ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA EN SISTEMAS

"ANÁLISIS COMPARATIVO DE LA PRODUCTIVIDAD ENTRE LOS PATRONES DE DISEÑO MODELO VISTA CONTROLADOR (MVC) Y MODELO VISTA PRESENTADOR (MVP) APLICADO AL DESARROLLO DEL SISTEMA NÓMINA DE EMPLEADOS Y ROL DE PAGOS DE LA "DISTRIBUIDORA SORIA C.A" "

TESIS DE GRADO

Previa la obtención del título de

INGENIERO EN SISTEMAS INFORMÁTICOS

Presentado por:

Jenny Germania Carrera Guanoluisa

Riobamba – Ecuador
2014

AGRADECIMIENTO

En primer lugar *a Dios*, por cuidar todos mis pasos y guiar mi camino, por protegerme y darme fuerzas para superar obstáculos y dificultades a lo largo de toda mi carrera estudiantil.

A mis padres, por apoyarme en toda etapa de mi vida, por ser un ejemplo de trabajo, superación y haberme brindado amor, cariño y comprensión en todo el trayecto de mi vida.

A mi amado esposo Jorge, por su amor, confianza y apoyo incondicional, para lograr esta meta ya culminada.

A Dr. Julio Santillán, director de tesis, por su valiosa guía y asesoramiento hicieron posible la culminación de la tesis y por su gran calidad humana.

Jenny Carrera

DEDICATORIA

A mi Padre Dios, por permitirme llegar a este momento tan especial en mi vida, por los triunfos y los momentos difíciles que me han enseñado a valorarlo cada día más.

A mis padres, Silvia y Segundo quienes son los pilares fundamentales de apoyo en mi vida, con sus sabios consejos e infinito amor han sabido guiarme.

A mi amado esposo Jorge, por su amor, paciencia y apoyo incondicional por ser el protagonista principal en todos mis logros obtenidos en mi vida profesional.

A mi amada hija Lesly, por ser mi fuerza y motor para haber culminado esta etapa de mi vida.

Con mucho amor y cariño para ustedes

Jenny Carrera

FIRMAS RESPONSABLES Y NOTAS

NOMBRES	FIRMAS	FECHA
Ing. Gonzalo Samaniego **DECANO DE LA FACULTAD DE INFORMÁTICA Y ELECTRÓNICA**	_____	_____
Dr. Julio Santillán Castillo **DIRECTOR DE ESCUELA DE INGENIERÍA EN SISTEMAS**	_____	_____
Dr. Julio Santillán **DIRECTOR DE TESIS**	_____	_____
Ing. Paúl Paguay **MIEMBRO DEL TRIBUNAL**	_____	_____
DIRECTOR DEL CENTRO DE DOCUMENTACIÓN	_____	_____

Nota De La Tesis: _____

RESPONSABILIDAD DEL AUTOR

Yo **Jenny Germania Carrera Guanoluisa,** soy la responsable de las ideas, doctrinas y resultados expuestos en esta Tesis de Grado y el patrimonio intelectual de la misma pertenece a la **"ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO**".

_____

Jenny Germania Carrera Guanoluisa

ÍNDICE DE ABREVIATURAS

API: Application Programming Interface (Interfaz de Programación de Aplicaciones).

BD: Base de Datos.

DAO: Data Access Object (Objeto de Acceso a Datos)

DDL: Data Definition Language (Lenguaje de definición de Datos).

DML: Data Manipulation Language (Lenguaje de Manipulación de Datos).

HTML: HyperText Markup Language (Lenguaje de Marcado Hipertextual)

HTTP: HyperText Transfer Protocol (Protocolo de Transferencia De Hipertexto)

IDE: Integrated Development Environment (Entorno Integrado de Desarrollo)

MSF: Microsoft Solution Framework.

MVC: Modelo Vista Controlador

MVP: Modelo Vista Presentador

POO: Programación Orientada a Objetos

RDBMS: Relational Database Management System (Sistema de Administración de Base de Datos Relacional).

RUP: Rational Unified Process (Proceso Unificado de Racional)

SQL: Structured Query Language (Lenguaje de Consulta Estructurados)

TCP/IP: *Transmission Control Protocol/Internet Protocol (*Protocolo de Control de Transmisión/Protocolo de Internet*)*

URL: Uniform Resource Locator (Localizador de Recursos Uniforme)

WWW: World Wide Web, Sistema de Documentos de Hipertexto

XML: Extensible Markup Language (Lenguaje de Marcas Extensibles)

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

El presente trabajo de investigación trata sobre el **Análisis comparativo de la productividad entre los Patrones de Diseño Modelo Vista Controlador (MVC) Y Modelo Vista Presentador (MVP)** y el posterior desarrollo de un sistema denominado Nómina de Empleados y Rol de Pagos para la Distribuidora SORIA C.A, para lo cual nos enfocaremos principalmente en Patrones de Diseño, porque son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software.

Para realizar un correcto proceso de investigación del tema planteado, es necesario primero analizar sus causas: una de ellas y se podría decir que la principal es que en la actualidad las empresas tanto pequeñas, grandes, privadas o públicas, no llevan un control adecuado, ordenado y sobre todo automatizado en cuanto a sus empleados y sus respectivos salarios.

En base a lo expuesto anteriormente, en el capítulo 1 se trata el Marco Referencial, así se tiene la problematización, justificación del proyecto de tesis, objetivos e hipótesis. En el capítulo 2 se detalla el Marco Teórico, definiendo todo lo relacionado a los Patrones de Diseño, historia, definición, ventajas, desventajas, característica, objetivos. En el capítulo 3 se hace el Análisis Comparativo entre los patrones de diseño propuestos, detallando todo lo referente a Modelo Vista Controlador y Modelo Vista Presentador, definiendo cada variable con sus respectivos parámetros e indicadores. En el capítulo 4 se estudia brevemente lo referente a la empresa para la cual se va a desarrollar el sistema, y posterior a esto se hace

el Análisis de Requerimientos y se detalla las tareas involucradas en los procesos de Documentación y Evaluación del Sistema.

CAPÍTULO I

MARCO REFERENCIAL

1.1 Antecedentes

Planteamiento del Problema

En la actualidad, debido al ritmo competitivo que se da tanto en los mercados nacionales como internacionales y desde luego en la industria, donde la globalización ya no es una alternativa de crecimiento sino una realidad que debe ser asumida para no quedar rezagados frente a otros competidores y para no perder oportunidades de crecimiento económico, es fundamental cambiar la visión de las formas bajo las cuales se crea y se sostiene la empresa.

La productividad es un aspecto importante en las empresas que se dedican al desarrollo de software, las cuales tratan de utilizar de manera eficiente los recursos como tiempo, dinero y personal, para obtener la mayor cantidad de productos (software) y así lograr mayores beneficios. Un problema constante de discusión y análisis para los programadores es determinar las herramientas que permiten la mejor productividad, en lo que respecta a programación, diseño, facilidad de uso, costo, entre otros factores.

Cabe destacar, que en las instituciones públicas y privadas el fortalecimiento de los procesos automatizados se ha vuelto una prioridad, uno de los objetivos es mejorar las atención a la ciudadanía en general, proporcionando el acceso a internet y a la tecnología; que no se ha empezado aún por explotar en gran medida el uso de la misma, es por ello que no se ha percibido en gran medida el impacto, los beneficios de las diferentes aplicaciones software.

Se ha adoptado la utilización de Patrones de Diseño, porque son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces, entre ellos podemos mencionar MVC y el MVP.

Análisis

Para la creación ágil y ordenada de un sistema se realizará un análisis profundo de los patrones de diseño MVC y MVP, para ello se estudiarán sus diferentes partes como:

- Modelo
- Vista
- Controlador

Lugar de aplicación

Distribuidora SORIA C.A

Alcance

Este tema de tesis se enmarca en el análisis de los patrones de diseño MVC y MVP para el desarrollo de aplicaciones web, su estructura, funcionalidades, facilidades que brindan y el uso de estos para desarrollar un sistema de nómina de empleados y rol de pagos. Se realizará un estudio para determinar la productividad en el desarrollo de aplicaciones web para determinar el patrón de diseño con mayor productividad en el desarrollo de la aplicación para la "Distribuidora SORIA C.A"

Limitación

La falta de experiencia y conocimiento de patrones de diseño para el desarrollo de aplicaciones web puede generar un desfase de las actividades planificadas. El personal de la "Distribuidora SORIA C.A" deberá recibir capacitación sobre el manejo del software que se creara, ya que actualmente no cuentan con un sistema informático que les ayude de forma automatizada en sus funciones cotidianas dentro de la empresa.

1.2 Objetivos

1.2.1 Objetivo General

Realizar un análisis comparativo entre los patrones de diseño Modelo Vista Controlador (MVC) y Modelo Vista Presentador (MVP) para determinar el patrón de diseño con mayor productividad en el desarrollo de la aplicación web para la "Distribuidora SORIA C.A"

1.2.2 Objetivos Específicos

- Estudiar los Patrones de Diseño MVC (Modelo Vista Controlador) y MVP (Modelo Vista Presentador).
- Determinar los parámetros de comparación para el desarrollo de los patrones de diseño MVC y MVP.
- Evaluar los Patrones de Diseño y seleccionar el mejor conforme a los parámetros de comparación establecidos y las pruebas realizadas en el prototipo del Módulo Rol de Pagos.
- Desarrollar una aplicación web en base al Patrón de Diseño seleccionado para la Distribuidora SORIA C.A.

1.3 Justificación de la Investigación

1.3.1 Justificación Teórica

En la actualidad las empresas de desarrollo de software necesitan optimizar recursos para ello necesitan escoger el patrón de diseño más productivo que permita obtener mejores beneficios para la empresa. En el desarrollo para las distintas aplicaciones se necesita tener conocimiento amplio para tener una visión detallada de cuál es el patrón de diseño más productivo y que otorgue ventajas al desarrollo de software; y que de valor agregado al negocio o usuario final.

El presente proyecto de tesis determinará el tipo de patrón de diseño que nos proporcione una mejor productividad para el desarrollo de la aplicación y mediante

el análisis comparativo será el que utilizaremos para implementar en el producto final.

1.3.2 Justificación Metodológica

Para dar soporte a la realización de este proyecto se pretende desarrollar dos prototipos utilizando los Patrones de Diseño MVC y MVP de un mismo módulo el cual será una parte del sistema (**MÓDULO ROL DE PAGOS**) con el que se realizarán las pruebas necesarias, ya que es la mejor forma de evaluar y al mismo tiempo comprobar si el proyecto tuvo éxito.

Para el desarrollo del proyecto se plantearan ciertos parámetros como son: líneas de código, tiempo de desarrollo, los cuáles nos ayudaran a medir la productividad de nuestra aplicación de desarrollo y de esta manera optar por la mejor opción.

1.3.3 Justificación Práctica

La Distribuidora "SORIA C.A" actualmente efectúa el proceso de nómina de empleados y rol de pagos manualmente, esto implica un proceso tedioso y dificultades para gestionar los diferentes actividades como emitir la nómina de los empleados y roles de pagos de los mismos, por lo que la implementación de un sistema que automatice mencionados procesos ayudaría a un mejor control adecuado de los empleados y roles de pagos, mejorando el servicio de los usuarios de la empresa y de esta manera almacenar de forma ordenada toda la información perfeccionando la eficiencia y productividad de la empresa.

1.4 Hipótesis

La adopción del patrón MVP, mejorará la productividad frente al patrón MVC, en el desarrollo de una aplicación software web.

CAPÍTULO II

2. MARCO TEÓRICO

2.1 Patrones de Diseño

Normalmente cuando se codifica hay pequeños trozos de código que se crean, a veces, sin pensar en ellos porque resultan evidentes. Si ese código sólo sirve para ese programa, todo va más o menos bien.

Sin embargo, es normal que luego se tenga que hacer otro programa similar. Es bastante normal también que se copie el primer programa o trozos del mismo en otro sitio y se empiece a modificar, porque el código que se debe hacer se parece al que ya se tenía, pero no es exactamente igual.

Cuando un programador se ve obligado a hacer eso una y otra vez, a copiar y pegar el mismo código y modificarlo una y otra vez, empieza a pensar en la forma de hacer ese trozo de código de forma que la tarea de llevárselo a otro sitio sea más cómoda para él.

El programador empieza a inventar diferentes formas de hacer ese trozo de código de forma que sea lo suficientemente versátil y configurable como para no tener que volver a tocarlo nunca más. Idealmente, le debería bastar con llevarse la librería ya compilada de un programa a otro.

Ha habido en la historia miles de programadores con este problema y miles de ellos han llegado a las mismas soluciones o formas de hacer esos trozos de código de forma que sean completamente reutilizables. Esas soluciones son los "patrones de diseño".

Una de las cosas que los programadores y desarrolladores han aprendido en base a la experiencia es que no deben resolver los problemas como si fuese la primera vez que se presentan. Para lograrlo, reutilizan el código que desarrollaron en el pasado, es decir que, cuando encuentran una buena solución a un problema concreto, recurren a ella una y otra vez. Los patrones de diseño suelen resolver problemas específicos del diseño y hacen que éste sea más flexible, elegante y reutilizable.

2.1.1 Historia

La idea de los patrones de diseño de software nació originalmente del campo de arquitectura. Christopher Alexander, un arquitecto, escribió dos libros revolucionarios que describen los patrones de arquitectura y de planificación urbana: A Pattern Lenguaje: pueblos, edificios, construcción (1977) y The TimelessWay of Building (1979).

Las ideas presentadas en esos libros son aplicables a varios campos fuera de arquitectura, incluso el desarrollo del software.

En 1987, Ward Cunningham y Kent Beck usaron algunas de las ideas de Alexander para desarrollar cinco modelos para el diseño de interfaces de usuario. Ellos publicaron un estudio en OOPSLA-87 titulado "Using Pattern Languages for Object-Oriented Programs."

En los años noventa, Erich Gamma, Richard Helm, John Vlissides, and Ralph Johnson empezaron a trabajar en uno de los libros de computación más influyente de la última década: Design Patterns. Publicado en 1994 y a menudo llamado la "Gang of Four" (Banda de los Cuatro) o GoF, este popularizó la idea de patrones de diseño.

En los últimos anos los patrones de diseño ha ido ganado aceptación, y se fueron extendiendo a otras áreas dentro del desarrollo y mantenimiento de software.

2.1.2 Definición

Un Patrón de Diseño es una solución repetible a un problema recurrente en el diseño de software. Esta solución no es un diseño terminado que puede traducirse directamente a código, sino más bien una descripción sobre cómo resolver el problema, la cual puede ser utilizada en diversas situaciones.

Los patrones de diseño reflejan todo el rediseño y re-modificación que los desarrolladores han ido haciendo a medida que intentaban conseguir mayor reutilización y flexibilidad en su software.

“Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software.”

Los patrones documentan y explican problemas de diseño, y luego discuten una buena solución a dicho problema. Con el tiempo, los patrones comienzan a incorporarse al conocimiento y experiencia colectiva de la industria del software, lo que demuestra que el origen de los mismos radica en la práctica misma más que en la teoría.

Christopher Alexander el arquitecto a quién se le atribuye el inicio de los patrones de diseño, los describe como: “cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, así como la solución a ese problema, de tal modo que se pueda aplicar esta solución un millón de veces, sin hacer lo mismo dos veces”.

Si bien ésta definición es sobre patrones de ciudades y edificios la idea es aplicable a la industria del software: encontrar una solución a un problema dentro de un contexto. Un patrón de diseño nomina, abstrae e identifica los aspectos clave de una estructura de diseño común, lo que los hace útiles para crear un diseño orientado a objetos reutilizable.

El patrón de diseño identifica las clases e instancias participantes, sus roles y colaboraciones, y la distribución de responsabilidades. Cada patrón de diseño se centra en un problema concreto, describiendo cuándo aplicarlo y si tiene sentido hacerlo teniendo en cuenta otras restricciones de diseño, así como las consecuencias, ventajas e inconvenientes de su uso.[1]

[1] http://es.scribd.com/doc/133467199/PATRONES-DE-DISEÑO

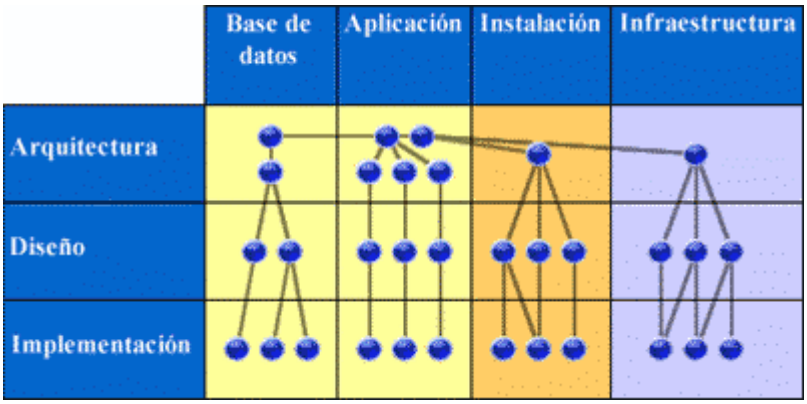


Figura II. 1 Marco de Patrones según Microsoft
Fuente: http://msdn.microsoft.com/es-es/library/bb972240.aspx

En el diagrama de la **Figura II. 1**, Microsoft sostiene que las columnas de la tabla son enfoques o puntos de vista de la solución, mientras que las filas representan los niveles de abstracción.

Un ejemplo muy conocido en el ambiente de desarrollo Web es el patrón de diseño de capas (*layers*), el cual plantea la separación de una aplicación en una capa de presentación, una de negocio y una de datos.[2]

2.1.3 Elementos de los Patrones de Diseño

En general, un patrón tiene cuatro elementos esenciales:

a. El **nombre del patrón** permite describir, en una o dos palabras, un problema de diseño junto con sus soluciones y consecuencias. Al dar nombre a un patrón inmediatamente estamos incrementado nuestro vocabulario de diseño, lo que nos

[2] http://msdn.microsoft.com/es-es/library/bb972240.aspx

permite diseñar a un mayor nivel de abstracción. Tener un vocabulario de patrones nos facilita pensar en nuestros diseños y transmitirlos a otros, juntos con sus ventajas e inconvenientes. Encontrar buenos nombres ha sido una de las partes más difíciles al desarrollar nuestro catálogo.

b. **El problema** describe cuando aplicar el patrón. Explica el problema y su contexto. Puede describir problemas concretos (por ejemplo, como representar algoritmos como objetos), así como las estructura de clases u objetos que son sintomáticas de un diseño inflexible. A veces, el problema incluye una serie de condiciones que deben darse para que tenga sentido aplicar el patrón.

c. **La solución** describe los elementos que constituyen el diseño, sus relaciones responsabilidades y colaboraciones. La solución no describe un diseño o una implementación en concreto, sino que un patrón es más bien como una plantilla que puede aplicarse en muchas situaciones diferentes. El patrón proporciona una descripción abstracta de un problema de diseño y como una lo resuelve una disposición general de elementos (en nuestro caso, clases y objetos).

d. **Las consecuencias** son los resultados así como las ventajas e inconvenientes de aplicar el patrón. Aunque cuando se describen decisiones de diseño muchas veces no se reflejan sus consecuencias, éstas son críticas para evaluar las alternativas de diseño y comprender los costes y beneficios de aplicar el patrón. Las consecuencias en el software suelen referirse al equilibrio entre espacio y tiempo. También pueden tratar cuestiones de lenguaje e implementación. Por otro lado, puesto que la reutilización suele ser uno de los factores de los diseños orientados a objetos, las consecuencias de un patrón incluyen su impacto sobre la

flexibilidad, extensibilidad y portabilidad de un sistema. Incluir estas consecuencias de un modo. [3]

2.1.4 Describir Patrones de Diseño

Para reutilizar el diseño, se debe hacer constar las decisiones, alternativas, ventajas e inconvenientes que dieron lugar a él. También son importantes los ejemplos concretos, porque ayudan a ver el diseño en acción. Es así que el libro "DesignPatterns: Elements of Reusable Object-Oriented Software" describe una plantilla la cual se encuentra dividida en secciones con el objetivo de facilitar el aprendizaje, comprensión y uso de los patrones de diseño.[4]

La plantilla propuesta para la descripción de patrones de diseño es la siguiente:

Nombre del Patrón y Clasificación

El nombre del patrón transmite brevemente su esencia. Un buen nombre es vital, porque pasará a formar parte del vocabulario de diseño. La clasificación del patrón refleja el esquema que se presenta en la sección

Propósito

Una frase breve que responde a las siguientes cuestiones: ¿Qué hace este patrón de diseño? ¿En qué se basa? ¿Cuál es el problema concreto de diseño que resuelve?

[3] http://ocw.unican.es/ensenanzas-tecnicas/ingenieria-del-software-i/materiales-de-clase-1/is1_t06_Patrones.pdf
[4] http://webdiis.unizar.es/~jmerse/IS-2/TeoriaPatronesV2_2.pdf

También conocido Como

Otros nombres, si existen, por los que se conoce al patrón.

Motivación

Un escenario que ilustra un problema de diseño y cómo las estructuras de clases y objetos del patrón resuelven el problema. El escenario ayudará a entender la descripción que sigue.

Aplicabilidad

¿En qué situaciones se puede aplicar el patrón de diseño? ¿Qué ejemplos hay de malos diseños que el patrón puede resolver? ¿Cómo se puede reconocer dichas situaciones?

Estructura

Una representación gráfica de las clases del patrón usando una notación basada en la Técnica de Modelado de Objetos. También se hace uso de diagramas de interacción para mostrar secuencias de peticiones y colaboraciones entre objetos.

Participantes

Las clases y objetos participantes en el patrón de diseño, junto con sus responsabilidades.

Colaboraciones

Como colaboran los participantes para llevar a cabo sus responsabilidades

Consecuencias

¿Cómo consigue el patrón sus objetivos? ¿Cuáles son las ventajas e inconvenientes y los resultados de usar el patrón? ¿Qué aspectos de la estructura del sistema se pueden modificar de forma independiente?

Implementación

¿Cuáles son las dificultades, trucos o técnicas que se deberían tener en cuenta a la hora de aplicar el patrón? ¿Hay cuestiones específicas del lenguaje?

Código de Ejemplo

Fragmentos de código que muestran cómo se puede implementar el patrón.

Usos Conocidos

Ejemplos de los patrones en sistemas reales

2.1.5 Tipos de Patrones de Diseño

Los Patrones generalmente se clasifican en:

- De Creación
- Estructurales
- De Comportamiento

2.1.5.1 De Creación: Abstraen el proceso de creación de instancias de objetos. Ayudan a hacer a un sistema independiente de cómo se crean, se componen y se representan sus objetos.

- Abstract Factory (Fábrica abstracta): Permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando.
- Builder (Constructor virtual): Abstrae el proceso de creación de un objeto complejo, centralizando dicho proceso en un único punto.
- Factory Method (Método de fabricación): Centraliza en una clase constructora la creación de objetos de un subtipo de un tipo determinado, ocultando al usuario la casuística para elegir el subtipo que crear.
- Prototype (Prototipo): Crea nuevos objetos clonándolos de una instancia ya existente.
- Singleton (Instancia única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia.

2.1.5.2 Estructurales: Tratan con la composición de clases u objetos. Se ocupan de cómo las clases y objetos son utilizados para componer estructuras de mayor tamaño.

- Adapter (Adaptador): Adapta una interfaz para que pueda ser utilizada por una clase que de otro modo no podría utilizarla.
- Bridge (Puente): Desacopla una abstracción de su implementación.

- Composite (Objeto compuesto): Permite tratar objetos compuestos como si se trate de un simple.
- Decorator (Envoltorio): Añade funcionalidad a una clase dinámicamente.
- Facade (Fachada): Provee de una interfaz unificada simple para acceder a una interfaz o grupo de interfaces de un subsistema.
- Flyweight (Peso ligero): Reduce la redundancia cuando gran cantidad de objetos poseen idéntica información.
- Proxy: Mantiene un representante de un objeto.

2.1.5.3 De Comportamiento: Caracterizan el modo en que las clases y objetos interactúan y se reparten la responsabilidad. Atañen a los algoritmos y a la asignación de responsabilidades entre objetos. [5]

- Chain of Responsability (Cadena de Responsabilidad): Permite establecer la línea que deben llevar los mensajes para que los objetos realicen la tarea indicada.
- Command (Orden): Encapsula una operación en un objeto, permitiendo ejecutar dicha operación sin necesidad de conocer el contenido de la misma.
- Interpreter (Interprete**):** Dado un lenguaje, define una gramática para dicho lenguaje, así como las herramientas necesarias para interpretarlo.
- Iterator (Iterador): Permite realizar recorridos sobre objetos compuestos independientemente de estos.

[5] http://es.wikipedia.org/wiki/Patr%C3%B3n_de_dise%C3%B1o

- Mediator (Mediador): Define un objeto que coordine la comunicación entre objetos de distintas clases, pero que funcionan como un conjunto.
- Memento (Recuerdo): Permite volver a estados anteriores del sistema.
- Observer (Observador): Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambie de estado se notifiquen y actualicen automáticamente todos los objetos que dependen de él.
- State (Estado): Permite que un objeto modifique su comportamiento cada vez que cambie su estado interno.
- Strategy (Estrategia): Permite disponer de varios métodos para resolver un problema y elegir cual utilizar en tiempo de ejecución.
- TemplateMethod (Método Plantilla): Define en una operación el esqueleto de un algoritmo, delegando en las subclases algunos de sus pasos esto permite que las subclases redefinan ciertos pasos de un algoritmo sin cambiar su estructura.
- Visitor (Visitante): Permite definir nuevas operaciones sobre una jerarquía de clases sin modificar las clases sobre las que opera.

2.1.6 Ventajas de los Patrones de Diseño

- Son soluciones concretas:

✓ Un catálogo de patrones es un conjunto de recetas de diseño.

✓ Aunque se pueden clasificar, cada patrón es independiente del resto.

- Son soluciones técnicas:

✓ Dada una determinada situación, los patrones indican cómo resolverla mediante un D.O.O.

✓ Existen patrones específicos para un lenguaje determinado, y otros de carácter más general.

- Se aplican en situaciones muy comunes:

✓ Proceden de la experiencia.

✓ Han demostrado su utilidad para resolver problemas que aparecen frecuentemente en el D.O.O.

- Son soluciones simples:

✓ Indican cómo resolver un problema particular utilizando un pequeño número de clases relacionadas de forma determinada.

✓ No indican cómo diseñar un sistema completo, sino sólo aspectos puntuales del mismo.

- Facilitan la reutilización de las clases y del propio diseño:

✓ Los patrones favorecen la reutilización de clases ya existentes y la programación de clases reutilizables.

✓ La propia estructura del patrón es reutilizada cada vez que se aplica. [**6**]

2.1.7 Desventajas de los Patrones de Diseño

- El uso de un patrón no se refleja claramente en el código:

[6] http://es.scribd.com/doc/3930805/Patrones-de-Diseno

- ✓ A partir de la implementación es difícil determinar qué patrón de diseño se ha utilizado.
- ✓ No es posible hacer ingeniería inversa.
- Referencias a “self”:
- ✓ Muchos patrones utilizan la delegación de operaciones y esto provoca el conocido problema del ámbito local de los objetos (self).
- Es difícil reutilizar la implementación de un patrón:
- ✓ Las clases del patrón son roles genéricos, pero en la implementación aparecen clases concretas.
- Los patrones suponen cierta sobrecarga de trabajo a la hora de implementar:
- ✓ Se usan más clases de las estrictamente necesarias.
- ✓ A menudo un mensaje se resuelve mediante delegación de varios mensajes a otros objetos.

2.1.8 Objetivos de los Patrones de Diseño

Los patrones de diseño pretenden:

- Proporcionar catálogos de elementos reusables en el diseño de sistemas software.
- Separar la interfaz de usuario de la lógica de las aplicaciones,
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Estandarizar el modo en que se realiza el diseño.

- Facilitar el aprendizaje de las nuevas generaciones de diseñadores condensando conocimiento ya existente.

2.2 Patrón de Diseño Modelo Vista Controlador (MVC)

2.2.1 Descripción

Un problema muy común para los programadores es la reutilización del código que ya tienen hecho. A veces hay que resolver un problema parecido a algo que ya tenemos hecho, mejorar el aspecto de un programa o mejorar su algoritmo. Esta tarea se facilita mucho si a la hora de programar tenemos la precaución de separar el código en varias partes que sean susceptibles de ser reutilizadas sin modificaciones.

La arquitectura MVC separa la lógica de negocio (el modelo) y la presentación (la vista) por lo que se consigue un mantenimiento más sencillo de las aplicaciones. Si por ejemplo una misma aplicación debe ejecutarse tanto en un navegador estándar como en un navegador de un dispositivo móvil, solamente es necesario crear una vista nueva para cada dispositivo; manteniendo el controlador y el modelo original.

MVC es usado principalmente en aplicaciones que manejan gran cantidad de datos y transacciones complejas donde se requiere una mejor separación de conceptos para que el desarrollo esté estructurado de una mejor manera, facilitando la programación en diferentes capas de manera paralela e independiente.

El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email, etc.). El modelo se

encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación.

Una vista debe asegurarse de que su apariencia refleja el estado del modelo. Cada vez que cambian los datos del modelo, éste se encarga de avisar a las vistas que dependen de él. Como respuesta a dicha notificación, cada vista tiene la oportunidad de actualizarse a sí misma. Este enfoque permite asignar varias vistas a un modelo para ofrecer diferentes presentaciones.

También se pueden crear nuevas vistas de un modelo sin necesidad de volver a escribir éste.

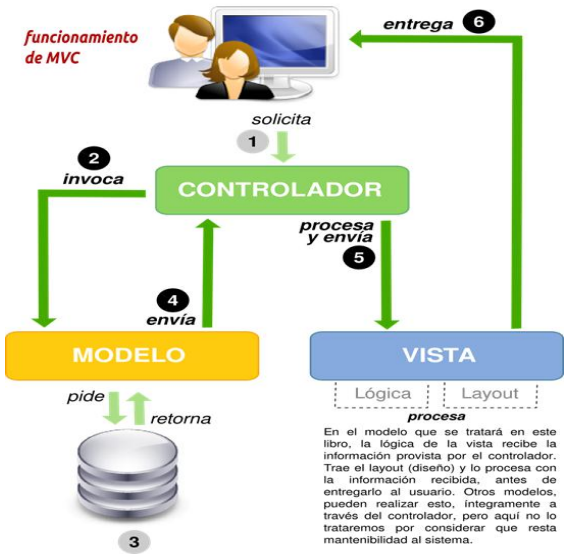


Figura II. 2 Funcionamiento del MVC

Fuente:https://www.google.com.ec/?gfe_rd=ctrl&ei=oygeU_BOotDyB_SKgPgL&gws_rd=cr#q=FUNCIonamiento+del+MVC+imagen

2.2.2 Historia del MVC

La arquitectura MVC (Model/View/Controller) fue introducida como parte de la versión Smalltalk-80 del lenguaje de programación Smalltalk. Fue diseñada para reducir el esfuerzo de programación necesario en la implementación de sistemas múltiples y sincronizados de los mismos datos.

Inicialmente diseñado para aplicaciones de escritorio cuya idea principal era separar la presentación del modelo de dominio.[7]

En aquel entonces la vista se encargaba del dibujado de mapas de bits, el controlador atendía acciones del mouse y teclado, y el modelo guardaba el estado de la ventana y la información mostrada en ella.

Cada vista estaba asociada con un controlador y un modelo en su creación. Cualquier modelo podía ser asociado débilmente a cualquier vista ya que un modelo podía relacionarse con varias vistas. El controlador no se encargaba de manejar la lógica del negocio en la aplicación, su propósito era manejar la interacción del mouse y el teclado, y actualizar la vista acorde, lo cual no era una tarea trivial.

2.2.3 Definición

Es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones.

---

[7] https://www.google.com.ec/?gfe_rd=ctrl&ei=oygeU_BOotDyB_SKgPgL&gws_rd=cr#q=FUNCIonamiento+del+MVC+imagen

Para ello MVC propone la construcción de tres componentes distintos que son el **modelo**, la **vista** y el **controlador**, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario.

Este patrón de diseño se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

2.2.3.1 Modelo

El modelo suele ser independiente de cómo se desea que el programa recoja los resultados o cómo los presente. Por ejemplo, si se quiere hacer un juego de ajedrez, todas las reglas del ajedrez son totalmente independientes de cómo se va a dibujar el tablero en 3D o plano, con figuras blancas y negras tradicionales o cualquier otro tipo de figura. Este código constituiría el modelo.

En el juego del ajedrez el modelo podría ser una clase (o conjunto de clases) que mantengan un registro piezas, que permita mover dichas piezas verificando que los movimientos son legales, que detecte los jaques, jaque mate, tablas, etc. De hecho, las metodologías orientadas a objeto nos introducen en este tipo de clases, a las que llaman **clases del negocio.**

Las principales características del **Modelo** son**:**

- Consiste en un conjunto de Clases y Objetos correspondientes al Modelo del Negocio para nuestra aplicación (estados y funcionalidad).

- Tiene un bajo acoplamiento con Vistas y Controladores.
- En el definen métodos para realizar consultas (informar el estado), comandos (modificar el estado), y mecanismos de notificación (para informar a los observadores/ vistas).

2.2.3.2 Vista

Es la representación del modelo en forma gráfica disponible para la interacción con el usuario. En el caso de una aplicación Web, la Vista es una página HTML con contenido dinámico sobre el cual el usuario puede realizar operaciones. En el ejemplo del ajedrez serían las posibles interfaces gráficas (3D, plano).

Esta parte del código es la vista. Consiste en la ventana que dibuja el tablero con las figuras de las piezas, que permiten arrastrar con el ratón una pieza para moverla, botones, lista visual con los movimientos realizados, etc.

Las principales características de las **Vistas** son:

- Administra la visualización y presentación de la información
- Observa al Modelo para actualizar los cambios
- Al definirse en el modelo una interfaz clara y estable, es fácil implementar múltiples Vistas para un mismo modelo
- Altamente dependiente del dispositivo y tecnología de visualización
- Muy dependiente del Modelo (debe conocerlo).

2.2.3.3 Controlador

Es la capa encargada de manejar y responder las solicitudes del usuario, procesando la información necesaria y modificando el Modelo en caso de ser necesario.

Es código que no tiene que ver con las ventanas visuales ni con las reglas de nuestro modelo. En el ejemplo formarían parte de esto el algoritmo para pensar las jugadas (el más complejo de todo el juego).

Las principales características del **Controlador** son:

- Responsable de definir el comportamiento de la aplicación
- Recibe los eventos del usuario y decide qué es lo que se debe hacer, mapeándolos en comandos (mensajes) hacia el Modelo
- Altamente dependiente de los dispositivos y mecanismos de interacción del usuario
- Muy dependiente del Modelo (debe conocerlo).

2.2.4 Interacción de los Componentes

Aunque se pueden encontrar diferentes implementaciones de **MVC**, el flujo de control que se sigue generalmente es el siguiente:

1. El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.)
2. El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.

3. El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.
4. El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se reflejan los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra). El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, se podría utilizar el patrón Observador para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. Este uso del patrón Observador no es posible en las aplicaciones Web puesto que las clases de la vista están desconectadas del modelo y del controlador. En general el controlador no pasa objetos de dominio (el modelo) a la vista aunque puede dar la orden a la vista para que se actualice.
5. La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente.

2.2.5 MVC y bases de datos

Muchos sistemas informáticos utilizan un Sistema de Gestión de Base de Datos para gestionar los datos que debe utilizar la aplicación; en líneas generales del **MVC** dicha gestión corresponde al modelo.

La unión entre capa de presentación y capa de negocio conocido en el paradigma de la Programación por capas representaría la integración entre la **Vista** y su correspondiente **Controlador** de eventos y acceso a datos, MVC no pretende discriminar entre capa de negocio y capa de presentación pero si pretende separar la capa visual gráfica de su correspondiente programación y acceso a datos, algo que mejora el desarrollo y mantenimiento de la Vista y el Controlador en paralelo, ya que ambos cumplen ciclos de vida muy distintos entre sí.

2.2.6 Uso en aplicaciones Web

Aunque originalmente MVC fue desarrollado para aplicaciones de escritorio, ha sido ampliamente adaptado como arquitectura para diseñar e implementar aplicaciones web en los principales lenguajes de programación.

Se han desarrollado multitud de frameworks, comerciales y no comerciales, que implementan este patrón; estos frameworks se diferencian básicamente en la interpretación de como las funciones MVC se dividen entre cliente y servidor.

Los primeros frameworks MVC para desarrollo web planteaban un enfoque de cliente ligero en el que casi todas las funciones, tanto de la vista, el modelo y el controlador recaían en el servidor.

En este enfoque, el cliente manda la petición de cualquier hiperenlace o formulario al controlador y después recibe de la vista una página completa y actualizada (u otro documento); tanto el modelo como el controlador (y buena parte de la vista) están completamente alojados en el servidor.

Como las tecnologías web han madurado, ahora existen frameworks como JavaScriptMVC, Backbone o jQuery que permiten que ciertos componentes MVC se ejecuten parcial o totalmente en el cliente.

2.3 Patrón de Diseño Modelo Vista Presentador (MVP)

2.3.1 Descripción

El concepto de este patrón es bastante sencillo. Por un lado se tiene la vista, que se encarga de mostrar la información al usuario y de interactuar con él para hacer ciertas operaciones. Por otro lado, se tiene el modelo que, ignorante de cómo la información es mostrada al usuario, realiza toda la lógica de las aplicaciones usando las entidades del dominio. Y por último se tiene al presentador que es el que “presenta” a ambos actores sin que haya ningún tipo de dependencia entre ellos.

El propósito del presentador tal y como se ha definido en este patrón no establece de manera clara el grado de control que este puede hacer de la vista.

Dependiendo de este grado de control podemos encontrarnos variaciones en la manera de ver MVP. De hecho, algunas de ellas, dejan de ser MVP para convertirse en MVC, por lo cual debemos tener cuidado.

El modelo es normalmente los datos de la aplicación y la lógica para recuperar y conservar los datos. A menudo, se trata de un modelo de dominio que puede basarse en una base de datos o los resultados de los servicios web.

En algunos casos, que el modelo de dominio corresponde perfectamente a lo que se ve en la pantalla, pero en otros casos ha de ser adaptada, agregados o extendido a ser utilizable.

La vista es típicamente un control de usuario o formulario que combina varios en una interfaz de usuario. El usuario puede interactuar con los controles en la vista, pero cuando se necesita cierta lógica para iniciarse, la vista este delegado al presentador.

El presentador tiene toda la lógica de la vista y es responsable de sincronizar el modelo y la vista. Cuando la vista notifica el presentador que el usuario ha hecho algo (por ejemplo, hacer clic en un botón), el presentador a continuación, actualizar el modelo y sincronizar los cambios entre el modelo y la vista.

Una cosa importante a mencionar es que el presentador no comunica directamente a la vista. En lugar, se comunica a través de una interfaz. De esta forma, el presentador y el modelo se pueden probar en aislamiento.

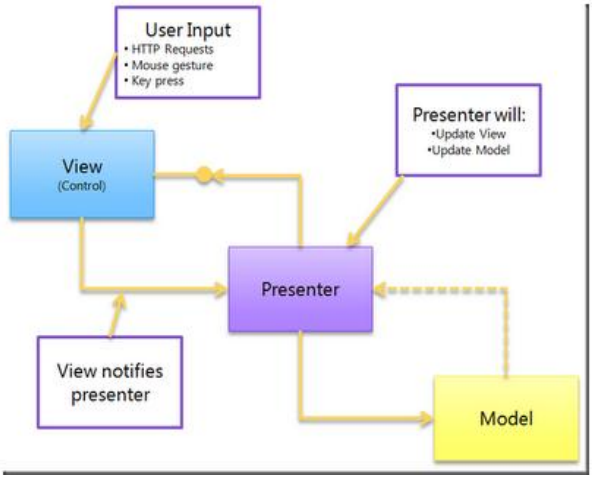


Figura II. 3 Funcionamiento del MVP

Fuente:http://gatuneg.wikispaces.com/Modelo+vista+Presentador

Así que después de la aparición de entornos de programación visuales y la introducción de controles que encapsulan la visualización y el código de interacción con el usuario, se convirtió en la necesidad de crear una clase de controlador independiente menos.

Pero la gente todavía encuentra la necesidad de una presentación de forma separada, sólo que esta vez con un mayor nivel de abstracción.

Porque resultó si crea un formulario que se compone de varios controles y también contiene la lógica de la interfaz de usuario y los datos, el patrón modelo vista presentador describe una manera de separar los elementos visuales (controles) de la lógica (lo que sucede cuando se interactúa con los controles) y los datos (los datos que se muestran en la vista). [[8]]

[8]http://gatuneg.wikispaces.com/Modelo+vista+Presentador

2.3.2 Historia del MVP

La arquitectura MVP (Model/View/Presenter) nació a partir del modelo MVC (Model View Controller), pero decirles que MVC fue usado por Smalltalk a partir de los años 80 y de acá MVP es una mutación si podríamos llamarlo así que nace a partir de MVC.

Así que después de la aparición de entornos de programación visuales y la introducción de controles que encapsulan la visualización y el código de interacción con el usuario, se convirtió en la necesidad de crear una clase de controlador independiente menos.

Pero la gente todavía encuentra la necesidad de una presentación de forma separada, sólo que esta vez con un mayor nivel de abstracción.

Porque resultó si crea un formulario que se compone de varios controles y también contiene la lógica de la interfaz de usuario y los datos, patrón modelo vista presentador describe una manera de separar los elementos visuales (controles) de la lógica (lo que sucede cuando se interactúa con los controles) y los datos (los datos que se muestran en la vista).

2.3.3 Definición

Es un derivado del patrón Modelo Vista Controlador (MVC) patrón de software, también se utiliza principalmente para la construcción de interfaces de usuario.

MVP es una interfaz de usuario patrón de diseño de ingeniería para facilitar las pruebas unitarias automatizadas y mejorar la separación de las preocupaciones en la lógica de presentación.

La clave del patrón MVP es una estricta regulación de la interacción entre la vista y el controlador, aunque en el patrón MVP, al controlador se le conoce como presentador.

2.3.3.1 Modelo

Es la representación en objetos del negocio y será manipulado desde y hacia la vista pero no en forma directa sino que será el "Presenter" el responsable de hacerlo y de enlazar a ambos, al "Model" y la "View".

Algunas veces el modelo está dado directamente por la estructura de base de datos y otras está dado por objetos del tipo entidad que representan al negocio y que serán persistidos en la base de datos, además es donde se lleva a cabo toda la lógica de negocio.

Las principales características del **Modelo** son:

- Es toda la lógica de negocio.
- El modelo realiza toda la operación lógica de las aplicaciones usando el dominio.

2.3.3.2 Vista

Son las interfaces de usuario propiamente dichas y contra las que el interacciona. En lo que es la tecnología .Net ejemplos concretos son las páginas Web con extensión .aspx o formularios de Windows con extensión .cs.

Cada vista es una clase que implementara una interfaz, esta interfaz posee propiedades que modelan los datos que serán persistidos y recuperados hacia y desde el modelo por el "Presenter". Un ejemplo concreto de vista es la clase "WebForm" que implementa la interfaz "IView".

Otro punto a destacar es que la vista será totalmente pasiva esto significa que todo el comportamiento básicamente métodos, serán implementados por la clase "Presenter" y quitados de la vista.

Esto permite que las pruebas se centren en el "Presenter" y la "View" minimizando los riesgos de los problemas sobre la vista.

Las principales características de las **Vistas** son:

- Son las ventanas y controles que forman la interfaz de usuario de la aplicación en sí.
- La vista, que se encarga de mostrar la información al usuario y de interactuar con él para hacer ciertas operaciones.
- La vista no conoce el modelo.
- La vista es testearles puesto que está basada en un contrato.

2.3.3.3 Presentador

El "Presenter" tendrá la responsabilidad de implementar el comportamiento que antes estaba implementado en la propia View. Como un ejemplo concreto, para facilitar la comprensión, podemos dar el que sigue:

Supongamos el método de atención al evento OnClick de un botón de un formulario el cual realiza la persistencia de datos desde el formulario a la base de datos, este método será candidato a ser migrado como un método de la clase "Presenter", luego este método será invocado desde la vista.

Además escucha los eventos que se producen en la vista y ejecuta las acciones necesarias a través del modelo. Además puede tener acceso a las vistas a través de las interfaces que la vista debe implementar.

Las principales características del **Presentador** son:

- Son los eventos que se producen en la vista y también ejecuta acciones a través del modelo.
- El presentador es el que se encarga de mostrar la vista y el modelo.
- El presentador es independiente de la tecnología de interfaz de usuario.
- El presentador son testeables puesto que está basada en un contrato.

2.3.4 Problemas que soluciona el patrón MVP

El MVP es la solución a los siguientes dos grandes problemas que se presentan con gran frecuencia en el desarrollo del software:

- **Realizar pruebas automáticas sobre la interfaz de usuario (unit test*):***

  El patrón nos permite realizar pruebas automáticas (unit test) utilizando un objeto que simula la interfaz de usuario (View), para esto la misma debe implementar una interfaz programática que será luego la misma que implementara nuestro objeto simulador (Mock en inglés) que utilizaremos para realizar los unit test.

- **Independizar a la aplicación del tipo de interfaz de usuario elegida**:

  El patrón nos permite independizar la aplicación de la interfaz de usuario, así podremos lograr que la misma funcione tanto como una aplicación de escritorio

(Desktop), como una aplicación Web o como una aplicación móvil con un mínimo de impacto, también podemos lograr que nuestra aplicación de escritorio tenga varias vistas de un mismo modelo por ejemplo mostrar uno u otro front-end o skin.

2.3.5 Un ejemplo ingenuo

A continuación daremos un ejemplo no productivo ni sacado del mundo real, sino desarrollado a los fines de poder aumentar la comprensión del patrón MVP, el ejemplo que daremos toma como partida una aplicación desarrollada de forma tradicional (Active View) dejando unidas la presentación y el comportamiento en una misma clase e impidiendo realizar test de unidad sobre la vista, luego la continuaremos evolucionándola hasta llegar a la misma aplicación, desde el punto de vista funcional, pero desarrollada cumpliendo el patrón MVP (Pasive View) convirtiendo a la View solo en una mera cáscara sin funcionalidad ni comportamiento alguno, algo así como una piel que la aplicación expone y que puede intercambiar por otra.

- **Funcionalidad de la aplicación de ejemplo*:***

La aplicación solo tiene la responsabilidad de convertir el texto que ingresa el usuario a mayúsculas. [9]

[9] http://www.monografias.com/trabajos93/mvp-model-view-presenter/mvp-model-view-presenter.shtml

Captura de pantalla que muestra la interfaz de usuario encargada de convertir el texto ingresado a letras mayúsculas.

Captura de pantalla del proyecto aun sin implementar el MVP.

Este es el código fuente inicial, observemos que el comportamiento se encuentra en la misma clase del formulario, es una (Active View). La única forma de probar en forma automática es mediante un software que interacciona sobre la interfaz de usuario o realizar una prueba no automática mediante un beta tester o persona que actúa, vía interfaz de usuario, como un experto y es capas de verificar el correcto funcionamiento de la misma. El otro problema es que si queremos cambia de aplicación de escritorio a Web, debemos trasladar el comportamiento de una a otra ya que este es parte privada de la vista, cuando estrictamente solo deberíamos crear la nueva vista o piel.

```
public partial class ConvertidorView : Form
{
   public ConvertidorView()
   {
      InitializeComponent();
   }

   private void buttonConvertir_Click(object sender, EventArgs e)
   {
      string mayuscula = textBoxDato.Text;
      mayuscula = mayuscula.ToUpper();
      textBoxDato.Text = mayuscula;
   }
}
```

Figura II. 4 Funcionalidad de la Aplicación de ejemplo

Fuente: http://www.monografias.com/trabajos93/mvp-model-view-presenter/mvp-model-view-presenter.shtml

CAPÍTULO III

ANÁLISIS COMPARATIVO ENTRE LOS PATRONES DE DISEÑO MODELO VISTA CONTROLADOR (MVC) Y MODELO VISTA PRESENTADOR (MVP)

3.1 Introducción

En este capítulo se hace un análisis comparativo de los dos patrones de diseño Modelo Vista Controlador y Modelo Vista Presentador utilizando varios criterios o parámetros los cuales utilizarán varios indicadores que nos ayudarán a determinar cuál de los dos patrones es el más óptimo para el desarrollo de aplicaciones web y así permita brindar una solución factible y fiable en la realización del sistema Nómina de Empleados y Rol de Pagos de la "Distribuidora SORIA C.A"

3.2 Análisis Comparativo

3.2.1 Fase 1: Patrones de Diseño a Evaluar

En base a la variedad de patrones de diseño existentes para el desarrollo de sistemas software, se considera analizar los siguientes patrones de diseño:

- Modelo Vista Controlador(MVC)
- Modelo Vista Presentador(MVP)

3.2.2 Fase 2: Variables a Evaluar

A continuación se describen los criterios o parámetros con sus respectivos indicadores para el análisis de los dos patrones de diseño durante la etapa de planificación de la Metodología MSF, así como la especificación de cada uno de los mismos para comprender de mejor manera el análisis:

TABLA III. I Parámetros de Comparación

Criterios / Parámetros de los Patrones de Diseño	Indicadores
P 1. Compatibilidad	**I 1.1** Sistemas Operativo **I 1.2** Navegadores Web
P 2. Recursos	**I 2.1** Uso de memoria RAM **I 2.2** Uso de CPU **I 2.3** Tiempo de Respuesta
P 3. Desarrollo	**I 3.1** Tiempo de Desarrollo **I 3.2** Líneas de Código Escritas

Fuente: Autora

A continuación se indica la descripción de cada parámetro y sus indicadores, necesario para comprender el análisis:

P 1. Compatibilidad.- Característica que tiene un Patrón de diseño que indica que el mismo puede estar disponible para otros sistemas o usuarios, puede ser una pieza completa de software y ser transportada y ejecutada en ambientes diferentes y con mínimas modificaciones, además de ser compatible con patrones de su entorno de ejecución como: sistemas operativos navegadores web, como otros patrones de diseño, etc.

I 1.1 Sistemas Operativos.- Es un programa o conjunto de programas, un sistema informático que gestiona los recursos de hardware y provee servicios a programas de aplicación y herramientas de gestión ejecutándose en modo privilegio respecto a los restantes y anteriores próximos y viceversa.

I 1.2 Navegadores Web.- La función básica es permitir visualizar el contenido de los patrones de diseño, realizar actividades en él, poder enlazarse con otros componentes, imprimir, enviar y recibir información de otros servicios web. Y que estén disponibles para el uso del usuario.

P 2. Recursos.- Es el factor que determina las características mínimas del equipo para iniciar a trabajar con un patrón de diseño y de esta manera no tener inconvenientes.

I 2.1 Uso de memoria.- Se refiere a la cantidad de memoria ocupada por la aplicación web creada por medio del patrón de diseño al momento de realizar un proceso.

I 2.2 Uso de CPU.- Se refiere a la cantidad de CPU ocupada por la aplicación web creada por medio del patrón de diseño al momento de realizar un proceso.

I 2.3 Tiempo de Respuesta.- Se refiere al tiempo transcurrido desde que se envía una petición hasta que se recibe una respuesta.

P 3. Desarrollo.- Este parámetro tiene la finalidad de conocer las facilidades que brinda cada patrón de diseño para el desarrollo de la aplicación web, así como los componentes que estos poseen para mejorar la interfaz del usuario en cuanto a accesibilidad.

I 3.1 Tiempo de Desarrollo.- Se refiere al tiempo (**horas**) que se demore en realizar cada prototipo con cada patrón de diseño.

I 3.2 Líneas de Código escritas.- Este indicador permitirá evaluar la cantidad de líneas de código que se llevan a cabo para realizar un determinado proceso con MVC y MVP, con la finalidad de establecer el patrón de diseño que menos líneas de código genera.

3.3 Establecimiento de las Escalas de Valoración para cada uno de los Parámetros (indicadores).

Para obtener resultados cuantitativos y cualitativos que permitan una selección adecuada de uno de los patrones de diseños analizados para desarrollar aplicaciones web empresariales, se realizará la calificación de cada uno de los indicadores de comparación.

La calificación **Regular** será para aquel patrón de diseño que no cumplan con los indicadores establecidos dentro del parámetro que se está evaluando, **Bueno** recibirá el patrón de diseño que en parte cumpla con los indicadores establecidos dentro del parámetro citado en la comparación, mientras que la calificación **Excelente** será para el patrón de diseño que cumpla con el indicador definido y se acople al cien por ciento.

Donde cada una de las variables van a ser evaluadas sobre el máximo que es **3** que equivale al 100%, y cada uno de los ítems de la interpretación incluye la siguiente nomenclatura (x, y)/w en donde cada letra significa lo siguiente.

La calificación definitiva del patrón de diseño en base a cada parámetro de comparación se obtiene sumando los puntajes obtenidos del análisis, utilizando las siguientes formulas:

$$P_{MVP} = \sum(y),$$

$$P_{MVC} = \sum(x),$$

$$P_c = \sum(w)$$

Donde el significado de cada variable utilizada se describe en la Tabla III. II

TABLA III. II Significado de Nomenclatura

X	Representa el puntaje que obtiene el patrón de diseño MVC
Y	Representa el puntaje que obtiene el patrón de diseño MVP
W	Representa la base del puntaje sobre la cual se está calificando el parámetro

Fuente: Autora

Aplicando la Fórmula anterior se tiene:

$$P_{MVC} = \sum(x),$$

Calificación de MVC: $MVC = (P_{MVC} / P_c) * 100\%$

$$P_{MVP} = \sum(y),$$

Calificación de MVP: $MVP = (P_{MVP} / P_c) * 100\%$

En donde:

TABLA III. III Significado de Variables

P_{MVP}	Puntaje acumulado por MVP en el parámetro analizado
P_{MVC}	Puntaje acumulado por MVC en el parámetro analizado
P_c	Puntaje total sobre el que se califica el parámetro analizado
mvc	Porcentaje de la calificación total que obtuvo MVC en el parámetro
mvp	Porcentaje de la calificación total que obtuvo MVP en el parámetro

Fuente: Autora

3.4 Parámetro 1: Compatibilidad

3.4.1 Determinación de las Variables

a. Sistemas Operativos
b. Navegadores Web

- **Valorizaciones**

A continuación se describe las consideraciones a evaluar en cada uno de las variables ya determinadas en el parámetro Compatibilidad.

a. Variable Sistemas Operativos

Este criterio se refiere a la adaptabilidad del patrón de diseño con diferentes frameworks (JSF, Swing, .Net), los mismos que pueden funcionar en diferentes sistemas operativos sean estos libres o propietarios.

b. Variable Navegadores Web

Este criterio se refiere a la adaptabilidad del patrón de diseño con los diferentes navegadores web existentes como Google Chrome, Firefox Mozilla.

Utilizando el procedimiento detallado en el punto **3.3** se obtuvieron los siguientes datos como se muestra en la Tabla III. IV

TABLA III. IV Resumen variables parámetro Compatibilidad

VARIABLE	MVC	Valor	MVP	Valor
Sistemas Operativos	Excelente	3	Excelente	3
Navegadores Web	Excelente	3	Excelente	3

Fuente: Autora

3.4.2 Interpretación de la valorización

- En cuanto a sistemas operativos, para los patrones de diseño MVC y MVP se pueden encontrar múltiples frameworks que pueden adaptarse a este patrón ya sea en .Net o Java entre ellos tenemos: MVC# Framework, Claymore, JFace, GWT, Swing, los cuales pueden ser utilizados en diferentes sistemas operativos y diferentes entornos de programación.
- En cuanto a lo que respecta a los navegadores web como Google Chrome, Firefox Mozilla, para los patrones de diseño MVC y MVP poseen una excelente compatibilidad, puesto que cada uno de ellos en la vista implementan código HTML, independientemente de la tecnología (como son lenguajes, framework, librerías, etc.) usada para construir la interfaz.
- **Calificación**

$$P_c = \sum(w) = 3+3 = 6$$

$$P_{MVC} = \sum(x) = I1 + I2 = P1$$

$$P_{MVC} = \sum(x) = 3+3 = 6$$

$$MVC : (P_{MVC} / P_c) * 100\% = (6/6) * 100\% = 100\%$$

$$P_c = \sum(w) = 3 + 3 = 6$$

$$P_{MVP} = \sum(y) = I1 + I2 = P1$$
$$P_{MVP} = \sum(y) = 3 + 3 = 6$$

$$MVP : (P_{MVP} / P_c) * 100\% = (6/6) * 100\% = 100\%$$

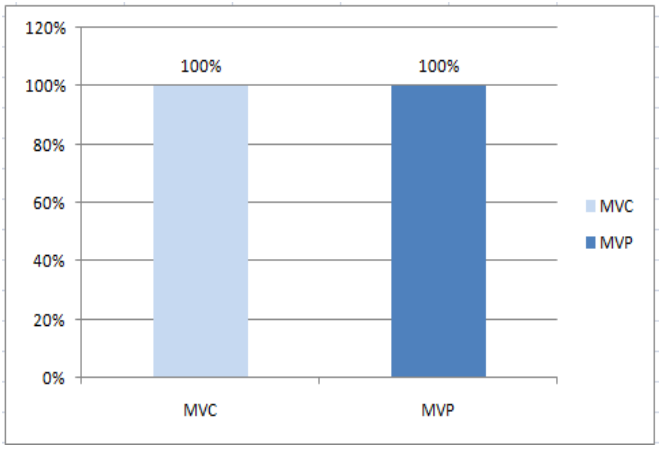


FIGURA III. 5 Comparación estadística del Parámetro Compatibilidad

Fuente: Autora

3.5 Parámetro 2: Recursos

3.5.1 Determinación de las Variables

a. Uso de CPU

b. Uso de Memoria RAM

c. Tiempo de Respuesta

- **Valorizaciones**

A continuación se describe las consideraciones a evaluar en cada uno de las variables ya determinadas en el parámetro Recursos.

a. Uso de CPU

Se refiere a la cantidad de CPU ocupada por la aplicación web creada, utilizando el patrón de diseño al momento de crear un registro y listarlo junto con todos los ingresados anteriormente.

Tabla III. V Escala de Calificación para el Indicador Uso de CPU

Numeración	Porcentaje	Equivalencia
1	> 80 %g	REGULAR
2	>= 75 % y <= 80 %	BUENO
3	< 75 %	EXCELENTE

Fuente: Autora

Herramientas para realizar las Pruebas

Para realizar las pruebas y recopilación de información, para este indicador se utilizó la herramienta que viene incorporada en Visual Studio 2012 en la opción Analizar.

Recopilación de Información

Posterior al desarrollo de los aplicativos en las siguientes Figura III. 6 y Figura III. 7, se presentan los resultados obtenidos de las pruebas realizadas.

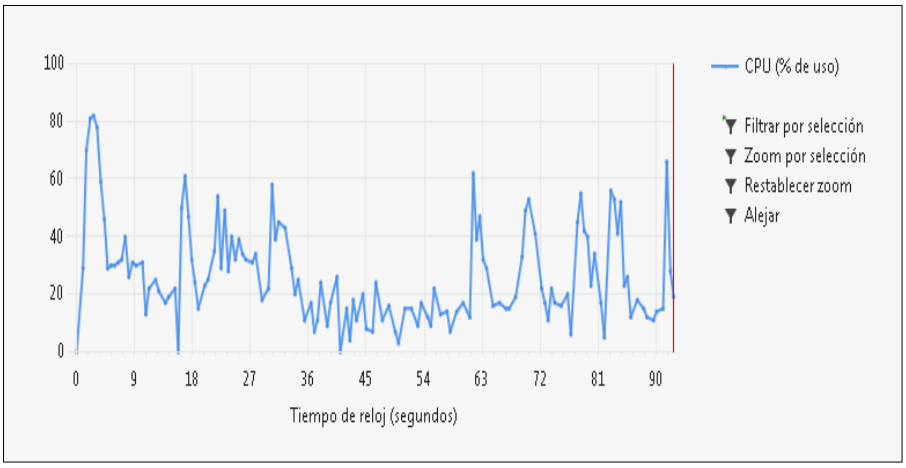


FIGURA III. 6 Test de CPU de MVC con Visual Studio 2012

Fuente: Autora

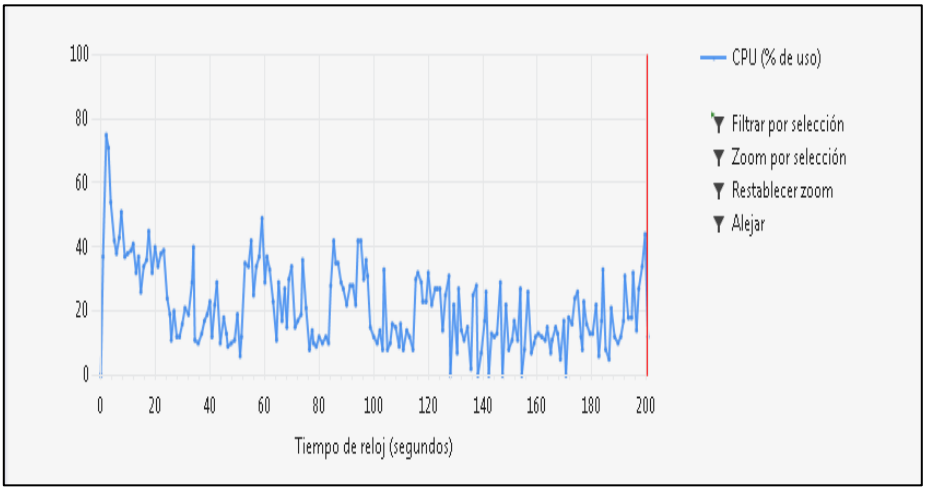


FIGURA III. 7 Test de CPU de MVP con Visual Studio 2012

Fuente: Autora

b. Uso de Memoria

Se refiere a la cantidad de memoria ocupada por la aplicación web creada, por medio del patrón de diseño al momento de crear un registro y listarlo junto con todos los ingresados anteriormente.

Tabla III. VI Escala de Calificación para el Indicador Uso de Memoria RAM

Numeración	Porcentaje	Equivalencia
1	>70%	REGULAR
2	>= 40% y <= 70%	BUENO
3	<40%	EXCELENTE

Fuente: Autora

Escenarios de Prueba

Para poder valorizar e interpretar cada variable este parámetro se realizó una prueba mediante la ejecución de las dos aplicaciones anteriores, con el fin de comparar las características de rendimiento, en estas pruebas se va a determinar el uso de memoria y uso de procesador al momento de ejecutar la inserción y listado de registros. Con este propósito se crearon las aplicaciones en los siguientes escenarios.

TABLA III. VII Escenario 1. MVC

Ítem	Descripción
HARDWARE	
Procesador	Intel Core i5 2.5 Ghz
Memoria RAM	4 GB
Disco Duro	600 GB
Cache	2.5 g Ghz
SOFTWARE	
Sistema Operativo	Windows 7 Home Premium
IDE	Netbeans
Base de Datos	PostgreSQL
Patrón de diseño	MVC

Fuente: Autora

TABLA III. VIII Escenario 2. MVP

Ítem	Descripción
HARDWARE	
Procesador	Intel Core i5 2.5 Ghz
Memoria RAM	4 GB
Disco Duro	600 GB
Cache	2.5 g Ghz
SOFTWARE	
Sistema Operativo	Windows 7 Home Premium
IDE	Netbeans
Base de Datos	PostgreSQL
Patrón de diseño	MVP

Fuente: Autora

Herramientas para realizar las Pruebas

Para realizar las pruebas y recopilación de información, para este indicador se utilizó la herramienta que viene incorporada en Windows que es el Administrador de Tareas en la opción Analizar.

Recopilación de Información

Posterior al desarrollo de los aplicativos en las siguientes Figura III. 8 y Figura III. 9, se presentan los resultados obtenidos de las pruebas realizadas.

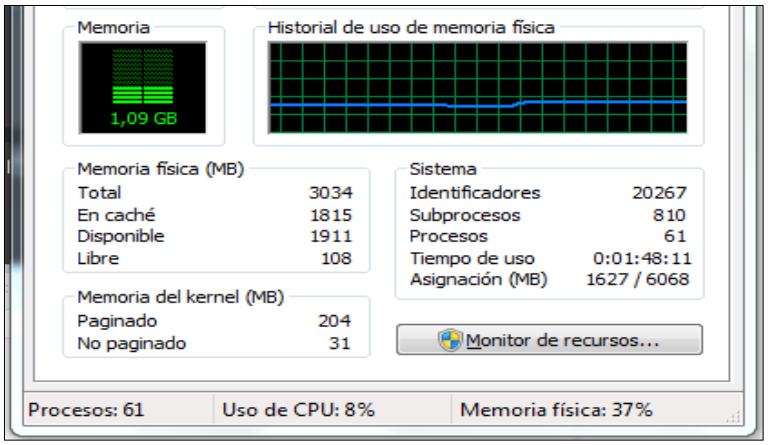


FIGURA III. 8 Test de Memoria de MVP con ADMINISTRADOR DE TAREAS DE WINDOWS

Fuente: Autora

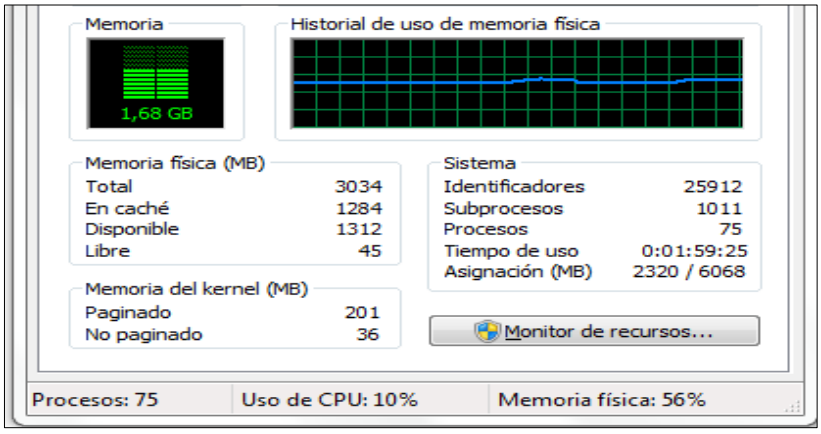


FIGURA III. 9 Test de Memoria de MVC con ADMINISTRADOR DE TAREAS DE WINDOWS

Fuente: Autora

c. Tiempo de Respuesta

Se refiere al tiempo empleado por la aplicación al momento de recibir la petición de Insertar un nuevo registro hasta que devuelva un listado con el registro nuevo agregado a los anteriores.

NOTA: Para este indicador se ha tomado como referencia 500 ms que equivale a medio segundo que es el 100%. No se tomó como base 1 seg (1000 ms) porque es demasiado

grande esa cantidad y nuestros valores nunca iban a ingresar en ese rango. (Porque solo se van a comparar los ms ya que en segundos, siempre coinciden ambos patrones de diseño)

TABLA III. IX Escala de Calificación para el Indicador Tiempo de Respuesta

Numeración	Porcentaje	Equivalencia
1	> 300 ms	REGULAR
2	>= 80 ms y <= 300 ms	BUENO
3	< 80 ms	EXCELENTE

Fuente: Autora

Herramientas de recopilación de Datos

Para realizar las pruebas y recopilación de información, para este indicador, se la ha realizado mediante la herramienta **FIDDLER WEB DEBUGGER**, la cual es una herramienta Libre y está disponible en el siguiente link: http://www.descargargratis.com/fiddler/descargar.

Recopilación de Información

Posterior al desarrollo de los aplicativos en las siguientes Figuras III. 10, hasta la Figura III. 15, se presentan los resultados obtenidos de las pruebas realizadas con la herramienta antes mencionada.

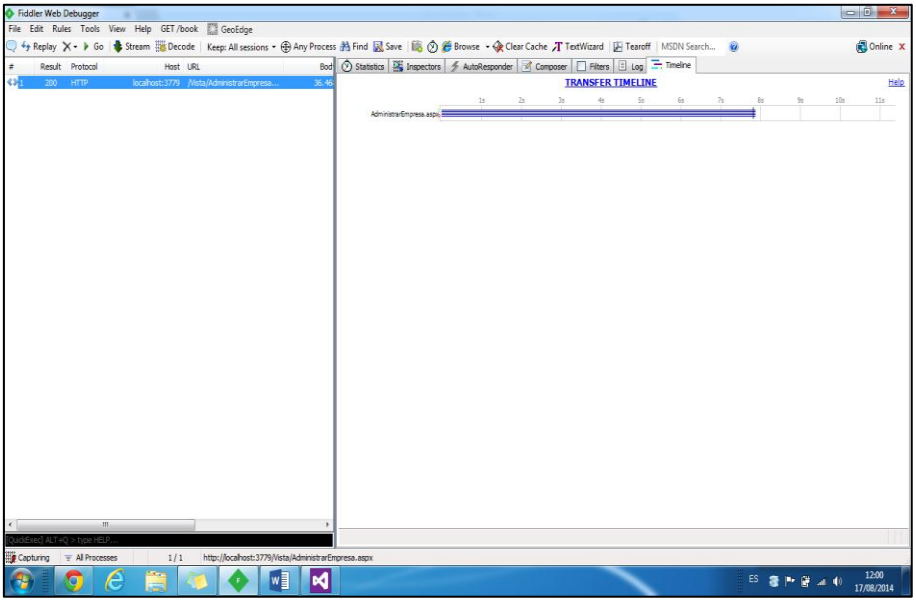


Figura III. 10 Test de Tiempo de Respuesta de MVP con FIDDLER WEB DEBUGGER
Tiempo en Cargar la Página
Fuente: Autora

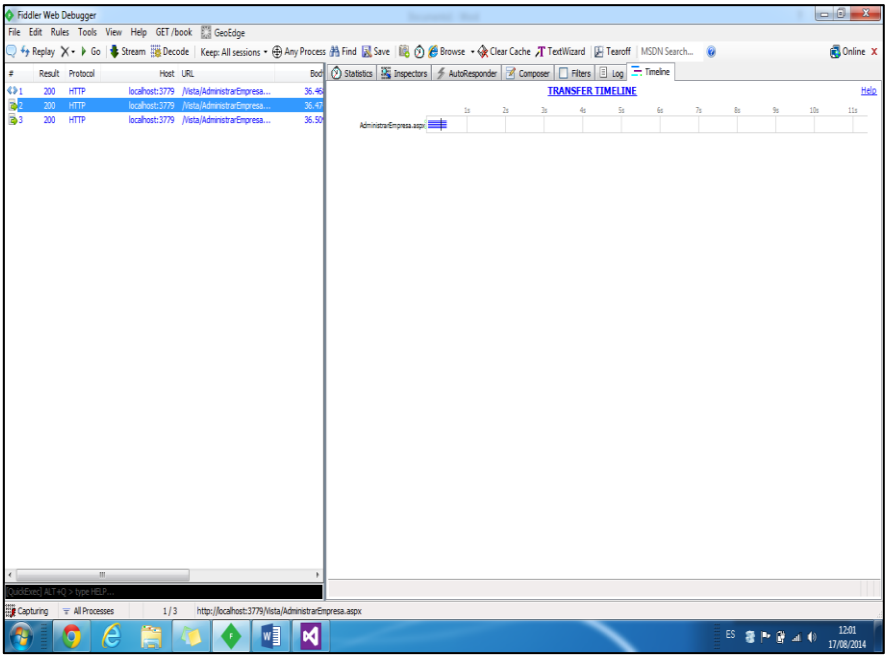


Figura III. 11 Test de Tiempo de Respuesta de MVP con FIDDLER WEB DEBUGGER
Tiempo en enviar la Petición
Fuente: Autora

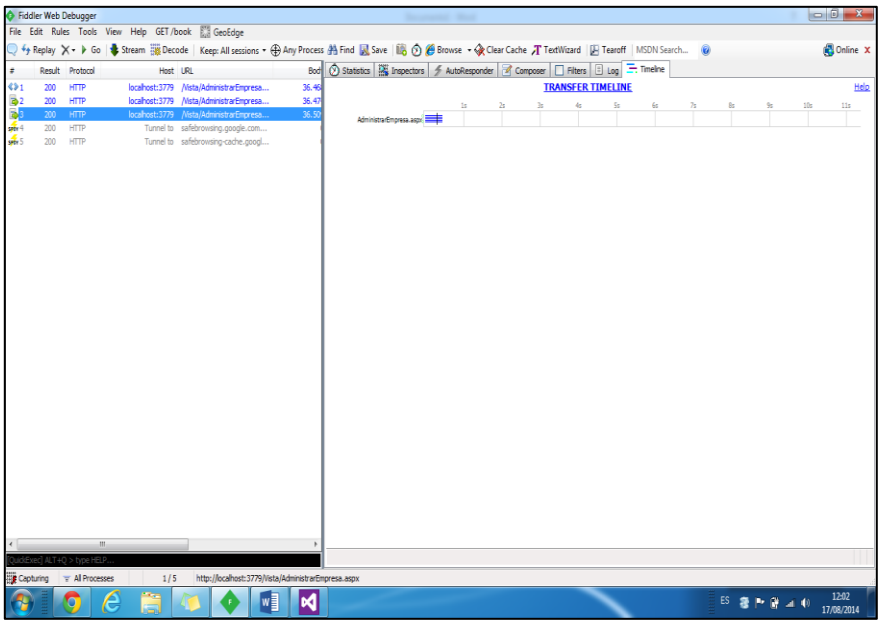


Figura III. 12 Test de Tiempo de Respuesta de MVP con FIDDLER WEB DEBUGGER

Tiempo en devolver la Respuesta
Fuente: Autora

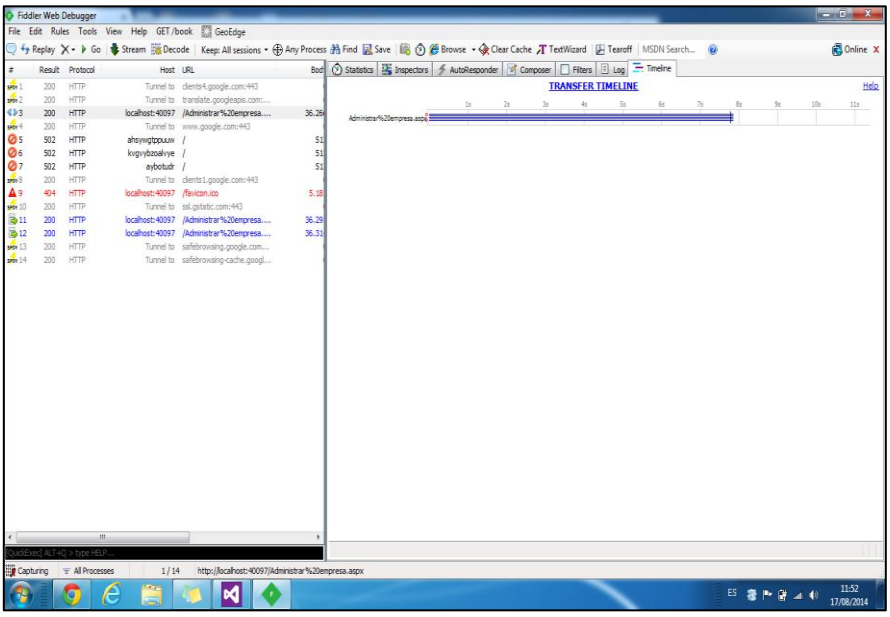


Figura III. 13 Test de Tiempo de Respuesta de MVC con FIDDLER WEB DEBUGGER
Tiempo en Cargar la Página
Fuente: Autora

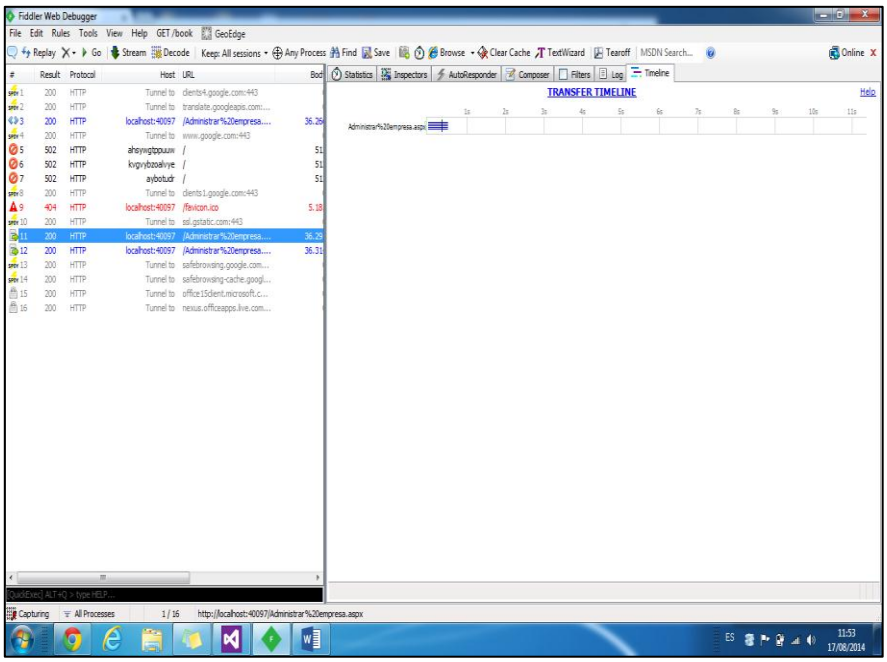


Figura III. 14 Test de Tiempo de Respuesta de MVC con FIDDLER WEB DEBUGGER
Tiempo en enviar la Petición
Fuente: Autora

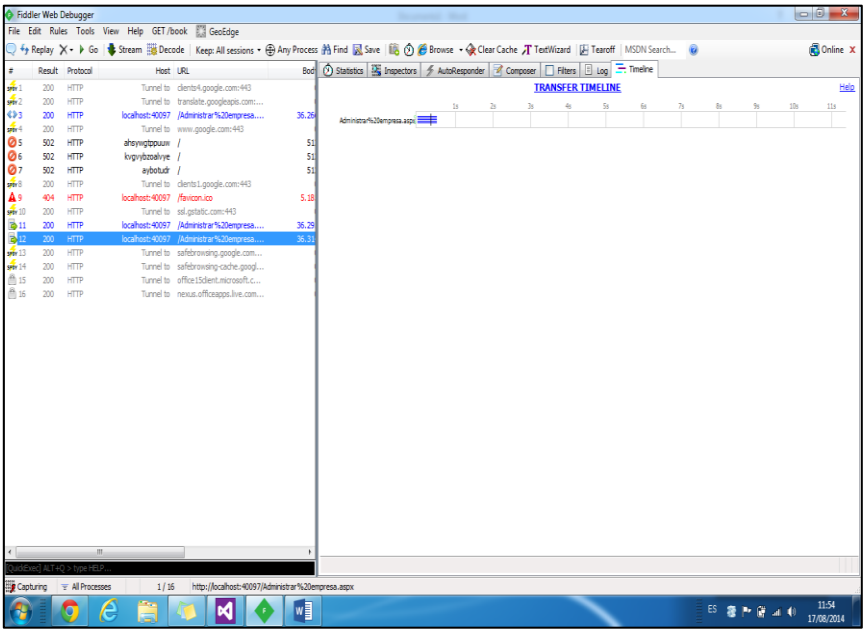


Figura III. 15 Test de Tiempo de Respuesta de MVC con FIDDLER WEB DEBUGGER
Tiempo en Devolver la Respuesta
Fuente: Autora

Utilizando el procedimiento detallado en el punto **3.3** se obtuvieron los siguientes datos como se muestra en la Tabla III. X

TABLA III. X Resumen variables parámetro Recursos

	MVC			MVP		
VARIABLE	**Resultado**	**Valor**	**Equivalencia**	**Resultado**	**Valor**	**Equivalencia**
Uso de memoria RAM	56 %	2	Bueno	37 %	3	Excelente
Uso de CPU	80 %	2	Bueno	70 %	3	Excelente
Tiempo de Respuesta	101 ms	2	Bueno	7 ms	3	Excelente

Fuente: Autora

3.5.2 Interpretación de la valorización

- En cuanto al uso del CPU se comprobó que MVP utiliza un porcentaje de 70% este recurso, con la ventaja que se reduce el tiempo de uso de este recurso siendo este un 90 seg, caso contrario a MVC que utiliza un porcentaje 80% de este recurso pero por una mayor cantidad de tiempo, siendo este 200 seg.
- Al analizar las gráficas de Memoria de los dos patrones de diseño se observa que la cantidad de memoria requerida por MVP es de 37 %, a la ocupada por MVC que es 56 %.
- Al analizar las gráficas del Tiempo de Respuesta de las páginas web creadas con los dos patrones de diseño en estudio mediante la herramienta **FIDDLER WEB DEBUGGER**, se observa que la aplicación web creada con el MVP emite una respuesta en menos tiempo siendo este 8 seg con 7 ms, que la realizada empleando el MVC obteniendo un tiempo de respuesta de 8 seg con 101 ms.

- **Calificación**

$$P_c = \sum(w) = 3+3+3 = 9$$

$$P_{MVC} = \sum(x) = I1+I2+I3 = P2$$

$$P_{MVC} = \sum(x) = 2+2+2 = 6$$

$$MVC : (P_{MVC} / P_c) * 100\% = (6/9) * 100\% = 66{,}66\%$$

$$P_c = \sum(w) = 3+3+3 = 9$$

$$P_{MVP} = \sum(y) = I1+I2+I3 = P2$$

$$P_{MVP} = \sum(y) = 3+3+3 = 9$$

$$MVP : (P_{MVP} / P_c) * 100\% = (9/9) * 100\% = 100\%$$

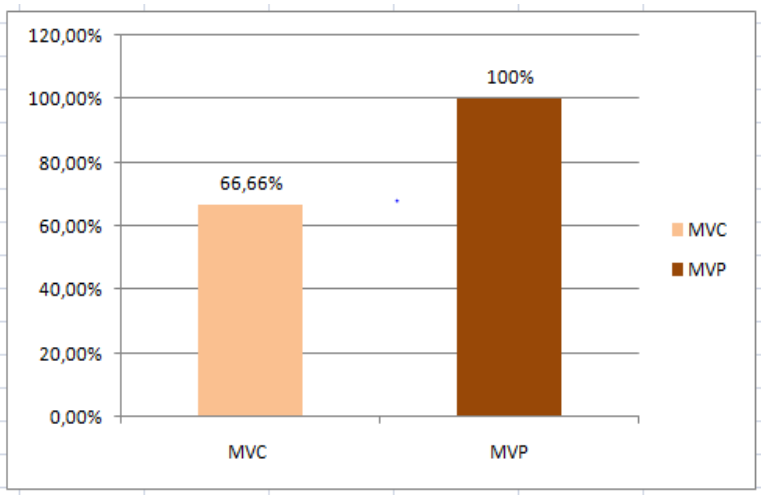


Figura III. 16 Comparación Estadística del Parámetro Recursos

Fuente: Autora

3.6 Parámetro 3: Desarrollo

3.6.1 Determinación de las Variables

a. Tiempo de Desarrollo

b. Líneas de Código Escritas

- **Valorizaciones**

A continuación se describe las consideraciones a evaluar en cada uno de las variables ya determinadas en el parámetro Desarrollo.

a. Tiempo de Desarrollo

Este criterio se refiere al tiempo (**horas**) que necesita el programador en crear la estructura de diseño, de tal manera que se demore el menor tiempo posible en construir la aplicación para su respectiva ejecución.

NOTA: Para este indicador se ha tomado como referencia 100 horas (8 horas/día) laborables, que equivale al 100%, porque son solo pequeños prototipos de prueba.

TABLA III. XI Escala de Calificación para el Indicador Tiempo de Desarrollo

Numeración	Porcentaje	Equivalencia
1	> 70 horas	REGULAR
2	>= 25horas y <= 70 horas	BUENO
3	< 25 horas	EXCELENTE

Fuente: Autora

b. Líneas de Código Escritas

Este indicador se refiere a la cantidad de líneas de código que se llevan a cabo para realizar un determinado proceso con MVC y MVP, con la finalidad de establecer el patrón de diseño que menos líneas de código genera.

NOTA: Para este indicador se ha tomado como referencia 500 LCE, que equivale al 100%, porque son solo prototipos para saber cuál patrón de diseño es mejor.

TABLA III. XII Escala de Calificación para el Indicador Líneas de Código Escritas

Numeración	Porcentaje	Equivalencia
1	> 350 LCE	REGULAR
2	>= 200 LCE y <= 350 LCE	BUENO
3	< 200 LCE	EXCELENTE

Fuente: Autora

Utilizando el procedimiento detallado en el punto **3.3** se obtuvieron los siguientes datos como se muestra en la Tabla III. XIII

TABLA III. XIII Resumen variables parámetro Desarrollo

	MVC			MVP		
VARIABLE	**Resultado**	**Valor**	**Equivalencia**	**Resultado**	**Valor**	**Equivalencia**
Tiempo de Desarrollo	25 h	2	Bueno	20 h	3	Excelente
Líneas de Código Escritas	238	2	Bueno	191	3	Excelente

Fuente: Autora

3.6.2 Interpretación de la valorización

- En cuanto al tiempo de desarrollo, para los patrones de diseño, en el caso de MVP presenta cierta ventaja al separar claramente el código que se utiliza en la vista del modelo y a la vez este puede ser reutilizado por medio de la interfaz y del

presentador lo que reduce tiempo a 20 hora en comparación a MVC que se demoró 25 horas.

- En cuanto a lo que respecta a líneas de código, para los patrones de diseño MVC y MVP se puede concluir que en MVP se generan menos líneas de código en la vista, con un valor de 191 LCE, que en MVC que obtuvo 238 LCE lo cual se debe a que en el código de interfaz de usuario solamente existen componentes gráficos con sus respectivos SET y GET, caso contrario a MVC que en el código de la interfaz incluye también elementos u objetos de la lógica de negocio.

- **Calificación**

$$P_c = \sum(w) = 3+3 = 6$$

$$P_{MVC} = \sum(x) = I1 + I2 = P3$$
$$P_{MVC} = \sum(x) = 2+2 = 4$$

$$MVC : (P_{MVC} / P_c) * 100\% = (4/6) * 100\% = 66{,}66\%$$

$$P_c = \sum(w) = 3+3 = 6$$

$$P_{MVP} = \sum(y) = I1 + I2 = P3$$

$$P_{MVP} = \sum(y) = 3+3 = 6$$

$$MVP : (P_{MVP} / P_c) * 100\% = (6/6) * 100\% = 100\%$$

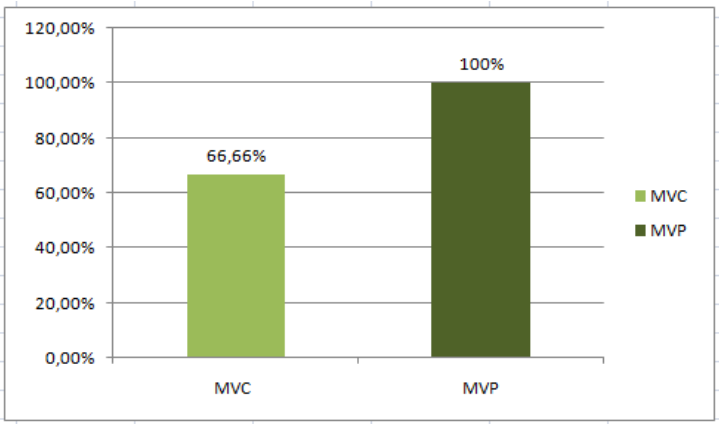


FIGURA III. 17 Comparación estadística del Parámetro Desarrollo

Fuente: Autora

3.7 Análisis de los Resultados Obtenidos

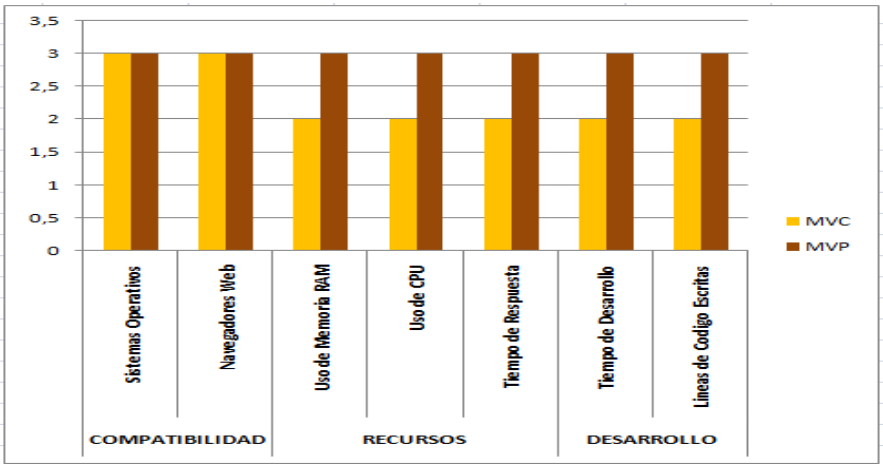


FIGURA III. 18 Resultado final por Parámetro

Fuente: Autora

$$PT_{MVC} = 6+6+4 = 16$$

$$PT_{MVP} = 6+9+6 = 21$$

$$(\%MVC) = (16/21)*100 = 76{,}19\%$$ **Equivalente a BUENO**

$$(\%MVP) = (21/21)*100 = 100\%$$ **Equivalente a EXCELENTE**

Resultados que podemos representar de la siguiente manera:

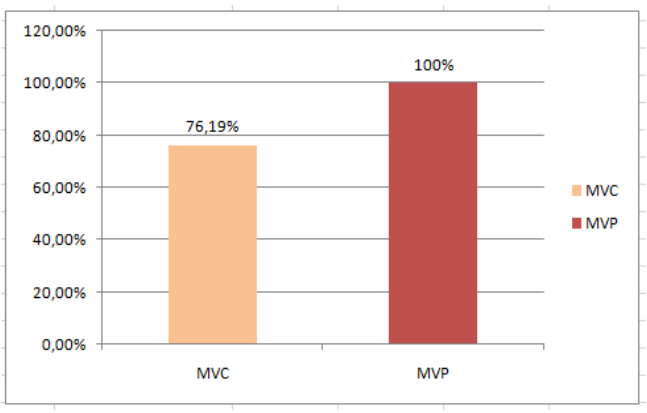


FIGURA III. 19 Diagrama General de Resultados

Fuente: Autora

Después de haber realizado el análisis comparativo correspondiente de los respectivos parámetros con sus variables, como se puede observar en la Figura III.19 se ha determinado que para desarrollar una aplicación web empresarial el patrón de diseño MVP, es el que tiene una máxima calificación de 100% correspondiente a excelente por lo que se determina que es el patrón de diseño más adecuado para desarrollar la aplicación web empresarial; por otro lado MVC obtuvo una calificación de 76,19% que equivale a Bueno.

3.8 Comprobación de la Hipótesis

3.8.1 Análisis de Resultados de la Hipótesis

Se ha realizado el análisis para demostrar la hipótesis que señala que:

“El desarrollo de una aplicación web bajo el Patrón de Diseño Modelo Vista Presentador seleccionado mejorará la productividad de desarrollo”, de la tesis titulada: “Análisis comparativo de la productividad entre los patrones de diseño Modelo Vista Controlador

(MVC) y Modelo Vista Presentador (MVP) aplicado al desarrollo del sistema Nómina de Empleados y Rol de Pagos de la Distribuidora SORIA C.A"

Se determinó que la estadística inferencial y específicamente el CHI Cuadrado es el método más adecuado para este estudio, puesto que se aplica a pequeñas muestras (ya que la muestra es apenas de veinte tiempos de respuesta(valores) en diferentes horas, para diferentes procesos) y no existe un sistema desarrollado con el patrón de diseño MVP.

3.8.2 Estadística Inferencial

La inferencia estadística o estadística inferencial es una parte de la Estadística que comprende los métodos y procedimientos para deducir propiedades (hacer inferencias) de una población o cantidad, a partir de una pequeña parte de la misma (muestra).

La bondad de estas deducciones se miden en términos probabilísticos, es decir, toda inferencia se acompaña de su probabilidad de acierto.

3.8.3 Análisis de los Parámetros a ser evaluados para comprobar la Productividad.

Para poder realizar la demostración de la hipótesis de este trabajo investigativo se va a probar que la utilización del Patrón de Diseño MVP mejorará la PRODUCTIVIDAD, en el desarrollo de una aplicación software web.

Se tomará en consideración los parámetros elegidos para la comparación, que inciden en el desarrollo de la aplicación web contribuyendo en mejorar la productividad, estos son: **Compatibilidad**, **Recursos y Desarrollo,** cada uno con sus respectivos indicadores.

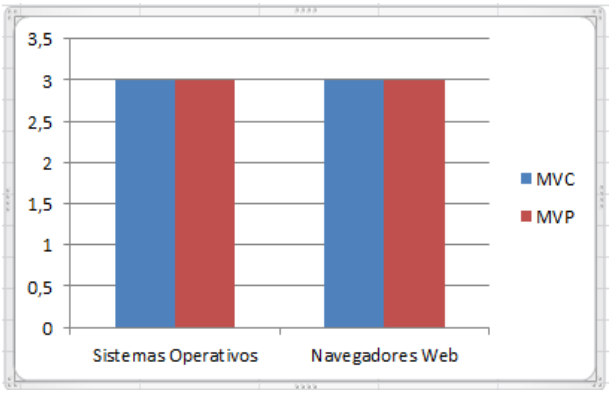


FIGURA III.20 Resultado Final del Parámetro Compatibilidad MVC vs MVP

Fuente: Autora

Como podemos observar en la Figura III.20, en cuanto a Sistemas Operativos para los patrones de diseño MVC y MVP, no hay ninguna dificultad se pueden encontrar múltiples frameworks que pueden adaptarse a estos patrones ya sea en .Net o Java entre ellos tenemos: MVC# Framework, Claymore, JFace, GWT, Swing, los cuales pueden ser utilizados en diferentes sistemas operativos y diferentes entornos de programación.

En cuanto a lo que respecta a los Navegadores Web como Google Chrome, Firefox Mozilla, para los patrones de diseño MVC y MVP poseen una excelente compatibilidad, puesto que cada uno de ellos en la vista implementan código HTML, independientemente de la tecnología (como son lenguajes, framework, librerías, etc.) usada para construir la interfaz.

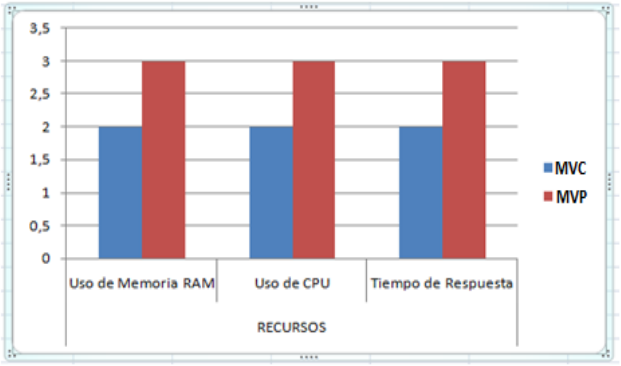


FIGURA III.21 Resultado Final del Parámetro Recursos MVC vs MVP

Fuente: Autora

Como podemos observar en la Figura III.21, en cuanto al Uso de Memoria RAM, de los dos patrones de diseño se observa que la cantidad de memoria requerida por MVP es menor a la ocupada por MVC, En cuanto al uso del CPU se comprobó que MVP utiliza mayor porcentaje este recurso, con la ventaja que se reduce el tiempo de uso de este recurso siendo este un 90 seg, caso contrario a MVC que utiliza menor cantidad de este recurso pero por una mayor cantidad de tiempo, siendo este 200 seg. En cuanto al Tiempo de Respuesta de las páginas web creadas con los dos patrones de diseño en estudio mediante la herramienta **FIDDLER WEB DEBUGGER**, se observa que la aplicación web creada con el MVP emite una respuesta en menos tiempo siendo este 8 seg con 7 ms, que la realizada empleando el MVC obteniendo un tiempo de respuesta de 8 seg con 101 ms

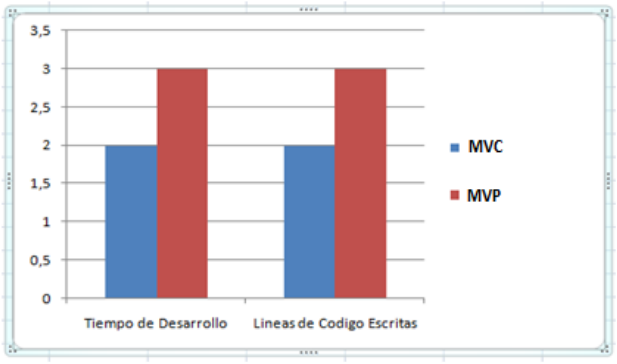


FIGURA III. 202 Resultado Final del Parámetro Desarrollo MVC vs MVP

Fuente: Autora

Como podemos observar en la Figura III.22, en cuanto al Tiempo de Desarrollo, para los patrones de diseño, en el caso de MVP presenta cierta ventaja al separar claramente el código que se utiliza en la vista del modelo y a la vez este puede ser reutilizado por medio de la interfaz y del presentador lo que reduce tiempo. En cuanto a lo que respecta a Líneas de Código, para los patrones de diseño MVC y MVP se puede concluir que en MVP se generan menos líneas de código en la vista que en MVC debido a que en el código de interfaz de usuario solamente existen componentes gráficos con sus respectivos SET y GET, caso contrario a MVC que en el código de la interfaz incluye también elementos u objetos de la lógica de negocio.

Mediante las Figuras expuestas anteriormente, se puede verificar que la productividad de una aplicación web con el patrón de diseño MVP es mejor puesto que mediante este patrón se utiliza de mejor manera los recursos mejorando inevitablemente el tiempo de respuesta y disminuyendo el tiempo que invierte el programador a la hora de desarrollar la aplicación web.

De los resultados obtenidos, se puede concluir que luego de haber realizado el análisis comparativo y realizado todos los cálculos correspondientes para la demostración de la hipótesis, y analizando variable por variable del estudio efectuado, se puede concluir que la utilización del patrón de diseño MVP mejorará la productividad en el desarrollo de aplicaciones Web empresariales.

Observación: Para comprobar la hipótesis de esta investigación con una Prueba Estadística, en este caso CHI cuadrado, que es la más indicada por las características del proyecto, y así comprobar efectivamente que la adopción del patrón MVP, mejorará la productividad frente al patrón MVC, en el desarrollo de una aplicación web.

Se ha seleccionado parámetros cada uno con sus respectivos indicadores, como se demostró en la comparación ya citada. Pero para probar que la hipótesis de la investigación es afirmativa, se ha tomado como referencia un solo indicador (TIEMPO DE RESPUESTA, del parámetro RECURSOS) ya que este es el que más predomina a la hora de ver cuán productivo es un software en el momento de su ejecución.

3.8.4 Selección y determinación de la Muestra

Se realizó la selección de un grupo de datos obtenidos de ambos prototipos MVC y MVP (tiempos de respuestas) para diferentes procesos (Insertar, Actualizar y Eliminar) en diferentes horas del día, con la finalidad de obtener información necesaria para la comprobación de la hipótesis.

3.8.5 Clasificación y organización de los datos

Las respuestas emitidas por ambos prototipos son las siguientes:

TABLA III. XIV Valores de los Resultados

HORA	DATOS	MVC			MVP		
		INSERTAR	ACTUALIZAR	ELIMINAR	INSERTAR	ACTUALIZAR	ELIMINAR
20:00pm	**1**	520 ms	1,136	789 ms	512 ms	999 ms	498 ms
21:00pm	**2**	516 ms	1,176	1,651	493 ms	959 ms	444 ms
22:00pm	**3**	575 ms	967 ms	963 ms	613 ms	1,069	517 ms
23:00pm	**4**	600 ms	992 ms	980 ms	587 ms	1,104	607 ms
01:00am	**5**	510 ms	1,142	1,350	490 ms	975 ms	459 ms
05:00am	**6**	634 ms	896 ms	879 ms	612 ms	960 ms	798 ms
06:00am	**7**	595 ms	820 ms	953 ms	621 ms	1,110	837 ms
07:00am	**8**	595 ms	820 ms	953 ms	621 ms	1,110	837 ms
08:00am	**9**	600 ms	827 ms	963 ms	488 ms	1,161	845 ms
09:00am	**10**	575 ms	968 ms	963 ms	530 ms	1,069	745 ms
10:00am	**11**	580 ms	978 ms	998 ms	525 ms	1,065	740 ms
11:00am	**12**	515 ms	1,165	789 ms	498 ms	957 ms	450 ms
12:00pm	**13**	530 ms	1,153	897 ms	510 ms	1,069	963 ms
13:00pm	**14**	575 ms	968 ms	963 ms	530 ms	1,069	745 ms
14:00pm	**15**	510 ms	1,175	1,550	498 ms	957 ms	450 ms
15:00pm	**16**	479 ms	934 ms	1,222	690 ms	1,053	453 ms
16:00pm	**17**	600 ms	827 ms	963 ms	488 ms	1,161	845 ms
17:00pm	**18**	510 ms	1,175	1,550	498 ms	957 ms	450 ms
18:00pm	**19**	495 ms	1,153	1,455	495 ms	923 ms	460 ms

19:00pm	**20**	510 ms	1,153	455 ms	445 ms	923 ms	480 ms
TOTAL		**11, 024**	**20, 425**	**21, 286**	**10,744**	**20, 650**	**12,623**

Fuente: Autora

3.8.6 Análisis de Datos

Se muestran los datos de las respuestas emitidas por la aplicación (tanto para el Patrón MVC y MVP) al haber realizado los diferentes procesos en diferentes horas del día, tabulados en una matriz dada por filas (i) y columnas (j), para poder aplicar Chi-cuadrado con mayor facilidad.

TABLA III. XV Tabla de Contingencia

P. D / TIEMPO R.	MVC	MVP	TOTAL
Insertar	11,024	10,744	**21, 768**
Actualizar	20,425	20,650	**41, 075**
Eliminar	21,286	12,623	**33, 909**
TOTAL	**52, 735**	**44, 017**	**96, 752**

Fuente: Autora

Una vez obtenida la matriz de datos tabulados se procedió a calcular los valores esperados, a partir de los datos abstraídos, para esto se debe multiplicar los respectivos marginales y dividir por el gran total.

$$V.E = (\sum fila\,(i) * \sum columna\,(j)\ ) \ / \ total$$

$V.E_{1\,1}$ = (52, 735 * 21, 768) / 96,752 = 11, 864

$V.E_{1\,2}$ = (52, 735 * 41, 075) / 96,752 = 22, 388

$V.E_{2\,1}$ = (52, 735 * 33, 909) / 96,752 = 18,482

$V.E_{2\,2}$ = (44, 017 * 21, 768) / 96,752 = 9, 903

$V.E_{3\,1}$ = (44, 017 * 41, 075) / 96,752 = 18, 686

$V.E_{3\,2}$ = (44, 017 * 33, 909) / 96,752 = 15,426

3.8.7 Valores Esperados

TABLA III. XVI Valores Esperados

P. D / TIEMPO R.	MVC	MVP	TOTAL
Insertar	11, 864	22, 388	**34, 252**
Actualizar	18, 482	9, 903	**28, 385**
Eliminar	18, 686	15, 426	**34, 112**
TOTAL	**49, 032**	**47, 717**	**96, 749**

Fuente: Autora

A través de Chi –cuadrado se probara de forma afirmativa o negativa que la distribución de las frecuencias abstraídas difiere significativamente en relación a la distribución de las frecuencias que deberíamos esperar.

$$X^2 = \sum((O-E)^2 / E)$$

En donde:

O = Frecuencia o valores Observados

E = Frecuencia o valores Esperados

TABLA III. XVII Valores Observados y Valores Esperados

O	E	O − E	$(O - E)^2$	$(O - E)^2$/ E
11, 024	11, 864	- 0, 84	0, 705	0, 059
10, 744	22, 388	- 11, 644	135, 582	6, 056

20, 425	18, 482	1, 943	3, 775	0, 204
20, 650	9, 903	10, 747	115, 498	11, 662
21, 286	18, 686	2, 60	6, 76	0, 361
12, 623	15, 426	- 2, 803	7, 856	0, 509
			$x^2 =$	**18, 851**

Fuente: Autora

Para afirmar o negar la hipótesis se debe comparar el valor obtenido (y), con el Chi - cuadrado crítico de la tabla de valores crítico, los parámetros que se deben tomar en cuenta son los grados de libertad (GL) y el nivel de significación (P); el primero se define como él (número de columnas - 1) X (número de filas - 1), en este caso es (2 - 1) X (3 - 1) = 2 para ambos casos; el nivel de significación también conocido como nivel de confianza se refiere a la probabilidad de que los resultados observados se deban al azar, este valor es fijado por el investigador usualmente es de 5 % o 10 % (es decir 0,05 o 0,1).

De acuerdo a la tabla de distribución estadística CHI cuadrado, con un nivel de significancia de 0,05 y 2 grados de libertad genera un valor X_T^2 **= 5, 991**

TABLA III. XVIII Distribución Estadística CHI cuadrado

	0,995	0,990	0,975	0,950	0,900	0,750	0,500	0,250	0,100	**0,050**	0,025	0,010	0,005
1	0,000	0,000	0,001	0,004	0,016	0,102	0,455	1,323	2,706	3,841	5,024	6,635	7,879
2	0,010	0,020	0,051	0,103	0,211	0,575	1,386	2,773	4,605	5,991	7,378	9,210	10,597
3	0,072	0,115	0,216	0,352	0,584	1,213	2,366	4,108	6,251	7,815	9,348	11,345	12,838
4	0,207	0,297	0,484	0,711	1,064	1,923	3,357	5,385	7,779	9,488	11,143	13,277	14,860
5	0,412	0,554	0,831	1,145	1,610	2,675	4,351	6,626	9,236	11,070	12,833	15,086	16,750
6	0,676	0,872	1,237	1,635	2,204	3,455	5,348	7,841	10,645	12,592	14,449	16,812	18,548
7	0,989	1,239	1,690	2,167	2,833	4,255	6,346	9,037	12,017	14,067	16,013	18,475	20,278
8	1,344	1,646	2,180	2,733	3,490	5,071	7,344	10,219	13,362	15,507	17,535	20,090	21,955
9	1,735	2,088	2,700	3,325	4,168	5,899	8,343	11,389	14,684	16,919	19,023	21,666	23,589
10	2,156	2,558	3,247	3,940	4,865	6,737	9,342	12,549	15,987	18,307	20,483	23,209	25,188

Fuente: http://www.monografias.com/trabajos97/prueba-hipotesis-chi-cuadrado-empleando-excel-y-winstats/prueba-hipotesis-chi-cuadrado-empleando-excel-y-winstats.shtml

3.8.9 Procedimiento de Comprobación

Para la comprobación de la hipótesis, en este caso como el factor que incide es el Tiempo de Respuesta, el procedimiento es el siguiente:

$\boldsymbol{H_0}$: $T.R_{MVP} < = T.R_{MVC}$

$\boldsymbol{H_1}$: $T.R_{MVP} > T.R_{MVC}$

Nivel de Significancia

Par el siguiente estudio estadístico se tomará como nivel de significancia 0,05 para obtener un nivel de confianza aceptable.

CRITERIOS DE DECISIÓN:

$$\boldsymbol{X_T^2 = 5,991 \quad y \quad X_C^2 = 18,851}$$

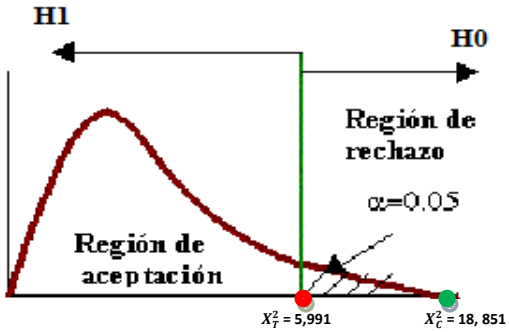
Si $X_C^2 > X_T^2$ Rechazo H_0 y Acepto H_1

CASO CONTRARIO:

Si $X_C^2 < X_T^2$ Acepto H_0 y Rechazo H_1

3.8.10 Representación gráfica de los Resultados

Para aceptar la hipótesis H_0 $X_C^2 < X_T^2$ en caso contrario se rechaza la H_0 y se acepta la hipótesis alterna H_1.



Determinación de la Hipótesis
Fuente: Autora

Se concluye que se rechaza la hipótesis $\boldsymbol{H_0}$ porque cae en la zona de rechazo de la $\boldsymbol{H_0}$ por tal razón se acepta la hipótesis $\boldsymbol{H_1}$ alterna de la investigación.

3.8.11 Validación de la Hipótesis

El desarrollo de una aplicación web utilizando el patrón de diseño como es MVP tiene muchas ventajas sobre todo es más productivo en el consumo de los recursos y desarrollo en relación al patrón de diseño MVC.

3.8.12 Conclusión del análisis de la Hipótesis

De los resultados obtenidos, se encontró que el Chi cuadrado calculado es mayor que el Chi cuadrado critico dado por defecto, por lo que se puede confirmar la validez de la hipótesis, es decir, que el resultado es efectivo por la naturaleza de los datos obtenidos.

Los resultados emitidos por ambos prototipos afirman que el empleo de MVP desarrollado reducen el consumo de los recursos y optimiza el tiempo de desarrollo de un proyecto web; lo que implica que, el desarrollo de una aplicación web utilizando el patrón de diseño

MVP(Modelo Visa Presentador) mejora la productividad en el desarrollo de un software web.

CAPÍTULO IV

DESARROLLO DEL SISTEMA NÓMINA DE EMPLEADOS Y ROL DE PAGOS DE LA "DISTRIBUIDORA SORIA C.A"

Introducción

En el presente capítulo se pone en ejecución la parte aplicativa del sistema, los módulos de la solución web para la nómina de empleados y rol de pagos para la Distribuidora SORIA C.A, usando los patrones de diseño MVC Y MVP.

4.1 Microsoft Solution Framework

4.1.1 Definición

[10]Microsoft Solution Framework es un marco de trabajo de referencia para construir e implantar sistemas empresariales distribuidos basados en herramientas y tecnologías de Microsoft. MSF comprende un conjunto de modelos, conceptos y guías que contribuyen a

[10]http://wiki.monagas.udo.edu.ve/index.php/Microsoft_Solution_Framework_(MSF)

alinear los objetivos de negocio y tecnológicos, reducir los costos de la utilización de nuevas tecnologías, y asegurar el éxito en la implantación de las tecnologías Microsoft. MSF es el resultado de las experiencias de diferentes áreas de Microsoft con proyectos exitosos.

El MSF es un modelo diseñado específicamente para crear productos de buena calidad, donde para cumplir el objetivo prima la comunicación tanto entre el equipo de desarrollo como entre ellos y los clientes.

4.1.2 Fases

El MSF está compuesto por las siguientes fases:

- Visión
- Planeación
- Desarrollo
- Estabilización
- Instalación
- Soporte

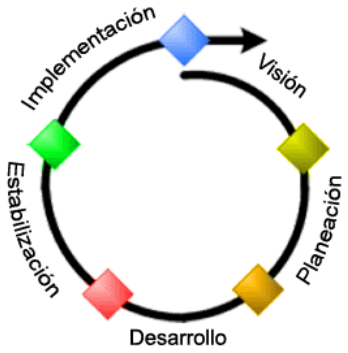


FIGURA IV. 213 Fases de MSF

Fuente: http://audiemangt.blogspot.com/2010/05/metodologia-agil-msf-microsoft-solution.html

4.1.3 Ventajas

- MSF ayuda a implantar soluciones en base a la tecnología
- Crear una disciplina de análisis de riesgos que ayuda y evoluciona con el proyecto
- Vinculación con el cliente como también orientado al trabajo en equipo
- Tiene facilidad de soporte y mantenimiento
- Es adaptable, se puede utilizar para proyectos de cualquier magnitud
- El modelo tiene facilidad de manejo por ser una empresa conocida
- Aplica mucho e incentiva al trabajo en equipo y a la colaboración
- Cuenta con plantillas que ayudan para el desarrollo de documentos

4.1.4 Desventajas

- Al estar basado en tecnologías Microsoft, trata de obligar a usar herramientas de ellos mismo
- Solicita demasiada documentación en sus fases
- Si el análisis de riesgos se hace exhaustivo puede retardar el proyecto
- Los precios de licencias, capacitación y soporte son caros

4.1.5 Fase de Visión

En esta fase se debe realizar un estudio de lo que pretendemos en el futuro que haga nuestra aplicación para ello debemos realizar un documento de estrategia y alcance donde debe quedar pactada la necesidad de funcionalidad y servicio que se debe contar en la solución.

Para asegurar el éxito del proyecto es importante tener en cuenta el análisis de riesgos y plan de contingencia.

4.1.6 Definición del Problema

La falta de automatización de ciertos procesos para llevar un control adecuado de los datos de los empleados, puestos de trabajo, horarios de trabajo, egresos e ingresos y generación de Rol de Pagos, han ocasionado múltiples problemas al momento de obtener informes como son la Nómina de Empleados y la emisión de Roles de Pagos.

- La falta de información a los empleados en cuanto a la forma de pago acerca de su salario dificulta la comunicación con la parte administrativa.
- La falta de automatización de los proceso en una institución dificulta un control adecuado de la información dentro de la misma.
- La falta de un control adecuado de empleados en una institución dificulta su respectivo registro laboral.

4.1.7 Visión del Proyecto

Desarrollar una aplicación para la Distribuidora Soria C.A que permita llevar un control adecuado de todos sus empleados y emisiones de Roles de Pago.

Proveer información detallada al empleado acerca de sus salarios.

Generar informes que permitan conocer la información de los empleados de la empresa.

- **Módulo de Administración**

Mediante este módulo se creara información sobre usuarios y roles mediante los cuales se restringirá el acceso a los diferentes empleados existentes en la empresa.

- **Módulo de Tipos**

  Módulo Tipo de Sangre

  Módulo Tipo Grado

  Módulo Tipo Contrato

  Módulo Tipo Gasto

  Módulo Estado Civil

  Módulo Nivel Titulo

- **Módulo de Aportes**

  Módulo de Fondos de Reserva

  Módulo de Aporte Patronal

  Módulo de Aporte Personal

- **Módulo de Empleados**

  Módulo de Información Personal

  Módulo de Formación Académica

  Módulo de Capacitación

  Módulo de Experiencia Laboral

  Módulo Grupo Familiar

- **Módulo de Egresos**

  Módulo de Descuento

  Módulo de Multas

  Módulo de Atrasos

  Módulo de Faltas

Módulo de Anticipos

Módulo de Préstamos

- **Módulo de Ingresos**

  Módulo de Horas extra

- **Módulo de Pagos**

  Módulo Décimo Tercero

  Módulo Décimo Cuarto

  Módulo Rol de Pagos

4.2 Perfil de Usuario

Al contar con la información requerida en la Distribuidora Soria C.A se realizó el análisis para los usuarios que va a contener la aplicación para un mejor funcionamiento: Usuario Administrador, Gerente Administrativo, Contadora General, Auxiliar de Contabilidad.

TABLA IV. XIX Perfil de Usuario

Tipos	Acceso	Características
Usuario Administrador	Web	Encargado de crear usuarios y de interactuar con todos los Empleados Acceso a los reportes
Usuario Gerente Administrativo	Web	Encargado de coordinar el trabajo de la empresa. Acceso a reportes y consultas sobre montos a pagar o tributar. Autoriza permisos y anticipos a los empleados.

Usuario Contadora General	Web	Encargado de realizar ajustes en roles de pago.
Usuario Auxiliar de Contabilidad	Web	Encargado de digitar horas de entrada y salida. Registrar permisos, ausencias. Elaborar roles de pago Registro de información de empleados

Fuente: Autora

4.3 Ámbito del Proyecto

Desarrollo de una aplicación web para Nomina de Empleados y Rol de Pagos para la Distribuidora Soria C.A de la ciudad de Ambato.

4.3.1 Requerimientos Funcionales

Requerimiento 1: Gestionar los datos del usuario

Requerimiento 2: Registrar los datos del empleado

Requerimiento 3: Registrar los datos de formación académica.

Requerimiento 4: Registrar los datos de experiencia.

Requerimiento 5: Registrar los datos de capacitación.

Requerimiento 6: Registrar los datos de Grupo Familiar.

Requerimiento 7: Registrar pagos

Requerimiento 8: Registrar los datos de ingresos.

Requerimiento 9: Registrar los datos de egresos.

Requerimiento 10: Registrar roles de pago

Requerimiento 11: Validar los diferentes usuarios para el acceso al sistema

Reportes:

Emitir la ficha del empleado

Emitir listado de empleados

Emitir rol de pagos.

Emitir un reporte de descuentos por empleado.

Emitir un reporte de préstamos por empleado.

Emitir un reporte de atrasos por empleado.

Emitir un reporte de multas por empleado.

Emitir un reporte de horas extras por empleado.

4.3.2 Requerimientos No Funcionales

- Rendimiento
- Disponibilidad
- Seguridad
- Portabilidad
- Mantenibilidad
- Estabilidad
- Reusabilidad

4.4 Objetivos del Proyecto

4.4.1 Objetivos del Negocio

- Registrar a los diferentes usuarios con su respectiva cuenta
- Llevar el control transparente de los procesos
- Limitar el acceso autorizado solo a personas autorizadas
- Generación de reportes

4.4.2 Objetivos del Diseño

- Garantizar un acceso rápido a la aplicación
- Proteger contra el acceso de intrusos mediante la autentificación de usuarios
- Proporcionar una interfaz web amigable y de fácil manejo

4.5 Riesgos

El riesgo implica modificaciones que pueden darse por cambios de opinión, de acciones, de lugares, es inevitable e implica incertidumbre, además pérdida de tiempo cuando el riesgo se ha convertido en problema.

Debemos analizar qué riesgos podrían hacer que nuestro proyecto fracasara, y se debe escoger adecuadamente que acciones pueden convertirse en riesgos para de esta manera lograr superarlos.

El propósito esencial de este proceso de análisis de riesgos va enfocado a prevenir muchos acontecimientos que pueden dificultar el desarrollo de la Nómina de Empleados y Rol de Pagos de los mismos.

Por lo cual debemos darle una gran importancia a este análisis tomando en cuenta todos los tipos de riesgos que se puede encontrar en el desarrollo del sistema al mismo tiempo gestionar aquellos riesgos con gran probabilidad de impacto.

4.6 Identificación del Riesgo

La identificación del riesgo es un intento sistemático para especificar las amenazas al plan de proyecto (estimaciones, planificación temporal, carga de recursos, etc.).

Identificando los riesgos conocidos y predecibles, el gestor del proyecto da un paso adelante para evitarlos cuando sea posible y controlarlos cuando sea necesario.

Existen tres tipos de riesgo:

- Riesgo del proyecto
- Riesgo técnico
- Riesgo del negocio

TABLA IV. XX Identificación de Riesgos

Id	Descripción del riesgo	Categoría	Consecuencia
Req1	Incompleta especificación de los requerimientos	Proyecto	Retraso del proyecto Pérdida de tiempo

Req2	Cambios continuos de los requerimientos por parte de la Distribuidora Soria C.A	Negocio	Diseños realizados sin necesidad Pérdida de recursos económicos
Req3	Falta de conocimiento de los desarrolladores del lenguaje de programación	Técnico	Amenaza la calidad Capacitación a los programadores Retraso del proyecto
Req4	Costo final del proyecto exagerado	Proyecto	Retraso del proyecto Costo del proyecto
Req5	Interfaces inadecuadas para la validación	Técnico	Amenaza la calidad del software Implementación difícil
Req6	Falta de comunicación con el personal de la Distribuidora C.A.	Proyecto	Retraso del proyecto
Req7	Información modificada no se actualiza en la base de datos	Negocio	Provoca inconsistencia en los datos

Fuente: Autora

4.6.1 Análisis de Riesgos

TABLA IV. XXI Valoración de Riesgos

Rango de Probabilidad	Descripción	Valor
1% - 33%	Baja	1
34% - 67%	Media	2
68% - 99%	Alta	3

Fuente: Autora

TABLA IV. XXII Probabilidad

Identificación	Probabilidad		
	%	Valor	Probabilidad
Req1	20	1	Baja
Req2	20	1	Baja
Req3	30	1	Baja
Req4	40	2	Baja
Req5	20	1	Baja
Req6	50	1	Media
Req7	60	1	Media

Fuente: Autora

Determinación del Impacto

TABLA IV. XXIII Impacto del Riesgo

Impacto	Retraso	Impacto Técnico	Costo	Valor
Bajo	1 semana	Ligero efecto en el desarrollo del proyecto	<1%	1
Moderado	2 semanas	Moderado efecto en el desarrollo del proyecto	<5%	2
Alto	1 mes	Severo efecto en el desarrollo del proyecto	<10%	3
Crítico	>1 mes	Proyecto no puede ser culminado	>10%	4

Fuente: Autora

Impacto - Riesgo

TABLA IV. XXIV Impacto - Riesgo

Identificación	Impacto	
	Valor	**Impacto**
Req1	3	Alto
Req2	3	Alto
Req3	1	Bajo
Req4	3	Alto
Req5	1	Bajo
Req6	2	Media
Req7	3	Media

Fuente: Autoras

Determinación de la exposición al Riesgo

Exposición al riesgo	Valor	Color
Baja	1 o 2	
Media	3 o 4	
Alta	>6	

Fuente: Autora

Impacto / Probabilidad	Baja=1	Mediado=2	Alto=3	Crítico=4
Alta=3	3	6	9	12
Media=2	2	4	6	8
Baja=1	1	2	3	4

Fuente: Autora

A continuación en base a los valores de riesgo y a la línea de corte se determina los Riesgos más importantes a tomar en cuenta para poder tomar una solución oportuna.

TABLA IV. XXV Riesgo

Id	Probabilidad			Impacto		Exposición al Riesgo	
	%	Valor	Probabilidad	Valor	Impacto	Valor	Exposición
Req1	20	1	Baja	3	Alto	3	Media
Req2	20	1	Baja	3	Alto	3	Media
Req3	30	1	Baja	1	Bajo	1	Baja
Req4	40	2	Media	3	Alto	6	Alta
Req5	20	1	Baja	1	Bajo	1	Baja
Req6	50	1	Media	2	Moderado	2	Baja
Req7	60	2	Media	3	Alto	6	Media

Fuente: Autora

TABLA IV. XXVI Prioridades del Riesgo

Id	Prioridad	Exposición
Req4	1	6
Req1	2	3
Req2	2	3
Req7	2	3
Req3	4	1
Req5	4	1
Req6	3	2

Fuente: Autora

4.6.2 Planeación y Programación del Riesgo

Para mitigar el riesgo se utiliza la Hoja de Gestión de Riesgo

TABLA IV. XXVII Hoja de Gestión de Riesgo 1

<table>
<tr><th colspan="4">HOJA DE GESTIÓN DEL RIESGO</th></tr>
<tr><td colspan="2">ID. DEL RIESGO: Requerimiento 1</td><td colspan="2">FECHA: 27/07/2014</td></tr>
<tr><td>Probabilidad: Baja
Valor: 1</td><td>Impacto: Alto
Valor: 3</td><td>Exposición: Alta
Valor: 3</td><td>Prioridad: 1
Valor:</td></tr>
<tr><td colspan="4">DESCRIPCIÓN: Especificación incompleta de requerimientos</td></tr>
<tr><td colspan="4">REFINAMIENTO:
Causas:
Mala información de requerimientos por parte de los usuarios
Mala interpretación de los requerimientos por parte de los desarrolladores

Consecuencias:
Retardo del proyecto
Incremento del costo del proyecto</td></tr>
<tr><td colspan="4">REDUCCIÓN:
• Mejorar la comunicación entre cliente y desarrolladores
• Evitar los frecuentes cambios de requerimientos</td></tr>
<tr><td colspan="4">SUPERVISIÓN:
• Supervisar el ambiente de trabajo de los miembros del equipo de trabajo y cliente
• Acuerdo entre el cliente y responsables sobre sus necesidades</td></tr>
</table>

GESTIÓN • Realizar los trabajos manteniendo la integridad y ética profesional • Mutuo acuerdo para establecer los requerimientos
ESTADO ACTUAL: Frase de reducción iniciada [x] Fase de supervisión iniciada [ ] Gestionando el riesgo [ ]
RESPONSABLES: Jenny Carrera

Fuente: Autora

TABLA IV. XXVIII Hoja de Gestión de Riesgo 1

HOJA DE GESTIÓN DEL RIESGO			
ID. DEL RIESGO: Requerimiento 2		**FECHA:** 27/07/2014	
Probabilidad: Baja **Valor:** 1	**Impacto:** Alto **Valor:** 3	**Exposición:** Alta **Valor:** 3	**Prioridad:** 1 **Valor:**
DESCRIPCIÓN: Cambios continuos de los requerimientos por parte de la Distribuidora SORIA C.A			
REFINAMIENTO: **Causas:** Falta de análisis previo de los requerimientos por parte del cliente			

Consecuencias: Pérdida de recursos No complacer las expectativas para el sistema
REDUCCIÓN: • Asignación de recursos para una correcta captación de los requerimientos
SUPERVISIÓN: • Seguimiento de las funcionalidades para los requerimientos planteados
GESTIÓN • Desplegar los beneficios de un requerimiento gestionado
ESTADO ACTUAL: Frase de reducción iniciada [X] Fase de supervisión iniciada [ ] Gestionando el riesgo [ ]
RESPONSABLES: Jenny Carrera

Fuente: Autora

4.7 Planificación Inicial

4.7.1 Factibilidad

Factibilidad Técnica

La factibilidad técnica ayuda a determinar si la propuesta puede ser implementada con el hardware, software y recurso humano disponible.

Para el desarrollo de la aplicación web "Nomina de Empleados y Rol de Pagos" para la Distribuidora Soria C.A, cuenta con los recursos necesarios tanto en la parte hardware y software. A continuación se detalla los recursos existentes

Hardware Existente

Hardware con el que cuenta para el desarrollo de la aplicación:

TABLA IV. XXIX Hardware Existente

Cantidad	Descripción	Observación
1	Computadora	Desarrollo de la aplicación y la documentación
1	Infraestructura de red	Acceder a internet para consultas en el desarrollo de la aplicación y realizar las correspondientes pruebas

Fuente: Autora

Hardware Requerido

TABLA IV. XXX Hardware Existente

Cantidad	Descripción	Observación
1	Servidor	Realizar pruebas de la configuración del servidor
1	Impresora	Impresión de reportes

Fuente: Autora

Software Existente

TABLA IV. XXXI Hardware Existente

Nombre	Descripción
Windows 7 Profesional	Sistema Operativo
ArgoUML	Herramienta para diseño UML

Fuente: Autora

Software Requerido

TABLA IV. XXXII Hardware Existente

Nombre	Descripción
Visual Studio 2012	Entorno de desarrollo
Report Viewer	Reportes
IIS	Servidor de aplicaciones web

Fuente: Autora

Recurso Humano Requerido

TABLA IV. XXXIII Recurso Humano Requerido

Función	Formación
Jefe de Proyecto	Ingeniería en Sistemas
Equipo de desarrolladores	Estudiante de Ingeniería en Sistemas
Administrador de Base de Datos	Ingeniería en Sistemas
Diseñador	Estudiante de Ingeniería en Sistemas

Fuente: Autora

Facilidad Operativa

Recurso Humano

El recurso humano que participará del sistema son:

Usuarios directos

Los usuarios directos a capacitar para el manejo del sistema son:

Personal a Capacitar

TABLA IV. XXXIV Personal a Capacitar

Nombre	Función
U. Gerente Administrativo	Se encarga de coordinar el trabajo de la empresa como tal.
U. Contadora General	Se encarga de supervisar el trabajo desarrollado por los auxiliares de contabilidad.
U. Auxiliares de contabilidad	Se encargan de realizar los procesos propiamente dichos para la empresa.

Fuente: Autora

Factibilidad Económica

El tiempo de duración de este proyecto será de 12 meses

Costos

TABLA IV. XXXV Costo de Desarrollo

Costos de desarrollo		
Costo personal (mensual)		
Jefe de proyecto y desarrollador	$800	$9.600
Administrador de BD y diseñador	$160	$3.200
	Total Costo Personal	**$12.800**
Costo de hardware y software		

Costos Hardware		
Computadoras	$1200	$2400
	Total Costo Hardware	**$2400**
Costo Software		
Internet	$400	
	Total Costo Software	**$400**
Costos varios		
Suministros	$300	
Alimentación	$800	
Papel A4	$10	
	Total Costos Varios	**$1.210**
COSTO TOTAL DE DESARROLLO		**$16.810**

Fuente: Autora

Análisis Costo Beneficio

Los beneficios que se podrá obtener con la utilización de este sistema son los siguientes:

Permitirá realizar el proceso de gestión de bienes (materiales) existentes para la obras de una forma eficiente y real, permitiendo organizar, reducir el flujo de trabajo de los departamentos, además usar la tecnología existen en la institución y reducir la contaminación con el uso de papel, tinta, cartuchos y control del uso de los materiales destinado a los lugares para los que fueron adquiridos.

Se podrá realizar un mayor control del uso de los combustibles y de los bienes que existen en la institución.

Mediante el sistema se puede generar reportes que ayudarán a la toma de decisiones de las autoridades.

Se optimizará el tiempo en la elaboración de informes.

La utilización del sistema será fácil de utilizar y además los usuarios pueden ingresar desde cualquier lugar.

4.8 Fase de Planificación

Obtener un cronograma de trabajo que cumpla con lo especificado en la fase de visión, desarrollar los requerimientos funcionales y no funcionales, describir el escenario, diagramar casos de uso.

PLANEACIÓN

En esta fase se realiza la preparación de la especificación funcional, diseño conceptual.

4.8.1 Diseño Conceptual

4.8.1.1 Requerimientos Funcionales

A continuación se describe cada uno de los requerimientos funcionales

4.8.1.1.1 Requerimiento Funcional 1

- **Introducción**

  El sistema deberá permitir gestionar los datos del usuario

- **Entrada**

Fuentes de entradas: User y Password

Frecuencia: Bajo demanda

Requisitos de control: Controla que los campos del formulario estén llenos y que los datos sean los correctos

Entradas válidas: Todos los campos sean válidos

- **Procesos:**

1. El usuario deberá ingresar su user y password para la autenticación.

2. El sistema validará la información

3. Si el dato es verdadero

    a. Ingresará al sistema

4. Caso Contrario

    a. Mensajes de error

    b. Ingreso

    c. Aceptar

5. Validación de datos

    a. Si todos los campos están llenos y los datos son correctos

Actualización de la base de datos y visualización de resultados

b. Caso Contrario

Mensaje de error

- **Salidas**

Destino de las salidas

Muestra listado con los datos ingresados cuando son correctos

Interfaces de hardware

El principal medio hardware de visualización es la pantalla de su computador, se utilizará para mostrar cada uno de los procesos que se efectuarán.

Interfaces de Software

- La herramienta de desarrollo que se utilizará será Visual Studio 2012(.NET) para que la aplicación cumpla los requerimientos.
- El repositorio de datos será Postgres SQL 8.4.

Interfaces de Software

Para establecer la comunicación el sistema utilizará el protocolo TCP/IP.

4.8.1.1.2 Requerimiento Funcional 2

- **Introducción**

El sistema deberá permitir gestionar los datos del empleado

- **Entrada**

Fuentes de entradas: User y password

Frecuencia: Bajo demanda

Requisitos de control: Controla que los campos del formulario estén llenos y que los datos sean los correctos

Entradas válidas: Todos los campos sean válidos

- **Procesos:**

1. El auxiliar de contabilidad deberá ingresar su cuenta y clave para la autenticación.

2. El sistema validará la información

3. Si el dato es verdadero

    a. Ingresará al sistema

4. Caso Contrario

    a. Mensajes de error

    b. Ingreso

    c. Aceptar

5. Validación de datos

    a. Si todos los campos están llenos y los datos son correctos

Actualización de la base de datos y visualización de resultados

b. Caso Contrario

Mensaje de error

- **Salidas**

Destino de las salidas

Muestra listado con los datos ingresados cuando son correctos

Interfaces de hardware

El principal medio hardware de visualización es la pantalla de su computador, se utilizará para mostrar cada uno de los procesos que se efectuarán.

Interfaces de Software

- La herramienta de desarrollo que se utilizará será Visual Studio 1012(.net) para que la aplicación cumpla los requerimientos.
- El repositorio de datos será PostgreSQL 8.4

Interfaces de Software

Para establecer la comunicación el sistema utilizará el protocolo TCP/IP.

4.8.1.1.3 Requerimiento Funcional 3

- **Introducción**

El sistema deberá permitir registrar los datos de formación académica.

- **Entrada**

Fuentes de entradas: Cuenta y clave

Frecuencia: Bajo demanda

Requisitos de control: Controla que los campos del formulario estén llenos y que los datos sean los correctos

Entradas válidas: Todos los campos sean válidos

- **Procesos:**

1. El auxiliar de contabilidad deberá ingresar su usuario y clave para la autenticación.

2. El sistema validará la información

3. Si el dato es verdadero

    a. Ingresará al sistema

4. Caso Contrario

    a. Mensajes de error

    b. Ingreso

    c. Aceptar

5. Validación de datos

a. Si todos los campos están llenos y los datos son correctos

Actualización de la base de datos y visualización de resultados

b. Caso Contrario

Mensaje de error

- **Salidas**

Destino de las salidas

Muestra listado con los datos ingresados cuando son correctos

Interfaces de hardware

El principal medio hardware de visualización es la pantalla de su computador, se utilizará para mostrar cada uno de los procesos que se efectuarán.

Interfaces de Software

- La herramienta de desarrollo que se utilizará será Visual Studio 2012 para que la aplicación cumpla los requerimientos.
- El repositorio de datos será PostgreSQL 8.4

Interfaces de Software

Para establecer la comunicación el sistema utilizará el protocolo TCP/IP.

4.8.1.1.4 Requerimiento Funcional 4

- **Introducción**

El sistema deberá permitir registrar los datos de experiencia

- **Entrada**

Fuentes de entradas: Cuenta y clave

Frecuencia: Bajo demanda

Requisitos de control: Controla que los campos del formulario estén llenos y que los datos sean los correctos

Entradas válidas: Todos los campos sean válidos

- **Procesos:**

1. El auxiliar de contabilidad deberá ingresar su usuario y clave para la autenticación.

2. El sistema validará la información

3. Si el dato es verdadero

    a. Ingresará al sistema

4. Caso Contrario

    a. Mensajes de error

    b. Ingreso

    c. Aceptar

5. Validación de datos

a. Si todos los campos están llenos y los datos son correctos

Actualización de la base de datos y visualización de resultados

b. Caso Contrario

Mensaje de error

- **Salidas**

Destino de las salidas

Muestra listado con los datos ingresados cuando son correctos

Interfaces de hardware

El principal medio hardware de visualización es la pantalla de su computador, se utilizará para mostrar cada uno de los procesos que se efectuarán.

Interfaces de Software

- La herramienta de desarrollo que se utilizará será Visual Studio 2012(.NET) para que la aplicación cumpla los requerimientos.
- El repositorio de datos será PostgreSQL 8.4

Interfaces de Software

Para establecer la comunicación el sistema utilizará el protocolo TCP/IP.

4.8.1.1.5 Requerimiento Funcional 5

- **Introducción**

El sistema deberá permitir registrar los datos de capacitación.

- **Entrada**

Fuentes de entradas: Cuenta y clave

Frecuencia: Bajo demanda

Requisitos de control: Controla que los campos del formulario estén llenos y que los datos sean los correctos

Entradas válidas: Todos los campos sean válidos

- **Procesos:**

1. El auxiliar de contabilidad deberá ingresar su usuario y clave para la autenticación.

2. El sistema validará la información

3. Si el dato es verdadero

    a. Ingresará al sistema

4. Caso Contrario

    a. Mensajes de error

    b. Ingreso

c. Aceptar

5. Validación de datos

a. Si todos los campos están llenos y los datos son correctos

Actualización de la base de datos y visualización de resultados

b. Caso Contrario

Mensaje de error

- **Salidas**

Destino de las salidas

Muestra listado con los datos ingresados cuando son correctos

Interfaces de hardware

El principal medio hardware de visualización es la pantalla de su computador, se utilizará para mostrar cada uno de los procesos que se efectuarán.

Interfaces de Software

- La herramienta de desarrollo que se utilizará será Visual Studio 2012(.NET) para que la aplicación cumpla los requerimientos.
- El repositorio de datos será PostgreSQL 8.4

Interfaces de Software

Para establecer la comunicación el sistema utilizará el protocolo TCP/IP.

4.8.1.1.6 Requerimiento Funcional 6

- **Introducción**

El sistema deberá permitir registrar los datos de grupo familiar.

- **Entrada**

Fuentes de entradas: Cuenta y clave

Frecuencia: Bajo demanda

Requisitos de control: Controla que los campos del formulario estén llenos y que los datos sean los correctos

Entradas válidas: Todos los campos sean válidos

- **Procesos:**

1. El auxiliar de contabilidad deberá ingresar su usuario y su clave para la autenticación.

2. El sistema validará la información

3. Si el dato es verdadero

   a. Ingresará al sistema

4. Caso Contrario

   a. Mensajes de error

b. Ingreso

c. Aceptar

5. Validación de datos

a. Si todos los campos están llenos y los datos son correctos

Actualización de la base de datos y visualización de resultados

b. Caso Contrario

Mensaje de error

- **Salidas**

Destino de las salidas

Muestra listado con los datos ingresados cuando son correctos

Interfaces de hardware

El principal medio hardware de visualización es la pantalla de su computador, se utilizará para mostrar cada uno de los procesos que se efectuarán.

Interfaces de Software

- La herramienta de desarrollo que se utilizará será Visual Studio 2012(.NET) para que la aplicación cumpla los requerimientos.
- El repositorio de datos será Postgres SQL 8.4

Interfaces de Software

Para establecer la comunicación el sistema utilizará el protocolo TCP/IP.

4.8.1.1.7 Requerimiento Funcional 7

- **Introducción**

  El sistema deberá permitir registrar los datos de pagos (decimo tercero, decimo cuarto, rol de pagos)

- **Entrada**

  Fuentes de entradas:

  Frecuencia: Bajo demanda

  Requisitos de control: Controla que los campos del formulario estén llenos y que los datos sean los correctos

  Entradas válidas: Todos los campos sean válidos

- **Procesos:**

  1. El auxiliar de contabilidad deberá digitar su usuario y clave para la autenticación.

  2. El sistema validará la información

  3. Si el dato es verdadero

     a. Ingresará al sistema

  4. Caso Contrario

a. Mensajes de error

b. Ingreso

c. Aceptar

5. Validación de datos

a. Si todos los campos están llenos y los datos son correctos

Actualización de la base de datos y visualización de resultados

b. Caso Contrario

Mensaje de error

- **Salidas**

Destino de las salidas

Muestra listado con los datos ingresados cuando son correctos

Interfaces de hardware

El principal medio hardware de visualización es la pantalla de su computador, se utilizará para mostrar cada uno de los procesos que se efectuarán.

Interfaces de Software

- La herramienta de desarrollo que se utilizará será Visual Studio 2012(.NET) para que la aplicación cumpla los requerimientos.
- El repositorio de datos será Postgres SQL 8.4

Interfaces de Software

Para establecer la comunicación el sistema utilizará el protocolo TCP/IP.

4.8.1.1.8 Requerimiento Funcional 8

- **Introducción**

  El sistema deberá permitir registrar los datos de ingreso (horas extras, comisiones)

- **Entrada**

  Fuentes de entradas:

  Frecuencia: Bajo demanda

  Requisitos de control: Controla que los campos del formulario estén llenos y que los datos sean los correctos

  Entradas válidas: Todos los campos sean válidos

- **Procesos:**

  1. El auxiliar de contabilidad deberá digitar su usuario y clave para la autenticación.

  2. El sistema validará la información

  3. Si el dato es verdadero

     a. Ingresará al sistema

  4. Caso Contrario

a. Mensajes de error

b. Ingreso

c. Aceptar

5. Validación de datos

a. Si todos los campos están llenos y los datos son correctos

Actualización de la base de datos y visualización de resultados

b. Caso Contrario

Mensaje de error

- **Salidas**

Destino de las salidas

Muestra listado con los datos ingresados cuando son correctos

Interfaces de hardware

El principal medio hardware de visualización es la pantalla de su computador, se utilizará para mostrar cada uno de los procesos que se efectuarán.

Interfaces de Software

- La herramienta de desarrollo que se utilizará será Visual Studio 2012(.NET) para que la aplicación cumpla los requerimientos.
- El repositorio de datos será Postgres SQL 8.4

Interfaces de Software

Para establecer la comunicación el sistema utilizará el protocolo TCP/IP.

4.8.1.1.9 Requerimiento Funcional 9

- **Introducción**

  El sistema deberá permitir registrar los datos de egresos (descuentos, multas, atrasos, préstamos, faltas)

- **Entrada**

  Fuentes de entradas:

  Frecuencia: Bajo demanda

  Requisitos de control: Controla que los campos del formulario estén llenos y que los datos sean los correctos

  Entradas válidas: Todos los campos sean válidos

- **Procesos:**

  1. El auxiliar de contabilidad deberá digitar su usuario y clave para la autenticación.

  2. El sistema validará la información

  3. Si el dato es verdadero

     a. Ingresará al sistema

4. Caso Contrario

a. Mensajes de error

b. Ingreso

c. Aceptar

5. Validación de datos

a. Si todos los campos están llenos y los datos son correctos

Actualización de la base de datos y visualización de resultados

b. Caso Contrario

Mensaje de error

- **Salidas**

Destino de las salidas

Muestra listado con los datos ingresados cuando son correctos

Interfaces de hardware

El principal medio hardware de visualización es la pantalla de su computador, se utilizará para mostrar cada uno de los procesos que se efectuarán.

Interfaces de Software

- La herramienta de desarrollo que se utilizará será Visual Studio 2012(.NET) para que la aplicación cumpla los requerimientos.

- El repositorio de datos será Postgres SQL 8.4

Interfaces de Software

Para establecer la comunicación el sistema utilizará el protocolo TCP/IP.

4.8.1.1.10 Requerimiento Funcional 10

- **Introducción**

  El sistema deberá permitir registrar los datos de roles de pago.

- **Entrada**

  Fuentes de entradas:

  Frecuencia: Bajo demanda

  Requisitos de control: Controla que los campos del formulario estén llenos y que los datos sean los correctos

  Entradas válidas: Todos los campos sean válidos

- **Procesos:**

  1. El auxiliar de contabilidad deberá digitar su usuario y clave para la autenticación.

  2. El sistema validará la información

  3. Si el dato es verdadero

     a. Ingresará al sistema

4. Caso Contrario

a. Mensajes de error

b. Ingreso

c. Aceptar

5. Validación de datos

a. Si todos los campos están llenos y los datos son correctos

Actualización de la base de datos y visualización de resultados

b. Caso Contrario

Mensaje de error

- **Salidas**

Destino de las salidas

Muestra listado con los datos ingresados cuando son correctos

Interfaces de hardware

El principal medio hardware de visualización es la pantalla de su computador, se utilizará para mostrar cada uno de los procesos que se efectuarán.

Interfaces de Software

- La herramienta de desarrollo que se utilizará será Visual Studio 2012(.NET) para que la aplicación cumpla los requerimientos.

- El repositorio de datos será Postgres SQL 8.4

Interfaces de Software

Para establecer la comunicación el sistema utilizará el protocolo TCP/IP.

4.8.1.2 Requerimientos no Funcionales

Requerimientos no funcionales con sus perspectivas

- **Rendimiento**

  Tiempos de respuesta para acceder a la página de autentificación del sistema máximo de 30 segundos

- **Disponibilidad**

  El sistema estará fuera de línea máximo 30 minutos. Empleo de sistemas de respaldo.

- **Seguridad**

  Para ingresar al sistema primero deberá el usuario autentificarse dependiendo su tipo de usuario asignado por el administrador para acceder al ambiente de trabajo

- **Portabilidad**

  Garantizar compatibilidad con los otros sistemas operativos: Windows XP, Windows 7

- **Mantenibilidad**

  Documentación del diseño y la codificación de la aplicación

- **Escabilidad**

  Diseño de la arquitectura empleando módulos independientes

- **Interfaces**

  Interfaces realizadas en Visual Studio 2012(.NET)

4.8.1.3 Actores

- **Usuario Administrador:** Que se encarga de la creación de usuarios con sus respectivos roles para restringir la información y no ser manipulada por accesos erróneos.

- **Usuario Gerente Administrativo**, que se encarga de coordinar el trabajo de la empresa como tal. El no realiza los trabajos operativos, pero sí recibe informes, realiza consultas sobre montos a pagar o tributar, concede permisos a los empleados, autoriza anticipos, junto con el Cuerpo Administrativo interviene en la fijación de sueldos y salarios de los empleados.
- **Usuario Contadora General,** que es encarga de supervisar el trabajo desarrollado por los auxiliares de contabilidad. No realiza los roles de pagos, pero puede efectuar ajustes sobre los mismos. Realiza consultas, al igual que el Gerente administrativo, sobre resultados del proceso, con la diferencia de que los puede modificar también.
- **Usuarios Auxiliares de Contabilidad**, que se encargan de realizar los procesos propiamente dichos; digitan horas de entrada y salida de los empleados, registran los permisos, ausencias, elaboran los roles de pagos es decir, serán las personas que realizarán el ingreso de datos en la aplicación.

4.8.1.3.1 Casos de Uso

Caso de Uso - Administrador

El usuario administrador tendrá control total de la aplicación

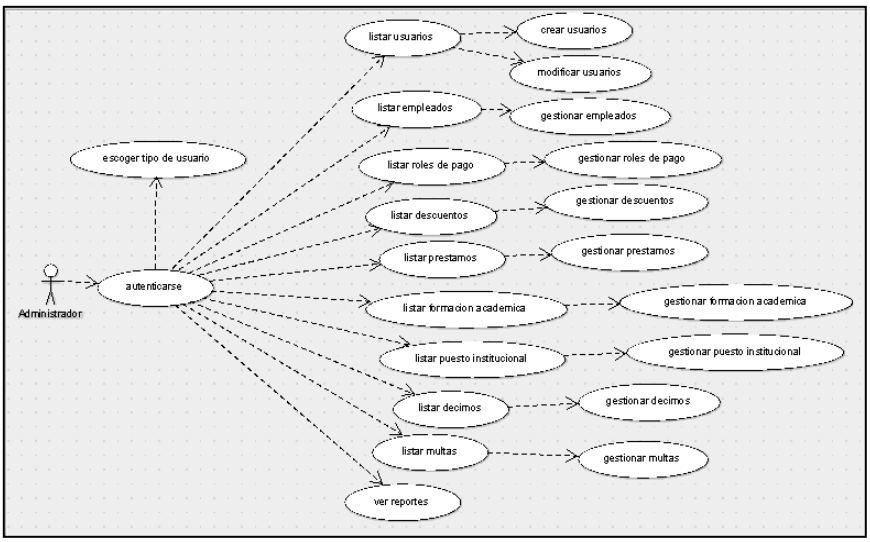


FIGURA IV. 24 Caso de Uso – Administrador

Fuente: Autora

Caso de Uso – Gerente Administrativo

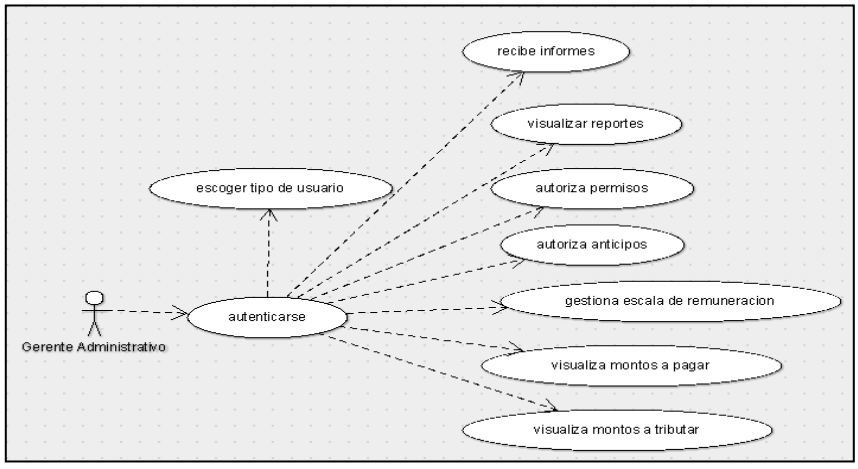


FIGURA IV. 25 Caso de Uso – Gerente Administrativo

Fuente: Autora

Caso de Uso – Contadora General

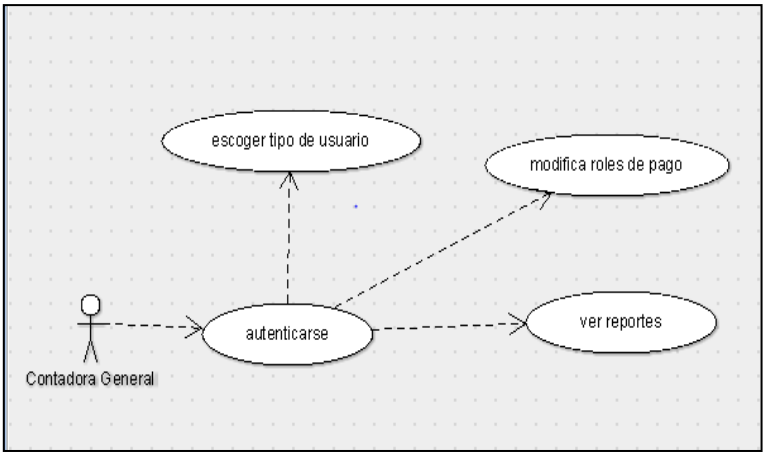


FIGURA IV. 26 Caso de Uso – Contadora General

Fuente: Autora

Caso de Uso – Auxiliar de Contabilidad

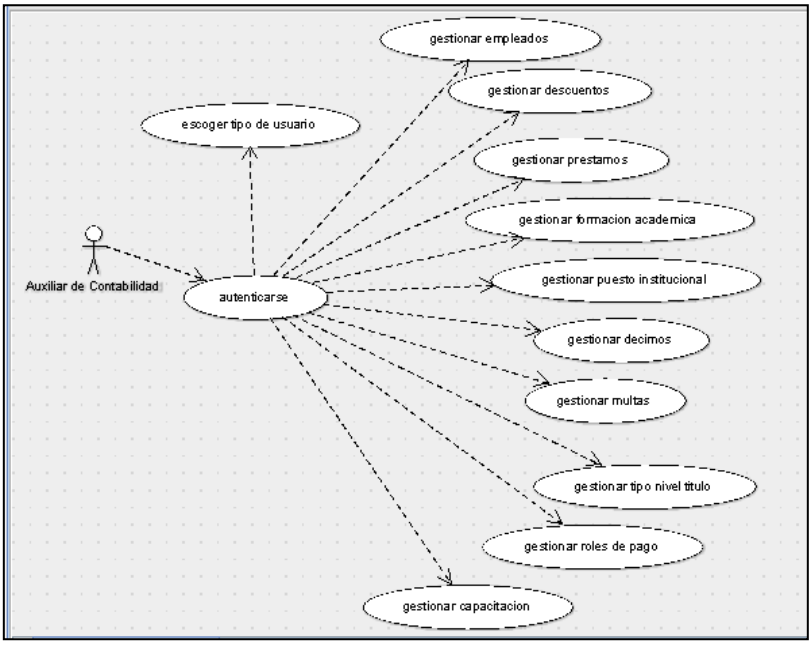


FIGURA IV. 22 Caso de Uso – Auxiliar de Contabilidad

Fuente: Autora

4.8.1.3.2 Escenarios

Autentificación Usuario Administrador

- **Personas Alternativas**

TABLA IV. XXXVI Personas Alternativas para el Administrador

Persona	Desviaciones (si es aplicable)
Persona delegada por el representante de la Distribuidora SORIA C.A.	Ninguno

Fuente: Autora

- **Descripción del Escenario**

El administrador ingresará su usuario, password y tipo en la interfaz de autentificación. Podrá crear usuarios e interactuar con la aplicación.

4.8.1.3.3 Glosario de Términos

TABLA IV. XXXVII Glosario de Términos

Término	Descripción
MSF	Microsoft Solution Framework, metodología empleada para el desarrollo de la aplicación
UML	Lenguaje Unificado de Modelado, es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema
Módulos	Distintas dependencias del sistema que brinda información del estudiante
Hardware	Son las características físicas y tangibles del computador utilizado
Software	Es la parte intangible del computador

Password	Es una serie secreta de caracteres que permite al usuario tener acceso al computador, programa o sitio
Productividad	Capacidad de evolución y aprovechamiento del sistema
Servidor	Es un ordenador de gran potencia, que se encarga de "prestar un servicio" a otros ordenadores que se conectan a él
Sistema Operativo	Es un programa informático que actúa de interfaz entre los dispositivos de hardware y el usuario.
Base de Datos	Conjunto de datos organizados para su almacenamiento en la memoria del computador, diseñado para facilitar su mantenimiento y acceso de una estándar
Usuario	Persona que utiliza el sistema
Administrador	Es la persona encargada de administrar el sistema en su totalidad, además puede generar reportes
Reporte	Documento que se utilizará para visualizar información de acuerdo a lo necesitado
Metodología	Conjunto de pasos lógicos y ordenados para realizar una actividad
Internet	Conjunto descentralizado en redes de comunicación interconectadas que utilizan TCP/IP

Fuente: Autora

4.8.1.3.4 Refinar los Casos de Uso

TABLA IV. XXXVIII Autentificación del Usuario Administrador

Identificar Caso de Uso	CU- Administrador
Nombre del Caso de Uso	Realizar administración del sistema
Actores	Responsables del departamento
Propósito	Crear el correcto ambiente de trabajo

<table>
<tr><td>Visión General</td><td>El responsable para realizar los procesos deberá autentificarse con su user y password</td></tr>
<tr><td>Tipo</td><td>Primario, real y extendido</td></tr>
<tr><td>Referencias</td><td>Caso de Uso: Administrador</td></tr>
<tr><td colspan="2">Curso Típico de Eventos</td></tr>
<tr><td>Acción del Actor</td><td>Respuesta del Sistema</td></tr>
<tr><td>1. El Administrador empieza creando el ambiente de trabajo</td><td>2. El sistema cuenta con un control para la autentificación</td></tr>
<tr><td>3. Autentificación, ingresar el user y password</td><td>4. Valida los datos ingresados</td></tr>
<tr><td></td><td>5. Al autentificarse correctamente podrá ingresar a los procesos correspondientes</td></tr>
<tr><td colspan="2">Cursos Alternativos</td></tr>
<tr><td colspan="2">2.1 El sistema no presenta pantallas para el ingreso de datos</td></tr>
<tr><td colspan="2">Control de campos vacíos

4.1 Ingreso del Usuario

Cuando no se ingresa algún campo obligatorio le informa con un error para llenar

4.2 Si el usuario no está en la base de datos, el sistema informará que no existe</td></tr>
</table>

Fuente: Autora

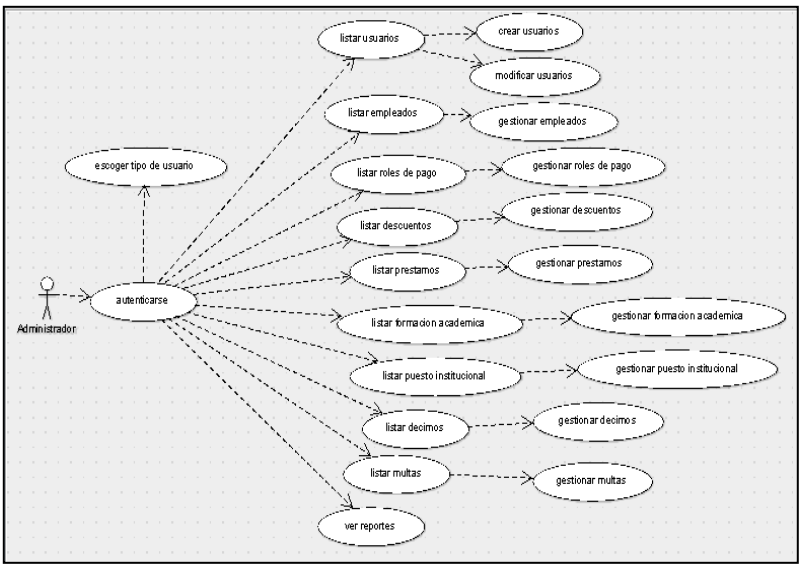


FIGURA IV. 23 Caso de Uso Refinado - Administrador

Fuente: Autora

TABLA IV. XXXIX Autentificación del Usuario Gerente Administrativo

Identificar Caso de Uso	CU- Gerente Administrativo
Nombre del Caso de Uso	Visualiza Reportes
Actores	Gerente Administrativo
Propósito	Obtener los datos de pagos de los empleados
Visión General	El gerente administrativo deberá autentificarse con su user y password, luego visualizara los reportes.
Tipo	Primario, real y extendido
Referencias	Caso de Uso: Gerente Administrativo
Curso Típico de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El Administrador empieza creando el ambiente de trabajo	2. El sistema cuenta con un control para la autentificación

3. Autenticación, ingresar el user y password	4. Valida los datos ingresados
	5. Al autentificarse correctamente podrá ingresar a los procesos correspondientes
	6. Presenta la interfaz en la que el gerente administrativo podrá visualizar reportes, autorizar permisos.
Cursos Alternativos	
2.1 El sistema no presenta pantallas para el ingreso de datos	
Control de campos vacíos 4.1 Ingreso del Usuario Cuando no se ingresa algún campo obligatorio le informa con un error para llenar 4.2 Si el usuario no está en la base de datos, el sistema informará que no existe	

Fuente: Autora

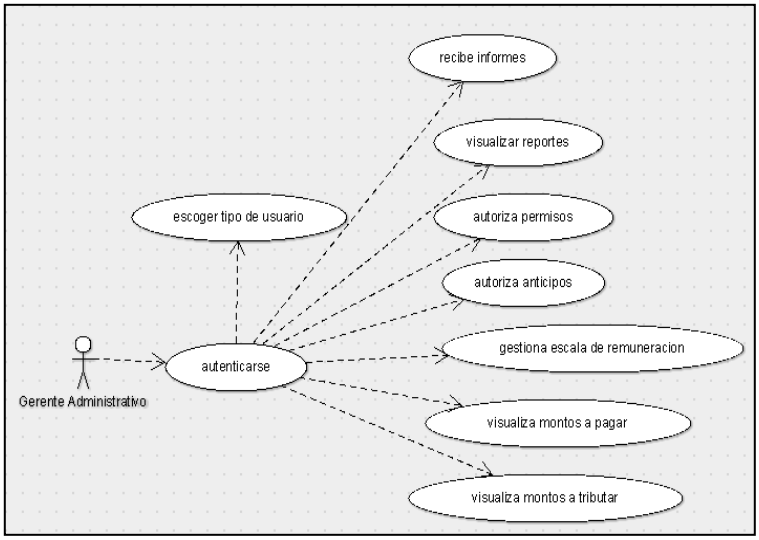


FIGURA IV. 24 Caso de Uso – Gerente Administrativo

Fuente: Autora

TABLA IV. XL Autentificación del Usuario Contadora General

Identificar Caso de Uso	CU- Contadora General
Nombre del Caso de Uso	Realizar administración del sistema
Actores	Responsables del departamento
Propósito	Crear el correcto ambiente de trabajo
Visión General	El responsable para realizar los procesos deberá autentificarse con su user y password
Tipo	Primario, real y extendido
Referencias	Caso de Uso: Contadora General
Curso Típico de Eventos	
Acción del Actor	**Respuesta del Sistema**
6. El Administrador empieza creando el ambiente de trabajo	7. El sistema cuenta con un control para la autentificación
8. Autentificación, ingresar el user y password	9. Valida los datos ingresados
	10. Al autentificarse correctamente podrá ingresar a los procesos correspondientes
Cursos Alternativos	
2.1 El sistema no presenta pantallas para el ingreso de datos	
Control de campos vacíos 4.1 Ingreso del Usuario Cuando no se ingresa algún campo obligatorio le informa con un error para llenar 4.2 Si el usuario no está en la base de datos, el sistema informará que no existe	

Fuente: Autora

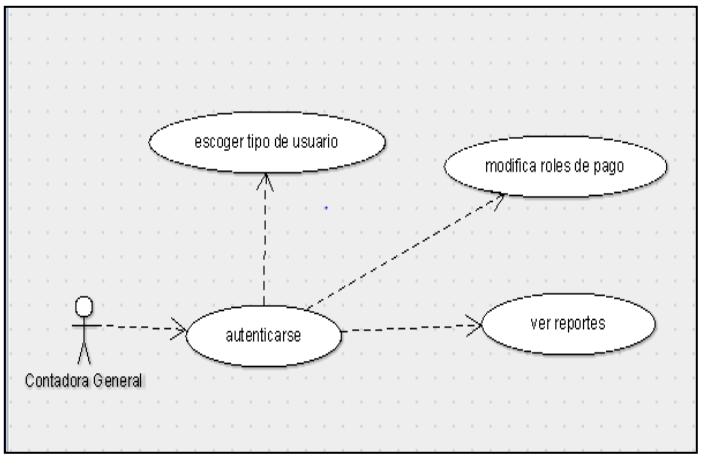


FIGURA IV. 30 Caso de Uso Refinado – Contadora General

Fuente: Autora

TABLA IV. XLI Autentificación del Usuario Auxiliar de Contabilidad

Identificar Caso de Uso	CU- Auxiliar de Contabilidad
Nombre del Caso de Uso	Realizar administración del sistema
Actores	Responsables del departamento
Propósito	Crear el correcto ambiente de trabajo
Visión General	El responsable para realizar los procesos deberá autentificarse con su user y password
Tipo	Primario, real y extendido
Referencias	Caso de Uso: Auxiliar de Contabilidad
Curso Típico de Eventos	
Acción del Actor	**Respuesta del Sistema**
11. El Administrador empieza creando el ambiente de trabajo	12. El sistema cuenta con un control para la autentificación
13. Autentificación, ingresar el user y password	14. Valida los datos ingresados

	15. Al autentificarse correctamente podrá ingresar a los procesos correspondientes
Cursos Alternativos	
2.1 El sistema no presenta pantallas para el ingreso de datos	
Control de campos vacíos 4.1 Ingreso del Usuario Cuando no se ingresa algún campo obligatorio le informa con un error para llenar 4.2 Si el usuario no está en la base de datos, el sistema informará que no existe	

Fuente: Autora

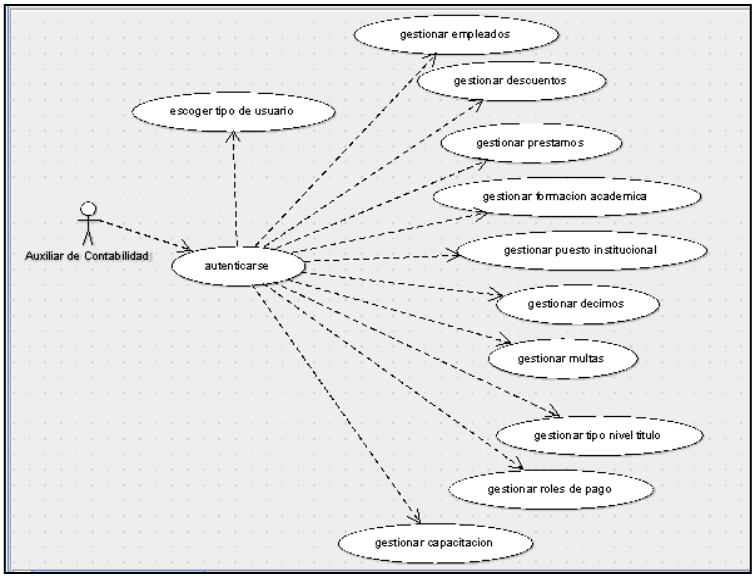


FIGURA IV. 25 Caso de Uso Refinado – Auxiliar de Contabilidad

Fuente: Autora

4.8.2 Diseño Lógico

4.8.2.1 Tecnología a utilizar en el proyecto

- Entorno de desarrollo Visual Studio 2012(.NET)
- Base de Datos PostgreSQL 8.4

- Documentación MSF

4.8.2.2 Diagrama de Secuencia

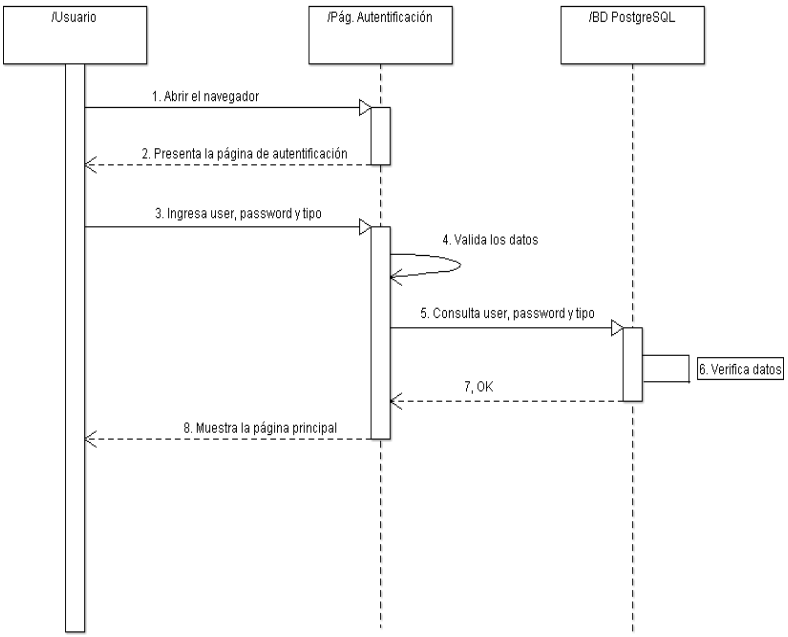


FIGURA IV. 26 Diagrama de Secuencia

Fuente: Autora

4.8.2.3 Diagrama de Clases

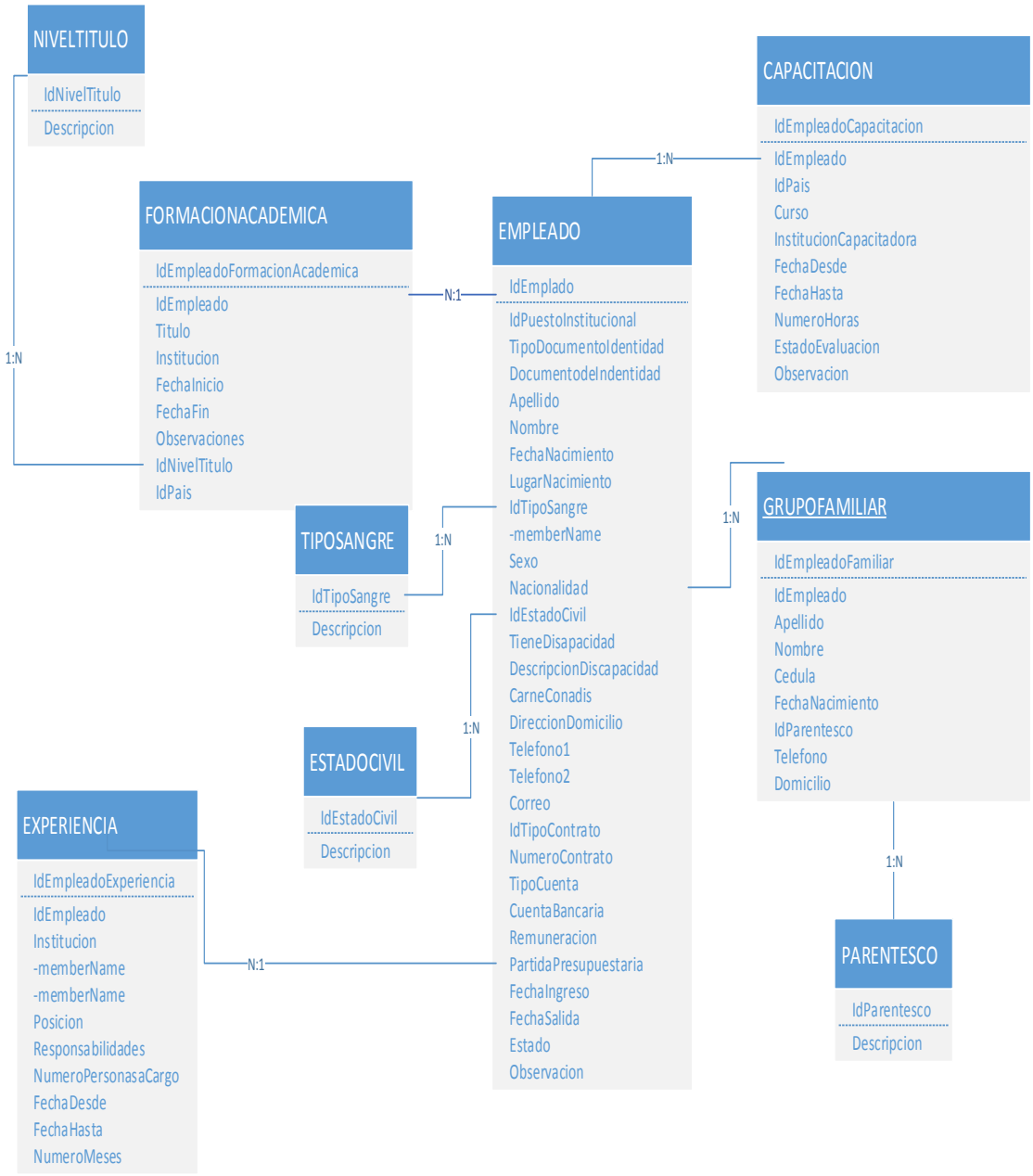


Figura IV. 27 Diagrama de Clases
Datos del Empleado
Fuente: Autora

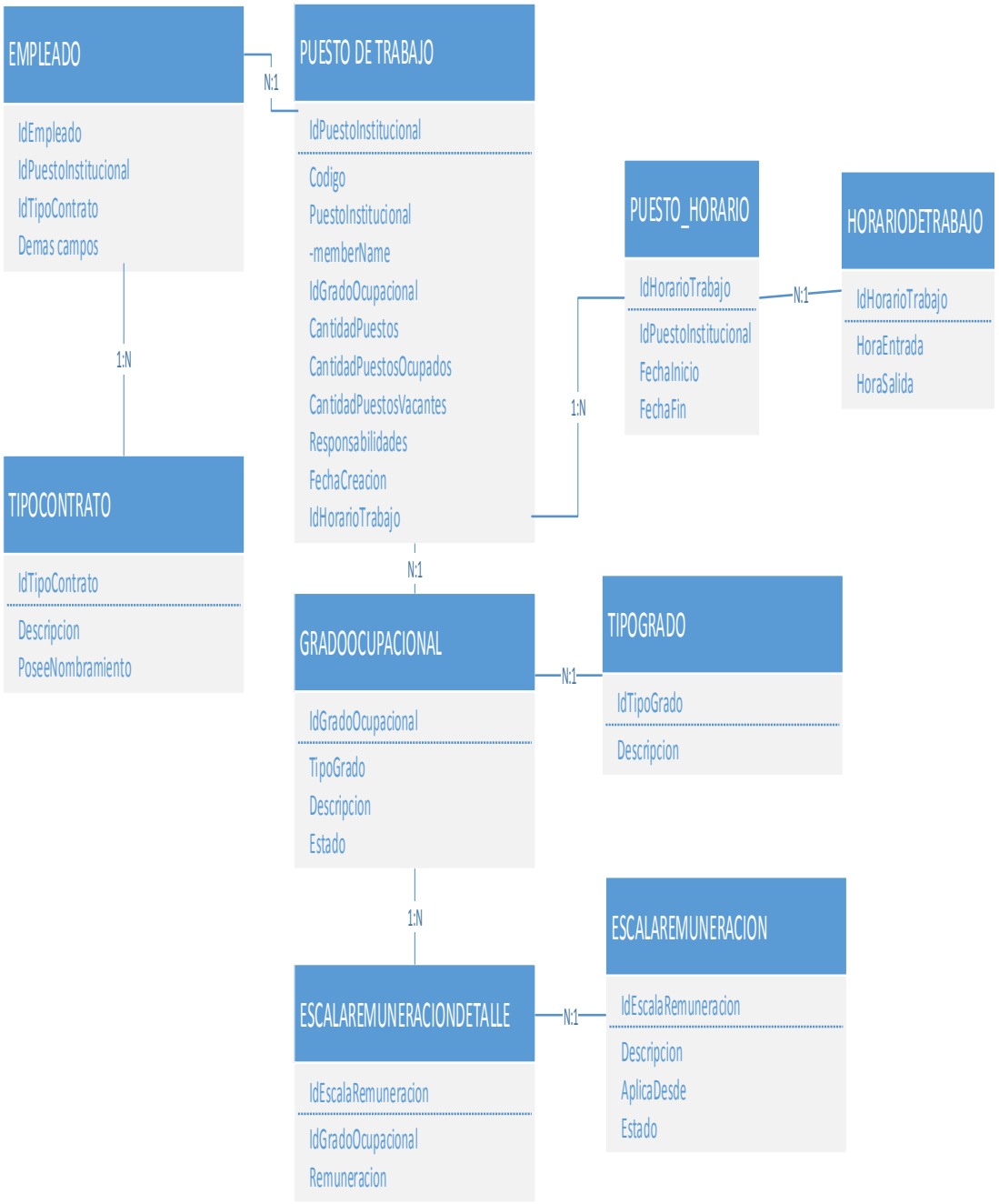


Figura IV. 28 Diagrama de Clases
Datos del Empleado con puesto de Trabajo
Fuente: Autora

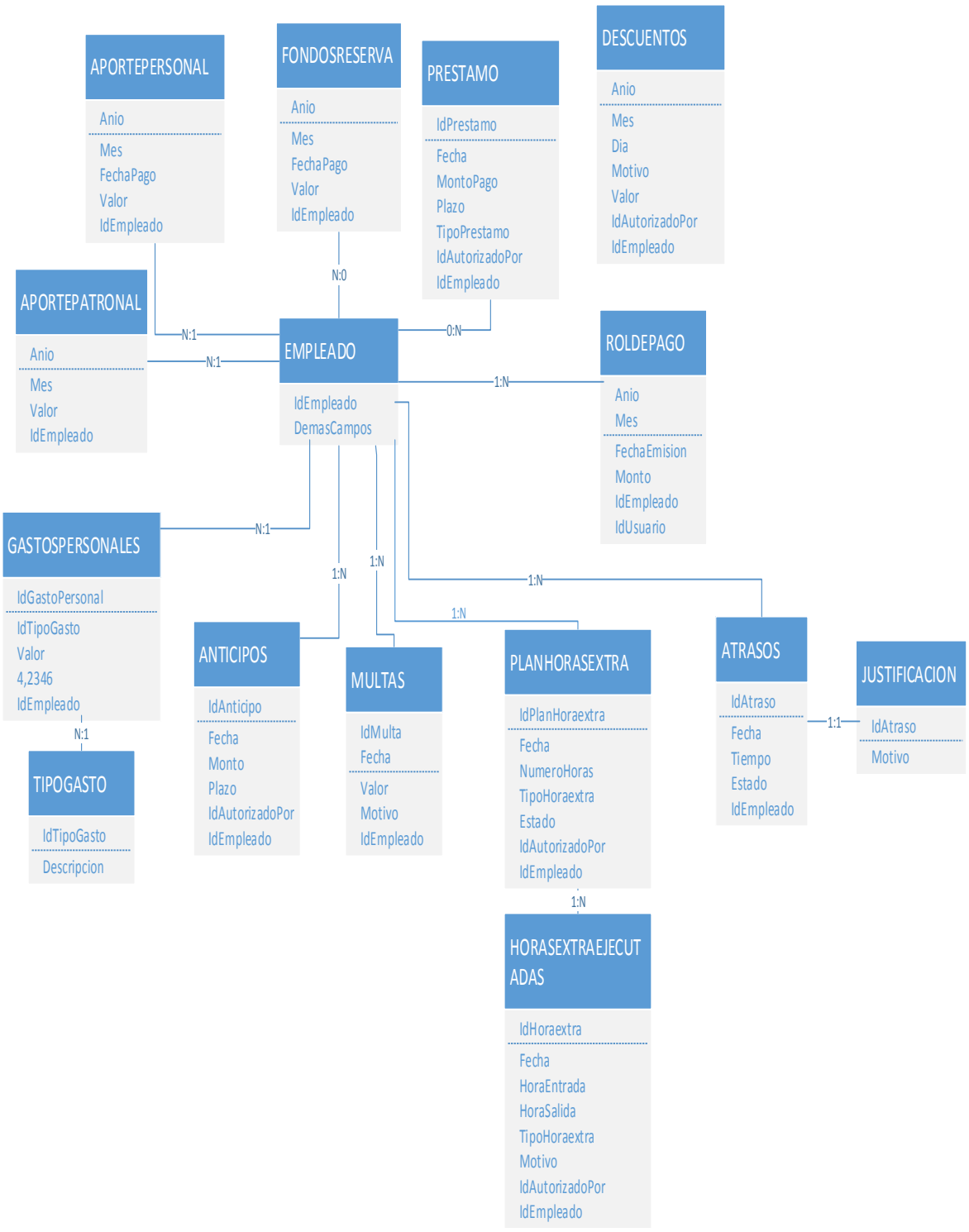
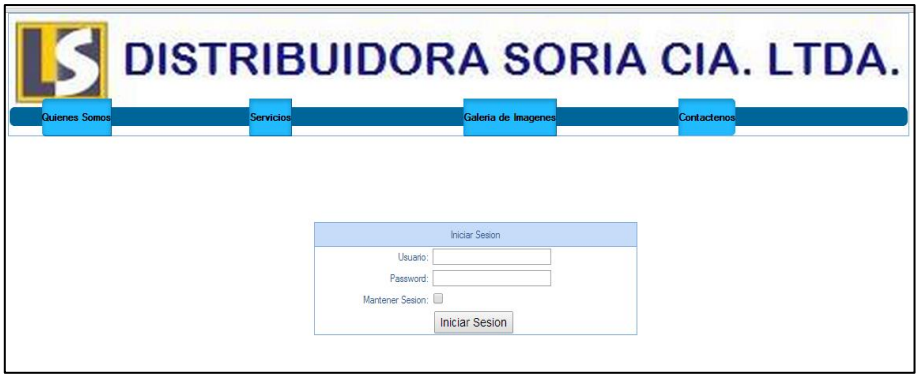


Figura IV. 29 Diagrama de Clases
Fuente: Autora

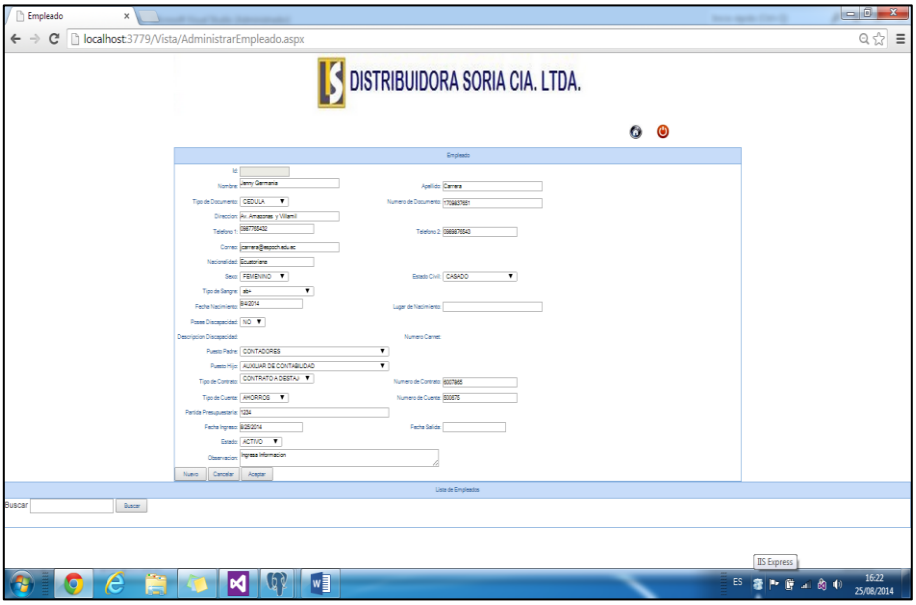
4.8.2.4 Diseño de Interfaces

- Autentificación de usuarios



- Interfaz principal del administrador

- Interfaz para el ingreso de los empleados



4.8.3 Diseño Físico

4.8.3.1 Diagrama de Actividades

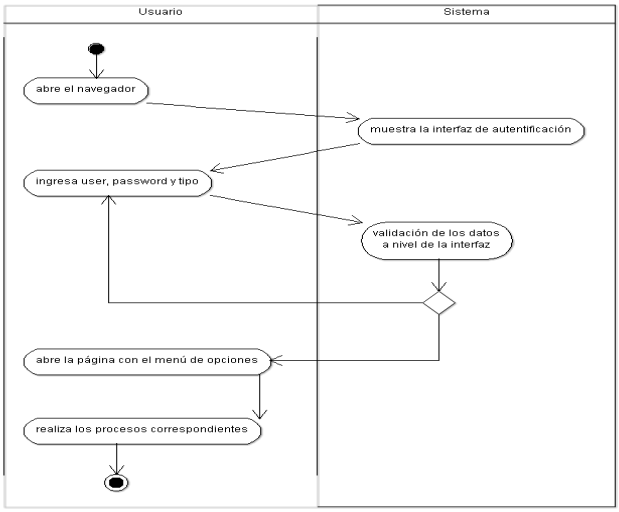


FIGURA IV. 30 Diagrama de Actividades

Fuente: Autora

4.8.3.2 Diagrama de Componentes

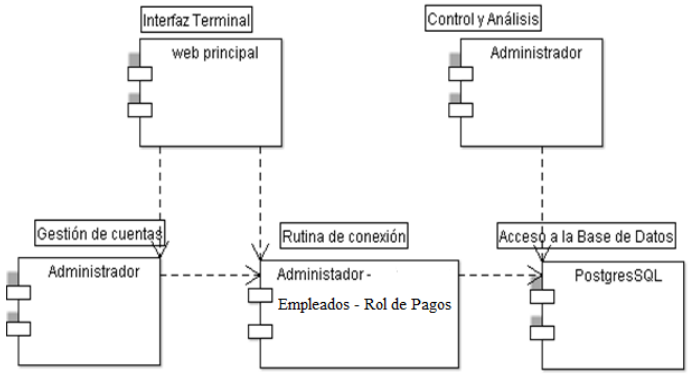


FIGURA IV. 31 Diagrama de Componentes

Fuente: Autora

4.8.3.3 Diagrama de Implementación

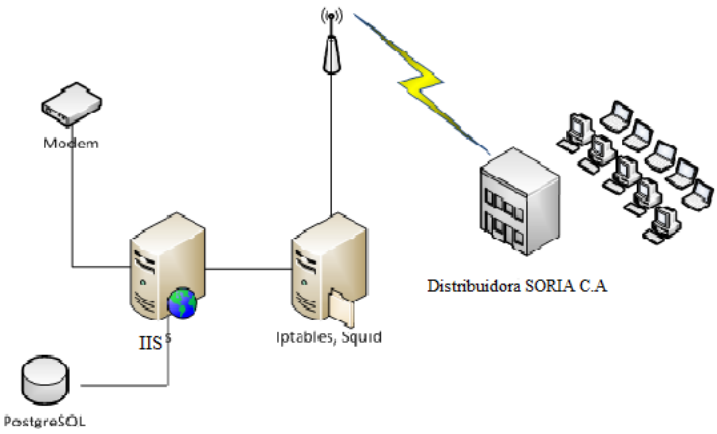


FIGURA IV. 38 Diagrama de Implementación

Fuente: Autora

4.8.3.4 Modelo Físico de la Base de Datos

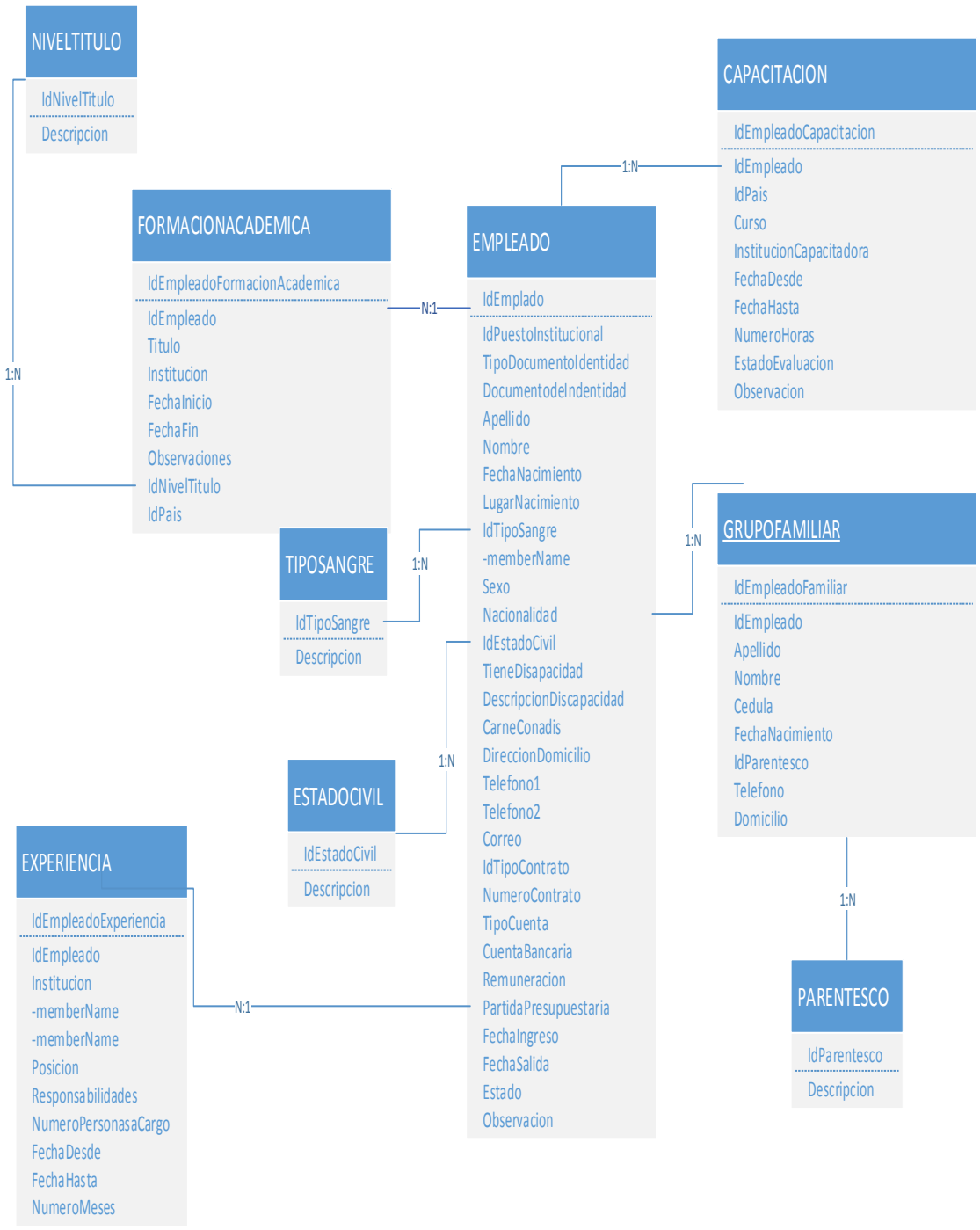


Figura IV. 32 Diagrama de clases
Datos del Empleado
Fuente: Autora

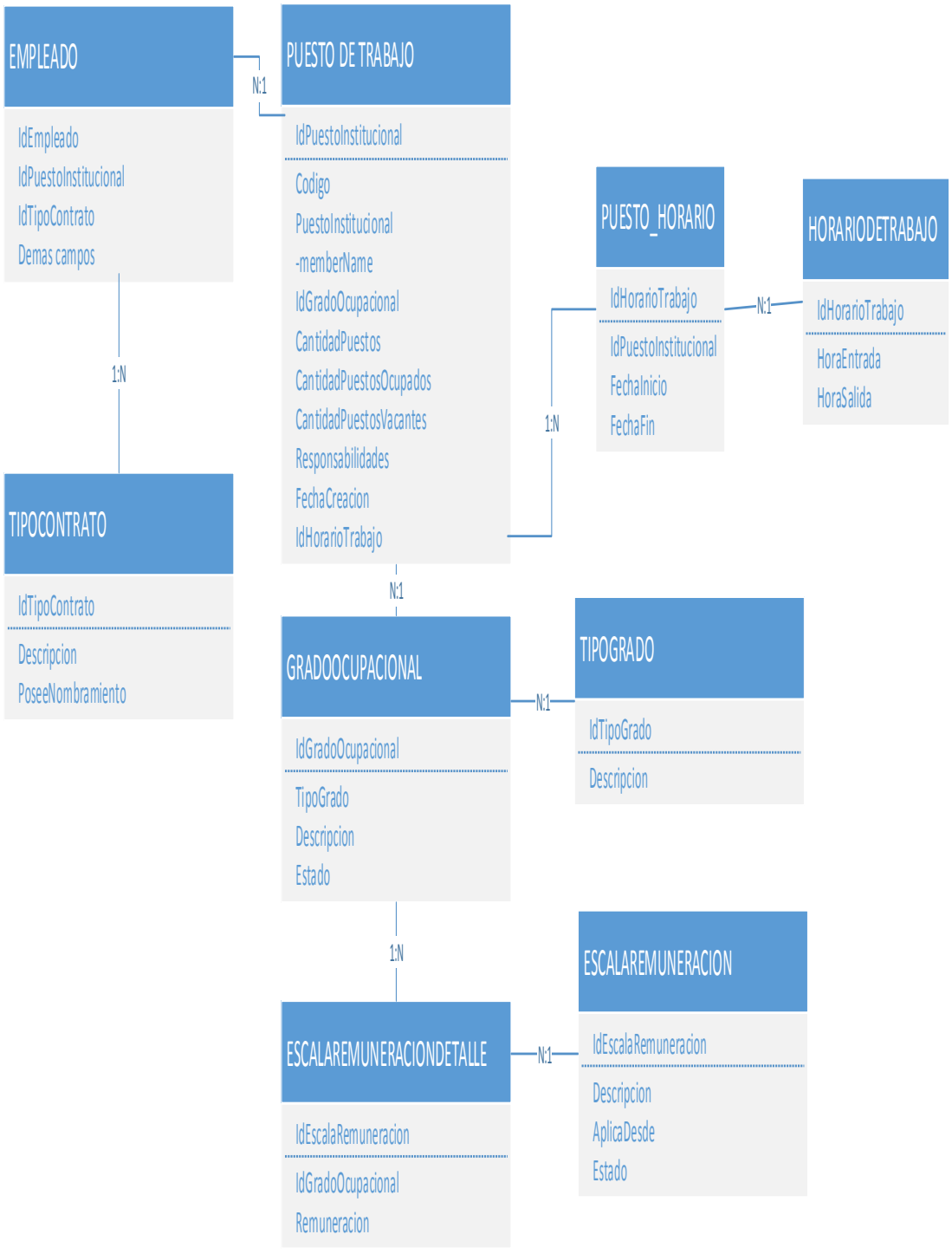


Figura IV. 40 Diagrama de Clases
Datos del Empleado con puesto de Trabajo
Fuente: **Autora**

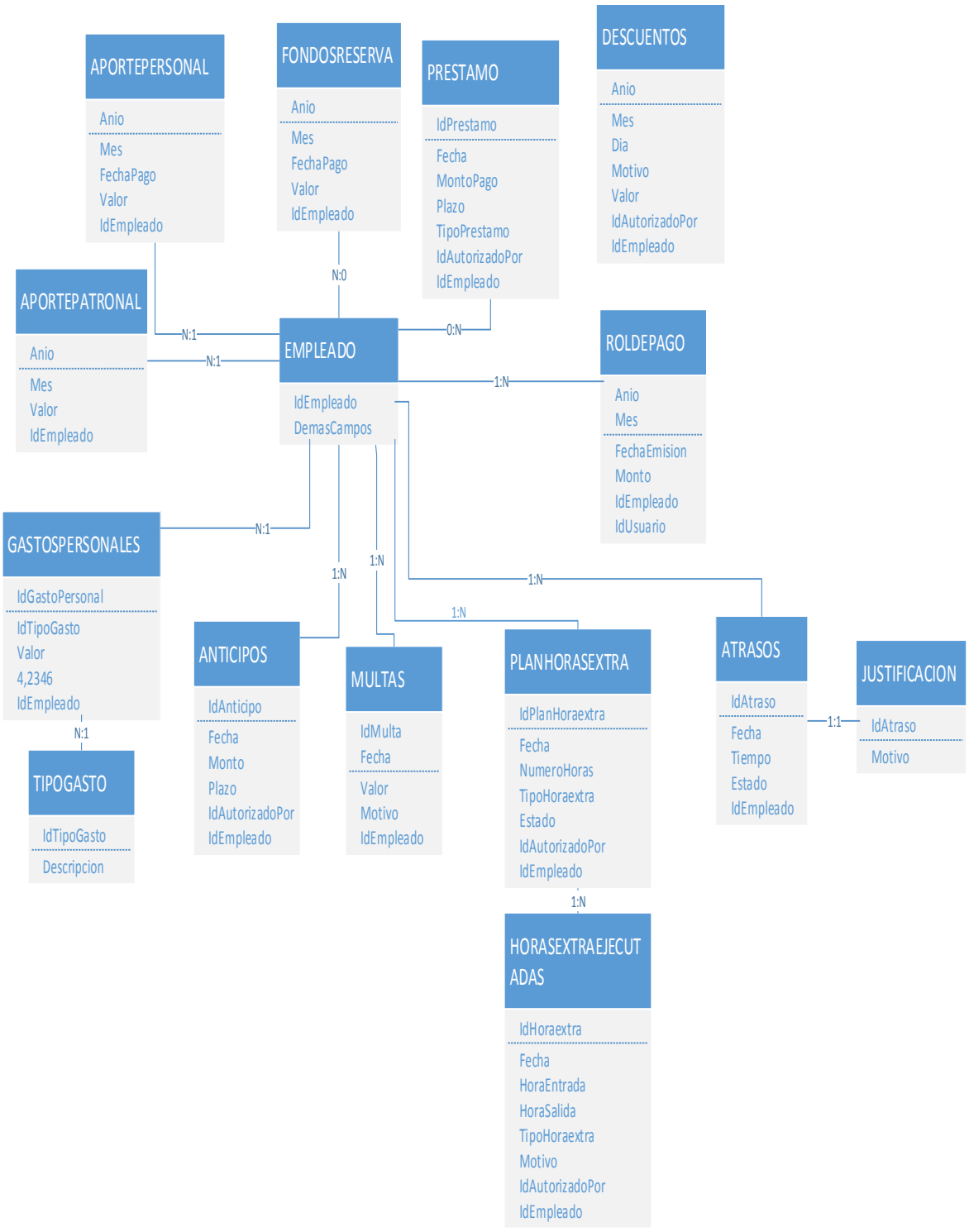


Figura IV.33 Diagrama de Clases
Fuente: Autora

4.9 Fase de Desarrollo

Las versiones permiten a los clientes tener una visión clara de la aplicación desarrollada

4.9.1 Nomenclatura y Estándares

A continuación se detalla:

TABLA IV. XLII Nomenclatura y Estándares

Archivo	**Extensión**
Programa en .net	fileName.aspx
Imágenes	Image.jpg
Clases	nameClase.cs

Fuente: Autora

4.9.2 Capa de Datos

4.9.2.1 Diccionario de Datos

TABLA IV. XLIII Diccionario de Datos

Nombre del Campo	**Tipo de Dato**	**Longitud**	**Clave Primaria**	**Calculado**
Tabla: Empresa				
IdEmpresa	Serial	-	X	No
Strnombre	character varying	40	-	No
Strdireccion	character varying	200	-	No

strTelefono1	character varying	10	-	No
strTelefono2	character varying	10	-	No
Tabla: Usuario				
IdUsuario	Serial	-	X	No
Strnombre	character varying	40	-	No
Strcuenta	character varying	30	-	No
Strclave	character varying	30	-	No
Introl	Integer	-	-	No
Intestado	Integer	-	-	No
Tabla: Tipo Concepto Gasto				
IdTipoConceptoGasto	Serial	-	X	No
Strdescripcion	character varying	200	-	No
Strorigen	character varying	8	-	No
Tabla: Comisión Servicio				
IdComisionServicio	Serial	-	X	No

Intnumerocomision	Integer	-	-	No
Strobjetivo	character varying	200	-	No
Stragenda	character varying	400	-	No
Strobservacion	character varying	200	-	No
Intestado	Integer	-	-	No
Intidcreadopor	Integer	-	-	No
Dtfecharegistro	Date	-	-	No
Tabla: País				
Idpais	Serial	-	X	No
Strnombre	character varying	50	-	No
Tabla: Atraso				
IdAtraso	Serial	-	X	No
Dtfecha	character varying	40	-	No
Inttiempo	Integer	-	-	No
Intestado	Integer	-	-	No

intIdEmpleado	Integer	-	-	No
Tabla: Multa				
IdMulta serial	Serial	-	X	No
Datefecha	Date	-	-	No
Fltvalor	Numeric	9,2	-	No
Motivo	character varying	200	-	No
intIdEmpleado	Integer	-	-	No
Tabla: Anticipo				
IdAnticipo	Serial	-	X	No
Datefecha	Date	-		No
Monto	Numeric	9,2	-	No
Intplazo	Integer	-	-	No
Intidautorizadopor	Integer	-	-	No
intIdEmpleado	Integer	-	-	No
Tabla: Gastos Personales				

IdGastoPersonal	Serial	-	X	No
Intidtipogasto	Integer	-	-	No
Fltvalor	Numeric	9,2	-	No
intIdEmpleado	Integer	-	-	No
Strdescripcion	character varying	200	-	No
Tabla: Descuento				
IdDescuento	Serial	-	X	No
Dtfecha	Date	-	-	No
Motivo	character varying	200	-	No
Fltvalor	Numeric	9,2	-	No
Intidautorizadopor	Integer	-	-	No
intIdempleado	Integer	-	-	No
Tabla: Préstamo				
IdPrestamo	Serial	-	X	No
Dtfecha	Date	-	-	No

Fltmontopago	Numeric	9,2	-	No
Intplazo	Integer	-	-	No
Inttipoprestamo	Integer	-	-	No
Intidautorizadopor	Integer	-	-	No
intIdempleado	Integer	-	-	No
Tabla: Grupo Familiar				
intid_grupo_familiar	Serial	-	X	No
Strnombre	character varying	40	-	No
Strapellido	character varying	40	-	No
strrelacion_parentesco	character varying	40	-	No
Intedad	Integer	-	-	No

4.9.2.2 Script de la Base de Datos

```
create table niveltitulo
(
 IdNivelTitulo serial not null,
descripcion character varying(200) not null,
constraint pk_niveltitulo primary key (IdNivelTitulo)
```

```
)
create table tiposangre
(
 IdTipoSangre serial not null,
descripcion character varying(10) not null,
constraint pk_tiposangre primary key (IdTipoSangre)
)
create table estadocivil
(
 IdEstadoCivil serial not null,
descripcion character varying(10) not null,
constraint pk_estadocivil primary key (IdEstadoCivil)
)
create table tipogrado
(
 IdTipoGrado serial not null,
descripcion character varying(200),
constraint pk_tipogrado primary key (IdTipoGrado)
)
create table gradoocupacional
(
 IdGradoOcupacional serial not null,
idtipogrado int,
descripcion character varying(200),
estado int,
constraint pk_gradoocupacional primary key(IdGradoOcupacional),
```

```
constraint fk_gradoocupacional_tipogrado foreign key (idtipogrado) references
tipogrado(IdTipoGrado)

 )

create table puestoinstitucional

(

  IdPuestoInstitucional serial not null,

  IdEmpresa int,

codigo character varying(32),

nombrepuestoinstitucional character varying(100),

  IdPuestoInstitucionalPadre int,

idgradoocupacional int,

cantidadpuestos int,

cantidadpuestosocupados int,

cantidadpuestosvacantes int,

responsabilidades character varying(200),

fechacreacion date,

constraint pk_puestoinstitucional primary key (IdPuestoInstitucional),

constraint fk_puestoinstitucional_gradoocupacional foreign key(idgradoocupacional)
references gradoocupacional (IdgradoOcupacional),

constraint fk_puestoinstitucional_puestoinstitucional foreign
key(IdPuestoInstitucionalPadre) references puestoinstitucional(IdPuestoInstitucional)

)

create table tipocontrato

(

 IdTipoContrato serial not null,

descripcion character varying(200),

poseenombramiento int,
```

```
constraint pk_tipocontrato primary key(IdTipoContrato)
)
create table empleado
(
 IdEmpleado serial not null,
idpuestoinstitucional int not null,
tipodocumentoidentidad int not null,
documentoidentidad character varying(15) not null,
apellido character varying(50) not null,
nombre character varying(50)not null,
fechanacimiento date,
lugarnacimiento character varying(100),
idtiposangre int,
sexo character varying(9),
nacionalidad character varying(30),
idestadocivil int,
tienediscapacidad int,
descripciondiscapacidad character varying(50),
carnetconadis character varying(20),
direcciondomicilio character varying(100),
 telefono1 character varying(10),
 telefono2 character varying(10),
correo character varying(50),
 IdTipoContrato int,
numerocontrato character varying(30),
tipocuenta character varying(10),
```

```
cuentabancaria character varying(20),

remuneracion numeric(9,2),

partidapresupuestaria character varying(20),

fechaingreso date,

fechasalida date,

estado character varying(8),

observacion character varying(200),

constraint pk_empleado primary key(IdEmpleado),

constraint fk_empleado_estadocivil foreign key(idestadocivil) references
estadocivil(IdEstadoCivil),

constraint fk_empleado_tiposangre foreign key(idtiposangre) references
tiposangre(IdtipoSangre),

constraint fk_empleado_tipocontrato foreign key(idtipocontrato) references
tipocontrato(IdTipoContrato),

constraint fk_empleado_puestoinstitucional foreign key(idpuestoinstitucional) references
puestoinstitucional(IdPuestoInstitucional)

)

create table horariotrabajo

(

 IdHorarioTrabajo serial not null,

horaentrada time without time zone,

horasalida time without time zone,

constraint pk_horariotrabajo primary key(IdHorarioTrabajo)

)

ALTER TABLE horariotrabajo add column horaentrada time without time zone

ALTER TABLE horariotrabajo add column horasalida time without time zone

create table puesto_horario

(
```

```
 IdHorarioTrabajo int not null,

 IdPuestoInstitucional int not null,

fechainicio date,

fechafin date,

constraint pk_puesto_horario primary key(IdHorarioTrabajo, IdPuestoInstitucional),

constraint fk_puesto_horario_puestoinstitucional foreign key(IdPuestoInstitucional)
references puestoinstitucional(IdPuestoInstitucional),

constraint fk_puesto_horario_horariotrabajo foreign key(IdHorarioTrabajo) references
horariotrabajo (IdHorarioTrabajo)

)

create table escalaremuneracion

(

 IdEscalaRemuneracion serial not null,

descripcion character varying(200),

aplicadesde date,

estado int,

constraint pk_escalaremuneracion primary key(IdEscalaRemuneracion)

)

create table escalaremuneraciondetalle

(

 IdEscalaRemuneracion int not null,

 IdGradoOcupacional int not null,

remuneracion numeric(9,2),

constraint pk_escalaremuneraciondetalle primary
key(IdEscalaRemuneracion,IdGradoOcupacional),

constraint fk_escalaremuneraciondetalle_gradoocupacional foreign key
(IdGradoOcupacional) references gradoocupacional(IdGradoOcupacional),
```

```
constraint fk_escalaremuneraciondetalle_escalaremuneracion foreign key
(IdEscalaremuneracion) references escalaremuneracion(IdEscalaRemuneracion)

)

create table formacionacademica

(

 IdEmpleadoFormacionAcademica serial not null,

IdEmpleado int not null,

titulo character varying(100)  ,

institucion character varying(100) ,

fechainicio date ,

fechafin date ,

observaciones character varying(100),

 Idniveltitulo int,

 IdPais int,

constraint pk_formacionacdemica primary key(IdEmpleadoFormacionAcademica),

constraint fk_formacionacademica_empleado foreign key(IdEmpleado) references
empleado(Idempleado),

constraint fk_formacionacademica_niveltitulo foreign key (Idniveltitulo) references
niveltitulo(IdNivelTitulo)

)

create table capacitacion

(

 IdEmpleadoCapacitacion serial not null,

 IdEmpleado int,

idpais int,

curso character varying(100),

institucioncapacitadora character varying(100),
```

```
fechadesde date,

fechahasta date,

numerohoras int,

estadoevaluacion character varying(8),

observacion character varying(200),

constraint pk_capacitacion primary key (IdEmpleadoCapacitacion),

constraint fk_capacitacion_empleado foreign key (IdEmpleado) references
empleado(IdEmpleado)

)

create table experiencia

(

 IdEmpleadoExperiencia serial not null,

 IdEmpleado int,

institucion character varying(100),

posicion character varying(100),

responsabilidades character varying(200),

numeropersonascargo int,

fechadesde date,

fechahasta date,

numeromeses int,

constraint pk_experiencia primary key (IdEmpleadoExperiencia),

constraint pk_experiencia_empleado foreign key (IdEmpleado) references
empleado(IdEmpleado)

)

create table parentesco

(

 IdParentesco serial not null,
```

```
descripcion character varying(20),
constraint pk_parentesco primary key (IdParentesco)
)
create table grupofamiliar
(
 IdEmpleadoFamiliar serial not null,
IdEmpleado int,
apellido character varying(50),
nombre character varying(50),
cedula character varying(10),
fechanacimiento date,
idparentesco int,
telefono character varying(10),
domicilio character varying(200),
constraint pk_grupofamiliar primary key(IdEmpleadoFamiliar),
constraint fk_grupofamiliar_empleado foreign key(IdEmpleado) references
empleado(IdEmpleado),
constraint fk_grupofamiliar_parentesco foreign key(idparentesco) references
parentesco(IdParentesco)
)
create table prestamo
(
 IdPrestamo serial not null,
fecha date,
montopago numeric(9,2),
plazo int,
tipoprestamo int,
```

```
idautorizadopor int,
 Idempleado int,
constraint pk_prestamo primary key (IdPrestamo),
constraint fk_prestamo_empleado foreign key (Idempleado) references empleado
(IdEmpleado)
)
create table descuento
(
 IdDescuento serial not null,
fecha date not null,
motivo character varying(200),
valor numeric(9,2),
idautorizadopor int,
 Idempleado int,
constraint pk_descuento primary key (IdDescuento),
constraint fk_descuento_empleado foreign key(Idempleado) references
empleado(IdEmpleado)
)
create table fondoreserva
(
anio int not null,
mes int not null,
fechapago date,
valor numeric(9,2),
 IdEmpleado int,
constraint pk_fondosreserva primary key(anio,mes),
```

```
constraint fk_fondosreserva_empleado foreign key (IdEmpleado) references
empleado(IdEmpleado)

)

create table aporteiess

(

anio int not null,

mes int not null,

fechapago date,

valor numeric(9,2),

 IdEmpleado int,

constraint pk_aporteiess primary key(anio,mes),

constraint fk_aporteiess_empleado foreign key(IdEmpleado) references
empleado(IdEmpleado)

)

create table aportepatronal

(

anio int not null,

mes int not null,

fechapago date,

valor numeric(9,2),

 IdEmpleado int,

constraint pk_aportepatronal primary key(anio,mes),

constraint fk_aportepatronal foreign key (IdEmpleado) references empleado(IdEmpleado)

)

create table tipogasto

(

 IdTipoGasto int,
```

```
descripcion character varying(200),
constraint pk_tipogasto primary key(IdTipoGasto)
)
create table gastospersonales
(
 IdGastoPersonal serial not null,
idtipogasto int,
valor numeric(9,2),
descripcion character varying(200),
 IdEmpleado int,
constraint pk_gastospersonales primary key(IdGastoPersonal),
constraint fk_gastospersonales_tipogasto foreign key(idtipogasto) references
tipogasto(IdTipoGasto),
constraint fk_gastospersonales_empleado foreign key(IdEmpleado) references
empleado(IdEmpleado)
)
create table anticipo
(
 IdAnticipo serial not null,
fecha date,
monto numeric(9,2),
plazo int,
idautorizadopor int,
 IdEmpleado int,
constraint pk_anticipo primary key(IdAnticipo),
constraint fk_anticipo_empleado foreign key(IdEmpleado) references
empleado(IdEmpleado)
```

```
)
create table multa
(
 IdMulta serial not null,
fecha date,
valor numeric(9,2),
motivo character varying(200),
 IdEmpleado int,
constraint pk_multa primary key (IdMulta),
constraint fk_multa_empleado foreign key(IdEmpleado) references empleado(IdEmpleado)
)
create table atraso
(
 IdAtraso serial not null,
fecha date,
tiempo int,
estado int,
 IdEmpleado int,
constraint pk_atraso primary key(IdAtraso),
constraint fk_atraso_empleado foreign key(IdEmpleado) references empleado(IdEmpleado)
)
create table justificacion
(
 IdAtraso int not null,
descripcion character varying(200),
constraint pk_justificacion primary key(IdAtraso),
```

```
constraint fk_justificacion_atraso foreign key(IdAtraso) references atraso(IdAtraso)
 )
create table planhorasextra
 (
  IdPlanHoraExtra serial not null,
fecha date,
numerohoras int,
tipohoraextra int,
estado int,
idautorizadopor int,
  IdEmpleado int,
constraint pk_planhorasextra primary key(IdPlanHoraExtra),
constraint fk_planhorasextra_empledo foreign key(IdEmpleado) references
empleado(IdEmpleado)
 )
create table horasextraejecutadas
 (
  IdHoraExtraEjecutada serial not null,
fecha date,
horaentrada time without time zone,
horasalida time without time zone,
tipohoraextra int,
motivo character varying(200),
idautorizadopor int,
  IdPlanHoraExtra int,
constraint pk_horasextraejecutadas primary key (IdHoraExtraEjecutada),
```

```
constraint fk_horasextraejecutadas_planhoraextra foreign key(IdPlanHoraExtra) references
planhorasextra(IdPLanHoraExtra)

)

ALTER TABLE horasextraejecutadas add column horaentrada time without time zone

ALTER TABLE horasextraejecutadas add column horasalida time without time zone

create table pais

(

  Idpais serial not null,

nombre character varying(50),

constraint pk_pais primary key(IdPais)

)

create table comisionservicio

(

  IdComisionServicio serial not null,

numerocomision int,

objetivo character varying(200),

agenda character varying(400),

observacion character varying(200),

estado int,

idcreadopor int,

fecharegistro date,

constraint pk_comisionservicio primary key(IdComisionServicio)

 )

create table comisionserviciointegrante

 (

   IdComisionServicioIntegrante serial not null,
```

```
idcomisionservicio int,
  IdEmpleado int,
idpuestoinstitucional int,
nivelviatico int,
fechasalidaplanificada date,
fecharetornoplanificada date,
diascomisionplanificada int,
horascomisionplanificada int,
fechasalidareal date,
fecharetornoreal date,
diascomisionreal int,
horascomisionreal int,
essecretario int,
constraint pk_comisionserviciointegrante primary key(IdComisionServicioIntegrante),
constraint fk_comisionserviciointegrante_comisionservicio foreign key
(idcomisionservicio) references comisionservicio(IdComisionServicio),
constraint fk_comisionserviciointegrante_empleado foreign key (IdEmpleado) references
empleado(IdEmpleado),
constraint fk_comisionserviciointegrante_puestoinstitucional foreign
key(idpuestoinstitucional) references  puestoinstitucional(IdPuestoInstitucional)
 )
create table comisionserviciodestino
 (
  IdComisionServicioDestino serial not null,
idcomisionservicio int,
codigoadministrativo character varying(8),
zonaviatico character  varying(8),
```

```
fecharetorno date,

constraint pk_comisionserviciodestino primary key(IdComisionServicioDestino),

constraint fk_comisionserviciodestino_comisionservicio foreign key(idcomisionservicio)
references comisionservicio(IdComisionServicio)

 )

create table comisionserviciointegrantedestino

 (

  IdComisionServicioIntegranteDestino serial not null,

idcomisionserviciointegrante int,

tiporegistro character varying(8),

fecha date,

codigoadministrativo character varying(8),

zonaviatico character varying(8),

esfecharetorno date,

constraint pk_comisionserviciointegrantedestino primary
key(IdComisionServicioIntegranteDestino),

constraint fk_comisionserviciointegrantedestino_comisionserviciointegrante foreign
key(idcomisionserviciointegrante) references
comisionserviciointegrante(IdComisionServicioIntegrante)

)

create table solicitudviatico

 (

  IdSolicitudViatico serial not null,

idcomisionserviciointegrante int,

numerosolicitud character varying(10),

fecharegistro date,

requierevehiculoempresa int,

requierepasajeaereo int,
```

```
requieregastosvarios int,

requiereaprobacionferiado int,

requierevehiculoparticular int,

observacion character varying(200),

estaaprobadoanticipo int,

valoranticiposugerido numeric(9,2),

valoranticipo numeric(9,2),

constraint pk_solicitudviatico primary key(IdSolicitudViatico),

constraint fk_solicitudviatico_comisionserviciointegrante foreign
key(idcomisionserviciointegrante) references
comisionserviciointegrante(IdComisionServicioIntegrante)

 )

create table informecomisionservicio

 (

  IdSolicitudViatico int not null,

fechapresentacion date,

estado character varying(8),

requiereaprobacionferiado int,

constraint pk_informecomisionservicio primary key(IdSolicitudViatico),

constraint fk_informecomisionservicio foreign key(IdSolicitudViatico) references
solicitudviatico(IdSolicitudViatico)

 )

create table solicitudviaticoautorizacion

 (

  IdSolicitudViatico int not null,

tipoautorizacion character varying(8) not null,

IdEmpleado int,
```

```
idpuestoinstitucional int,

fechaautorizacion date,

constraint pk_solicitudviaticoautorizacion primary
key(IdSolicitudViatico,tipoautorizacion),

constraint fk_solicitudviaticoautorizacion_solicitudviatico foreign key(IdSolicitudViatico)
references solicitudviatico(IdSolicitudViatico),

constraint fk_solicitudviaticoautorizacion foreign key(IdEmpleado) references empleado
(IdEmpleado),

constraint fk_solicitudviaticoautorizacion_puestointstitucional foreign key
(idpuestoinstitucional) references puestoinstitucional(IdPuestoInstitucional)

)

create table liquidacionviatico

  (

   IdSolicitudviatico int not null,

numeroliquidacion character varying(8),

requierepasajeaereo int,

comentariorequierepasajeaereo character varying(200),

requierevehiculoparticular int,

comentariorequierevehiculoparticular character varying(200),

requierevehiculoempresa int,

comentariorequierevehiculoempresa character varying(200),

valortotal numeric(9,2),

valoranticipo numeric(9,2),

saldofavorempresa numeric(9,2),

saldofavorempleado numeric(9,2),

solicitudpagofecha date,

solicitudpagoreferencia character varying(32),

solicitudpagovalor numeric(9,2),
```

```
observacion character varying(200),
elaboradopor character varying(50),
elaboradoporcargo character varying(50),
aprobadopor character varying(50),
aprobadoporcargo character varying(50),
estado int,
constraint pk_liquidacionviatico primary key(IdSolicitudViatico),
constraint fk_liquidacionviatico_solicitudviatico foreign key(IdSolicitudViatico) references
solicitudviatico(IdSolicitudViatico)
 )
create table tipoconceptogasto
(
 IdTipoConceptoGasto serial not null,
descripcion character varying(200),
origen character varying(8),
constraint pk_tipoconceptogasto primary key(IdTipoConceptoGasto)
)
create table zonaviatico
 (
zonaviatico character varying(8),
porcentajealimentacion numeric(9,2),
porcentajesubsistencia numeric(9,2),
porcentajeanticipo numeric(9,2),
constraint pk_zonaviatico primary key(zonaviatico)
 )
create table viaticovalor
```

```
(
zonaviatico character varying(8) not null,
nivelviatico int not null,
montoviatico numeric(9,2),
constraint pk_viaticovalor primary key(zonaviatico,nivelviatico),
constraint fk_viaticovalor_zonaviatico foreign key(zonaviatico) references
zonaviatico(zonaviatico)
)
create table liquidacionviaticocalculo
(
IdLiquidacionViaticoCalculo serial not null,
   IdSolicitudViatico int,
tiporegistro character varying(8),
idtipoconceptogasto int,
fecha date,
zonaviatico character varying(8),
nivelviatico int,
montoviatico numeric(9,2),
porcentajealimentacion numeric(9,2),
porcentajesubsistencia numeric(9,2),
valor numeric(9,2),
porcentajeanticipo numeric(9,2),
constraint pk_liquidacionviaticocalculo primary key(IdLiquidacionViaticoCalculo),
constraint fk_liquidacionviaticocalculo_tipoconceptogasto foreign
key(idtipoconceptogasto)references tipoconceptogasto(IdTipoConceptoGasto),
constraint fk_liquidacionviaticocalculo_solicitudviatico foreign key(IdSolicitudViatico)
references liquidacionviatico(IdSolicitudViatico),
```

```
constraint fk_liquidacionviaticocalculo_viaticovalor foreign key(zonaviatico,nivelviatico)
references viaticovalor (zonaviatico,nivelviatico)

 )

create table liquidacionviaticoconcepto

(

  IdLiquidacionViaticoConcepto serial not null,

  IdSolicitudViatico int,

idtipoconceptogasto int,

fecha date,

comentario character varying(200),

cantidad numeric(9,2),

valorsugerido numeric(9,2),

valor numeric(9,2),

motivocambiovalor character varying(200),

constraint pk_liquidacionviaticoncepto primary key(IdLiquidacionViaticoConcepto),

constraint fk_liquidacionviaticoconcepto_solicitudvaitico foreign key(IdSolicitudViatico)
references liquidacionviatico(IdSolicitudViatico),

constraint fk_liquidacionviaticoconcepto_tipoconceptogasto foreign
key(idtipoconceptogasto) references tipoconceptogasto(IdTipoConceptoGasto)

)

create table decimocuarto

(

anio int not null,

mes int not null,

  IdEmpleado int,

fechapago date,

numeromeses int,
```

```
valorpago numeric(9,2),

constraint pk_decimocuarto primary key(anio),

constraint fk_decimocuarto_empleado foreign key(IdEmpleado) references empleado
(IdEmpleado)

)

create table decimotercero

(

anio int not null,

mes int not null,

IdEmpleado int,

fechapago date,

numeromeses int,

valorpago numeric(9,2),

constraint pk_decimotercero primary key(anio),

constraint fk_decimotercero_empleado foreign key(IdEmpleado) references empleado
(IdEmpleado)

)

create table rolpagos

(

anio int not null,

mes int not null,

fechaemision date,

monto numeric(9,2),

  IdEmpleado int,

idemitidopor int,

constraint pk_rolpagos primary key(anio,mes),
```

```
constraint fk_rolpagos_empleado foreign key(IdEmpleado) references
empleado(IdEmpleado)

)

ALTER TABLE horariotrabajo add column horaentrada timestamp

ALTER TABLE horariotrabajo add column horasalida timestamp

create table usuario

(

 IdUsuario serial not null,

nombre character varying(100) not null,

cuenta character varying(30) not null,

clave character varying(30) not null,

rol int,

estado int,

constraint pk_usuario primary key (IdUsuario)

)

create table empresa

(

 IdEmpresa serial not null,

nombre character varying(100),

direccion character varying(200),

telefono1 character varying(10),

telefono2 character varying(10),

constraint pk_empresa primary key(IdEmpresa)

)
```

4.9.2.3 Implementación de la Base de Datos

Estructura objetos de la base de datos

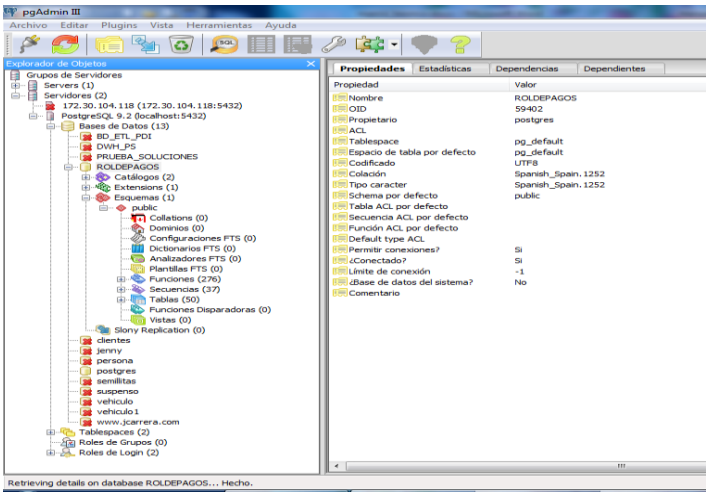


FIGURA IV. 34 Objetos de la Base de Datos

Fuente: Autora

Estructura de las tablas de la base de datos

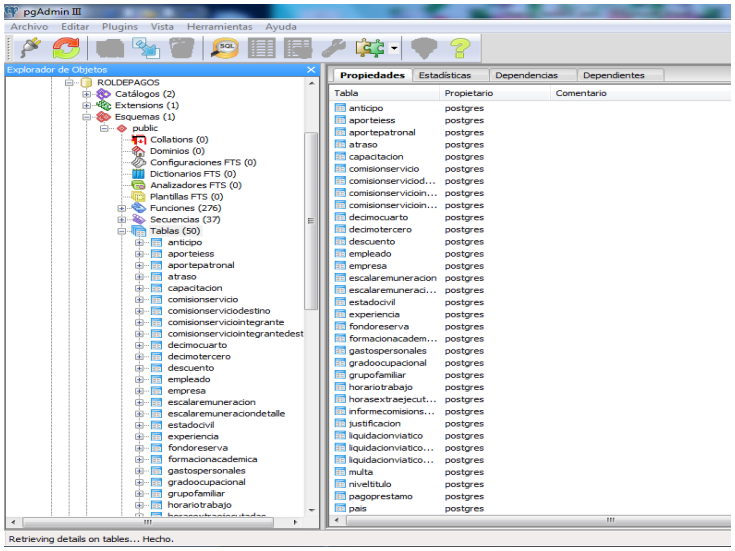


FIGURA IV. 43 Tablas de la Base de Datos

Fuente: Autora

Estructura de una Tabla

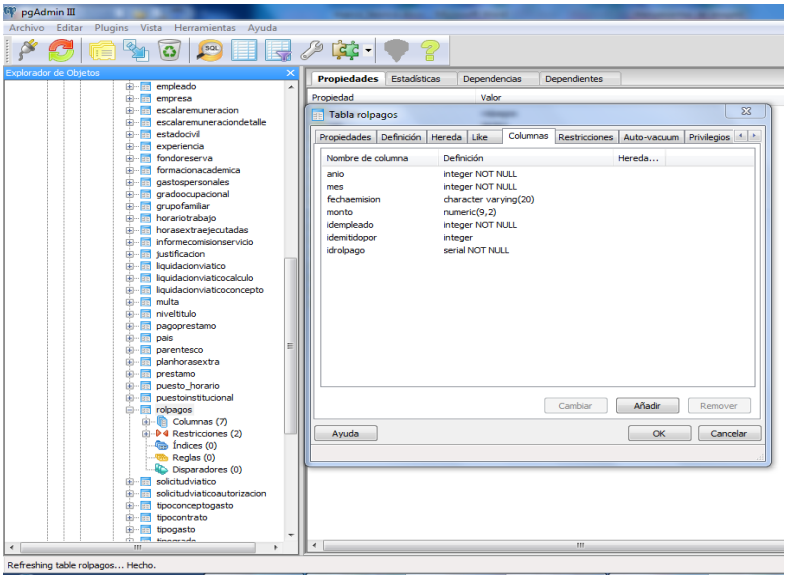


FIGURA IV. 35 Estructura de una Tabla

Fuente: Autora

4.10 Fase de Estabilización

Esta fase permite validar los requerimientos de la solución implementada. El desarrollo de la solución del proyecto ha sido completado y realizadas las pruebas correspondientes

Documentación

Para facilitar la aplicación, su uso y manipulación se presenta los siguientes manuales:

- Manual Técnico
- Manual del Usuario

CONCLUSIONES

- Con esta investigación se logró estudiar todo lo referente a los patrones de diseño MVC y MVP para tener mejores conocimientos de cada uno de ellos, pero en cuanto a información hubo más acerca del MVC ya que este es el más utilizado por los desarrolladores en la actualidad, por otro lado hubo muy poca información acerca de MVP ya que este no está siendo utilizado actualmente.
- Se ha determinado los parámetros Compatibilidad, Recursos, y Desarrollo, para la comparación entre los patrones MVC y MVP para de esta manera saber cuál es el más productivo a la hora de desarrollar una aplicación web.
- Se ha evaluado y se ha determinado que el patrón de diseño MVP crea sistemas web en menos tiempo y empleando menos recursos que el desarrollo de un sistema creado con el patrón de diseño MVC.

- Con esta investigación se logró desarrollar una aplicación web con el patrón de diseño MVP, el cual alcanzo un 100 % en productividad por lo que se determina que es el patrón de diseño más adecuado para desarrollar la aplicación web empresarial; por otro lado MVC obtuvo una calificación de 76,19% lo que indica que no es un patrón de diseño que proporcione las facilidades en el momento de desarrollar una aplicación web.

RECOMENDACIONES

- Se sugiere que se impartan más conocimiento en las aulas, acerca del patrón MVP, ya que este es el más productivo a la hora de desarrollar una aplicación web y sin embargo pocos desarrolladores lo utilizan en la actualidad.
- Se recomienda que al medir la productividad del software se debe establecer adecuadamente que parte del proceso de la ingeniería es la que se va a evaluar y en base a eso buscar estándares que permitan logra este fin.
- Los parámetros a ser utilizados en el estudio comparativo deben ser seleccionados cuidadosamente, con el fin de que estén relacionados al ámbito de aplicación y que permitan la comprobación de la hipótesis de forma efectiva.
- Se recomienda el patrón de diseño MVP a la hora de desarrollar aplicaciones web ya que se puede utilizar para el desarrollo de cualquier proyecto, en especial en aquellos que manejan gran cantidad de datos y transacciones complejas.

RESUMEN

Se realizó un estudio comparativo de los patrones de diseño MVP y MVC, para el desarrollo de un sistema web y elaborar Nómina de Empleados y Rol de Pagos denominado SISNEROLPA en la Distribuidora SORIA C.A (Ambato), con la finalidad de demostrar que la productividad en el desarrollo, mediante el uso de patrones de diseño, mejora notablemente.

Se definieron parámetros de productividad tales como: Compatibilidad, Recursos y Desarrollo. Una vez obtenido los resultados de la comparación de los mismos, en cuanto a Compatibilidad, observamos que tanto MVC como MVP obtuvieron un 100%, mientras que en Recursos, MVC alcanzo un 76,19 % y MVP obtuvo 100% y, en Desarrollo MVC alcanzo un 76,19 % y MVP obtuvo 100%.

Para la demostración de la hipótesis se aplicó el estadístico CHI cuadrado con 2 grados de libertad y un nivel de confianza de 0,05 obteniéndose un valor $X_T^2 = 5{,}991$; mientras que el valor calculado en esta investigación es $X_C^2 = 18{,}851$ el cual es superior y cae en la zona de rechazo de la H_0 con lo que podemos concluir que se rechaza la hipótesis nula y se acepta la hipótesis de la investigación.

Se determinó que MVP crea sistemas web en menos tiempo optimizando recursos, frente al desarrollo de un sistema creado en MVC. Recomendándose la implementación del sistema SISNEROLPA desarrollado para una mejor seguridad de la información de la empresa.

Palabras Claves: /ESTUDIO COMPARATIVO // PATRONES DE DISEÑO WEB // DESARROLLO DE APLICACIONES WEB/

SUMARY

A comparative study of design patterns MVP (Model View Presenter) and MVC (Model View Controller) was performed, for developing a web system and develop Payroll Payments and Role of Employees called SISNEROLPA in SORIA C.A Delivery (Ambato), in order to show that the development productivity through the use of design patterns improves significantly.

It was defined productivity parameters such as: Support, Resources and Development. After obtaining the results of the comparison between them in terms of compatibility, they note that both MVC and MVP scored 100% while in Recourse, reached a 76.19% MVC and MVP scored 100% and reached Developing 76.19% and 100% MVP obtained.

To demonstrate the hypothesis CHI-square static with 2 degrees of freedom and a confidence level of 0.05 was applied yielding a value $X_T^2 = 5{,}991$; while the value calculated in this study is $X_C^2 = 18{,}851$ which exceeds and falls into the rejection region of H_0 with what can conclude that the null hypothesis and it is rejected and the research hypothesis is accepted.

MVP was determined that creates web optimizing system resources in less time compared to the development of a system in MVC creator. We recommend the implementation of SISNEROLPA system developed for better information Security Company.

GLOSARIO

Aplicación web: Aplicación informática que los usuarios utilizan accediendo a un servidor web a través de Internet o de una Intranet.

Arquitectura: Representación abstracta de los componentes de un sistema y su comportamiento

Base de Datos: son recursos que recopilan todo tipo de información, para atender las necesidades de un amplio grupo de usuarios. Su tipología es variada y se caracterizan por una alta estructuración y estandarización de la información.

Controlador: Es el elemento más abstracto. Recibe, trata y responde los eventos enviados por el usuario o por la propia aplicación. Interactúa tanto con el modelo como con la vista.

Framework Web: Se define como una estructura definida, reusable en el que sus componentes facilitan la creación de aplicaciones web.

HTTP: es el método más común de intercambio de información en la world wide web, el método mediante el cual se transfieren las páginas web a un ordenador.

Modelo: Es la representación de la información en el sistema. Trabaja junto a la vista para mostrar la información al usuario y es accedida por el controlador para añadir, eliminar, consultar o actualizar datos.

Metodología: hace referencia al conjunto de procedimientos racionales utilizados para alcanzar una gama de objetivos que rigen una investigación científica.

MVP: Es una interfaz de usuario patrón de diseño de ingeniería para facilitar las pruebas unitarias automatizadas y mejorar la separación de las preocupaciones en la lógica de presentación.

MVC: Patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

Objeto: Entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad (métodos). Se corresponde con los objetos reales del mundo que nos rodea, o a objetos internos del sistema (del programa). Es una instancia a una clase.

Patrón de diseño: Un patrón de diseño es una solución a un problema de diseño no trivial que es efectiva (ya se resolvió el problema satisfactoriamente en ocasiones anteriores) y reusable (se puede aplicar a diferentes problemas de diseño en distintas circunstancias).

Patrón: es un tipo de tema de sucesos u objetos recurrentes, como por ejemplo grecas, a veces referidos como ornamentos de un conjunto de objetos.

Productividad: se refiere al manejo de recursos para llegar a la realización óptima de una meta; ya sea en términos de producto=ganancia o esfuerzo=recompensa.

Vista: Es la representación del modelo en un formato adecuado para que el usuario pueda interactuar con él, casi siempre es la interfaz de usuario.

BIBLIOGRAFÍA

1. **Alexander, Christopher;** experiencias paralelas y los patrones de diseño. [En línea] Inglaterra, 1977. [Citado el: 8 de Febrero de 2014.]

http://www.usolab.com/wl/2007/06/experiencias-paralelas-christo-1.php.

2. **Blanco, Carlos;** patrones de diseño, ingeniería del software I . [En línea] [Citado el: 12 de Marzo de 2014.]

http://ocw.unican.es/ensenanzas-tecnicas/ingenieria-del-software-i/materiales-de-clase-1/is1_t06_Patrones.pdf.

3. **Baumann, Ignacio;** administración de la productividad en las organizaciones. [En línea] Mc Graw Hil, 2007. [Citado el: 8 de Febrero de 2014.] http://blogs.southworks.net/ibaumann/category/patrones-practicas/.

4. **Estudiantes de la Universidad Carlos III; p**atrón de arquitectura Modelo Vista Controlador (MVC). [En línea] [Citado el: 18 de Marzo de 2014.] http://www.lab.inf.uc3m.es/~a0080802/RAI/mvc.html..

5. **Gatuneg;** Modelo-Vista-Presentador (MVP). [En línea] [Citado el: 6 de Abril de 2014.] http://gatuneg.wikispaces.com/Modelo+vista+Presentador.

6. **Gamma, E; Helm, R; Johnson, R; Vlissides, J; Patterns, Design;** taller de patrones de diseño. [En línea] Wesley Professional Computing Series, 1995. [Citado el: 5 de Marzo de 2014.] http://webdiis.unizar.es/~jmerse/IS-2/TeoriaPatronesV2_2.pdf. ISBN: 0-201-63361-2..

7. **García, Yupanqui; Jasper, Glen;** diseño de sistemas. [En línea] [Citado el: 25 de Abril de 2014.]

http://es.scribd.com/doc/3930805/Patrones-de-Diseno.

8. **Helm, Erik; Johnson, Richard; Vlissides, Ralph;** patrones de diseño. [En línea] 1998. [Citado el: 20 de Febrero de 2014.]

http://www.slideshare.net/gueste39de6/patrones-de-diseo.

9. **Ines;** productividad en el desarrollo de software. [En línea] [Citado el: 25 de Marzo de 2014.] http://www.ines.org.es/node/1485.

10. **Martinez, Ivette;** diseño con patrones. [En línea] 2011. [Citado el: 20 de Febrero de 2014.] http://www.slideshare.net/EdwinRomanCastrillon/diseo-de-patrones.

11. **Miller, Jeremy;** diferencias entre los patrones de diseño MVC y MVP. [En línea] 2007. [Citado el: 16 de Abril de 2014.] http://codebetter.com/blogs/Jeremy.Miller/archive/2007/10/31/Development-Trivial-Pursuit-the-Difference-Between-MVC-and-the-different-Flavors-of-MVP.aspx.

12. **Microsoft;** ¿Qué es un patrón de diseño? [En línea] [Citado el: 22 de Febrero de 2014.] http://msdn.microsoft.com/es-es/library/bb972240.aspx.

13. **Microsoft;** patrones y antipatrones. [En línea] [Citado el: 10 de Marzo de 2014.] http://msdn.microsoft.com/es-es/library/bb972242.aspx.

14. **Pavón Mestras, Juan; e**l patrón Modelo-Vista-Controlador (MVC). [En línea] Universidad Complutense Madrid, 2008. [Citado el: 3 de Marzo de 2014.] http://www.fdi.ucm.es/profesor/jpavon/poo/2.14.mvc.pdf.

15. **Tore; Aldo, Horacio;** patrón de diseño modelo-vista-presentador. [En línea] [Citado el: 12 de Abril de 2014.]

http://www.monografias.com/trabajos93/mvp-model-view-presenter/mvp-model-view-presenter.shtml.

16. **Wikipedia;** Modelo Vista Controlador . [En línea] [Citado el: 11 de Marzo de 2014.] http://es.wikipedia.org/wiki/Modelo_Vista_Controlador.

17. **Wikipedia;** ptrones de diseño. [En línea] [Citado el: 12 de Febrero de 2014.] http://es.wikipedia.org/wiki/Patr%C3%B3n_de_dise%C3%B1o.

18. **Wikipedia;** patrones de diseño introducción. [En línea] [Citado el: 12 de Febrero de 2014.] http://es.wikipedia.org/wiki/Patr%C3%B3n_de_dise%C3%B1o.

19. **Wikipedia;** lenguaje de patrón. [En línea] [Citado el: 2014 de Marzo de 2014.] http://es.wikipedia.org/wiki/Lenguaje_de_patr%C3%B3n#Origen.

ANEXOS

ANEXOS 1

MANUAL DE USUARIO

1. INGRESO AL SISTEMA

INICIAR SESIÓN

1. Digite su cuenta
2. Digite la clave que fue creada para su acceso al sistema.
3. Dar clic en el botón Iniciar Sesión
4. Si fueron correctos los datos ingresados se desplegara página con las diferentes operaciones que están de acuerdo con su perfil de usuario.

MÓDULOS DATOS GENERALES

Dentro de este módulo encontramos:

- Datos de la Empresa.
- Tipos de Grado Ocupacional
- Grado Ocupacional
- Tipo de Contrato
- Escala Remuneración
- Escala Remuneración Detalle
- Valores

1. Damos clic en cualquiera de los ítems, en este caso como ejemplo seleccionaremos Empresa.

2. Posterior se despliega el listado con el registro que ha sido ingresado.

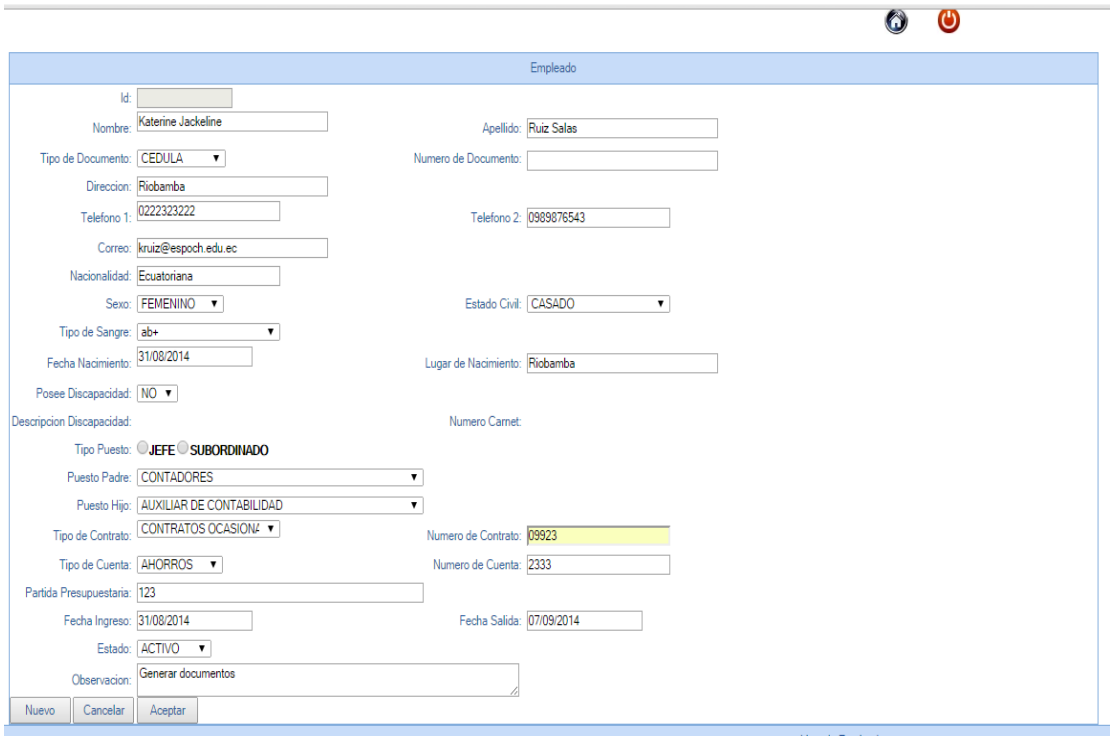
CREACIÓN DE EMPLEADO

Los datos del empleado están conformados por:

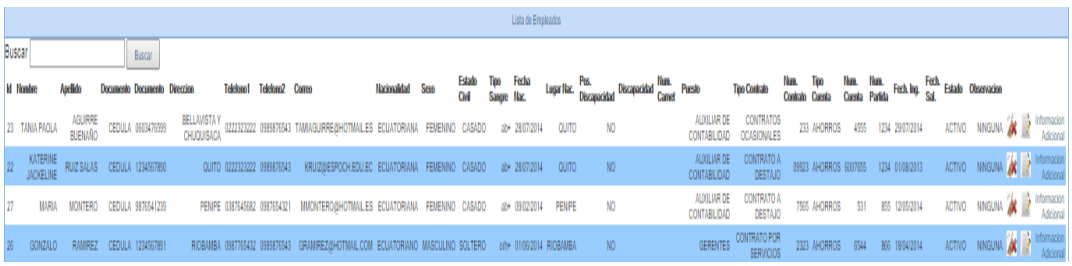
- Datos personales
- Formación Académica
- Capacitación
- Experiencia
- Grupo Familiar

1. En el menú anterior damos clic en el menú Empleado.

DATOS PERSONALES



2. En el caso de que el empleado posea Discapacidad se deberá llenar los campos Descripción de Discapacidad y el número de Carnet.
3. Damos clic en el botón Aceptar
4. Se listara la lista de empleados con el nuevo registro ingresado y con las opciones de editar y eliminar.



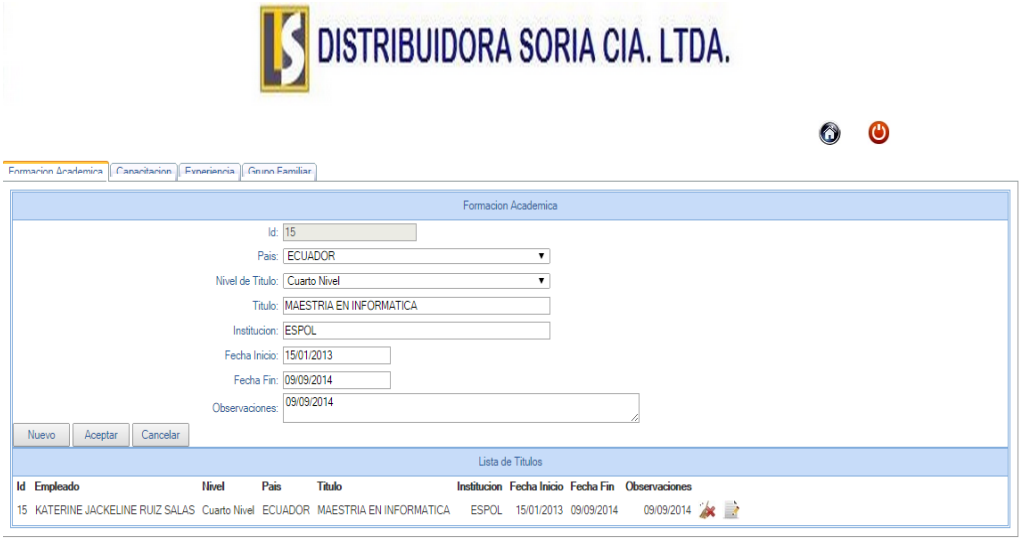
5. Damos clic en Información Adicional para llenar los datos correspondientes a Formación Académica, Capacitación, Experiencia, Grupo Familiar.

DATOS ADICIONALES

1. En cualquiera de las opciones seleccionadas damos clic en nuevo y procedemos a llenar los datos que se nos solicita y damos clic en aceptar.

2. Posterior se despliega el listado con el registro que ha sido ingresado.

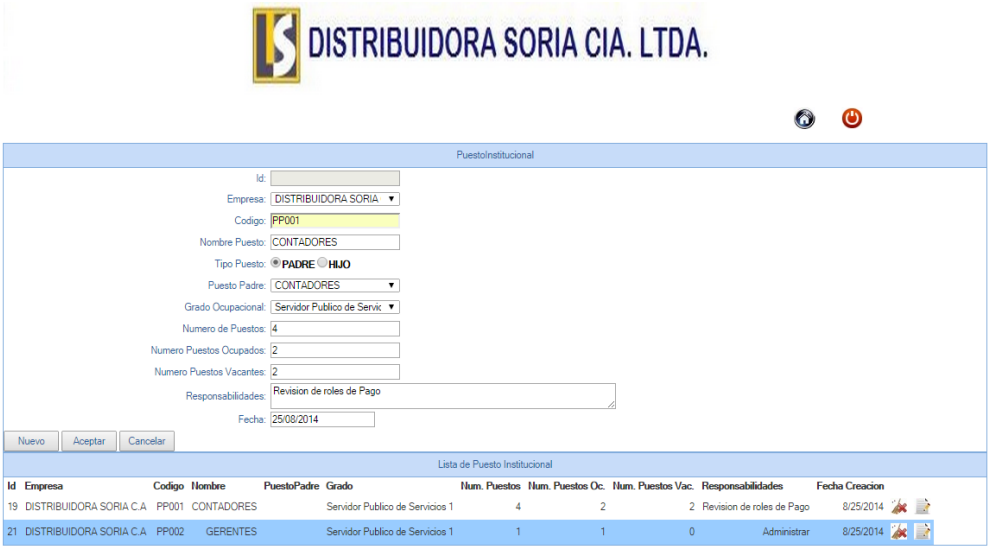
3. En el caso de que se desee editar la información ingresada dar clic en el botón  que se encuentra en la tabla mediante y se cargaran los datos actuales, modifiquelos y de clic en Aceptar, posterior a ello se mostraran los datos modificados .

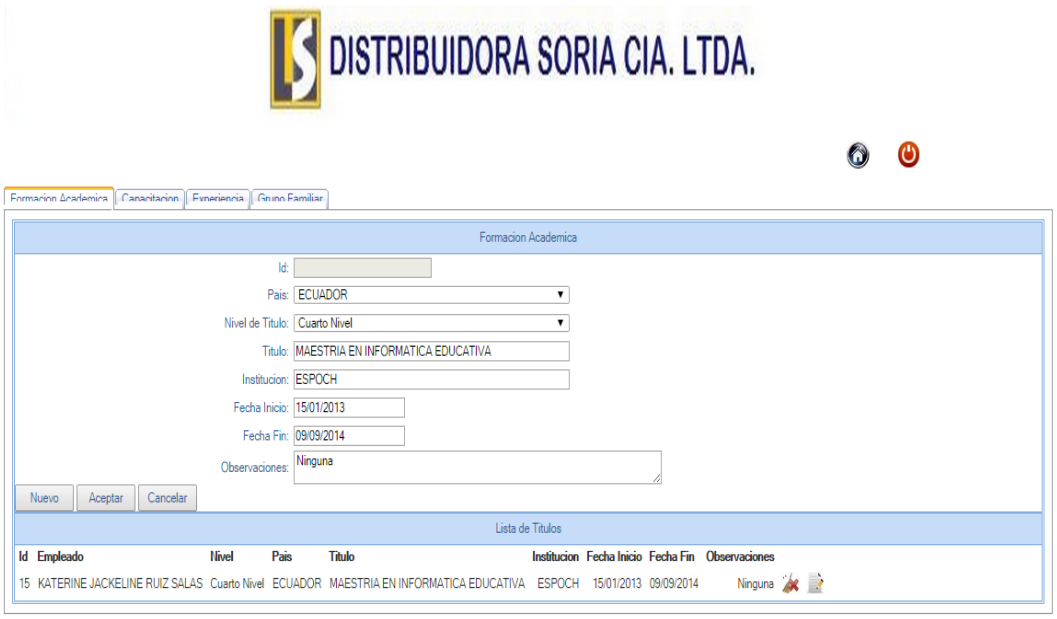


4. Por otro lado si se desea eliminar los datos añadidos daremos clic en el boton .
5. Si no se desea seguir agregando mas datos podemos seguir en las siguientes pesatañas caso contrario podemos regresar al menu principal dando click en el icono o cerrar sesion dando click en el boton .

CREAR PUESTO INSTITUCIONAL

3. Damos clic en nuevo y procedemos a llenar los datos que se nos solicita y damos clic en aceptar.
4. Posterior se despliega el listado con el registro que ha sido ingresado.





5. En el caso de que se desee editar la información ingresada dar clic en el botón  que se encuentra en la tabla mediante y se cargaran los datos actuales, modifiquelos y de clic en Aceptar, posterior a ello se mostraran los datos modificados .
6. Por otro lado si se desea eliminar los datos añadidos daremos clic en el boton  .
7. Si no se desea seguir agregando mas datos podemos seguir en las siguientes pesatañas caso contrario podemos regresar al menu principal dando click en el icono  o cerrar sesion dando click en el boton  .

MÓDULO DE APORTES

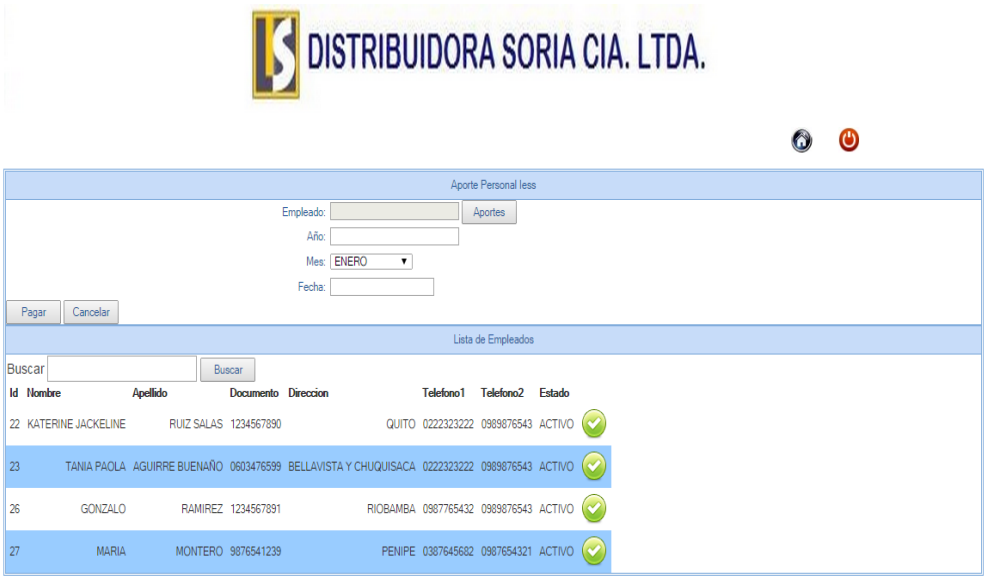
Dentro de este módulo encontramos:

- Aporte Personal IESS
- Aporte Patronal IESS
- Fondo Reserva
- Gastos Personales

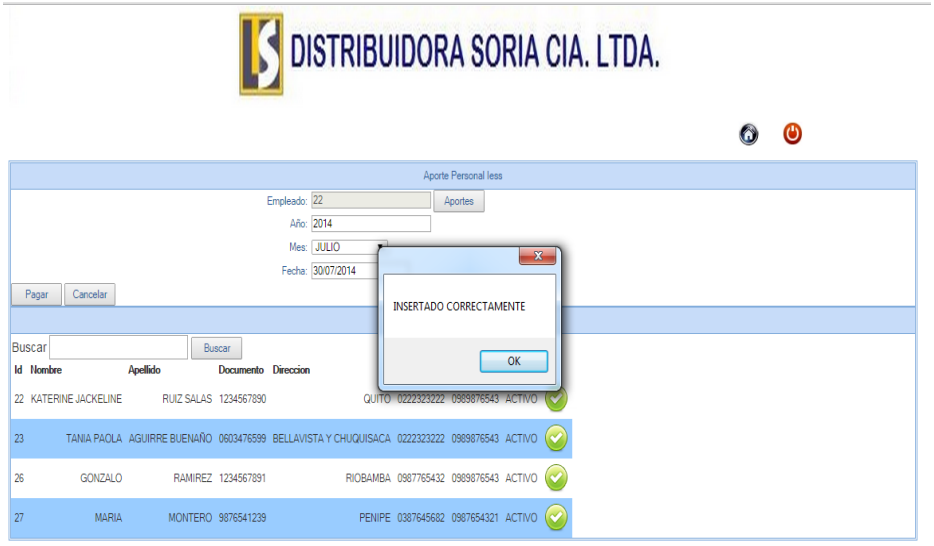
INSERTAR APORTE

1. Seleccionamos el aporte del cual se desea registrar el pago del menú.

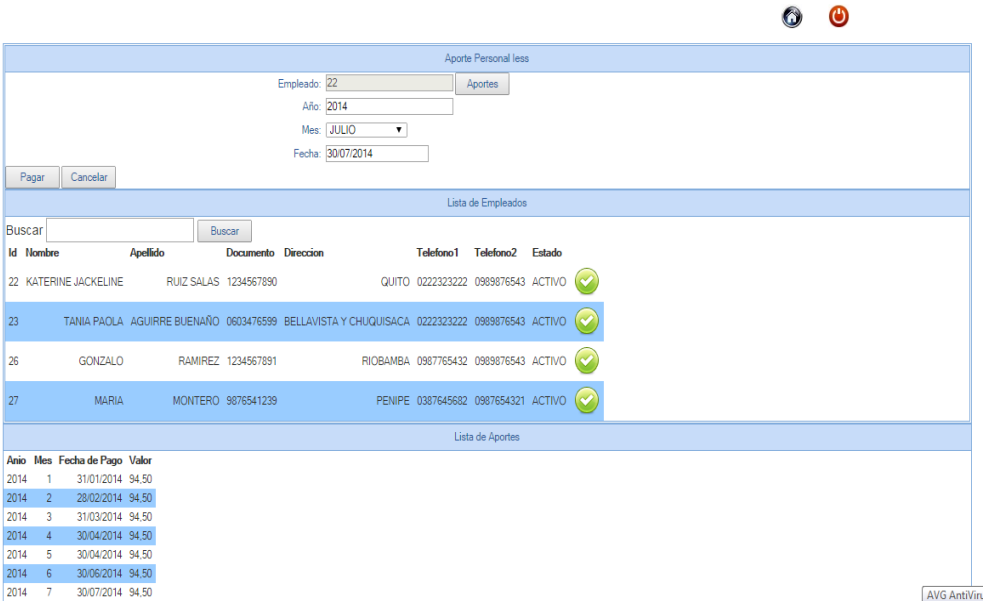
2. Dependiendo de la opción que selecciones se nos muestra la respectiva pantalla con los datos a ser ingresados y el listado de los empleados activos.



3. Seleccionamos mediante le botón  el empleado del cual se desea ingresar el pago.
4. Escribimos el año y seleccionamos el mes y la fecha en la que se le está realizando el pago y damos click en Pagar.
5. En caso de que se haya insertado correctamente aparecerá una pantalla con un mensaje de que se ha insertado correctamente.



6. Si se desea un listado de los aportes que tiene ese empleado incluido el pago recientemente ingresado se debe dar clic en el botón Aportes.



7. Si no se desea seguir agregando mas datos podemos seguir en las siguientes pestañas caso contrario podemos regresar al menu principal dando click en el icono  o cerrar sesion dando click en el boton  .

MÓDULO DE PAGOS

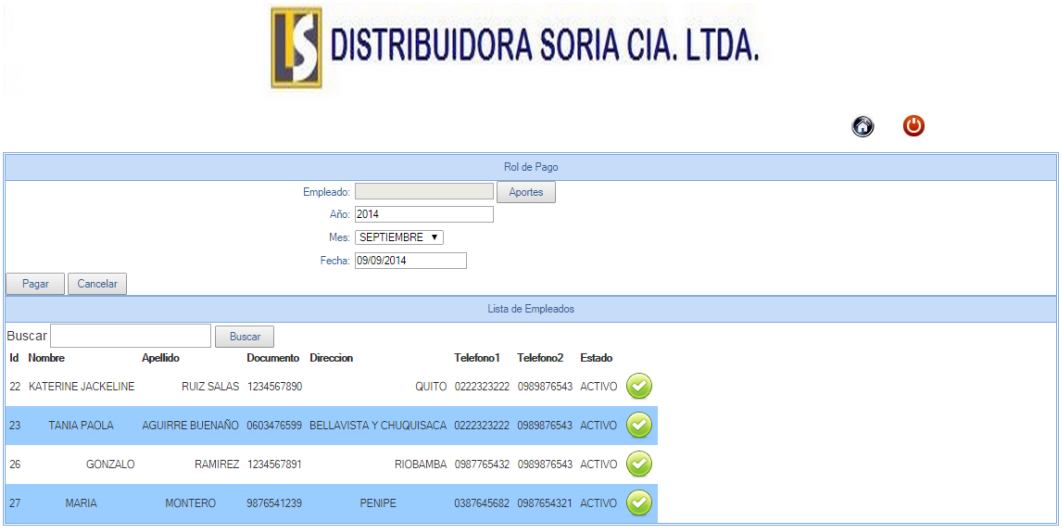
Dentro de este módulo encontramos:

- Rol de Pagos
- Pago Décimo Tercero
- Pago Décimo Cuarto

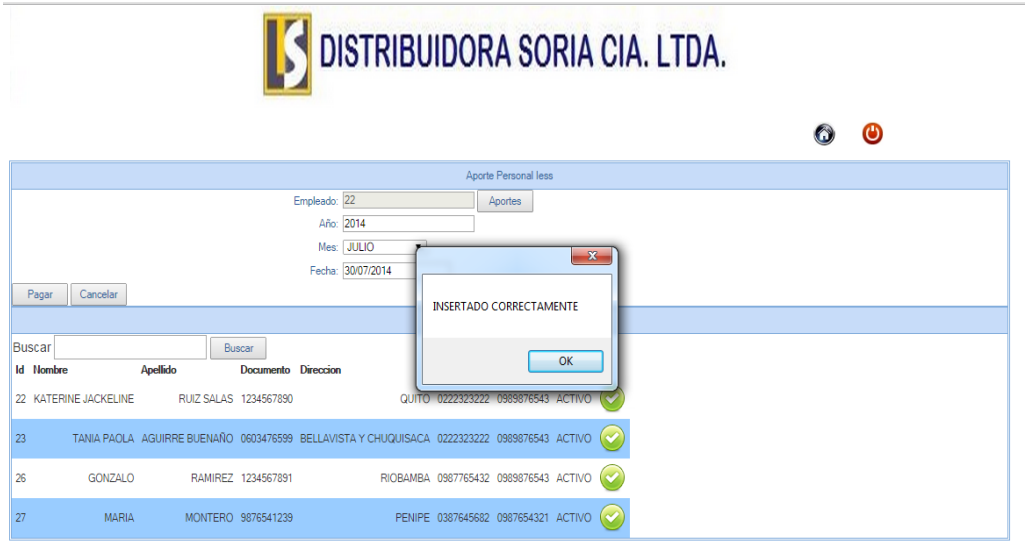
INSERTAR PAGO

1. Seleccionamos el aporte del cual se desea registrar el pago del menú.

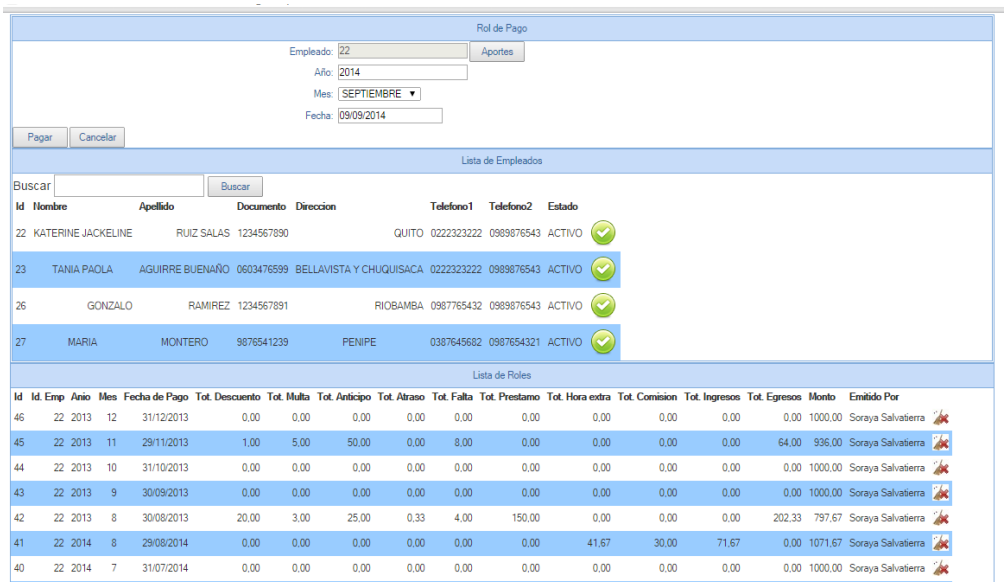
2. Dependiendo de la opción que selecciones se nos muestra la respectiva pantalla con los datos a ser ingresados y el listado de los empleados activos.



3. Seleccionamos mediante le botón  el empleado del cual se desea ingresar el pago.
4. Escribimos el año y seleccionamos el mes y la fecha en la que se le está realizando el pago y damos clic en aceptar.
5. En caso de que se haya insertado correctamente aparecerá una pantalla con un mensaje de que se ha insertado correctamente.



6. Si se desea un listado de los aportes que tiene ese empleado incluido el pago recientemente ingresado se debe dar clic en el botón Aportes.



7. Si no se desea seguir agregando mas datos podemos seguir en las siguientes pesatañas caso contrario podemos regresar al menu principal dando click en el icono  o cerrar sesion dando click en el boton  .

MÓDULO DE EGRESOS

Dentro de este módulo encontramos:

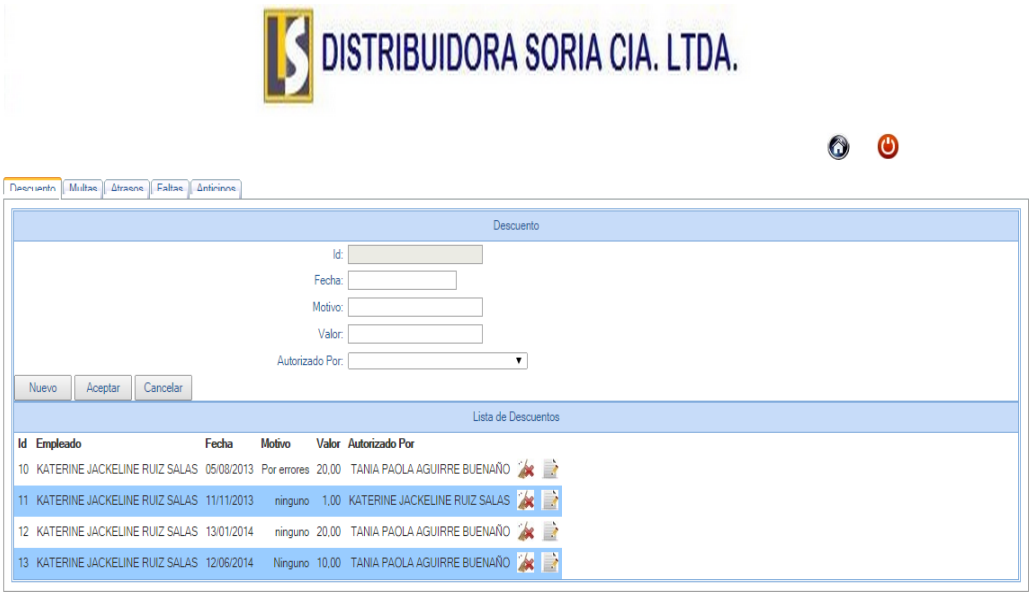
- Descuentos
  - ✓ Multa
  - ✓ Falta
  - ✓ Atraso
  - ✓ Anticipo
- Préstamo

INSERTAR DESCUENTOS

1. Damos clic en egresos y seleccionamos Descuentos del menú correspondiente.

2. Se nos muestra una pantalla con el listado de los empleados activos.

3. Damos clic en Ingresar Descuento.



4. Seleccionamos que deseamos ingresar ya sea este un descuento, multa, atraso, falta o anticipo.
5. Damos clic en nuevo y llenamos los datos que se nos solicitan.

6. En caso de que se haya insertado correctamente se nos mostrara un listado con la información recientemente ingresada.

7. En el caso de que se desee editar la información ingresada dar clic en el botón  que se encuentra en la tabla mediante y se cargaran los datos acttuales, modifiquelos y de clic en Aceptar, posterior a ello se mostraran los datos modificados .
8. Por otro lado si se desea eliminar los datos añadidos daremos clic en el boton  .
9. Si se desea registrar un pago se debemos dar click en Registrar Pago.
10. Llenamos los datos y se listara el pago que se realizó.

11. Si no se desea seguir agregando mas datos podemos seguir en las siguientes pestañas caso contrario podemos regresar al menu principal dando click en el icono  o cerrar sesion dando click en el boton  .

INSERTAR PRESTAMO

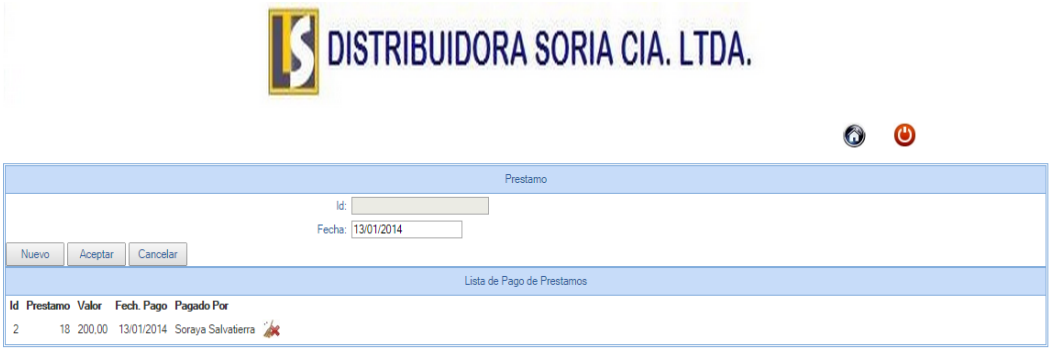
1. Damos clic en egresos del menú correspondiente.

2. En la pantalla que se muestra debemos ingresar los datos que se nos solicitan, se debe tener en cuenta que los empleados se filtran mediante el número de cedula en el combo box.
3. En el caso de que se haya ingresado correctamente la información se mostrara el listado de préstamos con el registro recientemente creado.

4. En el caso de que se desee editar la información ingresada dar clic en el botón  que se encuentra en la tabla mediante y se cargaran los datos acttuales, modifiquelos y de clic en Aceptar, posterior a ello se mostraran los datos modificados .
5. Por otro lado si se desea eliminar los datos añadidos daremos clic en el boton  .

6. Si se desea registrar un pago se debemos dar click en Registrar Pago.

7. Llenamos los datos y se listara el pago que se realizó.



8. Si no se desea seguir agregando mas datos podemos seguir en las siguientes pestañas caso contrario podemos regresar al menu principal dando click en el icono  o cerrar sesion dando click en el boton  .

MÓDULO DE INGRESOS

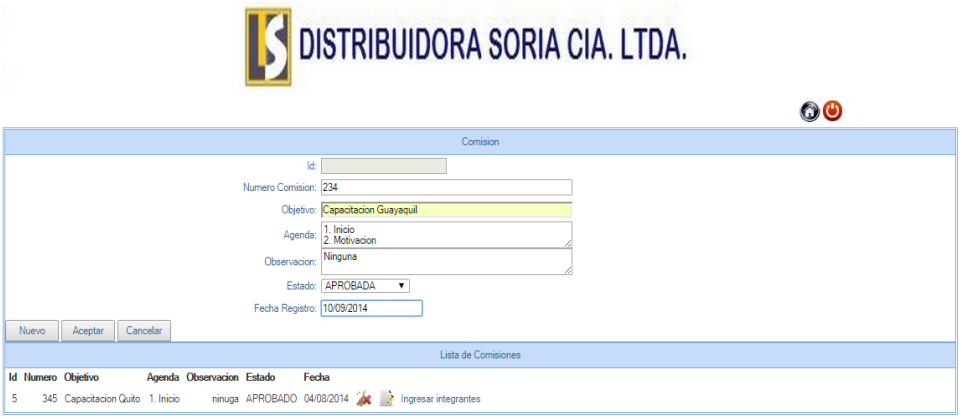
Dentro de este módulo encontramos:

- Comisión
- Plan Hora Extra
- Horas Extra Ejecutadas

INSERTAR COMISION

1. Damos clic en egresos y seleccionamos Descuentos del menú correspondiente.

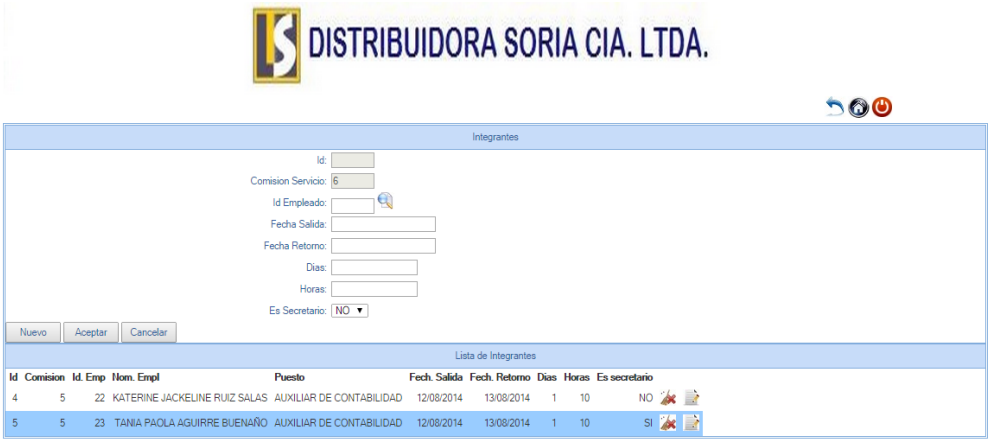
2. Se nos muestra una pantalla en donde se realizara el ingreso de datos dando clic en Nuevo.



3. Damos clic en Aceptar y se nos muestra el listado de comisiones con el registro recientemente insertado.

4. Para agregar los integrantes que forman la comisión damos clic en Ingresar integrantes.

5. Damos clic en nuevo.
6. En esta pantalla damos clic en el icono  para seleccionar el empleado.
7. Llenamos todos los datos que se nos solicita y damos clic en Aceptar.
8. En caso de que se haya insertado correctamente se nos mostrara un listado con la información recientemente ingresada.



9. En el caso de que se desee editar la información ingresada dar clic en el botón que se encuentra en la tabla mediante y se cargaran los datos actgtgwdewdergtewrgtgwergddgfwtuales, modifiquelos y de clic en Aceptar, posterior a ello se mostraran los datos modificados .
10. Por otro lado si se desea eliminar los datos añadidos daremos clic en el boton .
11. Si no se desea seguir agregando mas datos podemos seguir en las siguientes pestañas caso contrario podemos regresar al menu principal dando click en el icono o cerrar sesion dando click en el boton .

MÓDULO TIPOS

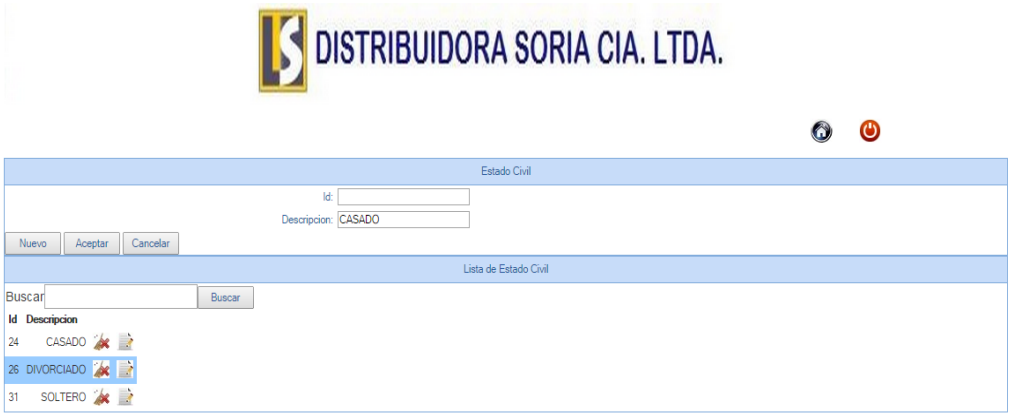
Dentro de este módulo encontramos:

- Tipo de Sangre
- Nivel Titulo
- Estado Civil
- Tipo Grado
- Grado Ocupacional
- Tipo Contrato
- Valores

INSERTAR TIPO

1. Seleccionamos cualquiera de las opciones del menú correspondiente.

2. Se nos muestra una pantalla con los datos que deben ser ingresados, estos serán distintos de acuerdo a la opción seleccionada.



3. En el caso de que se desee editar la información ingresada dar clic en el botón que se encuentra en la tabla mediante y se cargaran los datos acttuales, modifiquelos y de clic en Aceptar, posterior a ello se mostraran los datos modificados .
4. Por otro lado si se desea eliminar los datos añadidos daremos clic en el boton .
5. Si no se desea seguir agregando mas datos podemos seguir en las siguientes pestañas caso contrario podemos regresar al menu principal dando click en el icono o cerrar sesion dando click en el boton .

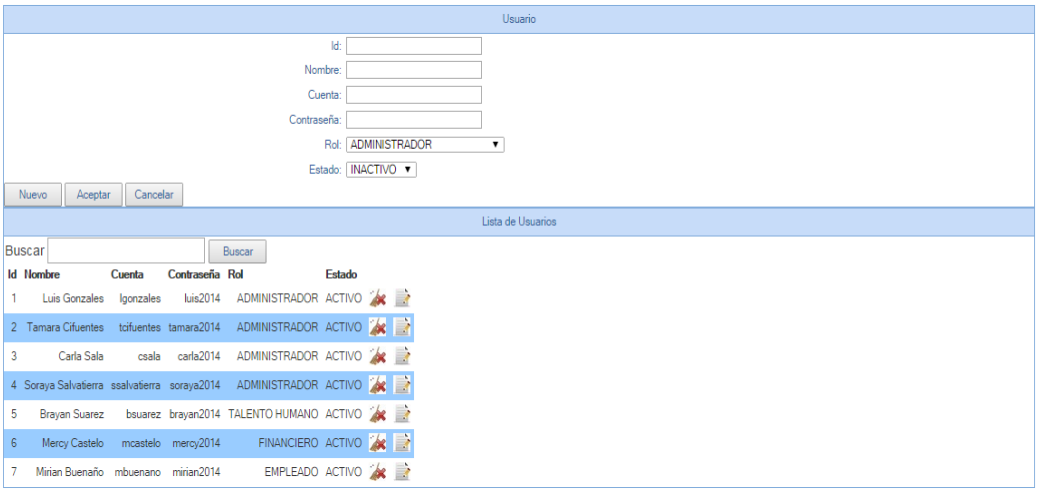
MÓDULO DE ADMINISTRACION

Luego de iniciar la sesión como usuario administrador se presentara la página que contiene las diferentes operaciones que pueden ser ejecutadas con este perfil.

USUARIO

Mediante esta opción podemos administrar las cuentas de usuario.

1. Para agregar un nuevo usuario se deben llenar todos los datos requeridos y dar en clic Aceptar posterior a ello se mostrara los datos que han sido añadidos.



2. En el caso de que se desee editar la información ingresada dar clic en el botón  que se encuentra en la tabla mediante y se cargaran los datos acttuales, modifiquelos y de clic en Aceptar, posterior a ello se mostraran los datos modificados .
3. Por otro lado si se desea eliminar los datos añadidos daremos clic en el boton  .
4. Si no se desea seguir agregando mas datos podemos seguir en las siguientes pestañas caso contrario podemos regresar al menu principal dando click en el icono  o cerrar sesion dando click en el boton  .

MÓDULO REPORTES

Luego de iniciar la sesión como usuario GERENTE ADMINISTRATIVO, CONTADORA GENERAL O AUXILIAR DE CONTABILIDAD se presentara la página que contiene las diferentes operaciones que pueden ser ejecutadas con este perfil.

1. Damos clic en el menú reportes.

REPORTES GENERALES

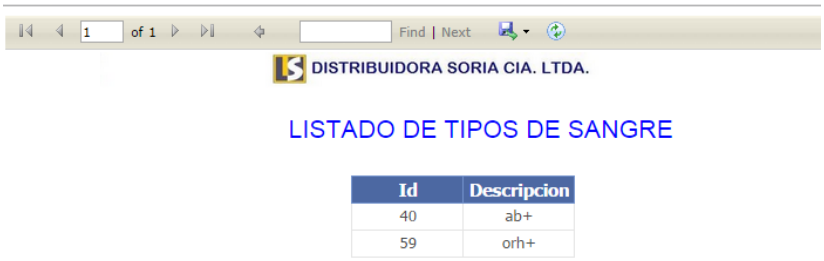
1. Si damos clic en Usuarios se nos mostrara su respectivo reporte que puede ser exportado como PDF, Word o Excel.

DISTRIBUIDORA SORIA CIA. LTDA.

LISTADO DE USUARIOS

Id	Nombre	Cuenta	Contraseña	Estado	Rol
1	Luis Gonzales	lgonzales	luis2014	ACTIVO	ADMINISTRADOR
2	Tamara Cifuentes	tcifuentes	tamara2014	ACTIVO	ADMINISTRADOR
3	Carla Sala	csala	carla2014	ACTIVO	ADMINISTRADOR
4	Soraya Salvatierra	ssalvatierra	soraya2014	ACTIVO	ADMINISTRADOR
5	Brayan Suarez	bsuarez	brayan2014	ACTIVO	TALENTO HUMANO

2. Si damos clic en Tipos Sangre se nos mostrara su respectivo reporte que puede ser exportado como PDF, Word o Excel.



DISTRIBUIDORA SORIA CIA. LTDA.

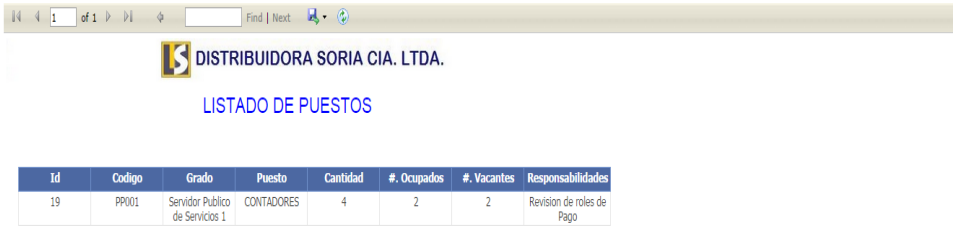
LISTADO DE TIPOS DE SANGRE

Id	Descripcion
40	ab+
59	orh+

3. Si damos clic en Empleado se nos mostrara su respectivo reporte que puede ser exportado como PDF, Word o Excel.

Id	Cedula	Apellidos	Nombres	Telefono1	Telefono2	Direccion	Correo
23	0603476599	AGUIRRE BUENAÑO	TANIA PAOLA	0222323222	0989876543	BELLAVISTA Y CHUQUISACA	TAMIAGUIRRE@HOTMAIL.ES
22	1234567890	RUIZ SALAS	KATERINE JACKELINE	0222323222	0989876543	QUITO	KRUIZ@ESPOCH.EDU.EC
27	9876541239	MONTERO	MARIA	0387645682	0987654321	PENIPE	MMONTERO@HOTMAIL.ES
26	1234567891	RAMIREZ	GONZALO	0987765432	0989876543	RIOBAMBA	GRAMIREZ@HOTMAIL.COM

4. Si damos clic en Puesto Institucional se nos mostrara su respectivo reporte que puede ser exportado como PDF, Word o Excel.



Id	Codigo	Grado	Puesto	Cantidad	#. Ocupados	#. Vacantes	Responsabilidades
19	PP001	Servidor Publico de Servicios 1	CONTADORES	4	2	2	Revision de roles de Pago

ROLES DE PAGO

ROLES DE PAGO POR AÑO

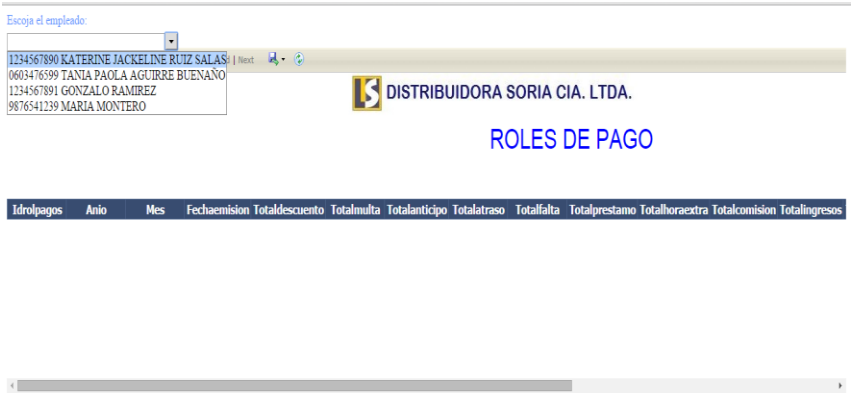
1. Si damos clic en Roles de Pago Por Año se nos mostrara una pantalla donde nos solicita que se ingrese el año.

2. Digitamos el año del que se desea el reporte y damos clic en el botón Mostrar.

Idrolpagos	Anio	Mes	Empleado	Fechaemision	Totalingresos	Totalegresos	Monto	Emitido por
41	2014	8	KATERINE JACKELINE RUIZ SALAS	29/08/2014	71.67	0.00	1071.67	Soraya Salvatierra
40	2014	7	KATERINE JACKELINE RUIZ SALAS	31/07/2014	0.00	0.00	1000.00	Soraya Salvatierra
39	2014	6	KATERINE JACKELINE RUIZ SALAS	30/06/2014	0.00	120.67	879.33	Soraya Salvatierra
38	2014	5	KATERINE JACKELINE RUIZ SALAS	30/05/2014	0.00	0.00	1000.00	Soraya Salvatierra

ROLES DE PAGO DE EMPLEADO

1. Si damos en Roles de Pago de Empleado se nos mostrara una pantalla donde se nos solicita que se escoja el empleado del que se desea el reporte.



2. Seleccionamos el empleado del cual se desea el reporte y automáticamente se
3. muestra el reporte.

ROL DE PAGO INDIVIDUAL

1. Si damos clic en Roles de Pago Individual se nos mostrara una pantalla donde se nos solicita cierta información.

2. Procedemos a llenar los datos y se nos muestra el reporte.

DISTRIBUIDORA SORIA CIA.

ROL DE PAGO

Nombre: KATERINE JACKELINE RUIZ SALAS

Fecha: 9/9/2014 11:38:59 AM

Anio: 2014 **Mes:** 1

Id	Fecha Emision	Total Desc.	Tot. Multa	Tot. Anticipo	Tot. Atraso
34	31/01/2014	20.00	4.00	50.00	1.00

Total Ingresos: 8.33

Total Egresos: 287.00

Total : 721.33