mayoría de los casos su utilidad puede ser simulada con facilidad mediante herencia múltiple de interfaces. De todos modos, esto vuelve a ser más bien una característica propia del CTS que de C#.

Por otro lado y a diferencia de Java, en C# se ha optado por hacer que todos los métodos sean por defecto sellados y que los redefinibles hayan de, marcarse con el modificador virtual (como en C++), lo que permite evitar errores derivados de redefiniciones accidentales. Además, un efecto secundario de esto es que las llamadas a los métodos serán más eficientes por defecto al no tenerse que buscar en la tabla de funciones virtuales la implementación de los mismos a la que se ha de llamar. Otro efecto secundario es que se permite que las llamadas a los métodos sean más eficientes por defecto al no tenerse que buscar en la tabla de funciones virtuales la implementación de los mismos a la que se ha de llamar.

Otro efecto secundario es que permite que las llamadas a los métodos virtuales se puedan hacer más eficientemente al contribuir a que el tamaño de dicha tabla se reduzca.

Orientación a componentes: La propia sintaxis de C# incluye elementos propios del diseño de componentes que otros lenguajes tienen que simular mediante construcciones más o menos complejas. Es decir, la sintaxis de C# permite definir cómodamente propiedades (similares a campos de acceso controlado), eventos (asociación controlada de funciones de respuesta a notificaciones) o atributos (información sobre un tipo o sus miembros).

Seguridad de tipos: C# incluye mecanismo que permiten asegurar que los accesos a tipos de datos siempre se realicen correctamente, lo que permite evitar que se produzcan errores difíciles de detectar por acceso a memoria no perteneciente a ningún objeto y es especialmente necesario en un entorno gestionado por un recolector de basura. Para ello se toman medidas del tipo: sólo se admiten conversiones entre tipos compatibles. Esto es, entre un tipo y

antecesores suyos, entre tipos para los que explícitamente se haya definido un operador de conversión, y entre un tipo y un tipo hijo suyo del que un objeto del primero almacenase una referencia del segundo (downcasting). Obviamente, lo último sólo puede comprobarlo en tiempo de ejecución el CLR y no el compilador, por lo que en realidad el CLR y el compilador colaboran para asegurar la corrección de las conversiones.

No se pueden usar variables no inicializadas. El compilador de a los campos un valor por defecto consistente en ponerlos a cero y controla mediante análisis del flujo de control del fuente que no se lea ninguna variable local sin que se le haya asignado previamente algún valor.

Se comprueba que todo acceso a los elementos de una tabla se realice con índices que se encuentren dentro del rango de la misma.

Se puede controlar la producción de desbordamientos en operaciones aritméticas, informándose de ello con una excepción cuando ocurra. Sin embargo, para conseguirse un mayor rendimiento en la aritmética estas comprobaciones no se hacen por defecto al operar con variables sino sólo con constantes (se pueden detectar en tiempo de compilación).

Incluye Delegados: A diferencia de Java, C# incluye delegados, que son similares a los punteros a funciones de C++ pero siguen un enfoque orientado a objetos, pueden almacenar referencias a varios métodos simultáneamente, y se comprueba que los métodos a los que apunten tengan parámetros y valor de retorno del tipo indicado al definirlos.

Pueden definirse métodos que admitan un número indefinido de parámetros de un cierto tipo, y a diferencia lenguajes como C/C++, en C# siempre se comprueba que los valores que se les pasen en cada llamada sean de los tipos apropiados.

Instrucciones seguras: Para evitar errores muy comunes, en C# se han impuesto una serie de restricciones en el uso de las instrucciones de control más comunes. Por ejemplo, la guarda de toda condición ha de ser una expresión condicional y no aritmética, con lo que se evitan errores por confusión del operador igualdad (==) con el de asignación (=); y todo caso de un switch ha de terminar en un break o goto que indique cual es la siguiente acción a realizar, lo que evita la ejecución accidental de casos y facilita su reordenación.

Sistema de tipos unificado: A diferencia de C++, en C# todos los tipos de datos que se definan siempre derivarán, aunque sea de manera implícita, de una clase base llamada System.Object, por lo que dispondrán de todos los miembros definidos en esta clase.

A diferencia de Java, en C# esto también es aplicable a los tipos de datos básicos.
Además, para conseguir que ello no tenga una repercusión negativa en su nivel de rendimiento, se ha incluido un mecanismo transparente de boxing y unboxing con el que se consigue que sólo sean tratados como objetos cuando la situación lo requiera, y mientras tanto puede aplicárseles optimizaciones específicas. El hecho de que todos los tipos del lenguaje deriven de una clase común facilita enormemente el diseño de colecciones genéricas que puedan almacenar objetos de cualquier tipo.

Extensibilidad de tipos básicos: C# permite definir, a través de estructuras, tipos de datos para los que se apliquen las mismas optimizaciones que para los tipos de datos básicos. Es decir, que se puedan almacenar directamente en la pila (luego su creación, destrucción y acceso será más rápido) y se asignen por valor y no por referencia. Para conseguir que lo último no tenga efectos negativos al pasar estructuras como parámetros de métodos, se da la posibilidad de pasar referencias a pila a través del modificador de parámetro ref.

Extensibilidad de operadores: Para facilitar la legibilidad del código y conseguir que los nuevos tipos de datos básicos que se definan a través de las estructuras estén al mismo nivel que los básicos predefinidos en el lenguaje, al igual que en C++ y a diferencia de Java, C#

permite redefinir el significado de la mayoría de los operadores (incluidos los de conversión, tanto para conversiones implícitas como explícitas) cuando se apliquen a diferentes tipos de objetos. Las redefiniciones de operadores se hacen de manera inteligente, de modo que a partir de una única definición de los operadores ++ y -- el compilador puede deducir automáticamente como ejecutarlos de manera prefijas y postfija; y definiendo operadores simples (como +), el compilador deduce como aplicar su versión de asignación compuesta (+=). Además, para asegurar la consistencia, el compilador vigila que los operadores con opuesto siempre se redefinan por parejas (por ejemplo, si se redefine ==, también hay que redefinir !=). También se da la posibilidad, a través del concepto de indizador, de redefinir el significado del operador [] para los tipos de dato redefinidos por el usuario, con lo que se consigue que se pueda acceder al mismo como si fuese una tabla. Esto es muy útil para trabajar con tipos que actúen como colecciones de objetos.

Extensibilidad de modificadores: C# ofrece, a través del concepto de atributos, la posibilidad de añadir a los metadatos del módulo resultante de la compilación de cualquier fuente de información adicional a la generada por el compilador que luego podrá ser consultada en tiempo de ejecución a través de la librería de reflexión de .NET. Esto, que más bien es una característica propia de la plataforma .NET y no de C#, puede usarse como un mecanismo para definir nuevos modificadores.

Versionable: C# incluye una política de versionado que permite crear nuevas versiones de tipos sin temor a que la introducción de nuevos miembros provoquen errores difíciles de detectar en tipos hijo previamente desarrollados y ya extendidos con miembros de igual nombre a los recién introducidos. Si una clase introduce un nuevo método cuyas redefiniciones deban seguir la regla de llamar a la versión de su padre en algún punto de su código, difícilmente seguirán esta regla miembros de su misma signatura definidos en clases hijas previamente a la definición del mismo en la clase padre; o si introduce un nuevo campo con el mismo nombre que algún método de una clase hija, la clase hija dejará de funcionar. Para evitar que esto ocurra, en C# se toman dos medidas: se obliga a que toda redefinición deba

incluir el modificador override, con lo que la versión de la clase hija nunca sería considerada como una redefinición de la versión miembro en la clase padre ya que no incluiría override.

Para evitar que por accidente un programador incluya este modificador, sólo se permite incluirlo en miembros que tengan la misma signatura que los miembros marcados como redefinibles mediante el modificador virtual. Así, además, se evita el error tan frecuente en Java de creerse haber redefinido un miembro, pues si el miembro con override no existe en la clase padre se producirá un error de compilación. Si no se considera redefinición, entonces se considera que lo que se desea es ocultar el método de la clase padre, de modo que para la clase hija sea como si nunca hubiere existido. El compilador avisará de esta decisión a través de un mensaje de aviso que puede suprimirse incluyendo el modificador new en la definición del miembro en la clase hija para así indicarle explícitamente la intención de ocultación.

Eficiente: En principio, en C# todo el código incluye numerosas restricciones para asegurar su seguridad y no permite el uso de punteros. Sin embargo, y a diferencia de Java, en C# es posible saltarse dichas restricciones manipulando objetos a través de punteros. Para ello basta marcar regiones de código como inseguras (modificador unsafe) y podrán usarse en ellas punteros de forma similar a cómo se hace en C++, lo que puede resultar vital para situaciones donde se necesite una eficiencia y velocidad de procesamiento muy grandes.

Compatible: Para facilitar la migración de programadores, C# no sólo mantiene una sintaxis muy similar a C, C++ o Java que permite incluir directamente código escrito en C# fragmento de código escritos es estos lenguajes, sino que el CLR también ofrece, a través de los llamados Platform Invocation Services (Pinvoke), la posibilidad de acceder a código nativo escrito como funciones sueltas no orientadas a objetos tales como las DLLs de la API Win32.

Nótese que la capacidad de usar punteros en código inseguro permite que se pueda acceder con facilidad a este tipo de funciones, ya que estas muchas veces esperan recibir o devuelven punteros. También es posible acceder desde código escrito en C# a objetos COM. Para facilitar

esto, el .NET Framework SDK incluye unas herramientas llamadas tlbimp y regasm mediante las que es posible generar automáticamente clases *proxy* que permitan, respectivamente, usar objetos COM desde .NET como si de objetos .NET se tratase y registrar objetos .NET para su uso desde COM. También es posible usar controles ActiveX desde código .NET y viceversa.

2.3 SQL Server Compact Edition

SQL Server Compact Edition, o más comúnmente conocido cómo SQL Server CE, es el motor de base de datos que proporciona Microsoft para sus Sistemas Operativos, Microsoft Windows CE .NET, Pocket PC 2002 con Windows o Pocket PC 2000. Su creación y funcionamiento está basado en el proveedor de datos de Compact Framework, el cual fue desarrollado con la tecnología .NET.

SQL Server CE amplía las funcionalidades de almacenamiento de datos de Microsoft SQL Server para los dispositivos portátiles, a la vez que proporciona un modelo de programación coherente para un rápido desarrollo de las aplicaciones.

SQL Server CE proporciona funcionalidades esenciales de las bases de datos relacionales de una manera sencilla.

2.3.1 Compact Framework

Es un entorno de propiedad de Microsoft. Los sistemas que se desarrollen con este framework necesitan tener instalado un sistema operativo Windows CE o Windows Mobile. El entorno de .NET Compact Framework utiliza una versión aproximada a la versión completa de Microsoft .NET, se puede implementar en C#, Visual Basic, Visual C, C++.

2.3.2 Emuladores

Es un software que se encarga de simular el comportamiento de los dispositivos móviles, especialmente en el desarrollo de aplicaciones, puesto que inicialmente es una garantía probar en este tipo de software para posteriormente implantarlo en el dispositivo físico.

Es comprobado que los recursos de memoria y procesamiento no pueden tener una repartición adecuada de los recursos hardware, es por ello que el funcionamiento de un programa es mejor en el emulador que en el dispositivo real.

2.3.3 Características de SQL CE

- Sincronización:
    - Soporte para múltiples suscripciones
    - Soporte multiusuario; acceso concurrente
    - Control del estado del proceso de sincronización
    - Seguimiento a nivel de columna
- Motor de ejecución:
    - Autoshrink
- Procesador de consultas:
    - Optimización basada en costes
- Máximo tamaño de BBDD 4 GB
- Integridad referencial completa con actualizaciones y eliminaciones en cascada
- Múltiples conexiones simultáneas
- Cifrado a nivel de fichero de 128 bits
- Soporte transaccional
- Plena coherencia transaccional, incluso con
- SQL Server Compact 3.5 soporta replicación de datos con SQL Server 2008 usando Microsoft Synchronization Services for ADO.NET*.

2.4 ANÁLISIS DE SISTEMAS EXISTENTES

2.4.1 INGENIERÍA INVERSA A UNA BASE DE DATOS

2.4.2 Introducción

La Ingeniería Inversa de Bases de Datos es el conjunto de técnicas que permite la obtención de una representación conceptual de un esquema de base de datos a partir de su codificación.
Sus aplicaciones son múltiples: Re-documentar, reconstruir y/o actualizar documentación perdida o inexistente de bases de datos, servir como pivote en un proceso de migración de datos, y ayudar en la exploración y extracción de datos en bases poco documentadas.
Ahora se comienza a realizar el análisis por el cual obtendremos el modelo conceptual de una base de datos a partir de un modelo físico. La ingeniería inversa de datos se aplica sobre algún código de bases de datos (aplicación, código SQL, etc.) para obtener los modelos relacionales o sobre el modelo relacional para obtener el diagrama entidad-relación. Hay que tener en cuenta que la ingeniería inversa se puede dar entre distintos productos del ciclo de vida de una aplicación.
Existen muchas técnicas para hacer ingeniería inversa de base de datos, algunos de los cuales se pueden ver resumidos:

- **Baltín et al (1992):**
    - **Suposiciones:** 3FN. Consistencia en el nombrado de los atributos. Sin homónimos. Clave principal y clave candidata.
    - **Entradas:** Dependencias. Esquemas de relación.
    - **Salidas:** Entidades. Relaciones binarias. Categorías. Generalizaciones.
    - Basado en el método de Navathe de Awongs.

- **Chiang et al (1994,1996):**
  - **Suposiciones:** 3FN[7]. Consistencia en el nombrado de los atributos. Sin errores en los atributos claves.
  - **Entradas:** Esquema de relación. Instancia de datos. Dependencias.
  - **Salidas:** Entidades. Relaciones binarias. Generalizaciones. Agregaciones.
  - **Características:** Requiere el conocimiento acerca del nombre de los atributos. Propone un marco de trabajo para la evaluación de los métodos de ingeniería inversa. Identifica claramente los casos en los que las entradas de los humanos son necesarias.

- **Davids & Arora (1987):**
  - **Suposiciones:** 3FN. Sin homónimos ni sinónimos.
  - **Entradas:** Esquemas de relación. Restricciones de claves ajenas.
  - **Salidas:** Conjunto de entidades. Relaciones binarias. Relaciones n-aria.
  - **Características:** Apunta a una transformación reversible desde el esquema relacional al esquema conceptual.

- **Johannessin (1994):**
  - **Suposiciones:** 3FN. Consultas de dominio independientes.
  - **Entradas:** Dependencias funcionales y de inclusión. Esquemas de relación.
  - **Salidas:** Generalizaciones. Entidades. Relaciones binarias.
  - **Características:** Basado en los conceptos establecidos de la teoría de bases de datos relacionales. Proceso de mapeo simple y automático.

- **Markowitz & Makowsky (1990):**
  - **Suposiciones:** FN Boycce-Codd.
  - **Entradas:** Esquemas de relación. Dependencias de claves. Dependencias de integridad referencial.

[7] Tercera Forma Normal

- **Salidas:** Entidades. Relaciones binarias. Generalizaciones y agregaciones.
- **Características:** Requiere todas las dependencias de clave.

- **Navathe & Among (1987):**
  - **Suposiciones:** 3FN y algunos 2FN. Consistencia en el nombrado de atributos. Sin ambigüedades de clave ajena. Claves candidatas específicas.
  - **Entradas:** Esquemas de relación.
  - **Salidas:** Entidades. Relaciones binarias. Categorías. Cardinalidades.
  - **Características:** Es vulnerable a la ambigüedad en el reconocimiento de claves ajenas.

- **Petit et al (1996):**
  - **Suposiciones:** 1FN. Atributos únicos.
  - **Entradas:** Esquemas de relación. Instancia de datos y código.
  - **Salidas:** Entidades. Relaciones. Generalizaciones.
  - **Características:** Analiza las consultas en los programas de aplicación. No pone restricciones en el nombre de los atributos.

- **Permerlani & Blaha (1993,1994):**
  - **Suposiciones:** Sin F3N. Comprensión semántica de aplicación.
  - **Entradas:** Esquemas de relación. Observaciones de patrones de datos.
  - **Salidas:** Clases. Asociaciones. Generalizaciones. Multiplicidad. Agregación.
  - **Características:** Requiere un alto nivel de entrada de los humanos. Enfatiza en el análisis de las claves.

- **Sotou (1997,1998):**
  - **Suposiciones:** Sin nombres únicos de atributos. Dependencias desconocidas.

- **Entradas:** Esquema de datos. Instancia de datos. Diccionario de datos.
- **Salidas:** Cardinalidad de las restricciones de relaciones n-arias.
- **Características:** Proceso automatizado completamente para SQL.

Dentro de la metodología que se propone, para la realización de una ingeniería inversa, la podemos dividir en dos actividades que se denominan Análisis de datos y Abstracción Conceptual (Hausi et al., 2000).

La primera fase de análisis de datos, pretende recuperar un esquema de datos lógico completo, que esté lo suficientemente bien documentado y con su semántica correctamente identificada.

El esquema lógico que se obtiene en esta fase será la entrada de la siguiente fase, que como se ha comentado anteriormente, se denomina Abstracción conceptual, la cual transformará el esquema lógico a un esquema conceptual equivalente basado en modelo entidad-relación, el cual proporcionará una rica base en la fase de reingeniería de la base de datos (RBD) (etapa posterior a la ingeniería inversa).

Tanto en la etapa de ingeniería inversa como en la etapa de reingeniería, los ingenieros o diseñadores van a contar con una importante ayuda que es la que proporcionará el trabajo cooperativo de los usuarios que, por medio de unas herramientas sencillas, conseguirán acumular más información tanto de la LDB (Base de Datos Legada) como de la NDB (Base de Datos Nueva). El trabajo de los usuarios se basará en una utilización de un trabajo automatizado, interactivo e iterativo para de esa forma conseguir un modelo lo más correcto y eficiente posible, de tal forma que no solo se obtendrá un nuevo modelo sino que sería muy eficiente obtener un modelo que permitiera de forma automatizada realizar una migración de los datos.

Es cierto que muchos de los problemas que se expondrá, son típicos de muchas organizaciones y sistemas de información, pero ciertamente en la gestión de entidades municipales son problemas de envergadura, que hacen que un proceso de ingeniería inversa y/o reingeniería sea una tarea indicada de forma bastante significativa en este tipo de organizaciones. Podemos considerar como problemas comunes:

- Estructuras implícitas: algunas estructuras no fueron declaradas en el momento de su diseño de forma explícita en la especificación DDL (Lenguaje de Definición de Datos) de la base de datos, sino que para completar la semántica de la base de datos se han utilizado *triggers, checks, procedimientos,* etc., por lo que para encontrar características como la integridad no basta con observar las definiciones de las tablas (DDL).
- Diseño pobre o construcciones poco legibles: no todas las bases de datos son perfectas aunque funcionen correctamente, a veces podemos encontrar estructuras pobres o incluso erróneas, redundantes o simplemente poco legibles al tener nombres los campos poco clarificadores, etc.
- Poca documentación: se realizó poca o no existe documentación sobre el diseño.
- Migrar los datos desde modelos de datos antiguos al modelo relacional.
- Cambiar sistema de gestión de base de datos.
- Detección de errores de integridad.
- Migración de modelos de datos relacionales a modelo orientado a objetos.

2.4.3 Ingeniería Inversa de Base de Datos

Como se ha comentado anteriormente el proceso de Ingeniería Inversa sobre las bases de datos, es comúnmente dividido en dos fases claramente diferenciadas:

- Extracción de las estructuras de datos, obteniendo como resultado el esquema lógico. (Fase I)

- Conceptualización de las estructuras de datos, obteniendo como resultado el esquema conceptual. (Fase II)

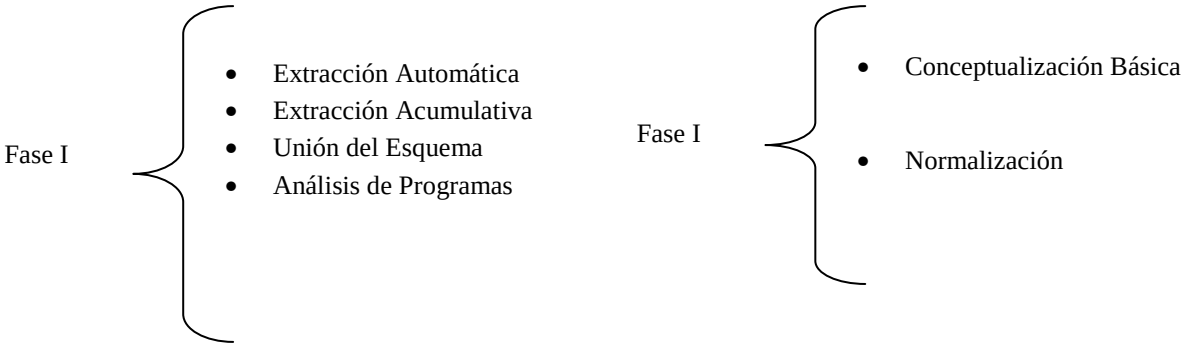


Figura 9. Esquema del proceso de ingeniería inversa en bases de datos

2.4.3.1 Fase I.

En esta fase se va a realizar una extracción de las estructuras existentes actualmente en el sistema de información, dividiéndose para este fin en dos etapas para cumplir con la extracción de información.

2.4.3.1.1 Etapa 1: Extracción Automática.

Inicialmente se va a extraer mediante herramientas automáticas todas las estructuras de la base de datos cómo fueron diseñadas inicialmente por los desarrolladores. Se trata por tanto de una etapa típica de traducción inmediata del código para así extraer las estructuras de datos explícitas.

2.4.3.1.2 Etapa 2: Extracción Acumulativa.

Se trata de una etapa en la que la participación de los usuarios del modelo de datos con el que trabaja supondrá acumular más información de la obtenida en la etapa anterior. Como se ha

comentado anteriormente es posible que ciertas reglas no puedan obtenerse directamente en la etapa 1, por lo que aprovechando el conocimiento adquirido por los usuarios en el trabajo diario se podrá obtener información muy interesante.

Así, aspectos importantes, sobre los que los usuarios nos pueden ayudar son:

a) Análisis de Nombres: El usuario hará una descripción de aquellos campos en los que es posible que tengamos dudas acerca de su rol, tipos de datos, relación, etc, para de esta forma despejar incertidumbres dentro de las mismas.

b) Extracción de claves externas: Se sabe que en la etapa 1, de forma sencilla se pueden obtener las claves principales, pero la obtención de claves externas a veces no es tarea sencilla, y la información aportada por el usuario es vital.

2.4.3.1.3 Etapa 3: Unión del Esquema:

La unión del esquema, se trata de una etapa ciertamente compleja, y de la que su éxito depende en gran medida el proceso de reingeniería, pues consiste en unir y reconvertir las estructuras y restricciones obtenidas en las dos fases anteriores, a este proceso. Así, con la información obtenida en la etapa 2 se pretende complementar la información obtenida en la etapa 1, ya sea para encontrar estructuras no explícitas o que fueron perdidas cuando la base de datos fue diseñada.

Para ello se localizarán:

a) Campos multivaluados.
b) Tipos registros e identificadores de campos multivaluados.
c) Campos opcionales.
d) Claves.

e) Redundancias.

f) Dominios.

g) Significado de los campos.

En esta misma etapa y dado que no se les puede dar a los usuarios la responsabilidad de cubrir las posibles lagunas existentes por un mal diseño ya que este no podrá cubrir con todas las espectativas, es conveniente realizar una etapa posterior para comprobar la corrección de lo realizado en esta fase.

2.4.3.1.4 **Etapa 4: Análisis de Programas**.

En esta paso se va a realizar un estudio del código fuente existente, para comprobar que las restricciones, forma de procesar los datos, significado, etc. se corresponde con el estudio realizado en las fases anteriores. Es posible que esta etapa modifique el resultado obtenido en las etapas anteriores, pero sería deseable que no existieran cambios, pues de esa forma podríamos asegurarnos que lo hecho en las etapas anteriores es correcto y vamos por un buen camino.

En esta misma etapa podríamos incluir el análisis de formularios, consultas, informes, etc. que puedan existir, ya que de esa forma podemos ver si los resultados obtenido con la LBD coinciden con el modelo extraído.

Se Quiere en este punto describir que la ejecución de consultas para analizar los datos, puede permitirnos obtener claves externas. Así, si en una base de datos con cierto rodaje de trabajo, habitualmente se ejecuta una consulta en una tabla B para encontrar en algún campo el valor de un campo clave de una tabla A, y el resultado obtenido es cero, nos indicaría que posiblemente no es una clave externa.

2.4.3.2 Fase II.

En esta fase se extraerá el esquema conceptual a partir del esquema lógico. En mucha bibliografía, a esta fase se le denomina Interpretación de las estructuras de datos, pues se va a realizar una optimización del esquema lógico.

2.4.3.2.1 Etapa 1: Conceptualización básica.

Así, un ejemplo de conceptualización básica podría ser si tenemos campos que tienen la misma estructura y que se refieren a atributos de la entidad iguales, se transformarán en un atributo multivaluado.

2.4.3.2.2 Etapa 2: Normalización.

La reforma del esquema conceptual tiene como objeto hacer una comprensión de dicho modelo. En esta etapa se pretende aportar un significado en la semántica de las construcciones explícitas. El objetivo es representar la información con una metodología estándar que sea altamente legible.
El modelo entidad relación cumple con amplitud el objetivo planteado por lo que existen muchos modelos aplicables a bases de datos relacionales.

Se trata de una etapa relativamente fácil respecto a la recuperación de las estructuras estáticas, pero lo realmente complejo y difícil de abordar de una forma totalmente precisa son la recuperación de las reglas de integridad. Es en este apartado como se ha comentado anteriormente es donde se conjugará la utilización de código fuente, para ver estructuras condicionales respectos a datos del modelo, y por supuesto del usuario el cual puede aportarnos información muy sensible e importante.

Para la obtención de las estructuras podemos se seguirán los pasos descritos por Premerlani (Premerlani, 1994), que considera que cada fichero es una posible tabla y cada campo del fichero será un campo de la tabla.

a) Identificar un conjunto de campos que puedan ser la clave principal para sus respetivos ficheros. En esta fase puede ser de utilidad la participación del usuario, aunque es cierto que pueden existir ficheros que no pertenezcan al modelo y que se utilicen simplemente como almacén de datos temporales, etc.

b) Determinar las llaves externas, que serán habitualmente del mismo tipo que sus correspondientes en la tabla principal. En esta fase es muy interesante apoyarnos en el estudio del código fuente y en las instrucciones donde se asigne a un campo el contenido del campo de otra tabla, o que se comparen campos de distintas tablas, etc. Es una forma de encontrar posibles relaciones y asociaciones entre las tablas.

c) Determinar qué ficheros encontrados pueden dejar de considerarse como tablas. Por ejemplo, aquellos que no se haya podido determinar campo clave.
d) Buscar y encontrar generalizaciones, que puede hacerse mediante la localización de grandes grupos de claves ajenas. Así podemos decir que una clave formada únicamente por claves ajenas suele indicar una relación de esta tabla o entidad con n entidades, formando por tanto una relación de grado n+1.

e) Encontrar asociaciones mediante la repetición de campos en una tabla. Si en un determinado campo se encuentra repetido un dato que aparece en otra tabla, podemos indicar que existe una relación entre ellas.

En todas estas fases es fundamental la ejecución de consultas para comparar los resultados con los esperados, o incluso con datos que se hayan obtenido en ejecuciones del sistema de información anterior.

En este punto, alguien nos podría preguntar qué grado de participación tiene el usuario. Evidentemente tiene una gran participación, que es la de determinar si un objeto externo (Leiva, 2003; Leiva, Guevara 1998) es almacenable o no. Estas decisiones no son siempre certeras, pues como se ha comentado en más de una ocasión el usuario puede cometer errores, pero realmente es una información muy valiosa la que nos proporciona.

Una vez que se ha detectado las tablas y su estructura, podemos utilizar la siguiente notación por ejemplo: RESERVA[Número, FechaReserva, CodCliente, NúmFactura] donde en mayúsculas aparece el nombre de la tabla, y entre corchetes los campos de dicha tabla.

2.5 INTEGRACION DE SISTEMAS

2.5.1 SISTEMAS DE TRABAJO EN GRUPO

El trabajo en grupo es una tarea complicada y susceptible a muchos fallos, existen gran cantidad de aplicaciones diseñadas para facilitar esta tarea, como pueden ser gestores de documentación, sistemas de control de versionado, calendarios y contactos compartidos, etc.
Todos los proyectos generan gran cantidad de documentación, normalmente contienen, en su mayoría, documentos técnicos que son modificados a lo largo del tiempo por varias personas. El tiempo que se pierde en encontrar el documento actualizado o las pérdidas de información cuando varias personas acceden a la vez a un mismo documento son inaceptables.
Existen sistemas de documentación que evitan dichas pérdidas además de aportar información adicional sobre el documento (porqué se modificó, quien lo hizo, etc) y almacenar versiones del mismo.

A lo largo de la vida de una empresa se genera una gran cantidad de documentos que es necesario almacenar y clasificar. La documentación de una empresa no puede depender de la fiabilidad de cada uno de los ordenadores personales donde se almacene. Es indispensable

centralizar todos los documentos para realizar copias de seguridad periódicas y garantizar su permanencia en el tiempo.

Es frecuente que el conocimiento resida en los trabajadores y no en la empresa. Esta dependencia puede generar problemas de diversa índole: retrasos, pérdidas de proyectos, recursos parados, etc. Es imprescindible documentar el Know How de las distintas áreas para romper esta dependencia, así como permitir el acceso sólo a los usuarios o grupos apropiados.

2.5.1.1 Servicios

La plataforma de trabajo en grupo ofrece un sistema unificado para la gestión y el acceso a sus datos. Todos los servicios que se describen a continuación están totalmente integrados, de tal forma que los usuarios usarán un único identificador de acceso y la misma contraseña para acceder a todos los servicios. Cuando el usuario cambie su contraseña, sólo lo tendrá que realizar desde una de las aplicaciones del sistema, y esa nueva contraseña se usará para acceder al resto de los servicios sin necesidad de realizar ninguna tarea adicional.

Servidor LDAP, donde se almacena toda la información personal de usuarios, configuraciones de correo, libretas de direcciones personales y compartidas, entradas DNS y DHCP, etc. Permite gestionar de manera centralizada todos los datos referentes a usuarios y sistemas, y evitar así la duplicidad de los mismos en distintos servicios o la necesidad de sincronización. Al estar basado en el protocolo estándar LDAP es totalmente compatible con aplicaciones de terceros.

Servidor Samba. Permite la compartición de archivos e impresoras en red, asignando permisos sobre el acceso y la utilización de los mismos. También actúa como controlador de dominio, simulando un servidor Windows 2000/2003. Este servicio incluye también un servicio antivirus que analiza en tiempo real el acceso a los archivos.

Servidor DNS y DHCP. Ofrece servicios de asignación dinámica de direcciones y resolución DNS a los equipos de la red. La información se almacena dinámicamente y en tiempo real en el servidor LDAP.

Servidor de correo. Incluye los servicios SMTP, POP3 e IMAP así como un cliente web para trabajo en grupo, que contiene módulos de correo web, agenda, libreta de direcciones, tareas, etc. El sistema integra una solución antispam y antivirus, por lo que todos los mensajes de correo que entran y salen son analizados y bloqueados en caso de incluir contenido malicioso, así como una interfaz de consulta de los mensajes bloqueados.

Entorno web para acceso al sistema de colaboración. Una completa interfaz web permite el acceso universal de los usuarios a su correo, tareas, calendario, notas y libreta de direcciones. Todos estos datos pueden ser compartidos entre o usuarios o grupos de usuarios. Además, existen conectores que permiten acceder a estos datos en tiempo real, sin necesidad de sincronización, desde programas externos como Microsoft Outlook, Mozilla Thunderbird/Sunbird, o Gnome-Novell Evolution.

Mensajería instantánea. Permite a los usuarios del sistema comunicarse de forma sencilla e instantánea a través de mensajes de texto de modo similar a MSN Live Messenger, Yahoo Messenger o Google Talk. La ventaja de este sistema frente a los comentados, es que éste es un sistema interno, por lo que los usuarios sólo podrán comunicarse con el resto de usuarios de la empresa, y no por Internet.

También existe la posibilidad de la interconexión del sistema de mensajería de la organización con los sistemas de mensajería externos descritos anteriormente; esta interconexión puede ser usada de manera general, o solo por ciertos usuarios o grupos de la organización.

Acceso FTP. Permite a los usuarios acceder a sus documentos a través del protocolo FTP cuando estén fuera de la oficina.

2.5.2 Gestion de Sistemas

La Gestión de Sistemas se ocupa de integrar, planificar y controlar los aspectos técnicos, humanos, organizativos, comerciales y sociales del proceso completo (desde el análisis y el diseño hasta la vida operativa del sistema). Los objetivos principales de la Gestión de Sistemas suelen ser:

1. Planificar y controlar el proceso completo de análisis, diseño y operación del sistema dentro del presupuesto, plazo, calidad y restantes condiciones convenidas
2. Controlar la validez de los criterios de diseño
3. Controlar la adecuación del producto del diseño a los requisitos establecidos en el análisis.
4. Planificar y desarrollar las necesidades de mantenimiento
5. Planificar y desarrollar las necesidades de formación del personal que va a operar el sistema.
6. Planificar la supervisión del funcionamiento del sistema

En grandes proyectos de ingeniería, y dentro del ámbito de la gestión, el ingeniero de sistemas suele funcionar como asesor del director del proyecto, obteniendo, elaborando y presentando informaciones en un formato adecuado para que éste pueda tomar las decisiones pertinentes.

2.5.3 GESTION DE BASE DE DATOS

El propósito general de los sistemas de gestión de base de datos es el de manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante, para un buen manejo de datos. Dentro de este aspecto se analizará todo lo concerniente a las operaciones de Administración a ejecutarse sobre la(s) Bases de Datos que se utilizaron en el desarrollo de esta Tesis, tanto dbAme propia de las Entidades Municipales alojada en un servidor, así como db_sisnycs que reside en la aplicación móvil. En

dicha administración se busca explicar la manera en que se manipularán los datos sobre cada una de las Bases de datos antes mencionadas.

2.5.3.1 Modelo de Arquitectura de la Base de Datos

Las bases de datos respetan la arquitectura de tres niveles definida, para cualquier tipo de base de datos:

- Nivel Interno
- Nivel Conceptual
- Nivel Externo

- **Nivel interno:** es el nivel más bajo de abstracción, y define cómo se almacenan los datos en el soporte físico, así como los métodos de acceso.

- **Nivel conceptual:** es el nivel medio de abstracción. Se trata de la representación de los datos realizada por la organización, que recoge las vistas parciales de los requerimientos de los diferentes usuarios y las aplicaciones posibles. Se configura como visión organizativa total, e incluye la definición de datos y las relaciones entre ellos.

- **Nivel externo:** es el nivel de mayor abstracción. A este nivel corresponden las diferentes vistas parciales que tienen de la base de datos los diferentes usuarios. En cierto modo, es la parte del modelo conceptual a la que tienen acceso.

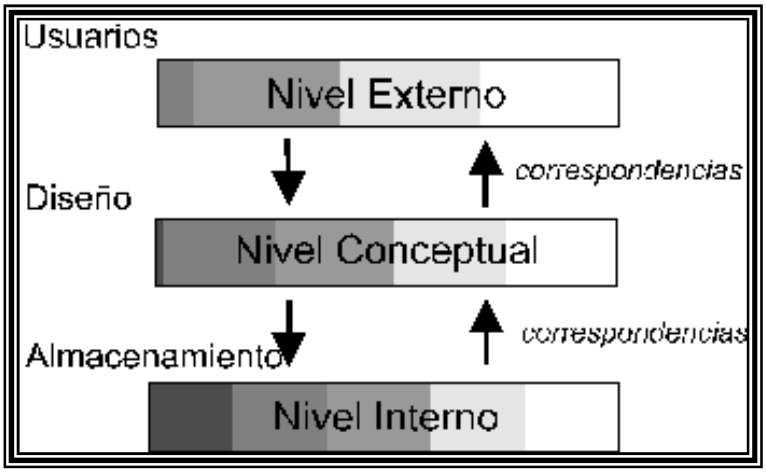


Figura 10. Modelo de la Arquitectura de las Bases de Datos.

En ocasiones puede encontrarse el nivel conceptual divido en dos niveles, conceptual y lógico. El primero de ellos corresponde a la visión del sistema global desde un punto de vista organizativo independiente, no informático. El segundo correspondería a la visión de la base de datos expresada en términos del sistema que se va a implantar con medios informáticos.

2.5.3.2 Administración de Base de Datos

El administrador de base de datos (DBA) es la persona responsable de los aspectos ambientales de una base de datos incluye:

- **Recuperabilidad** - Crear y probar Respaldos
- **Integridad** - Verificar o ayudar a la verificación en la integridad de datos
- **Seguridad** - Definir y/o implementar controles de acceso a los datos
- **Disponibilidad** - Asegurarse del mayor tiempo de encendido
- **Desempeño** - Asegurarse del máximo desempeño incluso con las limitaciones
- **Desarrollo y soporte a pruebas** - Ayudar a los programadores e ingenieros a utilizar eficientemente la base de datos.

Acciones a ejecutar periódicamente sobre las Bases de Datos:

- **Publicación -** Es una colección uno o más artículos, y un artículo es una agrupación de datos para replicarse. Un artículo puede ser una tabla entera, o sólo ciertas columnas (que usan un filtro vertical), o solamente ciertas filas (que usan un filtro horizontal), o incluso un procedimiento almacenado (en algunos tipos de réplica).

    - **Publicador -** El Publicador es el servidor que hace los datos disponibles para la réplica hacia otros servidores. Además de identificar los datos para su replicación, el publicar detecta los datos que han cambiado y mantiene la información sobre todas las publicaciones en el servidor. Cualquier elemento de datos se replica en un solo publicador, incluso si los datos se ponen al día por cualquier número de suscriptores o publicar de nuevo por otro.

- **Suscripción por Tirón**. Es en la cual el suscriptor pide las actualizaciones periódicas de los cambios en el editor. Las suscripciones por tirón son las mejores para las publicaciones que tienen una gran cantidad de suscriptores (por ejemplo, suscriptores que usen el Internet). Las suscripciones por tirón son también las mejores para los usuarios móviles autónomos porque se determinan cuando se sincronizan los cambios de los datos.

  Una sola publicación puede apoyar una mezcla de suscripciones por tirón o por empuje.

- **Suscripciones por Empuje**. Es en la cual el publicador propaga los cambios al suscriptor sin una petición específica del suscriptor. Las suscripciones por empuje se utilizan para propagar cambios cada que éstos ocurren, o para de acuerdo al horario fijado por el publicador.

- **Suscriptor.** Los suscriptores son los servidores que almacenan las reproducciones y reciben actualizaciones. Dependiendo de la versión de SQL, se permite a los suscriptores hagan actualizaciones a los datos, aunque un suscriptor que hace actualizaciones no es igual que un publicador. Un suscriptor puede, alternadamente, también publicar hacia otros suscriptores.

- **Replicación por Mezcla -** Proporciona el nivel más alto de la autonomía para cualquier solución de réplica. Los publicadores y los suscriptores pueden trabajar independientemente y volverse a conectar periódicamente para combinar sus resultados. Si un conflicto es creado por los cambios que son hechos al mismo elemento de datos en los sitios múltiples, esos cambios serán resueltos automáticamente.

  Estas características hacen de la Replicación por Mezcla la solución ideal para usos tales como automatización de fuerzas de ventas, en la cual los usuarios necesitan el acceso de lectura/grabación por completo en las reproducciones locales de datos en ambiente altamente desconectado. Cuando ocurren conflictos, la réplica de la mezcla proporciona su resolución automática.

2.5.4 Homogeneización de Sistemas

Hablamos de Homogeneización cuando logramos que dispositivos a nivel hardware y software logren mantener una comunicación punto a punto, sin importar la tecnología con la que fueron fabricados, versionamiento de software, tipo de plataforma de desarrollo, para el caso específico de este estudio podemos resumir que la homogenización en este proyecto se da al lograr establecer un diálogo entre el equipo Servidor y el dispositivo Móvil, a continuación detallaremos las características de cada uno de los equipos mencionados:

SERVIDOR:

Software	Descripción
Windows 2003 Server	
Sistema GCS (Gestión de Comercialización de Servicios)	.NET 2003 Framework 1.1
Base de Datos SQL Server 2000	

Tabla 1. Características del Servidor con el Sistema GCS

DISPOSITIVO MOVIL:

Software	Descripción
Windows Mobile / Windows CE	
SISNYCS (Sistema de Notificación Y Cobro de Servicios)	.NET 2005 Framework 2.0
Base de Datos SQL Server CE (Compact Edition)	

Tabla 2. Características del Dispositivo Móvil, para el Módulo SISNYCS

2.5.5 Análisis de Sistemas de Seguridad Informática

Por la confidencialidad y seguridad de la Información que maneja cada una de las instituciones Municipales, es necesario tener un plan de contingencia en el que se pueda detallar la manera en que se administrará el acceso a dicha información, para ello se ha visto la necesidad de analizar el manejo de contraseñas y usuarios del sistema operativo y de los sistemas Informáticos con los que se desempeña el trabajo en dichas Instituciones.

Planteamos dentro del estudio la utilización de:

- **Usuarios del Sistema y del Dominio.**

  Dichos usuarios serán administrados en su totalidad por el personal encargado del mismo en cada Institución, para ello hace uso del Servidor, en el que claramente se

identifica la creación, eliminación, modificación de las Cuentas de Usuario, empleando diversas técnicas o procedimientos, en su mayoría el uso de directorios LDAP.

Con respecto a la aplicación Movil consideramos no ser necesaria la creación de un usuario de la aplicación puesto que dificultaría el trabajo de recolección del personal encargado del mismo, todo esto basado en los requerimientos proporcionados por el personal de la Institución.

- **Usuarios de las Bases de Datos**

  Podemos definir a los usuarios como toda persona que tenga todo tipo de contacto con el sistema de base de datos desde que este se diseña, elabora, termina y se usa.

  En el caso de nuestro estudio pudimos identificar un usuario dentro de la BD db_Ame, propia del Servidor y a la que tiene acceso la aplicación de Escritorio llamada GCS (Gestión de Cobro de Servicios), el mismo que se llama **Cosem** y tiene el total acceso a la Base de Datos del sistema en mención.

  Con respecto a la Base de Datos del dispositivo Móvil, no es necesario implantarlo además mediante pruebas en el desarrollo del aplicativo se pudo constatar que el trabajo y funcionamiento de la aplicación decaía en un alto índice.

CAPITULO III

3 MARCO PROPOSITIVO

En el propósito de brindar un servicio optimizado y de facilitar la labor de quienes lo realizan, nos ha llevado a plantear este proyecto de Tesis, que resume la gran posibilidad de contar con un sistema automatizado para el cálculo de varios de los servicios que prestan las Municipalidades a los habitantes de un determinado cantón.

Durante el transcurso de los años han venido mejorando paulatinamente la manera de calcular y recolectar dichos servicios, pero encontrándolos hasta la actualidad con la ejecución de en su mayoría cálculos manuales, es por ello que se ha planteado la realización de un convenio entre la ESPOCH y el AME, dos instituciones de gran respeto y connotación en el País, mismo que se aprobó, quedando mutuamente beneficiadas ambas partes, la una en facilitar el personal capacitado para el desarrollo del proyecto con la oportunidad de terminar su carrera, y la otra para facilitar la labor de las Municipalidades que tiene a su cargo.

Previo al desarrollo del producto, se realizó un análisis en el que se determinó las necesidades de los involucrados, aplicando técnicas para la recolección de los datos, dichos datos fueron estudiados a fondo, para la presentación y planteamiento de una solución que ahora la ponemos a vista de quienes interesamos en el resultado.

3.1 Determinación de Tecnologías de Desarrollo para Plataformas Móviles

Las Plataformas de Desarrollo son herramientas software, utilizadas con la finalidad de crear nuevos paquetes software. Estas herramientas facilitan al programador toda una estructura, que normalmente se compone de un editor en el que se digitará todo el código de un nuevo software, un compilador que es el responsable de transformar todo el código digitado en otro código entendible para el computador, permitiendo así la ejecución del software.

Previo al Desarrollo se realizó un análisis para determinar las posibles plataformas y herramientas que se ajusten al modelo lógico y conceptual de la propuesta de solución planteada, teniendo en cuenta para el estudio los siguientes elementos, por ser los más usados y populares en lo que se refiere al desarrollo de sistemas móviles, y de acuerdo a las habilidades y conocimientos adquiridos por parte de los desarrolladores.

En base a estudios realizados por la empresa **TIOBE** Software, que realizó un estudio sobre cuáles son los lenguajes de programación más populares, y para eso se basó en el análisis en la demanda laboral, los cursos que se imparten de esos lenguajes, y además basándonos en análisis de foros en la página web Todo Sobre Pocket Pc al referirse a este tema, y además basándonos en el conocimiento adquirido respecto a programación en dispositivos móviles, se ha llegado a consensuar con la siguiente tabla.

Para ello se han tomado en cuenta parámetros como, la Facilidad de uso, la Popularidad que tiene el Lenguaje en el Mercado, orientado al desarrollo del sistema objeto de estudio del presente proyecto, y el dominio y conocimiento de dichos lenguajes por parte de los desarrolladores del presente proyecto.

Lenguajes de Programación	Facilidad	Popularidad	Dominio	Total
Visual Basic .Net	7	8	6	21
C# .Net	9	9	9	27
Pocket Studio	8	6	7	21
Super Waba	7	8	6	21
Java	7	9	6	22
Python	6	7	6	19

Tabla 3. Lenguajes de Programación más Populares (TOBIE Software)

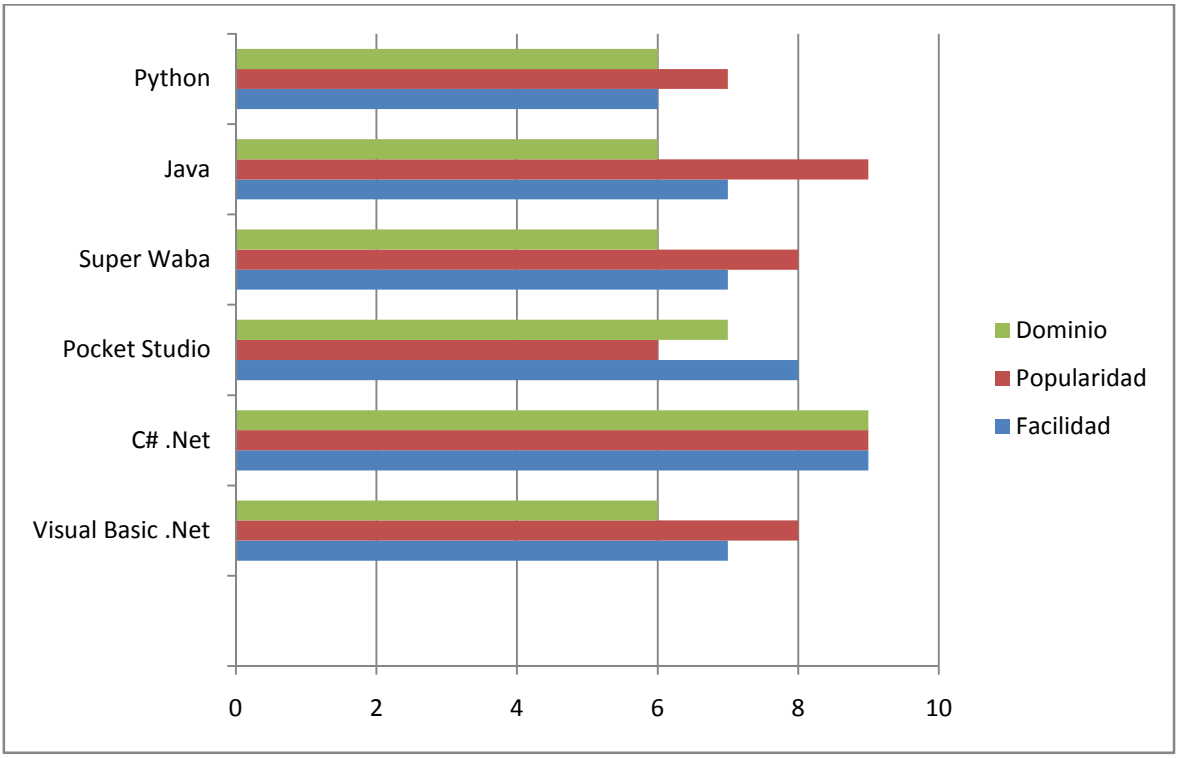


Figura 11. Análisis de Lenguajes de Programación para Dispositivos Móviles

Nota: Esta tabla no indica cual es el mejor lenguaje ni cual tiene más líneas de código escritas. Simplemente nos muestra tendencias de uso del mercado actual.

En base a este estudio se han tomado en cuenta como objeto de análisis los siguientes lenguajes de programación para Dispositivos Móviles,

3.1.1 C# .NET

C# es el nuevo lenguaje de propósito general orientado a objetos creado por Microsoft para su nueva plataforma .NET.
Microsoft.NET es el conjunto de nuevas tecnologías en las que Microsoft ha estado trabajando estos últimos años con el objetivo de mejorar tanto su sistema operativo como su modelo de componentes (COM) para obtener una plataforma con la que sea sencillo el desarrollo de software en forma de servicios web.

Los servicios web son un novedoso tipo de componentes software que se caracterizan a la hora de trabajar por su total independencia respecto a su ubicación física real, la plataforma sobre la que corre, el lenguaje de programación con el que hayan sido desarrollados o el modelo de componentes utilizado para ello.
El acceso a estos servicios se realiza en base a estándares de Internet, como son diferentes mecanismos del protocolo HTTP (GET y PUT) o el novedoso protocolo RPC conocido como SOAP (Simple Access Object Protocol), que no es más que una combinación de estándares como HTTP y XML para realizar llamadas a los miembros de estos servicios web. La idea detrás de SOAP consiste sencillamente en utilizar HTTP como medio de transporte para el envío de los mensajes de solicitud de ejecución de los miembros de servicios web remotos (lo que permite atravesar barreras tales como firewalls) y utilizar XML como lenguaje con el que escribir los cuerpos de estos mensajes.

Pero la plataforma .NET no son sólo los servicios web, sino que también ofrece numerosos servicios a las aplicaciones que para ella se escriban, como son un recolección de basura, independencia de la plataforma, total integración entre lenguajes (por ejemplo, es posible escribir una clase en C# que derive de otra escrita en Visual Basic.NET que a su vez derive de otra escrita en Cobol)

Como se deduce del párrafo anterior, es posible programar la plataforma .NET en prácticamente cualquier lenguaje, pero Microsoft ha decidido sacar uno nuevo porque ha visto conveniente poder disponer de un lenguaje diseñado desde 0 con vistas a ser utilizado en .NET, un lenguaje que no cuente con elementos heredados de versiones anteriores e innecesarios en esta plataforma y que por tanto sea lo más sencillo posible para programarla aprovechando toda su potencia y versatilidad.

C# combina los mejores elementos de múltiples lenguajes de amplia difusión como C++, Java, Visual Basic o Delphi. De hecho, su creador Anders Heljsberg fue también el creador de muchos otros lenguajes y entornos como Turbo Pascal, Delphi o Visual J++. La idea principal detrás del lenguaje es combinar la potencia de lenguajes como C++ con la sencillez de lenguajes como Visual Basic, y que además la migración a este lenguaje por los porgramadores de C/C++/Java sea lo más inmediata posible.

Además de C#, Microsoft propociona Visual Studio.NET, la nueva versión de su entorno de desarrollo adaptada a la plataforma .NET y que ofrece una interfaz común para trabajar de manera cómoda y visual con cualquiera de los lenguajes de la plataforma .NET (por defecto, C++, C#, Visual Basic.NET y JScript.NET, aunque pueden añadirse nuevos lenguajes mediante los plugins que proporcionen sus fabricantes).

3.1.1.1 HISTORIA

Este Lenguaje, desarrollado por Microsoft por un equipo dirigido por Andres Helsberj y Scott Wiltamuth, se diseño en específico para la plataforma .Net como un lenguaje que permitiera a los programadores migrar con facilidad hacia .Net. Tiene sus raíces en Java, C y C++; adapta las mejores características de cada uno de estos lenguajes y agrega características propias. C# está orientado a objetos y contiene una poderosa bibiloteca de clases (FCL por sus siglas en Ingles; FrameWork Class Library), mejor conocida como Biblioteca de Clases de Framework, que consta de componentes preconstruidos que permiten a los programadores desarrollar aplicaciones con rapidez, además este lenguaje es apropiado para desarrollar

apliaciones de escritorio (Windows Forms), asi como Smart Clients, Aplicaciones Web (ASP .Net), Aplicaciones Móviles, entre otras.

C# es un lenguaje de programación visual controlado por eventos, en el cual se crean programas mediante el uso de un Entorno de Desarrollo Integrado (IDE Por sus siglas en Inglés; Integrated Development Environment). Con un IDE un programador puede crear, ejecutar, probar y depurar programas en C#, con lo cual se reduce el tiempo requerido para producir un programa funcional en una fracción del tiempo que llevaría sin utilizar el IDE. La plataforma .Net permite la interoperabilidad de los lenguajes: los componentes de software de distintos lenguajes pueden interactuar como nunca antes se habia hecho. Los desarrolladores pueden empaquetar incluso hasta el software antiguo para que trabaje con nuevos programas en C#. Además, las aplicaciones en C# pueden interactuar a través de Internet mediante el uso de estándares industriales de comunicación como XML (eXtensible Markup Language) o el SOAP (Simple Object Access Protocol).

El lenguaje de programación C# original se estandarizó a través de la Ecma International en Diciembre del 2002 como *Estándar ECMA-334: Especificación del Lenguaje C#*. Desde entonces, Microsoft propuso varias extensiones del lenguaje que se han adoptado como parte del estándar Ecma C# revisado. Microsoft hace referencia al lenguaje C# completo (incluyendo las extensiones adoptadas) como **C# 2.0**

3.1.1.2 TECNOLOGIA .NET

La tecnología. NET se basa en una estructura de capas de servicios responsables del control de las aplicaciones que se ejecutan en su estructura. Ella tiene una gran librería de código, que sustituye a la actual API de Windows. El .NET Framework tiene su principal idea, muy similar a la plataforma Java, que utiliza una máquina virtual. Sin embargo la idea. NET Framework se muestra más completo y ambicioso, ya que dio la apertura para que

varios lenguajes puedan elaborarse y ejecutarse en su estructura, es decir, que aporta una independencia en el lenguaje de programación. (Franklin, 2002).

3.1.1.3 Principales Recursos del Framework

Ensamblados: En Windows siempre hay problemas en las DLL que se utilizan en la mayoría de las aplicaciones. Este problema llegó a ser llamado "el infierno de las DLL.
Un caso muy común para ilustrar este problema es una aplicación que sustituya una DLL por otra del mismo nombre, pero la versión o Desarrolladores diferente. Esto tuvo como consecuencia, en la mayoría de casos, los problemas en el funcionamiento del software en el que se ha reemplazado el archivo DLL, con ello, había que volver a instalar el software, de modo que regresar a los programas de trabajo ocasionaba problemas.
Más .NET Framework creció un poco con la creación de ensamblados que, de alguna manera reemplazaban a las DLL o EXE.

Los ensambladores son un conjunto de archivos físicos, que en su mayoría contienen datos, tales como partes de código, clases, pero también pueden contener otros tipos de archivos como fotos, por ejemplo. (ibid).

En .NET Framework, el código administrado se implementa como ensamblados. Usted puede considerar que los ensambladores se parecen a las DLL o EXEs de VB, aunque mucho más simples que los EXEs o DLL, que básicamente son archivos binarios.

Los ensamblados pueden ser de dos tipos: privados y compartidos. El ensamblador privado, se limitan al programa que los creó para ser utilizada por él, con ello lograría minimizar el problema de las DLL, de esta forma las aplicaciones tienen sus componentes protegidos.
El ensamblador compartido puede acceder a todos los ensambladores de la máquina, o cuando un programa se ejecuta, intenta encontrar un ensamblador privado.

Metadatos: .NET Framework se ha creado en la opinión de que el código debe ser auto descriptivo, de modo que otros instrumentos de desarrollo se realizan a través de los metadatos, los mismos que pueden aprender a interactuar con el programa.

Los metadatos son importantes para el acceso remoto a datos, ya que se definen las formas de acceso a los datos, es decir, si se quiere tienen acceso directo a los datos, o si los datos deben ser codificados antes de transmitirse por la red y la decodificación en la salida.

Así pues, los metadatos se pueden utilizar para localizar y cargar clases, generar código nativo y garantizar la seguridad de la información que manejan los programas, como en el caso de la red, el destinatario es el que guarda la información básica de los programas.

Common Language Runtime (CLR): El CLR es básicamente el medio ambiente de ejecución de las aplicaciones. NET. Es el responsable de la interacción con el sistema operativo.

También se ocupan de la gestión de memoria a través de un recolector de basura, que recorre y limpia la memoria en las partes correspondientes a datos que ya no están en uso, mejorando así el rendimiento y evita errores en el sistema.

Cuando un código se compila a CLR se llama código administrado, es decir, el código que contiene los metadatos de los tipos de información, los miembros, las referencias en el código, entre otros.

El CLR se encarga de las siguientes acciones dentro del Framework:

- Sistema de tipos comunes (Comum Type System)
- Recolector de Basura (Garbage Collector)
- Compilación Just-In-Time (se explica más adelante)
- Control de versiones (se explica más adelante)
- Seguridad

Compilación Just-In-Time (JIT): En algunos entornos de desarrollo el código sólo se compila en el momento de la implantación, pero con la aparición de el código byte de Java, lo que

retrasa la compilación hasta el momento de la ejecución, la posible ejecución de código en diferentes plataformas sin la necesidad de diferentes paquetes de instalación.

Cuando un programa se realiza a través de Visual Basic. NET y compilado para el idioma IL, este código es independiente de plataforma, por lo tanto, puede funcionar en cualquier equipo que tenga instalado el CLR el equipo.

Desde el momento en que el código se ejecuta inicia proceso del compilador JIT. El CLR comprueba que este código ha sido compilado y almacenado en caché, a continuación, se verificará si el código es de tipo "caja de seguridad, o se implementa la seguridad a través de los tipos CLR y el compilador JIT.

Por lo tanto, la función básica del JIT es la de convertir las instrucciones en IL, para instrucciones específicas del lenguaje de máquina de la computadora donde el código se está ejecutando.

Este compilador está diseñado para máquinas de bajo rendimiento, con pocos recursos de CPU, como los handhelds, por ejemplo. También mantener un registro de los códigos, a fin de eliminar los códigos menos utilizados, y volver a compilar, si es necesario, con el fin de ahorrar memoria del programa.

Namespaces : Los Namespaces se utilizan para proporcionar componentes, tales como clases, la estructura y la interfaz entre otros, que son organizados y utilizados, porque para el uso de COMS sólo existen dos niveles, el nombre del componente y de clase, sin embargo con. NET se puede tener muchas combinaciones.

Las aplicaciones COM usan la GUID (Global Unic Identifier), utilizados como un identificador único para los componentes, de modo que los nombres se limitan a la asociación con el nombre de la biblioteca de una clase. En .NET los GUID se encuentran sustituidos por los Namespaces, lo que da una organización más lógica mediante la reducción de las limitaciones de nombre.

En .NET los componentes están vinculados a un espacio de nombres (**Namespaces**).

Biblioteca de clases de. NET Framework: .NET Framework proporciona una variedad de clases y componentes listos para su uso con lenguaje compatible con CLR. Estos elementos pueden ser ventanas del sistema, una biblioteca, la entrada y salida de datos, el acceso a las funciones de I / O, gestión de memoria, acceso a los datos y la seguridad. NET. Con ellos puedes conseguir muchos recursos que se habían reparado antes de que sea desarrollado a partir de cero. La biblioteca de clases está incluida en los espacios de nombres, y dentro de el Sistema de nombres son todas las clases que representan todos los tipos, los datos básicos utilizados para construir aplicaciones.

Compact Framework

.NET Compact Framework (. NET CF) es una parte de .NET Framework y tiene por objeto proporcionar un entorno para desarrollar aplicaciones de forma fácil para dispositivos con Windows CE. El beneficio es que los desarrolladores tienen la misma facilidad en el desarrollo de aplicaciones para los dispositivos móviles utilizando el mismo IDE y cualquiera de los lenguajes que tiene .NET.

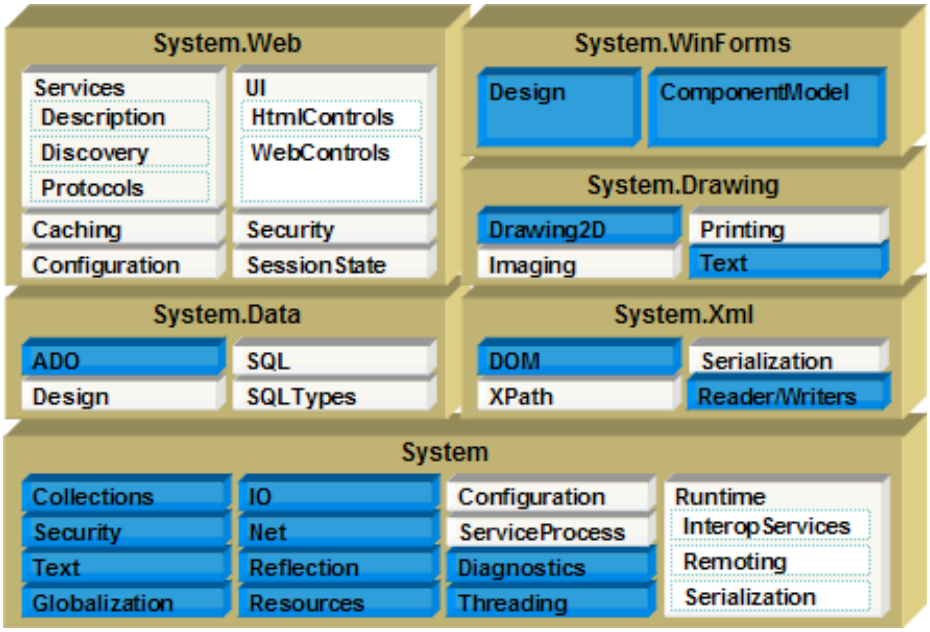


Figura 12. Clases de Compact Framework

3.1.2 POCKETSTUDIO

El PocketStudio fue especialmente desarrollado para que los programadores de Delphi también programen para PalmOS, sin muchas dificultades en su adaptación.

3.1.2.1 Historia de PocketStudio

PocketStudio es la primera herramienta en utilizar el lenguaje Pascal como base. Su interfaz es visual y es muy similar a Delphi, y se ha convertido en una herramienta muy esperada por estos desarrolladores.
Esta herramienta fue desarrollada en 1999 por Pocket -- Tecnologías, que funciona en el estado de Minnesota en los Estados Unidos.

3.1.2.2 Características Pocket Studio

El Pocket Studio permite crear ejecutables nativos de PalmOS, con la posibilidad de acceder a la API del sistema operativo. Durante el comienzo de un nuevo proyecto, esta herramienta es capaz de indicar la versión de software de PalmOS o el software a ser desarrollado, lo que puede traer varias ventajas, como un programa que va a funcionar en diferentes versiones del sistema operativo sin inconvenientes.

Esta herramienta también permite depurar directamente en el emulador, que facilita el descubrimiento de errores en el software. La utilidad del emulador también permite la creación de software sin siquiera tener un ordenador de mano Palm, ya que todos las pruebas se pueden hacer directamente al emulador.

Tiene acceso a la base de datos Oracle Lite o la base de datos Palm, que es una base de datos nativa del PalmOS, accesorios y tecnologías de apoyo como Conductos que se encarga

de la comunicación con el ordenador, bluetooth y para la comunicación inalámbrica con otros equipos inalámbricos.

3.1.3 JAVA

El lenguaje Java es un lenguaje de programación que fue orientado a objetos desde su surgimiento. En Java, todo es una clase con propiedades y métodos. En el momento de su desarrollo, su creador, James Gosling, a sabiendas de que el lenguaje C y C + + eran de alto conocimiento entre los desarrolladores de Sun, decidió hacer de este nuevo lenguaje con una sintaxis básica similar al lenguaje C + +.

3.1.3.1 Historia

En 1991 Sun Microsystems, empresa creadora del lenguaje Java, con el apoyo algunos desarrolladores de hacer algo innovador, en la manera que consideraron mejor. Estos desarrolladores decidieron crear algo para integrar digitalmente los diferentes productos tales como televisores, reproductores de CD, ordenadores y otros aparatos.

Durante este desarrollo, diversas tecnologías han sido probadas e incluso inventadas. Uno de ellos fue el lenguaje C++, que es un buen lenguaje para programación, pero no ofrece portabilidad y tamaños pequeños lo suficientemente pequeña como para instalar en los dispositivos pequeños. Para resolver estos problemas, surgió el Oak (Roble), que es un lenguaje creado por James Gosling y en su inicio fue utilizado sólo internamente por Sun Microsystems.

Tras darse cuenta de que el nuevo lenguaje es un potencial muy grande, había cambiado su nombre a Java.

Un año después de su creación, fue lanzado un pequeño dispositivo con una pequeña pantalla. Este dispositivo se llama Estrella-7 y era un handheld. Para hacer el uso de

este equipo más interesante, se creó un personaje animado, llamado Duke, que fue adoptado como mascota oficial del Lenguaje Java, tanto por el creador de Java como el primer dispositivo creado, como se ilustra en la siguiente figura:

Figura 13 . James Gosling, Star-7 y Duke

A principios de 1993, una buena oportunidad para que el Lenguaje Java, ya que puede ser instalado en los convertidores de televisión por cable. A pesar de esta oportunidad, la empresa que requeria de estos servicios acabo por utilizar otra tecnología. Otras oportunidades surgirán para la utilización del Lenguaje Java, pero ninguno se le dio continuidad.

Con estas dificultades, varios miembros abandonaron el proyecto de este lenguaje, pero en la misma temporada, surge un programa llamado Mosaic que utiliza el protocolo HTTP y permita la visualización de documentos HTML a través de Internet. Con la aparición de este programa, los miembros restantes decidió crear un programa similar pero con una interfaz interactiva, segura, robusta e independiente de la arquitectura, es decir, dando la posibilidad de funcionar en cualquier sistema operativo o en otro equipo digital.

Así, durante 1994, fue creado un navegador llamando HotJava, que poseia un mecanismo que permitia que 2 usuarios interactuen por medio de páginas HTML, hubo cosas interesantes, como objetos en 3D animados que se movían como el movimiento del mouse.

En 1995, Sun Microsystems tuvo una actitud importante disponibilizando libremente el Lenguaje Java con el fin de obtener la atención de desarrolladores de todo el mundo. Esta

actitud fue un éxito, después de un poco tiempo los miembros del equipo pasaban días y noches respondiendo e-mails, discutiendo errores y preparando nuevas versiones.

3.1.3.2 Funcionamiento de Java

Como ya se ha establecido para el desarrollo de los pequeños dispositivos electrónicos, el lenguaje Java demostró ser ideal para ser utilizada en Internet. Lo que lo hace tan atractivo para este tipo de aplicación es que los programas pueden ser desarrollados para ser implantados prácticamente en cualquier plataforma, es decir, se trata de un lenguaje común.

Un software desarrollado en Java, una vez compilado, tiene su código fuente transformado en código de bytes, también conocido como j-código. Este código byte es interpretado por la máquina virtual Java (JVM), que no es sino una máquina virtual que comprende todo el contenido del código de bytes y se convierte en un sistema operativo que está siendo ejecutando. La JVM puede ser utilizada a través de software, es decir, cuando está instalado en la plataforma que se ejecutará a través de hardware o por medio de un microchip, que permita ejecutar el código de bytes mucho más rápido

Con la existencia de la máquina virtual, el código byte se genera sólo una vez, porque la máquina virtual para cada plataforma se ocupará del resto, que en si es la aplicación en la plataforma deseada. Esta portabilidad no es posible en el lenguaje C ya que este lenguaje necesita que exista una compilación propia para cada plataforma.

Actualmente existen compiladores que convierten el código byte en instrucciones nativas de las máquinas, como por ejemplo, el compilador Just In Time (JIT), lo que hace que el software desarrollado se ejecute más rápido a través de la maquina virtual. El compilador necesita una versión específica para la plataforma en la que el programa se compilara y la velocidad de ejecución también dependerá de la cantidad de memoria, porque un código byte compilado por el JIT significa un proceso tres veces superior.

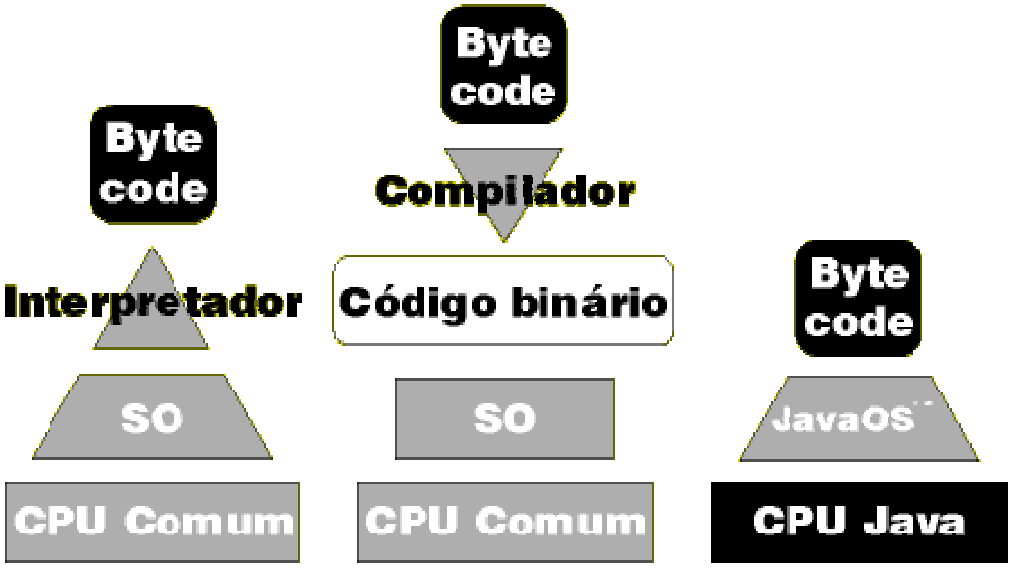


Figura 14. Estructura lógica de funcionamiento de Java.

A pesar que el Lenguaje Java inicialmente fue diseccionado para ser utilizado en aplicaciones para Applets, que permitían la interacción con documentos HTML, este lenguaje ofrece muchos otros recursos para la construcción de aplicaciones standalone, que son independientes del entorno de Internet.

Cómo es independiente de Internet, este lenguaje se pueden utilizar en el desarrollo de software completos, con el acceso a la base de datos, utilizando altos recursos gráficos entre otros. También se utiliza en el desarrollo de software para los equipos como celulares, reproductor de MP3, handhelds.

3.1.3.3 Características de Java

El lenguaje Java directamente en su arquitectura ofrece muchas funcionalidades, lo que hace que este lenguaje sea muy completo. A continuación sus características:

- **Al igual que en el lenguaje C++:** Cómo fue basado en el lenguaje C++ durante su desarrollo, Java hace una buena elección para los desarrolladores del lenguaje C++

que requieren portabilidad, incluso la filosofía del lenguaje es diferente. El lenguaje Java también se deriva de las tareas que el programador puede manejar un error, como gestión de memoria y punteros. Esto significa que los desarrolladores pueden dedicar menos tiempo a preocuparse por los errores en la aplicación en el código y dedicar este tiempo en el desarrollo de la funcionalidad. Java también tiene muchas otras características heredadas de otros lenguajes de programación: Objective-C, Smalltalk, Eiffel, Modula-3, entre otros. Muchas de las características de este lenguaje no son totalmente nuevas. Java es una feliz unión de tecnología probada por diversos centros de investigación y desarrollo de software.

- **Portabilidad:** El software desarrollado en Java puede ejecutarse en cualquier equipo que tenga un intérprete de Java instalado. Como el Java es interpretado, señala que cada equipo que desee utilización debe tener instalado un intérprete. Diferentes plataformas puede ser usado en cualquier versión de Windows, superior o igual a Windows 95, Linux, Solaris, MAC / OS y otros.

- **Robustez:** Java es robusto, porque consigue resolver los problemas de los programadores en precisar resolver un programa de tipo portátil. El Lenguaje Java posee un control automático de memoria, y no necesita la utilización de punteros, que es el mayor causante de problemas en el Lenguaje C++.

  Este control o recolector de basura, escanea la memoria, y libera la información que ya no está siendo utilizada.

- **Orientación a Objetos**

  El lenguaje Java es orientado a objetos, lo que permite todas las funciones, como ejemplo las clases, herencia, polimorfismo, y la excepción de herencia múltiple. Los objetos, permiten la reutilización de código, con lo que también una dinámica durante la programación.

Distribuido

Java proporciona una colección de clases para su uso en aplicaciones de red, que permiten abrir sockets y establecer y aceptar conexiones con servidores o clientes remotos, facilitando así la creación de aplicaciones distribuidas.

Interpretado y Compilado a la vez

Java es compilado, en la medida en que su código fuente se transforma en una especie de código máquina, los bytecodes, semejantes a las instrucciones de ensamblador. Por otra parte, es interpretado, ya que los bytecodes se pueden ejecutar directamente sobre cualquier máquina a la cual se hayan portado el intérprete y el sistema de ejecución en tiempo real (run-time).

Robusto

Java fue diseñado para crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Sus características de memoria liberan a los programadores de una familia entera de errores (la aritmética de punteros), ya que se ha prescindido por completo los punteros, y la recolección de basura elimina la necesidad de liberación explícita de memoria.

Seguro (?)

Dada la naturaleza distribuida de Java, donde las applets se bajan desde cualquier punto de la Red, la seguridad se impuso como una necesidad de vital importancia. A nadie le gustaría ejecutar en su ordenador programas con acceso total a su sistema, procedentes de fuentes desconocidas. Así que se implementaron barreras de seguridad en el lenguaje y en el sistema de ejecución en tiempo real.

Indiferente q la Arquitectura

Java está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, desde Unix a Windows Nt, pasando por Mac y estaciones de trabajo, sobre

arquitecturas distintas y con sistemas operativos diversos. Para acomodar requisitos de ejecución tan variopintos, el compilador de Java genera bytecodes: un formato intermedio indiferente a la arquitectura diseñado para transportar el código eficientemente a múltiples plataformas hardware y software.

El resto de problemas los soluciona el intérprete de Java.

Portable

La indiferencia a la arquitectura representa sólo una parte de su portabilidad. Además, Java especifica los tamaños de sus tipos de datos básicos y el comportamiento de sus operadores aritméticos, de manera que los programas son iguales en todas las plataformas.

Estas dos últimas características se conocen como la *Máquina Virtual Java* (JVM).

Alto rendimiento

Multihebra

Hoy en día ya se ven como terriblemente limitadas las aplicaciones que sólo pueden ejecutar una acción a la vez. Java soporta sincronización de múltiples hilos de ejecución (*multithreading*) a nivel de lenguaje, especialmente útiles en la creación de aplicaciones de red distribuidas. Así, mientras un hilo se encarga de la comunicación, otro puede interactuar con el usuario mientras otro presenta una animación en pantalla y otro realiza cálculos.

Dinámico

El lenguaje Java y su sistema de ejecución en tiempo real son dinámicos en la fase de enlazado. Las clases sólo se enlazan a medida que son necesitadas. Se pueden enlazar nuevos módulos de código bajo demanda, procedente de fuentes muy variadas, incluso desde la Red.

Produce applets

Java puede ser usado para crear dos tipos de programas: aplicaciones independientes y applets. Las aplicaciones independientes se comportan como cualquier otro programa escrito en cualquier lenguaje, como por ejemplo el navegador de Web HotJava, escrito íntegramente en Java.

Por su parte, las applets son pequeños programas que aparecen embebidos en las páginas Web, como aparecen los gráficos o el texto, pero con la capacidad de ejecutar acciones muy complejas, como animar imágenes, establecer conexiones de red, presentar menús y cuadros de diálogo para luego emprender acciones, etc.

3.1.4 SuperWaba

SuperWaba es una plataforma para la programación de dispositivos pequeños. (PDA's). Define un lenguaje, una máquina virtual, un formato de ficheros .class y un conjunto de clases base. SuperWaba desciende de Waba y es compatible con esta. Se puede decir que SuperWaba es Waba pero más desarrollada.

Debido al método en que ha sido diseñado, los programadores pueden usar herramientas de desarrollo orientadas a JAVA para crear programas que se ejecuten en SuperWaba. Muchas de estas herramientas son baratas e incluso gratis.

La sintaxis de los programas escritos para SuperWaba es un subconjunto del lenguaje JAVA, lo que permite que los desarrolladores que estén familiarizados con JAVA puedan comenzar rápidamente a utilizar el SuperWaba.

El formato de los ficheros clase *(.class)* de SuperWaba son también subconjuntos del formato JAVA. Sin embargo SuperWaba no deriva de JAVA ni tiene que ver con Sun Microsystems.

El lenguaje definido por SuperWaba, su máquina virtual y el formato de los ficheros clase han sido diseñados de forma tal que sean óptimos para su uso en PDA's.Las características de

JAVA que usaban mucha memoria o que eran innecesarias para los PDA's han sido omitidas en el diseño del lenguaje y su máquina virtual.

SuperWaba posee un conjunto de clases base diseñadas para ser lo más pequeñasposible y aun así contener las funcionalidades necesarias para escribir programas profesionales. Contiene también un conjunto de clases que permiten que los programas escritos en este lenguaje puedan ejecutarse en cualquier plataforma que interprete el JAVA. Por ejemplolos programas escritos en este lenguaje pueden ejecutarse en una PalmPilot, Ipaq, como Applets,o como aplicaciones independientes en UNIX , WINDOWS ,MS-DOS e incluso en una calculadora TI (¿pensabas que tu PALM era un dispositivo pequeño?).

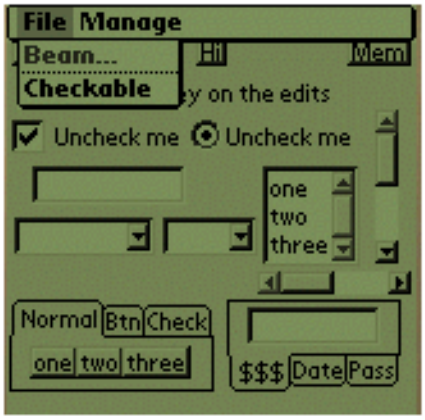


Figura 15. Waba ejecutándose en una Palm.

SuperWaba ha sido diseñado para dispositivos pequeños. Su Máquina Virtual incluyéndolas clases bases esta en el orden de los KiloBytes (270 Kb) en su última versión necesita muy poca memoria para ejecutarse.

- **Funcionalidad**

  SuperWaba permite desarrollar programas profesionales en casi todas las plataformas en un lenguaje muy extendido y fácil de usar. Además es orientado a objeto se incluye características de JAVA como el recolector de basura y el chequeo de rangos que agilizan el tiempo de desarrollo y permite la creación de aplicaciones más robustas.

- **Seguridad**

  Los PDA's normalmente no poseen dispositivos de almacenamientos externos como diskettes o discos duros, lo que significa que sin un programa corrompe esa memoria seguramente habrá que resetear el dispositivo, perdiéndose así todos los datos almacenados. SuperWaba protege los accesos a memoria para prevenir estos fallos. Además al tener un recolector de basura, son muy raros los gastos de memoria innecesaria comparada con programas desarrollados en otros lenguajes.

- **Multiplataforma**

  Con SuperWaba, puedes escribir un programa que se ejecute en PalmOs, Windows CE, o cualquier otra plataforma que soporte el JAVA (JDK 1.02, 1.1 , 1.2 o 2.0). Las alternativas a programar en SuperWaba es programar en un lenguaje específico de la plataforma. Lo que significa escribir código C nativo y APIS específicas. Estas APIS suelen ser muchas, difíciles de programar y pequeños errores de programación suelen llevar a errores difíciles de encontrar y corregir.

3.2 Metodología

Para realizar un análisis de los lenguajes de programación: Super Waba, Pocket Studio y Visual C#. Net, que se utilizan en el desarrollo de software para dispositivos móviles se debe realizar una comparación, se utilizaron criterios como: evaluación del desempeño de cada uno de los lenguajes mediante pruebas realizadas en los dispositivos móviles, usando un modelo Zire71 con un sistema operativo PalmOS, y otro modelo HP IPaq1940 con el sistema operativo Windows CE con el Shell Pocket PC.

Para evaluar el desempeño de los lenguajes se ha desarrollado un software que tiene por objeto realizar algunas pruebas, que se dividen en tres tipos: resultados de vídeo, matemáticas y memoria.

Para cada una de las pruebas que se realizaron, independientemente del tipo al que pertenece, se ha medido su tiempo de ejecución en segundos, que es la diferencia entre la hora de inicio y la hora de finalización. Con la cantidad de tiempo para cada tipo de prueba se calculó el tiempo total de la prueba. Para evaluar de mejor cada prueba, se han dividido en sub-pruebas, como se muestra a continuación:

3.2.1 Rendimiento VIDEO

Cambiar los colores en la pantalla; Un cuadrado de tamaño 160 por 160 píxeles, donde dentro de un bucle de repetición se han cambiado los colores en forma aleatoria por 3000 veces.

Crear círculos de colores en la pantalla; Dentro de un área de tamaño de 160 por 160 píxeles, se han diseñado círculos para cubrir toda la zona este de izquierda a derecha y de arriba abajo dentro de un bucle de repetición que cambia los colores de manera aleatoria por 3000 veces.

3.2.2 Rendimiento MATEMÁTICO

Para el cálculo de la función factorial; Se calculó el factorial del número 700.000.

Se ejecuta una función para calcular el número PI, por 700000 veces, donde cada vez su precisión es mayor.

Se calculó la función del número primo; para el número 700000, dividiendo por números consecutivos del 1 al 700.000.

3.2.3 Rendimiento de Memoria

Guardar 2000 registros en un archivo de datos;
Leer 2000 de un archivo de datos;

Este software fue implementado en la misma forma en cada lenguaje para cada sistema operativo, en consonancia, es decir, PocketStudio para el PalmOS, **Visual C#. Net** para Windows CE y SuperWaba para ambos sistemas operativos.

Debido a que SuperWaba trabaja a través de una máquina virtual, que está disponible para ambos sistemas operativos dirigida, se puede utilizar SuperWaba como base para la comparación, porque en función del rendimiento en ambos dispositivos móviles, puede definir la diferencia en rendimiento entre un idioma y un lenguaje interpretado.

La evaluación del desempeño de cada lenguaje se realiza de la siguiente manera:

- En primer lugar, se ejecutó el programa de pruebas desarrollado con el lenguaje SuperWaba los dos sistemas operativos. Una última vez obtenidos en cada sistema operativo, se define en cuál de los dos equipos se está teniendo un mejor rendimiento.
- En segundo lugar se corre el programa de pruebas desarrollado con **Visual C#. Net** para el sistema operativo Windows ce, obteniendo un tiempo medio final para este lenguaje.
- Por último se realizó un programa de prueba desarrollado con PocketStudio para el sistema operativo PalmOS, Obteniendo un tiempo final para este lenguaje. En base a los datos obtenidos con el lenguaje SuperWaba para el sistema operativo PalmOS se podría deducir que el idioma nativo a obtenido un rendimiento significativamente superior al lenguaje interpretado.

Con los resultados de las comparaciones en el desarrollo del lenguaje podría ayudar a un programador en la elección de qué sistema operativo debe incluir en un pocket y qué lenguaje de programación es el más ventajoso para el desarrollo de software.

3.3 ANÁLISIS COMPARATIVO

A través de la evaluación comparativa y el desarrollo de prototipos, se evaluará el desempeño de los lenguajes para ayudar en la elección de la herramienta de desarrollo más idónea.

3.3.1 Flujograma de Funcionamiento del Prototipo para la evaluación de los Lenguajes

El Flujo grama, del Anexo 1, muestra cómo el prototipo funcionará. Se divide en 4 partes.

La primera parte, llamada el prototipo de funcionamiento, muestra el orden lógico de la operación, es decir, el orden como las pruebas se aplican.

La segunda parte, con el nombre de prueba de vídeo es un sub-proceso de ejecución del prototipo, que muestra el orden en que las pruebas se ejecutan en el código fuente. En la segunda, tercera y cuarta parte, llamada la prueba de matemáticas y de memoria de ensayo, respectivamente, también son sub-procesos de la ejecución de prototipos, pruebas de que los informes de orden matemático y otro para poner a prueba la memoria. Los sub-procesos se muestran en detalle en portugués estructurado.

El procedimiento Ejecutar es el responsable de la aplicación de las pruebas. Este será el primero en ser ejecutado por las llamadas funciones Test_Video, responsable de pruebas de vídeo, Test_Matemático responsable de las pruebas matemáticas y Test_Memoria, para las pruebas de memoria.

Cada una de estas funciones son evaluadas en segundos, tiempo que le toma a cada proceso. Para un mejor análisis de cada una de estas funciones también se subdividen en pequeños trozos medibles en segundos, que se imprime en la pantalla paso a paso dando seguimiento a los resultados. La estructura está disponible en el Anexo2.

3.3.2 Evaluación de los Lenguajes

3.3.2.1 Legibilidad

- **Simplicidad Mundial**

PocketStudio, además de heredar los componentes básicos de Pascal, en gran necesidad utiliza la API de Palm para muchas operaciones, lo que implica una gran dificultad para mejorar la legibilidad, ya que el programador tiene que saber los comandos de la API de Palm PocketStudio.

Visual Studio .NET continúa siendo el punto de referencia para la productividad de los programadores. Con un único entorno de programación (IDE) integrado para todos los lenguajes, incluidos Microsoft Visual Basic® .NET, Microsoft Visual C++® .NET y Microsoft Visual C#™ .NET, las organizaciones de programación pueden aprovechar las ventajas de un cuadro de herramientas, un depurador y una ventana de tareas comunes, reduciendo enormemente la curva de aprendizaje del programador y garantizado que siempre podrán elegir el lenguaje más apropiado para su tarea y conocimientos. Con la función para completar instrucciones de IntelliSense y la comprobación automática de errores de sintaxis, Visual Studio .NET informa a los programadores cuando el código es incorrecto y proporciona el dominio inmediato de las jerarquías de clases y las API. Con el Explorador de soluciones, los programadores pueden reutilizar fácilmente código a través de diferentes proyectos e, incluso, generar soluciones multilenguaje que satisfagan con mayor eficacia sus necesidades empresariales. Y, gracias al entorno IDE totalmente extensible, los programadores pueden disfrutar de las ventajas de una activa comunidad de componentes y complementos de otros fabricantes, con componentes y controles que contribuyen a personalizar y ampliar el entorno de acuerdo con sus necesidades.

Los asistentes para aplicaciones, las plantillas de proyecto y los ejemplos de código fuente de Visual Studio .NET permiten a los programadores crear aplicaciones con rapidez para Windows, el Web y dispositivos con una inversión inicial mínima. La ayuda dinámica y Microsoft Developer Network (MSDN®) proporcionan asistencia basada en la tarea y el lenguaje de programación actuales, garantizado que los programadores no se encuentren nunca perdidos en la plataforma Microsoft .NET o en el lenguaje que hayan elegido.

Finalmente, los programadores pueden elegir entre un conjunto de lenguajes modernizados y utilizar el más apropiado para solucionar sus problemas empresariales.
Visual C# .NET, ofrece mayor productividad para el programador de C y C++. Visual C# .NET proporciona soporte de primera clase para componentes con propiedades, métodos, indizadores, atributos, control de versiones y eventos, a la vez que proporciona un soporte sólido y eficaz para la plataforma .NET.
SuperWaba es una plataforma para la programación de dispositivos pequeños. (de ahora en adelante PDA's). Define un lenguaje, una máquina virtual, un formato de ficheros .class y un conjunto de clases base. SuperWabadesciende de Waba y es compatible con esta. Se puede decir que SuperWaba es Waba pero más desarrollada.

Debido al método en que ha sido diseñado, los programadores pueden usar herramientas de desarrollo orientadas a JAVA para crear programas que se ejecuten en SuperWaba. Muchas de estas herramientas son baratas e incluso gratis.

- **Ortogonalidad**

La ortogonalidad es una propiedad de las CPU. Se dice que un conjunto de instrucciones es ortogonal cuando se puede utilizar cualquier modo de direccionamiento en cualquier instrucción. La búsqueda de la ortogonalidad hace que el diseño de la CPU sea más complejo pero aporta una mayor facilidad de programación.

Todos los idiomas han evaluado un conjunto de instrucciones primitivas que permiten la creación de nuevos tipos de variables, creación de matrices y registros, siendo que un registro puede ser creado colocando distintos atributos de varios tipos, incluidos los registros u otros tipos de variables creadas.

- **Control de Instrucciones**

En comparación con los tres idiomas SuperWaba es el que no soporta el comando GOTO.
Dado que se trata de un comando que no se debe utilizar, ya que hace más difícil la legibilidad del programa.

- **Tipos de datos y estructuras**

Sobre el tipo y estructuras de datos, los lenguajes presentan un apoyo valioso para los tipos de variables (integer, real, fecha, etc), incluidos los de tipo lógico, variables de gran uso para este fin, cero se considera falso y uno como verdadero.

- **Consideraciones Acerca de la Sintaxis**

Ya que PocketStudio está basado en Pascal, tiene algunos problemas con la sintaxis, como el control de bloque de código, cuyo finalizador es siempre end, por lo tanto, no determinar qué bloque se está cerrado. Incluidos comandos repeat y end utilizan la sintaxis del resto de la estructura del lenguaje.

En contraste con Visual C#. NET existen los finalizadores de bloque, haciendo fácil identificar, el *if, for, while.*
Pero el SuperWaba, como lo utiliza el lenguaje Java, todos los bloques de código se identifican con abrir y cerrar llaves, las mismas que afectan a la sintaxis.

En los tres idiomas se impide el uso de palabras reservadas como nombres de variables o funciones, lo que ayuda a la legibilidad.

3.3.2.2 Capacidad de Escritura

Todos los lenguajes han evaluado un conjunto completo de instrucciones primitivas, como: *if, for, case, while, registros*, creación de nuevos tipos de variables, que pueden dar origen a estructuras más complejas.

PocketStudio tiene limitaciones en el tratamiento de strings, debe utilizar un array de caracteres, causando dificultades a la hora de trabajar con variables de tipo cadena, por ejemplo, para añadir dos cadenas, se debe utilizar la función concatenación.

- **Soporte de abstracción**

Los tres idiomas son evaluados de apoyo a la abstracción, es decir, posibilidad de crear funciones. Pero sólo SuperWaba y **Visual C#. Net** soportan la programación orientada a objetos, un punto importante en lo que se refiere a la abstracción.

- **Expresividad**

La expresividad es una facilidad con la cual los lenguajes de programación pueden expresar los procesos y estructuras complejas. Tanto SuperWaba, como Visual C#. NET tiene una buena expresividad, porque además permite construir una estructura ligera.

Sin embargo, PocketStudio no permite una buena integración entre uno y otro comando, ya que requieren más variables auxiliares y líneas de código, no permite escribir un código simple. A continuación puede ver un ejemplo, donde se necesitan cinco líneas de código.

```
StrIToA( Aux, Test_Video );
StrCopy( Buffer, 'Tiempo de Test VDO: ' );
```

```
StrCat( Buffer, Aux );
StrCat( Buffer, ' Segundo(s)' );
PSLabel.SetCaption( LblMediaTeste_VDO, Buffer );
```

3.3.2.3 FIABILIDAD

Verificación de Tipo

Es el proceso que sigue el intérprete para verificar que todas las construcciones en un programa tengan sentido en términos de sus tipos de constantes, variables, procedimientos y otras entidades.
Todos los lenguajes realizan control de errores en tiempo de desarrollo. Pocketstudio realiza los controles en el momento de la compilación, SuperWaba y Visual C#. NET durante la escritura del programa, e incluso da sugerencias sobre cómo resolver los errores.

Tanto Superwaba como PocketStudio, dan como resultado un número entero al dividir dos enteros, lo cual puede causar varios problemas al momento de la programación, ya que le verdadero valor debería ser de tipo real. Para resolver este problema es necesario que la división se lleve a efecto entre dos números reales.

Manejo de excepciones

Una vez evaluados los tres lenguajes, se concluye que sólo PocketStudio no es compatible con el tratamiento de excepciones, lo que desmejora la estabilidad del programa desarrollado. Un caso que podría convertirse en un problema es cuando ocurre una sobrecarga de capacidad en una variable de tipo real, ya que tanto SuperWaba como PocketStudio poner en cero la variable contenedora. Sólo en Visual C#. NET genera una excepción (error). Cuando la excepción no se genera en esta situación es problemática porque es difícil descubrir lo que está sucediendo, ya que el valor se devuelve mal.

Nombre (alias)

Este problema puede ocurrir normalmente en 2 situaciones: el uso de los punteros y en el uso de variables por referencia.

En el uso de punteros PocketStudio es el único que presenta este problema, ya que tanto el SuperWaba, como Visual C#. NET no son compatibles con los punteros.

En el uso de variables por referencia, los tres lenguajes evaluados pueden presentar problemas si la función se ha utilizado de manera incorrecta.

. NET permite el uso de la referencia *weak*, que es una característica, con la cual se puede apuntar de una clase a otra clase y después instanciarla.

Legibilidad y la capacidad de escribir

PocketStudio utiliza muchas líneas para hacer un código simple, como se muestra en el ejemplo siguiente.

```
Tempo_VDO := Teste_Video;
StrIToA( Aux, Tempo_VDO );
StrCopy( Buffer, 'Tempo Teste VDO: ' );
StrCat( Buffer, Aux );
StrCat( Buffer, ' Segundo(s)' );
PSLabel.SetCaption( LblMediaTeste_VDO, Buffer );
```

En contraste, ese mismo código de arriba puede ser escrito en sólo dos líneas de código, tanto en SuperWaba, como en Visual C#. NET. Como se muestra en el siguiente ejemplo de Visual Basic. NET y SuperWaba:

- Visual C#. NET

```
Tempo_VDO = Test_Video();
LblTeste_VDO.Text = "Tempo Teste VDO: “ + Convert.ToString(Tempo_VDO) + " Segundo(s)“;
```

- SuperWaba

```
Tempo_VDO = Teste_Video();
lblTeste_VDO.setText( "Tempo Teste VDO: "
+ String.valueOf( Tempo_VDO ) + " Segundo(s)" );
```

El IDE de SuperWaba como Visual C#. NET permiten una escritura rápida del código correcto, ya que hacen llamados a métodos para controlar el lenguaje.

3.3.2.4 Costo

Costo de Capacitación para los Desarrolladores

Los tres lenguajes evaluados tienen centros de formación, en especial Visual C# .Net, ya que ofrece capacitación a mas bajo costo, mientras que existen menos centros de capacitación y a un mayor costo para PocketStudio y SuperWaba. En algunos países en el caso de SuperWaba se tiene la opción de contratar a un instructor, con la condición de contar con una buena infraestructura para la formación.

Visual C#. NET es además el lenguaje que tiene más material para la autoeducación, así como numerosos libros y mucho material disponible en Internet.
En cambio SuperWaba, tiene una buena documentación, pero limitada en los que se refiere a ejemplos, estos están disponibles en la página web del fabricante.

PocketStudio es el lenguaje con menor material disponible en Internet y se limita a los ejemplos que vienen con la herramienta, ya que la documentación es muy pobre y a menudo, necesitan tener conocimiento de la API de Palm, que también tiene una documentación muy pobre.

Costo de la Escritura

SuperWaba y Visual C# .NET, tienen el IDE como gran apoyo, el mismo que indica los errores de escritura en el código y tiene la opción de auto completar, no haciendo necesario que el programador sepa cómo escribir cada comando o método disponible en el idioma. De los controles, clases y funciones que requieren el paso de parámetros, nos indicará que parámetros deben ser transmitidos. En cambio, la PocketStudio no permitir ninguno de estos recursos, lo que obstaculiza el aprendizaje y la programación.

Costos de Producción

Actualmente utilizado por las computadoras, compilar utilizando cualquiera de estos lenguajes evaluados es rápido y sencillo, sin necesidad de grandes inversiones.

Costo de Aplicación

Se evaluó el desempeño de los programas desarrollados y fue posible verificar que SuperWaba tiene una ejecución mucho más lenta, tanto en Pocket PC como en Palm, en comparación con el PocketStudio y Visual C#. NET. Esta aplicación puede requerir un equipo más potente para utilizar un programa desarrollado en este lenguaje.

Costo del sistema y la aplicación del lenguaje

Según los idiomas evaluados todos funcionan con hardware de bajo costo. PocketStudio tiene la gran ventaja en este aspecto, ya que también puede desarrollar aplicaciones de PalmOS

compatible con la versión 2, permitiendo el uso de equipos antiguos. Según la documentación de SuperWaba, aplicaciones desarrolladas con la biblioteca de trabajo en todos los dispositivos PalmOS a partir de la versión 3.0, sin embargo, el programa de pruebas desarrollado no funciona en la versión 3.5 de la PalmOS.

La máquina virtual SuperWaba requiere aproximadamente 1 MB de espacio libre en tu PDA, para instalar tanto en una Pocket PC, como en una Palm.

Costo de la escasa fiabilidad

Evaluó las tres lenguas, se descubrió un error en la lengua PocketStudio que pueden causar problemas en la aplicación después de ser desarrollado. Este error está relacionado con el uso de la instrucción MOD con cifras superiores a 10.000, cuando una división por cero error se produce, poner fin a la aplicación.

SuperWaba la confianza no se puede ejecutar si la versión 5.5 de la máquina virtual, una aplicación desarrollada para la versión 5.0 de la máquina virtual.

Costo de la escasa fiabilidad

Evaluados los tres lenguajes, se descubrió un error en el lenguaje PocketStudio que pueden causar problemas en la aplicación después de ser desarrollado. Este error está relacionado con el uso de la instrucción MOD con cifras superiores a 10.000, cuando una división por cero se produce un error,dando fin a la aplicación.

En SuperWaba por su parte no se puede ejecutar una aplicación echa en la versión 5.0 en la máquina virtual de la versión 5.5.

Costo de Programas de Mantenimiento

Como el SuperWaba y Visual C#. NET soportan la programación orientada a objetos, la forma de mantenimiento es mucho más fácil. El PocketStudio no está apoyado por la programación orientación a objetos y esto es su debilidad. Otro punto de la PocketStudio es que usa mucho la API de Palm, por esta razón el programador debe saber mucho de esta API, para proporcionar el mantenimiento. También debería considerarse el hecho de que la IDE de PocketStudio no trae la información importante en el código, como el tipo de variable que se declaró, ¿cuáles son los parámetros que devuelve una función. SuperWaba, usa una máquina virtual, existe un problema entre las versiones de las máquinas virtuales. Por ejemplo, la aplicación de rendimiento que se ha desarrollado para la versión 5.0 no funciona correctamente cuando se ejecuta en la versión 5.5 de la VM. Esto crea problemas de compatibilidad entre aplicaciones desarrolladas en diferentes versiones.

3.3.3 Documentación

La documentación de Visual C#. NET es muy buena porque tiene una herramienta de ayuda integrada con el método de la explicación para cada plataforma, incluyendo ejemplos y todavía cuenta con el apoyo del sitio de MSDN de Microsoft.
También hay una amplia variedad de libros que pueden ayudar a los programadores. SuperWaba contiene la documentación que acompaña al SDK, que contiene todos los métodos disponibles.

Una debilidad de esta documentación es que solo tiene algunos ejemplos. Pero en compensación tiene muchas aplicaciones de las fuentes disponibles y cuenta con el sitio Web del fabricante que puede ayudar a los programadores. La documentación que acompaña a la PocketStudio es muy simple y no tiene muchos ejemplos. El hecho de no incluir una ayuda en la API del sistema operativo PalmOS, representa un obstáculo.

3.3.4 Dificultades encontradas en el desarrollo

La siguiente es una lista de las limitaciones encontradas en cada uno de los lenguajes durante el desarrollo del prototipo.

PocketStudio

- No se pueden crear objetos en tiempo de ejecución (como un formulario), a menos que use la API de Palm. Por qué no tener esta opción, todos los componentes deben ser puestos en la pantalla visual en tiempo de diseño, lo que dificulta la organización de algunos componentes en la pantalla.
- 32kbytes como límite del tamaño de las variables.

  Esta limitación hace imposible algunas operaciones que requieren una gran cantidad de información almacenada en la unidad.

- Error en la División por o de instrucción MOD.

  Cuando la instrucción MOD se utiliza para número mayor a 100.000 presenta un error de división por cero. Con valores más bajos, dicho error no se presenta con datos menores; podemos deducir este problema es un punto bajo para la herramienta.

- Solo funciona con el emulador integrado en la versión 4 de las PalmOS. La versión actual del PalmOS es 5.4, pero el emulador no es compatible con la versión de esta herramienta de desarrollo. Esto excluye el uso del depurador de la aplicación desarrollada utilizando las características únicas de esta versión de PalmOS.

- Limitación en la resolución de solo 160x160 píxeles.

No es posible desarrollar aplicaciones en una resolución mayor a 160x160 píxeles. La mayoría de los nuevos Palm está disponible en 320x320 píxeles de resolución o de 480X320 píxeles.

SuperWaba

- **Complejidad en la instalación del Pocket PC.**

  La máquina virtual debe estar instalado en su Pocket PC necesariamente en la carpeta raíz: \ SuperWaba \ y la aplicación debe estar en la carpeta raíz: \ SuperWaba \ [Nombre de la aplicación] \, de otra manera la aplicación no funciona.

- **Tratamiento de Plataformas específicas**

  Al eliminar los archivos a través de código de programación se debe especificar la ruta completa (directorio) en el archivo de Pocket PC. Esto es malo, pues no se mantendrá la compatibilidad entre la aplicación de Pocket PC con la Palm.

- **Complejidad en la instalación del compilador**

  La instalación del compilador en la PC es complejo, porque requiere la instalación de varios archivos, como los de Sun J2RE, la biblioteca de la SDK y un editor Superwaba de Java, en Eclipse. También se requiere la configuración de las bibliotecas para cada nuevo proyecto.

- **La falta de un editor visual para construir las pantallas.**

  Dado que no existe ningún editor visual para trabajar con este lenguaje, nos encontramos existe mucha incoherencia entre diferentes resoluciones de pantalla.

- **API no permite el acceso a los equipos**

  Por tratarse de un lenguaje multiplataforma, por lo que no proporciona acceso a la API del dispositivo. Por consiguiente, puede traer algún tipo de limitación.

Visual C#. Net

A lo largo del proceso de desarrollo del prototipo no se tropezó con dificultades.
La única limitación que se encontró es, por ser multi-plataforma, no tiene acceso directo a las API del dispositivo, esto no puede limitar el uso de equipos específicos.

3.3.5 Evaluación de los Resultados Prototipo

Se desarrolló un prototipo para la realización de las pruebas de rendimiento de los lenguajes.

Los equipos utilizados para este análisis fueron los siguientes: un dispositivo usando el sistema operativo PalmOS y uno con Windows CE. Estos equipos tienen las siguientes características:

PalmOS	**Windows CE**
Palm Zire71	Pocket PC HP IPAQ 1940
Procesador: Texas Instruments de 144Mhz	Procesador: Samsung 266Mhz de reloj
Reloj	Memoria: 64 MB
Memoria: 16 MB	Vídeo: 320x240 píxeles - 65mil colores
Vídeo: 320x320 píxeles - 65mil colores	

La Tabla 4, muestra el lenguaje de programación y el sistema operativo evaluado con los lenguajes puestos a prueba. Debajo se muestra un cuadro con el tiempo obtenido para cada

ensayo, una vez que la aplicación se ha desarrollado en Superwaba y se ha ejecutado en Windows CE como el PalmOS, ya que es multiplataforma.

Leyenda
PS =PocketStudio (PalmOS) – Pascal C# =Visual C# .NET (Pocket PC) – C SW =SuperWaba (PalmOS/Pocket PC) – Java

Resultado de la Prueba General

	C# - Pocket PC	Ps – Palm	Sw – pocket PC	Sw - PalmOS
VDO - Cambiar Colores	15 Segundos	5 Segundos	14 Segundos	13 Segundos
VDO- Crear Objetos	10 Segundos	2 Segundos	14 Segundos	21 Segundos
Mat – Factorial	1 Segundos	9 Segundos	1 Segundos	34 Segundos
Mat - Número PI	1 Segundos	32 Segundos	1 Segundos	87 Segundos
Mat - Número Primo	1 Segundos	4 Segundos	1 Segundos	51 Segundos
MEM- Grabar Archivos	3 Segundos	4 Segundos	153 Segundos	17 Segundos
MEM-Leer Archivos	19 Segundos	1 Segundos	3 Segundos	12 Segundos
Tiempo Total	**50 Segundos**	**57 Segundos**	**187 Segundos**	**235 Segundos**

Tabla 4. Resultados de las pruebas de rendimiento Parte 1

Comparación entre el Visual C# .Net y SuperWaba en Windows CE

Desempeño

	C# - Pocket PC	SW – Pocket PC	C# con SW
VDO - Cambiar Colores	15 Segundos	14 Segundos	-6,67 %
VDO- Crear Objetos	10 Segundos	14 Segundos	40,00%
Mat – Factorial	1 Segundos	1 Segundos	0%
Mat - Número PI	1 Segundos	1 Segundos	0%
Mat - Número Primo	1 Segundos	1 Segundos	0%
MEM- Grabar	3 Segundos	153 Segundos	5000%

Archivos			
MEM-Leer Archivos	19 Segundos	3 Segundos	-84,21%
Tiempo Total	**50 Segundos**	**187 Segundos**	**274,00%**

Tabla 5. Comparación entre el Visual C# .Net y SuperWaba en Windows CE. Parte 1

Comparación entre PocketStudio y SuperWaba en PalmOS

Desempeño

	PS – Palm	**SW – PalmOS**	**PS con SW**
VDO - Cambiar Colores	4 Segundos	13 Segundos	225 %
VDO- Crear Objetos	3 Segundos	21 Segundos	600 %
Mat – Factorial	9 Segundos	34 Segundos	277,78 %
Mat - Número PI	32 Segundos	87 Segundos	171,88%
Mat - Número Primo	4 Segundos	51 Segundos	1175,00%
MEM- Grabar Archivos	5 Segundos	17 Segundos	240,00%
MEM-Leer Archivos	1 Segundos	12 Segundos	1100,00%
Tiempo Total	**58 Segundos**	**235 Segundos**	**305,17%**

Tabla 6. Comparación entre PocketStudio y SuperWaba en PalmOS. Parte 1

En la tabla de comparación entre el Visual C#. NET y SuperWaba, se puede ver que en Visual C#. NET tiene un rendimiento muy similar al presentado por SuperWaba. Aunque la tabla de resultados presenta una diferencia de 5.000% en la prueba de grabación de los archivos, porque SuperWaba genera un archivo de tipo. ".PDB", que es el nativo de PalmOS, para escribir los registros. Esta lentitud es reconocida incluso por el fabricante de SuperWaba. Si no existiese este problema la diferencia sería menor.

También se observó que la aplicación de Visual C#. NET es 84,21% más lenta que SuperWaba en la prueba de lectura de archivos archivos, 6,67% más lento en la prueba de cambio de colores y el 40% más rápido en la prueba para crear objetos. Como el código escrito en cada

uno de los lenguajes no tiene limitaciones, esta diferencia de tiempo es asignada directamente a cada uno de los lenguajes.

SuperWaba es un lenguaje que genera un código intermedio que será ejecutado por la máquina virtual, mientras se ejecuta la aplicación. Visual C#. NET también genera el código intermedio, pero este es compilado en su totalidad en el momento de la aplicación del Compact Framework. NET. Se puede concluir, a través de esta prueba que la metodología utilizada por. NET tiene un rendimiento respecto a SuperWaba en la mayoría de los casos.

En la comparación entre la PocketStudio y SuperWaba en la plataforma PalmOS se puede entender que un lenguaje que crea un ejecutable nativo en la plataforma es extremadamente rápido, ganando en todas las pruebas, llegando a más de 305% de diferencia en el resultado global.

Se realizó un segundo experimento con los mismos test, pero cambiando la función de cálculo en la función recursiva del factorial. También observamos que en la función que calcula el número PI no se obtiene un resultado correcto. Este problema se produce en los lenguajes PocketStudio y SuperWaba, y la división de números enteros, que devuelve un número truncado en lugar de devolver un número de punto flotante. Para eludir este problema se debió transformar los enteros en números de punto flotante, y luego hacer la división. Otro tema que se tomó en cuenta es que PocketStudio sólo utiliza una resolución de 160x160 píxeles de la pantalla que es más grande y se amplía para llenar toda la pantalla. Después de que este control se ha cambiado la aplicación desarrollada en dos lenguajes para el uso exclusivo de un área de 160x160 píxeles para los ensayos, con el equivalente PocketStudio. PocketStudio tiene un límite de sólo 32kbytes con el tamaño de las variables.

Con estos cambios se vio una diferencia en los resultados, como se muestra en las siguientes tablas, en comparación con la anterior prueba.

RESULTADOS DE LAS PRUEBAS DE RENDIMIENTO

PARTE 2

Leyenda
PS =PocketStudio (PalmOS) – Pascal C# =Visual C# .NET (Pocket PC) – C SW =SuperWaba (PalmOS/Pocket PC) – Java

Resultado de la Prueba General

	C# - Pocket PC	**Ps – Palm**	**Sw – pocket PC**	**Sw - PalmOS**
VDO - Cambiar Colores	8 Segundos	5 Segundos	10 Segundos	13 Segundos
VDO- Crear Objetos	8 Segundos	2 Segundos	11 Segundos	20 Segundos
Mat – Factorial	1 Segundos	0 Segundos	1 Segundos	78 Segundos
Mat - Número PI	1 Segundos	29 Segundos	1 Segundos	129 Segundos
Mat - Número Primo	1 Segundos	4 Segundos	1 Segundos	68 Segundos
MEM- Grabar Archivos	3 Segundos	4 Segundos	153 Segundos	19 Segundos
MEM-Leer Archivos	19 Segundos	1 Segundos	3 Segundos	12 Segundos
Tiempo Total	**41 Segundos**	**45 Segundos**	**180 Segundos**	**339 Segundos**

Tabla 7 .Resultados de las pruebas de Rendimiento Parte 2

Comparación entre el Visual C# .Net y SuperWaba en Windows CE

Desempeño

	C# - Pocket PC	**SW – Pocket PC**	**C# con SW**
VDO - Cambiar Colores	8 Segundos	10 Segundos	25,00%
VDO- Crear Objetos	8 Segundos	11 Segundos	37,50%
Mat – Factorial	1 Segundos	1 Segundos	0%
Mat - Número PI	1 Segundos	1 Segundos	0%
Mat - Número Primo	1 Segundos	1 Segundos	0%
MEM- Grabar	3 Segundos	153 Segundos	5000,00%

Archivos			
MEM-Leer Archivos	19 Segundos	3 Segundos	-94,74%
Tiempo Total	**41 Segundos**	**180 Segundos**	**339,02%**

Tabla 8 . Comparación entre el Visual C# .Net y SuperWaba en Windows CE Parte 2

Comparación entre PocketStudio y SuperWaba en PalmOS

Desempeño

	PS – Palm	**SW – PalmOS**	**PS con SW**
VDO - Cambiar Colores	5 Segundos	13 Segundos	160,00%
VDO- Crear Objetos	2 Segundos	20 Segundos	900,00%
Mat – Factorial	0 Segundos	78 Segundos	0,00%
Mat - Número PI	29 Segundos	129 Segundos	344,83%
Mat - Número Primo	4 Segundos	68 Segundos	1600,00%
MEM- Grabar Archivos	4 Segundos	19 Segundos	375,00%
MEM-Leer Archivos	1 Segundos	12 Segundos	1100,00%
Tiempo Total	**45 Segundos**	**339 Segundos**	**653,33%**

Tabla 9 . Comparación entre PocketStudio y SuperWaba en PalmOS. Parte 2

Con el cálculo del factorial a través de la recursividad provoca diferentes resultados tanto en SuperWaba como en PocketStudio. Es interesante destacar que con este cambio PocketStudio se ejecuta de una forma mucho más rápida, mientras que SuperWaba se incrementó en más del 100% en la duración de esta prueba.

El cambio en la resolución de vídeo a 160x160 píxeles para que los lenguajes SuperWaba y Visual C#. NET se obtuvo una reducción en el momento de la prueba de vídeo, pero sólo para Windows CE, mientras que SuperWaba con PalmOS sigue siendo igual, lo que demuestra que el área de la pantalla actualización puede hacer diferencia en el rendimiento, pero sólo en Windows CE.

Después de modificar el cálculo del número PI para lograr realizar las divisiones entre los números enteros por números reales, la información que PocketStudio sufre con este tipo de cálculo, es más lento, mientras que el SuperWaba ha demostrado un desempeño más rápido, lo que demuestra que este lenguaje ofrece un mejor rendimiento con los cálculos utilizando los números reales. Con la sustitución de la matriz por una única variable, durante la lectura de los registros, sólo SuperWaba ha reducido en un 66% en el momento de la lectura, pero el resto sigue siendo la misma.

Las pruebas con los archivos de registro y los números primos no han cambiado entre una prueba y otra, pero mostraron una reducción en el tiempo de las pruebas en el idioma SuperWaba ejecutando en PalmOS. Y este cambio de hora debe estar relacionado con otros cambios que se hicieron y, en definitiva, influyen en el resto de la prueba.

En las figuras 16 y 17 puede comparar mejor la diferencia que se produjo en los resultados de la Parte 1 y Parte 2 de las pruebas para la plataforma Pocket PC.

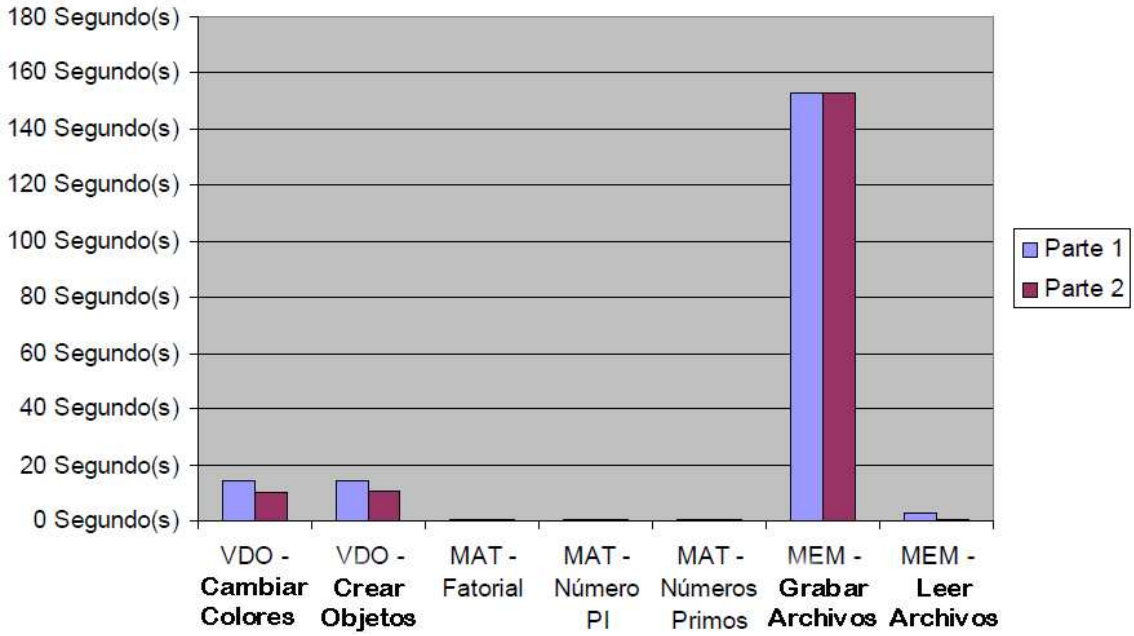


Figura 16. . COMPARACIÓN SUPERWABA - POCKET PC

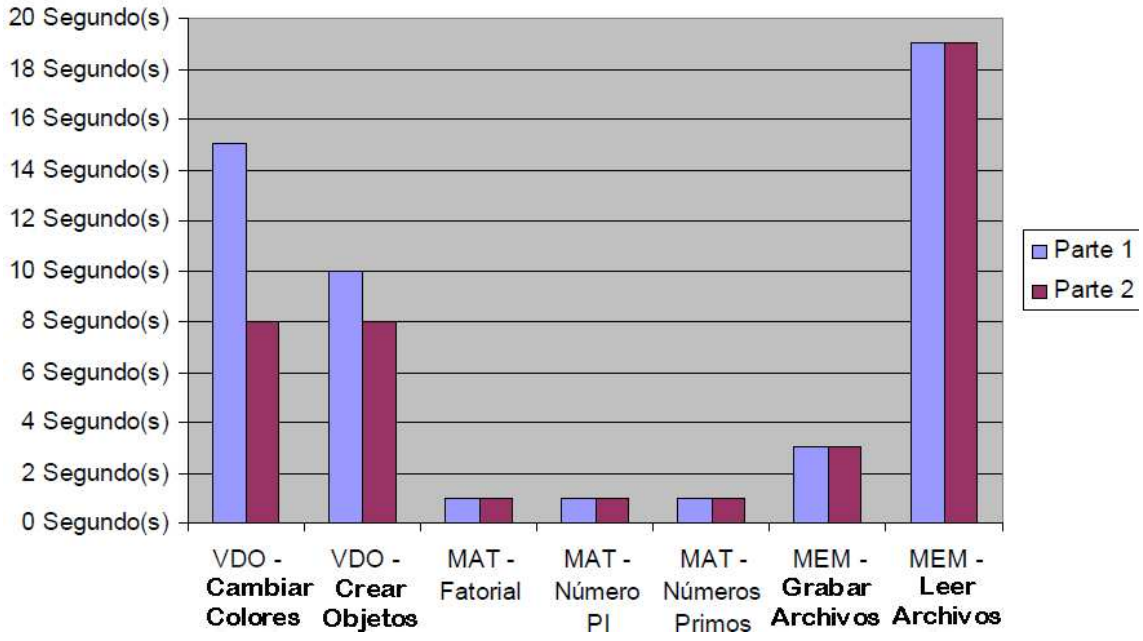


Figura 17. Comparación de Visual C#. Net - POCKET PC

La Figura 18 muestra la comparación entre el SuperWaba y Visual C#. NET, donde se puede comprobar fácilmente que SuperWaba es mucho más lento que Visual C#. NET, ya que tuvo cerca de 180 segundos para realizar toda la prueba, mientras que el prototipo desarrollado en Visual C#. NET tuvo menos de 50 segundos para realizar la totalidad del ensayo.

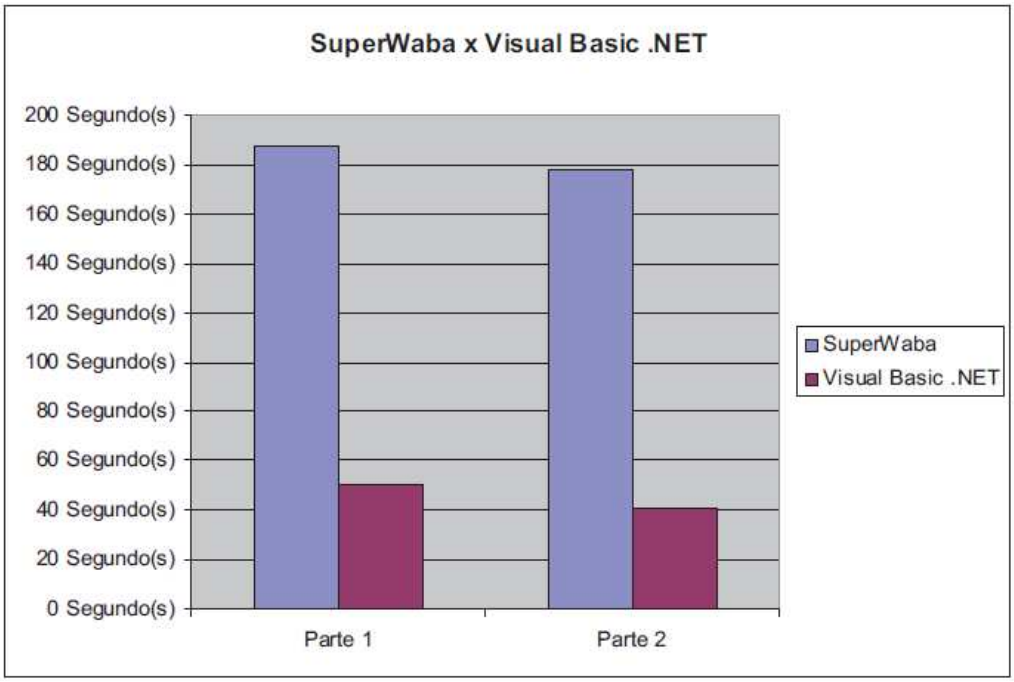


Figura 18. COMPARACIÓN SUPERWABA con VISUAL C#. NET

En las figuras 19 y 20 se aprecia de mejor manera la diferencia que se produjo en los resultados de la Parte 1 y Parte 2 de las pruebas para la plataforma PalmOS.

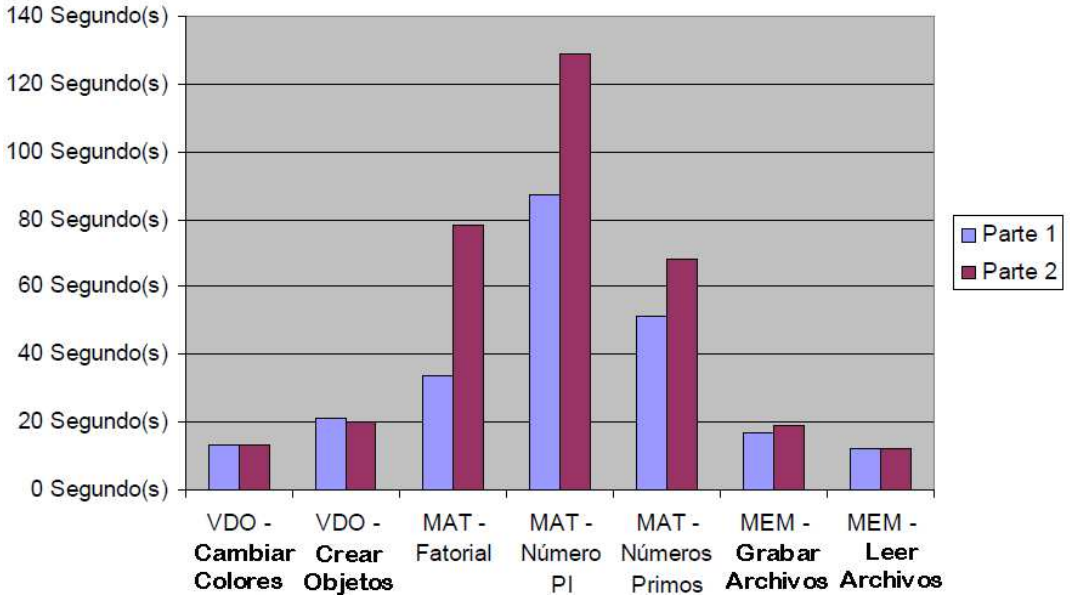


Figura 19. COMPARACIÓN SUPERWABA - Palm

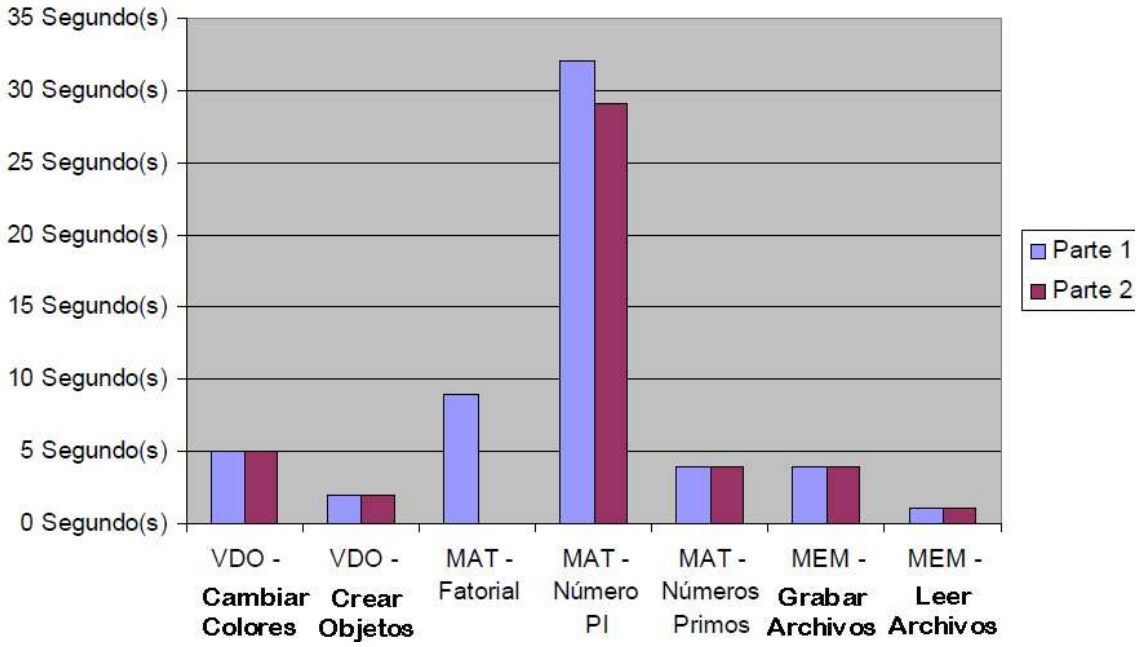


Figura 20. Comparación POCKETSTUDIO - PALM OS

La Figura 21 muestra la comparación entre el SuperWaba y PocketStudio donde se puede comprobar fácilmente que SuperWaba es mucho más lento que el PocketStudio, por lo tanto, tuvo casi 350 segundos para realizar todas la segunda prueba, mientras que el prototipo desarrollado en PocketStudio tomó menos de 50 segundos.

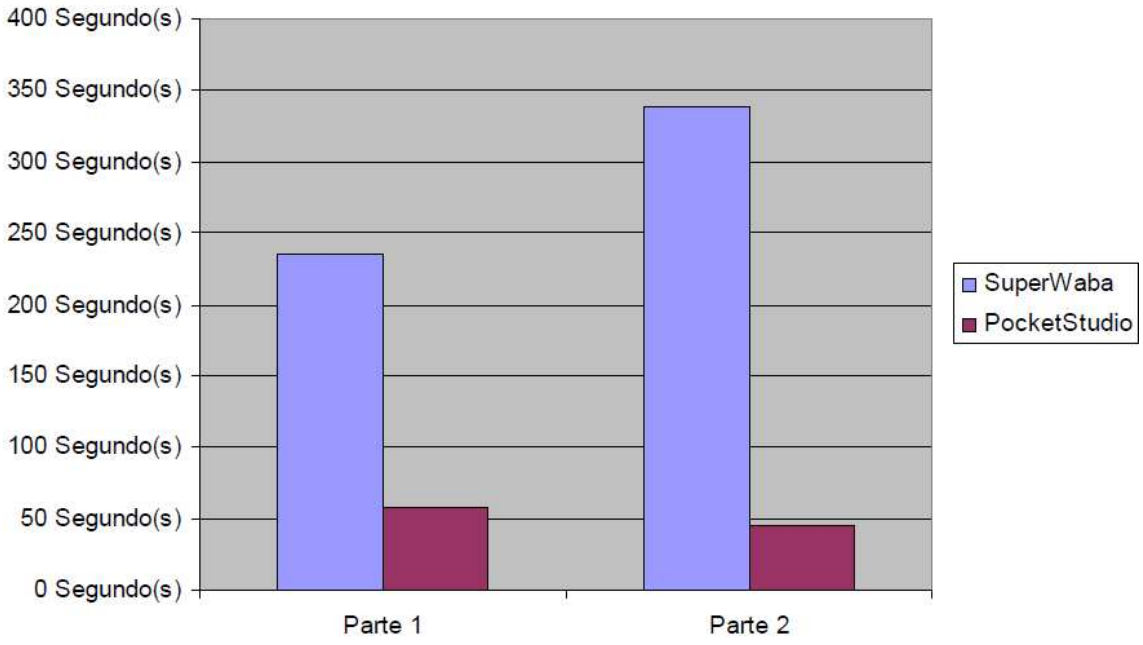


Figura 21. COMPARACIÓN SUPERWABA x POCKETSTUDIO

CONCLUSIÓN

Para obtener ayuda en la elección del lenguaje de programación, la presente investigación tiene por objeto comparar los lenguajes para los dispositivos móviles y crear un prototipo para probar el rendimiento entre los mismos. Los resultados obtenidos tienen información importante para la elección de un lenguaje de programación que satisfaga las necesidades del programador en el área para que el software sea desarrollado.

Entre los lenguajes que han sido evaluados, SuperWaba, PocketStudio y Visual C#. NET, se ha desarrollado un prototipo de esta investigación en la que se ha demostrado que estos lenguajes tienen muchas diferencias entre sí, tanto en la cuestión de la sintaxis, como también en el rendimiento.

Con este estudio se puede comprobar la efectividad y eficacia de C#. Net, con respecto a SuperWaba y PocketStudio, acotando además que este es un lenguaje conocido para nuestro

medio, y existen además varios medios de ayuda tanto escritos como el internet para poder capacitarnos.

Es posible comprobar las grandes diferencias en el rendimiento, donde SuperWaba fue más lento que los lenguajes nativos, sobre todo en cálculos matemáticos para Palm OS y la grabación de archivos en la PC de bolsillo, pero a cambio permite la portabilidad.
Se ha demostrado también que un simple cambio de código genera importantes diferencias en los resultados finales, como el cálculo de del factorial recursivamente ha resultado ser muy interesante.

Se debe prestar mucha atención en el desarrollo de la aplicación y no desperdiciar los recursos, porque el equipo es limitado en su poder de procesamiento en comparación con un PC.

Para una mayor movilidad y agilidad en las actividades de la vida cotidiana, es esencial elegir un lenguaje de programación para desarrollar soluciones inteligentes que integran los procesos y reunir a los usuarios con información actualizada.

Para trabajos futuros, podemos evaluar otros lenguajes de programación utilizando el mismo método y usando este trabajo de investigación como punto de referencia con el fin de dar más opciones para elegir al programador.

Con el auge de la tecnología. NET que promete la portabilidad, también puede ser evaluado el desempeño de lenguaje de Visual C#. NET.

CAPITULO IV

4 DISEÑO E IMPLEMENTACIÓN

INTRODUCCIÓN

La notificación es quizás el punto de contacto con el cliente más amplio en una empresa al registrar lecturas de medidores y entrega de notificaciones, a través del cual la empresa consolida su imagen y todos aquellos aspectos estratégicos que la condicionan dentro del medio en que trabaja porque permite establecer una comunicación directa entre la empresa y el consumidor, a través de la factura como el principal instrumento de cobranza de los servicios.

Para lograr este propósito se ha realizado un estudio exhaustivo de las necesidades tanto de los clientes como de los futuros usuarios del sistema a ser desarrollado en sus distintos módulos.

El proyecto PEINSEB, **"Proyecto para la Entrega Inmediata de Notificaciones de Servicios Básicos",** es el indicado para cumplir con todas estas demandas, y para hacerlo a éste se lo ha dividido en dos módulos que funcionan conjuntamente,. Pero en plataformas distintas:

SISNYCS es la definición elegida para el "***Sistema de Notificaciones y Cobro de Servicios***", el cual es la solución informática portable para el ingreso de lecturas del consumo de agua potable así como también para la notificación a los usuarios del valor exacto a ser cancelado en los Departamentos de Agua Potable, y la comunicación de valores adeudados por la compra de medidores, deudas anteriores, etc, del Cantón Baños.

SISTRAD es la definición elegida para ***"Sistema de Transferencia de Datos",*** dicha aplicación es parte de la solución que permite la extracción e inserción de datos entre las Bases de Datos del dispositivo Móvil y del Servidor, además de la generación de códigos de barra para los medidores de los usuarios del Agua Potable.

La función general del presente Sistema es la mantener el registro y control de la notificación y facturación de los clientes que se encuentran catastrados en el archivo de la empresa, determinación de los consumos de los clientes, por medio de la lectura de los medidores instalados en las conexiones domiciliarias o por estimación del volumen de agua con base en las características de las unidades de consumo con servicio directo.

Justificación Del Uso De La Metodología Xp (Programación Extrema)

La programación extrema se basa en una serie de reglas y principios de la ingeniería de software tradicional pero con la excepción de que XP da prioridad a las tareas que es lo que precisamente da resultados directos y reducen los procesos largos de recopilación de gran información para generar resultados esperados.

La programación extrema es un modelo de desarrollo muy disciplinado y que se apoya en ciertos valores fundamentales y necesarios acoplar a nuestro grupo de trabajo como son: Comunicación, Simplicidad, Realimentación, Coraje.

XP trata los problemas que ocurren con frecuencia en el desarrollo de los sistemas de información en lo que respecta a tiempo, presupuesto, calidad y ámbito.

Debido a que XP se enfoca en la entrega frecuente de resultados es necesario que en poco tiempo se pueda generar resultados de gran importancia, es por eso que si queremos más calidad esto implicara el trabajo e incorporación de un gran equipo de trabajo.

4.1 Desarrollo del La Ingeniería Inversa a la Base de Datos "dbame"

4.1.1 Fase I.

4.1.1.1 Etapa1: Extracción Automática

Para la consecución y elaboración de esta etapa se va a usar la herramienta Case[8] ER Studio 6, ya que siendo gratuita y de libre uso, esta muestra todas las bondades requeridas para la realización y análisis con Ingeniería Inversa de Bases de Datos.

4.1.1.1.1 Herramienta Case ER Studio 6.

Introducción

ER Studio es una herramienta de modelado de datos fácil de usar y multinivel, para el diseño y construcción de bases de datos a nivel físico y lógico. Direcciona las necesidades diarias de los administradores de bases de datos, desarrolladores y arquitectos de datos que construyen y mantienen aplicaciones de bases de datos grandes y complejos.

ER/Studio está equipado para crear y manejar diseños de bases de datos funcionales y confiables. Ofrece fuertes capacidades de diseño lógico, sincronización bidireccional de los

[8] **C**omputer **A**ided **S**oftware **E**ngineering, Ingeniería de Software Asistida por Ordenador

diseños físicos y lógicos, construcción automática de bases de datos, documentación y fácil creación de reportes.

ER/Studio ofrece las siguientes funcionalidades:

- Capacidad fuerte en el diseño lógico.
- Sincronización bidireccional de los diseños lógico y físico.
- Construcción automática de Base de Datos.
- Reingeniería inversa de Base de Datos.
- Documentación basada en HTML.
- Un Repositorio para el modelado.

ER/Studio es una herramienta que modela los datos, se usa para el diseño y la construcción lógica y física de base de datos. Su ambiente es de gran alcance, de varios niveles del diseño.

ER/Studio se diseña para hacer más fácil de entender el estado actual de los datos de la empresa. Simple y fácil al usuario, ayuda a organizaciones para tomar decisiones en cómo resolver embotellamientos de los datos, elimina redundancia y alcanza en última instancia usos de más alta calidad que entreguen datos más eficientes y exactos a la empresa.

Potencial de ER/Studio

Si se está comenzando un nuevo diseño o está manteniendo una base de datos existente, ER/Studio se combina con las características para ayudarle a conseguir el trabajo hecho con eficacia. Con el potencial y la facilidad de empleo de ER/Studio's que modela el ambiente, será productivo rápidamente y podrá casi demostrar resultados inmediatamente después de la instalación.

Diagramas

La creación de diagramas es clara y rápida. Tiene la posibilidad de realizar diagramas con desempeño rápido. También es posible cambiar el estilo de las líneas, los colores, tipos de letra, niveles de acercamiento, y modelos de despliegue. Es posible crear subvistas para separar y manejar áreas importantes. ER/Studio automáticamente mantiene todas las dependencias entre subvistas y el diagrama completo.

El Explorer Navigation facilita el trabajo hasta con los diagramas más grandes. Si se está trabajando con un modelo largo de Datos, ER/Studio ofrece un aumento en la ayuda y fácil navegación en sus modelos. La Apreciación global (overview). Se usa el browser Explorer para encontrar y seleccionar entidades. Un solo clic inmediatamente enfoca una ventana de diagrama.

Ayuda

Ya sea que se inicie un nuevo diseño o se mantenga uno existente, ER/Studio está equipado con elementos de ayuda para hacer el trabajo de manera efectiva. Las barras de herramientas tienen algunas sugerencias para el uso de las mismas, además de contar con ayuda en línea sensible al contexto.

Esquema de la base de datos

Las capacidades de diseño que contiene, ayudan a crear un diseño lógico que puede transformarse en cualquier número de diseños físicos. Como resultado, se puede mantener un diseño lógico normalizado mientras se des normalizan los diseños físicos para su desempeño. ER/Studio mantiene ligaduras entre todos los niveles de su diseño por lo tanto puede mezclar cambios en cualquier dirección entre ellos. ER/Studio revisa la normalización y la compilación con la sintaxis de la plataforma de la base de datos.

Código que genera el lenguaje

Genera otros objetos de base de datos: vistas, procedimientos almacenados, defaults, reglas, y tipos de datos de usuario, lo cual ayuda al auto ordenación de tipos de objetos para eliminar errores de dependencia al construir la base de datos.

Tiene una opción para generar código fuente o para construir bases de datos. Soporte para crear bases de datos para Servidores SQL; y otra, para incluir código SQL y verificar la creación de objetos. Además de la opción para incluir encabezados de comentarios.

Ingeniería hacia adelante

Una vez que se ha diseñado la base de datos, se pude construir o generar código fuente para todo o para parte de los diseños de la base de datos. Propiamente hace la secuencia de la creación de tipos de objetos diferentes para asegurar eficiencia, y construir bases de datos libres de errores.

Actualiza una base de datos del diagrama. ER/Studio permite aplicar cambios de diseño del modelo de datos directamente a la base de datos. Cuando se comparan las diferencias entre los dos, formula una estrategia de alteración inteligente que implementa el diseño de las modificaciones mientras se preserva la tabla con los datos existentes, privilegios de objetos, y dependencias en la base de datos.

Ingeniería de reverso

Cuenta con ingeniería de reverso, cuando necesite iniciar un trabajo de una base de datos existente, ER/Studio puede hacer una ingeniería de reverso al esquema completo para

cualquier plataforma de bases de datos. La operación de la ingeniería de reverso extrae eficientemente definiciones de objetos y construye un modelo de datos gráfico.

Sincronización de la Base de Datos

Sincronización entre el diagrama físico y el lógico. Mezcla entre cualquier par de diagramas físicos para la misma plataforma de bases de datos. Comparación lado-a-lado de las diferencias. El usuario puede decidir qué diferencias mezclar o ignorar.

Objetos reusables. Construir atributos reusables. Aplicarlos a atributos y columnas. Propagación global de actualizaciones. Construir tipos de datos personalizables.

Submodelado. Crear cualquier número de subvistas personalizables sobre un diagrama físico o lógico. Cualquier objeto puede existir en cualquier número de subvistas (relaciones de muchos a muchos entre objetos y subvistas).

Crear rápidamente subvistas eligiendo un área del diagrama.

Control independiente sobre el despliegue de la subvista, incluyendo posición del objeto, colores y letras.

Utilidad de búsqueda rápida. Editores en tabla para evitar la necesidad de poner en modo cascada los diálogos.

Diferenciación de color de llaves primarias y secundarias inherentes. Sombreado de cajas de la entidad.

Uso de la Herramienta Case ER Studio 6.

Partiremos del hecho que se tiene una base de datos instalada, para nuestro estudio nuestra base de datos estará implementada en Sql Server 2000 y se llamará dbame.

Para realizar la ingeniería inversa a la base de datos antes expuesta se procede de la siguiente forma:

Damos clic en File -> New

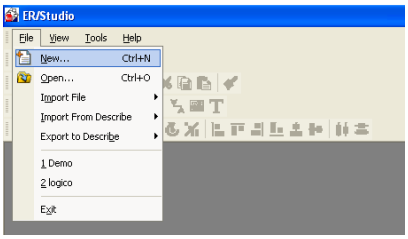


Figura 22. Nuevo Proyecto ER/Studio

A partir de esto se nos mostrará un pequeño submenú con tres opciones, para lo cual se deberá escoger Reverse-engineer an existing database, y pulsamos en Login…

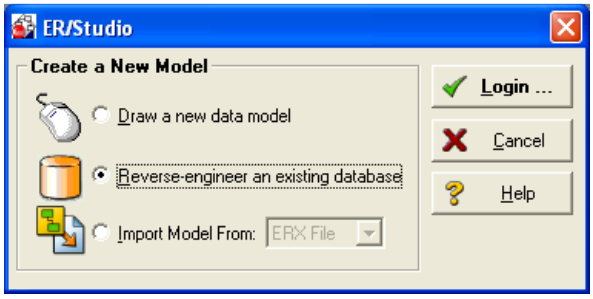


Figura 23. Opciones para crear un nuevo Proyecto ER/Studio

Con el siguiente asistente, se debe crear un nuevo ODBC (*conectividad abierta de bases de datos*), para de esta manera referenciar a nuestra base de datos a ser analizada, con sus tablas, claves, vistas, etc. Damos clic en Agregar…

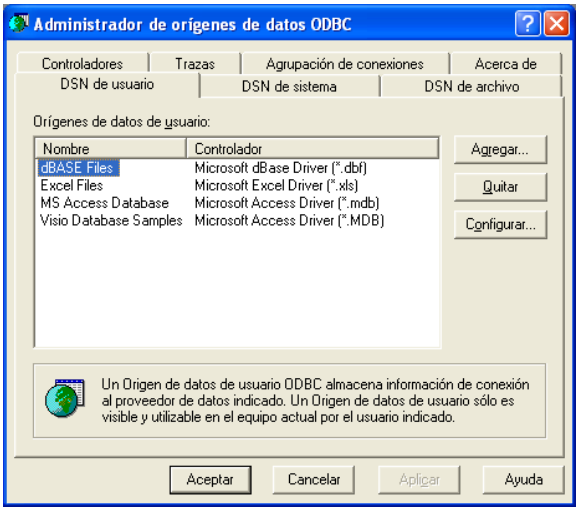


Figura 24. Administrador de Orígenes de ODBC

Para nuestro caso escogeremos como origen de datos a SQL Server y damos clic en Finalizar.

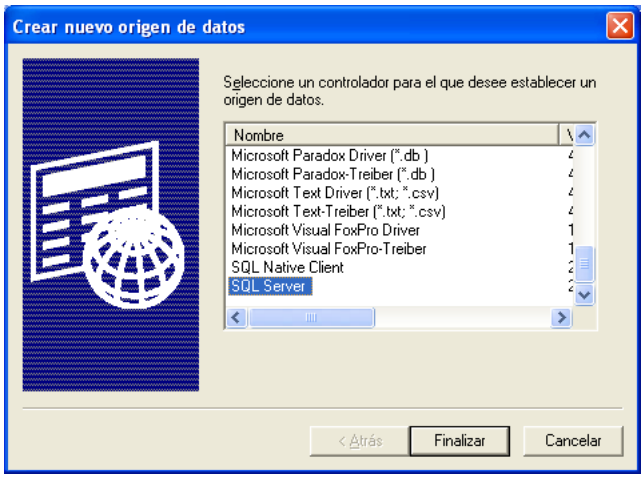


Figura 25. Controlador para establecer un origen de Datos

El siguiente asistente nos ayudará a crear un origen de datos ODBC que podrá ser usado para conectarnos de una forma instantánea a SQL Server.

Para lo cual debemos colocar el nombre del origen de datos, una descripción corta y el nombre del servidor al que se va a conectar, para el efecto (local). Y damos clic en Siguiente >.

Figura 26. Creación de un Origen de Datos

En nuestra siguiente pantalla nos autenticamos con SQL Server mediante el Id de inicio de sesión y una contraseña escritos por el usuario.

Y marcamos a la vez Conectar con SQL Server para obtener la configuración predeterminada de las opciones de configuración adicionales

Ahora en el inicio de sesión colocamos el nombre del usuario con el que accedemos a SQL Server, “sa” para nuestro caso; y la contraseña si la tenemos.

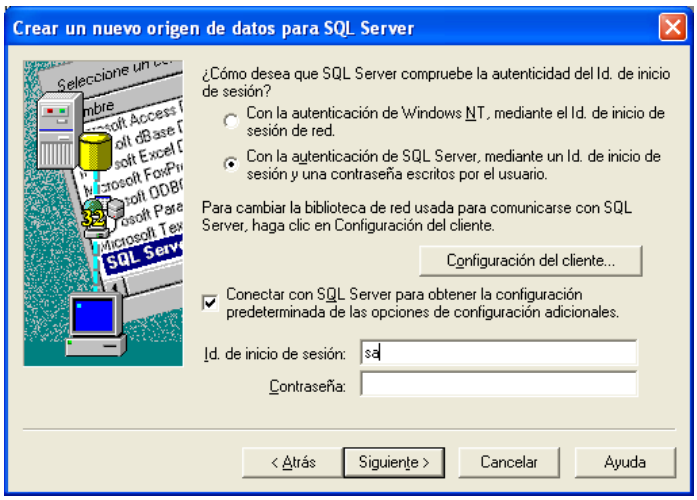


Figura 27. Autentificación de Usuario para la creación de un ODBC

Nos queda por establecer la base de datos de la cual se realizará la reingeniería.

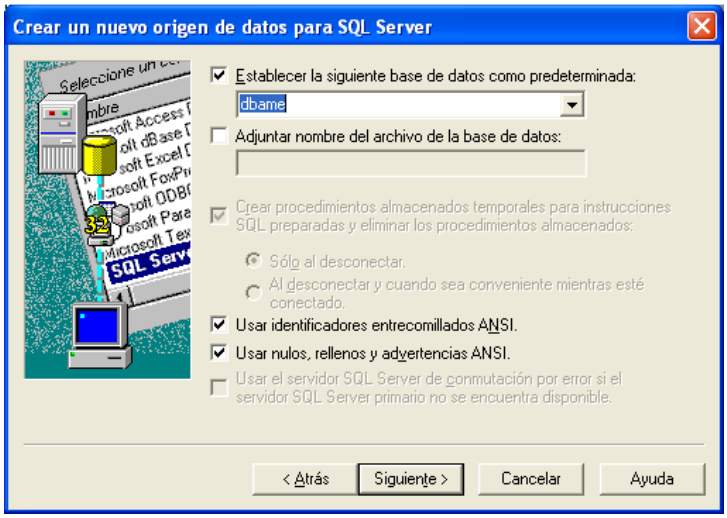


Figura 28. Establecimiento de la Base de Datos Predeterminada.

Si todo marcha bien, entonces se podrá observar el nombre de la nueva referencia ODBC para nuestra conexión.

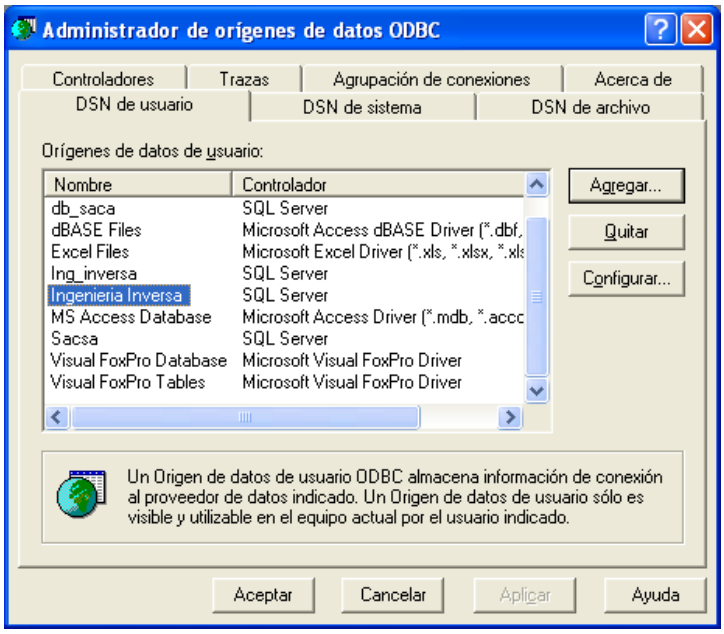


Figura 29. Administrador de Orígenes de Datos ODBC

Luego de establecer la conexión es hora de hacer uso de nuestro ODBC. Que para el efecto se llamó Ingeniería Inversa.

Además colocaremos el nombre del usuario de la Base de Datos y el Password, claro, si este existe.

Figura 30. Wizard para la realización de Ingeniería Inversa

Escogemos los objetos a ser evaluados en este caso las vistas y las tablas del usuario.

Figura 31. Wizard para la determinar la inclusión de Tablas y Vistas

Figura 32. Wizard para la inclusión de las Tablas de Base de Datos dbame.

Figura 33. Wizard para la determinación del tipo de esquema de representación.

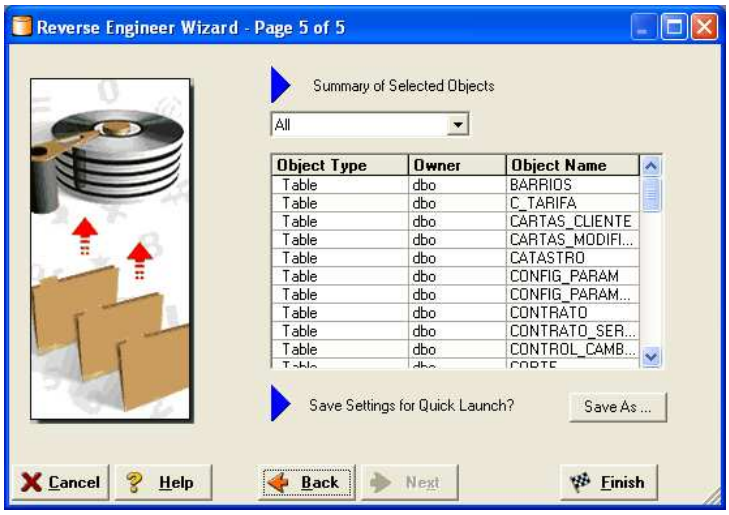


Figura 34. Wizard. Resumen de los Objetos seleccionados

Generación del Reporte.

Para el efecto, se generará un reporte en formato HTML, para una mejor manipulación de nuestros datos generados.

Para generar el reporte realizamos los siguientes pasos:

Damos clic en Tools, luego en Generate Reports.

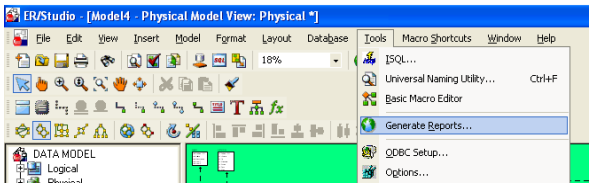


Figura 35. Generar Nuevo Reporte ER/Studio

Enseguida se nos muestra la siguiente pantalla de ayuda, en la cual se debe seleccionar el tipo de reporte, para nuestro caso, HTML report, y además el directorio en donde se guardará el reporte.

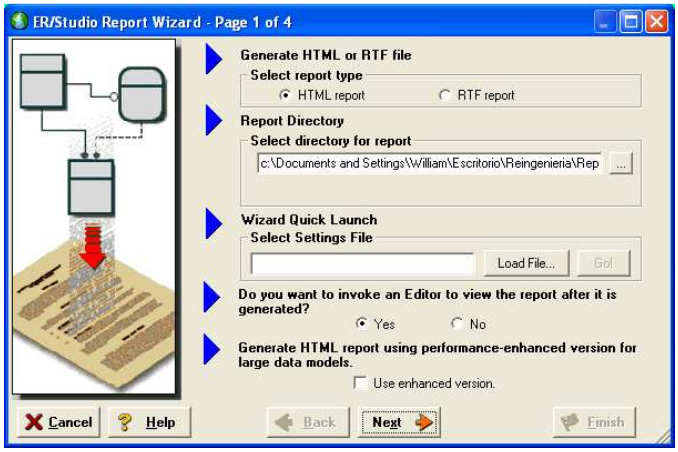


Figura 36. Especificaciones para el Nuevo Reporte

En esta página, seleccionaremos los objetos de nuestro modelo de datos, que queremos que se generen en nuestro reporte, tanto para el diagrama como para el diccionario de datos.

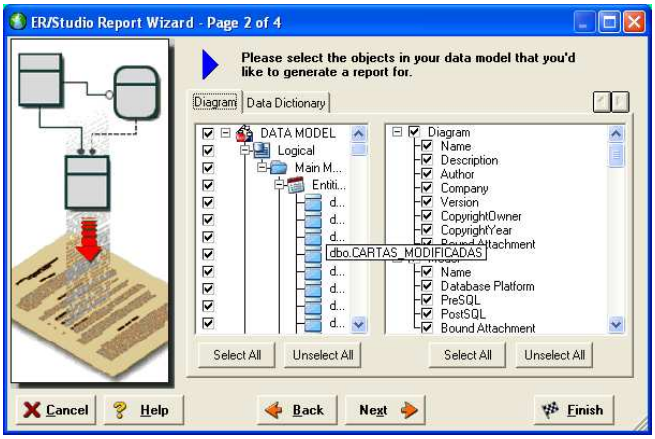


Figura 37. Selección de Objetos para el Reporte

De igual manera se seleccionará los objetos para el Diccionario de Datos.

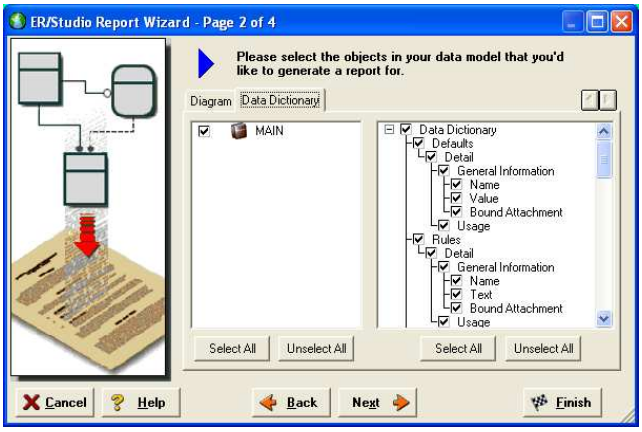


Figura 38. Selección de Objetos para el Diccionario de Datos

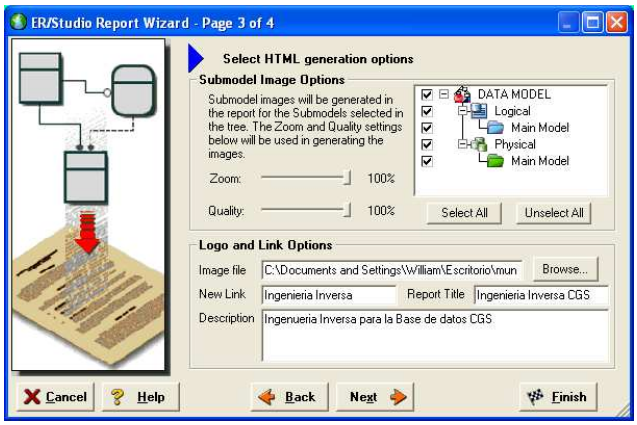


Figura 39. Selección de Objetos opcionales para el Reporte

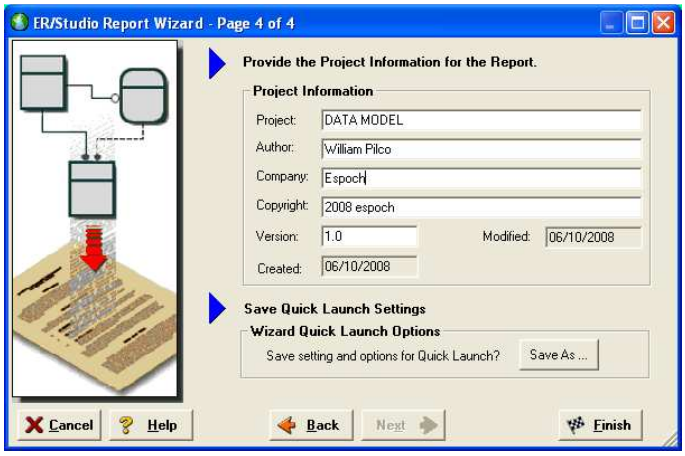


Figura 40. Inclusión de la Información para el Proyecto

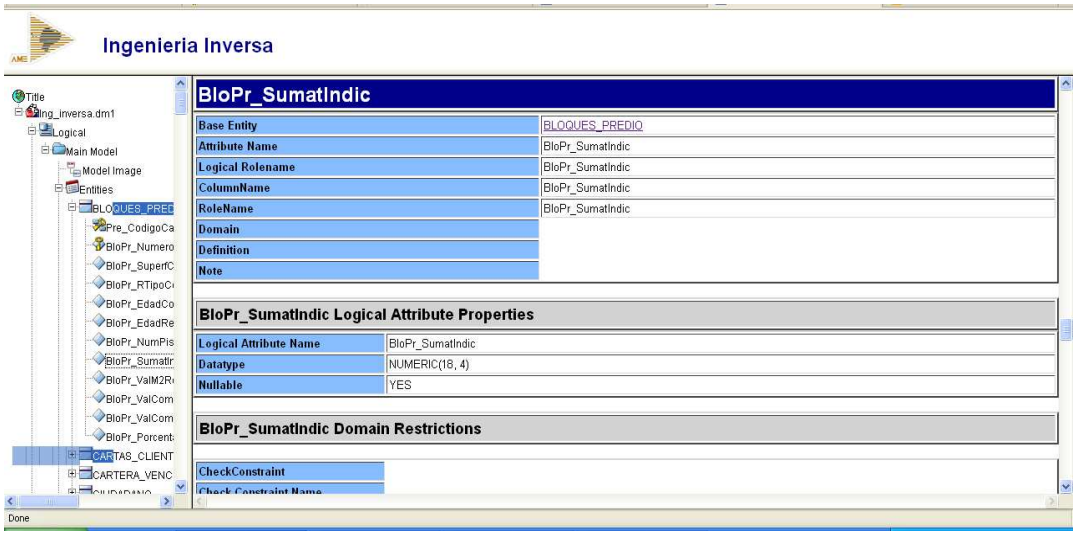


Figura 41. Reporte Html.

Como parte complementaria se realizará una consulta en nuestra Base de Datos para determinar los nombres y tipos de tablas que contiene la misma.

Lo primero que haremos será obtener todas las tablas que componen la base de datos. Para realizar esto debemos efectuar una consulta SQL.

En dicha consulta obtendremos los nombres de las tablas, su tipo, si es tabla o vista, el esquema. La consulta SQL que se utiliza es la siguiente:

```
SELECT    *
FROM      Information_Schema.Tables
```

De esta manera se podrán filtrar los resultados; si queremos seleccionar solo tablas o vistas.

```
SELECT    *
FROM      Information_Schema.Tables
WHERE     (Table_Type = 'BASE TABLE ')
```

```
SELECT    *
```

```
FROM        Information_Schema.Tables

WHERE     (Table_Type = 'View ')
```

El resultado de dicha consulta será el siguiente:

TABLE_CATALOG	TABLE_SCHEMA	TABLE_NAME	TABLE_TYPE
Dbame	Dbo	actfi_det_asignacion	BASE TABLE
Dbame	Dbo	actfi_det_tfisica	BASE TABLE
Dbame	Dbo	actfi_revalorizacion	BASE TABLE
Dbame	Dbo	activo_fijo_asignacion	BASE TABLE
Dbame	Dbo	ALICUOTA_PREDIO	BASE TABLE
Dbame	Dbo	Banco	BASE TABLE
Dbame	Dbo	BARRIOS	BASE TABLE
Dbame	Dbo	bcn_conciliacion_detalle	BASE TABLE
Dbame	Dbo	bco_cta_cheque	BASE TABLE
Dbame	Dbo	bco_cuenta	BASE TABLE
Dbame	Dbo	BLOQUES_PREDIO	BASE TABLE
Dbame	Dbo	bnc_conciliacion	BASE TABLE
Dbame	Dbo	C_TARIFA	BASE TABLE
Dbame	Dbo	CARTAS_CLIENTE	BASE TABLE
Dbame	Dbo	CARTAS_MODIFICADAS	BASE TABLE
Dbame	Dbo	CARTERA_VENCIDA	BASE TABLE
Dbame	Dbo	CATASTRO	BASE TABLE
Dbame	Dbo	Cierre	BASE TABLE
Dbame	Dbo	CIUDADANO	BASE TABLE
Dbame	Dbo	CLASE_TIERRAS	BASE TABLE
Dbame	Dbo	co_auditoria	BASE TABLE
Dbame	Dbo	co_cargo	BASE TABLE
Dbame	Dbo	co_cargo_departamento	BASE TABLE
Dbame	Dbo	co_comprobante_rent_detalle	BASE TABLE
Dbame	Dbo	co_comprobante_rentecion	BASE TABLE
Dbame	Dbo	co_conta_presu	BASE TABLE
Dbame	Dbo	co_cuenta_contable	BASE TABLE
Dbame	Dbo	co_cuenta_presupuesto	BASE TABLE
Dbame	Dbo	co_cuentas_banco	BASE TABLE
Dbame	Dbo	co_formas	BASE TABLE
Dbame	Dbo	co_fun_asociacion	BASE TABLE
Dbame	Dbo	co_fun_catedra	BASE TABLE
Dbame	Dbo	co_fun_codigo	BASE TABLE
Dbame	Dbo	co_fun_curso	BASE TABLE
Dbame	Dbo	co_fun_educacion	BASE TABLE
Dbame	Dbo	co_fun_estadistica	BASE TABLE
Dbame	Dbo	co_fun_evalua	BASE TABLE
Dbame	Dbo	co_fun_experiencia	BASE TABLE
Dbame	Dbo	co_fun_familia	BASE TABLE
Dbame	Dbo	co_fun_idioma	BASE TABLE

Dbame	Dbo	co_fun_imagen	BASE TABLE
Dbame	Dbo	co_fun_pais	BASE TABLE
Dbame	Dbo	co_fun_partida	BASE TABLE
Dbame	Dbo	co_fun_pasatiempo	BASE TABLE
Dbame	Dbo	co_fun_publicacion	BASE TABLE
Dbame	Dbo	co_fun_registro	BASE TABLE
Dbame	Dbo	co_fun_rol	BASE TABLE
Dbame	Dbo	co_fun_rol_detallle	BASE TABLE
Dbame	Dbo	co_fun_rubro_rol	BASE TABLE
Dbame	Dbo	co_fun_servicio	BASE TABLE
Dbame	Dbo	co_fun_tipo_rol	BASE TABLE
Dbame	Dbo	co_funcion_pre	BASE TABLE
Dbame	Dbo	co_gru_forma	BASE TABLE
Dbame	Dbo	co_grupos	BASE TABLE
Dbame	Dbo	co_nivel	BASE TABLE
Dbame	Dbo	co_normativa	BASE TABLE
Dbame	Dbo	co_pro_registro	BASE TABLE
Dbame	Dbo	co_rentecion_iva	BASE TABLE
Dbame	Dbo	co_rentecion_varias	BASE TABLE
Dbame	Dbo	co_tipo_cuenta	BASE TABLE
Dbame	Dbo	co_tipo_funcionario	BASE TABLE
Dbame	Dbo	co_tra_detalle	BASE TABLE
Dbame	Dbo	co_transaccion	BASE TABLE
Dbame	Dbo	co_usuario	BASE TABLE
Dbame	Dbo	Colaborador	BASE TABLE
Dbame	Dbo	CONDICIONES_IMP_ADICIONAL	BASE TABLE
Dbame	Dbo	CONFIG_PARAM	BASE TABLE
Dbame	Dbo	CONFIG_PARAM_AÑO	BASE TABLE
Dbame	Dbo	CONSERV_CULTIVOS	BASE TABLE
Dbame	Dbo	CONSERV_OBRAS_INTE	BASE TABLE
Dbame	Dbo	CONTRATO	BASE TABLE
Dbame	Dbo	CONTRATO_SERVICIO	BASE TABLE
Dbame	Dbo	Contribuyentes	BASE TABLE
Dbame	Dbo	CONTROL_CAMBIOS	BASE TABLE
Dbame	Dbo	CORTE	BASE TABLE
Dbame	Dbo	CULTIVOS_INVER_PREDIO	BASE TABLE
Dbame	Dbo	datos_municipio	BASE TABLE
Dbame	Dbo	Departamento	BASE TABLE
Dbame	Dbo	DESC_EDIF_BLOQUE	BASE TABLE
Dbame	Dbo	DESC_EDIFICACION	BASE TABLE
Dbame	Dbo	DESC_IMPUESTOS	BASE TABLE
Dbame	Dbo	DESC_INVERSIONES_PREDIO	BASE TABLE
Dbame	Dbo	DESC_TERRENO	BASE TABLE
Dbame	Dbo	DESC_TERRENO_PREDIO	BASE TABLE
Dbame	Dbo	DESMARCAR_TITULOS	BASE TABLE
Dbame	Dbo	DETALLE_PAGOS_SERVICIOS	BASE TABLE
Dbame	Dbo	DIVISION_POLITICA	BASE TABLE
Dbame	Dbo	division_politica_ctab	BASE TABLE
Dbame	Dbo	Dtproperties	BASE TABLE
Dbame	Dbo	FACTOR_SUPERFICIE	BASE TABLE

Dbame	Dbo	FACTURACION_CLIENTE	BASE TABLE
Dbame	Dbo	FICHA_MUNICIPAL	BASE TABLE
Dbame	Dbo	FORESTALES	BASE TABLE
Dbame	Dbo	FRENTE_FONDO	BASE TABLE
Dbame	Dbo	Func	BASE TABLE
Dbame	Dbo	Funcionario	BASE TABLE
Dbame	Dbo	HISTORIAL	BASE TABLE
Dbame	Dbo	INFRA_SERV	BASE TABLE
Dbame	Dbo	INFRA_SERV_PREDIO	BASE TABLE
Dbame	Dbo	INSTITUCION	BASE TABLE
Dbame	Dbo	INTERES_FINANCIAMIENTO	BASE TABLE
Dbame	Dbo	INTERES_MENSUAL	BASE TABLE
Dbame	Dbo	INTERES_MORA	BASE TABLE
Dbame	Dbo	inv_activo_fijo	BASE TABLE
Dbame	Dbo	inv_activo_fijo_ac	BASE TABLE
Dbame	Dbo	inv_det_orden_cc	BASE TABLE
Dbame	Dbo	inv_det_orden_cp	BASE TABLE
Dbame	Dbo	inv_det_orden_cv	BASE TABLE
Dbame	Dbo	inv_det_tfisica	BASE TABLE
Dbame	Dbo	inv_inventario	BASE TABLE
Dbame	Dbo	inv_orden_cc	BASE TABLE
Dbame	Dbo	inv_orden_cp	BASE TABLE
Dbame	Dbo	inv_orden_cv	BASE TABLE
Dbame	Dbo	inv_toma_fisica	BASE TABLE
Dbame	Dbo	inv_unidMedida	BASE TABLE
Dbame	Dbo	LECTURAS	BASE TABLE
Dbame	Dbo	LOCALIZACION_PREDIO	BASE TABLE
Dbame	Dbo	matriz_cabeza	BASE TABLE
Dbame	Dbo	matriz_detalle	BASE TABLE
Dbame	Dbo	MEDIDOR	BASE TABLE
Dbame	Dbo	MENU_USUARIO	BASE TABLE
Dbame	Dbo	MES_HISTORIAL	BASE TABLE
Dbame	Dbo	OBRAS_INT_INVER_PREDIO	BASE TABLE
Dbame	Dbo	Organización	BASE TABLE
Dbame	Dbo	PARAM_CUENTA	BASE TABLE
Dbame	Dbo	PARAMETROS_DETERM_PREDIO	BASE TABLE
Dbame	Dbo	PARAMETROS_DETERMINACION	BASE TABLE
Dbame	Dbo	PARAMETROS_GENERALES	BASE TABLE
Dbame	Dbo	parte_diario	BASE TABLE
Dbame	Dbo	parte_resumen	BASE TABLE
Dbame	Dbo	ped_det_cv	BASE TABLE
Dbame	Dbo	pedido_cv	BASE TABLE
Dbame	Dbo	PERIODO_CATASTRAL	BASE TABLE
Dbame	Dbo	pg_concepto	BASE TABLE
Dbame	Dbo	pg_trans_general	BASE TABLE
Dbame	Dbo	pg_transfer_cab	BASE TABLE
Dbame	Dbo	pg_transferencia	BASE TABLE
Dbame	Dbo	PISTA_AUDITORIA	BASE TABLE
Dbame	Dbo	pre_cert_detalle	BASE TABLE
Dbame	Dbo	pre_certificacion	BASE TABLE

Dbame	Dbo	Pre_Con_Beneficiario	BASE TABLE
Dbame	Dbo	Pre_Con_cabecera	BASE TABLE
Dbame	Dbo	Pre_Con_detalle	BASE TABLE
Dbame	Dbo	pre_reforma	BASE TABLE
Dbame	Dbo	pre_reforma_detalle	BASE TABLE
Dbame	Dbo	PRECIO_BASE	BASE TABLE
Dbame	Dbo	PREDIO	BASE TABLE
Dbame	Dbo	PRESTAMOS_HIPOTECARIOS	BASE TABLE
Dbame	Dbo	PROPIETARIO	BASE TABLE
Dbame	Dbo	Proveedor	BASE TABLE
Dbame	Dbo	proy_det_garantia	BASE TABLE
Dbame	Dbo	proy_garantia	BASE TABLE
Dbame	Dbo	proy_monto	BASE TABLE
Dbame	Dbo	Proyecto	BASE TABLE
Dbame	Dbo	RECAUDACION	BASE TABLE
Dbame	Dbo	RECAUDACION_CLIENTE	BASE TABLE
Dbame	Dbo	RECONEXION	BASE TABLE
Dbame	Dbo	Rubro	BASE TABLE
Dbame	Dbo	SALDO	BASE TABLE
Dbame	Dbo	SALDO_ANTERIOR	BASE TABLE
Dbame	Dbo	SERVICIOS	BASE TABLE
Dbame	Dbo	SOLICITUD_CAMBIO_CLIENTE	BASE TABLE
Dbame	Dbo	SOLO_TITULOS	VIEW
Dbame	Dbo	SUPERFICIE_RURAL	BASE TABLE
Dbame	Dbo	Sysconstraints	VIEW
Dbame	Dbo	Syssegments	VIEW
Dbame	Dbo	Tarea	BASE TABLE
Dbame	Dbo	tarea_verificacion	BASE TABLE
Dbame	Dbo	TERRENO_RURAL_PREDIO	BASE TABLE
Dbame	Dbo	TIPO_CULTIVO	BASE TABLE
Dbame	Dbo	TIPO_OBRAS_INTER	BASE TABLE
Dbame	Dbo	TIPO_ORGANIZACION	BASE TABLE
Dbame	Dbo	TITULOS	BASE TABLE
Dbame	Dbo	TITULOS_PAGADOS	BASE TABLE
Dbame	Dbo	TITULOS_PREDIO	BASE TABLE
Dbame	Dbo	TRAS_DOMINIO	BASE TABLE
Dbame	Dbo	TRAS_DOMINIO_PREDIO	BASE TABLE
Dbame	Dbo	UBICACIÓN	BASE TABLE
Dbame	Dbo	USO_CONEXION	BASE TABLE
Dbame	Dbo	USO_SUELO	BASE TABLE
Dbame	Dbo	USO_SUELO_PREDIO	BASE TABLE
Dbame	Dbo	USO_SUELO_RURAL	BASE TABLE
Dbame	Dbo	USUARIO	BASE TABLE
Dbame	Dbo	USUARIO_GCS	BASE TABLE
Dbame	Dbo	VALOR_APLIC_EDIFICACION	BASE TABLE
Dbame	Dbo	VALOR_APLIC_RURAL	BASE TABLE
Dbame	Dbo	VALOR_MES	BASE TABLE
Dbame	Dbo	VALOR_TABLA_PROG_DETER	BASE TABLE
Dbame	Dbo	VALORES_CLIENTE	BASE TABLE
Dbame	Dbo	VALORES_PREDIO	BASE TABLE

Dbame	Dbo	VENTA_MEDIDOR	BASE TABLE
Dbame	Dbo	view_cliente_conex	VIEW
Dbame	Dbo	view_cliente_conex_ValLec	VIEW
Dbame	Dbo	view_cliente_conex_ValLec_Pendiente	VIEW
Dbame	Dbo	view_cliente_usoconex	VIEW
Dbame	Dbo	view_clientereal_conex	VIEW
Dbame	Dbo	view_config_param	VIEW
Dbame	Dbo	view_contrato_saldo	VIEW
Dbame	Dbo	view_contrato_saldo_cliente	VIEW
Dbame	Dbo	view_verif_cartas	VIEW
Dbame	Dbo	VISTA_CARTERA	VIEW
Dbame	Dbo	VISTA_CARTERA_N_J	VIEW
Dbame	Dbo	VISTA_DESCEDIF	VIEW
Dbame	Dbo	VISTA_EX_RECARGOS_AFECT	VIEW
Dbame	Dbo	VISTA_FORESTAL	VIEW
Dbame	Dbo	VISTA_INFRA_ESTRUC_VALBLOQ	VIEW
Dbame	Dbo	VISTA_PLANTACION	VIEW
Dbame	Dbo	VISTA_PREDIO_RURAL	VIEW
Dbame	Dbo	VISTA_PredioBDD	VIEW
Dbame	Dbo	VISTA_PREDIOS_N_J	VIEW
Dbame	Dbo	VISTA_TERR_RURAL	VIEW
Dbame	Dbo	VISTA_TITULOS	VIEW
Dbame	Dbo	VISTA_VALORES	VIEW
Dbame	Dbo	Vista1	VIEW
Dbame	Dbo	Vista2	VIEW
Dbame	dbo	ZONA_HOMOGENEA	BASE TABLE

Tabla 10 .Tablas que componen la base de Datos dbame

Por cada fila habrá cuatro columnas. Las columnas significan lo siguiente: Nombre de la Base de Datos, el Propietario, Nombre de la Tabla, y el Tipo de la Tabla, cada tabla tendrá tantas filas en el resultado de la consulta como atributos posea.

4.1.1.2 Etapa2: Extracción Acumulativa

De acuerdo con la información obtenida por parte de los usuarios de la Base de Datos en cuestión se ha llegado a sacar los siguientes resultados.

4.1.1.2.1 Análisis de Nombres

Se hará una descripción de todas las posibles tablas y campos con mayor duda para su interpretación. Además de las tablas más importantes e influyentes para el correcto funcionamiento de la aplicación para esto se apoyará de la información aportada por el usuario.

Tabla Catastro

Se podría decir que se trata de la tabla más importante de la cual sale la mayoría de relaciones con las demás tablas, además de contener los datos personales de los clientes como sus nombres, direcciones, además de su ubicación.
Además contiene las claves foráneas para poder conectarnos con otras tablas como Cartas Cliente, Lecturas, Uso Conexión, entre las más importantes para el desarrollo de la aplicación móvil para el cálculo del valor de consumo del Agua Potable, Alcantarillado y Recolección de Basura.

Su llave primaria es Cli_NumCuenta, en la cual se registra el número de cuenta del cliente, está compuesta por 8 dígitos y sirve para relacionar con la tabla Contrato, Cartas Cliente, Lecturas, Uso Conexión, entre otras.
Entre los campos principales de esta tabla se tiene:

- **Cat_tipoCli**: en la cual se describe el tipo de cliente "1- REAL".
- **Cat_NombresCli**: Describe los nombres del cliente.
- **Cat_ApellidosCli**: Describe los apellidos del cliente.
- **Cat_CodLog:** El código de catastro, según lo impuesto por las municipalidades, compuesta por 14 dígitos, distribuidos de la siguiente manera:
    - **Cat_SectorLog:** Se establece el sector (2 dígitos).
    - **Cat_RutaLog:** Se establece la ruta(2 dígitos).
    - **Cat_ManzanaLog:** Se establece la manzana (2 dígitos).

  - **Cat_secuenciaLog:** Se establece la secuencia (4 dígitos).
  - **Cat_PisoLog:** Se establece el piso (2 dígitos).
  - **Cat_DptoLog:** Se establece el departamento (2 dígitos).

- **Cat_UbicacionMed:** Se Detalla sobre el tipo de ubicación en la que se encuentra el medidor (0-No tiene; 1-Buena)
- **Cat_Estado_Med:** Para conocer el estado actual del medidor 0- SIN MEDIDOR 1- FUNCIONANDO

Tabla Cartas_Cliente

En esta tabla se almacenan tanto los pagos como las deudas anteriores de los clientes para los servicios por consumo de agua potable, alcantarillado y recolección de basura.

Cada tupla cuenta con un registro del mes y año de la transacción.

Esta tabla se conecta a la tabla Catastro mediante el campo Cli_NumCuenta.
Se conecta además con la tabla DESMARCAR_TITULOS, mediante los campos: Cli_NumCuenta, CarCl_Año, CarCl_Tipo, CarCl_Mes.

Entre los campos destinados para el cobro de saldos estarían considerados los siguientes:

- **CarCl_ValBasura**: Para registrar el monto por pagar para el cobro de la basura.
- **CarCl_Otros:** Destinado al cobro de otros rubros.
- **CarCl_Totaliq:** Aquí se almacena la suma total de todos los rubros por pagar.
- **CarCl_ValAdmin:** Se almacenan los valores por gastos administrativos.
- **CarCl_Tipo:** Con este campo podemos saber si el cliente ha pagado la deuda (P), aun esta se encuentra por pagar (S).

- **CarCl_Consumo:** En este campo se almacenará el valor del consumo del agua potable en metros cúbicos.
- **CarCl_ValConsumo:** En este campo se almacenarán los valores a pagar por el consumo del agua potable.
- **CarCl_ValBasura:** En este campo se almacenarán los valores a pagar por la recolección de basura.
- **CarCl_ValAlcanta:** En este campo se almacenarán los valores a pagar por el servicio del alcantarillado.

Tabla Lecturas.

Para la consecución de nuestro proyecto la Tabla Lecturas es la tabla más importante y de mayor uso y trascendencia, ya que aquí se almacenarán los datos de la lectura realizada para cada medidor, en donde se registraran la fecha, observaciones, el valor leído para cada caso.

Entre los campos más importantes de esta tabla tenemos:

- **Lec_Año:** Se almacena el año de la lectura a ser procesada.
- **Lec_Mes:** Se almacena el mes de la lectura a ser procesada.
- **Lec_FecLec:** Se almacena la fecha en laque se realizó la lectura.
- **Lec_ValLec:** En este campo se almacenará el valor de la lectura actual.
- **Lec_Consumo:** Aquí se registra el consumo total del agua potable que viene dado por la resta de la lectura actual menos la lectura anterior.
- **Lec_Obs:** Se almacenan las observaciones para cada lectura.

Tabla Uso_Conexión.

En esta tabla se almacenan los tipos de conexiones las mismas que estarán disponibles para cada usuario y de esto dependerá el valor a pagar por el consumo del agua potable. Entre los

tipos más importantes de conexiones se tiene, conexiones Domésticas, Comerciales, Especiales, Tercera Edad, etc.

Entre los campos más importantes de esta tabla se tiene:

- **UsoCo_Desc:** Es en este campo en donde se almacenarán los tipos de conexiones antes citados.
- **UsoCo_Tipo:** Se almacenarán los tipos de servicio para cada tipo de conexión, (Agua, Alcantarillado).

Tabla C_Tarifa.

La Tabla C_Tarifa se encuentra entre las más importantes ya que es aquí en donde se configurarán los tipos de tarifa para el pago por el consumo del agua potable. Con esta tabla se realizarán los diferentes tipos de cálculos para sacar los valores a cancelar por el consumo del agua potable.
Los datos son almacenados por una serie de rangos para cada tipo de uso de conexión y para cada año.

Entre los campos más importantes de esta tabla se tiene:

- **CTar_UsoConex:** Este campo va relacionado con el campo UsoCo_Desc de la tabla Uso_Conexion para describir los tipos de conexión pertenecientes para cada usuario.
- **CTar_Tipo:** Se Trata de la clasificación del tipo de tarifas (Básica, Rangos y Mixtas), cada una con una interpretación de cálculo distintas. Para cada uno de estos tipos de tarifas existe un rango de valores con su respectivo monto de tarifa y con su respectivo excedente.
- **CTar_ValMin:** Este campo almacena el valor mínimo para un rango de consumo.
- **CTar_ValMin:** Este campo almacena el valor máximo para un rango de consumo.
- **CTar_Tarifa:** Se almacena aquí el costo por el consumo del rango consumido.

- **CTar_Excede:** Este campo almacenará un valor por exceso de consumo, es decir por sobrepasar al rango de consumo, establecido por el valor mínimo (CTar_ValMin) y el valor máximo (CTar_ValMax).
- **CTar_Desc:** Se describe el tipo de servicio a ser calculado (Agua, Alcantarillado).

Tabla Config_Param

Esta Tabla es de vital importancia para la realización del cálculo de los valores a pagar por el consumo de los servicios de agua potable, basura y alcantarillado, pues aquí se encuentran todos los valores de configuración (de cobro) como: Valores administrativos, otros pagos, multas, mese de corte, mese de mora, entre otros para un determinado año.

Además por medio de esta tabla se sabrá si el valor a ser pagado por el consumo de los servicios de agua potable, basura y alcantarillado viene dado por medio de un cálculo por valores fijos o por valores diferenciados, dependiendo de cada configuración anual.

Entre los campos de mayor trascendencia de esta tabla se encuentran:

- **ConPa_AñoProce:** Se trata del año en proceso, es decir el año que va a contener todos los parámetros de configuración.
- **ConPa_Adm:** Valor de cobro por concepto de manejos administrativos.
- **ConPa_SisCo:** Por medio de este campo se obtendrán dos valores (1-2), por medio de estos se sabrá:
    - **1:** Sistema de cobro con valor fijo, en este caso obligatoriamente se tendrá que coger tres campos (Alcantarillado1, Basura1, Medidor1), de la tabla Config_Param_Año, directamente para el cálculo de los servicios de agua potable, alcantarillado y basura.
    - **2:** Sistema de cobro con valor diferenciado.

- **ConPa_TipoAlc y ConPa_TipoAg:** Estos dos valores funcionan en conjunto, pues por medio de estos se pueden determinar los valores a cancelar para el alcantarillado y la

basura, siempre y cuando se esté calculando sistemas de cobro con valor diferenciado. Estos campos pueden obtener los valores de 1, 2, 3 y 4.

Por medio de estos valores se obtendrán los valores siguientes:

Alcantarillado	Basura	ConPa_SisCo	ConPa_TipoAlc	ConPa_TipoAg
Porcentaje	Porcentaje	2	1	1
Porcentaje	Estructura Tarifaria	2	1	4
Porcentaje	Valor Fijo	2	1	2
Porcentaje	V. Fijo Diferenciado	2	1	3
Estructura Tarifaria	Porcentaje	2	4	1
Estructura Tarifaria	Estructura Tarifaria	2	4	4
Estructura Tarifaria	Valor Fijo	2	4	2
Estructura Tarifaria	V. Fijo Diferenciado	2	4	3
Valor Fijo	Porcentaje	2	2	1
Valor Fijo	Estructura Tarifaria	2	2	4
Valor Fijo	Valor Fijo	2	2	2
Valor Fijo	V. Fijo Diferenciado	2	2	3
V. Fijo Diferenciado	Porcentaje	2	3	1
V. Fijo Diferenciado	Estructura Tarifaria	2	3	4
V. Fijo Diferenciado	Valor Fijo	2	3	2
V. Fijo Diferenciado	V. Fijo Diferenciado	2	3	3

Tabla 11. Combinaciones para ConPa_TipoAlc y ConPa_TipoAg con Sistemas Diferenciados

- **ConPa_ValorFijoAg, ConPa_ValorFijoAlc, ConPa_ValorFijoBas:** Estos tres campos funcionan en conjunto puesto que si ConPa_SisCo tiene el valor de 1, estos tendrán la opción de tener valores distintos de cero, encaso de que los tres campos tengan valor igual acero se procederá a tomar los valores de la tabla Config_Param_Año.

Tabla Config_Param_Año

Se trata de una Tabla reciclable ya que sirve para el almacenamiento de información de varias fuentes, esto para cada año.
Se consideran los servicios de Alcantarillado, Basura y Medidor, este último sirve para el almacenamiento de valores en caso de medidores dañados. Y por tratarse de una tabla que trabaja siempre y cuando exista un sistema de valor diferenciado se tendrán campos para diferenciar la categoría de la conexión, en este caso Domesticas, Comerciales, Industriales, Oficiales, Tercera Edad y Otras.

Tabla Contrato

En esta tabla se guardarán los datos concernientes a la compra de un determinado medidor, tanto en la compra al contado como en la compra a crédito, guardándose para esta última los meses de deuda, el saldo restante a ser cancelado, el responsable de la venta, entre otros.

Tabla Saldo

En esta tabla se registrarán los pagos para cuando un cliente adquiere un medidor a crédito, almacenándose aquí el número de cuotas pendientes, el saldo actual, entre otros.

4.1.1.2.2 Extracción de Claves Externas

Una Clave es un conjunto de campos cuyos valores determinan unívocamente a cada registro de la relación. Dicho conjunto de campos debe ser mínimo, esto es, ningún subconjunto propio de la clave puede actuar también como clave.

Clave Candidata: Cada uno de los campos o combinaciones de campos que pueden actuar como clave de la relación.

Clave Primaria (PK=Primary Key): Clave candidata elegida por el diseñador de la BD para la relación.

Clave Externa o Secundaria (FK=Foreign Key): Campo o combinación de campos de una relación que funciona como clave primaria de otra relación de la BD (*relación referenciada* o *relación padre* para la clave ajena).

1.- Las claves externas son esenciales en el Modelo Relacional, ya que permiten enlazar tablas de la BD.
2.- Una clave externa y la clave primaria de la relación referenciada asociada han de estar definidas sobre los mismos *dominios*.
3.- Una relación puede poseer más de una clave ajena (tendrá una clave ajena por cada relación referenciada de la cual dependa).
4.- Una relación puede no poseer ninguna clave externa.
5.- Una clave ajena puede enlazar una relación consigo misma (relaciones reflexivas).
6.- En el Modelo Relacional, toda relación posee clave primaria.
Observación: Nótese la diferencia con el Modelo Entidad-Relación, donde existen entidades que no poseen PK (las entidades débiles con dependencia en identificación).

Con este preámbulo se procederá a identificar las claves ajenas para cada tabla involucrada en el desarrollo de la aplicación.

Para lograr este cometido se procederá a navegar en el reporte creado anteriormente con la herramienta Case ER Studio 6.

Nos desplazamos hasta el modelo Físico (Physical) -> Main Model -> Foreign Keys[9]. Es en este punto en donde se pueden observar en la parte central los nombres de todas las tablas de

[9] Llaves Foráneas

la base de datos, si damos clic en alguna de estas se desplazará al lado derecho toda su descripción relacionada con las claves foráneas.

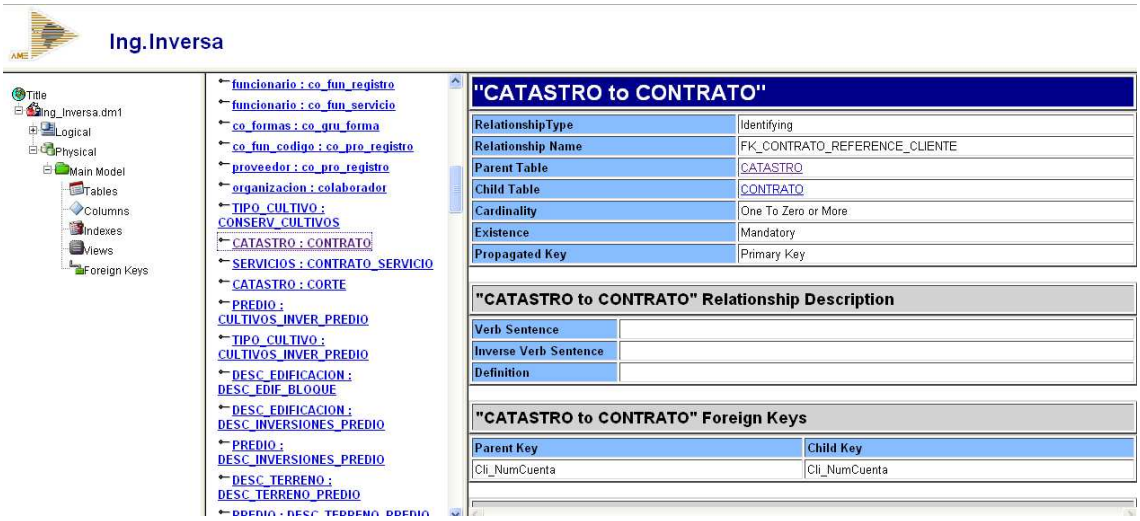


Figura 42. Identificación de claves foráneas con el Reporte Html.

En este punto se procederá al análisis de las tablas involucradas en el desarrollo de la aplicación para de esta manera encontrar los siguientes resultados.

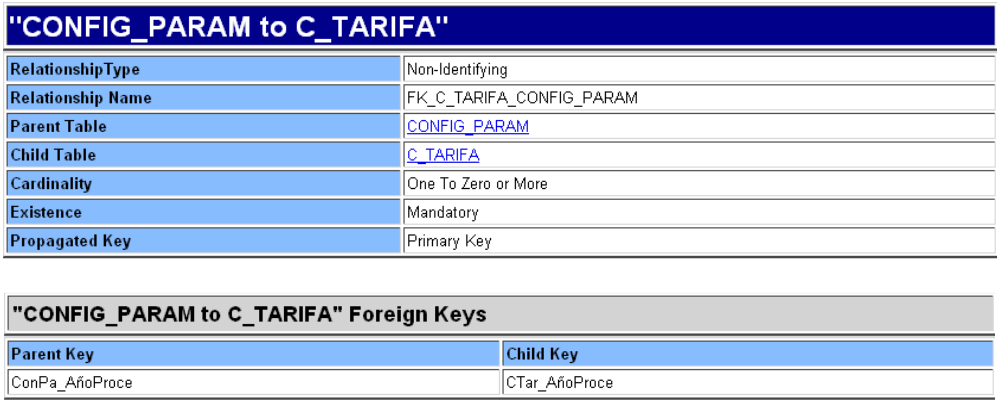


Figura 43. Clave Foránea Config_param a C_Tarifa

"CATASTRO to CARTAS_CLIENTE"	
RelationshipType	Identifying
Relationship Name	FK_CARTAS_C_REFERENCE_CLIENTE
Parent Table	CATASTRO
Child Table	CARTAS_CLIENTE
Cardinality	One To Zero or More
Existence	Mandatory
Propagated Key	Primary Key

"CATASTRO to CARTAS_CLIENTE" Foreign Keys	
Parent Key	Child Key
Cli_NumCuenta	Cli_NumCuenta

Figura 44. Catastro a Cartas Cartas_Cliente

"CONFIG_PARAM to CARTAS_CLIENTE"	
RelationshipType	Identifying
Relationship Name	FK_CARTAS_CLIENTE_CONFIG_PARAM
Parent Table	CONFIG_PARAM
Child Table	CARTAS_CLIENTE
Cardinality	One To Zero or More
Existence	Mandatory
Propagated Key	Primary Key

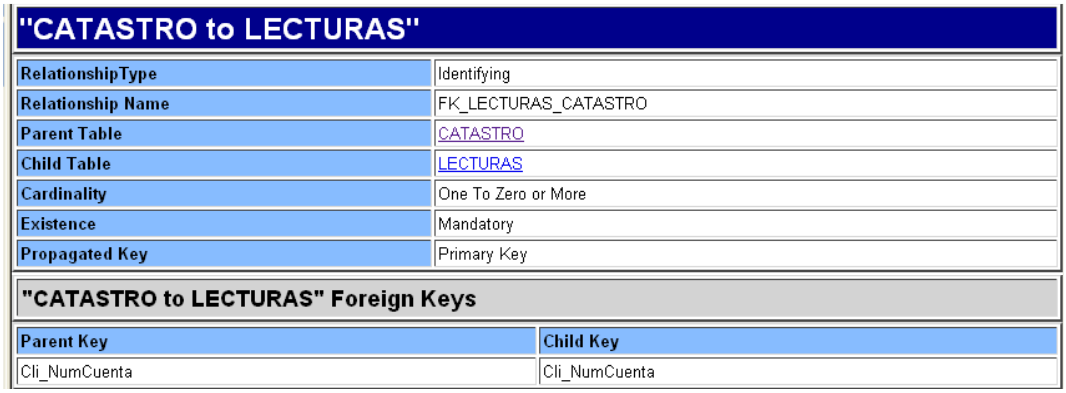
"CONFIG_PARAM to CARTAS_CLIENTE" Foreign Keys	
Parent Key	Child Key
ConPa_AñoProce	CarCl_Año

Figura 45. Config_Param a Cartas_Cliente

"USO_CONEXION to CATASTRO"	
RelationshipType	Non-Identifying
Relationship Name	FK_CATASTRO_USO_CONEXION
Parent Table	USO_CONEXION
Child Table	CATASTRO
Cardinality	Zero or One to Zero or More
Existence	Optional
Propagated Key	Primary Key

"USO_CONEXION to CATASTRO" Foreign Keys	
Parent Key	Child Key
UsoCo_Codigo	UsoCo_Codigo

Figura 46. Uso_Conexión a Catastro



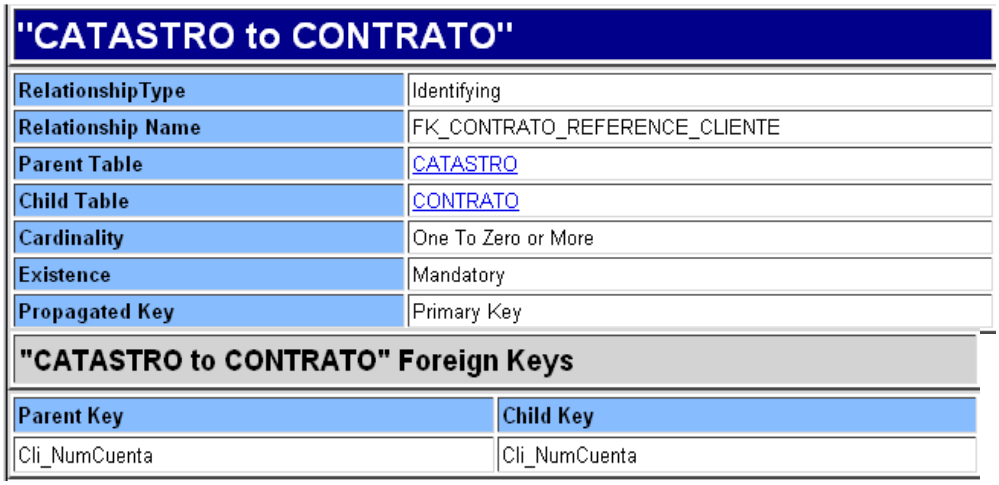
"CATASTRO to LECTURAS"

RelationshipType	Identifying
Relationship Name	FK_LECTURAS_CATASTRO
Parent Table	CATASTRO
Child Table	LECTURAS
Cardinality	One To Zero or More
Existence	Mandatory
Propagated Key	Primary Key

"CATASTRO to LECTURAS" Foreign Keys

Parent Key	Child Key
Cli_NumCuenta	Cli_NumCuenta

Figura 47. Clave Foránea Catastro a Lecturas



"CATASTRO to CONTRATO"

RelationshipType	Identifying
Relationship Name	FK_CONTRATO_REFERENCE_CLIENTE
Parent Table	CATASTRO
Child Table	CONTRATO
Cardinality	One To Zero or More
Existence	Mandatory
Propagated Key	Primary Key

"CATASTRO to CONTRATO" Foreign Keys

Parent Key	Child Key
Cli_NumCuenta	Cli_NumCuenta

Figura 48. Clave Foránea Catastro a Contrato

4.1.1.3 Etapa3: Unión del Esquema

Se deberá unir y reconvertir las estructuras y restricciones obtenidas en las fases anteriores.

4.1.1.3.1 Campos Multivaluados

Atributo Multivaluado: Es aquel que tiene más de una ocurrencia para un determinado valor de la clave. El DBMS exige que sus tablas tengan valores univaluados o atómicos.

Figura 49. Representación de Clave Foránea

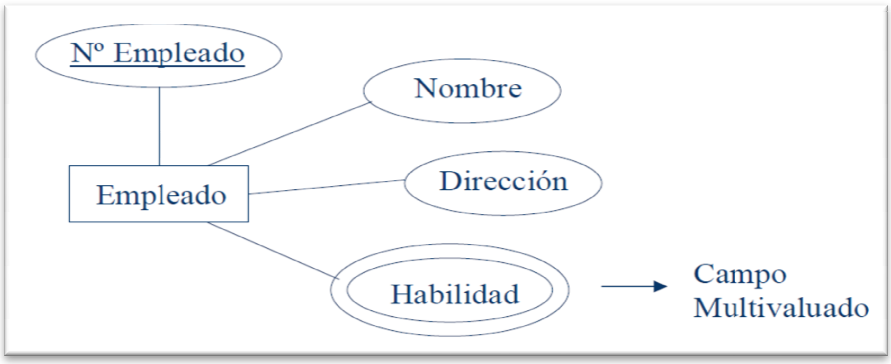


Figura 50. Representación Gráfica modelo E-R de Clave Foránea

Tabla Catastro

Luego de haber analizado esta tabla se pudo observar que no existen campos multivaluados, lo más cercano que se pudo encontrar son dos campos que hacen referencia a las direcciones del usuario (Cat_DirPCli, Cat_DirICli), en este caso la dirección principal y secundaria del usuario.

Tabla Lecturas

De igual forma luego de haber analizado esta tabla se pudo observar que no existen campos multivaluados, lo más cercano que se pudo encontrar son dos campos (Lec_Comentario y Lec_Obs), que hacen referencia a comentarios al momento de realizar las lecturas correspondientes que podrían interactuar de la misma forma.

Tabla Cartas_Cliente

Una vez analizada esta tabla no se han encontrado campos multivaluados bajo ninguna circunstancia.

Tabla Uso_Conexión

Los cuatro campos que contiene esta tabla, bajo ninguna circunstancia pueden ser multivaluados, ya que no almacenan de ninguna forma múltiples valores, y todos son de distinta índole.

Tabla C_Tarifa

Se ha analizado los campos que están contenidos en la tabla C_Tarifa, y se ha podido observar que no se ha detectado bajo ninguna circunstancia que estos pueden ser multivaluados, ya que no almacenan de ninguna forma múltiples valores, y todos son de distinta índole.

Tabla Config_Param

Luego de haber analizado esta tabla, se ha llagado a la conclusión que se trata de una tabla exclusiva para el al almacenamiento de valores numéricos indispensables para el cálculo de los precios a ser cancelados por el consumo del agua potable, alcantarillado y basura.

Tabla Config_Param_Año

De la misma manera luego de haber analizado esta tabla, se ha llagado a la conclusión que se trata de una tabla exclusiva para el al almacenamiento de valores numéricos

indispensables para el cálculo de los valores a ser cancelados por el consumo del agua potable, alcantarillado y basura.

Tabla Contrato

Se trata de una tabla de uso para el almacenamiento de datos para la realización de un determinado contrato por la compra de un medidor, y luego de haberla analizado se ha llegado a la conclusión que no existen campos multivaluados en este punto.

Tabla Saldo

Luego de analizar esta tabla se ha llagado a la conclusión que se trata de una tabla exclusiva para el al almacenamiento de valores numéricos indispensables para el cálculo de los valores a ser cancelados por la compra a crédito de medidores.

4.1.1.3.2 Redundancias

En bases de datos o en ficheros, la redundancia hace referencia al almacenamiento de los mismos datos varias veces en diferentes lugares. La redundancia de datos puede provocar problemas como:

- **Incremento del trabajo:** Como un mismo dato está almacenado en dos o más lugares, esto hace que cuando se graben o actualicen los datos, deban hacerse en todos los lugares a la vez.

- **Desperdicio de espacio de almacenamiento:** Ya que los mismos datos están almacenados en varios lugares distintos, ocupando así más bytes del medio de almacenamiento. Este problema es más evidente en grandes bases de datos.

- **Inconsistencia de datos:** Esto sucede cuando los datos redundantes no son iguales entre sí. Esto puede suceder, por ejemplo, cuando se actualiza el dato en un lugar, pero el dato duplicado en otro lugar no es actualizado.

- Si una base de datos está bien diseñada, no debería haber redundancia de datos (exceptuando la redundancia de datos controlada, que se emplea para mejorar el rendimiento en las consultas a las bases de datos).

Al analizar la Base de Datos en todo su contenido se ha podido observar redundancias en la información para el almacenamiento de sus datos.
Los campos que presentan redundancia en su información y por ende causan desperdicio en el espacio para el almacenamiento de la información, inconsistencia, etc., son:

Tabla Catastro

- ***Cat_DirPCli, Cat_DirLCli:*** Estos dos campos almacenan información referente a la dirección del cliente, pero llegado al caso los dos son del mismo tipo, para el objeto de estudio y para realizar las debidas transacciones solamente se ocupa el campo Cat_DirPCli, que se refiere a la dirección primaria del mismo, entonces el campo Cat_DirLCli, que almacena la dirección secundaria del cliente no se usa y en muchos de los casos se vuelve a guardar la misma información del campo Cat_DirPCli.

- ***Cat_TipoCli:*** Este campo tiene como tipo de dato (varchar (20)), y sirve simplemente para el almacenamiento de una clave que bien podría ser un capo de tipo (Int (1)), solo para almacenar un carácter.

- ***Cat_UbiLoc:*** De igual manera este campo es de tipo (varchar (20)), y sirve de clave foránea para enlazarse con otras tablas, también este campo podría llevar una variable ((Int (1)).

- ***Cat_DisponeAg:*** Este campo sirve para saber si el servicio de de agua está disponible o no, (1- SI TIENE, 0- NO TIENE), de igual manera existe desperdicio de memoria, también este campo podría llevar una variable ((Int (1)).

- ***Cat_EstadoMed:*** Campo que sirve para el almacenamiento del estado actual en el que se encuentra el medidor, de igual manera existe gran desperdicio de memoria, la cual se le podría ahorrar cambiando el tipo de dato por (Int(1)).

Tabla C_Tarifa

- ***UsoConex:*** Este campo almacena una clave foránea, la misma que nos sirve para enlazarnos con la tabla Uso_Conexión, es por esta razón que sería necesario que este campo solo ocupara 2 caracteres como máximo, por ejemplo (Int(2)), y no como sucede en la actualidad, ya que se trata de un campo (varchar(30)), causando desperdicio de memoria.

Se observan redundancias en las tablas en lo que se refiere a los campos que son claves foráneas, ya que es necesario cambiarlas el tipo de datos para lograr un mayor ahorro de memoria.

Además cabe destacar que existen muchos campos que no son ocupados en muchas ocasiones.

4.1.1.3.3 Dominios

Luego de haber analizado la base de datos por completo, se ha observado que en esta no se encuentran reglas de dominio por lo que se supone que los controles para el ingreso de datos se los ha realizado en la aplicación.

4.1.1.3.4 Significado de los Campos

Tabla CATASTRO

Esta tabla contiene las claves foráneas para poder conectarnos con otras tablas como Cartas Cliente, Lecturas, Uso Conexión, entre las más importantes para el desarrollo de la aplicación móvil[10] para el cálculo del valor de consumo del Agua Potable, Alcantarillado y Recolección de Basura.

- **Cli_NumCuenta**, Clave Primaria en la cual se registra el número de cuenta del cliente, está compuesta por 8 dígitos y sirve para relacionar con la tabla Contrato, Cartas Cliente, Lecturas, Uso Conexión, entre otras.
- **Cat_tipoCli**: En la cual se describe el tipo de cliente "1- REAL".
- **Cat_CedulaCli:** En este campo se almacena el número de cédula del cliente.
- **Cat_NombresCli**: Describe los nombres del cliente.
- **Cat_ApellidosCli**: Describe los apellidos del cliente.
- **Cat_DirPCli:** Se almacena la dirección principal del cliente.
- **Cat_DirLCli:** Se almacena una dirección secundaria del cliente.
- **UsoCo_Codigo:** Se trata de una llave foránea la misma que nos podemos enlazarnos con la tabla Uso_Conexión para de esta forma acceder al campo

[10] Son programas compilados y adaptados especialmente para ser utilizados desde cualquier soporte móvil

UsoCo_Desc, el mismo que contiene una descripción del tipo de conexión del usuario.

- **Cat_CodBarrioCli:** Almacena el código del barrio al que pertenece el cliente, para luego enlazarse con la tabla Barrios por medio del campo Bar_Codigo.
- **Cat_CodLog:** El código de catastro, según lo impuesto por las municipalidades, compuesta por 14 dígitos, distribuidos de la siguiente manera:
    - **Cat_SectorLog:** Se establece el sector (2 dígitos).
    - **Cat_RutaLog:** Se establece la ruta(2 dígitos).
    - **Cat_ManzanaLog:** Se establece la manzana (2 dígitos).
    - **Cat_secuenciaLog:** Se establece la secuencia (4 dígitos).
    - **Cat_PisoLog:** Se establece el piso (2 dígitos).
    - **Cat_DptoLog:** Se establece el departamento (2 dígitos).

- **Cat_UbicacionMed:** Se Detalla sobre el tipo de ubicación en la que se encuentra el medidor (0-No tiene; 1-Buena)
- **Cat_Estado_Med:** Para conocer el estado actual del medidor 0- SIN MEDIDOR 1- FUNCIONANDO
- **Cat_TelefonoCli:** Almacena el número telefónico del cliente en caso de que este los posea.
- **Cat_FecIngCli:** Almacena la fecha en que el usuario fue registrado en el sistema.
- **Cat_UsoConexAl:** Se trata de una llave foránea la misma que nos podemos enlazarnos con la tabla Uso_Conexión para de esta forma acceder al campo UsoCo_Desc, el mismo que contiene una descripción del tipo de conexión para el Alcantarillado del usuario.
- **Cat_NumeroMed:** Almacena el Número de Medidor
- **Cat_EstadoMed:** Almacena el estado actual en el que se encuentra el medidor, (0-SIN MEDIDOR, 1-FUNCIONANDO).

- **Cat_UbicacionMed:** Guarda la ubicación del actual en la que se encuentre el medidor (0- NO TIENE 1- BUENA).

Tabla CARTAS_CLIENTE

En esta tabla se almacenan tanto los pagos como las deudas anteriores de los clientes para los servicios por consumo de agua potable, alcantarillado y recolección de basura. Cada tupla cuenta con un registro del mes y año de la transacción.
Esta tabla se conecta a la tabla Catastro mediante el campo Cli_NumCuenta.
Se conecta además con la tabla DESMARCAR_TITULOS, mediante los campos: Cli_NumCuenta, CarCl_Año, CarCl_Tipo, CarCl_Mes.

- **Cli_NumCuenta:** Se trata de una clave foránea la cual sirve como nexo para conectarnos con la tabla Catastro.
- **CarCl_Año y CarCl_Mes:** Almacenan el año y mes en el proceso de cobro por los servicios.
- **CarCl_ValBasura**: Para registrar el monto por pagar para el cobro de la basura.
- **CarCl_Otros:** Destinado al cobro de otros rubros.
- **CarCl_Totaliq:** Aquí se almacena la suma total de todos los rubros por pagar.
- **CarCl_ValAdmin:** Se almacenan los valores por gastos administrativos.
- **CarCl_Tipo:** Con este campo podemos saber si el cliente ha pagado la deuda (P), aun esta se encuentra por pagar (S).
- **CarCl_Consumo:** En este campo se almacenará el valor del consumo del agua potable en metros cúbicos.
- **CarCl_ValConsumo:** En este campo se almacenarán los valores a pagar por el consumo del agua potable.
- **CarCl_ValBasura:** En este campo se almacenarán los valores a pagar por la recolección de basura.

- **CarCl_ValAlcanta:** En este campo se almacenarán los valores a pagar por el servicio del alcantarillado.
- **CarCl_IntMora:** Se almacenará el interés cobrado por la demora en los pagos.

Tabla Lecturas.

Para la consecución de nuestro proyecto la Tabla Lecturas es la tabla más importante y de mayor uso y trascendencia, ya que aquí se almacenarán los datos de la lectura realizada para cada medidor, en donde se registraran la fecha, observaciones, el valor leído para cada caso.

- **Cli_NumCuenta:** Se trata de una clave foránea la cual sirve como nexo para conectarnos con la tabla Catastro.
- **Lec_Año:** Se almacena el año de la lectura a ser procesada.
- **Lec_Mes:** Se almacena el mes de la lectura a ser procesada.
- **Lec_FecLec:** Se almacena la fecha en laque se realizó la lectura.
- **Lec_ValLec:** En este campo se almacenará el valor de la lectura actual.
- **Lec_Consumo:** Aquí se registra el consumo total del agua potable que viene dado por la resta de la lectura actual menos la lectura anterior.
- **Lec_Obs:** Se almacenan las observaciones para cada lectura.
- **Lec_TipCons:** Se almacena el tipo de consumo, si está leído o no un determinado medidor.
- **Lec_Usuario:** Se trata del código del usuario que realizó la lectura del medidor.
- **Lec_Estado:** Se trata del estado en que se encuentra el medidor en lo que refiere a su lectura.

Tabla Uso_Conexión.

En esta tabla se almacenan los tipos de conexiones las mismas que estarán disponibles para cada usuario y de esto dependerá el valor a pagar por el consumo del agua potable. Entre los tipos más importantes de conexiones se tiene, conexiones Domésticas, Comerciales, Especiales, Tercera Edad, etc.

- **UsoCo_Codigo:** Se trata de la clave principal para esta tabla.
- **UsoCo_Desc:** Es en este campo en donde se almacenarán los tipos de conexiones antes citados.
- **UsoCo_Tipo:** Se almacenarán los tipos de servicio para cada tipo de conexión, (Agua, Alcantarillado).
- **UsoCo_FecIng:** Almacenamos la fecha en la que se ingresó el nuevo tipo de conexión.

Tabla C_Tarifa.

La Tabla C_Tarifa se encuentra entre las más importantes ya que es aquí en donde se configurarán los tipos de tarifa para el pago por el consumo del agua potable. Con esta tabla se realizarán los diferentes tipos de cálculos para sacar los valores a cancelar por el consumo del agua potable. Los datos son almacenados por una serie de rangos para cada tipo de uso de conexión y para cada año.

- **CTar_Cod:** Se trata de la clave principal para esta tabla.
- **CTar_AñoProce:** Se refiere al año en el que fueron ingresados los rangos para las tarifas de cobro.
- **CTar_UsoConex:** Este campo va relacionado con el campo UsoCo_Desc de la tabla Uso_Conexion para describir los tipos de conexión pertenecientes para cada usuario.

- **CTar_Tipo:** Se Trata de la clasificación del tipo de tarifas (Básica, Rangos y Mixtas), cada una con una interpretación de cálculo distintas. Para cada uno de estos tipos de tarifas existe un rango de valores con su respectivo monto de tarifa y con su respectivo excedente.
- **CTar_ValMin:** Este campo almacena el valor mínimo para un rango de consumo.
- **CTar_ValMax:** Este campo almacena el valor máximo para un rango de consumo.
- **CTar_Tarifa:** Se almacena aquí el costo por el consumo del rango consumido.
- **CTar_Excede:** Este campo almacenará un valor por exceso de consumo, es decir por sobrepasar al rango de consumo, establecido por el valor mínimo (CTar_ValMin) y el valor máximo (CTar_ValMax).
- **CTar_Desc:** Se describe el tipo de servicio a ser calculado (Agua, Alcantarillado).

Tabla Config_Param

Esta Tabla es de vital importancia para la realización del cálculo de los valores a pagar por el consumo de los servicios de agua potable, basura y alcantarillado, pues aquí se encuentran todos los valores de configuración (de cobro) como: Valores administrativos, otros pagos, multas, mese de corte, mese de mora, entre otros para un determinado año.

Además por medio de esta tabla se sabrá si el valor a ser pagado por el consumo de los servicios de agua potable, basura y alcantarillado viene dado por medio de un cálculo por valores fijos o por valores diferenciados, dependiendo de cada configuración anual.

- **ConPa_AñoProce:** Se trata del año en proceso, es decir el año que va a contener todos los parámetros de configuración.
- **ConPa_Adm:** Valor de cobro por concepto de manejos administrativos.
- **ConPa_SisCo:** Por medio de este campo se obtendrán dos valores (1-2), por medio de estos se sabrá:
    - **1:** Sistema de cobro con valor fijo, en este caso obligatoriamente se tendrá que coger tres campos (Alcantarillado1, Basura1, Medidor1), de la tabla Config_Param_Año, directamente para el cálculo de los servicios de agua potable, alcantarillado y basura.
    - **2:** Sistema de cobro con valor diferenciado.
- **ConPa_TipoAlc y ConPa_TipoAg:** Estos dos valores funcionan en conjunto, pues por medio de estos se pueden determinar los valores a cancelar para el alcantarillado y la basura, siempre y cuando se esté calculando sistemas de cobro con valor diferenciado. Estos campos pueden obtener los valores de 1, 2, 3 y 4.
- **ConPa_ValorFijoAg, ConPa_ValorFijoAlc, ConPa_ValorFijoBas:** Estos tres campos funcionan en conjunto puesto que si ConPa_SisCo tiene el valor de 1, estos tendrán la opción de tener valores distintos de cero, encaso de que los tres campos tengan valor igual acero se procederá a tomar los valores de la tabla Config_Param_Año.
- **ConPa_Conex:** Valores a ser cancelados por el concepto de una nueva conexión.
- **ConPaReconex:** Valores a ser cancelados por el concepto de una re conexión del medidor.
- **ConPa_Multas:** Valores a ser cancelados por el concepto de multas el pago a destiempo de los servicios
- **ConPa_MesesCorte:** Se almacenan los meses que lleva de corte el medidor el mención.

- **ConPa_MesesMora:** Se almacenan los meses que lleva de mora un usuario para el pago de los servicios.
- **ConPa_PorcetBas:** En este campo se almacena el porcentaje para ser calculado con el consumo del agua potable para el servicio de basura.
- **ConPa_PorcetAlc:** En este campo se almacena el porcentaje para ser calculado con el consumo del agua potable para el servicio de alcantarillado.

Tabla Config_Param_Año

Se trata de una Tabla reciclable ya que sirve para el almacenamiento de información de varias fuentes, esto para cada año.

Se consideran los servicios de Alcantarillado, Basura y Medidor, este último sirve para el almacenamiento de valores en caso de medidores dañados.

Y por tratarse de una tabla que trabaja siempre y cuando exista un sistema de valor diferenciado se tendrán campos para diferenciar la categoría de la conexión, en este caso Domesticas, Comerciales, Industriales, Oficiales, Tercera Edad y Otras.

- **ConPa_AñoProce:** Se trata del año en proceso, es decir el año que va a contener todos los parámetros de configuración.
- **ConPaA_Tipo:** Campo que almacena el tipo de servicio al que se hace referencia (Alcantarillado, Basura, y Medidor), siendo este último usado para referirse cuando el medidor está dañado, etc.
- **ConPaA_BasDom:** Campo necesario para el almacenamiento de valores para las conexiones de tipo domésticas.
- **ConPaA_BasCom:** Campo necesario para el almacenamiento de valores para las conexiones de tipo comerciales.

- **ConPaA_BasInd:** Campo necesario para el almacenamiento de valores para las conexiones de tipo industriales.
- **ConPaA_BasOfi:** Campo necesario para el almacenamiento de valores para las conexiones de tipo Oficiales.
- **ConPaA_BasTra:** Campo necesario para el almacenamiento de valores para las conexiones de tipo Tercera Edad.
- **ConPaA_BasOtr:** Campo necesario para el almacenamiento de valores para otro tipo de conexiones.

Tabla Contrato

Tabla en la cual se almacenan los datos referentes a un contrato realizado `para la compra venta de un medidor.

- **Con_Numero:** Se almacena el número de contrato, es un campo automático.
- **Con_Servicio:** Se trata de una clave foránea, en la cual se referencian el tipo de contrato para ente caso Venta de Medidor.
- **Cli_NumCuenta:** Clave Foránea para el almacenamiento del número de cuenta del cliente.
- **Con_Fecha:** Fecha en que se realizó el contrato.
- **Con_FormaPago:** Se registra la forma de pago para la venta del medidor (Contado, Crédito).
- **Con_Valor:** Valor acordado a ser cancelado por la venta del medidor.
- **Con_ValContado:** Valor ingresado por el pago de contado.
- **Con_ValCredito:** Valor ingresado por el pago a crédito.
- **Con_Abono:** Valor abonado para el pago a crédito.
- **Con_CuotasPlazo:** Número de cuotas plazo a ser pagadas.
- **Con_CuotaValor:** Valor resultante al dividir el valor a pagar restante por el número de meses.

- **Con_ValAPagar:** Valor restante a pagar.
- **Con_FecPagCont:** Fecha de pago.
- **Con_ValPagCont:** Valor pagado como cuota.
- **Con_AreaAten:** Área en la que ha sido atendido el usuario.
- **Con_Responsable:** Persona responsable de la venta del medidor.
- **Con_Mesproceso:** Mes en el que se está procesando la venta del medidor.

Tabla Saldo

Tabla en la cual se almacena las transacciones referentes a la compra de un medidor a crédito.

- **Con_Numero:** Número de contrato para la compra venta de un medidor.
- **Sal_SaldoContra:** Saldo a pagar, por la compra venta de un medidor.
- **Sal_NumCuotas:** Número de cuotas acordadas para el pago.
- **Sal_CuotasPag:** Cuotas pagadas.
- **Sal_FecAct:** Fecha del pago de una cuota.
- **Cli_NumCuenta:** Número de cuenta del cliente.
- **Con_Usuario:** Código del usuario del sistema.

4.1.1.4 Etapa4: Análisis de programas

Se realizará un estudio del código fuente existente para comprobar que las restricciones, forma de procesar de los datos, significado, etc. se corresponde con el estudio realizado en las fases anteriores.

En esta misma etapa se podría incluir el análisis de formularios, consultas, informes, etc. que puedan existir, ya que de esa forma podemos ver si los resultados obtenidos con la LBD[11] coinciden con el modelo extraído.

De acuerdo a la introducción expuesta se procederá por lo tanto a analizar el Sistema de Gestión Comercial de Servicios GCS, el mismo que viene funcionando en varios municipios del Ecuador con el fin de realizar la gestión de cobro por servicios básicos. El sistema GCS, funciona en conjunto con la Base de Datos “dbame”, ingresando, actualizando o eliminando registros.

Ahora se analizarán las tablas afectadas al realizar las diferentes formas de configuración de las tarifas.

- **Formulario de Configuración del Sistema.**

  El ingreso y configuración de las tarifas de cobro se lo realiza en el módulo Configuración -> Sistema.

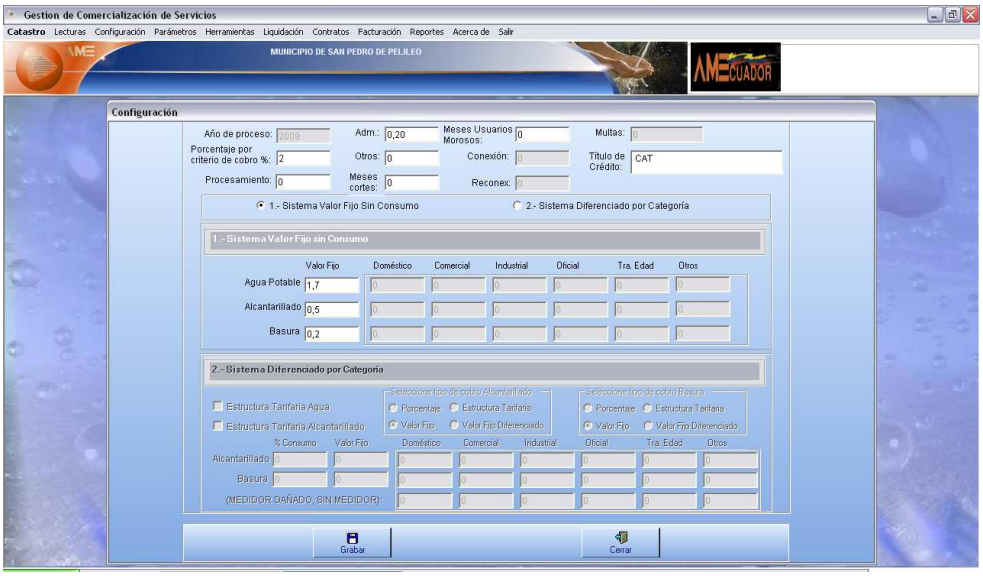


Figura 51. Pantalla de Configuración de Tarifas para el Sistema GCS

[11] Base de Datos Legada, Base de Datos Original

Este módulo cuenta con tres partes, la primera de ellas se trata de una configuración obligatoria, en la cual se registran datos como: el año procesado, valor a cobrarse por cuestiones administrativas, porcentaje por criterio de cobros, multas.

El segundo módulo se refiere a la configuración del sistema con valor fijo de consumo, en donde se configurarán, valores fijos para el consumo de Agua Potable, Alcantarillado y Basura. Este criterio se lo puede hacer de una manera general, para todos los tipos de conexión y de una manera específica para cada tipo de conexión.

El tercer módulo tiene que ver con la configuración para un sistema diferenciado por categoría, aquí se puede observar distintas formas de ingreso de los datos, ya sea por estructura tarifaria, porcentaje, valor dijo o valor fijo diferenciado.

Al ingreso de estos datos se afectará la tabla Config_Param, en cada uno de sus campos relacionados con este formulario; y de la misma manera con el ingreso para un sistema fijo sin consumo para cada tipo de conexión se afectará a la tabla Config_Param_Año.

- **Formulario Ingreso de Catastros**

Por medio de este formulario se podrá ingresar nuevos clientes, catastrados, ingresando para este fin la localización del predio, los datos personales del cliente, datos para el consumo del agua potable, alcantarillado y medidor; junto a ello se ingresará la forma de conexión para el consumo del agua potable.

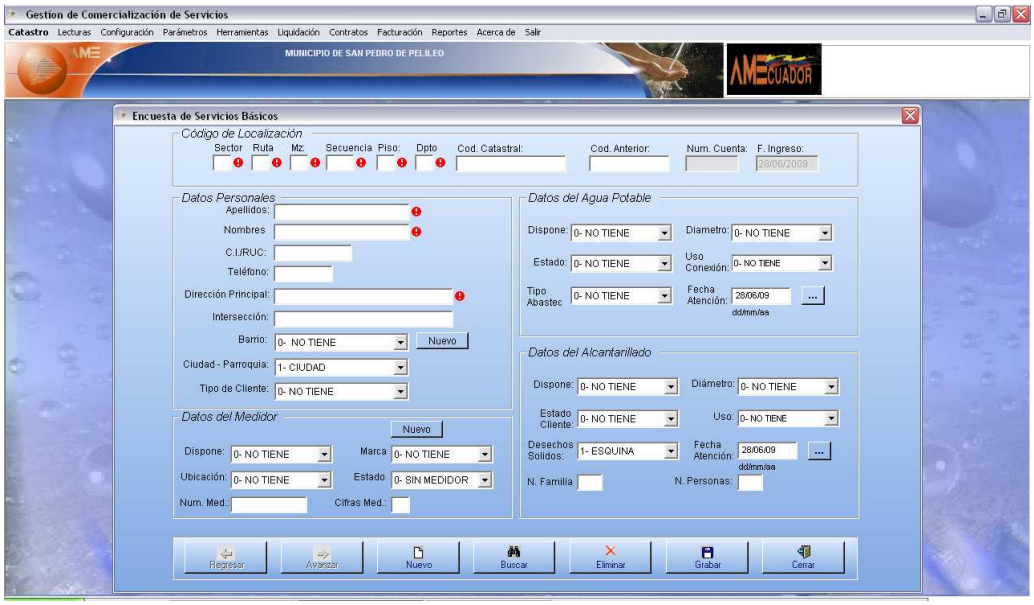


Figura 52. Formulario para el ingreso de nuevos Catastros para el Sistema GCS

Con el ingreso, eliminación y actualización de los datos a este formulario se afectará a la tabla Catastros de la base de datos dbame.

- **Formulario de Ingreso de Nuevas Lecturas.**

En este formulario se ingresará las nuevas lecturas para el consumo del servicio del agua potable, obteniendo al instante el cálculo para un determinado consumo de agua, dado por metros cúbicos.

La tabla afectada al realizar este proceso sin duda es la tabla Lecturas, en la cual se registrarán todos los datos concernientes a la toma de las lecturas del agua potable.

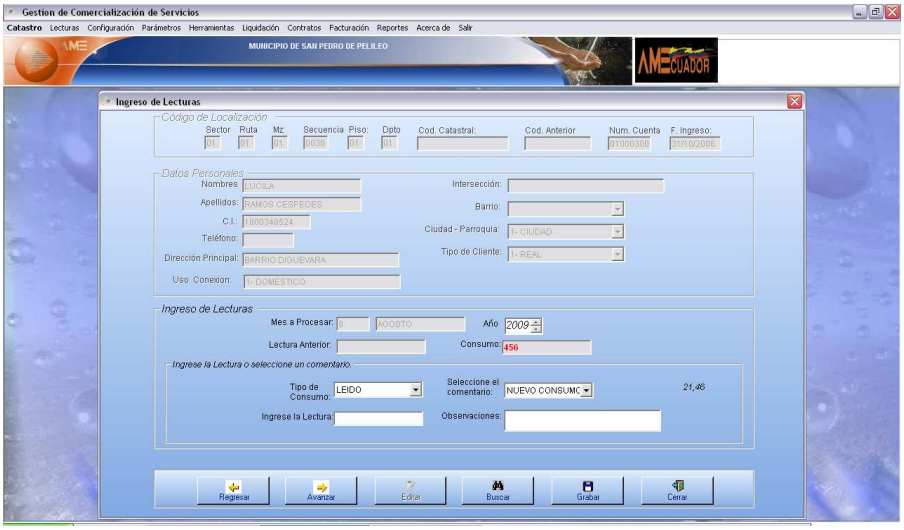


Figura 53. Formulario para el ingreso de nuevas Lecturas para el Sistema GCS

De igual manera al ingresar datos en este formulario y al realizar los respectivos cálculos se afectará a la tabla Cartas_Cliente.

- **Formulario para la venta de un Medidor**

En este formulario se ingresan todos los datos concernientes a la realización de una compra venta de un medidor.

El ingreso o actualización de estos datos afectará a la tabla contrato.

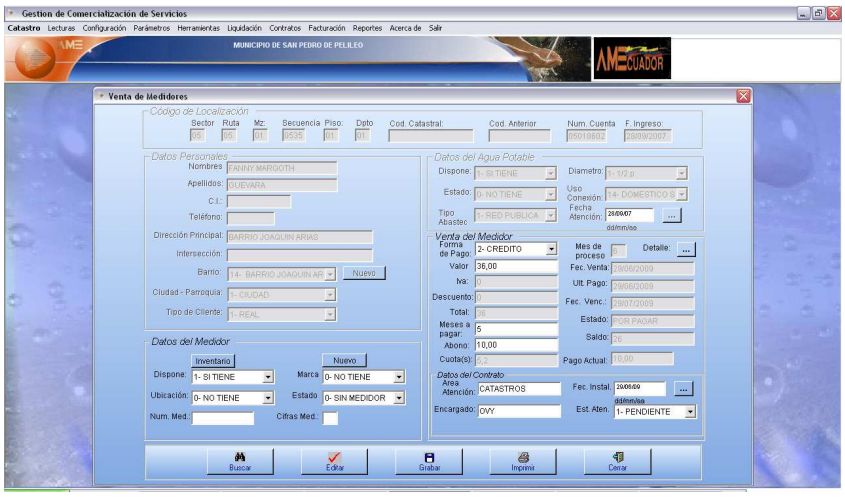


Figura 54. Formulario para el ingreso de nuevas ventas de medidores

4.1.2 Fase II.

4.1.2.1 Etapa1: Conceptualización Básica

Se busca en esta etapa encontrar campos que tengan la misma estructura y q se refieren a atributos de las mismas entidades, para transformarlos en capos multivaluados.
Se listarán los campos candidatos para ser transformados en campos multivaluados.

- Cat_DirPCli, Cat_DirLCli en la tabla Catastro.
- FicMu_Telefono, FicMu_Fax en la tabla Ficha_Municipal.

4.1.2.2 Etapa2: Normalización

En esta etapa se pretende aportar un significado en la semántica de las construcciones explícitas. El objetivo es representar la información con una metodología estándar que sea altamente legible.

El modelo entidad relación cumple con amplitud el objetivo planteado por lo que existen muchos modelos aplicables a bases de datos relacionales.

Para la obtención de las estructuras podemos seguir los pasos descritos por Premerlani (Premerlani, 1994), que considera que cada fichero es una posible tabla y cada campo del fichero será un campo de la tabla.

a) Identificar un conjunto de campos que puedan ser la clave principal para sus respetivos ficheros.

b) Determinar las llaves externas, que serán habitualmente del mismo tipo que sus correspondientes en la tabla principal.

c) Determinar qué ficheros encontrados pueden dejar de considerarse como tablas. Por ejemplo, aquellos que no se haya podido determinar campo clave.

d) Buscar y encontrar generalizaciones, que puede hacerse mediante la localización de grandes grupos de claves ajenas. Así podemos decir que una clave formada únicamente por claves ajenas suele indicar una relación de esta tabla o entidad con n entidades, formando por tanto una relación de grado n+1.

e) Encontrar asociaciones mediante la repetición de campos en una tabla. Si en un determinado campo se encuentra repetido un dato que aparece en otra tabla, podemos indicar que existe una relación entre ellas.

El usuario tiene una gran participación, que es la de determinar si un objeto externo es almacenable o no. Estas decisiones no son siempre certeras.

Una vez que se ha detectado las tablas y su estructura, podemos utilizar la siguiente notación: NOMBRE_TABLA[Campo1, Campo2, Campo3, Campo_n].

Se procederá entonces con la realización de este punto, para lo cual se conjugarán todos los pasos descritos anteriormente.

CATASTRO[Cli_NumCuenta, Cat_TipoCli, UsoCo_Codigo, Cat_CedulaCli, Cat_NombresCli, Cat_ApellidosCli, Cat_DirPCli, Cat_DirICli, Cat_CodBarrioCli, Cat_TelefonoCli, Cat_FecIngCli, Cat_ClasifCli, Cat_CicloFactCli, Cat_CodigoLoc, Cat_SectorLoc, Cat_RutaLoc, Cat_ManzanaLoc, Cat_SecuenciaLoc, Cat_PisoLoc, Cat_DptoLoc, Cat_CodCatLoc, Cat_ClaAntLoc, Cat_UbiLoc, Cat_DisponeAg,

Cat_DiametroAg, Cat_EstConexAg, Cat_TipoAbastAg, Cat_FecServAg, Cat_FacturaAg, Cat_ConsPromAg, Cat_FormaPagoAg, Cat_DisponeAl, Cat_DiametroAl, Cat_UsoConexAl, Cat_EstadoClienteAl, Cat_FecServAl, Cat_FacturaAl, Cat_Factura1Al, Cat_BaseCalcAl, Cat_ConsBaseAl, Cat_DisponeMed, Cat_MarcaMed, Cat_UbicacionMed, Cat_EstadoMed, Cat_NumeroMed, Cat_NumCifras, MedCat_NumFliasDes, Cat_NumPerDes, Cat_DescDes, Cat_DesCli].

USO_CONEXION[UsoCo_Codigo, UsoCo_Desc, UsoCo_FecIng, UsoCo_Tipo].

BARRIO[Bar_Codigo, Bar_Nombre].

MEDIDOR[Med_Codigo, Med_Marca, Med_FecIng].

LECTURAS[Cli_NumCuenta, Lec_Año, Lec_Mes, Lec_Ciclo, Lec_FecLec, Lec_ValLec, Lec_Comentario, Lec_Obs, Lec_Consumo, Lec_TipCons, Lec_DiasCons, Lec_Usuario, Lec_Estado].

VENTA_MEDIDOR[Med_Numero, Med_FormaPago, Med_Estimado, Med_Contador, Med_FecIng, Med_ValorMed, Cli_NumCuenta, Med_NumMesPagMedquitar, Med_ValMensualMedquitar, Med_AbonoMed, Med_FecVentaMed, Med_FecInstMed]

VALOR_MES[ValMe_Codigo, ValMe_Año, ValMe_Mes, ValMe_Valor, Cli_NumCuenta, Med_Numero].

CARTAS_CLIENTE[Cli_NumCuenta, CarCl_Año, CarCl_Tipo, CarCl_Mes, CarCl_Consumo, CarCl_ValConsumo, CarCl_ValBasura, CarCl_ValAlcanta, CarCl_ValAdmin, CarCl_Otros, CarCl_Proc, CarCl_ValConex, CarCl_ValReconex, CarCl_ValMultasCon, CarCl_ValVarios, CarCl_IntMora, CarCl_Interes, CarCl_TotaLiq, CarCl_TotaFac, CarCl_FechaLiq, CarCl_FechaFac, CarCl_FechaProc, CarCL_EstaLiq,

CarCl_ValPag, CarCl_ValPen, CarCl_ValIva, CarCl_ValDesc, CarCl_Codigo, Usu_Cod, SalAnt_Abono, CarCl_ConMed].

CONFIG_PARAM[ConPa_AñoProce, ConPa_PorcenCri, ConPa_SisCo, ConPa_Fecha, ConPa_Coment, ConPa_Adm, ConPa_Otros, ConPa_Proc, ConPa_Conex, ConPa_Reconex, ConPa_Multas, ConPa_MesesCorte, ConPa_MesesMora, ConPa_ValorFijoAg, ConPa_ValorFijoAlc, ConPa_ValorFijoBas, ConPa_PocentAlc, ConPa_PorcentBas, ConPa_ValFijoAlc1, ConPa_ValFijoBas1, ConPa_TipoAlc, ConPa_TipoAg].

CONFIG_PARAM_AÑO[ConPa_AñoProce, ConPaA_Tipo, ConPaA_BasDom, ConPaA_BasCom, ConPaA_BasInd, ConPaA_BasOfi, ConPaA_BasTra, ConPaA_BasOtr].

C_TARIFA[CTar_Codigo, CTar_AñoProce, CTar_UsoConex, CTar_Tipo, CTar_ValMin, CTar_ValMax, CTar_Tarifa, CTar_Excede, CTar_Desc].

CONTRATO[Con_Numero, Con_Servicio, Cli_NumCuenta, Con_Fecha, Con_FormaPago, Con_Valor, Con_ValContado, Con_ValCredito, Con_Abono, Con_CuotasPlazo, Con_CuotaValor, Con_CuotaNum, Con_ValAPagar, Con_Interes, Con_FecPagCont, Con_ValPagCont, Con_AreaAten, Con_Responsable, Con_FecPlanAten, Con_EstAten, Con_Desc, Con_EstFinanc, Con_Estado, Con_Mesproceso, Con_Usuario, Con_Iva, Con_Descuento].

SALDO[Con_Numero, Sal_SaldoContra, Sal_NumCuotas, Sal_CuotasPag, Sal_ValPagado, Sal_Interes, Sal_FecAct, Sal_desc, Con_Servicio, Cli_NumCuenta, Con_Usuario].

FACTURACION_CLIENTE[FacCl_Codigo, Cli_NumCuenta, FacCl_Tarifa, FacCl_TipoTarifa, FacCl_EmitFac, FacCl_FecIniCons, FacCl_ConsPromAcum, FacCl_FacContrato, FacCl_MetCalcUltFact, FacCl_MesesAtraso, FacCl_ValUltFac, FacCl_ValUltlec, FacCl_FecUltLec, FacCl_ValUltPago, FacCl_FecUltPago, FacCl_NumPagos].

VALORES_CLIENTE[Cli_NumCuenta, ValCl_Tipo, ValCl_Consumo, ValCl_Alcantarillado, ValCl_Manten, ValCl_Serv, ValCl_ValDeuda, ValCl_IntFinan, ValCl_IntMora, ValCl_Anticipo, ValCl_PagoParcial, ValCl_CorteCoactiva, ValCl_SistHidraulico, ValCl_Alcanta, Facli_Codigo].

RECONEXION[Cli_NumCuenta, Rec_Año, Rec_Mes, Rec_Fecha].

TITULOS[Tit_codigo, Tit_usuario, Tit_Fecha, Tit_Sec, Cli_NumCuenta].

SALDO_ANTERIOR[SalAnt_Codigo, Cli_Numcuenta, SalAnt_Valor, SalAnt_Abono, SalAnt_Fecha].

RECAUDACION_CLIENTE[RecCl_Codigo, RecCl_TipoCuenta, RecCl_FecRegistro, RecCl_Banco, Cli_NumCuenta,].

PARAM_CUENTA[ParCu_Codigo, ParCu_Tipo, ParCu_Mes, ParCu_Estado, ParCu_Comentario, Cli_NumCuenta].

Por último la representación gráfica en el Modelo Entidad relación de las tablas antes expuestas es el siguiente:

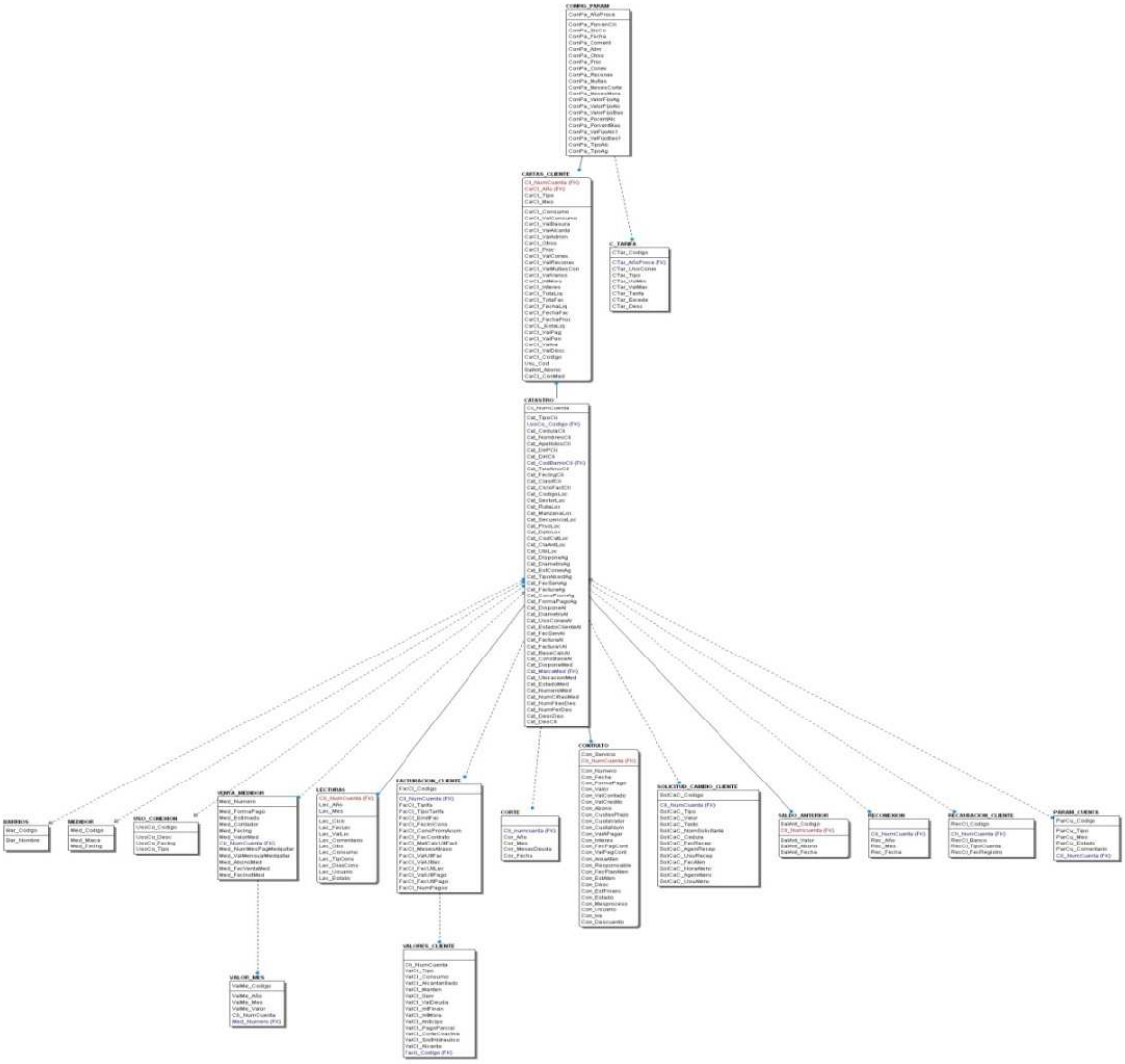


Figura 55. Modelo Entidad Relación Base de Datos dbame

4.2 FASES DE LA METODOLOGIA PROGRAMACIÓN EXTREMA

4.2.1 FASE I

4.2.1.1 Planificación del Proyecto

El objetivo que se desea alcanzar, en éste punto con la aplicación de una metodología de desarrollo ágil basada en una serie de valores y de prácticas de buenas prácticas que persiga

el objetivo de aumentar la productividad y funcionalidad a la hora de elaborar programas, por lo se desarrollará una aplicación cuyo objetivo es satisfacer los requerimientos especificados inicialmente, lo que mejorará el ambiente, calidad y rendimiento de trabajo de todos los usuarios de los Departamentos de Agua Potable de los Municipios en mención.

4.2.1.2 Historias de Usuario

Para conocer los requerimientos y necesidades principales de los usuarios que van hacer uso de la aplicación, debemos detallar cada uno de ellas, para que se tenga en claro el objetivo central de la aplicación.

A continuación citaremos las Historias de Usuario (H.U), requeridas por los usuarios:

H.U # 1

Desarrollo de Aplicación

“Realizar una aplicación que tenga como objetivo la toma de lecturas rápidas en los medidores de Agua Potable, mediante código de barra, para su posterior cálculo de acuerdo a la categorización de los usuarios y consumo, para una posterior impresión de notificaciones destinadas a los usuarios”.

H.U # 2

Transferencia de Datos

“La aplicación debe complementarse con un segundo módulo (Una aplicación de escritorio), que permita la transferencia Bidireccional (PC - Pocket y Pocket - PC) de los datos

recolectados en las lecturas, mediante el puerto TCP-IP, así como también de los usuarios que se encuentran catastrados en las bases de datos de los servidores departamentales.

H.U # 3

Presentación de Notificaciones

“La presentación de las notificaciones se las debe hacer in situ, es decir al momento de realizar las lecturas de los medidores, ya que estas serán entregadas a los usuarios para una posterior verificación de los valores a ser cancelados.”

H.U # 4

Presentación de Mensajes

“Se debe tomar en consideración mensajes de error o de aceptación cuando se realice alguna acción y a su vez considerar acciones inválidas en la aplicación.”

H.U # 5

Utilización de fuente de Datos

“Para la presentación de los reportes se debe tomar en consideración la información que actualmente poseen los Departamentos en cuestión, en las fuentes electrónicas y que se encuentran en uso”

H.U # 6

Generación de Código de Barras

“Se debe tener en cuenta además que existe un módulo para la generación de códigos de barras para los usuarios que se encuentren registrados como clientes de los departamentos de Agua Potable.”

H.U # 7

Consideraciones Adicionales

“Se debe considerar aspectos extras que mejoren o a su vez optimicen la funcionalidad, operatividad, o navegabilidad de la aplicación”

4.2.1.3 Planificación de Publicaciones

Para determinar el tiempo que conlleva el desarrollo y ejecución de los requerimientos de la aplicación especificados en las historias de usuario, debemos plasmar un cronograma tentativo de acuerdo al versionado de la aplicación, así tenemos:

La versión inicial abarca las historias de usuario principales que indican o destacan la funcionalidad principal de la aplicación

Número de la Versión: Versión 1.0 Beta	**Fecha de Publicación:** 25 de Octubre del 2009
Num. Historias de Uso	**Título de la Historia de Usuario**
H.U. #1	Desarrollo de la Aplicación
H.U. #3	Presentación de notificaciones
H.U. #4	Presentación de mensajes
H.U. #5	Utilización de fuente de datos

Tabla 12. Versión Inicial para las Historias de Usuario

En la versión completa a más de mejorar las historias de usuario definidas en la versión anterior, se toma en consideración las historias de usuario restantes que den la correcta funcionalidad de la aplicación.

Número de la Versión: Versión 1.0 Completa	**Fecha de Publicación:** 5 de Noviembre del 2009
Num. Historias de Uso	**Título de la Historia de Usuario**
H.U. #2	Transferencia de Datos
H.U. #6	Generación de Código de Barras
H.U. #7	Consideraciones Adicionales

Tabla 13. Versión Completa para las Historias de Usuario

4.2.1.4 Cronograma de cada Historia de Usuario

# H.U	**Responsables**	**F. Inicio**	**F.Final**	**Prioridad**	**Versión**
6	Program Manager	04-01-2009	07-01-2009	Alta	Beta
1	Product Manager Program Manager Designer	08-02-2009	15-02-2009	Alta	Beta
2	Program Manager	18-04-2009	21-04-2009	Alta	Completa
3	Product Manager Program Manager Designer	22-06-2009	21-06-2009	Alta	Beta

4	Designer	23-07-2009	24-07-2009	Media	Beta
5	Program Manager	25-07-2009	27-07-2009	Media	Completa
7	Product Manager Program Manager Designer	30-09-2009	03-09-2009	Baja	Completa

Tabla 14. Cronograma de cada Historia de Usuario

4.2.1.5 Iteraciones

Para el desarrollo de la aplicación fueron necesarias realizar tres iteraciones diferentes, en las que en cada una de ellas se realizan nuevas tareas asignadas a los diferentes miembros del grupo de trabajo.

Cada iteración debe tomar como tiempo máximo de desarrollo seis semanas, las mismas que se encuentran conformadas como a continuación se indica:

Iteración 1

Empieza con una reunión con el cliente, es decir con los miembros y personal que solicitaron la ejecución de la aplicación en donde ponen en conocimiento las historias de usuario principales con las que va a constar la aplicación.

Tiempo de Duración: 3 semanas

Fecha Inicial: 04-01-2009 **Fecha de Publicación:** 22-01-2009

De entre todas las historias de usuario, estudiamos las iniciales que estarán contempladas en la iteración inicial:

H.U. #5: Utilización de Fuente de Datos

Para la utilización de la fuente de datos., necesitamos acceder a los datos de los sistemas que se encuentran actualmente en funcionamiento, pero por ser una aplicación que permite presentar notificaciones consumiendo información de la base de datos, esta información no debe ser modificada bajo ninguna circunstancia. Los valores calculados en el dispositivo móvil son solamente temporales al momento de la impresión al momento de entregar la notificación.

Esta historia se clasifica en las siguientes tareas que estarán a cargo del grupo de desarrollo.

Tarea 1.-

Determinar el motor de base de datos con las que se desarrollará la aplicación

Responsables: Program Manager, Product Manager

Tiempo de Duración: 1 día

El Motor de Base de Datos de los Sistemas de Cobros GCS, en los diferentes Departamentos de Agua Potable, es Microsoft SQL Server.

De acuerdo a los requerimientos del cliente, se debe trabajar con el mismo motor de base de datos usado en el sistema antes mencionado, por lo que se optará por trabajar con SQL Compact Edition para la aplicación del dispositivo móvil.

Tarea 2.-

Realización de la Ingeniería Inversa a la Base de Datos "dbame", indispensable para comprender el funcionamiento del sistema GCS.

Responsables: Program Manager, Product Manager

Tiempo de Duración: 7 días

Para un mejor entendimiento de la estructura del sistema GCS y particularmente de la base de datos a la que se vincula se deberá realizar la ingeniería inversa a la misma, de esta manera se sabrá a ciencia cierta de las características y funciones de cada tabla, incluso de los tipos de datos a los que están vinculadas, para un mejor diseño de la aplicación móvil.

Tarea 3.-

Análisis y elección de la las diversas tablas de la base de datos “dbame”, del sistema GCS que serán usadas para la implementación de la aplicación móvil “SISNIYCS”, para su posterior implementación.

Responsables: Program Manager, Product Manager

Tiempo de Duración: 7 días

De acuerdo a los requerimientos de usuario, se debe realizar una aplicación para dispositivos móviles, la cual trabajará con algunas tablas de la base de datos “dbame”, del sistema GCS, para esto se realizará un análisis exhaustivo de todas las tablas antes citadas en la ingeniería inversa, y particularmente de sus campos con los tipos de datos que estos manejan, para de esta forma poder diseñar el esquema de la base de datos portable.

Una vez realizado el análisis y escogido las tablas, se procederá a su implementación de las mismas, para esto se usará el motor para base de datos portable SQL CE.

H.U. #1: Desarrollo de una Aplicación

Para el desarrollo de la aplicación se debe tener en consideración que la misma debe ser realizada para dispositivos móviles, específicamente para una Pocket PC, y en la cual debe residir la base de datos portable para por medio de los datos que esta almacena, realizar las diferentes cálculos para poder entregar las notificaciones impresas con los mismos.

Además cabe resaltar que la aplicación completa, destaca varios módulos, entre ellos, el módulo de transferencia de datos y el módulo para la generación de códigos de barras.

Dichos módulos serán contemplados en otras historias de usuarios ya que se desarrollarán en forma separada y en tiempos distintos de acuerdo al avance del trabajo de investigación.

Tarea 1.-

Determinar las herramientas de programación y lenguajes de programación para el desarrollo de la aplicación

Responsables: Program Manager, Product Manager, Designer

Tiempo de Duración: 14 días

Luego de haber analizado las herramientas disponibles para el desarrollo de las aplicaciones en sus distintos módulos se ha optado por utilizar C#. Net, tanto para dispositivos móviles como para aplicaciones de escritorio, y de esta manera poder complementar los módulos para una programación más eficiente y eficaz.

Tarea 2.-

Diseño de una aplicación móvil utilizando las herramientas seleccionadas.

Responsables: Program Manager, Product Manager, Designer

Tiempo de Duración: 70 días

Para el desarrollo de una plantilla de la interfaz y diseño de la aplicación, se utilizo las herramientas seleccionadas, para generar esquemas de la manera de cómo se va a ver la información, tipos de imágenes, ubicación de las imágenes en la pantalla, menús, cuadros de texto, etc.

Los diseñadores deben estar en la posibilidad de crear interfaces amigables y fáciles de manejar, para que los usuarios puedan manejar la aplicación sin problemas, los programadores deben realizar los pasos y procedimientos de las acciones que se realizarán en cada opción activada o ingresada.

H.U. #3: Presentación de Notificaciones

Las notificaciones están basadas en la información que posee la base de datos del sistema GCS, transferida al sistema móvil, la información está basada en los usuarios registrados en dicha base de datos.

Dicha notificaciones son dinámicas, es decir deben contener parámetros que puedan ser seleccionadas ingresados por el usuario del sistema y presentar su reporte con sus respectivos valores a ser cancelados.

Tarea 1.-

Determinar los parámetros y objetos de SQL Server que faciliten la manipulación de los datos

Responsables: Program Manager, Product Manager

Tiempo de Duración: 10 días

Los objetos que más se centran en la manipulación de la información son las mismas tablas, que son referenciadas en las diferentes consultas de datos, y a la vez son utilizados en la aplicación. Las acciones del programador, son trabajar sobre la información de las bases de datos de acuerdo a los índices requeridos y solicitados, los mismos que deben variar de acuerdo al usuario consultado y evaluado por el sistema.

Para poder presentar una propuesta que se rija a los requerimientos especificados, es necesario realizar un estudio previo sobre los parámetros involucrados y los que se han propuesto como solución a las peticiones involucradas por el Departamento de Agua Potable de la Municipalidad de Baños. PEINSEB, es una propuesta innovadora que se compromete a optimizar el rendimiento del control y gestión del sistema GCS, al momento de realizar la emisión de facturas a los usuarios, puesto que con la aplicación móvil se garantiza la satisfacción del cliente al momento de realizar los pagos, ya que con la entrega de las notificaciones, este ya sabrá el valor exacto a pagar.

4.2.1.6 Alcance del Proyecto Propuesto

4.2.1.6.1 Descripción de los Sistema Propuestos

Asignación del Gerente del Proyecto

Puesto que el Gerente de un proyecto debe poseer cualidades de líder, además de poseer conocimientos profundos de las herramientas que van a ser usadas en la creación de la solución, se ha definido como Gerente de este proyecto al Ing. Wladimir Casto, que a la vez es nuestro Director de Tesis, quién tendrá la dura tarea de guiar el proyecto hacia su término.

Planes iniciales para el desarrollo del Proyecto

Para determinar si el proyecto es viable, se debe determinar los factores básicos necesarios, los mismos que nos permiten tener una visión general de los recursos que serán analizados.
Estos recursos son: Económicos, Operativos y Tecnológicos.

Estrategia designada para el desarrollo del Proyecto

Luego de varias reuniones entre los integrantes del grupo de desarrollo se tomó la decisión de utilizar la estrategia Divide y Vencerás.

Justificación de la Estrategia

Divide y Vencerás, da las prestaciones requeridas para el desarrollo de este proyecto, puesto que el mismo puede ser divido en varias etapas, las cuales pueden ser tratadas por separado proveyendo un continuo avance en el desarrollo de la aplicación.

Fuentes de presupuesto

La municipalidad de Baños es la encargada de cubrir los gastos completos de la realización del Proyecto.

4.2.1.7 Ciclo de Vida del Proyecto

4.2.1.7.1 Estudio de viabilidad

Introducción

El Estudio de Viabilidad se especificará al nivel de detalle exigido, para evaluar si existe una solución factible o para seleccionar la más apropiada. El detalle de los requisitos, los riesgos, los planes, los costos, etc. para la solución se desarrollará en las fases posteriores.

Objetivos

- Establecer si una propuesta de desarrollo puede reunir los requisitos de la organización.
- Evaluar la conveniencia de la aplicación al desarrollo de DSDM
- Perfilar las posibles soluciones técnicas al problema comercial
- Obtener la primera estimación de tiempo de desarrollo y costos

4.2.1.7.2 Precondiciones

Acuerdo respecto del alcance de investigación.

Acuerdo inicial de la definición del problema comercial a ser solucionado.

Para el desarrollo de esta etapa se realizó varias con las partes implicadas, en este caso con nuestro director de tesis las mismas que se efectuaron diariamente en los meses de enero y febrero del 2009.

A continuación se presenta el desarrollo de lo acordado previamente:

4.2.1.7.3 Productos

Se especificaran una serie de interrogantes que nos permitirán determinar la viabilidad de la propuesta de solución:

4.2.1.7.4 Criterios de calidad

¿La definición del problema está acorde con las necesidades del contratista?

Si, debido a que se lo analizo de forma conjunta y con aceptación de ambas partes.

¿El alcance del proyecto es suficientemente claro para ser refinado dentro del Estudio Comercial?

Si, puesto que la propuesta es en si la refinación de un sistema existente y cuyo objetivo es incluir un sector de los usuarios (clientes registrados en el sistema GCS) que no han podido acceder a las utilidades del mismo y que han tenido inconvenientes con este sistema al momento de realizar los pagos en las ventanillas de cobros de los departamentos.

¿Están los objetivos comerciales claramente definidos?

Cada miembro del equipo de trabajo esta consciente de sus tareas y los objetivos que ellas persiguen, los mismos que en su conjunto buscan la plena satisfacción del contratistas, es este caso las municipalidad de Baños, específicamente el Departamento de Agua Potable.

¿La solución al problema es factible a las condiciones técnicas y comerciales?

Técnicamente como Tesistas utilizaremos recursos propios del contratista, por lo cual se sine a la realidad tecnológica de la misma.

Comercialmente la propuesta es lo bastante factible como para ser aceptada por el contratista, debido a que su costo y tiempo de entrega o implantación es el apropiado a un proyecto de estas características.

¿Los riesgos son aceptables después de verificar la Lista de Riesgos?

Todo proyecto incurre en una serie de riesgos, sean estos generados por los desarrolladores, contratistas o por las características mismas del problema. Debida a que la ponderación de los riesgos es de media a baja, se considera que dichos riesgos son aceptables.

¿El contratista ha aceptado lo que ha sido incluido y excluido del alcance?

Si, debido a que la propuesta se ha basado en sus requerimientos mínimos.

4.2.1.7.5 Viabilidad Económica

Las instituciones implicadas en el desarrollo de este proyecto se han visto en la necesidad de costear los gastos para el personal de desarrollo, en este caso, los dos Tesistas y el director de los mismos, por los que se han tomado en cuenta algunos factores para la instalación y pruebas de la aplicación en sus departamentos de agua potable.

Tiempo de Permanencia: 1 semana

Actividad	**Tiempo**	
Acoplamiento del servidor Central		
Instalación del sistema Sysnycs en los Pocket PC		
Instalación del sistema SisTrad en el Servidor Central	1 Día	Estadía 3 Personas
Generación de Códigos de Barras para los usuarios		
Pruebas de cálculos		
Impresión de códigos de Barras	1 Día	Estadía 3 Personas
Pruebas de Transferencia de Datos PC-Pocket		
Pruebas de Transferencia de Datos Pocket-PC	1 Día	Estadía 3 Personas
Pruebas para la impresión de Datos		
Comparación de Resultados	1 Día	Estadía 3 Personas
Pruebas de campo	1 Día	Estadía 3 Personas
Capacitación del manejo de los sistemas	1 Día	Estadía 3 Personas

Tabla 15. Viabilidad Económica

4.2.1.7.6 Beneficios

- Los beneficios se verán reflejados una vez que se optimice el sistema de cobros para el sistema GCS, puesto que el mismo contará con lecturas más precisas
- El cliente puede verificar que la lectura que está impresa en su notificación, es la misma que tiene su medidor de agua potable.
- La notificación es impresa y entregada a los clientes en 8 segundos. Disminución de reclamos por posibles errores de lecturas de medidores y entrega de facturas.
- Aumenta la ventaja competitiva de los Departamentos de Agua Potable de los municipios en mención con el resto de empresas prestadoras de servicios públicos domiciliarias a nivel Ecuador.
- Aumenta la confiablidad de los clientes para con el Departamento de Agua Potable, al pagar por el consumo exacto.

- El cliente llevara la cantidad exacta a pagar en los centros de recaudación una vez conocido el valor.

4.2.1.7.7 Viabilidad Tecnológica

Para el desarrollo de la aplicación completa con todos su módulos se utilizarán los recursos tecnológicos del AME (Asociación de Municipalidades del Ecuador), puesto que estas serán desarrolladas en primera instancia en Emuladores que corren sin ningún inconveniente en PCs.

Hardware

- 2 PCS para el desarrollo de la programación y documentación.
- Acceso al Internet con un ancho de banda de 2,5 Mbps

Software

- Contamos con la Base de Datos del Sistema GCS
- El Sistema GCS
- Herramientas de desarrollo (.Net, SQL Server, SQL Server Ce, MS Office, Racional Rose, StarUML, ER Estudio, utilitarios).

4.2.1.7.8 Análisis de riesgos

Identificación del Riesgo

A través de la experiencia del equipo de desarrollo se ha podido identificar los siguientes riesgos y amenazas:

RIESGOS
R1: Incumplimiento de los objetivos
R2: Incumplimiento del cronograma
R3: Falta de comunicación entre los miembros del equipo de trabajo
R4: Problemas de acceso a las bases de datos
R5: Insuficiente información para el desarrollo del trabajo
R6: No Adquisición de equipos móviles para realizar las pruebas
R7: Insatisfacción de los usuarios
R8: Falta de disponibilidad del personal técnico

Tabla 16. Identificación de Riesgos

4.2.1.7.9 Categorización del Riesgo

RIESGOS	PROBABILIDAD			DEVERIDAD		
	ALTA	MEDIA	BAJA	ALTA	MEDIA	BAJA
R1			X	X		
R2		X			X	
R3		X			X	
R4	X			X		
R5		X		X		
R6		X			X	
R7			X	X		
R8		X			X	

Tabla 17. Categorización del Riesgo

4.2.1.7.10 Gestión del Riesgo

RIESGO	PLAN DE GESTION
R1	**Problema:** Sistemas no cumplen con las expectativas ni con los objetivos planteados. **Actividades a desarrollar:** • Seguir y evaluar continuamente el avance del proyecto.

	• Interactuar continuamente con los contratistas y los desarrolladores **Responsable:** Gestor del proyecto (Ing. Wladimir Castro)
R2	**Problema:** Retraso en la entrega del producto software final. **Actividades a desarrollar:** • Planificar entregas parciales del trabajo desarrollado. • Controlar continuamente el avance del proyecto en base a la planificación inicialmente planteada. **Responsable:** Gestor del proyecto (Ing. Wladimir Castro)
R3	**Problema:** Incapacidad de llegar a consensos. **Actividades a desarrollar:** • Evaluar las capacidades de liderazgo entre los miembros del equipo. • Designar a un miembro que lidere las actividades de todos los miembros. **Responsable:** Gestor del proyecto (Ing. Wladimir Castro)
R4	**Problema:** Al no tener las facilidades para acceder a la lógica de la base de datos “dbame”, no podremos interactuar con el sistema GCS. **Actividades a desarrollar:** • Interrogar a los usuarios del sistema GCS. • De ser posible indagar con los desarrolladores de la base de datos dbame. • Analizar los datos arrojados luego de la ingeniería inversa. **Responsable:** Programador Analista (Personal Tesis)
R5	**Problema:** Mal enfoque de la solución del problema **Actividades a desarrollar:** • Planificar reuniones de trabajo para abstraer los verdaderos requerimientos del cliente. **Responsable:** Gestor del proyecto (Ing. Wladimir Castro)
R6	**Problema:** La no adquisición o demora en la compra de los equipos móviles por parte de las municipalidades en cuestión. **Actividades a desarrollar:**

	• Realizar los trámites necesarios con las municipalidades para la compra de los equipos. **Responsable:** Gestor del proyecto (Ing. Wladimir Castro)
R7	**Problema:** Desperdicio de los recursos empleados en una aplicación no utilizada. **Actividades a desarrollar:** • Identificar y evaluar permanentemente el cumplimiento de los requerimientos de los usuarios. **Responsable:** Programadores (Personal Tesis)
R8	**Problema:** Atraso en los cronogramas establecidos y en la respectiva entrega de los avances. **Actividades a desarrollar:** • Establecer políticas internas de control. • Fomentar el espíritu de responsabilidad. **Responsable:** Gestor del proyecto (Ing. Wladimir Castro)

Tabla 18. Gestión de Riesgos

4.2.1.8 Estudio del negocio

4.2.1.8.1 Objetivos

- Perfilar el desarrollo futuro en términos de entrega de prototipos y control del prototipado.

- Identificar a representantes de la clase usuario para las actividades de prototipado.

- Priorizar los requisitos del sistema propuesto

- Reimponer los riesgos del proyecto

- Mantener una base firme para el desarrollo técnico de los procedimientos

- Alcance de los requisitos no funcionales, en particular decidir los requisitos del mantenimiento.

4.2.1.8.2 Precondiciones

Acuerdo del Informe de Viabilidad, incluso el acuerdo de la viabilidad del desarrollo y de la pertinencia o aplicabilidad al proyecto de DSDM.

Para determinar los requerimientos se desarrollaron reuniones diarias con el gestor del proyecto en las cuales se logró acoger todas las necesidades y sugerencias entre los administradores, usuarios y desarrolladores.

A continuación se presenta el desarrollo de lo estudiado:

4.2.1.9 Productos

4.2.1.9.1 Definición del Área comercial

Debido a que el objetivo del proyecto es complementar un sistema existente, no tendría un valor comercial en el mercado, debido a que es complementario a este problema en particular.

4.2.1.9.2 Lista de Requisitos priorizada

Los requisitos, en orden de prioridad son:

Rq (1) Actualizar a nivel de servidor los datos ingresados de un nuevo cliente en el sistema GCS.

Rq (2) Migrar los datos de los clientes del sistema GCS, encontrados en la tabla Catastros una aplicación diseñada para dispositivos móviles, la cual funcionará en un principio en emuladores con una base de datos portable.

Rq (3) Migrar la información de las lecturas del último mes de los clientes para de esta manera tener datos para la realización de los cálculos correspondientes, estos datos se encuentran localizados en la tabla Lecturas del Sistema GCS.

Rq (4) Esquematizar y entender los distintos sistemas y categorizaciones de cobros para los clientes, de acuerdo al siguiente criterio:

- Sistema de Cobro con Valores Fijos para el cobro del servicio de Agua Potable, Alcantarillado y Basura.
- Sistema de Cobro con Valor Diferenciado.

Rq(5) Esquematizar y clasificar los tipos de datos y campos que se tomarán en cuenta para la realización de los distintos cálculos para la obtención de los valores a pagar por el consumo de los servicios básicos, de agua, basura, alcantarillado y deudas por contratos de compra de medidores.

Rq(6) Crear un modulo en C# que calcule el valor por consumo de Agua Potable, Basura y Alcantarillado de un determinado usuario.

Rq(7) Crear un modulo en C# que calcule el valor a pagar por concepto de deudas anteriores y por contratos de compra de medidores de un determinado usuario.

4.2.1.10 Plan de Desarrollo

4.2.1.10.1 Organización de equipo del proyecto

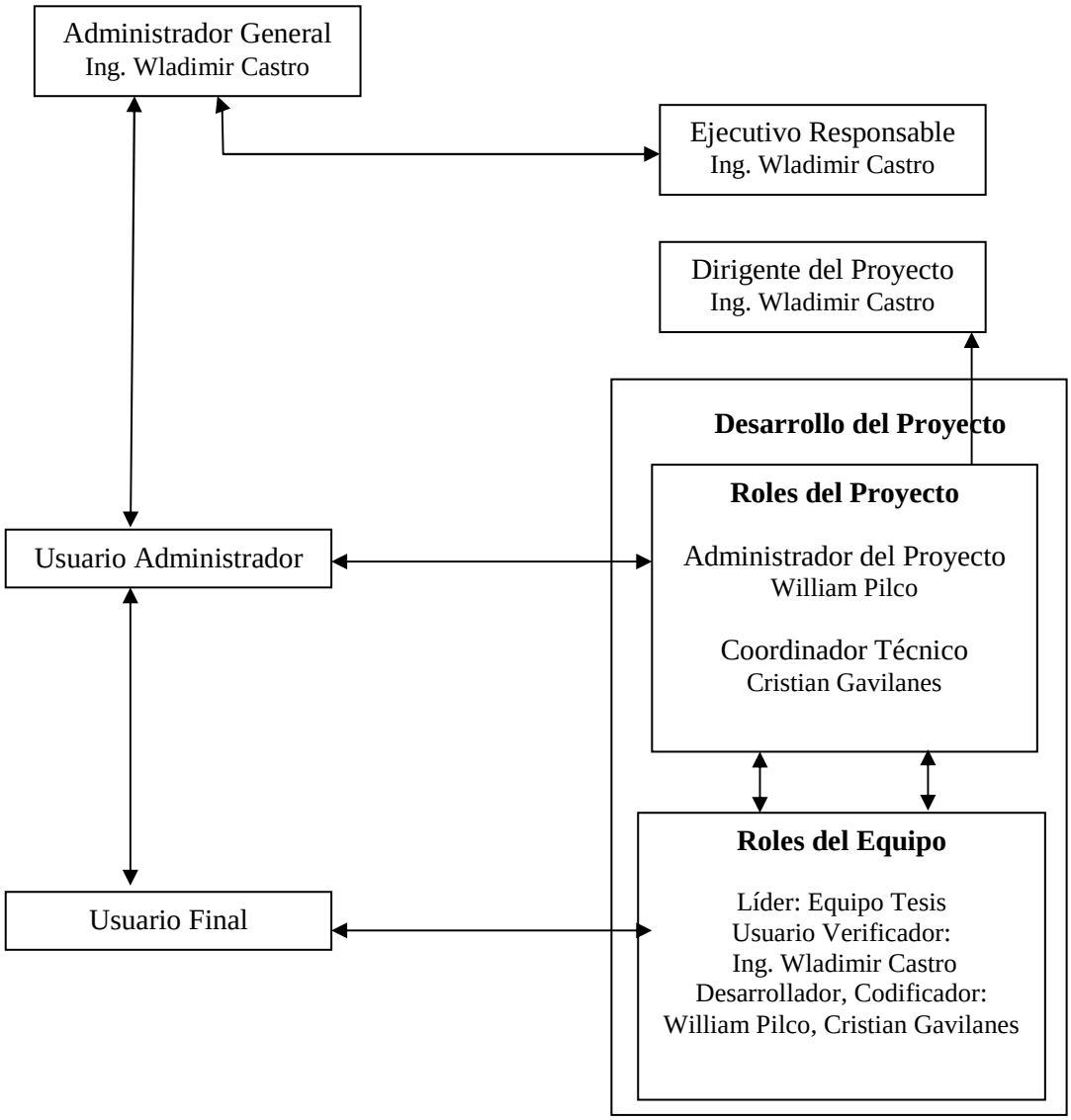


Figura 56. Organización del equipo del Proyecto

4.2.1.10.2 Cronograma de Trabajo

	Nombre de tarea	Duración	Comienzo	Fin
1	Proyecto	7 días	lun 01/12/08	mar 09/12/08
2	**Ciclo Vital del Proyecto**	**139 días**	**mié 10/12/08**	**lun 22/06/09**
3	Estudio de la Viabilida	7 días	mié 10/12/08	jue 18/12/08
4	Estudio del Negocio	9 días	vie 19/12/08	mié 31/12/08
5	Iteración Modelo Func	3 días	jue 01/01/09	lun 05/01/09
6	Diseño e Iteración de	70 días	mar 06/01/09	lun 13/04/09
7	Puesta en Práctica	50 días	mar 14/04/09	lun 22/06/09
8	Post Proyecto	15 días	lun 17/08/09	vie 04/09/09

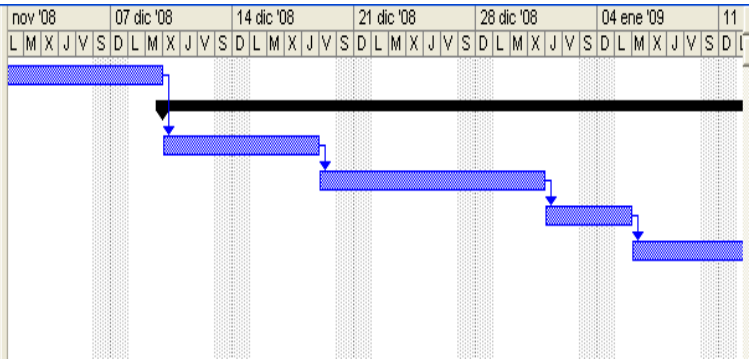


Figura 57. Cronograma de Trabajo

4.2.1.10.3 Costos

Id	Nombre del recurso	Tipo	Etiqueta de material	Iniciales	Grupo	Capacidad máxima	Tasa estándar	Tasa horas extra	Costo/Uso	Acumular	Calendario base	Código
1	Desarrolladores	Trabajo		D		25%	$8.00/hora	$0.00/hora	$640.00	Prorrateo	Estándar	
2	Movilizacion	Trabajo		M		25%	$0.50/hora	$0.00/hora	$20.00	Prorrateo	Estándar	
3	Material Bibliográfico	Trabajo		M		25%	$0.22/hora	$0.00/hora	$70.00	Prorrateo	Estándar	
4	Papelería	Trabajo		P		25%	$0.47/hora	$0.00/hora	$150.00	Prorrateo	Estándar	
5	Refrigerios	Trabajo		R		25%	$0.63/hora	$0.00/hora	$200.00	Prorrateo	Estándar	
6	10 % de Servicios Presta	Trabajo		SP		25%	$0.34/hora	$0.00/hora	$108.00	Prorrateo	Estándar	

Tabla 19. Análisis de Costos

4.2.1.10.4 Vista de Despliegue de PEINSEB

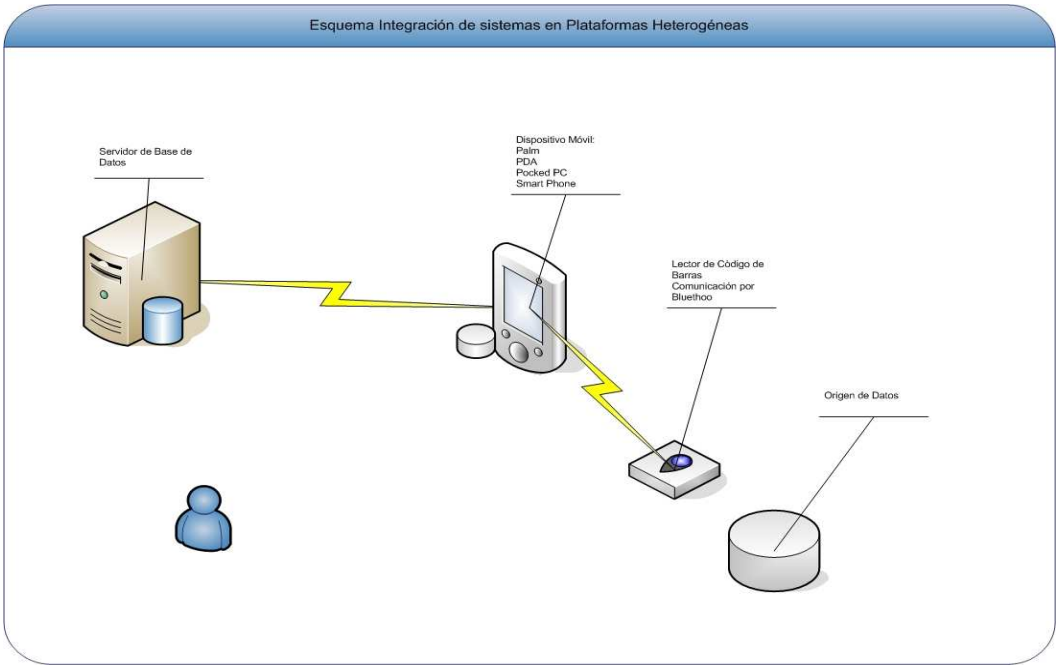


Figura 58 Vista de Despliegue de PEINSEB

4.2.1.11 Iteración del modelo funcional

4.2.1.11.1 Introducción

Una vez recogidos los requerimientos se procede a plasmarlos en prototipos; los mismos que serán probados por los usuario. Esta iteración de los prototipos con los usuarios nos permitirá mejorar el diseño e ir añadiendo nuevas funcionalidades a nuestro proyecto.

4.2.1.11.2 Objetivos

- Demostrar la funcionalidad requerida en un modelo operacional, mediante prototipos basados en los requerimientos.

- Determinar los requerimientos que no se han logrado cubrir con éxito, para que en el siguiente prototipo se los implemente.

4.2.1.11.3 Precondiciones

Establecer un acuerdo con los usuarios y administradores del sistema actual para determinar los requerimientos que deben ser cumplidos.

La interfaz del prototipo debe ser parecida a la de la aplicación final, puesto que el usuario que la va a usar se va a familiarizar con la misma.

Fijar un tiempo prudencial para el desarrollo del prototipo, este tiempo debe ser establecido entre el gerente del proyecto y los usuarios del sistema.

4.2.1.11.4 Productos

Identificar el producto funcional

Determinar las funcionalidades para ser puesto en ejecución en el prototipo, los resultados de esta iteración nos darán como resultado una aplicación funcional.

4.2.1.11.5 Analizar los requisitos

Anteriormente se especificó en orden de prioridad los requisitos, en esta etapa se procederá a determinar todos los recursos necesarios para el cumplimiento de este requerimiento:

Rq (1) Para realizar la actualización, primeramente se desarrolla un servidor vinculado, además es importante tener en cuenta que para poder realizar consultas distribuidas, se debe tener

corriendo en el servidor de SQL server, y empezar a trabajar con el servidor SQL Ce, para realizar la aplicación móvil.

Recursos Necesarios:

Para iniciar esta etapa se debe tener instalado el SQL Server, Actualizar a nivel de servidor los datos ingresados de un nuevo cliente en el sistema GCS.
Además como se mencionó anteriormente se deberá tener instalado el servidor Sql Ce, para trabajar con la Aplicación Móvil.

Rq (2) Migrar los datos de los clientes del sistema GCS, encontrados en la tabla Catastros una aplicación diseñada para dispositivos móviles, la cual funcionará en un principio en emuladores con una base de datos portable.
Tener instalado además Sql Ce, para trabajar con la aplicación móvil.

Recursos Necesarios:

Para poder ejecutar este requerimiento es primordial que se tenga instalado todas las herramientas de desarrollo con sus librerías adicionales.
Tener instalado además Sql Ce, para trabajar con la aplicación móvil.

Rq (3) Migrar la información de las lecturas del último mes de los clientes para de esta manera tener datos para la realización de los cálculos correspondientes, estos datos se encuentran localizados en la tabla Lecturas del Sistema GCS.

Recursos Necesarios:

Para poder ejecutar este requerimiento es primordial que se tenga instalado todas las herramientas de desarrollo con sus librerías adicionales.

Tener instalado además Sql Ce, para trabajar con la aplicación móvil.

Rq (4) Esquematizar y entender los distintos sistemas y categorizaciones de cobros para los clientes, de acuerdo al siguiente criterio:

- Sistema de Cobro con Valores Fijos para el cobro del servicio de Agua Potable, Alcantarillado y Basura.

- Sistema de Cobro con Valor Diferenciado.

Recursos Necesarios:

Para poder ejecutar este requerimiento es indispensable haber realizado la ingeniería inversa a la Base de Datos, y además de haber tenido reuniones con el grupo de desarrolladores del sistema GCS, para de esta forma entender los distintos algoritmos para el cálculo de los rubros.
Tener instalado además Sql Ce, para trabajar con la aplicación móvil.
Haber instalado el software necesario para realizar la ingeniería inversa a una base de datos.
También es fundamental tener instalada y corriendo la aplicación GCS, para entender de una mejor manera el funcionamiento de dicho sistema.

Rq(5) Esquematizar y clasificar los tipos de datos y campos que se tomarán en cuenta para la realización de los distintos cálculos para la obtención de los valores a pagar por el consumo de los servicios básicos, de agua, basura, alcantarillado y deudas por contratos de compra de medidores.

Recursos Necesarios:

Para poder ejecutar este requerimiento es indispensable haber realizado la ingeniería inversa a la Base de Datos, y además de haber tenido reuniones con el grupo de desarrolladores del sistema GCS, para de esta forma entender los distintos algoritmos para el cálculo de los rubros.

Tener instalado además Sql Ce, para trabajar con la aplicación móvil.

Haber instalado el software necesario para realizar la ingeniería inversa a una base de datos.

También es fundamental tener instalada y corriendo la aplicación GCS, para entender de una mejor manera el funcionamiento de dicho sistema.

Rq(6) Crear un modulo en C# que calcule el valor por consumo de Agua Potable, Basura y Alcantarillado de un determinado usuario.

Recursos Necesarios:

Para poder ejecutar este requerimiento es primordial que se tenga instalado todas las herramientas de desarrollo con sus librerías adicionales, en estas se tomarán en cuenta principalmente que se tengan instalados y funcionando los emuladores para Pocket PC, y el lenguaje C# de .Net.

Tener el claro los algoritmos de cálculo para los rubros por consumo de los servicios básicos.

Tener instalado además Sql Ce, para trabajar con la aplicación móvil.

También es fundamental tener instalada y corriendo la aplicación GCS, para entender de una mejor manera el funcionamiento de dicho sistema.

Rq(7) Crear un modulo en C# que calcule el valor a pagar por concepto de deudas anteriores y por contratos de compra de medidores de un determinado usuario.

Recursos Necesarios:

Para poder ejecutar este requerimiento es primordial que se tenga instalado todas las herramientas de desarrollo con sus librerías adicionales, en estas se tomarán en cuenta principalmente que se tengan instalados y funcionando los emuladores para Pocket PC, y el lenguaje C# de .Net.

Tener el claro los algoritmos de cálculo para los rubros por consumo de los servicios básicos.

Tener instalado además Sql Ce, para trabajar con la aplicación móvil.

También es fundamental tener instalada y corriendo la aplicación GCS, para entender de una mejor manera el funcionamiento de dicho sistema.

Requisitos de la lista de las iteraciones realizadas

SISNYCS V.1.0

En esta versión los desarrolladores se centraron en la prueba de los recursos entregados por parte de los departamentos de Agua potable del Municipio de Baños, además de la implementación de una parte inicial del módulo.

Como segundo punto se paso al desarrollo del problema de la vinculación de las dos bases de datos para así poder realizar la actualización de datos de un cliente registrado.

Para la implementación de esta versión se tomaron los requerimientos: Rq1, Rq2 y Rq3 establecidos y analizados anteriormente.

SISNYCS V.2.0

Continuando con el desarrollo se procedió en esta versión a implementar y adjuntar al módulo anterior; los cálculos necesarios para el cobro de los servicios de Agua Potable, Basura y Alcantarillado.

Para la implementación de esta versión se tomaron los requerimientos: Rq4, Rq5 y Rq6 establecidos y analizados anteriormente.

SISNYCS V.3.0

Para finalizar con la implementación, esta versión contiene los módulos anteriormente desarrollados junto con el cálculo del valor a pagar por concepto de deudas anteriores y por contratos de compra de medidores de un determinado usuario.

Para la implementación de esta versión se toma el requerimiento: Rq7 establecido y analizado anteriormente.

SISTRAD V.1.0

En esta versión los desarrolladores se centraron en la prueba de los resultados obtenidos en la funcionalidad de las previas versiones de SISNYCS, puesto que la transferencia bidireccional del flujo de datos está centrada en el correcto funcionamiento de SGCS y de SYSNICS con la correcta aplicabilidad y escalabilidad de los requerimientos que lo componen.

4.2.1.8.1 Convenir el horario

Entre el grupo de desarrolladores, será necesario acordar un horario en el cual se genere el prototipo especificado. Para la fase de Pre-proyecto se tomará 7 días en los cuales los desarrolladores se reunirán 2 horas diarias para definir tareas, dar sugerencias y tomar decisiones.

Para la fase del Ciclo de vida del proyecto se tomará 46.5 días en los cuales los desarrolladores se reunirán 2 horas diarias para definir tareas, dar sugerencias y tomar

decisiones. Para la fase Post-proyecto se tomará 46.5 días en los cuales los desarrolladores se reunirán 2 horas diarias para definir tareas, dar sugerencias y tomar decisiones.

4.2.1.11.6 Crear el prototipo funcional

Investigar

Cada vez que se desarrolle un nuevo prototipo es necesario analizar los anteriores, para de esta manera tener una base para tratar al máximo de evitar errores en futuras versiones del presente sistema.

SISNYCS V.1.0

Esta versión nos provee adecuadamente cada uno de los datos registrados de cada uno de los clientes del sistema GCS.
A pesar de su adecuada funcionalidad se nos presenta se debe primero presionar el botón aceptar para que estos datos se muestren.

SISNYCS V.2.0

Este prototipo funciona bien, además se ha logrado mostrar los datos de los usuarios registrados sin pulsar ningún botón, y controlar el ingreso de datos que no sean números.

SISNYCS V.3.0

En esta última versión se encontraron pequeños errores los cuales fueron gestionados en cada uno de los módulos anteriores, logrando así su adecuada funcionalidad, al mostrarnos cada uno de las vistas previas de las notificaciones enviadas.

SISTRAD V.1.0

Esta versión automatiza la transferencia entre medios heterogéneos vía TCP.

Refinar

SISNYCS V.1.0

Para mejorar la funcionalidad de prototipo es necesario conseguir todas las librearías necesarias para que la herramienta de desarrollo funcione correctamente, en este caso se acoplarán e integrarán librerías para que se permita la impresión mediante una impresora con puerto serial.

SISNYCS V.2.0

Para solucionar este problema se recurrió a descargar librerías gratuitas, que funcionan a manera de clases en c#, las cuales permitirán la conectividad con medios externos conectados a través del puerto serie, para poder realizar la impresión de notificaciones.

SISNYCS V.3.0

Luego de haber corregido las versiones anteriores, como tesistas contamos con una versión final, la cual cumple a cabalidad todos los requerimientos establecidos inicialmente

SISTRAD V.1.0

Luego de varias pruebas sobre SISNYCS y sobre SISTRAD se culmina con el refinamiento de la aplicación, puesto que se cumplió con los requerimientos solicitados.

Consolidar

Unir cada uno de los prototipos realizados anteriormente con el actual, para finalmente obtener nuestro módulo completo.

Repasar el prototipo funcional

Probar el Prototipo

Para esta etapa se procede a solicitar una reunión entre usuarios y desarrolladores para que las personas que van a manejar el prototipo presentado puedan manipularlo y exponer un criterio, el cual nos sugiera cambios o posibles modificaciones, todas estas pruebas se realizarán en los emuladores respectivos, en los cuales se encuentra corriendo la aplicación.

Prototipo de la Revisión

Se recoge las sugerencias vertidas por los diferentes tipos de usuarios para plasmarlos en el sistema y adjuntarlos estas mejoras en la documentación.

4.2.1.11.7 Diseño e iteración de la estructura

Introducción

En esta fase se integrarán los componentes que han sido desarrollados individualmente junto con su interfaz, para así obtener el sistema completo SISNYCS, y el sistema completo SISTRAD, y poder consolidarlos en un solo proyecto, además se desarrollarán las pruebas respectivas para determinar la existencia de fallos en los sistemas y también se analizará si la interfaz presentada es amigable o necesita cambios.

Objetivos

- Integrar los componentes desarrollados en una sola aplicación funcional
- Comprobar la funcionalidad del software integrado anteriormente.
- Realizar cambios en la aplicación de ser necesarios.
- Modificar la interfaz de usuario en caso de ser necesario.

Prerrequisitos

Como prerrequisito es fundamental que los componentes que han sido desarrollados funcionen correctamente, evitando así, fallas innecesarias al momento de integrar dichos componentes.

Además previamente se debe definir la estructura de la interfaz de usuario, para que todos los prototipos que vayan a ser generados tengan la misma visualización.

Producto

Se integrarán los componentes funcionales de la fase anterior en un sistema que satisface necesidades del usuario.

Una vez integrados los módulos se realizarán pruebas continuas, para verificar su adecuado funcionamiento.

En caso de que el usuario final no esté de acuerdo con algunos puntos realizados por nuestro sistema, se asignará a uno o varios miembros del equipo de desarrolladores 3 días para llevar a cabo una revisión de dichos aspectos.

La interfaz ha sido definida de acuerdo al usuario final.

SISNYCS V.1.0

4.2.1.11.8 Pantalla de Inicio

El primer formulario, es un splash, en el cual se mostrará el progreso de carga de la aplicación.

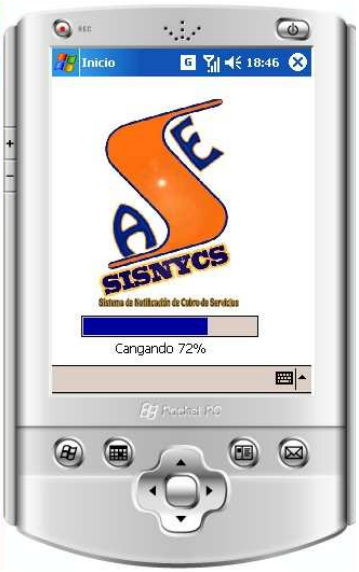


Figura 59 Pantalla Splash, SISNYCS

4.2.1.11.9 Pantallas de Ingresos de Datos

La siguiente pantalla muestra un formulario de ingreso de datos, que está dividido en dos partes, para el ingreso de datos como; el número de cuenta del usuario (dato ingresado a través del lector óptico de código de barras o manualmente), Apellidos, Nombres, Lectura Anterior, Dirección, Tipo (datos generados automáticamente luego de haber ingresado el número de cuenta), Lectura Actual, Comentario, Observación (datos a ser ingresados de forma manual por el usuario dependiendo del estado actual del medidor).

Figura 60. Pantallas de ingreso de Datos SISNYCS

Estas pantallas muestran además tres opciones principales: Grabar, Cancelar e Imprimir.

SISNYCS V.2.0

Pantalla de Ingreso de Datos

Figura 61 Control de Datos Ingresados Incorrectamente SISNYCS

Figura 62. Control de mensajes SISNYCS

Permite al usuario registrar todos los datos ingresados de una lectura actual, en caso de que esta sea ingresada por primera vez, y de actualizar los datos en caso de haber sido ingresada por más de dos ocasiones.

SISNYCS V.3.0

Pantalla de Impresión

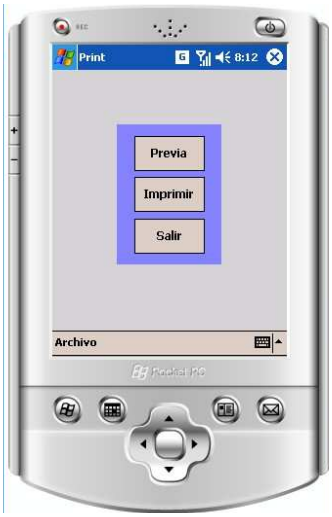


Figura 63 Pantalla de Impresión SISNYCS

Mediante esta opción primeramente se realizan una serie de cálculos necesarios para la impresión de las notificaciones, para luego presentar un nuevo formulario en el cual se muestran nuevas opciones referentes a la impresión del reporte antes evaluado.

Vistas Previas

Figura 64. Vistas Previas SISNYCS

Al elegir esta opción se mostrará una vista previa de la notificación a ser impresa, en la cual se podrán observar los valores ser cancelados por diversos conceptos, además de los datos personales del usuario a ser cobrado. Para regresar a la pantalla anterior se debe simplemente pulsar el botón **Atrás.**

Impresión de Notificaciones

Mediante opción se enviará a la impresora portátil, la notificación previamente observada y aprobada por el usuario

SISTRAD V.1.0

Figura 65. Pantalla de inicio SISTRAD

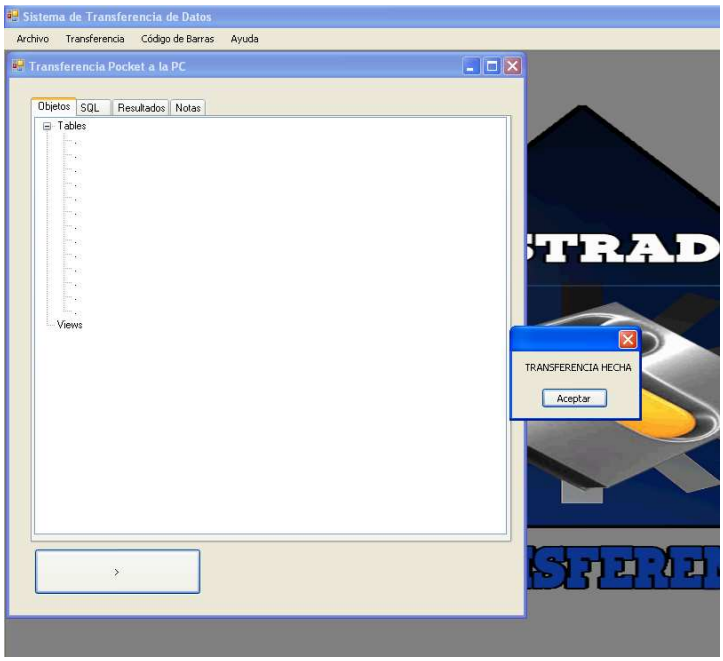


Figura 66. Pantalla Transferencia Pocket a la PC

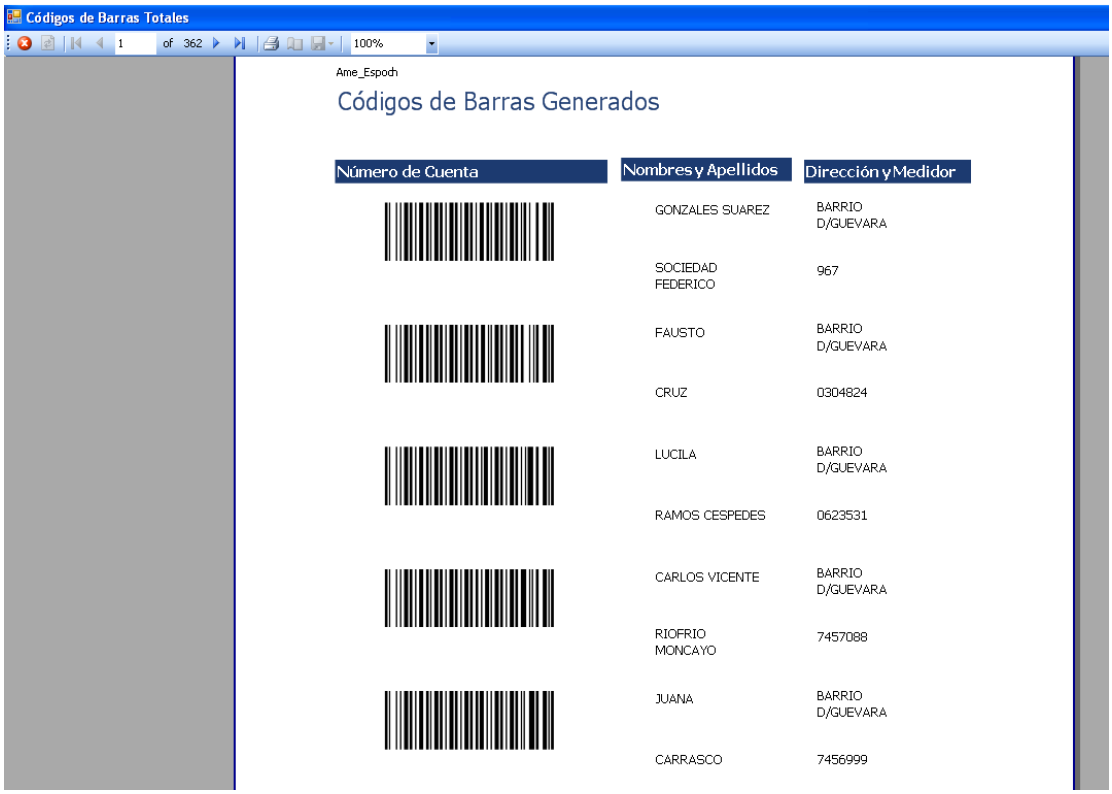


Figura 67. Código de Barras Generadas

4.2.1.11.10 Implementación

Introducción

Una vez superadas las fases anteriores, el proyecto final será entregado a los usuarios en su ambiente de trabajo, para la realización de pruebas y así determinar posibles fallas o mejoras en los requerimientos.

También será entregada la documentación, que en este caso vendrían a ser el Manual de Usuario, el Manual Técnico y el Manual de Administración y Gestión.

En caso de ser necesario se realizará una capacitación a las personas que manejarán la aplicación.

Objetivos

Realizar las pruebas de la aplicación necesarias en el ambiente de trabajo de los usuarios, para determinar la existencia de cambios y mejoras.

Instruir a los usuarios para el manejo de la nueva aplicación

Capacitar a los administradores del sistema para que puedan brindar un servicio de soporte.

4.2.1.8.1.1 Precondiciones

Iniciar la fase de pruebas para llegar a un acuerdo entre los técnicos Tesistas y los usuarios del proyecto, determinando así, la creación o no de nuevas funcionalidades; además de la corrección o mejora de las funciones ya implementadas, se especificará también un tiempo adecuado para la capacitación de los usuarios.

Productos

Documentación de usuario

En el documento que será creado para el usuario es necesario especificar la forma en la que este debe manejar dicho manual, además es necesario adjuntar las definiciones de los términos técnicos que no pueden ser conocidos por las personas que vayan a usar este manual. Como primer paso se le instruirá al usuario la forma en la que se debe manejar la aplicación, especificando su modo de uso mediante los menús principales.

Para una eficiente navegación por parte del usuario en el software entregado, se adjuntará un mapa del sitio creado.

Capacitación de usuarios

De acuerdo con el tipo de usuario que va a manejar nuestra aplicación se ha definido la realización de una capacitación al personal, la misma que tiene como objetivo aclarar las dudas sobre el manejo y funcionamiento de la nueva aplicación.

Capacitación Administradores y Usuarios Finales

La capacitación está prevista a realizarse en el periodo de una semana, a definirse por las partes interesadas.

Entrega del sistema

Luego de haber realizado las pruebas y el asesoramiento acordado, se procederá a entregar la aplicación a los departamentos de Agua Potable de la Municipalidad de Baños para que de inicio al uso del software entregado.

4.2.1.11.11 POST-PROYECTO

Introducción

Una vez implantado el sistema como tesistas nos aseguraremos de que el mismo opere eficazmente, para lo cual se realizarán mantenimientos, perfeccionamientos y gestión de riesgos.

Objetivos

- Mantener funcional al sistema entregado.

- Definir si los objetivos que se plantearon en las fases iniciales se han logrado a cumplir.
- Habilitar la posibilidad para la creación de nuevas versiones.
- Revisar el sistema entregado en funcionamiento.

Precondición

Para poder realizar el control del sistema entregado es necesario que el mismo se encuentre en funcionamiento.

Producto

Revisión de Post-Implementación

Conjuntamente con los usuarios se realizará una revisión de la aplicación en funcionamiento, para verificar si los objetivos han sido alcanzados en su totalidad, caso contrario se procederá a tomar sugerencias para posteriormente implementarlas.

4.2.1.11.12 Nuevas versiones

En cuanto a mantenimiento, de acuerdo al convenio realizado, por lo cual los cambios requeridos durante seis meses se los ejecutarán sin un costo previamente establecido en el contrato.

Al terminarse el tiempo convenido anteriormente para la realización de cambios, y en caso de que los departamentos municipales necesiten realizar alguna modificación, el costo por la misma deberá establecerse nuevamente, puesto que se trataría de una nueva versión de SISNYCS.

CAPITULO V

5 COMPROBACIÓN DE LA HIPOTESIS

5.1 HIPOTESIS

“La integración de sistemas en Plataforma Heterogéneas, mediante el uso de dispositivos hardware, mejoraran los procesos tradicionales para la recopilación de la Información manejada por las Municipalidades de Baños”.

5.2 Información Proporcionada

La información que se nos ha facilitado ha sido verificada y evaluada por el personal técnico de los Departamentos de Agua Potable, los mismos que de acuerdo a experiencias pasadas y tiempos evaluados han deducido lo siguiente:

Actualmente en las municipalidades en cuestión existe un tipo de facturación que la realiza el Departamento de Agua Potable, la Facturación Tradicional.

5.2.1 Facturación Tradicional

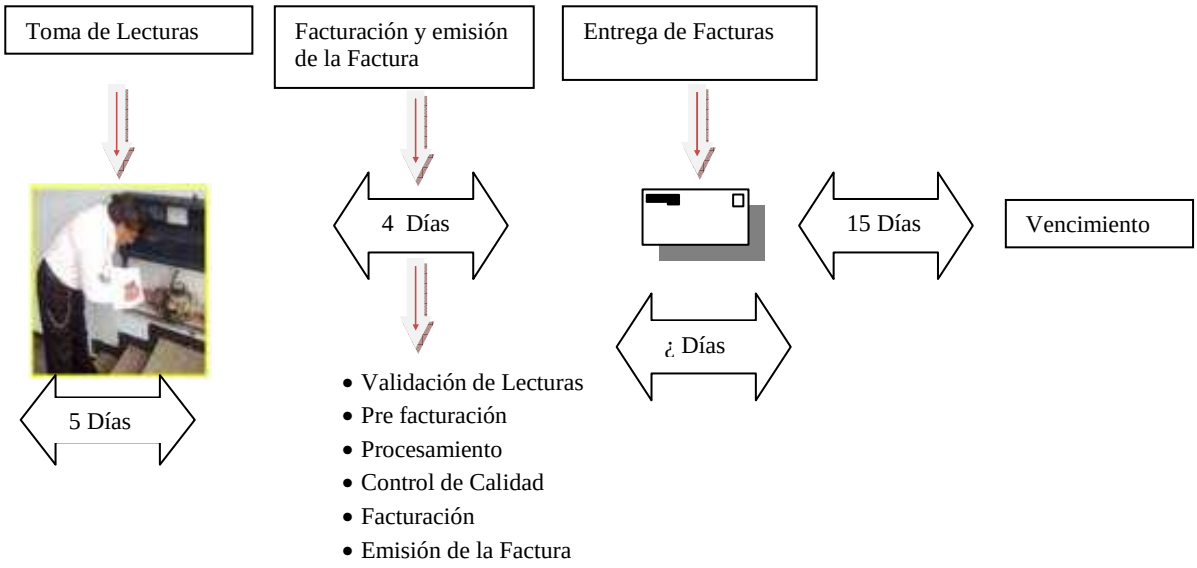


Figura 68. Sistema con Facturación Tradicional

5.2.2 Procesos del Sistema de Facturación Tradicional.

- Toma de lecturas de medidores.
- Emisión de facturas.
- Control de la emisión de facturas.
- Entrega de facturas.

5.2.2.1 Proceso De Toma De Lecturas

En este proceso, la Empresa, tiene contacto directo y permanente con los predios de los clientes, pues al existir aparatos de medición de flujos (hidrómetros o medidores de agua), estos deben ser “medidos” o “leídos”, lo que quiere decir que para llegar a saber con exactitud la cantidad de metros cúbicos de agua potable consumidos por un cliente en un determinado período se requiere verificar y anotar en reportes (listados), la cantidad que consta en el registro del medidor.

Cada inspector o lector es guiado mediante una ruta de recorrido previamente establecida, graficada y proporcionada por el Departamento de Agua Potable, en cuyo reporte constarán los medidores a ser leídos, así como también los datos de ubicación del predio, nombres del propietario, número de cuenta y demás datos relativos al medidor.

5.2.2.2 Proceso De Ingreso De Datos

Las lecturas de consumo y la información obtenida en el campo, se digitan en un archivo predeterminado, proceso que dura alrededor de 5 días y está sujeto a errores por parte de los digitadores debido a la gran cantidad de información obtenida. Es por eso que los empleados del Departamento de Agua Potable de las municipalidades en cuestión están sujetos a constantes reclamos por parte de los usuarios.

5.2.2.3 Proceso de determinación de los consumos.

Los consumos de los medidores de agua potable (que son instrumentos de precisión que registran gota a gota el agua que pasa desde la Red de Distribución hasta el interior del inmueble) son registrados y para efectos del cobro de servicios, el consumo de agua será determinado:

- Por el volumen registrado por el respectivo medidor domiciliario.
- Por el volumen estimado como consumo probable en la instalación domiciliaria, cuando no se tenga instalado un medidor.
- La lectura de los medidores es básica para el cobro justo de los consumos efectuados. Las lecturas son realizadas por lectores idóneos que garantizan su exactitud y confiabilidad.
- Antes de ser facturados, los consumos son sometidos a una crítica para determinar su normalidad, comprobar su exactitud y asegurar el cobro correcto de los servicios.

5.2.2.4 Proceso De Facturación

Este proceso comprende todas las operaciones necesarias para la creación, actualización y cancelación de las cuentas de los consumidores, y para el cálculo, ingreso y registro de la cuenta correspondiente de todos los valores provenientes, ya sea por la prestación directa de servicios así como de los pagos hechos por cada consumidor. Este proceso tiene una trascendencia fundamental, pues con base en sus resultados se puede dar fe pública y establecer la legalidad de todos los documentos y acciones que de él se deriven.

5.2.2.5 Emisión De Facturas

Es el conjunto de operaciones que se realizan para expresar los valores que debe cobrarse a cada consumidor, en un documento denominado factura.

Cuando la factura contiene como valor a ser pagado, solamente el correspondiente a los servicios y otras deudas ocurridas durante el período correspondiente al ciclo inmediatamente anterior (un mes), se denomina “facturación simple”. Cuando la factura contiene el valor del saldo deudor acumulado y ésta cancela todas las facturas anteriores, el proceso que la produce se llama “facturación con arrastre de la deuda”.

No obstante que en la facturación simple el saldo deudor no forma parte del valor total de la factura, es mucho más conveniente que en ésta aparezca como información para hacer presente la situación de atraso y se señale el número de meses o períodos en los que se halla vencido el cliente.

5.2.2.6 Control De La Emisión De La Facturación

Este control tiene por finalidad asegurar la facturación para su posterior cobranza del valor de los servicios prestados por la empresa, a todos los clientes. Se efectúa de la siguiente manera:

Verificando que el número de facturas emitidas por sector o ciclo de cada localidad y por características (categorías, agrupaciones, vencimientos), coincida con el número de

consumidores en actividad registrados en el catastro respectivo. Para esto es necesario mantener el registro del número de cuentas en actividad y ejercer el correspondiente control sobre los totales de nuevas cuentas, cancelaciones y supresiones.

Revisando la exactitud y correcta presentación del contenido de las facturas, mediante inspección y confrontación de una muestra de la facturación, para garantizar la veracidad de la cobranza justa de los valores correspondientes.

5.2.3 Proceso Con Facturación Inmediata

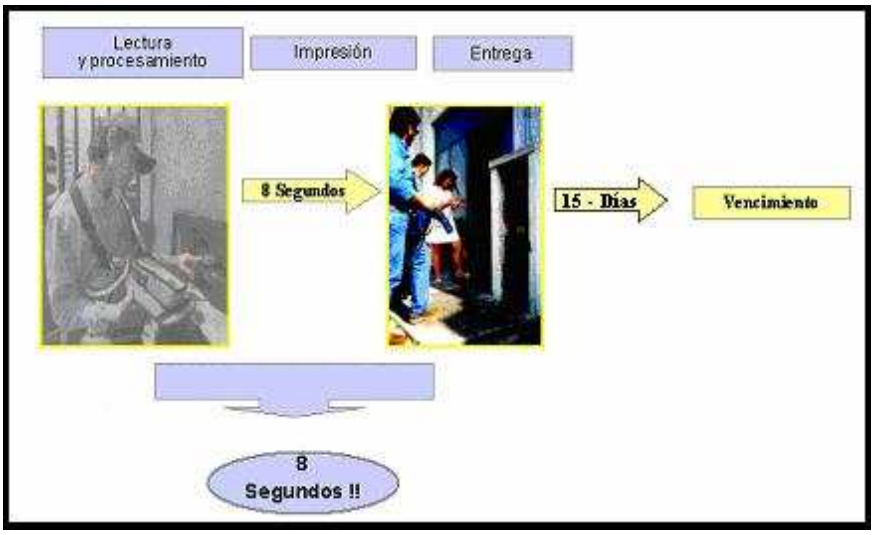


Figura 69. Sistema de Facturación Inmediata

Este sistema consiste en tomar la lectura del medidor del consumo de agua potable de todo el mes, anotarla en la “Terminal Portátil de Lectura” (TPL) previamente cargada con los datos del cliente y del medidor, además de datos de facturación del mes (cargos fijos, contratos, multas, ajustes, saldos atrasados, etc) y las funciones de cálculo correspondientes, los que luego de ser procesados y utilizando una impresora portátil se imprime una notificación en tiempo real en el domicilio del cliente, de tal manera que el consumidor puede comprobar que la lectura de su notificación, sea la misma que refleja el medidor.

5.3 Comprobación de la Hipótesis

Para la comprobación de la hipótesis anteriormente citada se ha realizado una evaluación en base a los resultados totales de los tiempos que toman los dos procesos para la recopilación de la información manejada por los Departamentos de Agua Potable de las Municipalidades, para su posterior Facturación y entrega de las mismas a los clientes.

Con el esquema planteado y con la hipótesis analizada con los tipos de variables examinadas se ha decidido por realizar un estudio para determinar la **Esperanza Matemática,** y de esta manera encontrar el **Tiempo Más Óptimo,** entre los tipos de procesos.

5.4 Determinación de las Variables a ser Analizadas

Se ha logrado identificar las variables que intervienen en el análisis para la comprobación de la hipótesis mencionada.

- **Variable Independiente:** Integración de Sistemas en Plataformas Heterogéneas, mediante el uso de dispositivos Hardware.
- **Variable Dependiente:** Proceso Tradicional para la recopilación de la información.

A continuación se indica el promedio total de horas al mes que se demora en la recopilación de la información con el Sistema Tradicional y el uso de dispositivos hardware. Para el cobro del Agua Potable por la Municipalidad de Baños.

RECOPILACION DE LA INFORMACION EN HORAS	PROCESO TRADICIONAL (horas)	USO DE DISPOSITIVOS HARDWARE (horas)
Tiempo de Lectura	120	120
Digitalización	96	0,08
TOTAL	**216**	**120,08**

Tabla 20. Promedio mensual en horas para la recopilación de datos usando los dos Sistemas

5.5 Análisis de los Datos.

En la siguiente tabla se puede apreciar la diferencia abismal en la demora de tiempos para el proceso de recopilación de la información.

USO DE DISPOSITIVO HARDWARE		
PROCESO TRADICIONAL	**TIEMPO LECTURA**	**DIGITALIZACION**
Tiempo de Lectura	0	-
Digitalización	-	95,92

Tabla 21. Comparación del Tiempo de Demora en recopilar la información del proceso tradicional vs. El uso del dispositivo hardware.

Observando la tabla de comparaciones si nos fijamos en la variable de tiempo de lectura la diferencia del tiempo del proceso tradicional menos el tiempo de lectura del uso del dispositivo hardware es igual a 0 por lo que podemos, decir que se demora el mismo tiempo en el tiempo de lectura de los dos procesos, pero en la variable digitalización de la información del proceso tradicional menos el uso del dispositivo hardware es igual a 95,92 horas. Por lo que se puede decir que en el proceso tradicional se demora 96 horas más en digitalizar la información que la del uso del dispositivo hardware.

Se puede decir que el tiempo total del proceso tradicional es de 216 horas en recopilar la información, mientras que con el uso del dispositivo hardware es de 120 horas, por lo que el tiempo de recopilación de información del proceso tradicional es mayor que el uso del dispositivo hardware. Entonces podemos decir que una mejor herramienta para la recopilación de información para el cobro del agua potable en la ciudad de Baños es el uso del dispositivo hardware. Por lo que se acepta la hipótesis planteada en el proyecto desarrollado, facilitando al Municipio de la ciudad de Baños, mejorar el manejo y recopilación de la información, para la facturación del cobro de los servicios básicos como Agua Potable, Alcantarillado, y Recolección de Basura.

“Luego podemos afirmar que la integración de sistemas en plataformas Heterogéneas, mediante el uso de dispositivos hardware mejoran los métodos y procesos tradicionales de lectura y procesamiento de datos referentes a los servicios básicos que ofrece la Ilustre Municipalidad del cantón Baños”

Representación Gráfica

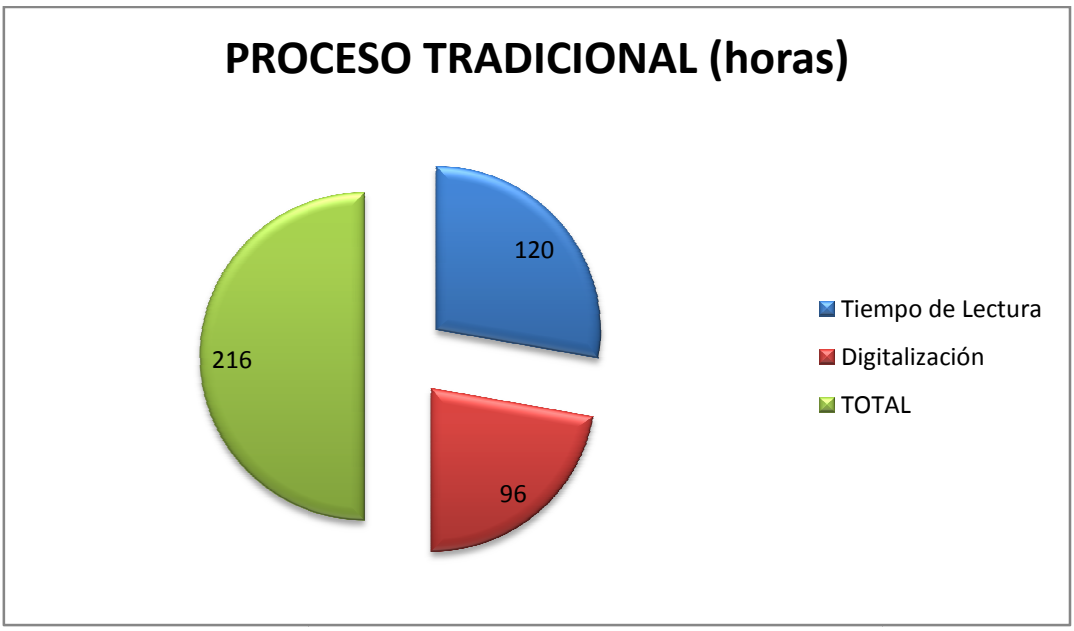


Figura 70 Proceso Tradicional para la toma de Lectura

5.5.1 Validación de la hipótesis

La implementación en sí es el núcleo base del desarrollo de un sistema, así como, la ingeniería de software realiza exhaustivas investigaciones para poder llegar a presentar la solución deseada, existen técnicas en equipo como el trabajo colaborativo con el fin de reducir los tiempos de esfuerzo y trabajo que realiza un desarrollador, sin descuidar características como la calidad, la seguridad, la eficiencia, etc.

Si podemos contar con componentes que nos permiten agilitar nuestro trabajo, estaremos en la capacidad de presentar una solución más productiva, de esta forma se demuestra que “*la*

integración de sistemas en plataformas Heterogéneas, mediante el uso de dispositivos hardware mejoran los métodos y procesos tradicionales de lectura y procesamiento de datos referentes a los servicios básicos que ofrece la Ilustre Municipalidad del cantón Baños" principalmente por permitir identificar la herramienta más productiva respecto a las necesidades que deseamos cubrir.

Conclusiones

- Se concluye que la estadística y específicamente, la comprobación directa para la determinación de tiempos óptimos es el método a aplicar cuando se desea realizar la demostración de la validez de una hipótesis en este caso específico de proyecto de tesis.
- La información proporcionada por el personal técnico de los Departamentos de Agua Potable, fue de gran ayuda para llegar a la consecución de la demostración de la Hipótesis planteada, y de esta manera se obtuvieron datos más óptimos y reales.
- Se ha demostrado que el Sistema de lecturas de medidores con Plataformas Heterogéneas es más óptimo en el ahorro de tiempo con comparación con el Sistema Tradicional, además evita errores de digitación de datos.

CONCLUSIONES

Como resultado del desarrollo del presente documento de tesis en donde se realizó es la automatización de los procesos de lectura y recolección de datos de los servicios prestados por la Municipalidad de Baños, se muestra las siguientes conclusiones:

1. Se puede concluir que la integración de dispositivos de características heterogéneas es posible de realizar como se ha demostrado en el presente proyecto si se utilizan las herramientas software y los equipos hardware adecuados.

2. El costo/beneficio del proyecto planteado se ve reducido en comparación al proceso tradicional, puesto que presenta una notable mejora en un corto plazo, reduciendo el tiempo de digitalización de los datos, menorando el requerimiento de personal, lo que implica el considerable ahorro en gastos.

3. Se determinó que la mejor herramienta de desarrollo para el caso presentado, es la utilización de Microsoft Visual Studio .NET, con el soporte brindado por su Compact Framework, junto con el alto desempeño de SQL Compact Edition.

4. La mejor manera de establecer una conectividad entre dispositivos es mediante protocolos TCP, para el caso se utilizaron los puertos nativos de SQL mediante el uso de la clase SqlClient, propia de .NET, la misma que brinda en su estructura una fácil y efectiva forma de enlazar datos entre 2 o más equipos, en nuestro proyecto entre el dispositivo móvil y el servidor centralizado.

5. El escogitamiento del Estándar de Código de Barras estará dado de acuerdo al soporte que brinde el equipo y el software para el fin, para el presente proyecto se utilizó el Estándar Code 128, y se lo implementó con la herramienta Telerik Report.

6. Nunca han sido tan poderosos los dispositivos portátiles, pues ofrecen a los usuarios nuevas y emocionantes formas de comunicación y administración para su negocio. Mantener la información accesible desde cualquier lugar es una necesidad más constante y en muchos casos simplemente la Internet no resuelve el problema. Para abordar esta necesidad, el uso de dispositivos móviles con la aplicación desarrollada para hacer frente a cada situación específica es una buena solución, ya que aporta al usuario la información de la forma que necesita. Pero para que esto sea posible es necesario utilizar un lenguaje de desarrollo para dispositivos móviles.

7. La evolución de la tecnología en general y de la informática móvil, en particular, a pesar de haber dado mucha importancia en las actividades profesionales no sean explotado en su totalidad, hasta el punto de utilizar todos los recursos que pueden ofrecer.

8. Con la puesta en marcha de esta Tesis, se gana precisión en la toma de datos, evitando quejas y reclamos por parte de los abonados de dicho servicio, puesto que anteriormente se presentaban casos de quejas por la recolección de datos de manera errónea, el sistema exige la lectura de abonado en abonado sin dejar de lado o fuera de dicho proceso a ningún usuario.

9. El manejo de la interfaz y su diseño ha tenido mucha más importancia de la que tiene en el desarrollo de aplicaciones para PC. Es un aspecto a tener en cuenta en la planificación del proyecto. También es de destacar la riqueza de elementos para construir la interfaz, tanto en controles como en funcionalidad de los existentes.

10. Durante el desarrollo de la Aplicación se observó la importancia vital que jugaron los patrones de software para el diseño y su posterior construcción. Los patrones encierran soluciones a problemas recurrentes, y plantean la mejor opción, se consideran buenas prácticas de programación, garantizando un diseño flexible, reutilizable y extensible.

RECOMENDACIONES

1. Ahorrar recursos en el momento de la programación, específicamente para el caso de dispositivos móviles, ya que sus características hardware son limitadas con respecto a las de una PC normal.

2. Dependiendo del caso, si se desea reconstruir o estudiar la estructura de una Base de Datos existente, se recomienda realizar su reingeniería o ingeniería inversa con la finalidad de comprender la estructura y el flujo de los datos dentro de la misma.

3. Se recomienda al desarrollador el perfecto escogitamiento de la herramienta y lenguaje de implementación, puesto que existen una gran cantidad de opciones, pero se debe estudiar y seleccionar la adecuada, acorde al caso especifico.

4. Puesto a la dificultad de información que presenta el uso de equipos móviles, se recomienda la constante investigación y desarrollo de habilidades técnicas en el programador.

5. En caso de proyectos exigentes se recomienda el uso de metodologías ágiles, específicamente la metodología XP, ya que esta implica explotar al máximo las capacidades de los desarrolladores y de esta manera obtener un buen producto que satisfaga las necesidades planteadas por el usuario.

RESUMEN

El objetivo de la tesis fue Integrar Sistemas con Plataformas Heterogéneas (Dispositivos Móviles y Terminales) usando hardware (Pocket Pc y PCs), aplicadas en la Municipalidad de Baños, Tungurahua, para automatizar lecturas y digitación de datos en el Departamento de Agua Potable, emitiendo notificaciones.

Aplicándose el método científico y deductivo así como herramientas de desarrollo e implantación, Visual Studio .Net, lenguaje C#, SQL Ce, SQL Sever, integrando un servidor de Base de Datos, sobre Windows.

Se ha integrado el sistema GCS, que funciona para la emisión de facturas por el consumo de agua potable, con el sistema portable SISNYCS, mediante la aplicación SISTRAD, que sincroniza y realiza el intercambio de la información.

Para la toma de lecturas se ingresan los datos mediante un lector de código de barras, para emitir instantáneamente la respectiva notificación al cliente. Terminado el recorrido de la ruta asignada se descarga de la información del dispositivo al terminal, para una posterior emisión de facturas, ahorrando el proceso de digitación manual, susceptible a errores.

Todo el proceso con el sistema tradicional toma 216 horas mensuales, mientras que con el proceso automatizado, se reduce a 120 horas. Afirmándose que una mejor herramienta para la recopilación de información para este fin es el uso de dispositivos hardware.

Recomendándose al administrador del sistema, al momento de la transferencia de la información entre dispositivos, se realice exhaustivamente, en el filtrado de sectores y zonas a ser cargadas, para que no se sature al dispositivo móvil, y pueda funcionar correctamente.

SUMARY

The objective of this thesis was to integrate systems with heterogeneous platform using hardware devices, applied in the City of Bath, in order to automate the current process of reading and typing data in the Departments of Water Supply.

The research was based on the scientific method and deductive and are used as tools of development and deployment, Visual Studio. Net, as a base language C #, SQL Compact Edition, SQL Sever, integrating a database served on a Windows platform.

By checking the assumptions we can say that the total process time in gathering information through the traditional method is 216 hours while using the hardware device is 120 hours, what time of compilation information through the traditional process is higher than with the use of hardware devices. Then we can say that a better tool for gathering information for the collection of drinking water in the city of Bath is to use the hardware device.

It has been shown that the meter-reading system with heterogeneous platform is more optimal time saving compared to the traditional system also avoids data typing errors. This will update the table entries, error-free data, thus avoiding the fingering process that involves the loss of time and is susceptible to input errors by the data entry, causing internal conflicts within the institution, besides loss of time and money.

With the completion of this project has been achieved through integration of systems across heterogeneous platforms, in addition to providing superior service and tailored to the needs and characteristics of Municipalidade concerned.

GLOSARIO

Aplicación: Sistema de tipo software que brinda solución a un cierto problema

Botón: Imagen con funcionalidad de navegación para los usuarios o personal relacionado con un sistema.

Código de Barras: Código que implica el uso de barras y espacios para de esta manera codificar caracteres alfanuméricos.

Common Language Runtime (CLR): Administra la memoria, ejecución de subprocesos, ejecución de código, comprobación de la seguridad del código, compilación y demás servicios del sistema. Podríamos decir que esta es la máquina virtual de .NET.

CTS (Common Type System, Sistema de Tipos Común). El CTS garantiza que todo el código administrado es autodescriptivo. Los diferentes compiladores de lenguajes de Microsoft y de terceros generan código administrado que se ajusta a CTS. Esto significa que el código administrado puede usar otros tipos e instancias administrados, al tiempo que se aplica inflexiblemente la fidelidad y seguridad de los tipos.

DLL: Una DDL, se utiliza para crear objetos de base de datos y definir la apariencia que tendrán los datos. Por ejemplo por medio de DDL no solo podemos crear bases de datos, tablas, vistas, etc., sino que también podemos eliminarlas.

GCS: Sistema para la Comercialización de Servicios Municipales, usado por los departamentos de Agua Potable de los mismos.

Impresora Portable: Dispositivo se salida cuya función es la de entregar reportes impresos desde un dispositivo móvil, y tiene como particularidad su portabilidad, y su funcionamiento depende del período de carga de la misma.

Lector Óptico: Dispositivo de entrada, el cual interpreta luego de haber leído un determinado código de barras y lo transforma a caracteres, entendibles para los humanos.

MSDN® (Microsoft Developer Network): Proporcionan asistencia basada en la tarea y el lenguaje de programación actuales, garantizado que los programadores no se encuentren nunca perdidos en la plataforma Microsoft .NET o en el lenguaje que hayan elegido.

Pantalla: Sitio que muestra una interfaz apta para la visualización e interacción con el usuario.

Protocolo RPC: Conocido como SOAP (Simple Access Object Protocol), que no es más que una combinación de estándares como HTTP y XML para realizar llamadas a los miembros de estos servicios web.

RBD Reingeniería de la Base de Datos: Es la etapa posterior a la Ingeniería Inversa, puesto que se parte de esta para realizar un rediseño de una base de Datos existente.

Reporte: Resultado obtenido a partir de un proceso u operación de cálculo.

SISTRAD: Sistema de Notificación y Cobro de Servicios, Sistema diseñado para funcionar en dispositivos móviles (Pocket PC), que sirve para la toma de lecturas de medidores de agua potable mediante lectores ópticos de códigos de barras, para su posterior impresión mediante impresoras portables.

SISNYCS: Sistema de Notificación y Cobro de Servicios, Sistema diseñado para funcionar en dispositivos móviles (Pocket PC), que sirve para la toma de lecturas de medidores de agua

potable mediante lectores ópticos de códigos de barras, para su posterior impresión mediante impresoras portables.

SQL Server Compact Edition: Comúnmente conocido cómo SQL Server CE, es el motor de base de datos que proporciona Microsoft para sus Sistemas Operativos, Microsoft Windows CE .NET, Pocket PC 2002 con Windows o Pocket PC 2000. Su creación y funcionamiento está basado en el proveedor de datos de Compact Framework, el cual fue desarrollado con la tecnología .NET.

Usuario: Persona que usa la tecnología para obtener un determinado objetivo.

BILBIOGRAFÍA

Bibliografía General.

(1) INTEGRACIÓN DE SISTEMAS

SOMMERVILLE, I. Ingeniería de Software. Sexta edición. México: Addison Wesley, 2008.

http://fferrer.dsic.upv.es/cursos/Integracion/html/index.html

200905

http://www.wikipedia.org.

200905

(2) ANÁLISIS DE SISTEMAS EXISTENTES

http://www.monografias.com/trabajos6/sista/sista.shtml

200905

(3) INGENIERÍA INVERSA

http://tgrajales.net/investipos.pdf

200905

http://www.wikilearning.com/curso_gratis/curso_de_ingenieria_inversa/3807

200904

http://www.hipermarketing.com/nuevo%204/CONTENIDO/tecnologia/nuevas%20tecnologias/nivel3REVERSEENGINNER.html

200905

http://gustavolarriera.tripod.com/doc/tech/clr97sli.pdf

200905

http://www.products.cl/Productos_Servicios/SolBD.htm

200905

(4) CÓDIGOS DE BARRAS

http://www.codigodebarras.org

200905

http://www.w3.org

200908

http://www.bureaucorp.net/ar/default2.asp

200907

(5) SISTEMAS OPERATIVOS MÓVILES

http://www.idg.es/pcworldtech

200905

http://www.versioncero.com/categor%C3%ADa/moviles/

200906

http://www.madboxpc.com/contenido.php?id=5308

200907

(6) Empresas dedicadas a Dispositivos Móviles

http://www.pc.ibm.com/store/ Sitio de IBM dedicado a POS

200908

http://www.mobilenug.com Mobile Net Users Group

200904

http://www.micmovilidad.es Centro de Innovación en Movilidad Microsoft

200902

http://pos.epson.com Sitio de sistemas POS de Epson

200904

www.palmone.com Sitio oficial de Palm One

200905

www.palmsource.com Sitio oficial de Palm OS

200906

www.handspring.com Sitio de uno de los co-fundadores de Palm

200907

www.gentepalm.com Sitio en español de Palm.

200908

www.todopalm.cl Sitio chileno dedicado a las PDA Palm.

200909

www.piensaenpalm.com Sitio dedicado a Palm. En español.

200907

www.garmin.com Sito de un fabricante de GPS's que ejecutan

200906

http://es.pscnet.com/html/hs1250.htm PSC venta de productos de retail.

200908

http://www.tekservepos.com/hardware.html Resellers de sistemas POS.

200909

www.sonystyle.com

200902

www.fossil.com

20094

www.symbol.com

200906

(8) METODO CIENTIFICO

http://www.hondurassilvestre.com/Reportajes/MetodoCientifico.htm

200903

(9) METODOLOGÍAS AGILES

http://www.controlcaos.com

200903

http://www.featuredrivendevelopment.com/

200907

http://members.aol.com/acockburn/

200907

http://www.adaptivesd.com/

200907

http://www.agilemodeling.com

200905

BIBLIOGRAFÍA RELACIONADA AL TEMA.

(10) COMPACT FRAMEWORK Y C#

STEPHEN, W, Microsoft .NET Compact Framework, Segunda Edición, México, Microsoft Press, 2007

GONZALEZ, J. El lenguaje de programación C#, Tercera Edición, Ecuador, Planeta, 2008

FOSTER, L. Palm OS Programming Bible, Primera Edición, México, Hungry Minds, 2008

http://msdn.microsoft.com Biblioteca MSDN.

200905

http://www.pdaexpertos.com.

200903

http://www.todopocketpc.com.

200907

(11) SQL CE

http://www.microsoft.com/downloads/details.aspx?familyid=8BDAA836-0BBA-4393-94DB-6C3C4A0C98A1&displaylang=en

200903

http://www.microsoft.com/Sqlserver/2005/en/us/compact.aspx

200902

(12) TRANSFERENCIA HOMOGENEA

http://www.codeproject.com

200905

http://www.syncguru.com

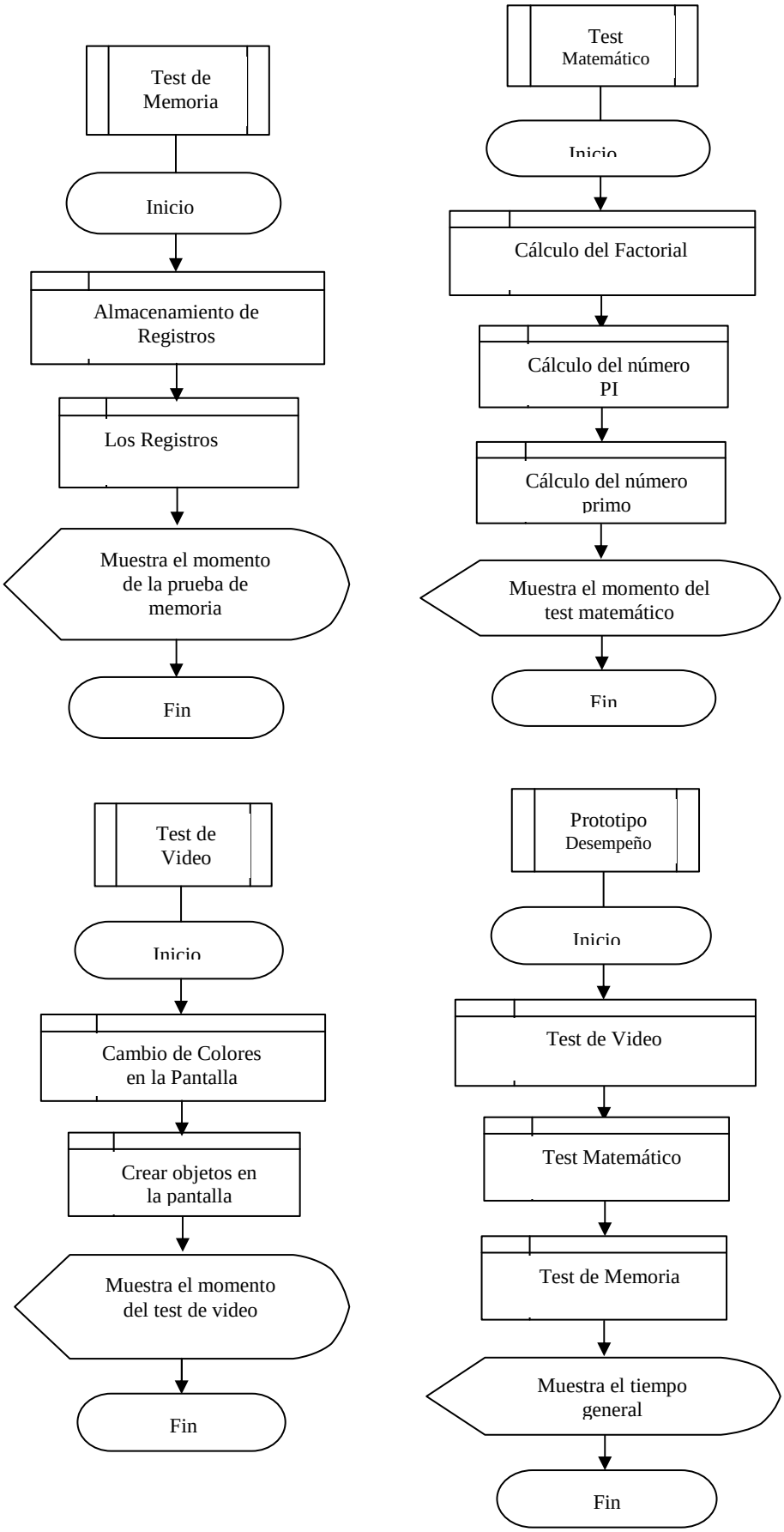
200904

http://foro.todopocketpc.com

200903

ANEXOS

Anexo 1. Flujograma de Funcionamiento



Anexo 2. Algoritmo para las Pruebas de Evaluación

```
Programa Rendimiento
Declaraciones
Inicio
Procedimiento Ejecutar
Declaraciones
Entero Tiempo_VDO, Tiempo_MAT, Tiempo_MEM, Tiempo_Total
Inicio
Tiempo_VDO <- Test_Video
Write( "Tiempo Test VDO: " + Tiempo_VDO + " Segundo(s)" )
Tiempo_MAT <- Test_Matematico
Write( "Tiempo Test MAT: " + Tiempo_MAT + " Segundo(s)" )
Tiempo_MEM <- Test_Memoria
Write( "Tiempo Test VDO: " + Tiempo_MEM + " Segundo(s)" )
Tiempo_Total <-Tiempo_VDO + Tiempo_MAT + Tiempo_MEM
Write( "Tiempo Geral: " + Tiempo_Total + " Segundo(s)" )
Fin
Entero funcion Test_Matematico
Declaraciones
Entero Test_1, Test_2, Test_3, Total
Inicio
Test_1 <- MAT_Fatorial
Write( "Fatorial: " + Test_1 + " Segundo(s)" )
Test_2 <- MAT_NumeroPI
Write( "Número PI: " + Test_2 + " Segundo(s)" )
Test_3 <- MAT_NumerosPrimos
Write( "Números Primos: " + Test_3 + " Segundo(s)" )
Total <- Test_1 + Test_2 + Test_3
```

```
Retorna( Total )
Fin

Entero funcion Test_Video
Declaraciones
Entero Test_1, Test_2, Total
Inicio
Test_1 <- VDO_MudarColores
Write( "Mudar Colores: " + Test_1 + " Segundo(s)" )
Test_2 <- VDO_CrearObjetos
Write( "Crear Objetos: " + Test_2 + " Segundo(s)" )
Total <- Test_1 + Test_2
Retorna( Total )
Fin
Entero funcion Test_Memoria
Declaraciones
Entero Test_1, Test_2, Total
Inicio
Test_1 <- MEM_GrabarArchivos
Write( "Grabar Archivos: " + Test_1 + " Segundo(s)" )
Test_2 <- MEM_LeerArchivos
Write( "Leer Archivos: " + Test_2 + " Segundo(s)" )
Total <- Test_1 + Test_2
Retorna( Total )
Fin
Entero funcion MAT_Fatorial
Declaraciones
Entero X, FAT
Hora Hora_Inicio, Hora_Fin
```

```
Inicio
Hora_Inicio <- HoraAtual
para X <- 1 hasta 70000 incremento 1
FAT = Fatorial( 10 )
Finpara

Hora_Fin <- HoraAtual
Retorna( Hora_Fin - Hora_Inicio )
Fin
Entero funcion Fatorial( Entero N )
Inicio
si( N > 1 ) entonces
Retorna( N * Fatorial( N - 1 )
caso_contrario
Retorna( 1 )
Fin
Entero funcion MAT_NumeroPI
Declaraciones
Hora Hora_Inicio, Hora_Fin
Real A, B, PI
Entero I, SIGN
Inicio
Hora_Inicio <- HoraAtual
A <- 16/5
B <- 4/239
PI <- A - B
SIGN <- -1
I <- 3
mientras( I < 700000 )
```

```
A <- A/25
B <- B/57121
PI <- PI + SIGN * ( A - B ) / I
SIGN <- - SIGN
I <- I + 2
Finmientras
Hora_Fin <- HoraAtual
Retorna( Hora_Fin - Hora_Inicio )

Fin
Entero funcion MAT_NumerosPrimos
Declaraciones
Entero Prox_Divisor, Qtd_Divisores
Hora Hora_Inicio, Hora_Fin
Logico Resultado
Inicio
Hora_Inicio <- HoraAtual
Prox_Divisor <- 1
Qtd_Divisores <- 2
mientras( Prox_Divisor <= 700000 )
si( 700000 mod Prox_Divisor = 0 ) entonces
Qtd_Divisores <- Qtd_Divisores + 1
FinSi
Prox_Divisor <- Prox_Divisor + 1
Finmientras
si( Qtd_Divisores = 2 ) entonces
Resultado <- Verdadeiro
caso_contrario
Resultado <- Falso
```

```
FinSi
Hora_Fin <- HoraAtual
Retorna( Hora_Fin - Hora_Inicio )
Fin
Entero funcion VDO_MudarColores
Declaraciones
Hora Hora_Inicio, Hora_Fin
Entero I, R, G, B
Inicio
Hora_Inicio <- HoraAtual
para I <- 0 hasta 3000 incremento 1
R <- Randomico( 255 )
G <- Randomico( 255 )
B <- Randomico( 255 )
DesenharQuadrado( 160, 160, Cor(R,G,B) )
Finpara
Hora_Fin <- HoraAtual
Retorna( Hora_Fin - Hora_Inicio )
Fin
Entero funcion VDO_CrearObjetos
Declaraciones
Hora Hora_Inicio, Hora_Fin
Entero I, X, Y, R, G, B, Raio, scLargura, scAltura
Inicio
Hora_Inicio <- HoraAtual
scLargura <- 160
scAltura <- 160
Raio <- 17
X <- Raio
```

```
Y <- Raio
para I <- 0 hasta 3000 incremento 1
R <- Randomico( 255 )
G <- Randomico( 255 )
B <- Randomico( 255 )
DesenharCirculo( 160, 160, Cor(R,G,B) )
si( X < scLargura - Raio ) entonces
X <- X + 5
caso_contrario
X <- Raio
si( Y < scAltura - Raio ) entonces
Y <- Y + 5
caso_contrario
Y <- Raio
FinSi
FinSi
Finpara
Hora_Fin <- HoraAtual
Retorna( Hora_Fin - Hora_Inicio )
Fin
Entero funcion MEM_GrabarArchivos
Declaraciones
Hora Hora_Inicio, Hora_Fin
Entero I
Inicio
Hora_Inicio <- HoraAtual
CrearArquivo( "Desempeño" )
para I <- 0 hasta 2000 incremento 1
WriteArquivo( "Proyecto de Grado - Test de Desempeño" )
```

```
Finpara
Hora_Fin <- HoraAtual
Retorna( Hora_Fin - Hora_Inicio )
Fin
Entero funcion MEM_LeerArchivos
Declaraciones
Hora Hora_Inicio, Hora_Fin
Caracter Texto
Entero I
Inicio
Hora_Inicio <- HoraAtual
AbrirArquivo( "Desempeño" )
para I <- 0 hasta 2000 incremento 1
Texto <- LeerArchivo( I )
Finpara

BorrarArchivo( "Desempeño" )
Hora_Fin <- HoraAtual
Retorna( Hora_Fin - Hora_Inicio )
Fin
Fin
```

Anexo 3. Manual de Usuario Módulo SISNYCS

1. Introducción

El presente sistema, denominado SISNYCS, ***Sistema de Notificación y Cobro de Servicios*** , que ha sido desarrollado con el fin de complementar al Sistema GCS, implantado en los Departamentos de Agua Potable del Municipio de Baños, al momento de realizar la recolección de datos y de esta forma se platea dar al mencionado sistema una nueva funcionabilidad más eficiente y efectiva con el fin de satisfacer la necesidades y expectativas de los usuarios a la hora de realizar sus pagos con el consumo de los servicios básicos; Agua, Recolección de Basura y Alcantarillado.

SISNYCS, ha sido denominada como una herramienta muy sencilla para su utilización, esta realiza la toma de lecturas en los medidores de agua potable de diferentes clientes catastrados y registrados en la Base de Datos del sistema GCS mediante un lector óptico de código de barras, para luego ingresar la lectura actual del medidor en cuestión, y concretar todos los cálculos pertinentes de acuerdo a la tarifa designada y de esta forma imprimir una notificación con los rubros de pago designados.

SISNYCS está dirigido para los empleados que realizan la toma de Lecturas de los medidores de Agua Potable, de los departamentos de esta dependencia para el municipio de Baños, el mismo que cuenta con el sistema de pagos GCS, y sus funcionalidades principales son las siguientes:

- Identificación del usuario a ser valorizado mediante código de barras, en el cual se identificarán su código de usuario.
- Cálculo exacto y preciso del rubro a pagar de acuerdo a la categorización del usuario en cuestión.

- Presentación impresa de notificaciones para los usuarios en la cual se identificarán los saldos a ser cancelados en el departamento de agua potable.
- Generación de tarifas ser canceladas por parte de créditos de medidores comprados a plazos por parte de los usuarios.
- Actualizaciones de nuevas lecturas de medidores en caso de existir lecturas erróneas de las mismas.
- Búsquedas interactivas del usuario a ser evaluado.

2. GENERALIDADES DEL SISTEMA

- Desarrollado con interfaz gráfica.
- Software para Dispositivos Móviles

3. Seguridades

En cuanto a las seguridades de la aplicación Móvil SISNYCS, no ha determinado uso de usuarios, ya que la misma está dirigida específicamente al personal que realiza la toma de lecturas en los Departamentos de Agua Potable, y cuyo objetivo principal es recolectar la información sobre la lectura de una determinada fecha.

4. Requisitos mínimos indispensables

Ud. solamente necesita un computador con buenas características y el acceso a la Red Institucional de la ESPOCH o al Internet para obtener acceso a SIAGERI. A continuación se detallan los requisitos mínimos indispensables de tal equipo:

Hardware:

- 32 ó 64 MB de RAM, 32 MB de ROM

- Pantalla 1/4 VGA
- Batería de litio-ion recargable.
- CPU: Intel Xscale de 400 Mhz
- Cuna para Ethernet de cuatro slots
- Cable serie de carga
- Opciones de cable para impresora

Software:

- Sistema operativo incluído Microsoft® Windows® CE 4.1 (CE .NET / Windows Mobile 2003)

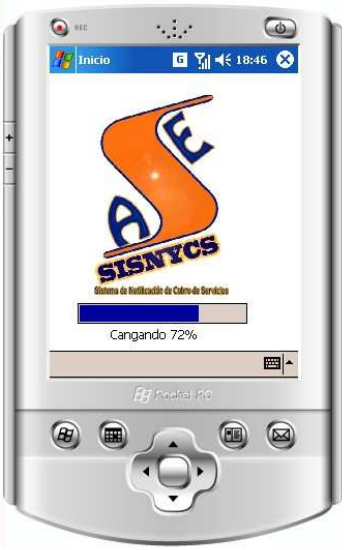
5. **Como usar el programa / Mi 1ra sesión**

Para empezar a utilizar el sistema, y por ser una Aplicación diseñada para dispositivos móviles, es necesaria su instalación en un Pocket PC para que el usuario pueda disfrutar y sacar provecho a todas las bondades del sistema, Cabe destacar además, que el dispositivo móvil debe contar con una impresora portátil y u lector óptico de código de barras instalado, para su óptima funcionabilidad.

6. ESTRUCTURA DEL SISTEMA

Formulario Principal

El primer formulario, es un splash, en el cual se mostrará el progreso de carga de la aplicación.



La siguiente pantalla muestra un formulario de ingreso de datos, que está dividido en dos partes, para el ingreso de datos como; el número de cuenta del usuario (dato ingresado a través del lector óptico de código de barras o manualmente), Apellidos, Nombres, Lectura Anterior, Dirección, Tipo (datos generados automáticamente luego de haber ingresado el número de cuenta), Lectura Actual, Comentario, Observación (datos a ser ingresados de forma manual por el usuario dependiendo del estado actual del medidor).

Estas pantallas muestran además tres opciones principales: Grabar, Cancelar e Imprimir.

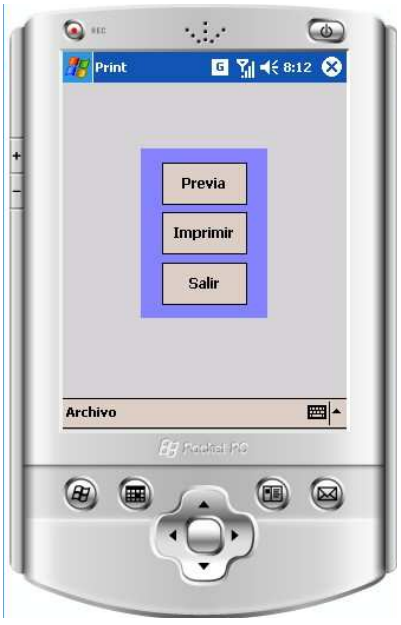
Opción Grabar

Permite al usuario registrar todos los datos ingresados de una lectura actual, en caso de que esta sea ingresada por primera vez, y de actualizar los datos en caso de haber sido ingresada por más de dos ocasiones.

Opción Cancelar

Esta opción simplemente permite la cancelación de todos los procesos de registro y actualización de lecturas.

Opción Imprimir



Mediante esta opción primeramente se realizan una serie de cálculos necesarios para la impresión de las notificaciones, para luego presentar un nuevo formulario en el cual se muestran nuevas opciones referentes a la impresión del reporte antes evaluado.

Opción Previa

Al elegir esta opción se mostrará una vista previa de la notificación a ser impresa, en la cual se podrán observar los valores ser cancelados por diversos conceptos, además de los datos personales del usuario a ser cobrado. Para regresar a la pantalla anterior se debe simplemente pulsar el botón **Atrás.**

Opción Imprimir

Mediante opción se enviará a la impresora portátil, la notificación previamente observada y aprobada por el usuario.

Anexo 4. Manual de Usuario Módulo SISTRAD

1. INTRODUCCIÓN.

Bienvenidos a “SISTRAD” ***Sistema de Transferencia de Datos***, en el cual se le indicará paso a paso algunas de las instrucciones a seguir para que se pueda hacer buen uso del sistema.
El presente sistema, denominado SISTRAD, que ha sido desarrollado con el fin de complementar al Sistema SGCS implantado en los Departamentos de Agua Potable de La Municipalidad de Baños, y también complemento al módulo SISNYCS, al momento de realizar la recolección de datos y de esta forma se plantea dar al mencionado sistema una nueva funcionabilidad mas eficiente y efectiva con el fin de satisfacer la necesidades y expectativas de los usuarios a la hora de realizar sus pagos con el consumo de los servicios básicos; Agua, Recolección de Basura y Alcantarillado.

SISTRAD, es una aplicación, que realiza la función de transferir datos entre Base de Datos de manera Bidireccional, entre un equipo Móvil y una PC Servidor

SISTRAD está dirigido para los empleados de los departamentos de agua potable de los distintos municipios que cuenten con el sistema de pagos GCS, y sus funcionalidades principales son las siguientes:

- Carga de Datos de la base de datos del Pocket a la PC.
- Carga de Datos de la base de datos de la PC al Pocket.
- Generar Código de Barras de manera General para todos los abonados con medidor.

Generar Código de Barras de manera específica por medio de Sector y Ruta para los abonados con medidor.

2. GENERALIDADES DEL SISTEMA

- Desarrollado con interfaz gráfica.
- Software para Dispositivos Móviles

3. Requisitos Mínimos Indispensables.

Ud. solamente necesita instalar la Aplicación en el Pc donde funcionará el sistema SISTRAD. No se detallará los requisitos hardware mínimos puesto que el servidor que se encuentra en funcionamiento cubre las expectativas necesarias y deseadas de la aplicación

Sistema Operativo: Microsoft Windows Server 2003 ó 2008, Microsoft Windows Xp

Aplicaciones: Framework 2.0, SQL Server, SQL CE, Componente Telerik, Sistema GCS, Base de Datos “dbame”.

Procesador: Pentium IV o superior para mejor rendimiento.

Velocidad: 3.0 GHZ o mayor.

Memoria RAM: 1GB en adelante.

Disco Duro: 40 GB libre mínimo.

Monitor: SVGA.

4. Como usar el programa / Mi 1ra sesión

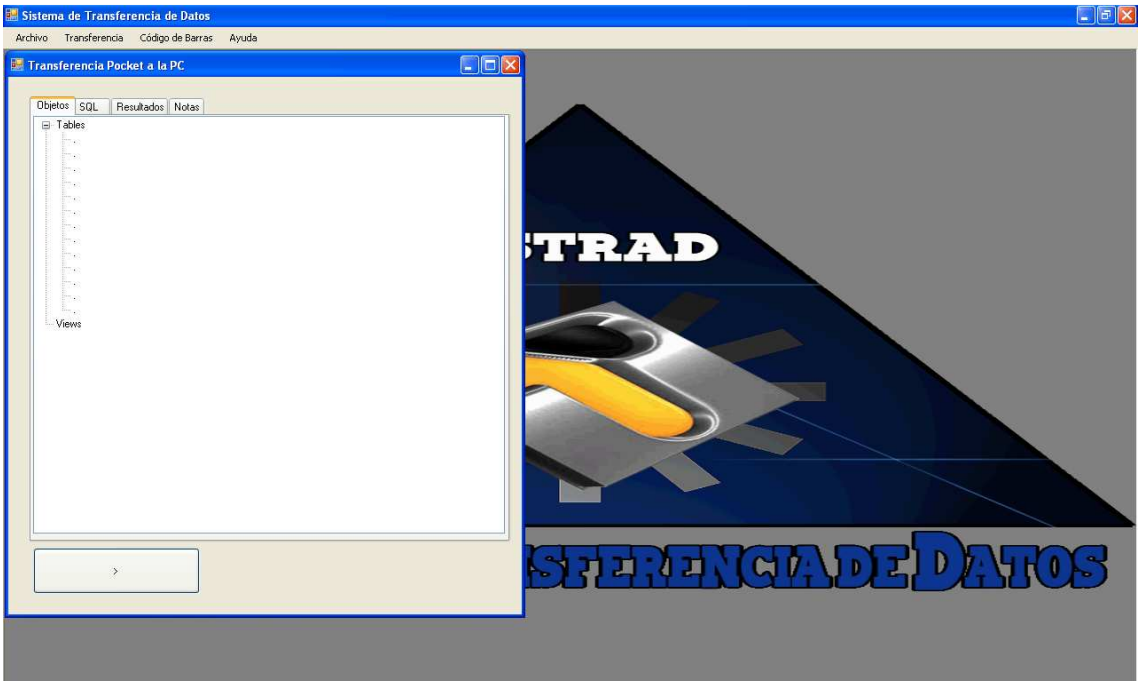
Para empezar a utilizar el sistema, y por ser una Aplicación diseñada para uso en entorno Windows, es necesaria su instalación en una PC para que el usuario pueda disfrutar y sacar provecho a todas las bondades del sistema, además de tener instalado el sistema GCS, corriendo con su respectiva Base de Datos “dbame”.

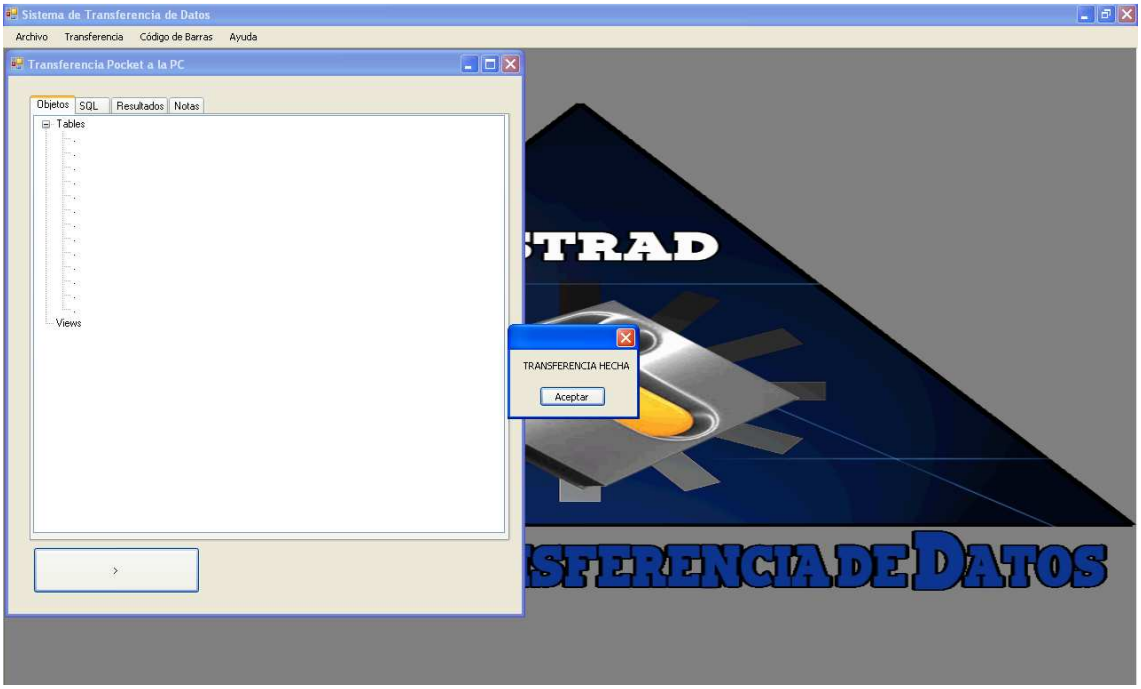
5. ESTRUCTURA DEL SISTEMA

Formulario Principal

El primer formulario, es un MDI container, en el cual se mostrará el menú y opciones de la aplicación.

La siguiente pantalla muestra un formulario de Transferencia de Datos en Dirección del Pocket a la PC, simplemente damos clic en el botón con forma de flecha, las tablas que se pasaran en esta dirección es: Lecturas.

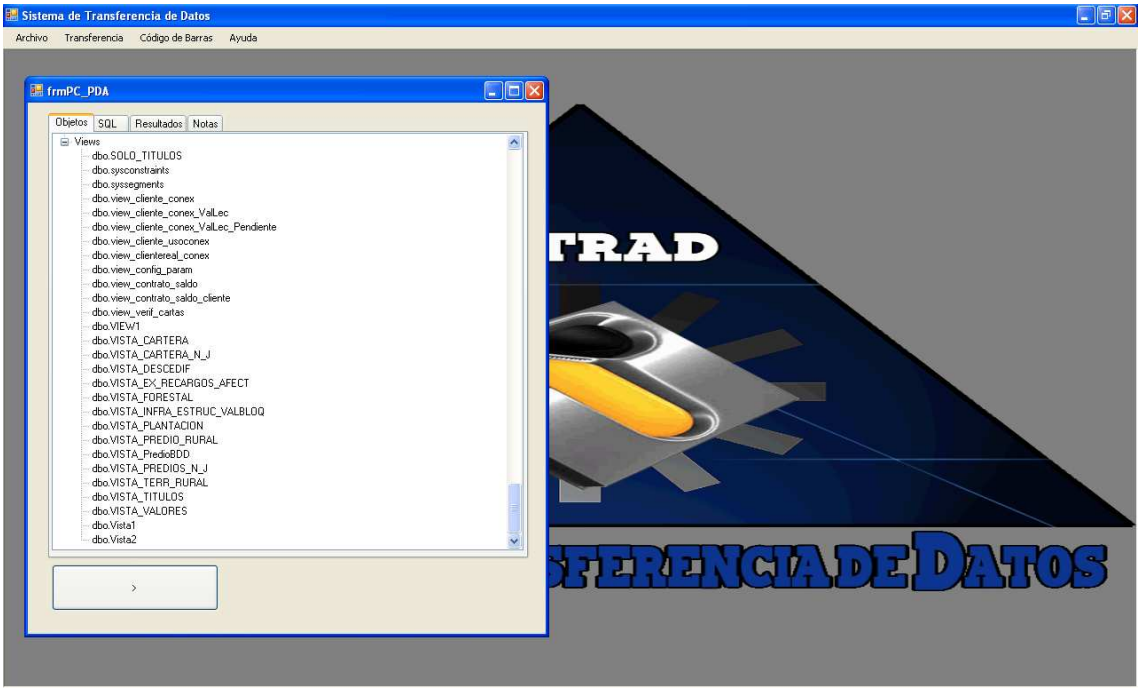


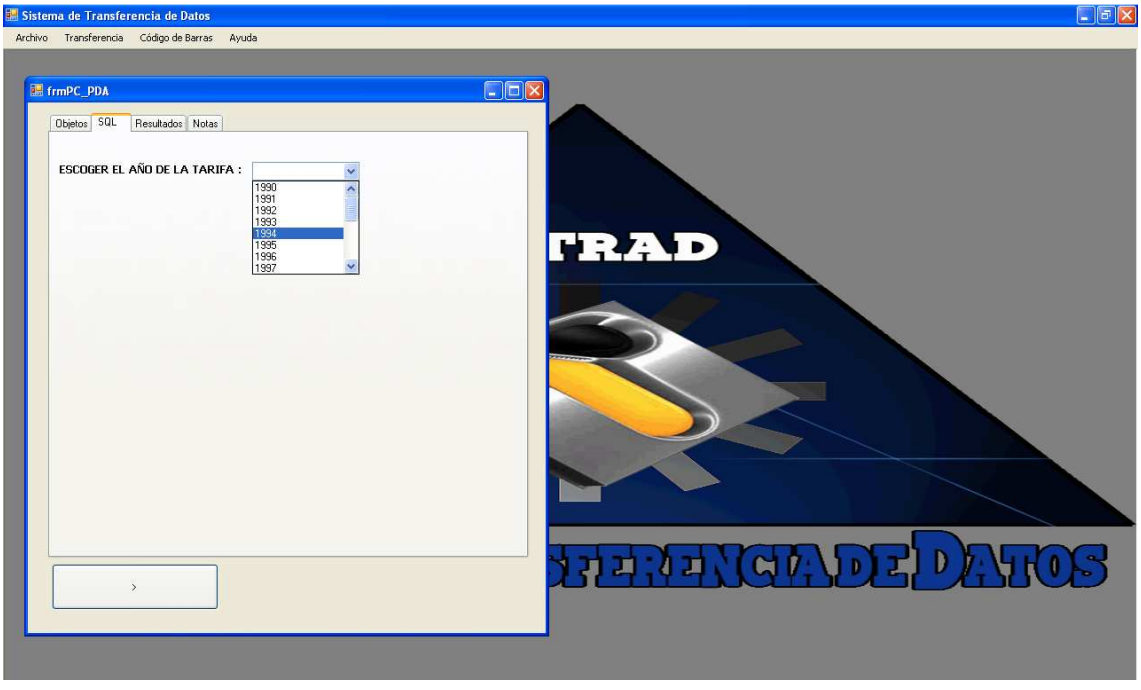


La siguiente pantalla muestra un formulario de Transferencia de Datos en Dirección de la PC a la Pocket, escogemos previamente el año de la tarifa a evaluar, misma que esta

ubicada en la pestaña SQL, y luego damos clic en el botón con forma de flecha, las tablas que se pasaran en esta dirección son:

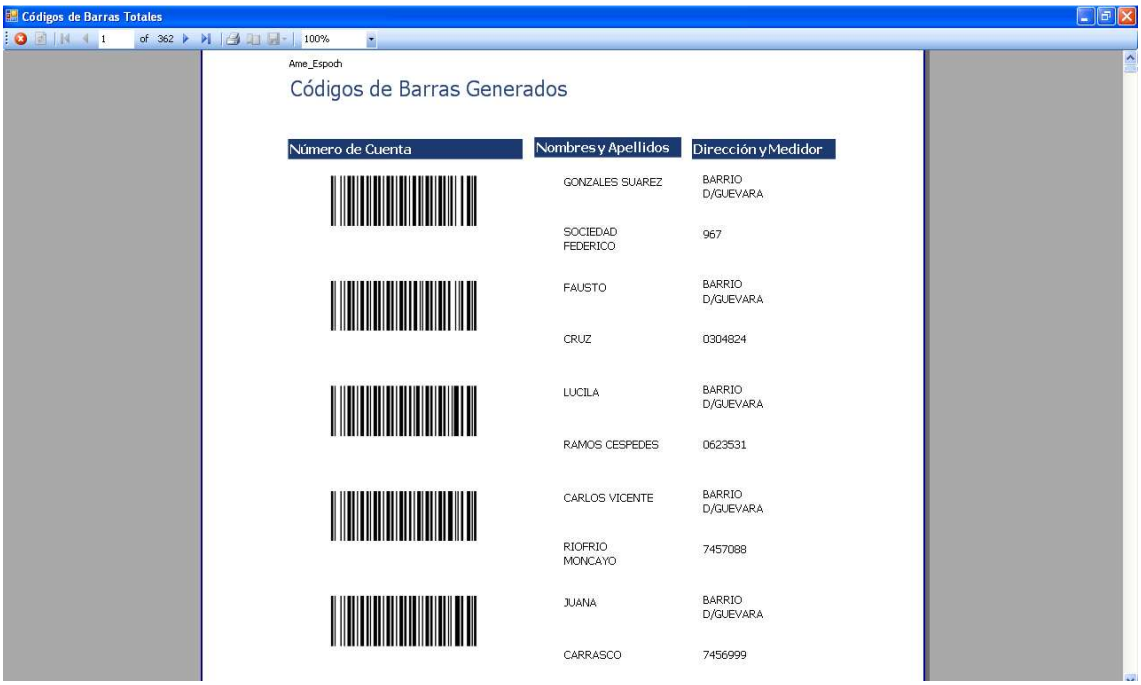
Cartas_Cliente, Catastro, C_Tarifa, Contrato,Datos_Municipio, Lecturas, Saldo, Uso_Conexion.





Códigos de Barra

Lo Podemos generar de la Base de Datos de manera general para todos los abonados



Lo Podemos generar de la Base de Datos de manera específica mediante Ruta y Sector


Sistema de Transferencia de Datos
Archivo
Transferencia
Código de Barras
Ayuda
Códigos de Barras Filtrados
Sector
01
Ruta
01
Generar >>>
SISTEMA DE TRANSFERENCIA DE DATOS


CodeBar
1
of 59
100%
Ame_Espoch
Códigos de Barras Generados
Número de Cuenta
Nombres y Apellidos
Dirección y Medidor
GONZALES SUAREZ
BARRIO D/GUEVARA
SOCIEDAD FEDERICO
967
FAUSTO
BARRIO D/GUEVARA
CRUZ
0304824
LUCILA
BARRIO D/GUEVARA
RAMOS CESPEDES
0623531
CARLOS VICENTE
BARRIO D/GUEVARA
RIOFRIO MONCAYO
7457088

Anexo 4. Diccionario de Datos

C_TARIFA Atributos

Nombre	Dominio	Tipo de Dato	NULL	Definición
CTar_Codigo		Varchar(8)	NO	Se trata de la clave principal para esta tabla.
CTar_AñoProce		Varchar(4)	NO	Se refiere al año en el que fueron ingresados los rangos para las tarifas de cobro.
CTar_UsoConex		Varchar(30)	YES	Este campo va relacionado con el campo UsoCo_Desc de la tabla Uso_Conexion para describir los tipos de conexión pertenecientes para cada usuario.
CTar_Tipo		Varchar(10)	YES	Se Trata de la clasificación del tipo de tarifas (Básica, Rangos y Mixtas), cada una con una interpretación de cálculo distintas. Para cada uno de estos tipos de tarifas existe un rango de valores con su respectivo monto de tarifa y con su respectivo excedente.
CTar_ValMin		Numeric(9, 0)	YES	Este campo almacena el valor mínimo para un rango de consumo.
CTar_ValMax		Numeric(9, 0)	YES	Este campo almacena el valor máximo para un rango de consumo.
CTar_Tarifa		Numeric(18, 4)	YES	Se almacena aquí el costo por el consumo del rango consumido.
CTar_Excede		Numeric(18, 4)	YES	Este campo almacenará un valor por exceso de consumo, es decir por sobrepasar al rango de consumo, establecido por el valor mínimo (CTar_ValMin) y el valor máximo (CTar_ValMax).

CTar_Desc		Varchar(15)	NO	Se describe el tipo de servicio a ser calculado (Agua, Alcantarillado).

CARTAS_CLIENTE Atributos

Nombre	Dominio	Tipo de Dato	NULL	Definición
Cli_NumCuenta		Varchar(8)	No	Se trata de una clave foránea la cual sirve como nexo para conectarnos con la tabla Catastro.
CarCl_Año		Varchar(4)	No	Almacenan el año y mes en el proceso de cobro por los servicios
CarCl_Tipo		Varchar(1)	No	Con este campo podemos saber si el cliente ha pagado la deuda (P), aun esta se encuentra por pagar (S).
CarCl_Mes		Varchar(2)	No	Almacenan el año y mes en el proceso de cobro por los servicios
CarCl_Consumo		Numeric(9, 0)	Yes	En este campo se almacenará el valor del consumo del agua potable en metros cúbicos.
CarCl_ValConsumo		Numeric(9, 2)	Yes	En este campo se almacenarán los valores a pagar por el consumo del agua potable.
CarCl_ValBasura		Numeric(9, 2)	Yes	Para registrar el monto por pagar para el cobro de la basura.
CarCl_ValAlcanta		Numeric(9, 2)	Yes	En este campo se almacenarán los valores a pagar por el servicio del alcantarillado.
CarCl_ValAdmin		Numeric(9, 2)	Yes	Se almacenan los valores por gastos administrativos.
CarCl_Otros		Numeric(9, 2)	Yes	Destinado al cobro de otros rubros.
CarCl_Proc		Numeric(9, 2)	Yes	

CarCl_ValConex		Numeric(9, 2)	Yes	
CarCl_ValReconex		Numeric(9, 2)	Yes	
CarCl_ValMultasCon		Numeric(9, 2)	Yes	
CarCl_ValVarios		Numeric(9, 2)	Yes	
CarCl_IntMora		Numeric(9, 2)	Yes	Se almacenará el interés cobrado por la demora en los pagos.
CarCl_Interes		Numeric(9, 2)	Yes	
CarCl_TotaLiq		Numeric(9, 2)	Yes	Aquí se almacena la suma total de todos los rubros por pagar.
CarCl_TotaFac		Numeric(9, 2)	Yes	
CarCl_FechaLiq		Datetime	Yes	
CarCl_FechaFac		Datetime	Yes	
CarCl_FechaProc		Datetime	Yes	
CarCL_EstaLiq		Varchar(1)	Yes	
CarCl_ValPag		Numeric(9, 2)	Yes	
CarCl_ValPen		Numeric(9, 2)	Yes	
CarCl_ValIva		Numeric(9, 2)	Yes	
CarCl_ValDesc		Numeric(9, 2)	Yes	
CarCl_Codigo		Numeric(18, 0)	Yes	
Usu_Cod		Numeric(9, 0)	Yes	
SalAnt_Abono		Numeric(18, 2)	Yes	
CarCl_ConMed		Numeric(18, 2)	Yes	

CATASTRO Atributos

Nombre	Dominio	Tipo de dato	NULL	Definición
Cli_NumCuenta		Varchar(8)	NO	Clave Primaria en la cual se registra el número de cuenta del cliente, está compuesta por 8 dígitos y sirve para relacionar con la tabla Contrato,

				Cartas Cliente, Lecturas, Uso Conexión, entre otras.
Cat_TipoCli		Varchar(20)	YES	En la cual se describe el tipo de cliente "1- REAL".
UsoCo_Codigo		Numeric(9, 0)	YES	Se trata de una llave foránea la misma que nos podemos enlazarnos con la tabla Uso_Conexión para de esta forma acceder al campo UsoCo_Desc, el mismo que contiene una descripción del tipo de conexión del usuario.
Cat_CedulaCli		Varchar(13)	YES	este campo se almacena el número de cédula del cliente.
Cat_NombresCli		Varchar(30)	YES	Describe los nombres del cliente.
Cat_ApellidosCli		Varchar(30)	YES	Describe los apellidos del cliente.
Cat_DirPCli		Varchar(50)	YES	Se almacena la dirección principal del cliente.
Cat_DirICli		Varchar(50)	YES	Se almacena una dirección secundaria del cliente.
Cat_CodBarrioCli		Numeric(9, 0)	YES	Almacena el código del barrio al que pertenece el cliente, para luego enlazarse con la tabla Barrios por medio del campo Bar_Codigo.
Cat_TelefonoCli		Varchar(9)	YES	Almacena el número telefónico del cliente en caso de que este los posea
Cat_FecIngCli		Datetime	YES	Almacena la fecha en que el usuario fue registrado en el sistema
Cat_ClasifCli		Varchar(10)	YES	
Cat_CicloFactCli		Numeric(9, 0)	YES	
Cat_CodigoLoc		Varchar(14)	YES	
Cat_SectorLoc		Varchar(2)	YES	
Cat_RutaLoc		Varchar(2)	YES	

Cat_ManzanaLoc		Varchar(2)	YES	
Cat_SecuenciaLoc		Varchar(4)	YES	
Cat_PisoLoc		Varchar(2)	YES	
Cat_DptoLoc		Varchar(2)	YES	
Cat_CodCatLoc		Varchar(18)	YES	" Cat_CodLog: El código de catastro, según lo impuesto por las municipalidades,
Cat_ClaAntLoc		Varchar(18)	YES	
Cat_UbiLoc		Varchar(25)	YES	
Cat_DisponeAg		Varchar(25)	YES	
Cat_DiametroAg		Varchar(25)	YES	
Cat_EstConexAg		Varchar(25)	YES	
Cat_TipoAbastAg		Varchar(25)	YES	
Cat_FecServAg		Datetime	YES	
Cat_FacturaAg		Varchar(10)	YES	
Cat_ConsPromAg		Numeric(9, 0)	YES	
Cat_FormaPagoAg		Varchar(10)	YES	
Cat_DisponeAl		Varchar(25)	YES	
Cat_DiametroAl		Varchar(25)	YES	
Cat_UsoConexAl		Numeric(9, 0)	YES	Se trata de una llave foránea la misma que nos podemos enlazarnos con la tabla Uso_Conexión para de esta forma acceder al campo UsoCo_Desc, el mismo que contiene una descripción del tipo de conexión para el Alcantarillado del usuario.
Cat_EstadoClienteAl		Varchar(25)	YES	
Cat_FecServAl		Datetime	YES	
Cat_FacturaAl		Varchar(10)	YES	
Cat_Factura1Al		Varchar(10)	YES	

Cat_BaseCalcAl		Numeric(9, 0)	YES	
Cat_ConsBaseAl		Numeric(9, 0)	YES	
Cat_DisponeMed		Varchar(25)	YES	
Cat_MarcaMed		Numeric(9, 0)	YES	
Cat_UbicacionMed		Varchar(25)	YES	Se Detalla sobre el tipo de ubicación en la que se encuentra el medidor (0-No tiene; 1-Buena)
Cat_EstadoMed		Varchar(25)	YES	Para conocer el estado actual del medidor 0- SIN MEDIDOR 1-FUNCIONANDO
Cat_NumeroMed		Varchar(10)	YES	Almacena el Número de Medidor
Cat_NumCifrasMed		Numeric(9, 0)	YES	
Cat_NumFliasDes		Numeric(9, 0)	YES	
Cat_NumPerDes		Numeric(9, 0)	YES	
Cat_DescDes		Varchar(25)	YES	
Cat_DesCli		Char(10)	YES	

CONFIG_PARAM Atributos

Nombre	Dominio	Tipo de dato	NULL	Definición
ConPa_AñoProce		Varchar(4)	NO	Se trata del año en proceso, es decir el año que va a contener todos los parámetros de configuración.
ConPa_PorcenCri		Numeric(9, 0)	YES	
ConPa_SisCo		Varchar(15)	YES	Por medio de este campo se obtendrán dos valores (1-2), por medio de estos se sabrá: o 1: Sistema de cobro con valor fijo, en este caso obligatoriamente se tendrá que coger tres campos (Alcantarillado1, Basura1, Medidor1),

				de la tabla Config_Param_Año, directamente para el cálculo de los servicios de agua potable, alcantarillado y basura. o 2: Sistema de cobro con valor diferenciado.
ConPa_Fecha		Datetime	YES	
ConPa_Coment		Varchar(200)	YES	
ConPa_Adm		Numeric(9, 2)	YES	Valor de cobro por concepto de manejos administrativos.
ConPa_Otros		Numeric(18, 4)	YES	
ConPa_Proc		Numeric(9, 2)	YES	
ConPa_Conex		Numeric(9, 2)	YES	Valores a ser cancelados por el concepto de una nueva conexión.
ConPa_Reconex		Numeric(9, 2)	YES	Valores a ser cancelados por el concepto de una re conexión del medidor.
ConPa_Multas		Numeric(9, 2)	YES	Valores a ser cancelados por el concepto de multas el pago a destiempo de los servicios
ConPa_MesesCorte		Numeric(9, 0)	YES	Se almacenan los meses que lleva de corte el medidor el mención.
ConPa_MesesMora		Numeric(9, 0)	YES	Se almacenan los meses que lleva de mora un usuario para el pago de los servicios.
ConPa_ValorFijoAg		Numeric(9, 2)	YES	" ConPa_ValorFijoAg, ConPa_ValorFijoAlc, ConPa_ValorFijoBas: Estos tres campos funcionan en conjunto puesto que si ConPa_SisCo tiene el valor de 1, estos tendrán la opción de tener valores distintos de cero, encaso de que los tres campos

				tengan valor igual acero se procederá a tomar los valores de la tabla Config_Param_Año.
ConPa_ValorFijoAlc		Numeric(9, 2)	YES	" ConPa_ValorFijoAg, ConPa_ValorFijoAlc, ConPa_ValorFijoBas: Estos tres campos funcionan en conjunto puesto que si ConPa_SisCo tiene el valor de 1, estos tendrán la opción de tener valores distintos de cero, encaso de que los tres campos tengan valor igual acero se procederá a tomar los valores de la tabla Config_Param_Año.
ConPa_ValorFijoBas		Numeric(9, 2)	YES	" ConPa_ValorFijoAg, ConPa_ValorFijoAlc, ConPa_ValorFijoBas: Estos tres campos funcionan en conjunto puesto que si ConPa_SisCo tiene el valor de 1, estos tendrán la opción de tener valores distintos de cero, encaso de que los tres campos tengan valor igual acero se procederá a tomar los valores de la tabla Config_Param_Año.
ConPa_PocentAlc		Numeric(9, 2)	YES	En este campo se almacena el porcentaje para ser calculado con el consumo del agua potable para el servicio de alcantarillado.
ConPa_PorcentBas		Numeric(9, 2)	YES	En este campo se almacena el porcentaje para ser calculado con el consumo del agua potable para el servicio de basura.
ConPa_ValFijoAlc1		Numeric(9, 2)	YES	
ConPa_ValFijoBas1		Numeric(9, 2)	YES	

ConPa_TipoAlc		Varchar(1)	YES	" ConPa_TipoAlc y ConPa_TipoAg: Estos dos valores funcionan en conjunto, pues por medio de estos se pueden determinar los valores a cancelar para el alcantarillado y la basura, siempre y cuando se esté calculando sistemas de cobro con valor diferenciado. Estos campos pueden obtener los valores de 1, 2, 3 y 4.
ConPa_TipoAg		Varchar(1)	YES	" ConPa_TipoAlc y ConPa_TipoAg: Estos dos valores funcionan en conjunto, pues por medio de estos se pueden determinar los valores a cancelar para el alcantarillado y la basura, siempre y cuando se esté calculando sistemas de cobro con valor diferenciado. Estos campos pueden obtener los valores de 1, 2, 3 y 4.

CONFIG_PARAM_AÑO Atributos

Nombre	Dominio	Tipo de dato	NULL	Definición
ConPa_AñoProce		Varchar(4)	NO	Se trata del año en proceso, es decir el año que va a contener todos los parámetros de configuración.
ConPaA_Tipo		Varchar(20)	NO	Campo que almacena el tipo de servicio al que se hace referencia (Alcantarillado, Basura, y Medidor), siendo este último usado para referirse cuando el medidor está dañado, etc.
ConPaA_BasDom		Numeric(9, 2)	YES	Campo necesario para el

				almacenamiento de valores para las conexiones de tipo domésticas.
ConPaA_BasCom		Numeric(9, 2)	YES	Campo necesario para el almacenamiento de valores para las conexiones de tipo comerciales.
ConPaA_BasInd		Numeric(9, 2)	YES	Campo necesario para el almacenamiento de valores para las conexiones de tipo industriales.
ConPaA_BasOfi		Numeric(9, 2)	YES	Campo necesario para el almacenamiento de valores para las conexiones de tipo Oficiales.
ConPaA_BasTra		Numeric(9, 2)	YES	Campo necesario para el almacenamiento de valores para las conexiones de tipo Tercera Edad.
ConPaA_BasOtr		Numeric(9, 2)	YES	Campo necesario para el almacenamiento de valores para otro tipo de conexiones.

CONTRATO Atributos

Nombre	Dominio	Tipo de Dato	NULL	Definición
Con_Numero		Numeric(9, 0)	NO	Se almacena el número de contrato, es un campo automático.
Con_Servicio		Varchar(30)	NO	Se trata de una clave foránea, en la cual se referencian el tipo de contrato para ente caso Venta de Medidor.
Cli_NumCuenta		Varchar(8)	NO	Clave Foránea para el almacenamiento del número de cuenta del cliente.
Con_Fecha		Datetime	YES	Fecha en que se realizó el contrato.
Con_FormaPago		Varchar(10)	YES	Se registra la forma de pago para la venta del medidor (Contado,

				Crédito).
Con_Valor		Numeric(9, 2)	YES	Valor acordado a ser cancelado por la venta del medidor.
Con_ValContado		Numeric(9, 2)	YES	Valor ingresado por el pago de contado.
Con_ValCredito		Numeric(9, 2)	YES	Valor ingresado por el pago a crédito.
Con_Abono		Numeric(9, 2)	YES	Valor abonado para el pago a crédito.
Con_CuotasPlazo		Numeric(9, 0)	YES	Número de cuotas plazo a ser pagadas.
Con_CuotaValor		Numeric(9, 2)	YES	Valor resultante al dividir el valor a pagar restante por el número de meses
Con_CuotaNum		Numeric(9, 0)	YES	
Con_ValAPagar		Numeric(9, 2)	YES	Valor restante a pagar.
Con_Interes		Numeric(9, 2)	YES	
Con_FecPagCont		Datetime	YES	Fecha de pago.
Con_ValPagCont		Numeric(9, 2)	YES	Valor pagado como cuota.
Con_AreaAten		Varchar(15)	YES	Área en la que ha sido atendido el usuario.
Con_Responsable		Varchar(60)	YES	Persona responsable de la venta del medidor.
Con_FecPlanAten		Datetime	YES	
Con_EstAten		Varchar(15)	YES	
Con_Desc		Varchar(80)	YES	
Con_EstFinanc		Varchar(15)	YES	
Con_Estado		Varchar(15)	YES	
Con_Mesproceso		Numeric(9, 0)	YES	Mes en el que se está procesando la venta del medidor.
Con_Usuario		Numeric(5, 0)	YES	
Con_Iva		Numeric(5, 2)	YES	

Con_Descuento		Numeric(5, 2)	YES	

LECTURAS Atributos

Nombre	Dominio	Tipo de Dato	NULL	Definición
Cli_NumCuenta		Varchar(8)	NO	Se trata de una clave foránea la cual sirve como nexo para conectarnos con la tabla Catastro.
Lec_Año		Varchar(4)	NO	Se almacena el año de la lectura a ser procesada.
Lec_Mes		Varchar(2)	NO	Se almacena el mes de la lectura a ser procesada.
Lec_Ciclo		Varchar(10)	YES	
Lec_FecLec		Datetime	YES	Se almacena la fecha en laque se realizó la lectura.
Lec_ValLec		Numeric(18, 0)	YES	En este campo se almacenará el valor de la lectura actual.
Lec_Comentario		Varchar(20)	YES	
Lec_Obs		Varchar(30)	YES	Se almacenan las observaciones para cada lectura.
Lec_Consumo		Numeric(18, 0)	YES	Aquí se registra el consumo total del agua potable que viene dado por la resta de la lectura actual menos la lectura anterior.
Lec_TipCons		Varchar(10)	YES	Se almacena el tipo de consumo, si está leído o no un determinado medidor.
Lec_DiasCons		Numeric(18, 0)	YES	
Lec_Usuario		Numeric(5, 0)	YES	Se trata del código del usuario que realizó la lectura del medidor.
Lec_Estado		Varchar(1)	YES	Se trata del estado en que se

				encuentra el medidor en lo que refiere a su lectura.

SALDO Atributos

Nombre	Dominio	Tipo de Datos	NULL	Definition
Con_Numero		Numeric(9, 0)	NO	Número de contrato para la compra venta de un medidor.
Sal_SaldoContra		Numeric(9, 2)	NO	Saldo a pagar, por la compra venta de un medidor.
Sal_NumCuotas		Numeric(9, 0)	NO	Número de cuotas acordadas para el pago.
Sal_CuotasPag		Numeric(18, 0)	NO	Cuotas pagadas.
Sal_ValPagado		Numeric(9, 2)	YES	
Sal_Interes		Numeric(9, 2)	YES	
Sal_FecAct		Datetime	YES	Fecha del pago de una cuota.
Sal_desc		Varchar(10)	YES	
Con_Servicio		Varchar(30)	YES	
Cli_NumCuenta		Varchar(8)	NO	Número de cuenta del cliente.
Con_Usuario		Numeric(5, 0)	YES	Código del usuario del sistema.

USO_CONEXION Atributos

Nombre	Dominio	Tipo de Dato	NULL	Definición
UsoCo_Codigo		NUMERIC(9, 0)	NO	Se trata de la clave principal para esta tabla.
UsoCo_Desc		VARCHAR(30)	NO	Es en este campo en donde se almacenarán los tipos de conexiones antes citados.

UsoCo_FecIng		DATETIME	YES	Almacenamos la fecha en la que se ingresó el nuevo tipo de conexión.
UsoCo_Tipo		VARCHAR(30)	YES	Se almacenarán los tipos de servicio para cada tipo de conexión, (Agua, Alcantarillado).

Anexo 5. Sistemas de Cobro de los Servicios para GCS

Cobro del rubro por consumo del Agua Potable

Se debe considerar que para este fin, tenemos dos tipos de sistemas; basándonos para esto en la tabla Config _Param, y específicamente en el campo ConPa_SisCo, el mismo que puede adoptar dos valores (1-2).

- 1: Identificador para el cobro del servicio del Agua Potable con el sistema de Valor Fijo, en el mismo que se especifican valores constantes para este fin.

- 2: Identificador para el cobro del servicio del Agua Potable con el sistema de Valor Diferenciado, en el mismo que se especifican distintas formas de cálculo dependiendo del tipo de usuario, por ejemplo Doméstico, Industrial, etc.

Y dependiendo de la forma de cobro con la que cuente cada municipio, afectará también para el cobro de los demás servicios.

Sistema de Cobro con Valores Fijos para el cobro del servicio de Agua Potable, Alcantarillado y Basura.

Como se dijo anteriormente a este tipo de sistema se lo puede identificar con el código “1” en el campo ConPa_SisCo de la tabla Config_Param.

En este sentido como primer punto se deben tomar en cuenta los valores ConPa_ValorAg (Valor fijo para el cobro del Agua Potable), ConPa_ValorFijoAlc (Valor fijo para el cobro del servicio de alcantarillado), ConPa_ValorFijoBas (Valor fijo para el cobro del servicio de Basura) de la tabla Config_Param; para la emisión del recibo correspondien siempre y cuando los tres tengan valores distintos de cero.

En caso de que estos valores adopten el valor de cero en los tres casos, entonces se procederá a obtener los datos de la tabla Config_ParamAño, tomando para esto los campos ConPaA_BasDom (Campo usado para el almacenamiento de valores fijos para clientes de tipo Doméstico), ConPaA_BasCom (Campo usado para el almacenamiento de valores fijos para clientes de tipo Comercial), ConPaA_BasInd (Campo usado para el almacenamiento de valores fijos para clientes de tipo Industrial), ConPaA_BasOfi (Campo usado para el almacenamiento de valores fijos para clientes de tipo Oficial), ConPaA_BasTra (Campo usado para el almacenamiento de valores fijos para clientes de tipo Tercera Edad), ConPaA_BasOtr (Campo usado para el almacenamiento de valores fijos para clientes de tipo Doméstico). Para dicho proceso primeramente se tomará en cuenta el valor de la variable UsoCo_Codigo de la tabla Catastro, con este código identificamos que tipo de cliente se trata.

Sistema de Cobro con Valor Diferenciado.

Como se describió en la parte anterior a este enunciado a este tipo de sistema se lo puede identificar con el código “2” en el campo ConPa_SisCo de la tabla Config_Param.

De igual manera para el cálculo del consumo de agua potable mediante este tipo de sistema se debe tener muy en cuenta la variable UsoCo_Codigo de la tabla Catastro (Variable que identifica al tipo de cliente, para el uso de los servicios de Agua Potable, Alcantarillado y Recolección de Basura).

Una vez obtenido el valor del capo UsoCo_Codigo de la tabla Catastro, procedemos a comparar con el campo UsoCo_Desc de la tabla Uso_Conexion en la cual se encuentran datos detallando la información de los tipos de conexión contratada por los clientes.

Una vez obtenida la información de este campo procedemos a compararla con el campo CTar_UsoConex (Campo que contiene la información detallada del tipo de contrato realizado por el cliente), de la tabla CTarifa, con este campo además se deberá obtener el tipo de servicio a ser calculado, para este caso práctico el servicio de agua potable, identificado CTar_Desc. Cabe resaltar también que se deberá obtener el año en el cual se va a realizar un proceso de cálculo, valor identificado en el campo CTar_AñoProce.

Con esta información hay que destacar que para cada año en proceso, municipio existe una forma diferente de realizar el cálculo de los determinados servicios. Es así que para cada año varían las formas de calcular el valor por el consumo de los distintos tipos de servicio.
Para la realización de futuros cálculos dados los datos anteriores nos basaremos en la información contenida en el campo CTar_Tipo, de la tabla CTarifa, de aquí se podrán clasificar en diferentes tipos de cálculo, dando así distintos valores de acuerdo al grado de configuración y rangos ingresados por el Administrador.

Se han obtenido una clasificación básica de acuerdo a las posibilidades de ingreso de datos en el campo CTar_Tipo en la Tabla CTarifa.

- ***Básica – Rangos:*** Este caso se da cuando la primera tupla de la tabla en cuestión contiene el valor de Básica y el resto de tuplas contiene el valor de Rangos. Para este tipo de posibilidad se realiza un cálculo de la siguiente forma.
    1. Contamos el número de filas que contiene la selección, ingresando como filtro en la tabla CTarifa (CTar_UsoConex, CTar_AñoProce, CTar_Desc). Y la almacenamos en una variable llamada num_filas, siempre y cuando el valor sea mayor que 1.
    2. Verificamos las dos primeras tuplas en el campo CTar_Tipo, que para este caso la primera debe ser igual a “Basica” y la segunda “Rangos”.
    3. Verificamos si nos encontramos en la primera tupla, para calcular en base al campo CTar_Tipo, que este caso será Básica.
    4. Almacenamos en una variable llamada ***Total*** el valor del campo CTar_Tarifa, que es el valor a cobrar si nos encontramos en este primer rango.
    5. Almacenamos el valor mínimo en la variable ***min***, que corresponde al campo CTar_ValMin, este valor es usado para el cálculo del rango contenido.
    6. Almacenamos el valor máximo en la variable ***max***, que corresponde al campo CTar_ValMax, este valor es usado para el cálculo del rango contenido.
    7. Almacenamos en la variable ***rango*** la diferencia entre ***max*** y ***min.***

8. Luego se almacena en la variable ***valor*** la diferencia entre la cantidad contenida en la variable ***valor*** que en su inicio es igual al consumo total en metros cúbicos de agua, y la variable ***rango***. Esta variable será usada para realizar un control del número de bucles que se debe realizar para el cálculo del valor a pagar por el consumo.
9. Para el caso de no encontrarnos en la primera tupla, igualmente recogemos las variables ***max*** y ***min,*** y calculamos el rango con la siguiente fórmula (max - min + 1).
10. Preguntamos si la variable ***valor*** es mayor que la variable ***rango,*** es decir comparamos si tenemos un excedente para dicho rango.
    - Para el caso de que esta aseveración sea positiva, procedemos a realizar el siguiente cálculo.
        - Recogemos el valor para la variable ***excedente*** del campo CTar_Excede. Y calculamos
        - ***total = total + (rango * excedente)***

      Con este cálculo se asegura la multiplicación del rango obtenido por un valor excedente fijado.
        - En el caso de ser negativa, es decir que no existe un excedente con respecto al rango calculado.
            - Recogemos el valor para la variable ***excedente*** del campo CTar_Excede. Y calculamos
            - ***total = total + (valor * excedente)***

      Con este calculo nos aseguramos de obtener un valor solamnente cuando no exista excedentes, es decir números menores al rango obtenido.
11. Finalmente el contro para el bucle es el siguiente: while ((***valor*** > 0) || (j > ***num_filas***)), donde se conrola que la variable ***valor***, sea mayor que 0 para poder seguir realizando los cálculos respectivos,es decir que siempre y cuando

exista un excedente, se podrá realizar los respectivos cómputos. En lo que se refiere a la variable j, es simplemente un contador para restringir el número total de filas que contiene la tabla seleccionada, dada los parámetros antes citados.

12. Finalmenete se obtendrá la variable ***total,*** que es el resultado conclusivo para el cálculo del consumo de agua potable.

- ***Mixta – Mixta:*** Este caso se da cuando la primera tupla de la tabla en cuestión contiene el valor de Mixta y el resto de tuplas contiene el valor de Mixta. Se trata de de una combinación con el sistema de cálculo anterior. Para este tipo de posibilidad se realiza un cálculo de la siguiente forma.

    1. Contamos el número de filas que contiene la selección, ingresando como filtro en la tabla CTarifa (CTar_UsoConex, CTar_AñoProce, CTar_Desc). Y la almacenamos en una variable llamada num_filas, siempre y cuando el valor sea mayor que 1.
    2. Verificamos las dos primeras tuplas en el campo CTar_Tipo, que para este caso la primera debe ser igual a “Mixta” y la segunda “Mixta”.
    3. Verificamos si nos encontramos en la primera tupla, para calcular en base al campo CTar_Tipo, que este caso será Básica.
    4. Almacenamos en una variable llamada ***Total*** el valor del campo CTar_Tarifa, que es el valor a cobrar si nos encontramos en este primer rango.
    5. Almacenamos el valor mínimo en la variable ***min***, que corresponde al campo CTar_ValMin, este valor es usado para el cálculo del rango contenido.
    6. Almacenamos el valor máximo en la variable ***max***, que corresponde al campo CTar_ValMax, este valor es usado para el cálculo del rango contenido.
    7. Almacenamos en la variable ***rango*** la diferencia entre ***max*** y ***min.***
    8. Luego se almacena en la variable ***valor*** la diferencia entre la cantidad contenida en la variable ***valor*** que en su inicio es igual al consumo total en

metros cúbicos de agua, y la variable ***rango***. Esta variable será usada para realizar un control del número de bucles que se debe realizar para el cálculo del valor a pagar por el consumo.

9. Para el caso de no encontrarnos en la primera tupla, igualmente recogemos las variables ***max*** y ***min,*** y calculamos el rango con la siguiente fórmula (max - min + 1).
10. Preguntamos si la variable ***valor*** es mayor que la variable ***rango,*** es decir comparamos si tenemos un excedente para dicho rango.
    - Para el caso de que esta aseveración sea positiva, procedemos a realizar el siguiente cálculo.
        - Recogemos el valor para la variable ***excedente*** del campo CTar_Excede. Y calculamos

    total = total + tarifa + (rango * exedente)

    En este tipo de cálculos siempre debemos tomar y tener en cuenta los valores excedentes, ya que de esta manera se obtendrán valores mas elevados para la variable ***total.***

    - En el caso de ser negativa, es decir que no existe un excedente con respecto al rango calculado.
        - Recogemos el valor para la variable ***excedente*** del campo CTar_Excede. Y calculamos

    total = total + tarifa + (valor * exedente)

13. Finalmente el contro para el bucle es el siguiente: while ((***valor*** > 0) || (j > ***num_filas***)), donde se conrola que la variable ***valor***, sea mayor que 0 para poder seguir realizando los cálculos respectivos,es decir que siempre y cuando exista un excedente, se podrá realizar los respectivos cómputos. En lo que se refiere a la variable j, es simplemente un contador para restringir el número

total de filas que contiene la tabla seleccionada, dada los parámetros antes citados.

14. Finalmenete se obtendrá la variable ***total,*** que es el resultado conclusivo para el cálculo del consumo de agua potable.

- **Cuando Existen Calculos Tarifarios con un solo rango de Valores*:*** Este caso se da cuando existe una sola tupla que contiene en solo rango generalmente de 0 a 9999999, para un cobro unitario y único sea cual sea el consumo del servicio. La variable total simplemente será el campo CTar_Tarifa.

Cobro del rubro por consumo del Alcantarillado y Basura.

Sistema de Cobro con Valor Diferenciado.

Para este tipo de cálculo, primeramente nos basamos y sacamos datos de la tabla Config_Param, específicamente trabajaremos con los campos ConPa_TipoAlc, ConPa_TipoAg, con los cuales podremos identificar los distintos tipos de combinaciones para el cálculo del servicio de recoleccion de basura y aclantarillado.

En la siguiente tabla se pueden observar los distintos valores que pueden tomar las campos ConPa_TipoAlc, ConPa_TipoAg, para los distintos tipos de descripciones.

Porcentaje (1): Con este parámetro nos aseguramos que el servicio a ser calculado representa un cierto porcentaje del valor total a cobrarse por el consumo de agua potable.

Valor Fijo (2): Se realizan los calculos,respectivos reutilizando la tabla Config_Param, sacando los valores ConPa_ValFijoAlc1, ConPa_ValFijoBas1, almacenadosen estas.

Valor Fijo Diferenciado (3): Se obtiene este valor de la tabla config_param_Año, especificamente de los campos ConPaA_BasDom (clientes de tipo domésticos), C onPaA_BasCom (clientes de tipo Comerciales) o ConPaA_BasInd (clientes de tipo Industriales), ConPaA_BasOfi (clientes de tipo oficiales), ConPaA_BasTra ((clientes de la terceraedad)) u Otros (Otro tipo de clientes), dependiendo el tipo de cliente se trabajará con un campo especificado.

Estructura Tarifaria (4): Con este parametro se calculará las tarifas a pagar para los servicios de alcantarillado y basura, mediante el Sistema de Cobro con Valor Diferenciado; al igual que para el agua potable.

Alcantarillado	**Basura**	**ConPa_TipoAlc**	**ConPa_TipoAg**
Porcentaje	Porcentaje	1	1
Porcentaje	Estructura Tarifaria	1	4
Porcentaje	Valor Fijo	1	2
Porcentaje	Valor Fijo Diferenciado	1	3
Estructura Tarifaria	Porcentaje	4	1
Estructura Tarifaria	Estructura Tarifaria	4	4
Estructura Tarifaria	Valor Fijo	4	2
Estructura Tarifaria	Valor Fijo Diferenciado	4	3
Valor Fijo	Porcentaje	2	1
Valor Fijo	Estructura Tarifaria	2	4
Valor Fijo	Valor Fijo	2	2
Valor Fijo	Valor Fijo Diferenciado	2	3
Valor Fijo Diferenciado	Porcentaje	3	1
Valor Fijo Diferenciado	Estructura Tarifaria	3	4
Valor Fijo Diferenciado	Valor Fijo	3	2
Valor Fijo Diferenciado	Valor Fijo Diferenciado	3	3