ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA EN SISTEMAS

"ANALISIS DE MÉTODOS DE REUTILIZACIÓN DE CÓDIGO JAVA PARA OPTIMIZAR EL DESARROLLO DE APLICACIONES WEB CASO PRÁCTICO: UNIDAD TÉCNICA DE PLANIFICACIÓN DE LA ESPOCH"

TESIS DE GRADO

PREVIA A LA OBTENCIÓN DEL TITULO DE

INGENIERIA EN SISTEMAS INFORMATICOS

PRESENTADO POR:

CARLOS ANDRÉS LÓPEZ ENRIQUEZ

NOE RAFAEL REMACHE COLCHA

RIOBAMBA – ECUADOR

2013

D E D I C A T O R I A

A 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 mi madre por ser la persona que me ha acompañado durante todo mi trayecto estudiantil y de mi vida, a mi Tía Norma que me ha acogido en su hogar durante todo este arduo camino para convertirme en un profesional. A mi padre quien con su apoyo incondicional fue mi soporte y motivación para culminar esta etapa de mi vida. A mis amigas y amigos, que gracias a nuestra amistad logramos llegar hasta el final del camino y que hasta el momento mantenemos la amistad

Carlos Andrés López Enriquez

AGRADECIMIENTO

Dedico este trabajo principalmente a Dios, por haberme dado la vida y permitirme el haber llegado hasta este momento tan importante de mi formación profesional. A mi madre, por ser el pilar más importante y por demostrarme siempre su cariño y apoyo incondicional sin importar nuestras diferencias de opiniones. A mi padre, a pesar de nuestra distancia física, siento que estás conmigo siempre. A mi tía Norma, a quien quiero mucho, por ayudarme todo este tiempo con esas pequeñas cosas que significaron tanto para mí. A Evelyn mi hermana por apoyarme incondicionalmente durante todo este largo camino de la vida.

Carlos Andrés López Enriquez

DEDICATORIA

Mi Tesis la dedico con todo mi esfuerzo y trabajo a nuestro Padre en los Cielos a quien agradezco la oportunidad de vivir y el haberme dado una familia trabajadora y maravillosa.

Con mucho cariño la dedico este trabajo a mis padres quienes se esforzaron por ayudarme a conseguir este gran triunfo en mi vida, a mi hermano Darío quien nunca dejo de apoyarme y creer en mí, a mis tíos, primos quienes de una forma u otra me apoyaron con sus palabras y consejos.

Noé Rafael Remache Colcha

AGRADECIMIENTO

Agradezco primero a Dios por permitirme llegar a lograr una meta más en mi vida, con mucho cariño agradezco a mis padres quienes se esforzaron por ayudarme a conseguir este gran triunfo en mi vida, renunciando muchas veces a sus metas personales por ayudar a sus hijos y nunca dejaron de apoyarme en los momentos difíciles de la vida. Gracias desde lo más profundo de mi corazón.

A mi hermano Darío quien nunca dejo de apoyarme y creer en mí, a mis tíos, primos quienes de una forma u otra me apoyaron con sus palabras y consejos.

A los docentes de la Escuela de Ingeniería en Sistemas quienes han tenido la máxima paciencia para ayudarnos con sus conocimientos para la realización de este trabajo y nos han comunicado sus experiencias para formarnos como profesionales responsables con la sociedad.

Noé Rafael Remache Colcha

FIRMA DE RESPONSABILIDAD

	FIRMA	**FECHA**
Ing. Iván Menes **DECANO DE LA FACULTAD** **INFORMÁTICA Y ELECTRÓNICA**	.....	.....
Ing. Jorge Huilca **DIRECTOR DE LA ESCUELA** **INGENIERÍA EN SISTEMAS**	.....	.....
Ing. Ivonne Rodríguez **DIRECTORA DE TESIS**	.....	.....
Ing. Gloria Arcos **MIEMBRO DEL TRIBUNAL**	.....	.....
Tlgo. Carlos Rodríguez **DIRECTOR CENTRO DE** **DOCUMENTACIÓN**	.....	.....

NOTA DE LA TESIS: ……

“Nosotros, CARLOS ANDRÉS LÓPEZ ENRIQUEZ y NOE RAFAEL REMACHE COLCHA somos responsables de las ideas, doctrinas y resultados expuestos en esta tesis; y el patrimonio intelectual de la Tesis de Grado pertenece a la ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO”.

_____ _____

CARLOS ANDRÉS LÓPEZ ENRIQUEZ NOÉ RAFAEL REMACHE COLCHA

INDICE DE ABREVIATURAS

ESPOCH	Escuela Superior Politécnica de Chimborazo
SENPLADES	Secretaría Nacional de Planificación y Desarrollo
POA	Plan Operativo Anual
PEDI	Plan Estratégico de Desarrollo Institucional
PND	Plan Nacional de Desarrollo
JSP	Java Server Pages
OOP	Del Inglés Programación Orientado a Objetos
CSS	Hoja de Estilo en Cascada
UML	Del Inglés Lenguaje Unificado de Modelado
SIGPOI	Sistema de Gestión de Planes Operativos e Inversión.

Índice General

Índice de Tablas

Índice de Figuras

INTRODUCCIÓN

La evolución de las comunicaciones y el aumento de uso de internet facilitan a los administradores integrar la información que antes tenían aislada mediante aplicaciones de software que funcionan en la red. La Escuela Superior Politécnica del Chimborazo, como institución educativa a nivel superior, cuenta con la Unidad Técnica de Planificación, la cual es la encargada de dar seguimiento a todos los proyectos propuestos y en ejecución, por lo que un sistema de gestión: de proyectos, planes operativos y planes de inversión será una herramienta que brindara un soporte tecnológico para una correcta administración de los diferentes tipos de proyectos planteados en la institución.

El presente documento tiene como objetivo realizar un análisis que permita destacar las ventajas de la reutilización de código en java utilizando el IDE Netbeans, aplicando al desarrollo del sistema informático de la Unidad de Planificación de la ESPOCH.

Se realizó el estudio, análisis, comparación de los métodos de reutilización de código “*herencia”* y “*composición”*; al comparar ambos métodos de reutilización de código se obtuvo los siguientes valores cuantitativos: El método de reutilización de código *“Herencia”* disminuye aproximadamente un 9% en el tiempo de desarrollo del sistema, disminuye un 9,01% en el esfuerzo realizado para desarrollar el sistema, disminuye un 8,04% en la cantidad de Líneas de código utilizadas para la implementación del sistema, disminuye un 0,18% en el uso del CPU e incrementa el 4,14% en el uso de la memoria RAM, mientras que el método de reutilización de código *“Composición”* incrementa aproximadamente un 3% en el tiempo de desarrollo del sistema, incrementa un 3% en el esfuerzo realizado para desarrollar el sistema, incrementa un 3,11% en la cantidad de Líneas de código utilizadas para la implementación del sistema, incrementa el 0,18% en el uso del CPU y disminuye un aproximado del 51,12% en el uso de la memoria RAM. Se determinó así que el método de reutilización de código más adecuada es la *“Herencia”* porque mejora la productividad en el desarrollo de aplicaciones web, basándonos en los valores obtenidos por los parámetros mencionados anteriormente.

El siguiente documento de tesis está conformado de la siguiente manera:

Capítulo 1, se encuentra el marco referencial, el cual describe objetivos, alcance, y referencias sobre los métodos de reutilización de código (herencia y composición).

Capítulo 2, se presenta de manera teórica conceptos, enunciados sobre los métodos de reutilización de código, plan operativo anual, plan de inversión anual.

Capítulo 3, se describe los prototipos, los parámetros de evaluación y la comparación entre los métodos de reutilización de código (herencia y composición).

Capítulo 4, se aplica la metodología XP para el desarrollo del SIG_POI explicando el esquema y diseño de la aplicación web.

CAPÍTULO I

MARCO REFERENCIAL

1.1. Antecedentes

La facilidad y disponibilidad de los servicios de internet que existe en nuestro país actualmente se hace necesaria la construcción de aplicaciones web que tenga disponibilidad las 24 horas del día, sean multiusuario y ofrezcan rapidez para acceder a los servicios que los usuarios consumen diariamente.

Existe métodos de reutilización de código para agilitar la construcción de sistemas web que permita obtener información rápida y oportuna, los métodos más utilizados actualmente en el desarrollo de aplicaciones son la Composición y la Herencia, estos métodos permiten disminuir el tiempo de desarrollo de una

aplicación, actualmente lenguajes de programación como Java Server Page que se basa en la reutilización de código, permite el diseño de componentes y el uso de objetos para la implementación de la solución web, lo que constituye una ventaja frente a otros lenguajes de programación que no soportan reutilización de código.

Sin embargo existe como se ha descrito anteriormente muchos lenguajes que no soportan la reutilización de código, obligando al desarrollador a consumir mucho más tiempo en el desarrollo de sistemas web, además en varias ocasiones estos métodos de reutilización de código no se utilizan de manera adecuada o su uso es parcial. Por lo que determinar que método de reutilización de código ofrece rapidez a la hora del desarrollo de sistemas web ofrecerá al desarrollador una ayuda en la disminución de tiempo de desarrollo y a disminuir la excesiva cantidad de código fuente que se genera al programar.

Actualmente la Unidad Técnica de Planificación de la ESPOCH (UTP en adelante) es el ente encargado de entregar información institucional a la Secretaría Nacional de Planificación y Desarrollo (SENPLADES) lo que demanda un flujo de información y emisión de reportes rápida, y en este caso estos procesos se hacen de manera manual ralentizando la entrega de información de la institución.

Debido a la necesidad de implementar una aplicación web que permita la administración de los Planes Operativos para la UTP de la Escuela Superior Politécnica del Chimborazo, accediendo a este mediante el servicio de internet que dispone dicho establecimiento de educación superior.

Actualmente la UTP posee un sistema web para la Administración de Planes Operativos desarrollado anteriormente bajo la modalidad de proyecto de tesis, el

cual no fue implementado debido a que presentaba problemas de rapidez de respuesta producidos por errores durante la ejecución de los servicios web que utilizaba el sistema por lo que su uso fue descontinuado.

1.2. Justificación del Tema

1.2.1. Justificación Teórica

Mediante el estudio de la reutilización de código en la implementación de aplicaciones web utilizando el lenguaje de programación JAVA se mostrará el manejo, creación de componentes, que integrados en conjunto formarán parte de una solución y la rápida implementación de actualizaciones que pueden sufrir las aplicaciones web con esta técnica de programación.

1.2.2. Justificación Práctica

La UTP realiza su Plan Operativo Anual, también es la encargada de entregar información institucional a la SENPLADES y articularse a los lineamientos de la Planificación Nacional de Gobierno. Al no disponer de un sistema automatizado que apoye su gestión realiza las tareas de forma manual de manera que se requiere un método de desarrollo rápido y eficaz de sistemas web, ya que las exigencias de organismos de control hace que la ESPOCH requiera de información rápida y oportuna convirtiéndose en una necesidad obligatoria para la institución.

Desarrollando los siguientes módulos:

Módulo de Plan Estratégico de Desarrollo Institucional: El cual contemplará la administración del PEDI 2013 y futuros.

Módulo de Plan Nacional de Desarrollo: Que contiene la parte informativa y los lineamientos a seguir planteados por el gobierno actual.

Módulo de Proyectos: El cual abarca la formulación de los proyectos que se contemplarán en el POA a continuación

Módulo de Planificación Operativa: El cual abarca la Elaboración del Plan Operativo Anual, su Monitorización y Evaluación y la Certificación del Plan y sus actividades.

Módulo de Inversión Pública: Que abarcará la Elaboración del Plan de Inversiones y su Monitoreo y Evaluación.

1.3. Objetivos

1.3.1. Objetivo General

Realizar un Análisis que permita destacar las ventajas de la reutilización de código en java utilizando el IDE Netbeans, aplicando al desarrollo del Sistema Informático de la Unidad de Planificación de la ESPOCH.

1.3.2. Objetivos Específicos

- ✓ Estudio de las diferentes formas de reutilización de Código presente en los IDEs de desarrollo bajo la plataforma Java.

- ✓ Destacar ventajas y desventajas de la reutilización de código para el desarrollo de aplicaciones web.

- ✓ Desarrollar un sistema web para la Unidad Técnica de Planificación de la ESPOCH que le permitirá automatizar los procesos de elaboración de:

- Monitoreo, evaluación y certificación del Plan Operativo Anual
- Elaboración, monitoreo y evaluación del Plan de Inversión Pública.

1.4. Hipótesis

La aplicación de métodos de reutilización de código utilizada por Java, permitirá mejorar la productividad en el desarrollo del sistema para Unidad Técnica de Planificación de la ESPOCH.

CAPÍTULO II

MARCO TEÓRICO

2.1. Introducción

Debido a la necesidad de disponer información de forma rápida y veraz, en la actualidad se están desarrollando sistemas webs utilizando metodologías de desarrollo como es la reutilización de código permitirá desarrollar de manera ágil y organizativa, también permitirá realizar el mantenimiento y futuras actualizaciones de la aplicación web basándose en el cambio o modificación de las clases u componentes utilizados en la implementación.

Existe métodos de reutilización de código para agilitar la construcción de sistemas web que permita obtener información rápida, oportuna, de los cuales se nombrarán los más usados tales como: La creación de clases, construcción de objetos a partir de las clases creadas previamente, herencia entre clases,

compartir características y comportamiento entre clases y objetos, polimorfismo del comportamiento de objetos.

2.2. Reutilización de Código

Por mucho tiempo la reutilización de código se había limitado únicamente al cortado y pegado de código, el programador se acordaba donde tenía un código igual o parecido para utilizarlo en el proceso o actividad de programación actual.

Con la programación estructurada se dio un gran paso para la optimización y ahorro en la construcción de código.

La reutilización de código es el uso de software existente para desarrollar un nuevo software, ha sido empleada desde los primeros días de la programación. Los programadores siempre han reutilizado partes de un código, planillas, funciones o procedimientos.

La idea es que parte o todo el código de un programa de computadora escrito una vez, sea o pueda ser usado en otros programas.

La reutilización de códigos programados es una técnica común que intenta ahorrar tiempo y energía, reduciendo el trabajo redundante.

Las bibliotecas o librerías de software son un buen ejemplo. Al utilizarlas se está reutilizando código.

Con los tipos abstractos de datos también se dio un buen avance en la abstracción y solución a situaciones donde se necesitaba una solución computacional. Los paradigmas presentes en la ingeniería del software permiten un aprovechamiento más eficiente del código ya construido.

La programación por componentes: También hace buen aporte para la reutilización de código y ahorro de esfuerzo, tiempos en tareas de construcción de software.

No se puede olvidar las metodologías, métodos, técnicas, estrategias y lenguajes de modelado de la ingeniería del software, algunas de estas son la programación extrema, Rup, MSF, Uml-lenguaje de modelado, SCRUM5, Crystal Methodologies, Dynamic Systems Development Method7 (DSDM), Adaptive Software Development8 (ASD), Feature -Driven Development9 (FDD), Lean Development10 (LD), entre otras.

Otra punto importante sería revisar las teorías importantes sobre **Arquitecturas de Software**, donde dependiendo del sistema a construir se puede seleccionar una u otra arquitectura. La arquitectura a grandes rasgos son los elementos, antes llamados componentes, las relaciones de estos componentes, su adecuada agregación, unión e interrelación para formar un todo.

Entonces cuando se habla de reutilización de código se busca el ahorro de tiempo y esfuerzo y la ingeniería del software evoluciona para brindar nuevas y mejores estrategias para la industria del software.

2.2.1. Características

Modularidad

Es un componente de un sistema más grande y opera dentro del sistema independientemente de las operaciones de otros componentes, también es una opción importante para la escalabilidad y comprensión de programas, además de ahorrar trabajo y tiempo en el desarrollo.

Acoplamiento

Es el grado de interdependencia entre las unidades de software (módulos, funciones, subrutinas, bibliotecas, etc.) de un sistema informático. El acoplamiento da la idea de lo dependiente que son las unidades de software entre sí, es decir, el grado en que una unidad puede funcionar sin recurrir a otras.

El bajo acoplamiento permite:

- ✓ Mejorar la mantenibilidad de las unidades de software.
- ✓ Aumentar la reutilización de las unidades de software.
- ✓ Evitar el efecto onda, ya que un defecto en una unidad puede propagarse a otras, haciendo incluso más difícil de detectar dónde está el problema.
- ✓ Minimiza el riesgo de tener que cambiar múltiples unidades de software cuando se debe alterar una.

Tipos de acoplamiento

- ✓ **Acoplamiento normal:** Una unidad de software llama a otra de un nivel inferior y tan solo intercambian datos (por ejemplo: parámetros de entrada/salida). Dentro de este tipo de acoplamiento existen 3 subtipos, dependiendo de los datos que intercambien las unidades de software.
- ✓ **Acoplamiento externo:** Las unidades de software están ligadas a componentes externos, como por ejemplo dispositivos de entrada/salida, protocolos de comunicaciones, etc.
- ✓ **Acoplamiento común:** Dos unidades de software acceden a un mismo recurso común, generalmente memoria compartida, una variable global o un fichero.

- ✓ **Acoplamiento de contenido:** Ocurre cuando una unidad de software necesita acceder a una parte de otra unidad de software.

Cohesión

Hace referencia a la forma en que agrupar unidades de software (módulos, subrutinas) en una unidad mayor. Un buen diseño es que la cohesión debe ser alta. Es decir, mientras más cohesionados estén los elementos agrupados, mejor.

El acoplamiento, junto con la modularidad, la cohesión y otros factores, permiten mejorar la programación y el diseño de sistemas informáticos y aplicaciones, y son cruciales en el incremento de la reutilización de los códigos.

Tipos de cohesión (criterios de agrupamiento)

- ✓ **Cohesión funcional:** Los elementos de la unidad de software están relacionados en el desarrollo de una única función. Es decir, las unidades de software trabajan juntas con un mismo fin. En general, es el criterio de agrupación más deseable. Probablemente haya entre las unidades un acoplamiento relativamente alto, por lo tanto es conveniente que estén juntas.
- ✓ **Cohesión secuencial:** Una unidad de software realiza distintas tareas en secuencia, de forma que las entradas de cada tarea son las salidas de la tarea anterior. En otras palabras, se agrupan las unidades que cumplen que los resultados o salidas que produce una sirven como entrada para que la próxima continúe trabajando.
- ✓ **Cohesión comunicacional o de datos:** La unidad de software realiza actividades paralelas usando los mismos datos de entrada y salida. En

otras palabras, cuando todas las unidades agrupadas trabajan sobre el mismo conjunto de datos.

- ✓ **Cohesión procedimental:** La unidad de software tiene una serie de funciones relacionadas por un procedimiento efectuado por el código. Es similar a la secuencial, pero incluyendo el paso de controles.
- ✓ **Cohesión lógica:** Cuando las unidades de software agrupadas realizan un trabajo en una misma categoría lógica, pero no necesariamente tienen relación entre sí.
- ✓ **Cohesión temporal:** Los elementos de la unidad de software están implicados en actividades relacionadas con el tiempo. En otras palabras, se agrupan unidades de software que tienen que ejecutarse más o menos en el mismo período de tiempo, sin que haya otro tipo de relación entre ellas. En general debe evitarse.
- ✓ **Cohesión casual o coincidente:** Los elementos de la unidad de software contribuyen a las actividades relacionándose mutuamente de una manera poco significativa. En otras palabras, es cualquier criterio que no caiga dentro de los anteriores. Este tipo de cohesión viola el principio de independencia y de caja negra de las unidades de software, por lo tanto debe evitarse.

Ocultación de Información

Se trata de la ocultación de la implementación de un programa o unidad de software, proveyendo a la vez una interfaz estable para acceder a éstos. La interfaz de una unidad de software es la única forma que tienen otras unidades de comunicarse e interactuar sobre ésta.

En los lenguajes de programación modernos existen múltiples formas de llevar a cabo la ocultación de información, por ejemplo, el encapsulamiento. Algunos autores toman como sinónimos la ocultación de la información y el encapsulamiento, mientras que otros consideran la primera como el principio y la segunda como un método para implementar el principio.

Escalabilidad

Es la propiedad deseable en un sistema, red o proceso que indica su habilidad para poder hacerse más grande sin perder calidad en sus servicios. La escalabilidad de un sistema requiere un pensamiento cuidadoso desde el principio de su desarrollo.

2.2.2. Paradigmas

Programación Orientada a Objetos (OOP)

Cuando se escribe un programa en un lenguaje orientado a objetos, se define una plantilla o clase que describe las características y el comportamiento de un conjunto de objetos similares. Los atributos o propiedades se refieren a las características que posee y lo hace único. El comportamiento se refiere al comportamiento, funciones o acciones que puede realizar.

La POO es un paradigma, es otra forma de pensar, es una filosofía única a diferencia de un lenguaje de programación orientado a objetos el cual existen muchos y permiten hacer uso de ese paradigma con el ánimo de solucionar problemas reales mediante la abstracción de los diferentes agentes, entidades o elementos que actúan en el planteamiento de un problema.

Componentes de la OOP

Los componentes de la programación orientada a objetos facilitan el desarrollo de aplicaciones para los programadores ofreciendo una amplia variedad en cuanto a recursos. A continuación se detallará los más utilizados por los programadores.

Clase, es una plantilla implementada en software que describe un conjunto de objetos con atributos y comportamiento similares. Las clases de objetos representan conceptos o entidades significativos en un problema determinado.

Objeto, es una instancia u objeto de una clase es una representación concreta y específica de una clase y que reside en la memoria del ordenador.

Atributos, son las características individuales que diferencian un objeto de otro y determinan su apariencia, estado u otras cualidades. La definición e implementación se la realiza en el mismo lugar, en un único fichero .java

Comportamiento, se implementa mediante funciones miembro o métodos. Un método es un conjunto de instrucciones que realizan una determinada tarea y son similares a las funciones de los lenguajes estructurados, también hay métodos de instancia y de clase. En el primer caso, un objeto llama a un método para realizar una determinada tarea, en el segundo, el método se llama desde la propia clase.

2.3. Reutilización de Código en Java

El programa se debe adaptar a la «jerga» del problema real, creando nuevos tipos de objetos, de manera que al leer el código que implementa la solución, se esté leyendo algo que expresa el problema en sí. Un objeto tiene un estado que puede modificar como consecuencia de su respuesta a un mensaje.

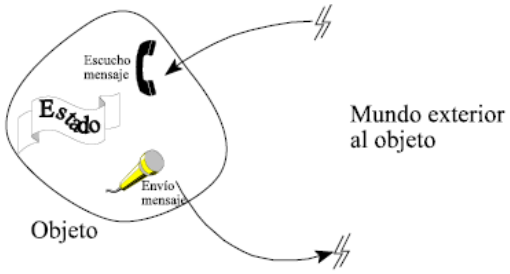


Figura II. 1. Modelo de programación orientado a objetos

Fuente: http://www.slideshare.net/senaticscesar/programacion-orientada-a-objetos-4540712

Se fundamente en 5 puntos básicos:

- ✓ Todo es un objeto.
- ✓ Un programa es un cúmulo de objetos que dicen entre sí lo que tienen que hacer a través de mensajes.
- ✓ Cada objeto tiene una memoria interna (estado), formada por objetos más pequeños; así hasta llegar a objetos básicos o primitivos.
- ✓ Todo objeto pertenece a un tipo. Según la terminología O.O. un tipo es lo mismo que una clase.
- ✓ Todos los objetos de un tipo concreto, son susceptibles de recibir los mismos mensajes.

2.3.1. Clases

Una **clase** es un modelo a partir del cual se puede construir un objeto mediante un método llamado constructor. El objeto sería una instancia de la clase, con un identificador y un estado independientes.

Para crear una clase se utiliza la palabra reservada **class** y a continuación el nombre de la clase. La definición de la clase se pone entre las llaves de apertura y cierre. El nombre de la clase empieza por letra mayúscula.

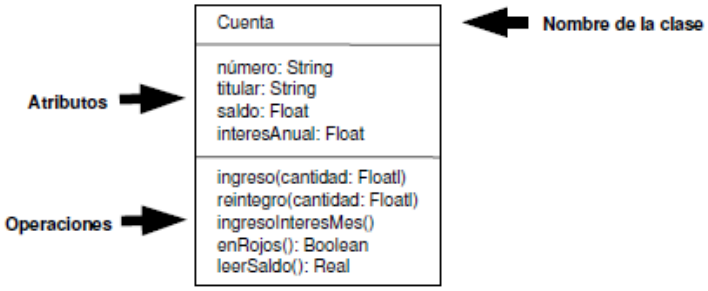


Figura II. 2. Clase en java

Fuente: http://www.sc.ehu.es/sbweb/fisica/cursoJava/fundamentos/clases1/clases.htm

2.3.2. Objetos

En la programación orientada a objetos (POO en adelante), se llama **objeto** a cualquier entidad que se puede representar en un programa, ya sea un objeto real o un concepto.

Durante la ejecución de la aplicación se producirá la instanciación de la clase, es decir, la creación de los objetos que representan cada uno de los individuos con sus características propias, es decir, valores específicos para sus atributos.

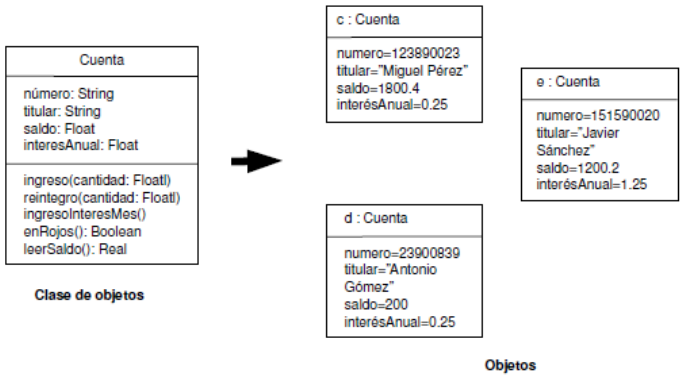


Figura II. 3. Objetos en java

Fuente: http://www.sc.ehu.es/sbweb/fisica/cursoJava/fundamentos/clases1/clases.htm

2.3.3. Paquetes

Los miembros no etiquetados son accesibles por parte de clases amigas. En C++ y otros lenguajes OO las clases amigas a una dada pueden indicarse explícitamente. En java se consideran amigas todas aquellas que forman parte del mismo paquete.

Un fichero fuente java forma en sí un paquete y por tanto todas las clases incluidas en él son amigas. Las clases incluidas en varios ficheros fuente pueden agruparse en un único paquete indicando el nombre de paquete al principio de cada fichero mediante el indicador package.

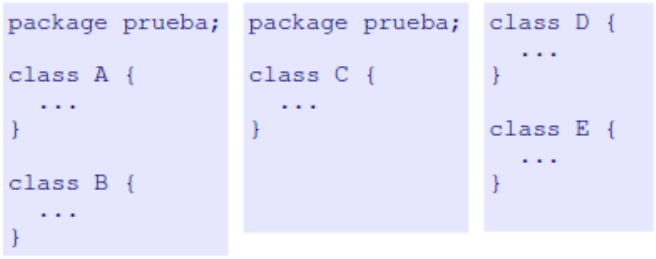


Figura II. 4. Paquetes en java

Fuente: http://www.sc.ehu.es/sbweb/fisica/cursoJava/fundamentos/clases1/clases.htm

2.3.4. Atributos

Los atributos pueden ser de cualquiera de los tipos básicos de Java: boolean, char, byte, short, int, long, float y double, referencias a otros objetos o arrays de elementos de alguno de los tipos citados.

El acceso a los atributos de la clase desde la implementación de las operaciones se realiza de forma directa Los atributos u operaciones estáticas (static) no son afectados por el proceso de instanciación de objetos a partir de la clase

De un atributo estático no se genera una copia por cada objeto que se crea. Existe una única copia compartida y accesible desde todos los objetos de la clase.

Una operación estática únicamente puede acceder a miembros estáticos. El principio de ocultación de información se plasma en los lenguajes OO en diversos mecanismos de protección de los miembros de la clase.

UML permite asociar tres niveles de protección diferentes a cada miembro de la clase:

- ✓ **Miembros públicos (+).** Sin ningún tipo de protección especial
- ✓ **Miembros privados ().** Inaccesibles desde el exterior de la clase
- ✓ **Miembros protegidos (#).** Similares a los privados aunque se permite su acceso desde las clases descendientes

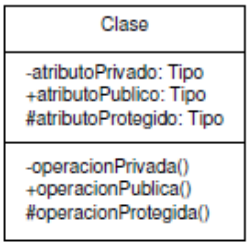


Figura II. 5. Atributos en java

Fuente: http://www.sc.ehu.es/sbweb/fisica/cursoJava/fundamentos/clases1/clases.htm

2.3.5. Comportamiento

En java un miembro se etiqueta como público colocando el identificador public delante de su declaración. Para los miembros privados se utiliza el identificador prívate.

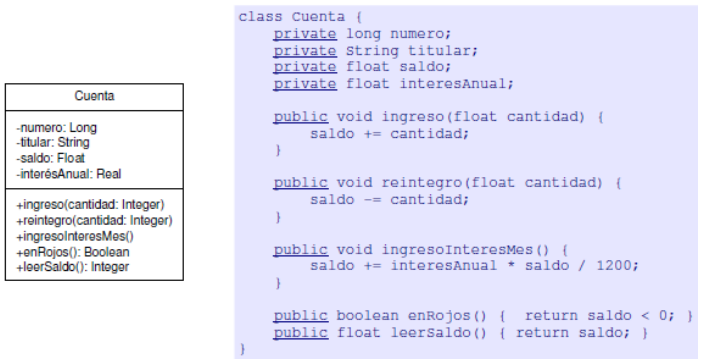


Figura II. 6. Comportamiento de clases

Fuente: http://danubuntu.wordpress.com/2008/07/30/conceptos-sobre-polimorfismo-y-programacion-orientada-a-objetos/

2.3.6. Polimorfismo

La idea de polimorfismo se basa en ocultar bajo una interface común, diferentes implementaciones de métodos. Así si durante la ejecución del programa en distintos instantes de tiempo.

Es un mecanismo que se aprovecha de la herencia (especialmente de interfaz) para manejar indistintamente objetos de las subclases como si fuesen objetos de la clase base, sin preocuparse por la clase en concreto a la que pertenecen. Interesa utilizarlo cuando un comportamiento varía en función del tipo de algo.

En programación orientada a objetos, se refiere a la posibilidad de acceder a un variado rango de funciones distintas a través del mismo interfaz. O sea, un mismo identificador puede tener distintas formas (distintos cuerpos de función, distintos comportamientos) dependiendo del contexto en el que se halle. El polimorfismo se puede establecer mediante sobrecarga, sobre-escritura y enlace dinámico.

También nos permite programar en forma general, en lugar de hacerlo en forma específica. En general nos sirve para programar objetos con características comunes y que todos estos compartan la misma superclase en una jerarquía de

clases, como si todas fueran objetos de la superclase. Esto nos simplifica la programación.

Utilización del Polimorfismo

- ✓ Se declaran atributos, parámetros o variables de la clase base.
- ✓ Se les asignan objetos de alguna de las subclases.
- ✓ Estamos seguros de que se pueden usar todos los métodos declarados en la clase base.
- ✓ Si necesitamos usar métodos de las subclases es necesario hacer un cast.
- ✓ La utilización del cast aumenta la posibilidad de hacer conversiones erróneas, por lo que es mejor evitarlo
- ✓ Se puede preguntar por la clase a la que pertenece un objeto: instanceof, objeto.getClass().getName()

2.3.7. Constructores

El cuerpo de un constructor es como el cuerpo de un método. Contiene declaraciones de variables locales, bucles y otras sentencias. Muy a menudo un constructor necesita aprovechar el código de inicialización de su superclase. De hecho, algunas clases deben llamar a un constructor de su superclase para trabajar correctamente. Si está presente, el constructor de la superclase debe ser la primera sentencia en un constructor: un objeto debe realizar las inicializaciones de nivel más alto antes. Un constructor de la superclase se llama mediante:

- ✓ super (parametros);

Cuando se declaran los constructores de una clase se pueden utilizar especificadores de acceso en la declaración del constructor para definir qué otros objetos pueden crear instancias de la clase:

- ✓ **Private:** Ninguna otra clase puede instanciar esta clase. La clase puede contener métodos de clase públicos (llamados a veces métodos *factory*) y esos métodos pueden construir un objeto de la clase y retornarlo.
- ✓ **Protected:** Solamente las subclases de la clase pueden crear instancias de ella.
- ✓ **Public:** Cualquier clase puede crear una instancia de esta clase.
- ✓ **Package:** Sólo las clases dentro del mismo paquete pueden construir instancias de esta clase.

2.4. Composición

La composición significa utilizar objetos dentro de otros objetos. Por ejemplo, un applet es un objeto que contiene en su interior otros objetos como botones, etiquetas, etc. Cada uno de los controles está descrito por una clase. La composición es un tipo de relación **dependiente** en dónde un objeto más complejo es conformado por objetos más pequeños. Es un tipo de relación de agregación. La clase “todo” controla la existencia de las clases “parte”.

- ✓ Al inicio: “todo” construye cada “parte”.
- ✓ Al final: “todo” destruye cada “parte”.

La composición es un tipo de agregación que añade el matiz de que la clase “todo” controla la existencia de las clases “parte”. Es decir, normalmente la clase “todo” creará al principio las clases “parte” y al final se encargará de su destrucción.

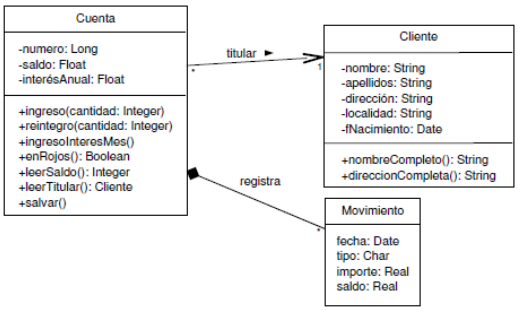


Figura II. 7. Clases con composición

Fuente: http://www.dsi.fceia.unr.edu.ar/downloads/informatica/info_II/mapeoUMLJava3.doc

Las composiciones tienen una implementación similar a las asociaciones, con la diferencia de que el objeto principal realizará en algún momento la construcción de los objetos compuestos.

```
import java.util.Date

class Movimiento {
    Date fecha;
    char tipo;
    float importe;
    float saldo;

    public Movimiento(Date aFecha, char aTipo, float aImporte, float aSaldo) {
        fecha = aFecha;
        tipo = aTipo;
        importe = aImporte;
        saldo = aSaldo;
    }
}
```

Figura II. 8. Composición en código Java

Fuente: http://www.dsi.fceia.unr.edu.ar/downloads/informatica/info_II/mapeoUMLJava3.doc

Relación Tiene-un: También se puede decidir que un elemento concreto compone a otro tipo de objetos más general.

Así, un objeto puede contener a muchos otros, y así sucesivamente. Cuando un objeto recibe un mensaje, dentro del método asociado a éste puede:

- ✓ Responder directamente.

- ✓ Reenviar el mensaje a otros objetos externos.
- ✓ Reenviar el mensaje a objetos que él mismo contiene.

Las composiciones son asociaciones que representan acumulaciones muy fuertes. Esto significa que las composiciones también forman relaciones completas, pero dichas relaciones son tan fuertes que las partes no pueden existir por sí mismas. Únicamente existen como parte del conjunto, y si este es destruido las partes también lo son.

En UML, las composiciones están representadas por un rombo sólido al lado del conjunto.

Al igual que en la agregación, es una relación estructural pero se le suma, que tiene un método de destrucción de los objetos. Y a diferencia de la asociación, el ciclo de vida del objeto área está relacionado con el del objeto ruta. Es decir que si la ruta de viaje se levanta, las áreas que surgían a partir de ella desaparecen. También se puede leer como que una ruta tiene varias áreas de cobertura.

La asociación más simple es una línea recta que interconecta a dos clases. Se define como una relación estructural que describe un conjunto de enlaces; un enlace es una conexión entre objetos.

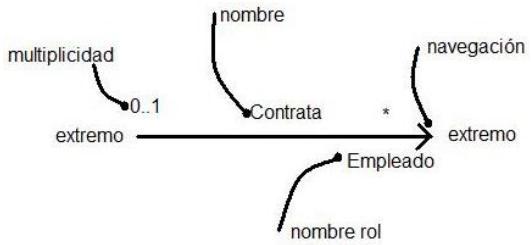


Figura II. 9. Ejemplo asociación

Fuente: http://www.dsi.fceia.unr.edu.ar/downloads/informatica/info_II/mapeoUMLJava3.doc

En los extremos se ubica a las clases que están cooperando. Entre ellas aparece la línea, la cual puede tener todas o ninguna de las descripciones vistas a saber.

Cuando una asociación conecta dos clases se denomina binaria y cuando lo hace con más de dos clases se denomina n-aria.

Nombre: El nombre describe la naturaleza de la relación, para que no exista ambigüedad en ella se puede colocar dirección a dicho nombre, de esta forma se identifica como leerlo.

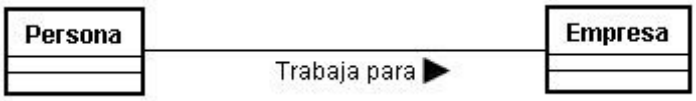


Figura II. 10. Relación por nombre

Fuente: http://www.dsi.fceia.unr.edu.ar/downloads/informatica/info_II/mapeoUMLJava3.doc

Esta relación del ejemplo se lee la persona trabaja para la empresa, la flecha del nombre da la clase que inicia.

Rol: define el papel específico que desempeña una clase en la relación de asociación. Se ve cómo la cara que ofrece una clase del extremo a la del otro extremo

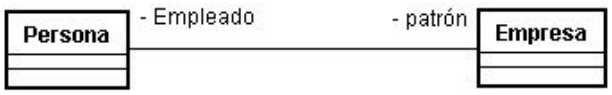


Figura II. 11. Relación por rol

Fuente: http://www.dsi.fceia.unr.edu.ar/downloads/informatica/info_II/mapeoUMLJava3.doc

Se lee Persona es empleado de la empresa. La empresa es patrón de la persona.

2.4.1. Multiplicidad

Señala cuantos objetos de la instancia de una clase se generan según el otro extremo puede ser 0,1,2,n (0..1) (0..n) (n..*) *, asterisco significa muchos sin límite dado n es un valor fijo cualquiera.

Este es una descripción muy especializada que será tratada de manera simple y que con seguridad se ampliará en el contexto de modelamiento de sistemas de información.

Figura II. 12. Relación por multiplicidad

Fuente: http://www.dsi.fceia.unr.edu.ar/downloads/informatica/info_II/mapeoUMLJava3.doc

- ✓ Uno o más empleados tienen varios patrones
- ✓ Muchos patrones tienen uno o muchos empleados.
- ✓ Una o muchas personas están en muchas empresas.
- ✓ Muchas empresas tienen una o muchas personas.

Es una forma asociación simétrica que connota una relación “todo/parte” en la cual la remoción del conjunto (también llamado “compuesto”) implica la eliminación (si es la decisión del “todo”) de las partes componentes y en la cual la “parte” puede pertenecer a un solo conjunto a la vez (aunque el “todo” puede transferir la propiedad de la “parte” a otro objeto, el cual pasará a ser responsable de su tiempo de vida).

Ejemplo: Auto y Motor

```
class Car {
private Engine engine = New  Engine( this ) ;
}
class Engine {
Car car = null ;
public Engine(Car car) {
this.car = car;
}}
```

Figura II. 13. Multiplicidad en código java

Fuente: http://www.dsi.fceia.unr.edu.ar/downloads/informatica/info_II/mapeoUMLJava3.doc

2.4.2. Funcionamiento

Nuevamente, al igual que para la relación de agregación, la implementación, es indistinguible de la asociación, aun cuando en UML representan conceptos diferentes, por tanto, cualquier tipo de representación sufrirá de una pérdida de información e introduce una discontinuidad entre diseño e implementación. Algunos autores afirman que la diferencia entre composición es a menudo un asunto de gusto más que una diferencia semántica y que la composición expresa que las partes están físicamente contenidas en el todo y que esto es imposible de implementar en java puesto que los atributos de una clase son sólo referencias o punteros a objetos de otra clase, nunca es un valor contenido físicamente.

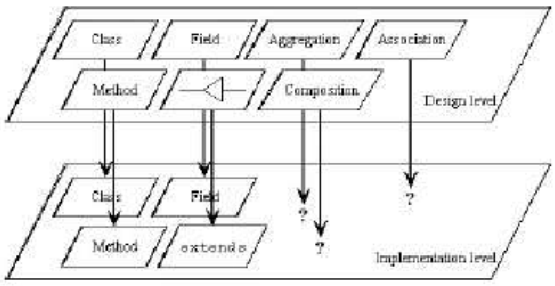
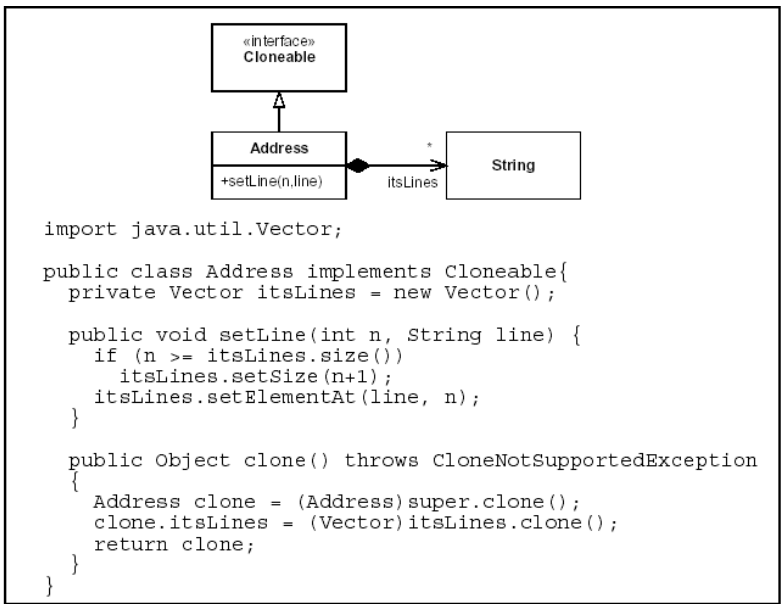


Figura II. 14. Funcionamiento de composición

Fuente: http://www.dsi.fceia.unr.edu.ar/downloads/informatica/info_II/mapeoUMLJava3.doc

En java la destrucción de objetos ocurre “entre bastidores” a cargo del recolector de basura (*garbage collector*), de forma tal que, muy rara vez se maneja el tiempo de vida de un objeto; esta es una de las razones por las cuales es infrecuente el uso de la relación de composición (y la de agregación) para describir programas.

Otro ejemplo: Aquí la composición es usada para indicar la clonación (en su forma “*deepcopy*”) de sus componentes, para impedir que los usuarios de una clase cambien los componentes de un objeto sin usar métodos del contenedor. En este ejemplo se tiene una clase que representa direcciones completas de personas (Address) que contiene muchos String, cada cadena contiene una línea de la dirección completa. Cuando se hace una copia de la dirección se desea que la copia cambie independientemente del original.




```
import java.util.Vector;

public class Address implements Cloneable{
  private Vector itsLines = new Vector();

  public void setLine(int n, String line) {
    if (n >= itsLines.size())
      itsLines.setSize(n+1);
    itsLines.setElementAt(line, n);
  }

  public Object clone() throws CloneNotSupportedException
  {
    Address clone = (Address)super.clone();
    clone.itsLines = (Vector)itsLines.clone();
    return clone;
  }
}
```

Figura II. 15. Composición en código java

Fuente: http://www.dsi.fceia.unr.edu.ar/downloads/informatica/info_II/mapeoUMLJava3.doc

2.4.3. Ventajas

- ✓ Maneja subobjetos dentro de la clase.

- ✓ Utiliza la lista de inicialización del constructor para construir los objetos.
- ✓ Se aplica cuando se quieren las características de una clase existente
- ✓ Dinámica: se define en tiempo de ejecución.
- ✓ Los objetos adquieren referencias de otros objetos.
- ✓ Hay menos dependencias de implementación.

2.4.4. Desventajas

- ✓ No utiliza la interfaz de la clase que lo compone.
- ✓ Reutilización de *caja negra*: no hay visibilidad de los aspectos internos de los objetos (objetos como cajas negras).
- ✓ Los objetos tienen que respetar las interfaces de los otros objetos.
- ✓ Requiere interfaces bien definidas.
- ✓ Habrá más objetos en el sistema y por tanto el comportamiento del sistema dependerá de las interacciones entre objetos en vez de estar definido en una clase.

2.5. Herencia

En orientación a objetos la herencia es el mecanismo fundamental para implementar la reutilización y extensibilidad del software. A través de ella los diseñadores pueden construir nuevas clases partiendo de una jerarquía de clases ya existente (comprobadas y verificadas) evitando con ello el rediseño, la remodificación y verificación de la parte ya implementada. La herencia facilita la creación de objetos a partir de otros ya existentes, obteniendo características (métodos y atributos) similares a los ya existentes.

La herencia es uno de los mecanismos de la programación orientada a objetos, por medio del cual una clase se deriva de otras, a la clase ya existente se le llama

superclase, clase base o clase padre y a la nueva clase se le llama subclase, clase derivada o clase hija.

Java permite el empleo de la herencia, característica muy potente que permite definir una clase tomando como base a otra clase ya existente. Esto es una de las bases de la reutilización de código, en lugar de copiar y pegar.

En java, la herencia se especifica agregando la cláusula extends después del nombre de la clase. En la cláusula extends se indica el nombre de la clase base de la cual se va a heredar.

Al heredar de una clase base, se hereda tanto los atributos como los métodos, mientras que los constructores son utilizados, pero no heredados.

2.5.1. Reglas

Se heredan todos los miembros (atributos y métodos), aunque sólo son accesibles los declarados public o protected, en caso de no tener calificador de acceso es posible el acceso si la subclase está declarada en el mismo directorio.

No se hereda un miembro de la superclase si la subclase declara un miembro idéntico (sobrescritura).

Los miembros declarados private se heredan aunque sin posibilidad de acceso

Las clases con el modificador final no pueden ser heredadas

2.5.2. Acceso

- ✓ Private: No hay posibilidad de acceso directo por lo que se eliminan los efectos laterales y dependencias entre clases. Por el contrario, exige añadir métodos para el acceso indirecto.

- ✓ Protected: Permite el acceso directo sólo a las subclases, esto es útil a veces, aunque añade dependencias en la jerarquía de clases.
- ✓ Public: Facilita el acceso directo de todas las clases, lo que por norma es peligroso.
- ✓ Sin modificador: Público en el mismo directorio y privado entre distintos directorios.

2.5.3. Tipos

Especialización. Dado un concepto B y otro concepto A que representa una especialización de A, entonces puede establecerse una relación de herencia entre las clases de objetos que representan a A y B. En estas situaciones, el enunciado "A es un B" suele ser aplicable.

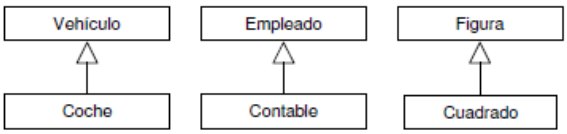


Figura II. 16. Especialización

Fuente: http://www.slideshare.net/mejiaff/java-orientado-a-objetos

Extensión. Una clase puede servir para extender la funcionalidad de una superclase sin que represente necesariamente un concepto más específico.

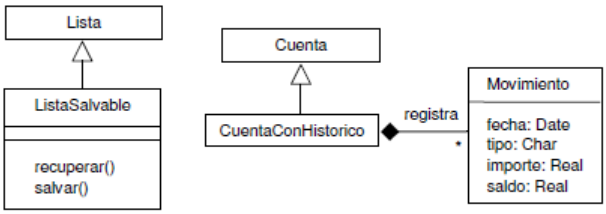


Figura II. 17. Extensión

Fuente: http://www.slideshare.net/mejiaff/java-orientado-a-objetos

Especificación. Una superclase puede servir para especificar la funcionalidad mínima común de un conjunto de descendientes. Existen mecanismos para obligar a la implementación de una serie de operaciones en estos descendientes.

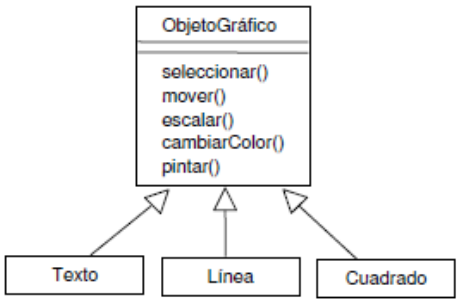


Figura II. 18. Especificación

Fuente: http://www.slideshare.net/mejiaff/java-orientado-a-objetos

Construcción. Una clase puede construirse a partir de otra, simplemente porque la hija puede aprovechar internamente parte o toda la funcionalidad del padre, aunque representen entidades sin conexión alguna.

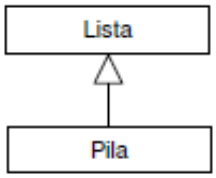


Figura II. 19. Construcción

Fuente: http://www.slideshare.net/mejiaff/java-orientado-a-objetos

Múltiple. No es soportada por Java, pero se la puede simular mediante la cláusula **implements** e **interfaces**, que sirve para implementar o cubrir una clase con respecto a otra.

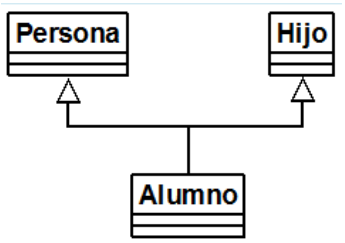


Figura II. 20. Múltiple

Fuente: http://www.slideshare.net/mejiaff/java-orientado-a-objetos

Para ilustrar mejor la figura 20 a continuación se mostrará cómo sería el código de la herencia múltiple en la figura 21.

```
public class Persona {

}


public class Hijo{
}
interface iHijo{
//La interface pertenece a la clase Hijo
}


public class Alumno extends Persona implements iHijo{
}
```

Figura II. 21. Código de herencia múltiple

Fuente: http://www.slideshare.net/mejiaff/java-orientado-a-objetos

2.5.4. Clases Abstractas

En ciertos casos, una clase se diseña directamente para ser extendida por un conjunto de subclases. En estos casos suele interesar no implementar alguno de sus métodos, pues no tiene ningún significado en la clase base.

Una de las características más útiles de cualquier lenguaje orientado a objetos es la posibilidad de declarar clases que definen como se utiliza solamente, sin tener

que implementar método. Esto en java se hace mediante interfaces y con clases abstractas.

Una clase abstracta es una clase de la que no se puede crear objetos. La utilidad de estas clases estriba en que otras clases hereden de ésta, por lo que con ello se puede conseguir reutilizar código. Para declarar una clase como abstracta se utiliza la palabra clave abstract.

Una clase que declara la existencia de métodos pero no la implementación de dichos métodos (o sea, las llaves { } y las sentencias entre ellas), se considera una clase abstracta. Una clase abstracta puede contener métodos no-abstractos pero al menos uno de los métodos debe ser declarado abstracto.

Para declarar una clase o un método como abstractos, se utiliza la palabra reservada **abstract**.

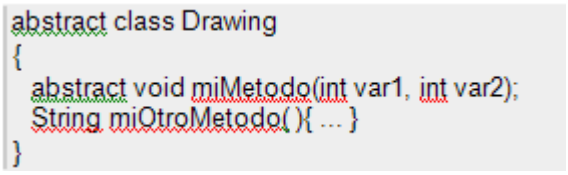


Figura II. 22. Ejemplo de clase abstracta

Fuente: http://www.slideshare.net/mejiaff/java-orientado-a-objetos

Una clase abstracta no se puede instanciar pero si se puede heredar y las clases hijas serán las encargadas de agregar la funcionalidad a los métodos abstractos. Si no lo hacen así, las clases hijas deben ser también abstractas.

2.5.5. Ventajas

- ✓ Modelado de la realidad. Son frecuentes las relaciones de especialización/generalización entre las entidades del mundo real, por

tanto es lógico que se disponga de un mecanismo similar entre las clases de objetos.

- ✓ Evitar redundancias. Toda la funcionalidad que aporta una clase de objetos es adoptada de manera inmediata por la clase que hereda, por tanto se quiere evitar la repetición de código entre clases semejantes.
- ✓ Facilitar la reutilización. Una clase no tiene por qué limitarse a recibir una serie de características de otra clase por herencia de forma pasiva. También disponen de cierto margen de adaptación de estas características.
- ✓ Soporte al polimorfismo.
- ✓ Reutilización de *caja blanca*: los aspectos internos de la superclase son visibles a las subclases.
- ✓ Soportada por el lenguaje de programación.
- ✓ Estática: se define en tiempo de compilación.

2.5.6. Desventajas

- ✓ El Lenguaje Java no soporta herencia múltiple.
- ✓ La herencia no permite cambios en tiempo de ejecución.
- ✓ La herencia rompe la encapsulación.
- ✓ La herencia impone al menos una parte de la representación física a las subclases.
- ✓ Cambios en la superclase pueden afectar a las subclases.
- ✓ Las implementaciones de superclase y subclases están ligadas.
- ✓ Si hacen falta cambios para reutilizar una clase en nuevos dominios de aplicación habrá que cambiarla.
- ✓ Limita la flexibilidad y al final la reutilización.

2.6. Plan Operativo Anual (POA)

El programa operativo es un programa concreto de acción de corto plazo, que emerge del plan de largo plazo, y contiene los elementos (objetivo, estrategia, meta y acción) que permiten la asignación de recursos humanos y materiales a las acciones que harán posible el cumplimiento de las metas y objetivos de un proyecto específico.

Los programas operativos se confeccionan en términos de unidades físicas de producto final o volumen de trabajo. Calculando los costos sobre los resultados esperados y distribuyendo los recursos financieros necesarios por partidas, según el objeto del gasto (clasificación contable), para solventar los costos a través de un presupuesto.

Los costos acumulados de cada proyecto darán como resultado el costo total de cada proyecto, y sucesivamente de programa y función, así como el costo total de cada dependencia; proporcionando con esta valoración los elementos necesarios para la toma de decisiones respecto de la distribución de recursos que realizan los órganos de gobierno de la universidad, ya que se conoce la magnitud de las erogaciones presupuestarias en todos y cada uno de los niveles, así como de las unidades a las que se destinan los recursos.

Esta herramienta de planeación, organización y control de nuestras actividades cotidianas, ofrece en corto plazo la certidumbre de las acciones a realizar; la despolitización de la misma; claridad en la relación costo-beneficio; hace posible el seguimiento del avance de metas y la participación en bolsas de recursos para los proyectos que trabajan con mayor eficiencia.

Los resultados del Plan Operativo Anual (POA) deben contribuir a la consolidación de la Planificación Estratégica (PE).

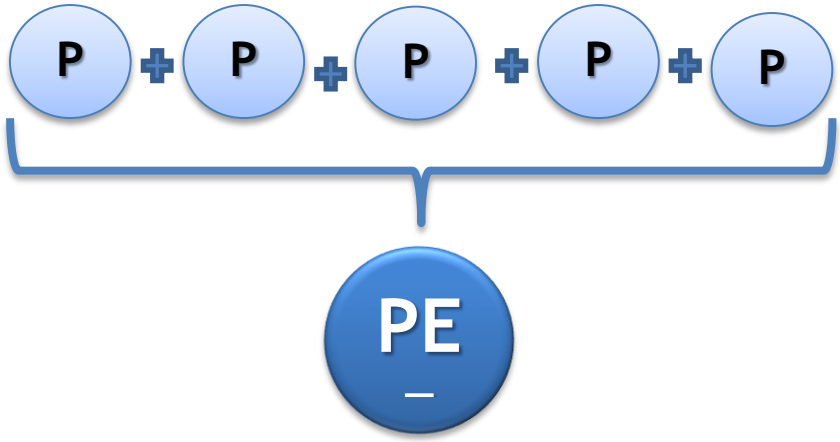


Figura II. 23. Diagrama de planificación estratégica

Fuente: Autores

2.6.1. Objetivos del POA

Los objetivos del POA son:

- ✓ **Uniformar** la conceptualización y presentación de los programas de trabajo, permitiendo realizar estudios comparativos entre las actividades que realizan las diferentes unidades administrativas.
- ✓ **Evaluar** los beneficios y los costos de cada programa, permitiendo con ello fijar prioridades de acción.
- ✓ **Estudiar** el grado de compatibilidad y consistencia interna de cada programa, a través de la relación de las metas cualitativas y cuantitativas con el resultado final de las tareas realizadas.
- ✓ **Establecer coeficientes de rendimiento** de los recursos para medir la eficiencia con que se utilizan y determinar la consistencia entre las metas y los medios empleados para lograrlas.

- ✓ **Facilitar la coordinación** entre la planeación operacional (de corto plazo), con el Plan Institucional de Desarrollo (de mediano y largo plazo).
- ✓ **Identificar** y medir los costos de los resultados finales, tanto unitariamente como a nivel general, facilitando la integración del presupuesto anual.
- ✓ **Desarrollar una herramienta** que facilite la contabilización de los gastos y la generación de estados financieros que permitan la evaluación económica y global de la institución.

2.6.2. Lineamientos de los POAs

Para poder verificar la congruencia debe existir entre los planes operacionales anuales y el Plan Institucional de Desarrollo, es necesario especificar las normas que permitirán la evaluación de los resultados.

- ✓ Debe definirse las funciones sustantivas de la institución, así como la estructura organizacional que permita su realización.
- ✓ Es indispensable describir las políticas que se implementarán para orientar al cumplimiento de las funciones.
- ✓ Presentar el resumen de la clasificación económica del gasto, analizando la distribución global entre el gasto operacional, las inversiones o las transferencias para verificar su congruencia con las políticas de austeridad o de desarrollo y del cumplimiento de las funciones sustantivas.
- ✓ Definir la estructura programática, que permita el logro de los objetivos institucionales, de acuerdo a los pasos siguientes:
    - o Describir brevemente los objetivos generales de mediano y largo plazo de la institución.
    - o Determinar las relaciones de agrupamiento y subordenación entre las funciones sustantivas, programas y proyectos.

- Explicitar en el nivel más analítico (proyecto) las acciones que permitan alcanzar los objetivos de corto plazo y que representan avances del mediano y largo plazo. Es conveniente distinguir los programas de acuerdo a la clasificación económica del gato (operacionales, inversión y transferencias).
- Numerar y subordenar los conceptos de la estructura.

✓ Proyectos deben detallarse, indicando sus componentes principales como son:

- Su identificación (de acuerdo a la estructura programática) y relación con las funciones sustantivas de la institución.
- Asignación de responsabilidad, con base en el identificador de unidades de la estructura organizacional.
- Los objetivos específicos y la estrategia a utilizar para lograrlos.
- Las metas establecidas en función del resultado final deseado, que se pretenden alanzar en el futuro inmediato.
- Las acciones que hay que realizar, para dar cumplimiento a las metas y objetivos, así como su calendarización; indicando la unidad de medida del resultado final y estableciendo en términos cuantitativos su magnitud.
- Los recursos necesarios para llevar a cabo las acciones y el tiempo en que serán requeridos.

✓ Resumir la estructura global de los costos totales de los proyectos, realizando clasificaciones cruzadas de proyectos y objeto de gasto, así como de los costos y las unidades responsables; para analizar tanto la

incidencia de los diferentes conceptos de gasto, como la apreciación del consumo de recursos de cada proyecto y de cada unidad responsable.

- ✓ Asignar los techos financieros, como producto del análisis de la relación entre los planes institucionales, la disponibilidad de recursos y la definición de prioridades.
- ✓ Facilitar la ejecución de las acciones, mediante la disponibilidad de los recursos (humanos, materiales, sistemas, tecnología y liderazgo), que se hayan comprometido.
- ✓ Establecer los métodos de reportes de actuación para darle seguimiento al cumplimiento de las metas contra lo planeado, tanto cualitativa como cuantitativamente.
- ✓ Definir los parámetros de medición de las acciones para supervisar y evaluar la relación costo – beneficio de cada proyecto, así como la eficiencia del trabajo de los recursos humanos y de las unidades responsables en su conjunto.
- ✓ Informar a planificadores y directivos los resultados del proceso de control, para que tomen decisiones respecto a las desviaciones y reorientación de actividades.
- ✓ Globalizar la información para analizar y evaluar el desarrollo de la institución.

2.6.3. El Ciclo Presupuestario

El ciclo presupuestario es un proceso que permite elaborar, ejecutar y controlar la asignación presupuestal a las diversas actividades que realiza la institución para dar cumplimientos a sus objetivos.

Este ciclo comprende las siguientes etapas:

Formulación

La formulación inicia con la programación presupuestaria, que es el planteamiento periódico, (generalmente anual) que permite definir los componentes concretos de las actividades que se pretende realizar en el periodo inmediato de operaciones.

Este planteamiento implica:

- ✓ Definir los niveles de actividad, que es la forma de vincular el plan de corto plazo con el de largo plazo de la institución, y debe provenir de una instrucción específica de las autoridades.
- ✓ Formular en cada unidad administrativa un programa de trabajo de acuerdo a criterios de prioridad, límites de tiempo, calidad y costo.
- ✓ Supervisar los programas de trabajo por parte de los niveles de autoridad superiores, siendo para ello necesario, que el programa contenga un resumen, que permita un juicio global acerca de su contenido.
- ✓ Concentrar los programas de diversas unidades administrativas por dependencias, para evaluar la participación de estas últimas en el plan global de la institución, la evaluación mencionada servirá para visualizar contradicciones, cuellos de botella, necesidades de ampliación o reducción de proyectos, reclasificaciones funcionales o coordinación de las actividades de organismos conexos.
- ✓ Por último presentar al más alto nivel de ejecución el plan global para su sanción y autorización.

Discusión y Aprobación

Si bien es cierto que en el proceso de formulación del presupuesto intervienen todos y cada uno de los niveles jerárquicos de la institución, corresponde a los directivos de mayor nivel su discusión y aprobación.

Es importante que los participantes en esta revisión tengan una actitud crítica pero a la vez comprensiva y que conozcan profundamente los problemas de la institución, ya que el objetivo de esta actividad es el enriquecimiento del proyecto presupuestario, aportando ideas que clarifiquen las necesidades reales que pretende solventar la institución.

El proceso de discusión debe estar organizado de tal manera que se limiten los excesos que pretendan desarticular el presupuesto, ó desvirtuar las actividades prioritarias de dependencias contrarias, es por ello necesario contar con defensores que tengan pleno conocimiento de los problemas presupuestarios y conozcan las consecuencias de los proyectos que plantean. Pero sobre todo debe existir una plena conciencia de que el plan global es un todo y no la suma de proyectos aislados.

Ejecución

Una vez autorizado el programa presupuestario se procede a ejecutarlo: iniciando con los ajustes ó reformulaciones presupuestales aprobadas; enseguida se realizan las asignaciones presupuestales que estarán condicionadas por las fuentes de ingreso disponibles; se procede a la disposición de recursos para poder realizar las actividades por medio de solicitudes de uso de recursos, mismas que pasaran por un proceso de revisión y autorización: primero del departamento presupuestal para verificar que el recurso haya sido programado. Y

segundo por los responsables de la administración financiera para comprobar la disponibilidad de los recursos.

Control

El control es el examen detallado de las actividades realizadas para verificar su eficiencia legalidad y contribución a los fines institucionales, el cual se efectúa por medio de las herramientas de contabilidad y auditoría.

La contabilización es el registro sistemático de las disposiciones de recursos que genera cada unidad administrativa, respecto a cada proyecto y cuya acumulación y estructuración originará reportes que permitirán el análisis de la gestión administrativa de todos los ejecutantes del presupuesto.

La contabilidad debe estar estructurada de tal manera que sea compatible con la presupuestación; que permita determinar la responsabilidad de las unidades administrativas; que propicie la medición de los costos de las actividades y proyectos y de conocer resultados financieros que faciliten la administración de la institución.

La revisión de los registros contables se llaman auditoría y puede efectuarse desde el interior ó exterior de la institución pero en ambos casos ofrecerá información que permita tomar medidas correctivas respecto a las desviaciones encontradas.

Evaluación

Es evidente que la planificación, programación y presupuestación encuentran su razón de ser en el proceso de relacionar los objetivos institucionales con los recursos disponibles para obtener resultados que permitan evaluar las acciones,

servicios ó productos que genera la institución, prestando mayor atención a los logros que al consumo de bienes que sólo son los medios para el cumplimiento de las funciones.

Esta presupuestación por programas y actividades que ha obligado a las unidades administrativas a formular programas específicos de trabajo y a comprometerse al cumplimiento de ciertas metas, debe incluir mecanismos para asegurar que la planificación sea adecuada; precisión en la asignación de recursos; una definición precisa y equilibrada de responsabilidades y la inexistencia de duplicidades funcionales. Por último, evaluar la eficiencia con que operan las diferentes unidades administrativas que aplican el presupuesto, requiere de una serie de indicadores de rendimiento que permita la comparación entre proyectos similares contra estándares institucionales propios y/o de otros organismos. Pero siempre el propósito esencial de toda evaluación será medir los resultados y estos en una institución universitaria están vinculados al grado de cumplimiento de nuestra misión académica.

2.7. Plan Anual de Inversión (PAI)

Es un elemento integral del sistema presupuesta, que tiene por objeto determinar los programas, subprogramas y proyectos de inversión a ejecutar durante la vigencia fiscal (contado el año a partir del 1ero de enero hasta el 31 de diciembre), también se lo considera como un instrumento de gestión que permite operacionalizar los objetivos y metas establecidas en el Plan de Desarrollo municipal, departamental o distrital, para cada vigencia fiscal.

Este instrumento debe ser programado con todos los recursos que van a ser destinados a inversión y debe precisar cada una de las fuentes de financiación del

plan, en especial, las de destinación específica para inversión, como el caso de los asignados por cada componente del Sistema General de Participaciones.

Es el instrumento mediante el cual se concretan las inversiones del Plan de Desarrollo, de acuerdo con las formas de financiación contenidas en el Plan Financiero. Señalará los proyectos de inversión clasificados por sectores, entidades y programas, que guardan concordancia con el Plan de Inversiones establecido en el Plan de Desarrollo. Se realiza la anualización para una vigencia fiscal de las inversiones del mediano y corto plazo contenidas en el Plan de Desarrollo. Por consiguiente, debe incluir los programas, subprogramas y proyectos a ejecutar durante la vigencia fiscal.

Es la base sobre la cual las distintas dependencias de la administración elaboran los planes de acción. El mismo facilita el seguimiento y la evaluación de los programas y proyectos que se van a ejecutar, y permite observar el nivel de cumplimiento de las metas fijadas en el Plan de Desarrollo en relación con la satisfacción de las necesidades básicas de la población.

CAPÍTULO III

ANÁLISIS DE LOS MÉTODOS DE REUTILIZACIÓN DE CÓDIGO

5.1. Introducción

En este capítulo se detallara los procesos, herramientas, materiales y parámetros que se utilizaron para realizar el análisis comparativo de los métodos de reutilización de código *“Herencia”* y *“Composición”*, con el objetivo de demostrar de qué manera la aplicación de estos métodos facilitará la optimización de desarrollo de un sistema informático en referencia a la disminución del tiempo, esfuerzo, líneas de código y dificultad, así como una mejora en el desempeño en el trabajo de los prototipos.

El análisis fue realizado mediante la implementación de tres prototipos de un módulo del sistema informático, estos prototipos fueron desarrollados de la siguiente manera: el primero fue desarrollado en su totalidad sin utilizar ningún método de reutilización, el segundo y tercer prototipo se utilizaron para su desarrollo la herencia y la composición respectivamente.

Cada uno de ellos fue desarrollado bajo la utilización del IDE Netbeans 7.3, una vez desplegados cada modelo, para determinar su tiempo de desarrollo, esfuerzo, dificultad se utilizó fórmulas matemáticas ya establecidas en la herramienta COCOMO, para la medición del uso de CPU y de memoria RAM del servidor se utilizó la herramienta NEOLOAD, la cual nos permite una vez desplegada la aplicación en el servidor, medir sus tiempos de respuesta, porcentajes de uso de los recursos del servidor y para el conteo de líneas de código se utilizó la herramienta Notepad++.

5.2. Prototipos de Prueba

Un prototipo es básicamente un ejemplar desarrollado para la demostración funcional de un sistema más grande y complejo, los prototipos utilizados para este estudio, contienen un módulo funcional: La Administración de Proyectos de la Unidad de Planificación de la ESPOCH.

Se realizará una descripción de cada prototipo utilizado indicando que método se utilizó y cuál será su objetivo y contribución al estudio realizado, resaltando que los objetivos de cada prototipo son los mismos ya que los tres fueron creados para que realicen la misma actividad, con los mismos componentes de interfaz gráfica.

5.2.1. Prototipo 1: sin ningún método de reutilización de código

El Prototipo 1, tiene como función el realizar la gestión de información de proyectos para la generación del POA, este prototipo fue desarrollado sin utilizar ningún método de reutilización de código, tanto la funcionalidad como la interfaz de usuario es el mismo que se aplicará a los demás prototipos.

Objetivo:

Su principal objetivo dentro de este estudio es obtener valores referentes al tiempo de desarrollo, esfuerzo usado, dificultad, cantidad de líneas de código, porcentaje de uso de CPU y porcentaje de uso de memoria RAM del servidor, estos valores servirán de base para la comparación de los resultados obtenidos con los prototipos que utilizan la herencia y la composición para determinar de qué manera contribuye su uso en la mejora de optimización del sistema.

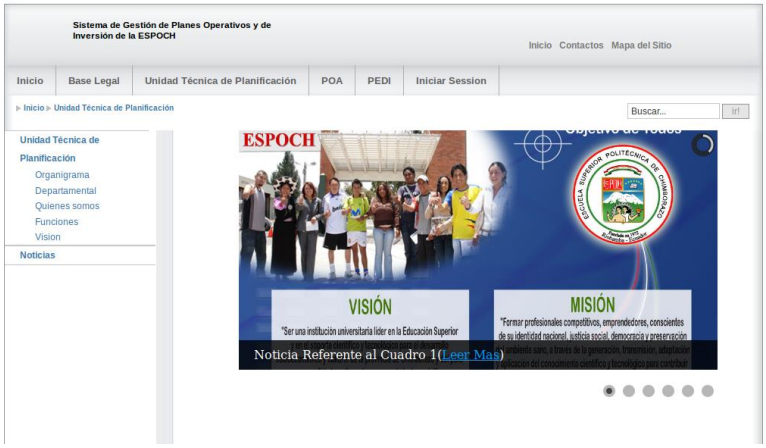


Figura III. 24. Pantalla Principal Prototipo 1

Fuente: Autores

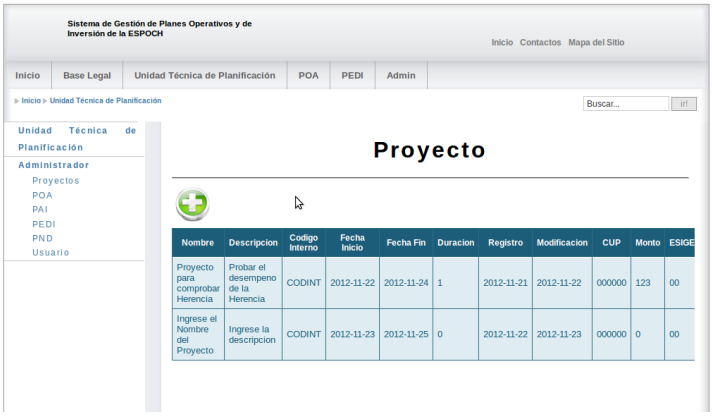


Figura III. 25. Pantalla Proyectos

Fuente: Autores

Figura III. 26. Pantalla Ingreso de Proyectos

Fuente: Autores

Clases Utilizadas

Las clases java utilizadas para la elaboración de este prototipo y que están directamente relacionadas son:

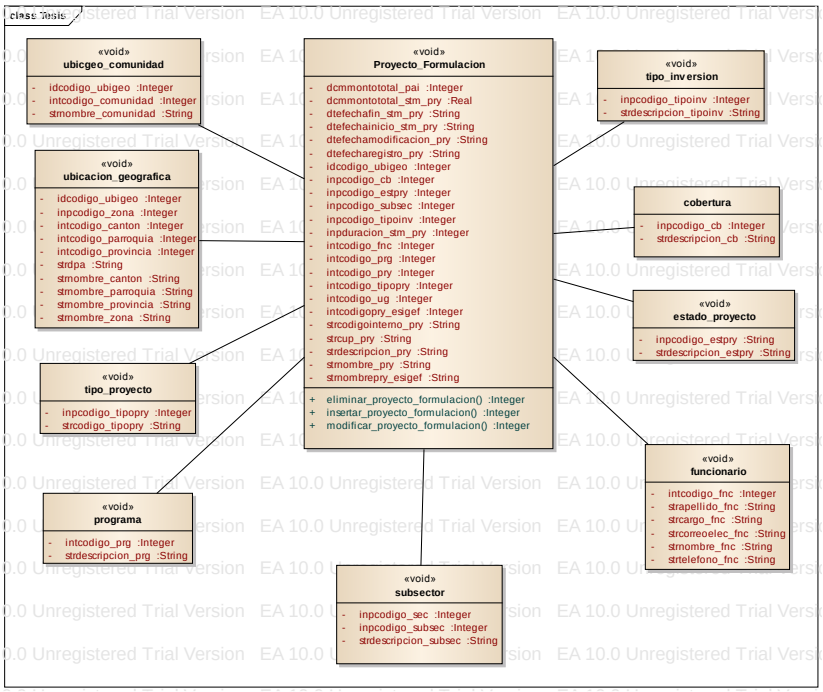


Figura III. 27. Clases Utilizadas Prototipo 1

Fuente: Autores

Donde se puede apreciar que cada clase tiene sus propios atributos, los mismos que pertenecen solo a la clase designada.

5.2.2. Prototipo 2: con composición

El Prototipo 2, tiene como función el realizar la gestión de información de los proyectos para la generación del POA, este prototipo fue desarrollado utilizando el método de reutilización de código llamada composición, tanto funcionalidad como interfaz de usuario es el mismo que se aplicará a los demás prototipos.

Objetivo:

Su principal objetivo dentro de este estudio es servir para la determinación de tiempos, esfuerzo, dificultad, cantidad de líneas de código, porcentaje de uso de RAM y de CPU del servidor del prototipo utilizando la composición como método de reutilización de clase, aquí se puede obtener valores que nos permitirán

realizar una comparación con el prototipo 1 para determinar de qué manera contribuye este método a la optimización del desarrollo del sistema.

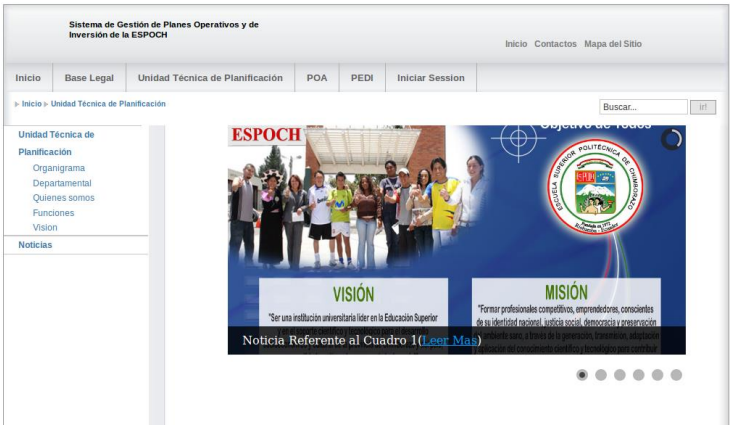


Figura III. 28. Pantalla Principal Prototipo 2

Fuente: Autores

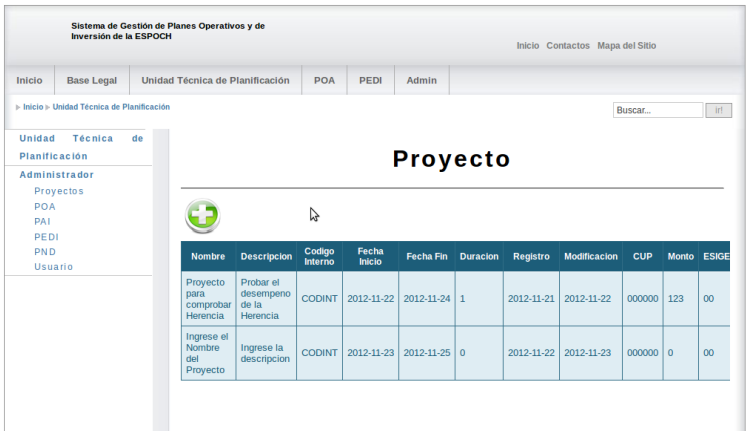


Figura III. 29. Pantalla Proyectos Prototipo 2

Fuente: Autores

Figura III. 30. Pantalla Ingreso de Proyectos del Prototipo 2

Fuente: Autores

Clases Utilizadas

Las clases java utilizadas para la elaboración de este prototipo y que están directamente relacionadas son:

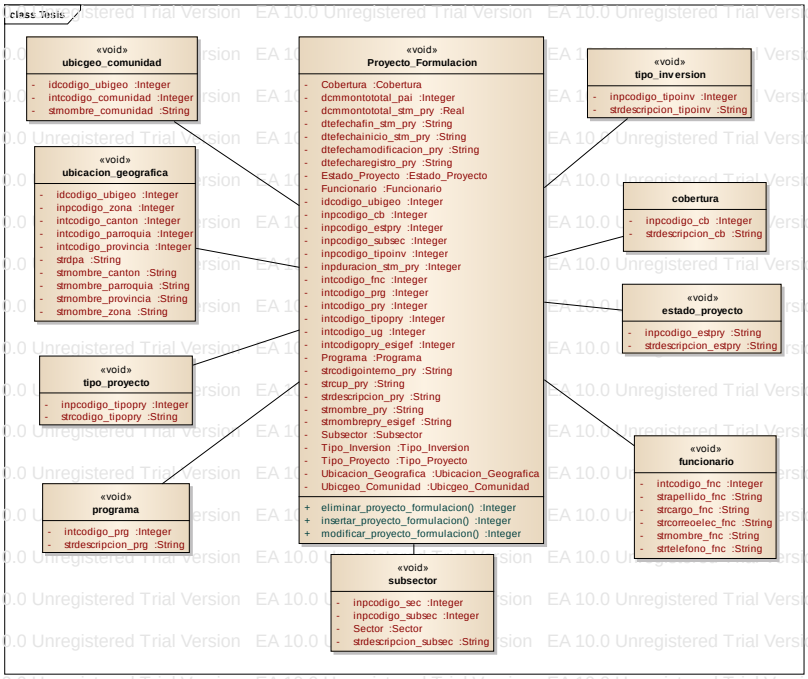


Figura III. 31. Clases Utilizadas Prototipo 2

Fuente: Autores

Donde se puede apreciar que cada clase tiene sus propios atributos, los mismos que pertenecen solo a la clase designada y que la clase principal posee los objetos compuestos que necesitan para completarse.

5.2.3. Prototipo 3: con herencia

El prototipo 3, tiene como función el realizar la gestión de información de los proyectos para la generación del POA, este prototipo fue desarrollado utilizando el método de reutilización de código llamada herencia, tanto funcionalidad como interfaz de usuario es el mismo que se aplicará a los demás prototipos.

Objetivo:

Su principal objetivo dentro de este estudio es servir para la determinación de tiempos, esfuerzo, dificultad, cantidad de líneas de código, porcentaje de uso de RAM y de CPU del servidor del prototipo utilizando la herencia como método de reutilización de clase, aquí se puede obtener los valores para determinar de qué manera contribuye este método a la optimización del desarrollo del sistema.1.

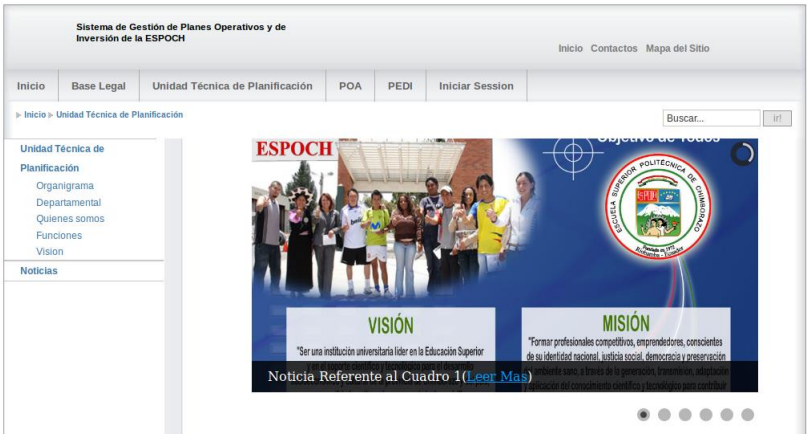


Figura III. 32. Pantalla Principal Prototipo 3

Fuente: Autores

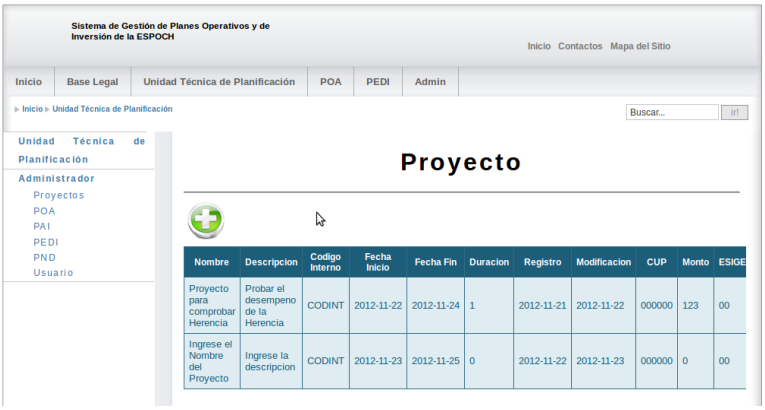


Figura III. 33. Pantalla Proyectos del Prototipo 3

Fuente: Autores

Figura III. 34. Pantalla Ingreso de Proyectos del Prototipo 3

Fuente: Autores

Clases Utilizadas

Las clases java utilizadas para la elaboración de este prototipo y que están directamente relacionadas son:

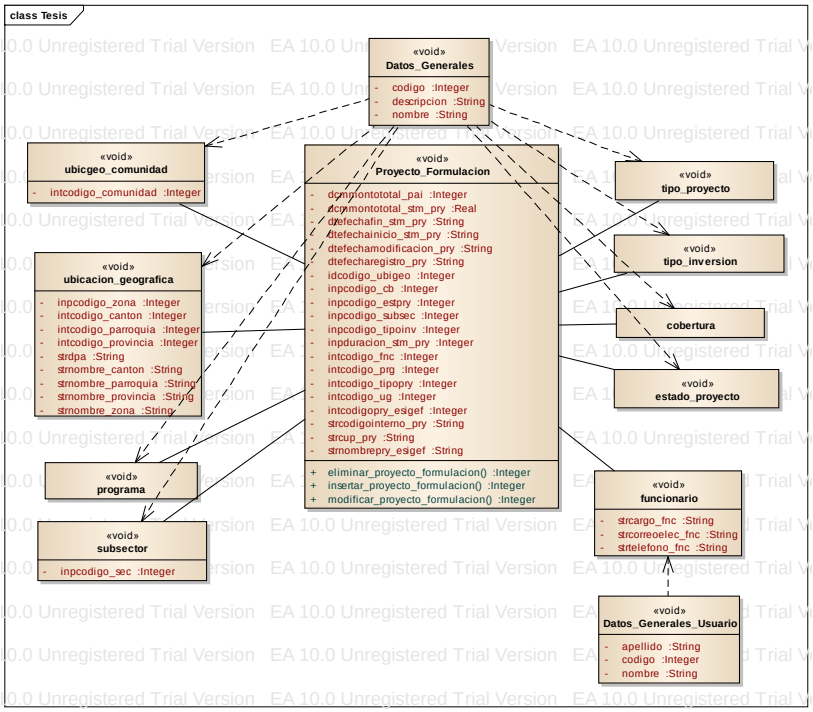


Figura III. 35. Clases Utilizadas Prototipo 3

Fuente: Autores

Donde se puede apreciar que cada clase tiene sus propios atributos, los mismos que pertenecen solo a la clase designada y que los datos comunes de cada clase se han incluido en una sola clase llamada Datos Generales que serán heredados por las demás clases, de igual forma se creó una clase llamada Datos Generales Usuario que poseen datos comunes para la herencia de otras clases.

5.3. Parámetros de análisis

Uno de los estamentos más importantes es realizar un correcto análisis de los parámetros que se han establecido para comprobar la hipótesis, para ello es necesaria la utilización de fórmulas, herramientas y la determinación de que parámetros de comparación serán los que se utilicen en el análisis.

Los parámetros que serán analizados son:

Tiempo: Referido al tiempo aproximado de desarrollo de la aplicación para determinar el tiempo se utilizará las fórmulas matemáticas básicas del COCOMO.

Dificultad: La dificultad de un proyecto informático esta dado en función de las líneas de código, esta se obtiene determinando el tamaño en kilobytes o megabytes de la clase.

Líneas de Código: Cantidad de líneas válidas escritas es decir que aquí se deben excluir espacios blancos y comentarios para el desarrollo de la aplicación informática más conocida como código fuente. Si bien el conteo de líneas de código (LDC) para la determinación del tiempo de desarrollo de una aplicación informática es una solución que nos ofrece valores aproximados, debido a que este conteo no garantiza tomar en cuenta muchas variables del entorno de programación.

Esfuerzo: Es el trabajo desarrollado por una persona al mes. El esfuerzo depende del número de líneas de código escritas por la persona.

Porcentaje Uso del CPU: Representa el porcentaje de uso de la capacidad del CPU del servidor en determinado tiempo, este análisis se obtiene de los resultados obtenidos con la herramienta NEOLOAD 4.0.4.

Porcentaje de Uso de RAM: Representa el porcentaje de uso de la capacidad de la memoria RAM del servidor en un determinado tiempo, análisis obtenido de los resultados de la herramienta NEOLOAD 4.0.4.

5.4. Herramientas

La herramienta de software que se utilizará será el paquete de software NEOLOAD 4.0.4 en su versión trial que será la encargada de proveer

información acerca del porcentaje de uso del CPU y la memoria RAM del servidor. NEOLOAD es una herramienta de prueba de carga y estrés para cualquier aplicación web, ya que mediante la simulación de uso de parte de varios usuarios permite determinar la robustez y rendimiento de la aplicación y del servidor. Al final esta herramienta nos ofrece un reporte completo y total del análisis.

Para la comprobación de desempeño de cada prototipo se ha considerado los siguientes puntos:

- Se ha determinado un máximos de 10 usuarios virtuales para cada escenario.
- Los módulos que se utilizaron para la medición de desempeño son el módulo de usuarios y el módulo de proyectos.
- Se mide el uso de CPU, memoria RAM y uso del servidor de bases de datos Postgres.

5.4.1. Modo de uso de NEOLOAD v4

El modo de uso de la herramienta NEOLOAD v4 se especificará detalladamente en el Anexo 1 en el presente documento.

5.4.2. Configuración de la medida del desempeño

La configuración de la medida del desempeño se especificará detalladamente en el Anexo 2 en el presente documento.

5.4.3. Configuración de ejecución de pruebas

La configuración de ejecución de pruebas se especificará detalladamente en el Anexo 3 en el presente documento.

5.5. Fórmulas Numéricas

Para poder realizar el análisis de los parámetros de comparación se utilizará las fórmulas matemáticas básicas del COCOMO.

La gran mayoría de las fórmulas se encuentran basados en el uso de las líneas de código (LDC) útiles es decir que aquí se deben excluir espacios blancos y comentarios.

COCOMO nos provee de tres niveles de aplicación para el cálculo de estimaciones:

1.- Básico: Es un modelo estático que calcula de manera aproximada el esfuerzo del desarrollo del software, en función de las líneas de código empleados en la elaboración del sistema.

2.- Intermedio: De igual manera se utiliza para el cálculo de estimaciones pero introduce 15 atributos de coste que se debe tener en cuenta en el entorno de trabajo, estos atributos se utilizan para ajustar el coste nominal del proyecto ofreciendo una estimación de costos y esfuerzo más precisa.

3.- Avanzado: Este modelo puede procesar todas las características del proyecto para construir una estimación. Introduce dos características principales

- ✓ Multiplicadores de esfuerzo sensitivos a la fase. Algunas fases se ven más afectadas que otras por los atributos. El modelo detallado proporciona un conjunto de multiplicadores de esfuerzo para cada atributo. Esto ayuda a determinar la asignación del personal para cada fase del proyecto.

- ✓ Jerarquía del producto a tres niveles. Se definen tres niveles de producto. Estos son módulo, subsistema y sistema. La cuantificación se realiza al nivel apropiado, esto es, al nivel al que es más susceptible la variación.

Cada uno de estos modelos tienen a su vez tres modos de identificación del tipo de proyecto que se va a estimar estos modos son:

Modo Orgánico: En este modo un pequeño grupo de desarrolladores experimentados desarrollan software familiar para ellos, el tamaño del software puede variar desde unas pocas miles líneas de código a una docenas de miles de línea de código.

Modo Semi-Acoplado (semi-detached): Un proyecto de software intermedio en tamaño y complejidad en el cual equipos con distintos niveles de experiencia debe satisfacer requerimientos poco y medio rígidos

Modo Acoplado (detached): un proyecto de software que debe ser desarrollado dentro un conjunto estricto de hardware, software y de restricciones operativas.

Además para la determinación del tiempo se utilizarán los datos y ecuaciones siguientes:

5.6. Dificultad de un proyecto en función de sus líneas de código

En la Tabla I se mostrará la medición de las líneas de código según los parámetros del número de programadores, la duración de medición, el número de líneas de código y un breve ejemplo de donde se encuentran específicamente estos parámetros.

Tabla III. I: Dificultad de un proyecto en función de sus líneas de código

Categoría	Programadores	Duración	Líneas de Código	Ejemplo
Trivial	1	0 – 4 semanas	< 1k	Utilidad de Ordenación
Pequeño	1	1 – 6 meses	1k – 3k	Biblioteca de Funciones
Media	2 – 5	0,5 – 2 años	3k – 50k	Compilador de C
Grande	5 – 20	2 – 3 años	50k – 100k	SO pequeño
Muy Grande	100 – 1000	4 – 5 años	100k – 1M	Grande SO
Gigante	1000 – 5000	5 – 10 años	> 1M	Sistema de Distribución

Fuente: Autores

5.6.1. Coeficientes COCOMO básico

En la Tabla II se mostrará los coeficientes básicos que utiliza la herramienta de medición COCOMO para evaluar la aplicación software, estos coeficientes se obtienen de manera empírica y por lo tanto se basa en la experiencia de datos anteriores.

Tabla III. II: Coeficientes COCOMO Básico

Proyecto de Software	A	b	c	d
Orgánico	2,4	1,05	2,5	0,38
Semi-Acoplado	3,0	1,12	2,5	0,35
Empotrado	3,6	1,20	2,5	0,32

Fuente: Autores

La ecuación del esfuerzo de COCOMO es:

$$E = aLDC^{b} \qquad T = cE^{d} \qquad N = \frac{E}{T}$$

Dónde:

E es el **Esfuerzo** que está expresado en **persona x mes**. **LDC** es el número de **Líneas de Código**. **T** es el **Tiempo** de duración del desarrollo y se expresa en meses. **N** es el número de desarrolladores, **a, b, c, d** son valores pertenecientes a la tabla II de los coeficientes de COCOMO.

5.7. Prototipo 1: sin ningún método de reutilización de código

5.7.1. Determinación del tiempo de desarrollo

Para realizar el conteo de líneas de código se recurrió al uso del programa Notepad ++ en su versión 6 que nos permite realizar un conteo manual de las líneas de código de cada página y clase creada en los prototipos, para el total del conteo de líneas de código no se han tomado en cuenta líneas en blanco y comentarios, solo las líneas de código válidas.

Gestión de Proyectos

En la siguiente tabla (Tabla III) se mostrará el número de líneas de código que se empleó en la implementación de las páginas web del sin usar ningún método de reutilización de código.

Tabla III. III: Número líneas de código de páginas web prototipo 1

Gestión de Proyectos					
Páginas Web					
Proyecto	**Ingresar Proyectos**	**Procesa Ingresar Proyectos**	**Modificar Proyecto**	**Procesa Modificar Proyectos**	**Eliminar Proyectos**
117	306	84	381	78	31
Total Líneas de Código					997

Fuente: Autores

En la tabla que se muestra a continuación (Tabla IV) se mostrará el número de líneas de código que se empleó en la implementación las clases del prototipo con sin utilizar ningún tipo de métodos de reutilización.

Tabla III. IV: Número de líneas de código de clases java prototipo 1

Clases Java										
Acceso Datos	Proyecto Formulación	Cobertura	Estado Proyecto	Funcionario	Programa	Subsector	Tipo Inversión	Tipo proyecto	Ubicación Comunidad	Ubicación Geográfica
244	483	118	128	171	120	135	120	120	140	312
Total Líneas de Código										2091
Total Líneas de Código de Gestión de Proyectos										3088

Fuente: Autores

Ahora se identifica el modo de utilización de los valores de COCOMO de la tabla de coeficientes, utilizando el modo semi-acoplado ya que este proyecto es de complejidad media.

Aplicando las fórmulas del esfuerzo de COCOMO se obtendrá los siguientes valores:

Previamente se transformará el total de líneas de código a tamaño en kilobytes:

$$\frac{3088}{1000} = 3{,}09\ kbytes$$

Esfuerzo realizado:

$E = aLDC^b$ $E = 3{,}0 * 3{,}09^{1{,}12}$ $E = 10{,}61389414\ persona * mes$

Tiempo realizado

$T = cE^d$ $T = 2{,}5 * 10{,}61389414^{0{,}35}$ $T = 5{,}714735993\ meses$

Número de personas para determinar el tiempo de desarrollo

Esta ecuación se utilizará para determinar el tiempo de desarrollo de la solución, utilizando para ello el valor conocido de N = 2.

$N = \frac{E}{T}$ $T = \frac{E}{N}$ $T = \frac{10{,}61389414}{2}$ $T = 5{,}31\ meses$

Dificultad

La dificultad durante el desarrollo de un sistema informático se determina manualmente utilizando la tabla de valores ya establecida:

Tabla III. V: Tabla de valores de dificultad prototipo 1

Categoría	Programadores	Duración	Líneas de Código	Ejemplo
Trivial	1	0 – 4 semanas	< 1k	Utilidad de Ordenación
Pequeño	1	1 – 6 meses	1k – 3k	Biblioteca de Funciones
Media	2 – 5	0,5 – 2 años	3k – 50k	Compilador de C
Categoría	**Programadores**	**Duración**	**Líneas de Código**	**Ejemplo**
Grande	5 – 20	2 – 3 años	50k – 100k	SO pequeño
Muy Grande	100 – 1000	4 – 5 años	100k – 1M	Grande SO

Fuente: Autores

El tiempo de desarrollo de este sistema tomará un tiempo aproximado de 5,31 meses y un valor establecido de 3,09 Kb aproximadamente por lo que se puede determinar que la dificultad es **MEDIA**.

5.7.2. Pruebas de desempeño

La herramienta NEOLOAD nos permite la generación de un reporte detallado acerca de los resultados obtenidos, estos resultados se generan en inglés por lo que para la realización de este reporte se traducirá textualmente todo el contenido del informe.

Tabla III. VI: Prueba desempeño prototipo 1

Proyecto	**Pruebas_Sin_Métodos_Reutilización**	**Políticas de Carga**	**Una cantidad de 10 usuario virtuales constantes.**
Escenario	escenario1		
Fecha de Inicio	22-nov-2012 10:30:02	Descripción	Para medir el desempeño de una aplicación con Composición
Fecha de Fin	22-nov-2012 10:32:17		
Duración	00:02:15	Filtros	Ninguno
Host	Localhost	Debug	Deshabilitado

Fuente: Autores

Resumen de estadísticas:

Tabla III. VII: Resumen de estadísticas prototipo 1

Total Páginas	**40**	**Media de Páginas**	**0,3**
Total Hits	860	Media de Hits	2,8
Total Usuarios en Uso	10	Media del Tiempo de Request	3,34 segundos
Total Iteraciones Completas	10	Media del Tiempo de Respuesta	28,1 segundos
Total de Paso de MB	80,35 Mb/s	Media de Paso de MB	1,38 Mb/s
Total Hits de Errores	60	Porcentaje de Error	17,6%
Total de Errores en la Acción	0	Total duración de Alertas	95,6%

Fuente: Autores

En la Tabla VII se detallan campos que deben ser explicados para una total comprensión del trabajo realizado:

Total páginas: Es el número total de veces que las páginas fueron visitadas por los usuarios del escenario.

Total usuarios en uso: Se refiere al número de usuarios virtuales usados durante las pruebas de desempeño de los prototipos en este caso se mantiene constante en 10 usuarios.

Total de paso de MB: indica el número total de MB retornados por el servidor, en este caso es para los 10 usuarios que intervinieron en la simulación.

Total de errores: Este punto indica la cantidad de errores que se registraron durante la simulación de uso de los prototipos, este punto muestra la cantidad total errores mostrados por los 10 usuarios, estos errores corresponden a imágenes y animaciones swf.

Total de errores en la acción: Muestra la cantidad de errores ocurridos durante el proceso de gestión (ingreso, modificación y eliminación) de los proyectos y del usuario.

Media de tiempo de solicitud (Request): Promedio del tiempo de solicitud al servidor.

5.7.3. Porcentaje de uso de CPU

Esta medida se hace con el propósito de indicar el porcentaje de uso en lo referente a la carga del CPU.

Tabla III. VIII: Porcentaje de uso de CPU en el Prototipo 1

Tiempo (s)	% Uso de CPU	Tiempo (s)	% Uso de CPU
1	53.0	**70**	11.0
5	78.0	**75**	27.0
10	51.0	**80**	18.0
15	22.0	**85**	7.0
20	11.0	**90**	25.0
25	19.0	**95**	10.0
30	16.0	**100**	24.0
35	12.0	**105**	7.0
40	12.0	**110**	20.0
45	10.0	**115**	13.0
50	18.0	**120**	38.0
55	17.0	**125**	9.0
60	9.0	**130**	14.0
65	22.0	**135**	7.0

Fuente: Autores

Como se puede apreciar en la tabla las medidas tomadas corresponden a un tiempo total de 2 minutos y 15 segundos, que es el tiempo total por usuario que la herramienta utiliza para las pruebas, se puede apreciar que al inicio de la aplicación hay un aumento en el uso del CPU que se sitúa en un 53% y se mantiene en niveles altos por lo menos 10 seg., para luego mantenerse a un nivel bajo constante de 18,0 %.

Tabla III. IX: Resumen estadístico uso de CPU prototipo 1

Mínimo	Media	Máximo	Mediana	Desviación Standard
4	18,0	78	15	13

Fuente: Autores

En seguida se muestra un gráfico estadístico del porcentaje de uso del CPU en la medición del prototipo en el que no se utilizó ningún método de reutilización de código.

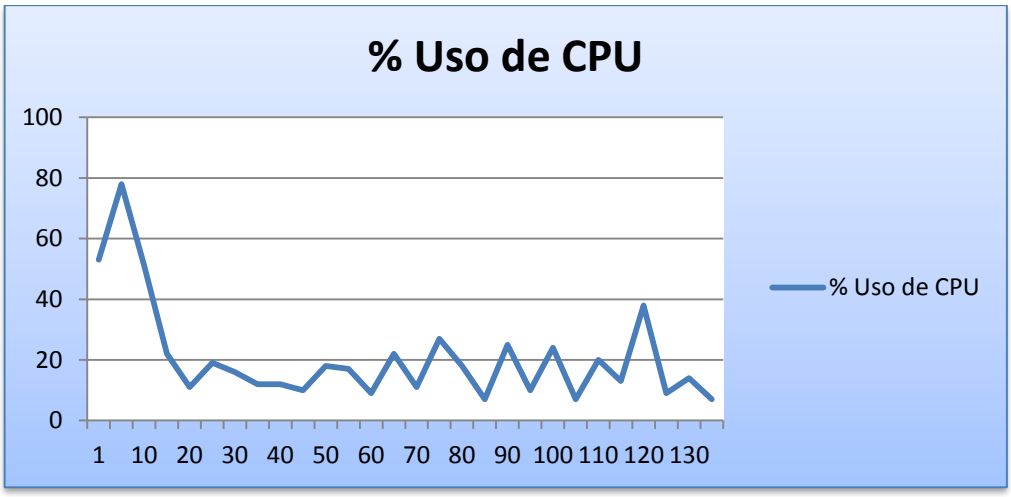


Figura III. 36. Porcentaje de uso de CPU en el prototipo 1

Fuente: Autores

5.7.4. Porcentaje de uso de memoria RAM

Medidas realizadas para indicar el porcentaje de uso de la memoria durante las pruebas del prototipo sin ningún tipo de método de reutilización de código.

Tabla III. X: Porcentaje de uso de memoria RAM Prototipo 1

Tiempo (s)	% Uso de Memoria	Tiempo (s)	% Uso de Memoria
1	12.0	**70**	9.0
5	16.0	**75**	9.0
10	8.0	**80**	9.0
15	8.0	**85**	9.0
20	8.0	**90**	10.0
25	8.0	**95**	9.0
30	7.0	**100**	9.0
35	9.0	**105**	10.0
40	9.0	**110**	9.0
45	9.0	**115**	10.0
50	8.0	**120**	10.0
55	8.0	**125**	11.0
60	9.0	**130**	11.0
65	8.0	**135**	14.0

Fuente: Autores

Como se puede apreciar en la tabla las medidas tomadas corresponden a un tiempo total de 2 minutos y 15 segundos, que es el tiempo total por usuario que la herramienta utiliza para las pruebas, se puede apreciar que el uso de la memoria se mantiene en un porcentaje promedio del 8,67 % y que se mantiene por debajo del 16% del porcentaje total de memoria RAM del servidor.

Tabla III. XI: Resumen estadístico uso de RAM en el prototipo 1

Mínimo	Media	Máximo	Mediana	Desviación Standard
7	8,67	16	9	1,54

Fuente: Autores

De la misma manera se presentará un gráfico estadístico indicando el porcentaje de uso de memoria con el prototipo que no utiliza ningún método de reutilización de código.

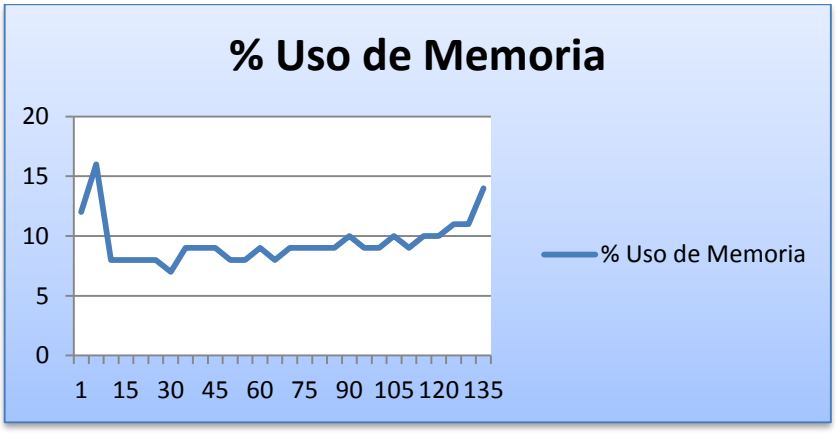


Figura III. 37. Porcentaje de uso de memoria RAM en el prototipo 1

Fuente: Autores

Tiempo medio de respuesta por página

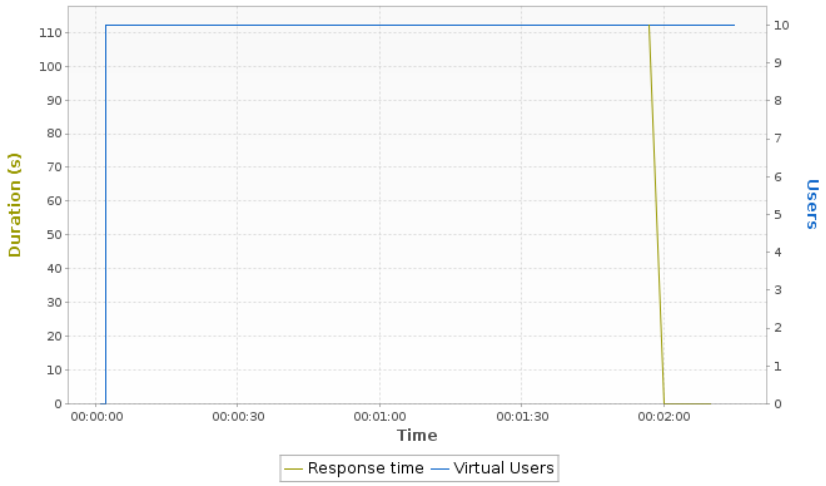


Figura III. 38. Tiempo de respuesta por página prototipo 1

Fuente: Autores

A continuación se mostrará las medidas estadísticas acerca del tiempo de respuesta en el caso de la medición del prototipo desarrollado sin ningún tipo de método de reutilización de código.

Tabla III. XII: Valores de respuesta por página prototipo 1

Valor Mínimo	Media	Valor Máximo	Mediana	Desviación Standard
0,01	38,397	112,162	0,019	53,77

Fuente: Autores

Tiempo de respuesta media de solicitudes (Request)

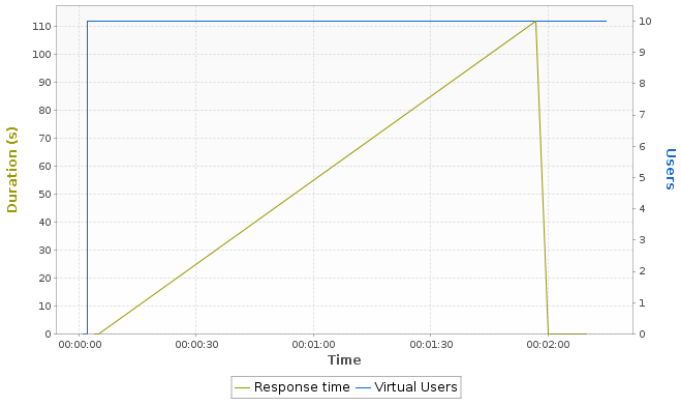


Figura III. 39. Valores de respuesta por solicitudes prototipo 1

Fuente: Autores

A continuación se mostrará las medidas estadísticas acerca del tiempo de respuesta por página en el caso de la medición del prototipo desarrollado sin ningún tipo de método de reutilización de código.

Tabla III. XIII: Valores de respuesta por solicitud prototipo 1

Valor Mínimo	Media	Valor Máximo	Mediana	Desviación Standard
0,01	4,387	111,866	0,032	21,599

Fuente: Autores

5.8. Prototipo 2: con composición

5.8.1. Determinación del tiempo de desarrollo

Para realizar el conteo de líneas de código se recurrió al uso del programa Notepad ++ en su versión 6 que nos permite realizar un conteo manual de las líneas de código de cada página y clase creada en los prototipos, para el total del conteo de Líneas de Código no se han tomado en cuenta líneas en blanco y comentarios.

Gestión de proyectos

En la siguiente tabla (Tabla XIV) se mostrará el número de líneas de código que se empleó en la implementación de las páginas web del prototipo con el modelo de composición.

Tabla III. XIV: Número líneas de código de páginas web prototipo 2

Gestión de Proyectos					
Páginas Web					
Proyecto	**Ingresar Proyectos**	**Procesa Ingresar Proyectos**	**Modificar Proyecto**	**Procesa Modificar Proyectos**	**Eliminar Proyectos**
117	306	84	381	78	31
Total Líneas de Código					997

Fuente: Autores

En la tabla XV se mostrará el número de líneas de código de la implementación del prototipo con el modelo de composición.

Tabla III. XV: Número de líneas de código de clases java prototipo 2

Clases Java												
Acceso Datos	**Datos Generales**	**Datos Persona**	**Proyecto Formulación**	**Cobertura**	**Estado Proyecto**	**Funcionario**	**Programa**	**Subsector**	**Tipo Inversión**	**Tipo proyecto**	**Ubicación Comunidad**	**Ubicación Geográfica**
244	58	58	486	123	118	190	122	140	122	121	140	262
Total Líneas de Código												2184
Total Líneas de Código de Gestión de Proyectos												3181

Fuente: Autores

Ahora se identifica el modo de utilización de los valores de COCOMO de la tabla de coeficientes, utilizando el modo semi-acoplado ya que este proyecto es de complejidad media.

Aplicando las ecuaciones del esfuerzo de COCOMO se obtendrá los siguientes valores:

Previamente se transformará el total de líneas de código a tamaño en kilobytes:

$$\frac{3181}{1000} = 3{,}181\ kbytes$$

Esfuerzo realizado:

$$E = aLDC^b \qquad E = 3{,}0 * 3{,}181^{1{,}12} \qquad E = 10{,}96459\ persona * mes$$

Tiempo realizado

$$T = cE^d \qquad T = 2{,}5 * 10{,}964594^{0{,}35} \qquad T = \ 5{,}78012\ meses$$

Número de personas

Esta ecuación se utilizará para determinar el tiempo de desarrollo de la solución, utilizando para ello el valor conocido de N = 2.

$$N = \frac{E}{T} \qquad T = \frac{E}{N} \qquad T = \frac{10{,}964594}{2} \qquad T = \ 5{,}48\ meses$$

Dificultad

La dificultad durante el desarrollo se determina manualmente utilizando la tabla de valores ya establecida:

Tabla III. XVI: Tabla de valores de dificultad prototipo 2

Categoría	Programadores	Duración	Líneas de Código	Ejemplo
Trivial	1	0 – 4 semanas	< 1k	Utilidad de Ordenación
Pequeño	1	1 – 6 meses	1k – 3k	Biblioteca de Funciones
Media	2 – 5	0,5 – 2 años	3k – 50k	Compilador de C
Grande	5 – 20	2 – 3 años	50k – 100k	SO pequeño
Muy Grande	100 – 1000	4 – 5 años	100k – 1M	Grande SO

Fuente: Autores

El tiempo de desarrollo de este sistema tomará un tiempo aproximado de 5,48 meses y un valor establecido de 3,181 Kb aproximadamente por lo que se puede determinar que la dificultad es **MEDIA**.

5.8.2. Pruebas de Desempeño

La herramienta nos permite la generación de un reporte detallado acerca de los resultados obtenidos, estos resultados se generan en inglés por lo que para la realización de este reporte se traducirá textualmente todo el contenido del informe.

Tabla III. XVII: Pruebas desempeño prototipo 2

Proyecto	Pruebas_Composición	Políticas de Carga	Una cantidad de 10 usuario virtuales constantes.
Escenario	scenario1		
Fecha de Inicio	22-nov-2012 10:26:16	Descripción	Para medir el desempeño de una aplicación con Composición
Fecha de Fin	22-nov-2012 10:28:17		
Duración	00:02:15	Filtros	Ninguno
Host	localhost	Debug	Deshabilitado

Fuente: Autores

Resumen de Estadísticas:

Tabla III. XVIII: Resumen de estadísticas prototipo 2

Total Páginas	**40**	**Media de Páginas**	**0,3**
Total Hits	860	Media de Hits	2,8
Total Usuarios en Uso	10	Media del Tiempo de Request	3,34 segundos
Total Iteraciones Completas	10	Media del Tiempo de Respuesta	28,1 segundos
Total de Paso de MB	78.52	Media de Paso de MB	1,38 Mb/s
Total Hits de Errores	60	Porcentaje de Error	17,6%
Total de Errores en la Acción	0	Total duración de Alertas	95,6%

Fuente: Autores

En la Tabla XVIII se detallan campos que deben ser explicados para una total comprensión del trabajo realizado:

Total páginas: Es el número total de veces que las páginas fueron visitadas por los usuarios del escenario.

Total usuarios en uso: Se refiere al número de usuarios virtuales usados durante las pruebas de desempeño de los prototipos en este caso se mantiene constante en 10 usuarios.

Total de paso de MB: indica el número total de MB retornados por el servidor, en este caso es para los 10 usuarios que intervinieron en la simulación.

Total de errores: Este punto indica la cantidad de errores que se registraron durante la simulación de uso de los prototipos, este punto muestra la cantidad total errores mostrados por los 10 usuarios, estos errores corresponden a imágenes y animaciones swf.

Total de errores en la acción: Muestra la cantidad de errores ocurridos durante el proceso de gestión (ingreso, modificación y eliminación) de los proyectos y del usuario.

Media de tiempo de solicitud (Request): Promedio del tiempo de Solicitud al servidor.

5.8.3. Porcentaje de uso de CPU

Esta medida se hace con el propósito de indicar el porcentaje de uso en lo referente a la carga del CPU.

Tabla III. XIX: Porcentaje de Uso de CPU prototipo 2

Tiempo (s)	% Uso de CPU	Tiempo (s)	% Uso de CPU
1	50.0	**70**	11.0
5	78.0	**75**	29.0
10	50.0	**80**	18.0
15	22.0	**85**	7.0
20	10.0	**90**	25.0
25	19.0	**95**	10.0
Tiempo (s)	**% Uso de CPU**	**Tiempo (s)**	**% Uso de CPU**
30	16.0	**100**	26.0
35	11.0	**105**	7.0
40	11.0	**110**	25.0
45	8.0	**115**	13.0
50	18.0	**120**	36.0
55	17.0	**125**	9.0
60	9.0	**130**	14.0
65	22.0	**135**	8.0

Fuente: Autores

Como se puede apreciar en la tabla las medidas tomadas corresponden a un tiempo total de 2 minutos y 15 segundos, que es el tiempo total por usuario que la herramienta utiliza para las pruebas, se puede apreciar que al inicio de la aplicación hay un aumento en el uso del CPU que se sitúa en un 50% y se

mantiene en niveles altos por lo menos 10 seg. , para luego mantenerse a un nivel bajo constante de 18,6 %.

Tabla III. XX: Resumen estadístico uso de CPU prototipo 2

Mínimo	Media	Máximo	Mediana	Desviación Standard
4	18,6	78	15	13

Fuente: Autores

En seguida se mostrará un gráfico estadístico del porcentaje de uso del CPU en la medición del prototipo de composición.

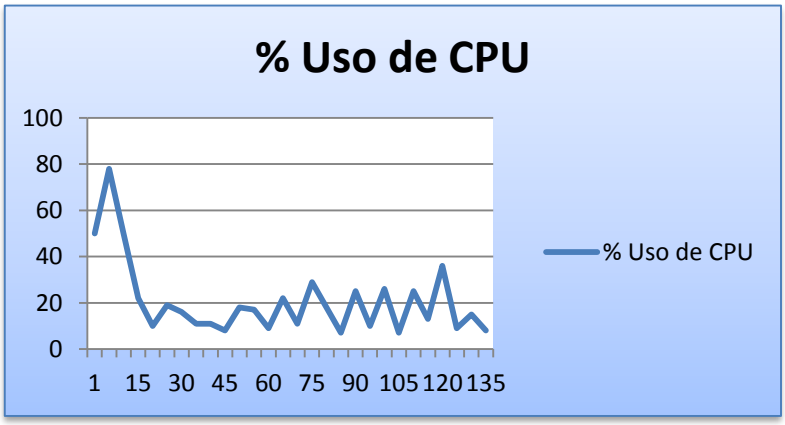


Figura III. 40. Porcentaje de uso de CPU en el prototipo 2

Fuente: Autores

5.8.4. Porcentaje de uso de memoria RAM

Medidas realizadas para indicar el porcentaje de uso de la memoria durante las pruebas del prototipo con herencia.

Tabla III. XXI: Porcentaje de Uso de Memoria RAM prototipo 2

Tiempo (s)	% Uso de Memoria	Tiempo (s)	% Uso de Memoria
1	11.0	**70**	9.0
5	16.0	**75**	9.0
10	7.0	**80**	9.0
15	7.0	**85**	9.0
20	7.0	**90**	9.0
25	7.0	**95**	9.0
30	8.0	**100**	9.0
35	8.0	**105**	9.0
40	8.0	**110**	9.0
45	8.0	**115**	10.0
50	8.0	**120**	10.0
55	8.0	**125**	11.0
60	8.0	**130**	11.0
65	8.0	**135**	13.0

Fuente: Autores

Como se puede apreciar en la tabla las medidas tomadas corresponden a un tiempo total de 2 minutos y 15 segundos, que es el tiempo total por usuario que la herramienta utiliza para las pruebas, se puede apreciar que el uso de la memoria se mantiene en un porcentaje promedio del 8,83 % y que se mantiene por debajo del 16% del porcentaje total de memoria RAM del computador.

Tabla III. XXII: Resumen estadístico uso de CPU prototipo 2

Mínimo	Media	Máximo	Mediana	Desviación Standard
7	8,83	16	9	1,54

Fuente: Autores

De la misma manera se presentará un gráfico estadístico indicando el porcentaje de uso de memoria con el prototipo de composición.

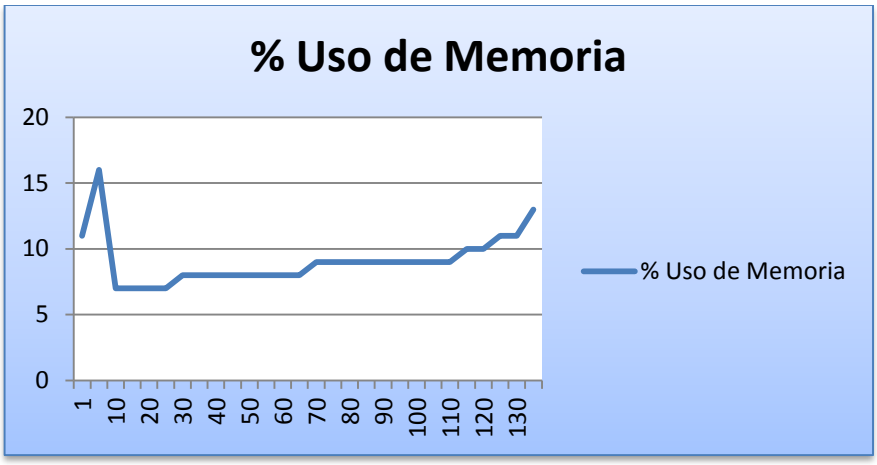


Figura III. 41. Porcentaje de uso de memoria RAM en el prototipo 2

Fuente: Autores

Tiempo medio de respuesta por página

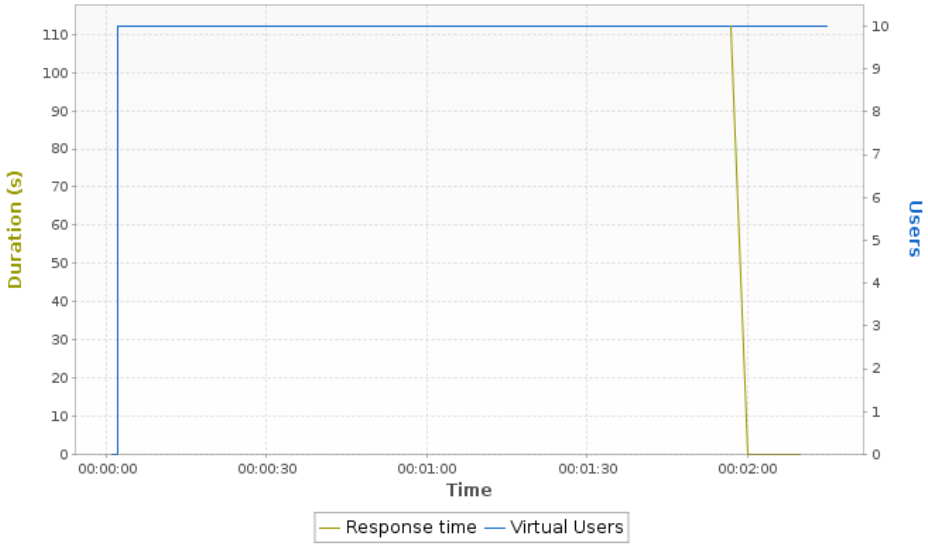


Figura III. 42. Tiempo de respuesta por página prototipo 2

Fuente: Autores

A continuación se mostrará las medidas estadísticas acerca del tiempo de respuesta en el caso de la medición del prototipo realizado utilizando la composición.

Tabla III. XXIII: Valores de respuesta por página prototipo 2

Valor Mínimo	Media	Valor Máximo	Mediana	Desviación Standard
0,01	37,397	112,162	0,019	53,77

Fuente: Autores

Tiempo de respuesta media de solicitudes (Request)

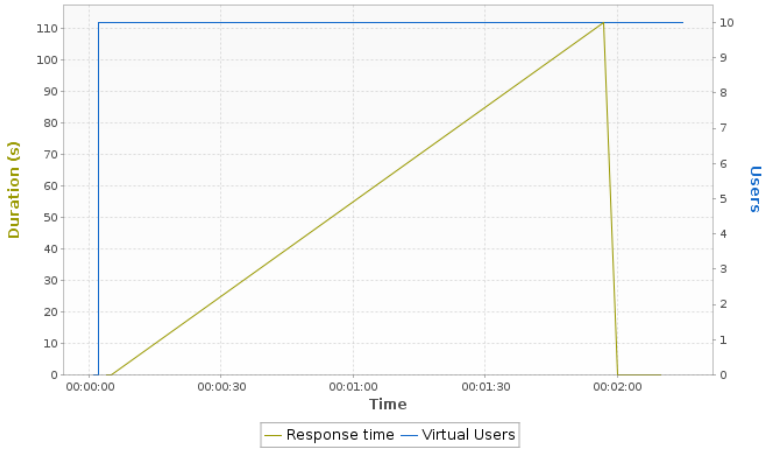


Figura III. 43. Tiempo de respuesta por solicitudes prototipo 2

Fuente: Autores

A continuación se mostrará las medidas estadísticas acerca del tiempo de respuesta por página en el caso de la medición del prototipo realizado utilizando la composición.

Tabla III. XXIV: Valores de respuesta por solicitud prototipo 2

Valor Mínimo	Media	Valor Máximo	Mediana	Desviación Standard
0,01	4,336	111,866	0,032	21,547

Fuente: Autores

5.9. Prototipo 3: con herencia

5.9.1. Determinación del tiempo de desarrollo

Para realizar el conteo de líneas de código se recurrió al uso del programa Notepad ++ en su versión 6 que nos permite realizar un conteo manual de las líneas de código de cada página y clase creada en los prototipos, para el total del conteo de líneas de código no se han tomado en cuenta líneas en blanco y comentarios.

Gestión de proyectos

En la Tabla III se mostrará las páginas web junto con el número de líneas reales de código en el módulo de “Gestión de Proyectos” que se ha desarrollado utilizando el modelo de herencia.

Tabla III. XXV: Número líneas de código de páginas web prototipo 3

Gestión de Proyectos					
Páginas Web					
Proyecto	**Ingresar Proyectos**	**Procesa Ingresar Proyectos**	**Modificar Proyecto**	**Procesa Modificar Proyectos**	**Eliminar Proyectos**
117	306	84	381	78	31
Total Líneas de Código					997

Fuente: Autores

En la Tabla IV se mostrará el número de líneas de código que se implementó en la creación de las clases Java para nuestra aplicación web utilizando el modelo de herencia.

Tabla III. XXVI: Número de líneas de código de clases java prototipo3

Clases Java												
Acceso Datos	Datos Generales	Datos Persona	Proyecto Formulación	Cobertura	Estado Proyecto	Funcionario	Programa	Subsector	Tipo Inversión	Tipo proyecto	Ubicación Comunidad	Ubicación Geográfica
244	58	58	438	95	88	142	88	106	88	87	107	244
Total Líneas de Código												1843
Total Líneas de Código de Gestión de Proyectos												2840

Fuente: Autores

Ahora se identifica el modo de utilización de los valores de COCOMO de la tabla de coeficientes, utilizando el modo semi-acoplado ya que este proyecto es de complejidad media.

Aplicando las ecuaciones del esfuerzo de COCOMO se obtendrá los siguientes valores:

Previamente se transformará el total de líneas de código a tamaño en kilobytes:

$$\frac{2840}{1000} = 2{,}84\ kbytes$$

Esfuerzo realizado:

$$E = aLDC^{b} \qquad E = 3{,}0 * 2{,}84^{1{,}12} \qquad E = 9{,}65690\ persona * mes$$

Tiempo realizado

$$T = cE^d \qquad T = 2{,}5 * 9{,}65690^{0{,}35} \qquad T = \ 5{,}5288\ meses$$

Número de personas

Esta ecuación se utilizará para determinar el tiempo de desarrollo de la solución, utilizando para ello el valor conocido de N = 2.

$$N = \frac{E}{T} \qquad T = \frac{E}{N} \qquad T = \frac{9{,}65690}{2} \qquad T = \ 4{,}82\ meses$$

Dificultad

La dificultad durante el desarrollo de un sistema informático se determina manualmente utilizando la tabla de valores ya establecida:

Tabla III. XXVII: Tabla de valores de dificultad prototipo 3

Categoría	Programadores	Duración	Líneas de Código	Ejemplo
Trivial	1	0 – 4 semanas	< 1k	Utilidad de Ordenación
Pequeño	1	1 – 6 meses	1k – 3k	Biblioteca de Funciones
Media	2 – 5	0,5 – 2 años	3k – 50k	Compilador de C
Grande	5 – 20	2 – 3 años	50k – 100k	SO pequeño
Muy Grande	100 – 1000	4 – 5 años	100k – 1M	Grande SO

Fuente: Autores

El tiempo de desarrollo de este sistema tomará un tiempo aproximado de 4,82 meses y un valor establecido de 2,8 Kb aproximadamente por lo que se puede determinar que la dificultad es **PEQUEÑA**.

5.9.2. Prueba de desempeño

La herramienta nos permite la generación de un reporte detallado acerca de los resultados obtenidos, estos resultados se generan en inglés por lo que para la

realización de este reporte se traducirá textualmente todo el contenido del informe.

Tabla III. XXVIII: Pruebas de desempeño prototipo 3

Proyecto	Pruebas_Herencia	Políticas de Carga	Una cantidad de 10 usuario virtuales constantes.
Escenario	Scenario1		
Fecha de Inicio	21-nov-2012 15:14:00	Descripción	Pruebas de Rendimiento en una aplicación con herencia
Fecha de Fin	21-nov-2012 15:16:01		
Duración	00:02:24	Filtros	Ninguno
Host	localhost	Debug	Deshabilitado

Fuente: Autores

Resumen de estadísticas:

Tabla III. XXIX: Resumen de estadísticas prototipo 3

Total Páginas	210	**Media de Páginas**	1,7
Total Hits	860	Media de Hits	7,1
Total Usuarios en Uso	10	Media del Tiempo de Request	0,021 segundos
Total Iteraciones Completas	10	Media del Tiempo de Respuesta	0,046 segundos
Total de Paso de MB	78.52	Media de Paso de MB	5,19
Total Hits de Errores	140	Porcentaje de Error	16,3%
Total de Errores en la Acción	0	Total duración de Alertas	95,7%

Fuente: Autores

En la Tabla XXIX se detallan campos que deben ser explicados para una total comprensión del trabajo realizado:

Total páginas: Es el número total de veces que las páginas fueron visitadas por los usuarios del escenario.

Total usuarios en uso: Se refiere al número de usuarios virtuales usados durante las pruebas de desempeño de los prototipos en este caso se mantiene constante en 10 usuarios.

Total de paso de MB: indica el número total de MB retornados por el servidor, en este caso es para los 10 usuarios que intervinieron en la simulación.

Total de errores: Este punto indica la cantidad de errores que se registraron durante la simulación de uso de los prototipos, este punto muestra la cantidad total errores mostrados por los 10 usuarios, estos errores corresponden a imágenes y animaciones swf.

Total de errores en la acción: Muestra la cantidad de errores ocurridos durante el proceso de gestión (ingreso, modificación y eliminación) de los proyectos y del usuario.

Media de tiempo de solicitud (Request): Promedio del tiempo de solicitud al servidor.

5.9.3. Porcentaje de uso de CPU

Esta medida se hace con el propósito de indicar el porcentaje de uso en lo referente a la carga del CPU.

Tabla III. XXX: Porcentaje de uso del CPU prototipo 3

Tiempo (s)	CPU (%)	Tiempo (s)	CPU (%)
1	62.0	**75**	7.0
5	49.0	**80**	17.0
10	57.0	**85**	8.0
15	6.0	**90**	4.0
20	11.0	**95**	12.0
25	9.0	**100**	8.0

Tiempo (s)	CPU (%)	Tiempo (s)	CPU (%)
30	11.0	**105**	8.0
35	5.0	**110**	2.0
40	8.0	**115**	40.0
45	5.0	**120**	5.0
50	11.0	**125**	17.0
55	6.0	**130**	4.0
60	4.0	**135**	8.0
65	12.0	**140**	6.0
70	11.0	**144**	18.0

Fuente: Autores

Como se puede apreciar en la tabla las medidas tomadas corresponden a un tiempo total de 2 minutos y 22 segundos, que es el tiempo total por usuario que la herramienta utiliza para las pruebas, se puede apreciar que al inicio de la aplicación hay un aumento en el uso del CPU que se sitúa en un 62% y se mantiene en niveles altos por lo menos 10 seg. , para luego mantenerse a un nivel bajo constante de 13,9 %.

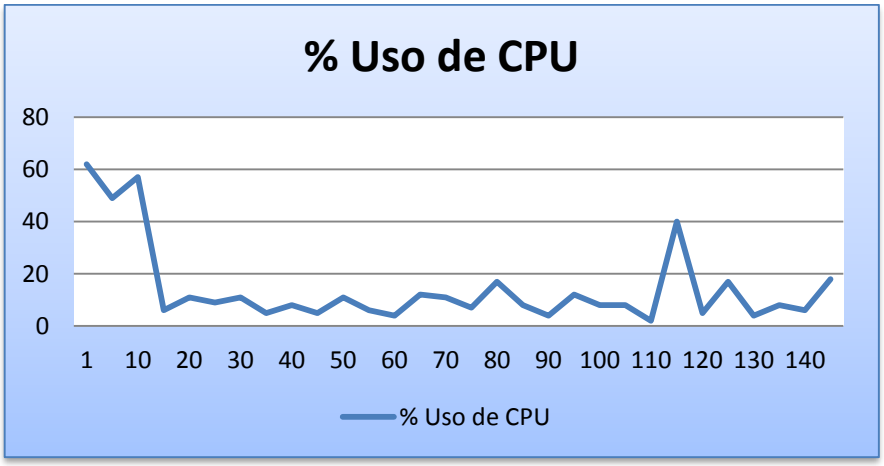


Figura III. 44. Porcentaje de uso de CPU en el prototipo 3

Fuente: Autores

5.9.4. Porcentaje de uso de memoria RAM

Medidas realizadas para indicar el porcentaje de uso de la memoria durante las pruebas del prototipo con herencia.

Tabla III. XXXI: Resumen estadístico uso de Memoria prototipo 3

Tiempo (s)	Memoria (%)	Tiempo (s)	Memoria (%)
1	11.0	75	17.0
5	15.0	80	18.0
10	9.0	85	18.0
15	9.0	90	18.0
20	10.0	95	19.0
25	10.0	100	19.0
30	10.0	105	19.0
35	10.0	110	20.0
40	13.0	115	20.0
45	14.0	120	5.0
50	14.0	125	6.0
55	15.0	130	6.0
60	15.0	135	7.0
65	17.0	140	10.0
70	17.0	144	11.0

Fuente: Autores

Como se puede apreciar en la tabla las medidas tomadas corresponden a un tiempo total de 2 minutos y 22 segundos, que es el tiempo total por usuario que la herramienta utiliza para las pruebas, se puede apreciar que el uso de la memoria se mantiene en un porcentaje promedio del 13,6 % y que se mantiene por debajo del 20% del porcentaje total de memoria RAM del computador.

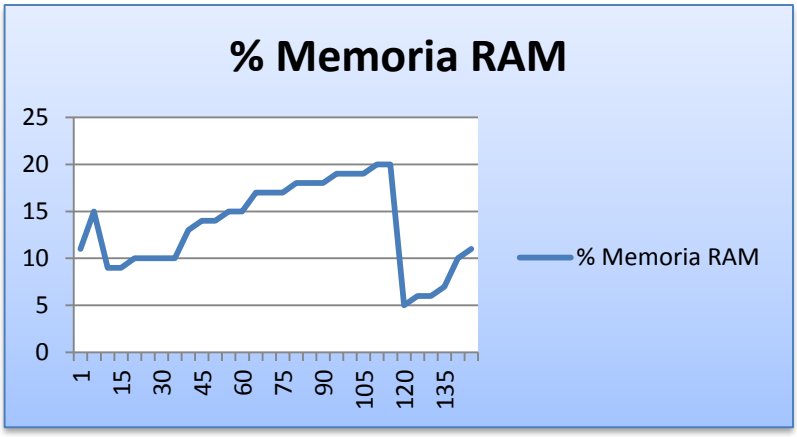


Figura III. 45. Porcentaje de uso de la memoria RAM en el prototipo 3

Fuente: Autores

Tiempo medio de respuesta por página

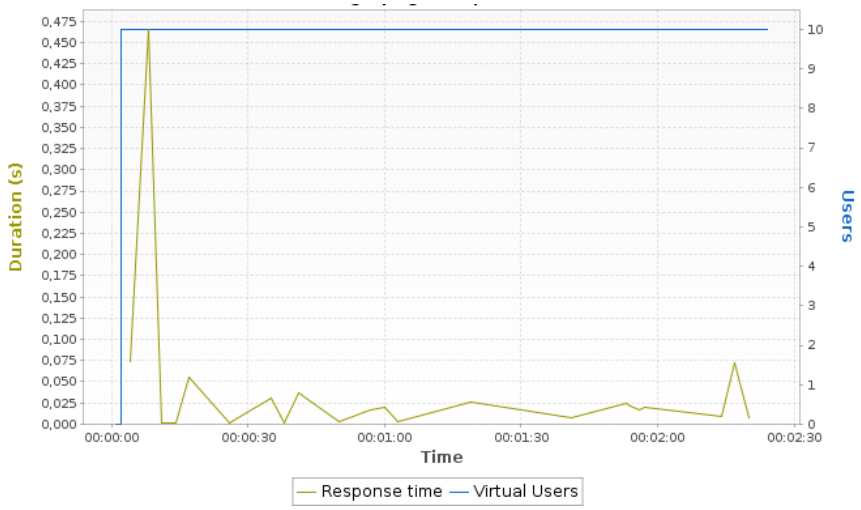


Figura III. 46. Tiempo de respuesta por página prototipo 3

Fuente: Autores

Tabla III. XXXII: Valores de respuesta por página prototipo 3

Valor Mínimo	Media	Valor Máximo	Mediana	Desviación Standard
0,001	0,044	0,466	0,017	0,099

Fuente: Autores

Tiempo de respuesta en segundos de todas las páginas.

Tiempo de respuesta media de solicitudes (Request)

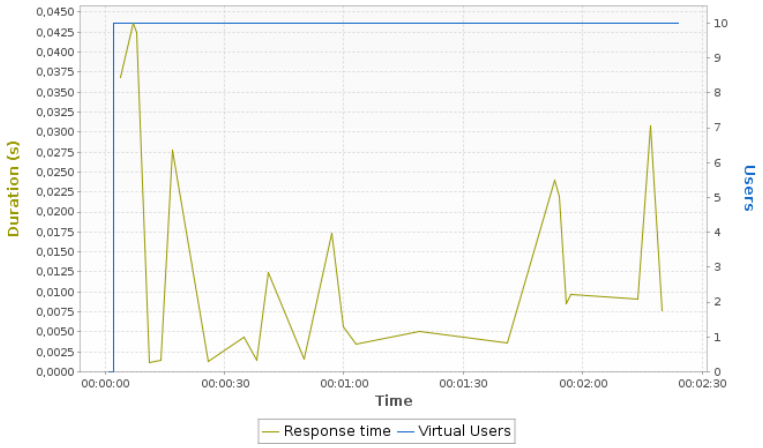


Figura III. 47. Tiempo de respuesta por solicitud prototipo 3

Fuente: Autores

A continuación se presentará una tabla indicando los parámetros estadísticos del tiempo de respuesta por página generado con la herramienta NEOLOAD.

Tabla III. XXXIII: Valores de respuesta por solicitud prototipo 3

Valor Mínimo	Media	Valor Máximo	Mediana	Desviación Standard
0,001	0,02	0,044	0,011	0,017

Fuente: Autores

5.10. Análisis de resultados

5.10.1. Indicador 1: Tiempo

En la Tabla XXXIV se indica una comparación del tiempo de desarrollo de los prototipos de herencia, composición y del prototipo que no utiliza ningún tipo de método de reutilización de código.

Tabla III. XXXIV: Comparativa de tiempos de desarrollo

Prototipo	Tiempo (meses)
Herencia	4,82
Composición	5,48
Sin Utilizar Métodos de Reutilización	5,31

Fuente: Autores

A continuación se puede establecer de manera aproximada el tiempo de desarrollo que conlleva cada prototipo, determinando que con un margen de 0,66; lo que significa una reducción de 19,8 días aproximadamente. Cabe recalcar que estos tiempos son aproximaciones que se realizan mediante la aplicación de las fórmulas del COCOMO básico.

Sin métodos de reutilización de código

$$5{,}31 meses \left[\frac{30\ dias\ aprox.}{1\ mes}\right] = 159{,}30\ días$$

Herencia

$$4{,}82 meses \left[\frac{30\ dias\ aprox.}{1\ mes}\right] = 144{,}6\ días$$

Composición

$$5{,}48 meses \left[\frac{30\ dias\ aprox.}{1\ mes}\right] = 164{,}4\ días$$

Las conversiones anteriores nos muestran el total de número de días que toma el desarrollo del prototipo, pudiendo notarse que el tiempo de desarrollo del sistema sin utilizar ningún tipo de método de reutilización de código toma un aproximado

de 159,30 días, este valor será considerado la base para realizar la comparación con los prototipos que usan herencia y composición.

El siguiente gráfico nos muestra la diferencia entre los tiempos de desarrollo de los tres prototipos.

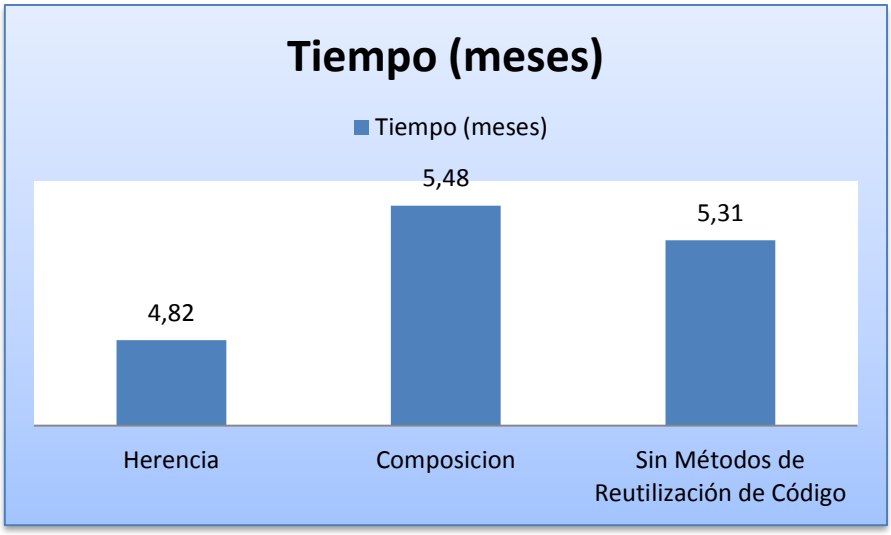


Figura III. 48. Comparativa de tiempos de desarrollo

Fuente: Autores

Interpretación de resultados

Se puede apreciar que al realizar una comparación de los tiempos de desarrollo que el tiempo de desarrollo sin utilizar ningún método de reutilización de código toma un aproximado de 5 meses, el prototipo con composición toma alrededor de 5,48 meses y el prototipo con herencia toma alrededor de aproximadamente4 meses, por lo que se puede concluir que al desarrollar la aplicación con herencia nos tomará menos tiempo de desarrollo, aproximadamente una diferencia de 0,66 meses frente al prototipo con composición y 0,49 meses menos al desarrollar el prototipo sin ningún método de reutilización de código.

Comparación entre Prototipo 1 y Prototipo 2

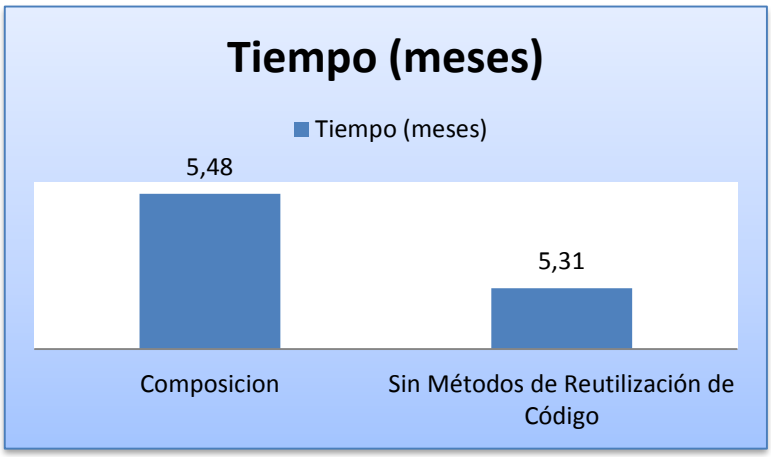


Figura III. 49. Comparativa de tiempos de desarrollo prototipo 1 y 2

Fuente: Autores

Interpretación de resultados

Los resultados obtenidos al realizar los cálculos nos indican que el tiempo de desarrollo del prototipo con composición es mayor al prototipo en el que no se utilizó algún método de reutilización de código, siendo una diferencia de 0,17 meses que equivale a un aproximado de 5 días, se deben consideran que estos resultados son producto de que al momento de aplicar la composición se deben crear además de las variables de cada clase un tipo anexo de las clases que compones la clase principal.

Comparación entre Prototipo 1 y Prototipo 3

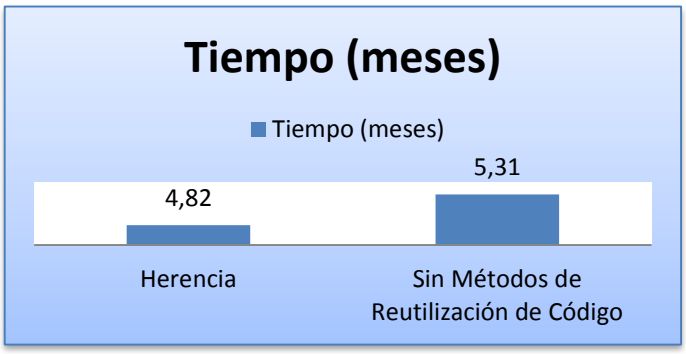


Figura III. 50. Comparativa de tiempos de desarrollo prototipo 1 y 3

Fuente: Autores

Interpretación de resultados

Los resultados obtenidos al realizar los cálculos nos indican que el tiempo de desarrollo del prototipo con herencia es menor al prototipo en el que no se utilizó algún método de reutilización de código, siendo una diferencia de 0,45 meses que equivale a un aproximado de 13 días.

Comparación entre Prototipo 2 y Prototipo 3

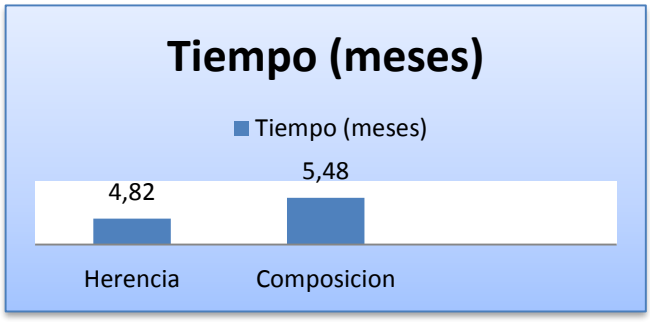


Figura III. 51. Comparativa de tiempos de desarrollo prototipo 2 y 3

Fuente: Autores

Interpretación de resultados

Al momento de realizar la comparación de los prototipos que utilizan la herencia y la composición se nota que al desarrollar el prototipo con herencia nos lleva menor tiempo de trabajo siendo la diferencia de tiempo de aproximadamente de 19 días.

5.10.2. Indicador 2: Esfuerzo

Para la Tabla VII se ha calculado el esfuerzo requerido por cada uno de los prototipos desarrollados.

Tabla III. XXXV: Comparativa esfuerzo requerido por prototipo

Prototipo	Esfuerzo (persona * mes)
Herencia	9,65690
Composición	10,964594
Sin Reutilización de Código	10,61389414

Fuente: Autores

Tabla III. XXXVI: Porcentajes de esfuerzo requerido por prototipo

Prototipo	Esfuerzo (%)
Herencia	90.99 %
Composición	103 %
Sin Reutilización de Código	100 %

Fuente: Autores

De igual manera se puede determinar que el esfuerzo requerido por persona para el desarrollo de la aplicación utilizando la herencia es considerablemente más bajo que el utilizado por la composición tomando en cuenta el esfuerzo obtenido al calcular el esfuerzo del prototipo en el que no se usó ningún tipo de métodos de reutilización de código.

El siguiente gráfico da prueba de la reducción del esfuerzo.

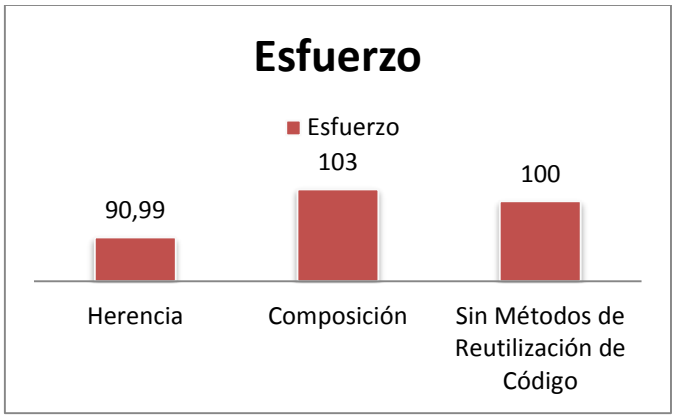


Figura III. 52. Comparativa esfuerzo requerido en porcentajes

Fuente: Autores

Interpretación de resultados

Se puede observar que al tomar como base el esfuerzo obtenido al desarrollar el prototipo sin métodos de reutilización de código, al cual se le ha asignado el valor de 100%, nos muestra que al desarrollar el prototipo con herencia disminuye un 10% aproximado con respecto a la base y que al momento de compararlo con el prototipo con composición este aumenta en un 3% con respecto a la base.

Comparación de Esfuerzo con el Prototipo 1 y Prototipo 2

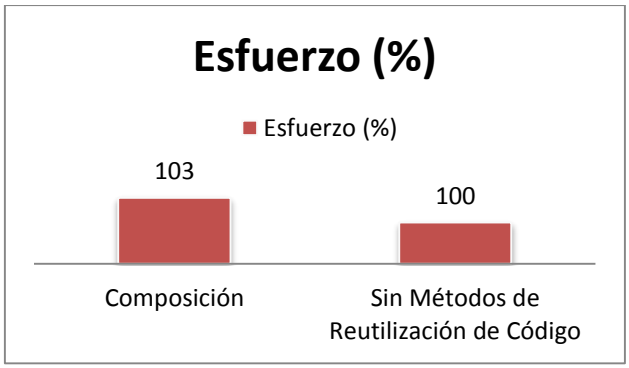


Figura III. 53. Comparativa esfuerzo requerido prototipo 1 y 2

Fuente: Autores

Interpretación de resultados

Al momento de realizar la comparación del prototipo realizado con composición y el prototipo realizado sin métodos de reutilización de código el esfuerzo aumenta un aproximado de 3 % del trabajo realizado.

Comparación de Esfuerzo con el Prototipo 1 y Prototipo 3

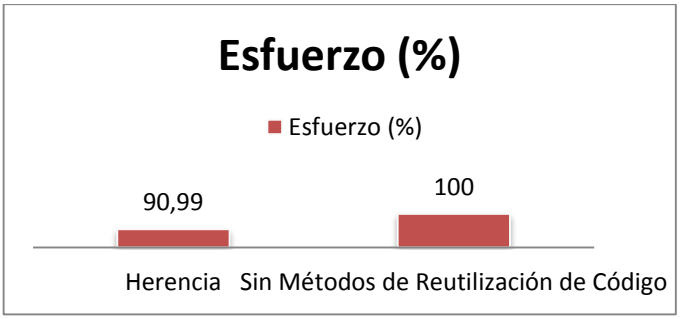


Figura III. 54. Comparativa esfuerzo requerido prototipo 1 y 3

Fuente: Autores

Interpretación de resultados

Al momento de realizar la comparación del prototipo realizado con herencia y el prototipo realizado sin métodos de reutilización de código el esfuerzo disminuye un aproximado de 10 % del trabajo realizado.

Comparación de Esfuerzo con el Prototipo 2 y Prototipo 3

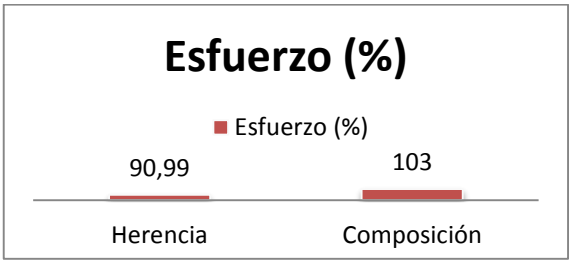


Figura III. 55. Comparativa esfuerzo requerido prototipo 2 y 3

Fuente: Autores

Interpretación de resultados

Al momento de realizar la comparación del prototipo realizado con composición y el prototipo realizado con herencia el esfuerzo aumenta un aproximado de 12.01 % del trabajo realizado.

5.10.3. Indicador 3: Líneas de Código

Para la Tabla XXXVII se realizará el conteo de las líneas de código utilizados en las clases java y en las páginas web desarrolladas para cada prototipo.

Tabla III. XXXVII: Comparativa líneas de código requerido por prototipo

Prototipo	Líneas de Código
Herencia	2840
Composición	3181
Sin Reutilización de Código	3088

Fuente: Autores

Se puede determinar mediante la tabla que el total de número de líneas de código usadas para el desarrollo de cada prototipo se reduce en el uso de la herencia y aumenta en la composición con respecto al total de líneas de código en el que no se utilizó ningún método de reutilización de código.

El siguiente gráfico da prueba de la reducción de las líneas de código.

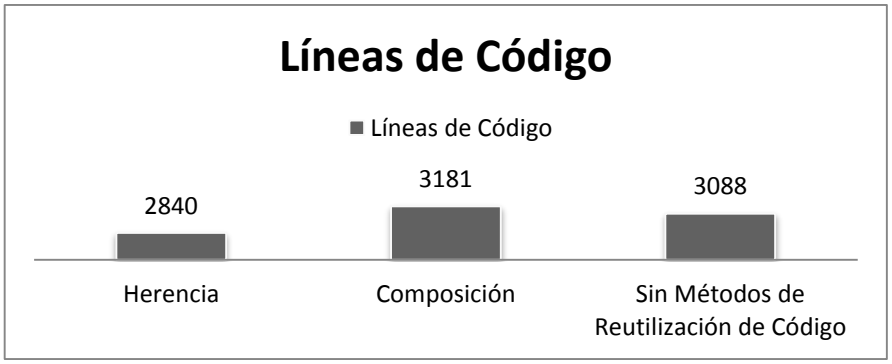


Figura III. 56. Comparativa líneas de código

Fuente: Autores

Interpretación de resultados

Se puede observar que la cantidad de líneas de código utilizadas en el prototipo sin métodos de reutilización de código es de un total de 3088 líneas de código, al compararlo con el prototipo de herencia disminuye un total de 248 líneas de código y que aumenta 93 líneas de código al comparar el prototipo con composición.

Comparación de Cantidad de Líneas de Código del Prototipo 1 y Prototipo 2

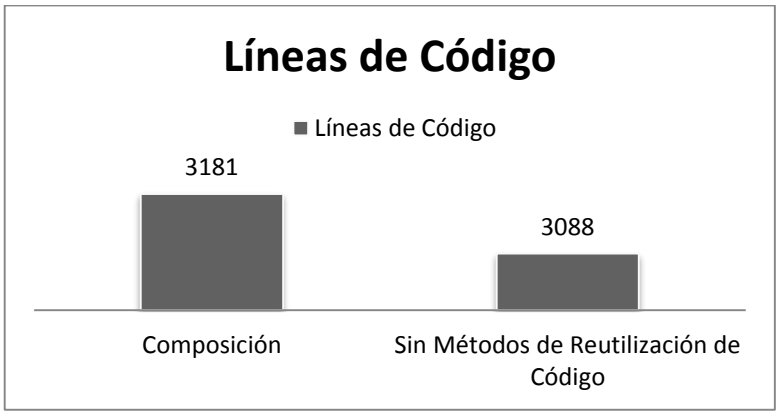


Figura III. 57. Comparativa líneas de código prototipo 1 y 2

Fuente: Autores

Interpretación de resultados

Se puede observar que la cantidad de líneas de código utilizadas en el prototipo sin métodos de reutilización de código es de un total de 3088, al compararlo con el prototipo de composición aumenta 93 líneas de código.

Comparación de cantidad de líneas de código del Prototipo 1 y Prototipo 3

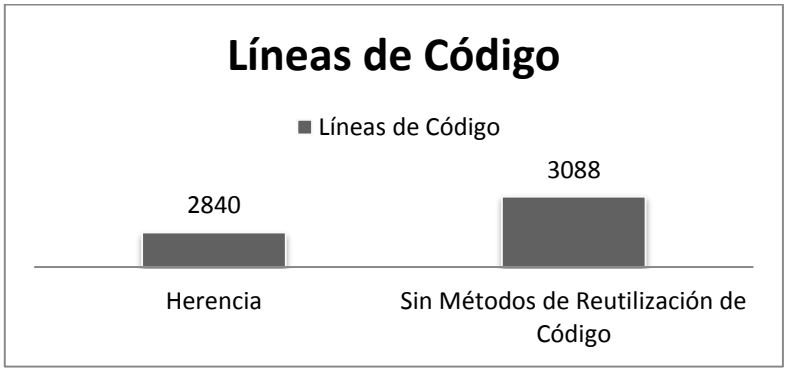


Figura III. 58. Comparativa líneas de código prototipo 1 y 3

Fuente: Autores

Interpretación de resultados

Se puede observar que la cantidad de líneas de código utilizadas en el prototipo sin métodos de reutilización de código es de un total de 3088, al compararlo con el prototipo de herencia disminuye un total de 248 líneas de código.

Comparación de cantidad de líneas de código del Prototipo 2 y Prototipo 3

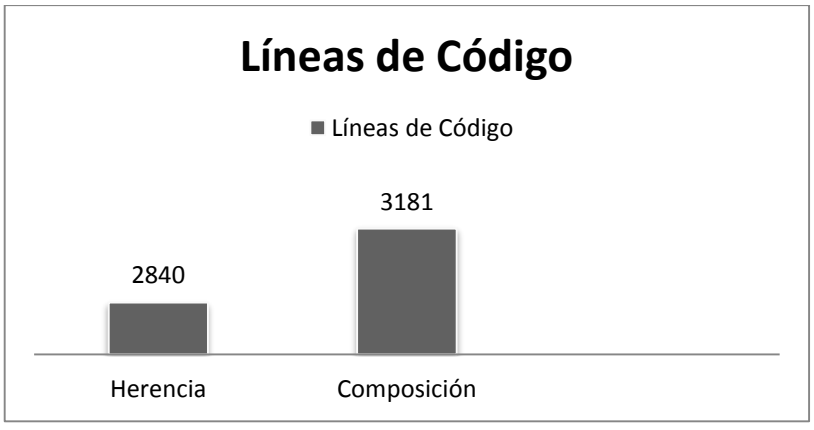


Figura III. 59. Comparativa líneas de código prototipo 2 y 3

Fuente: Autores

Interpretación de resultados

Se puede observar que la cantidad de líneas de código utilizadas en el prototipo con composición es de 3181 líneas de código y que al compararlo con el prototipo con herencia disminuye un total de 341 líneas de código.

5.10.4. Indicador 4: Dificultad del desarrollo

Dificultad

La dificultad durante el desarrollo de un sistema informático se muestra en la siguiente tabla donde se indica los resultados obtenidos por cada prototipo:

Tabla III. XXXVIII: Comparativa dificultad requerida por prototipo

Prototipo	Líneas de Código
Herencia	PEQUEÑA
Composición	MEDIA
Sin Reutilización de Código	MEDIA

Fuente: Autores

Análisis de Resultados:

Con la determinación de la Dificultad se determinará que al momento de usar el método de reutilización de código llamada herencia se observa que la dificultad disminuye: **PEQUEÑA**, frente a la dificultad obtenida con los prototipos que usan composición y otro que no usa ningún método de reutilización que son **MEDIA**.

5.10.5. Indicador 5: Porcentaje de uso del CPU

Comparativas entre los dos Prototipos.

La siguiente gráfica nos indica la diferencia entre el porcentaje de uso del CPU del servidor para cada prototipo.

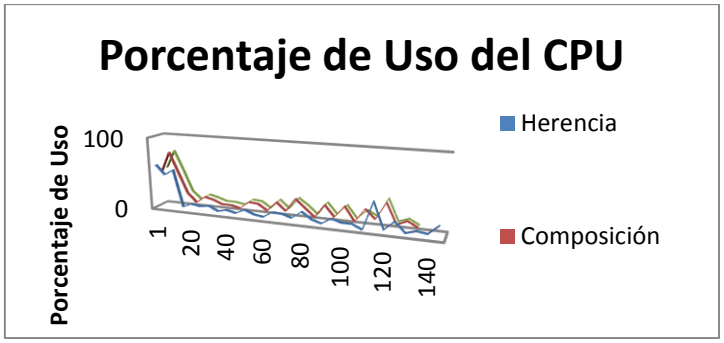


Figura III. 60. Porcentaje de uso del CPU

Fuente: Autores

Análisis de resultados:

Se puede notar que existe un margen de uso menor por parte de la solución empleada por la herencia, mientras que a su vez la solución que usa composición muestra índices de uso relativamente moderados, estos valores se determinan al obtener los datos del prototipo que no usan métodos de reutilización de códigos.

Comparación del porcentaje de uso del servidor entre el Prototipo 1 y 2

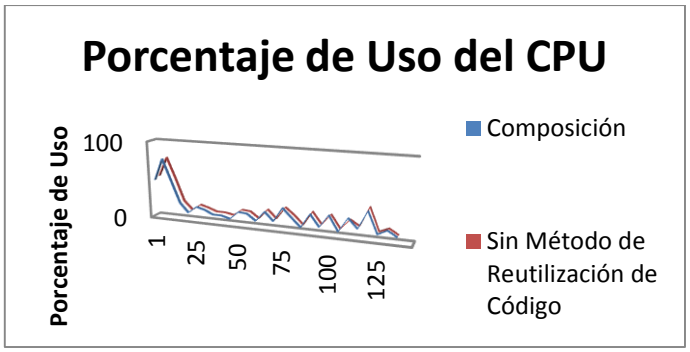


Figura III. 61. Porcentaje de uso del CPU prototipo 1 y 2

Fuente: Autores

Interpretación de resultados:

Se puede apreciar que con respectos a los datos obtenidos con el prototipo que no usa métodos de reutilización de código, el prototipo con composición mantiene un porcentaje de uso relativamente igual.

Comparación del porcentaje de uso del servidor entre el Prototipo 1 y 3

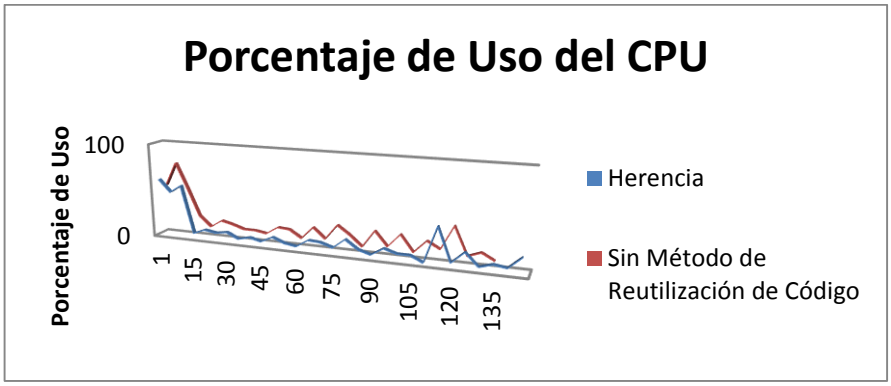


Figura III. 62. Porcentaje de uso del CPU prototipo 1 y 3

Fuente: Autores

Interpretación de resultados:

Se puede apreciar que con respectos a los datos obtenidos con el prototipo que no usa métodos de reutilización de código, el prototipo con herencia mantiene un porcentaje de uso relativamente más alto esto se debe a que la herencia usa el más el porcentaje de CPU.

Comparación del porcentaje de uso del servidor entre el Prototipo 2 y 3

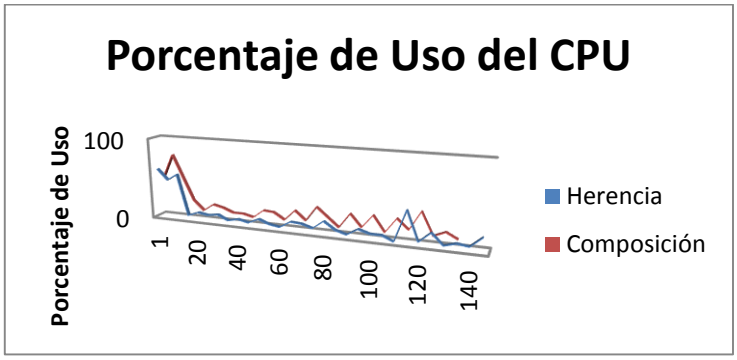


Figura III. 63. Porcentaje de uso del CPU prototipo 2 y 3

Fuente: Autores

Interpretación de resultados:

Cuando se realiza la comparación entre los prototipos que usan herencia y composición se determinará que el uso del CPU es más alto al momento de desplegar el prototipo con herencia.

5.10.6. Indicador 6: Porcentaje de uso de la memoria RAM

La siguiente grafica nos muestra el porcentaje de uso de la memoria RAM.

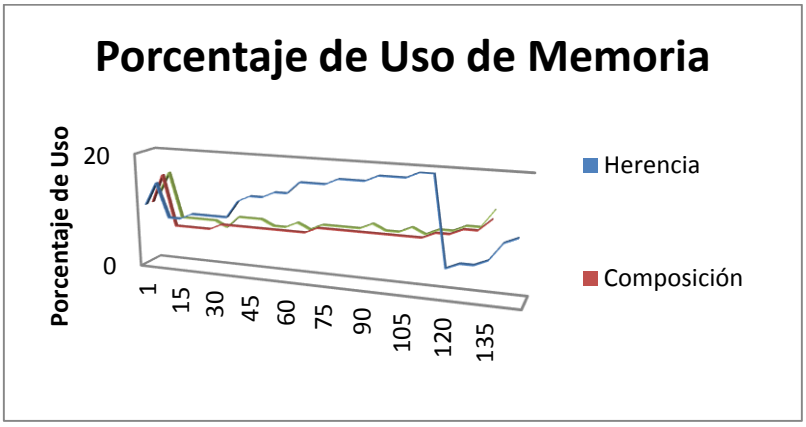


Figura III. 64. Porcentaje de uso de la memoria RAM

Fuente: Autores

Interpretación de resultados:

Sin embargo dados los resultados expuestos anteriormente, el porcentaje de uso de la memoria RAM del servidor se nota un porcentaje de uso relativamente alto por parte de la solución que utiliza la herencia frente a los niveles relativamente bajos que nos muestra la composición.

Comparación del porcentaje de uso de Memoria RAM del servidor entre el Prototipo 1 y Prototipo 2

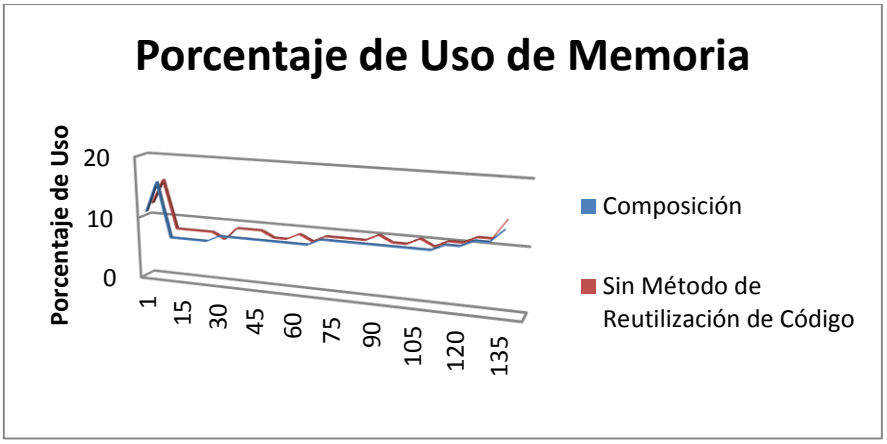


Figura III. 65. Porcentaje de uso de la memoria RAM prototipo 1 y 2

Fuente: Autores

Interpretación de resultados:

Cuando se realiza la comparación entre los prototipos que usan composición y sin ningún tipo de método de reutilización de código se observa ambos prototipos mantienen un nivel de uso de memoria RAM relativamente igual.

Comparación del porcentaje de uso de RAM del servidor entre el Prototipo 1 y Prototipo 3

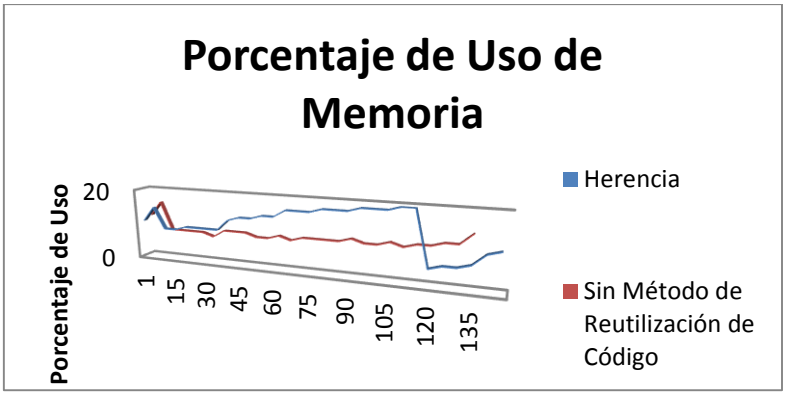


Figura III. 66. Porcentaje de uso de la memoria RAM prototipo 1 y 3

Fuente: Autores

Interpretación de resultados:

Cuando se realiza la comparación entre los prototipos que usan herencia y sin ningún tipo de método de reutilización de código se observa que el prototipo con herencia utiliza más la memoria RAM debido a que la memoria es usada constantemente por los objetos heredados.

Comparación del porcentaje de uso de RAM del servidor entre el Prototipo 2 y Prototipo 3

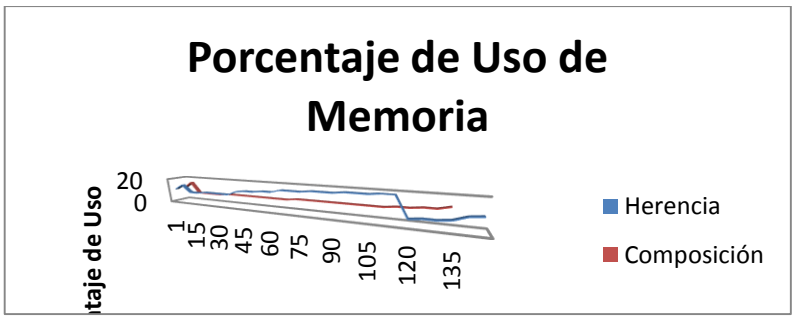


Figura III. 67. Porcentaje de uso de la memoria RAM prototipo 2 y 3

Fuente: Autores

Interpretación de resultados:

Cuando se realiza la comparación entre los prototipos que usan herencia y composición se determinará que el uso del CPU es más alto al momento de desplegar el prototipo con herencia.

5.11. Conclusiones Análisis de Resultados

Al momento de terminar con el análisis de cada indicador utilizado para el análisis de las ventajas del uso o no de metodologías de reutilización de código en java se ha podido determinar que al utilizar la Herencia como método de Reutilización de Código se obtiene una reducción considerable del tiempo de desarrollo de la aplicación, valor obtenido del indicador 1 con el que se aprecia una reducción de tiempo de 9% frente a la no utilización de ningún tipo de reutilización de código y una reducción de 12% frente a la utilización de la Composición.

Tabla III. XXXIX: Porcentaje de reducción de tiempos de desarrollo

Prototipo	Porcentaje de Reducción de tiempo
Herencia	91 %
Composición	103 %
Sin Utilizar Métodos de Reutilización	100 %

Fuente: Autores

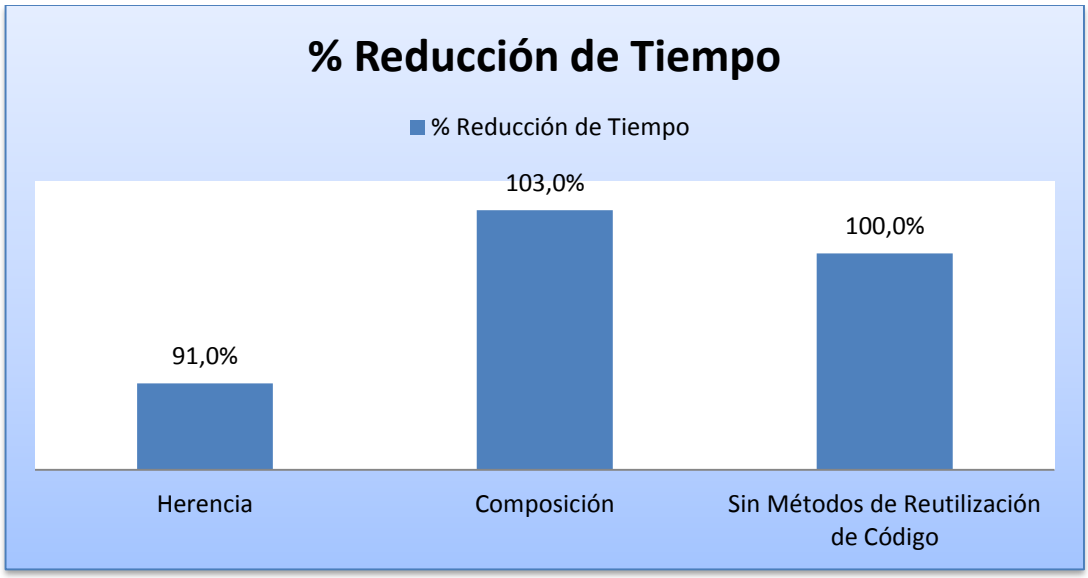


Figura III. 68: Reducción de tiempos de desarrollo

Fuente: Autores

El esfuerzo requerido para el desarrollo de la aplicación también se ve reducido de manera notable al utilizar la herencia como método de reutilización de código, ya que se puede apreciar que al utilizar la herencia el esfuerzo disminuye un 9,01% frente a la no utilización de ningún método de reutilización de código y una disminución de 12,01% frente a la utilización de la composición como método de reutilización de código.

Tabla III. XL: Porcentajes de esfuerzo requerido por prototipo

Prototipo	Porcentaje de Reducción del Esfuerzo
Herencia	90.99 %
Composición	103 %
Sin Reutilización de Código	100 %

Fuente: Autores

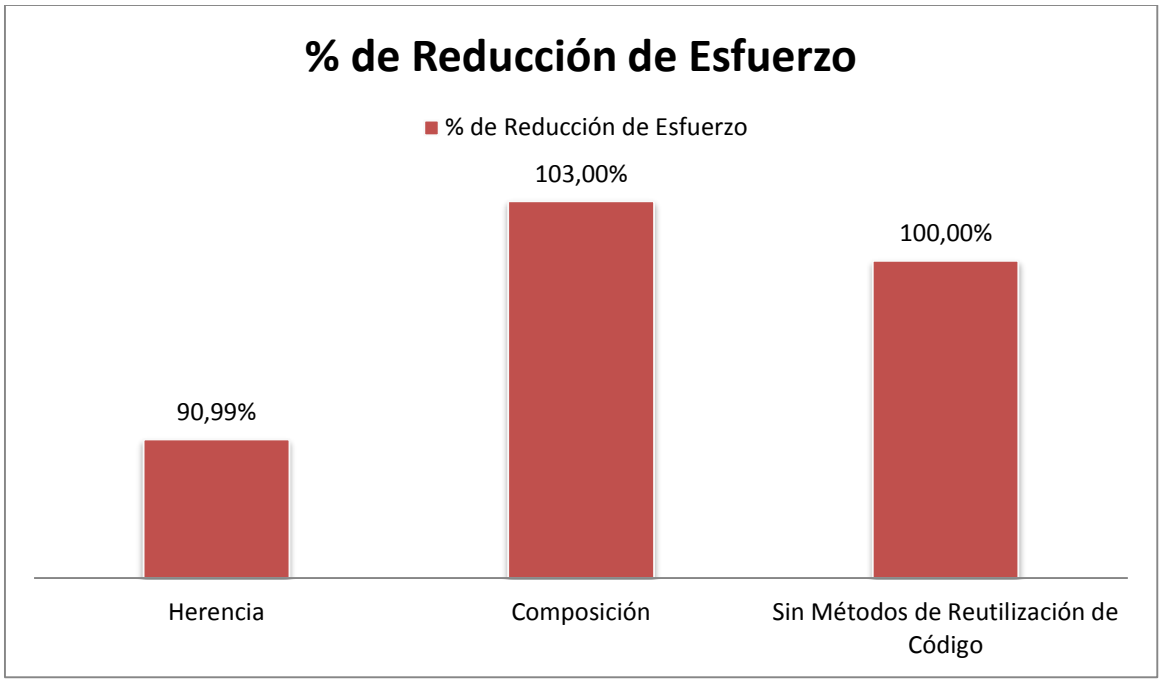


Figura III. 69. Porcentaje de reducción de esfuerzo por prototipo

Fuente: Autores

La disminución de líneas de código válido utilizados para la escritura del prototipo también es notable al momento de utilizar la metodología de reutilización de código llamada herencia ya que al compararlo con el modelo que no utilizó ningún tipo de metodología de reducción de código hay una disminución de 8,04% lo que se encuentra de acorde al indicador del esfuerzo ya que al disminuir el número de líneas de código disminuye el esfuerzo, también se relaciona con el tiempo ya que menos líneas de código escritas significa una reducción en el tiempo de desarrollo; estos valores se contrastan frente a la composición la cual exige más tiempo, más esfuerzo y más líneas de código ya que se nota al compararlo con la herencia una reducción del 11,04% al momento del total de LDC escritas.

Tabla III. XLI: Porcentaje de Reducción deLDC por prototipo

Prototipo	Porcentaje de Reducción de LDC
Herencia	91,96%
Composición	103,11%
Sin Reutilización de Código	100%

Fuente: Autores

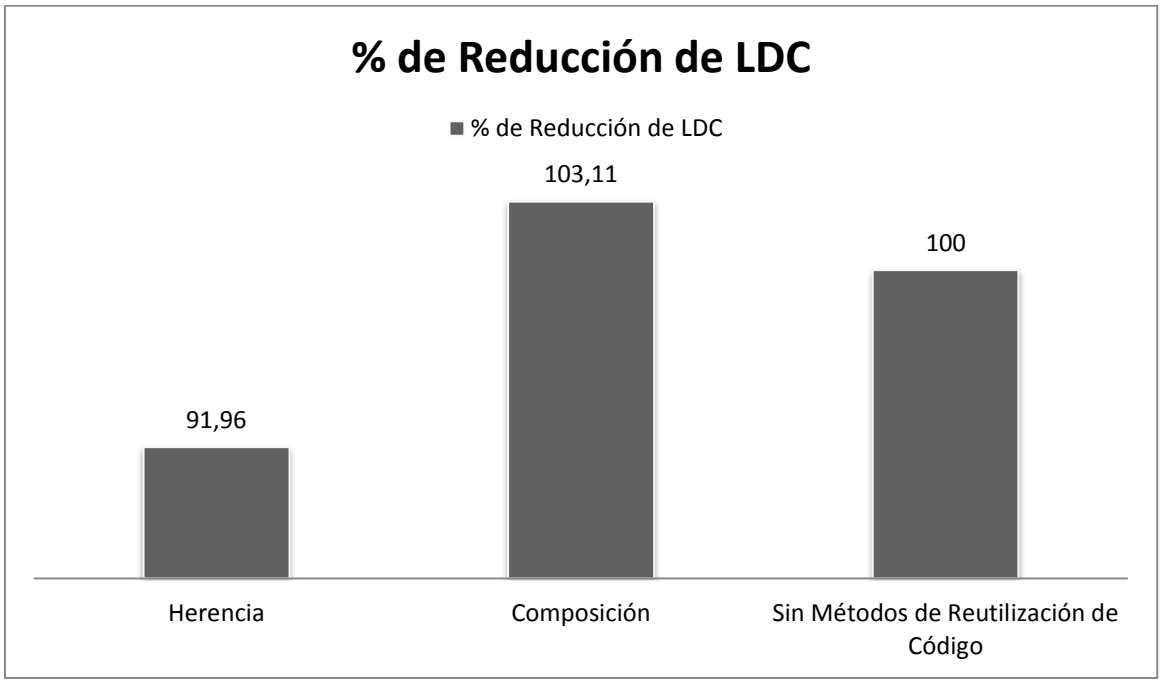


Figura III. 70. Porcentaje de reducción de LDC por prototipo

Fuente: Autores

La Dificultad al momento de desarrollar un proyecto informático esta dado directamente en relación al total de líneas de código escritas, esto se obtiene al comparar datos en la tabla XLI, por lo que no es necesario el realizar cálculos, la dificultad aumenta al incrementarse el total de LDC escritas y válidas por lo que al utilizar la herencia esta ofrece una reducción de la dificultad colocándose en una calificación de **PEQUEÑA**.

Tabla III. XLII: Dificultad por prototipo

Prototipo	Líneas de Código
Herencia	PEQUEÑA
Composición	MEDIA
Sin Reutilización de Código	MEDIA

Fuente: Autores

Los últimos indicadores si bien no están relacionadas al desarrollo de la aplicación en si nos ofrecen un resultado sobre la carga que deberán soportar los servidores donde las aplicaciones en las que se ha utilizado la herencia sean alojadas.

También se tiene otros indicadores (porcentaje de uso del CPU, porcentaje de uso de la memoria RAM) los cuales no están directamente involucrados con la reducción del tiempo de desarrollo de las aplicaciones web, determinando que la herencia ocupa más recursos (CPU, RAM) que los otros prototipos al momento de compilación y posteriormente se reduce los recursos del servidor de manera significativa en tiempo de ejecución.

Tabla III. XLIII: Porcentaje Reducción de Uso de CPU

Prototipo	Promedio Porcentaje de uso (%)	Porcentaje de Reducción de Uso CPU
Herencia	431/135 = 3,1925	74,30%
Composición	579/135 = 4,2888	99,82%
Sin Reutilización de Código	580/135 = 4,2962	100%

Fuente: Autores

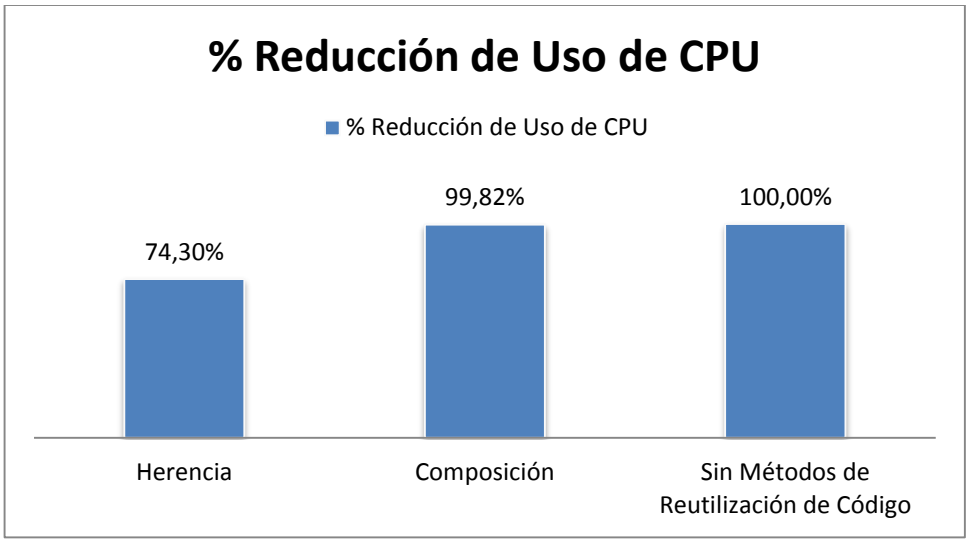


Figura III. 71. Porcentajes de reducción de uso de CPU y memoria RAM

Fuente: Autores

Tabla III. XLIV: Porcentaje Reducción de Uso de RAM

Prototipo	Promedio Porcentaje de uso RAM (%)	Porcentaje de reducción de Uso RAM
Herencia	402/135 = 2,9777	151,12%
Composición	255/135 = 1,8888	95,86%
Sin Reutilización de Código	266/135 = 1,9703	100%

Fuente: Autores

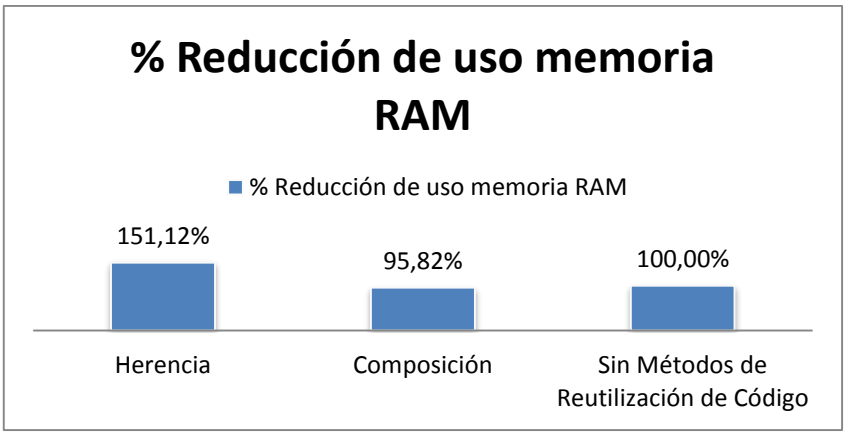


Figura III. 72. Porcentajes de reducción de uso de memoria RAM

Fuente: Autores

5.12. Demostración de la hipótesis

A continuación se citará la hipótesis: La aplicación de métodos de reutilización de código utilizada por Java, permitirá mejorar la productividad en el desarrollo del sistema para Unidad Técnica de Planificación de la ESPOCH.

5.12.1. Operacionalización conceptual

Tabla III. XLV: Operacionalización Conceptual

Variable	Tipo	Concepto
Métodos de reutilización de código	Independiente Cualitativa	Formas de programación para volver a utilizar código creado previamente.
Productividad	Dependiente Cuantitativa	Relación entre los resultados de utilización de métodos de reutilización de código y el tiempo en el desarrollo de aplicaciones web

Fuente: Autores

5.12.2. Operacionalización metodológica

Tabla III. XLVI: Operacionalización Metodológica

Variable	Indicadores	Técnica	Fuente
Métodos de reutilización de código	Tipos de métodos (Herencia y Composición)	Prototipos Software Clases	Creación de prototipos con y sin métodos de reutilización de código.
Productividad	Indicador 1: Tiempo Indicador 2: Esfuerzo Indicador 3: LDC Indicador 4: Dificultad de desarrollo Indicador 5: % uso CPU Indicador 6: % uso RAM	Prototipos Software	Neoload Notepad ++ COCOMO

Fuente: Autores

5.12.3. Asignación de pesos y variables

Para la demostración y validación de nuestra hipótesis se ha creado una tabla de valores, en la cual se indica un rango de valoración para cada indicador, estas valoraciones o pesos sirven para poder establecer un valor total de cada metodología de reutilización de código utilizada.

Estos valores están determinados en la escala del 1 al 6, tomándose en consideración lo siguiente, el peso número 3 es considerado la base para determinar la escala, esta base está asignada al prototipo que no utiliza ningún tipo de reutilización de código, los valores que van de 4 al 6 se consideran para los porcentajes bajos de 100% y los valores del 1 al 2 son utilizados para los porcentajes que sobrepasan el 100%.

Para la determinación de los pesos se utilizarán los porcentajes de reducción de tiempo, esfuerzo, líneas de código, porcentaje de uso de CPU y porcentaje de uso de la memoria RAM, en la dificultad se utilizará una denominación manual.

Tabla III. XLVII: Determinación de pesos

Rangos (%)	Pesos
0 – 33	6
34 – 66	5
67 – 99	4
100	3
101 – 133	2
134 – 166	1

Fuente: Autores

Tabla III. XLVIII: Determinación de pesos para en función de LDC

Categoría	Duración	Líneas de Código	Pesos
Trivial	0 – 4 semanas	< 1k	6
Pequeño	1 – 6 meses	1k – 3k	5
Media	0,5 – 2 años	3k – 50k	4
Grande	2 – 3 años	50k – 100k	3
Muy Grande	4 – 5 años	100k – 1M	2
Gigante	5 – 10 años	> 1M	1

Fuente: Autores

5.12.4. Asignación de pesos

Cada tabla a continuación en estas tablas se asigna los pesos de acuerdo a los valores de la tabla XLVIII.

Tabla III. XLIX: Asignación de pesos: Reducción de tiempos de desarrollo

Prototipo	Porcentaje de Reducción de tiempo	Pesos
Herencia	91 %	4
Composición	103 %	2
Sin Utilizar Métodos de Reutilización	100 %	3

Fuente: Autores

Tabla III. L: Asignación de pesos: Esfuerzo requerido por prototipo

Prototipo	Porcentaje de Reducción del Esfuerzo	Pesos
Herencia	90.99 %	4
Composición	103 %	2
Sin Reutilización de Código	100 %	3

Fuente: Autores

Tabla III. LI: Asignación de pesos: Reducción de LDC por prototipo

Prototipo	Porcentaje de Reducción de LDC	Pesos
Herencia	91,96%	4
Composición	103,11%	2
Sin Reutilización de Código	100%	3

Fuente: Autores

Tabla III. LII: Asignación de pesos: Dificultad requerida por prototipo

Prototipo	Líneas de Código	Pesos
Herencia	PEQUEÑA	5
Composición	MEDIA	4
Sin Reutilización de Código	MEDIA	4

Fuente: Autores

Tabla III. LIII: Asignación de pesos: Porcentaje reducción de uso de CPU

Prototipo	Porcentaje de Reducción de Uso CPU	Pesos
Herencia	74,30%	4
Composición	99,82%	4
Sin Reutilización de Código	100%	3

Fuente: Autores

Tabla III. LIV: Asignación de pesos: Porcentaje reducción de uso de memoria RAM

Prototipo	Porcentaje de reducción de Uso RAM	Pesos
Herencia	151,12%	1
Composición	95,86%	4
Sin Reutilización de Código	100%	3

Fuente: Autores

5.12.5. Comparativa valoración total obtenida por cada prototipo

Tabla III. LV: Asignación de pesos: Resultados finales

Prototipo / Indicadores	Sin reutilización de código	Herencia	Composición
Indicador 1: Tiempo	3	4	2
Indicador 2: Esfuerzo	3	4	2
Indicador 3: LDC	3	4	2
Indicador 4: Dificultad de desarrollo	4	5	4
Indicador 5: % uso CPU	3	4	4
Indicador 6: % uso RAM	3	1	4
Total Valoración	**19**	**22**	**18**

Fuente: Autores

5.12.6. Porcentajes de optimización de productividad de cada prototipo

Tabla III. LVI: Asignación de pesos: Porcentaje aumento de productividad

Prototipo	Porcentaje de aumento de productividad
Herencia	15,79%
Composición	5,27%
Sin Reutilización de Código	0%

Fuente: Autores

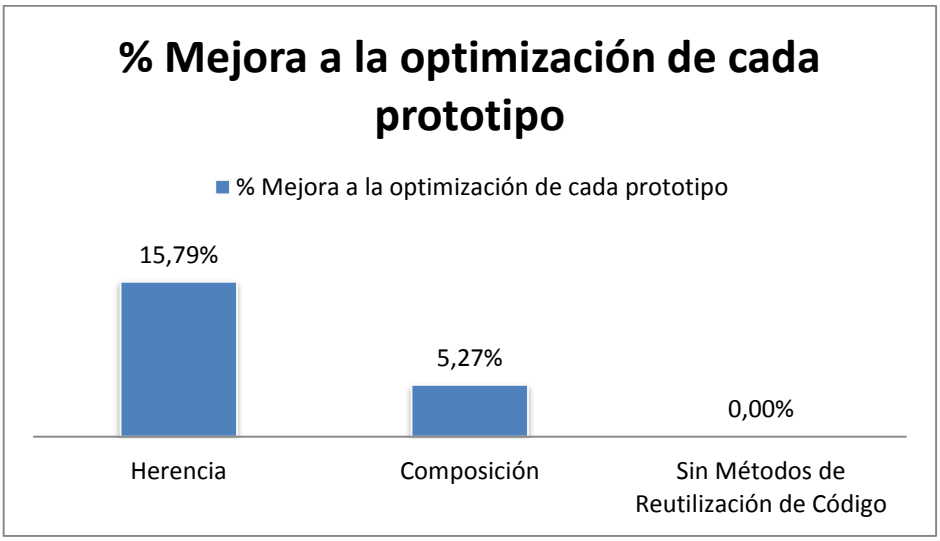


Figura III. 73. Porcentajes de optimización de productividad de cada prototipo

Fuente: Autores

De esta manera mediante la valoración que se realizó de cada indicador se puede demostrar que el método de reutilización de código: herencia demuestra ser la mejor metodología de reutilización de código en java ya que se logra obtener un 15,79% de optimización en la productividad en el desarrollo del sistema para la Unidad Técnica de Planificación de la ESPOCH, esta valoración es superior a la obtenida por el método de reutilización de código: composición que solo consiguió lograr una optimización del 5,27%,la herencia demuestra que contribuye a que el

tiempo de desarrollo, esfuerzo, dificultad y líneas de código producidos durante el desarrollo de una aplicación disminuya de manera apreciable, en contraste la composición solo contribuye a una optimización muy baja en la productividad de desarrollo del sistema.

De esta forma la Hipótesis dada para el estudio realizado se considera válida ya que la utilización de métodos de reutilización de código en java contribuye a mejorar la productividad de desarrollo de software, en especial la aplicación del método llamado herencia la cual muestra un porcentaje de optimización elevado.

CAPÍTULO IV

DESARROLLO DEL SISTEMA (SIG POI)

Como se puede apreciar en el capítulo anterior el método de reutilización de código herencia es la más adecuada para el desarrollo de aplicaciones web.

Por dicha razón este método de reutilización de código se usará para desarrollar el Sistema (SIGPOI en adelante) con una metodología de desarrollo de software eXtreme Programming (XP).

El sistema se desarrolló y lo utilizará la UTP de la ESPOCH, el mismo que ayudará a llevar un control automatizado de todos los procesos que se llevan a cabo en dicho departamento.

Todos estos temas serán tratados en el transcurso del presente capítulo, definiendo conceptos, características de la metodología y el desarrollo del sistema en sí.

12.1. Metodología XP

La Programación Extrema o XP nace oficialmente hace cinco años fundada por Kent Beck, es el más destacado de los procesos ágiles de desarrollo de software. La programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad.

Los defensores de la XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos.

Esta metodología de desarrollo de software posee cuatro características básicas que debe reunir el programador XP que son: la simplicidad, la comunicación y la retroalimentación o reutilización del código desarrollado (reciclado de código).

El ciclo de vida de XP se define en cuatro tareas fundamentales como se indica en la Figura 74.

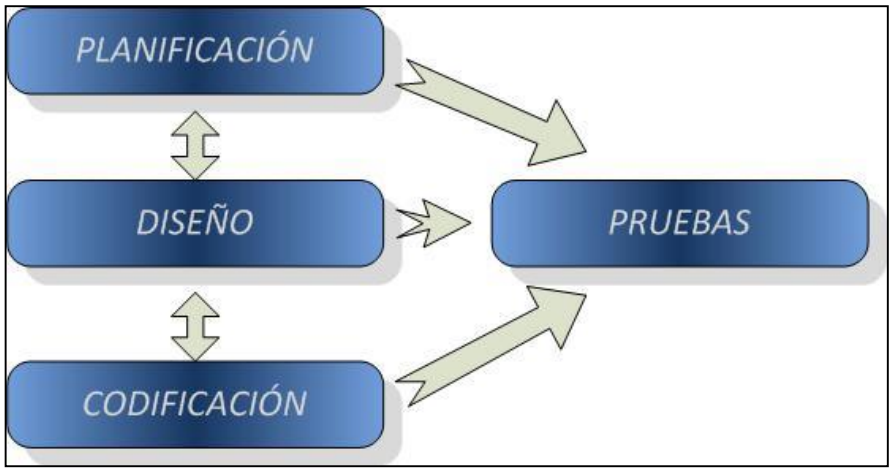


Figura IV. 74. Ciclo de vida XP

Fuente: Autores

12.2. Desarrollo del Sistema

En esta parte se detallara el desarrollo del sistema indicando los prototipos y las pantallas funcionales que manejará el usuario final.

12.2.1. Gestión del proyecto

Planificación del proyecto

Esta planificación se realizó al inicio de éste capítulo, tras estudiar el problema y reunir los requerimientos necesarios. De esta redacción inicial de historias de usuario se realizó una planificación inicial y posteriormente fue cambiada a lo largo del mismo, eliminando o cambiando historias de usuario, a medida que se tenía una concepción más clara del problema.

Integrantes y roles

Teniendo en cuenta la participación tanto de los jefes de proyecto, como de los usuarios y desarrolladores, se formará así el equipo encargado de la implementación de software. Esto implicará que los diseños deberán ser claros y

sencillos, los usuarios deberán disponer de versiones operativas cuanto antes para poder participar en el proceso creativo mediante sus sugerencias y aportaciones, el potenciar al máximo el trabajo en equipo es fundamental para el desarrollo del sistema, el equipo trabajo se ve ilustrado en la Tabla LVII definiendo Integrantes y Roles:

Tabla IV. LVII. Integrantes y Roles

Miembro	Grupo	Roles XP	Metodología
Carlos López	Tesistas	Rastreador, Testeador, Programador	**Xp**
Noe Remache	Tesistas	Rastreador, Programador, Testeador.	
Ing. Ivonne Rodriguez **Ing. María Isabel Uvidia** **Ing. Rosa Vallejo**	**Consultor**	**Entrenador**	

Fuente: Autores

12.2.2. Prototipos

Las interfaces de usuario son las interfaces más importantes ya que de esta dependerá el entendimiento fácil y rápido por parte del usuario al comenzar a manipular el sistema, la amigabilidad de la interfaz reside en el uso de ventanas, cuadros de diálogo, gráficos, etc. Se pretende lograr una interfaz sencilla y funcional, con un alto grado de compresión, por estas razones se crearon prototipos generales del sistema

A continuación una breve descripción del proceso principal del proyecto:

- Indicándose en la Figura 75 el inicio de sesión de usuarios:

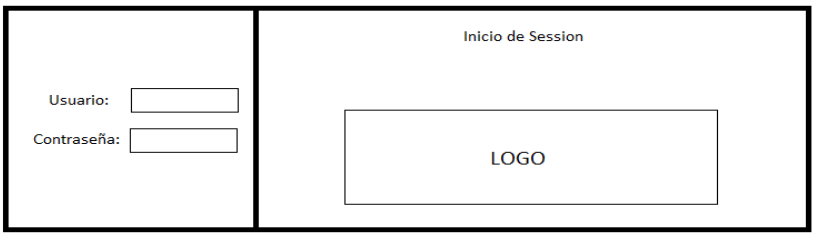


Figura IV. 75. Acceso de Usuarios

Fuente: Autores

- En lasfiguras76 - 78 se muestra la pantalla de FORMULACIÓN de PROYECTO la cual se divide en tres secciones que son: Datos Generales, Datos Específicos, Vinculación PEDI

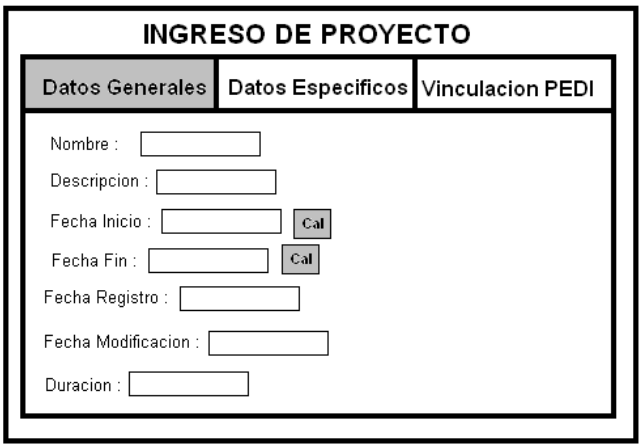


Figura IV. 76. Ingreso Proyecto: Datos Generales

Fuente: Autores

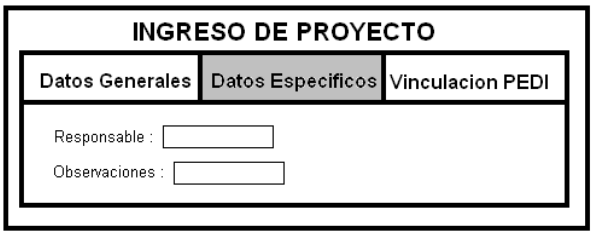


Figura IV. 77. Ingreso Proyecto: Datos Específicos

Fuente: Autores

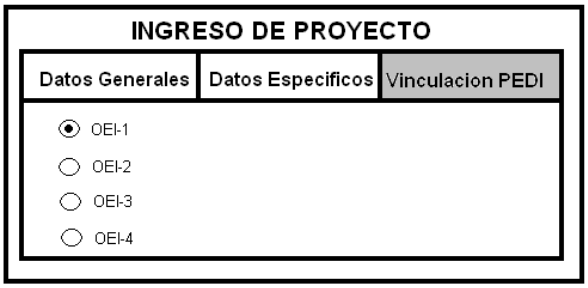


Figura IV. 78. Ingreso Proyecto: Vinculación PEDI

Fuente: Autores

- En la figura 79 se muestra la pantalla de FORMULACIÓN de POA

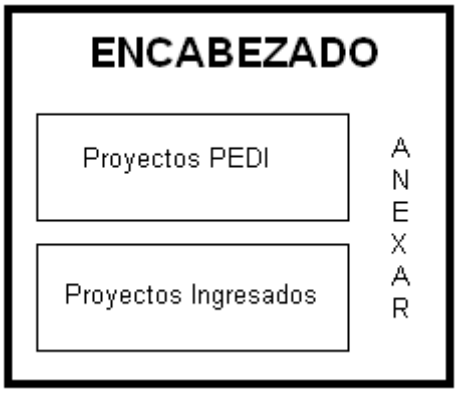


Figura IV. 79. Formulación POA

Fuente: Autores

- En la figura 80 se muestra la pantalla de PRIORIZACIÓN de PROYECTOS

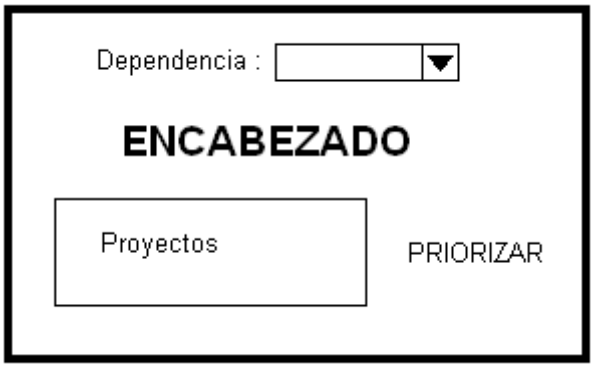


Figura IV. 80. Priorización de Proyectos

Fuente: Autores

- En la figura 81 se muestra la pantalla de DETERMINACIÓN de PROYECTOS

Figura IV. 81. Determinación de Proyectos
Fuente: Autores

- En la figura 82 se muestra la pantalla de FORMULACIÓN de PAI

Figura IV. 82. Formulación PAI
Fuente: Autores

12.2.3. Historias de usuarios

Las historias de usuario tienen como propósito ver las necesidades del sistema; por tanto serán descripciones cortas y escritas en el lenguaje del usuario, sin terminología técnica, además proporcionarán los detalles sobre la estimación del riesgo y cuánto tiempo conllevará la implementación de dicha historia de usuario.

Una historia se divide en varias tareas planificables y medibles en su estado de realización, estas forman parte de una iteración, este plan indicará, por tanto,

diferentes iteraciones del sistema y que se debe implementar en cada una de ellas. La realización de este plan debe tener en cuenta en la medida de lo posible, las prioridades de los usuarios, para satisfacerles en mayor medida, como se indica en la Tabla LVIII:

Tabla IV. LVIII. Historias de Usuarios

N°	NOMBRE	PRIORIDAD	RIESGO	ESFUERZO	ITERACION
1	Control de Acceso de Usuarios	Alta	Alto	Medio	1
2	Configuración del Sistema	Alta	Alto	Alto	1
3	Gestión PEDI, Objetivo Estratégico, Política, Estrategia	Alta	Alto	Alto	1
4	Gestión Función, Objetivo Operativo, Meta, Indicadores	Alta	Alto	Alto	1
5	Gestión Programa, Proyecto del PEDI	Alta	Alto	Alto	1
6	Gestión Proyectos	Alta	Alto	Alto	2
7	Gestión Meta	Alta	Alto	Alto	2
8	Gestión Financiamiento	Alta	Alto	Alto	2
9	Gestión Indicador	Alta	Alto	Alto	2
10	Gestión Programación	Alta	Alto	Alto	2
11	Control de Priorización de Proyectos	Alta	Alto	Alto	3
12	Determinación de Proyectos	Alta	Alto	Alto	3
13	Formulación POA y PAI	Alta	Alto	Alto	3
14	Control de Reportes	Alta	Alto	Alto	3

Fuente: Autores

12.2.4. Plan de entregas

El plan de entregas se usará para crear los planes de iteración para cada iteración. Es en este momento cuando los técnicos tomarán las decisiones. En esta reunión estarán presentes tanto desarrolladores como usuarios.

Con cada historia de usuario previamente evaluada en tiempo de desarrollo ideal, el usuario las agrupará en orden de importancia.

De esta forma se puede trazar el plan de entregas en función de estos dos parámetros: *tiempo de desarrollo ideal y grado de importancia para el usuario*. Las iteraciones individuales son planificadas en detalle justo antes de que comience cada iteración como se puede apreciar en las siguientes tablas y figuras.

Iteración 1

Tabla IV. LIX. Plan de Entrega Iteración 1.

Historia de Usuario	Duración en semanas
Control de Acceso de Usuarios	3
Configuración del Sistema	3
Gestión PEDI, Objetivo Estratégico, Política, Estrategia	3
Gestión Función, Objetivo Operativo, Meta, Indicadores	3
Gestión Programa, Proyecto del PEDI	3

Fuente: Autores

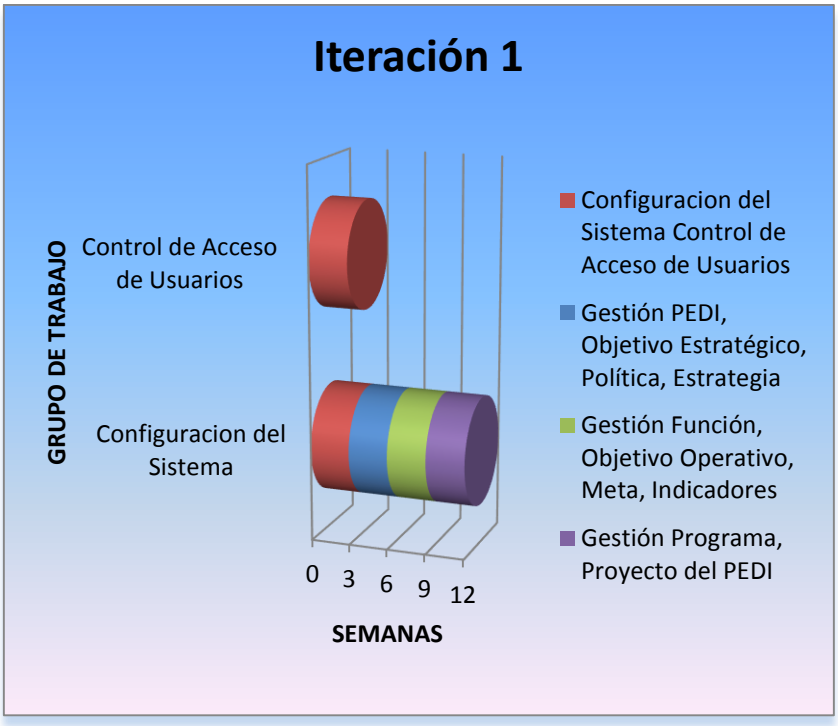


Figura IV. 83. Plan de Entrega. Iteración 1.

Fuente: Autores

Iteración 2

Tabla IV. LX. Plan de Entrega Iteración 2.

Historia de Usuario	Duración en semanas
Gestión Proyectos	3
Gestión Meta	3
Gestión Financiamiento	3
Gestión Indicador	3
Gestión Programación	3

Fuente: Autores

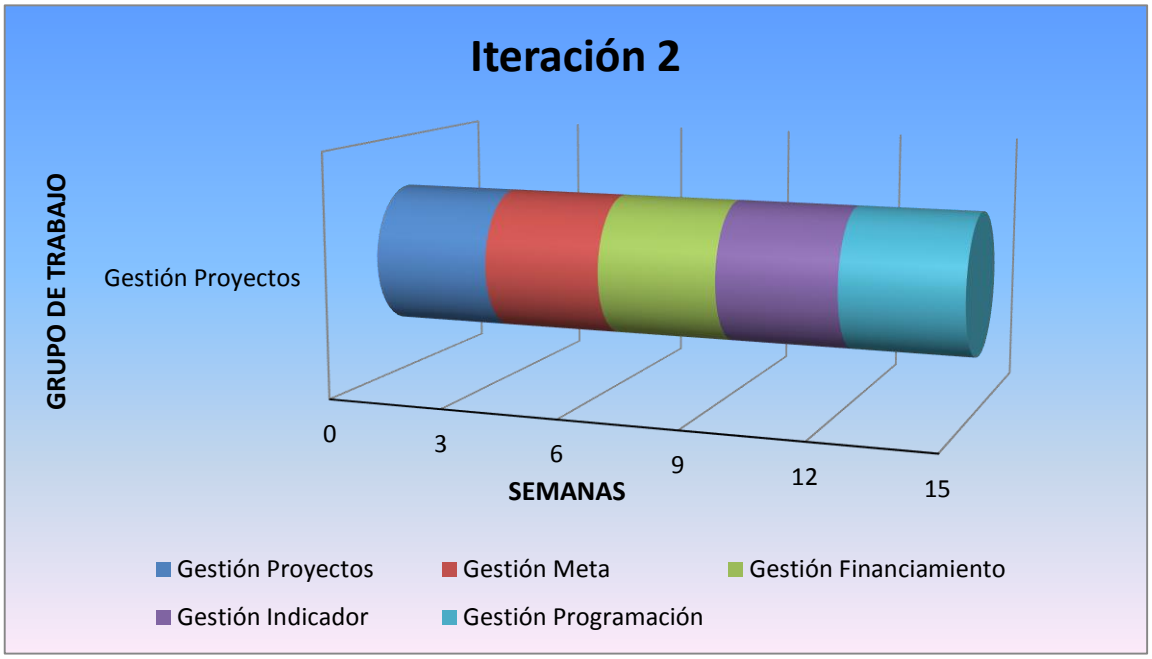


Figura IV. 84. Plan de Entrega. Iteración 2.

Fuente: Autores

Iteración 3

Tabla IV. LXI. Plan de Entrega Iteración 3.

Historia de Usuario	Duración en semanas
Control de Priorización de Proyectos	3
Determinación de Proyectos	3
Formulación POA y PAI	3
Control de Reportes	3

Fuente: Autores

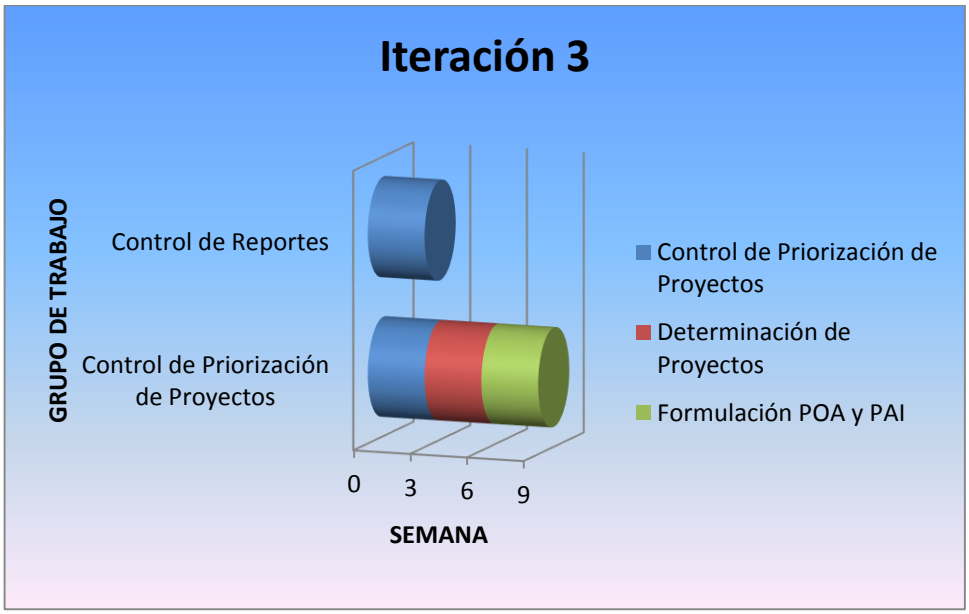


Figura IV. 85. Plan de Entrega. Iteración 3.

Fuente: Autores

12.2.5. Incidencia

Iteración primera: Se tratara de tener preparadas las funcionalidades básicas relacionadas con el usuario. Tras esta fase nos ha quedado claro que es prácticamente imposible crear una planificación inmutable, se debe esperar que la planificación mostrada en principio varíe así como la desaparición y sustitución de algunas historias de usuario. El ensamblaje de un grupo de trabajo es una labor

larga y laboriosa, pequeños problemas como la selección de herramientas y unificación de horarios se convierten en principal piedra de toque de esta fase.

Iteración segunda: La importancia del usuario final ha quedado demostrada como máxima, ya que la visión de los miembros del equipo puede llegar a tener una interpretación distinta a la hoja de usuario.

Iteración tercera: Siendo esta la última iteración se pretende entregar el producto acabado con todas las funcionalidades propuestas por el usuario.

A continuación se describirá de una manera más detallada, iteración por iteración y cada una de las historias de usuario que las conforman (Ver tablas LXII - LXXV).

Iteración 1

Tabla IV. LXII. Iteración 1. Historia 1.

Historia de Usuario	
Número: 1	**Usuario:***Planificación, Directores de Unidades Académicas y Administrativas, Rector.*
Nombre historia: *Control de Acceso de Usuarios.*	
Prioridad en negocio: Alta	**Riesgo en desarrollo:***Alto*
Esfuerzo: Medio	**Iteración asignada:** *1*
Programador responsable: *Carlos López / Noe Remache*	
Descripción: *Antes de iniciar la aplicación se solicita la cuenta de usuario y su clave para que tenga acceso a los datos que corresponden a su tipo de usuario.*	
Observaciones: *Hay tres tipos de usuarios: Planificación, Rector, Directores de Unidades Académicas y Administrativas con distintos permisos de acceso a los menús dependiendo de las funciones que les corresponden.*	

Fuente: Autores

Tabla IV. LXIII. Iteración 1. Historia 2.

Historia de Usuario	
Número: 2	**Usuario**: *Planificación*
Nombre historia: Configuración del Sistema	
Prioridad en negocio: Alta	**Riesgo en desarrollo**: *Alto*
Esfuerzo: Alto	**Iteración asignada**: *1*
Programador responsable: *Carlos López / Noe Remache*	
Descripción: *El usuario Planificación debe autenticarse en el sistema. En el cual deberá escoger el rol CONFIGURACIONES DEL SISTEMA, en el mismo que se encuentra la opción Fechas de apertura y cierre de los procesos, asignación de techos presupuestarios de cada unidad, elección de la programación de los indicadores de los proyectos.*	
Observaciones:	

Fuente: Autores

Tabla IV. LXIV. Iteración 1. Historia 3

Historia de Usuario	
Número: 3	**Usuario**:Planificación
Nombre historia: Gestión PEDI, Objetivo Estratégico, Política, Estrategia	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** *Alto*
Esfuerzo: Alto	**Iteración asignada**: *1*
Programador responsable: *Carlos López / Noe Remache*	
Descripción: *El usuario Planificación debe autenticarse en el sistema. En el cual deberá escoger el rol ADMINISTRACION DEL PND Y PEDI, en el mismo que se encuentra la opción Administrar Plan Estratégico Institucional en el cual nos presenta las opciones: Ingresar Plan Estratégico Institucional, Objetivo Estratégico, Política, Estrategia.*	
Observaciones*: Para insertar un nuevo Objetivo Estratégico se deberá contemplar el ingreso previo del PEDI.*	

Fuente: Autores

Tabla IV. LXV. Iteración 1. Historia 4

Historia de Usuario	
Número: 4	**Usuario**: Planificación
Nombrehistoria: Gestión Función, Objetivo Operativo, Meta, Indicadores	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alto
Esfuerzo: **Alto**	**Iteración asignada:** 1
Programador responsable: Carlos López / Noe Remache	
Descripción: *El usuario Planificación debe autenticarse en el sistema. En el cual deberá escoger el rol ADMINISTRACION DEL PND Y PEDI, en el mismo que se encuentra la opción Administrar Plan Estratégico Institucional en el cual nos presenta las opciones: Ingresar Función, Objetivo Operativo, Meta, Indicadores.*	
Observaciones: *Para el ingreso mencionado anteriormente, se necesita que estén ingresados los datos anteriores.*	

Fuente: Autores

Tabla IV. LXVI. Iteración 2. Historia 5

Historia de Usuario	
Número: 5	**Usuario**: *Planificación*
Nombre historia:*Gestión Programa, Proyecto del PEDI*	
Prioridad en negocio: Alta	**Riesgo en desarrollo**: *Alto*
Esfuerzo: Alto	**Iteración asignada**: *1*
Programador responsable: *Carlos López / Noe Remache*	
***Descripción**: El usuario Planificación debe autenticarse en el sistema. En el cual deberá escoger el rol ADMINISTRACION DEL PND Y PEDI, en el mismo que se encuentra la opción Administrar Plan Estratégico Institucional en el cual nos presenta las opciones: Ingresar Programa. Para el Proyecto del PEDI deberá escoger el rol INGRESO DE PROYECTOS, el cual se detalla más adelante.*	
Observaciones:	

Fuente: Autores

Iteración 2

Tabla IV. LXVII. Iteración 2. Historia 6

Historia de Usuario	
Número: 6	**Usuario:***Planificación, Directores de Unidades Académicas y Administrativas, Rector.*
Nombre historia:*Gestión Proyectos*	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** *Alto*
Esfuerzo:*Alto*	**Iteración asignada:*2***
Programador responsable: *Carlos López / Noe Remache*	
***Descripción**: Los usuarios una vez registrado en el sistema, en el cual deberá escoger el rol INGRESO DE PROYECTOS, se deberá escoger el Submenú Ingresar Proyecto con sus Datos Generales, Específicos y Vinculación con PEDI*	
***Observaciones**: Para el ingreso de los proyectos deberá controlar que todos y cada uno de los campos tenga datos llenos para evitar inconvenientes.*	

Fuente: Autores

Tabla IV. LXVIII. Iteración 2. Historia 7

Historia de Usuario	
Número:7	**Usuario**:*Planificación, Directores de Unidades Académicas y Administrativas, Rector.*
Nombre historia:*Gestión Meta*	
Prioridad en negocio: Alta	**Riesgo en desarrollo**: *Alto*
Esfuerzo: Alto	**Iteración asignada**: *2*
Programador responsable: *Carlos López / Noe Remache*	
Descripción: *El usuario una vez registrado en el sistema, en el cual deberá escoger el rol INGRESO DE META Y FINANCIAMIENTO, deberán ingresar la meta a cumplir del proyecto.*	
Observaciones: *Se debe tener ingresado previamente el proyecto.*	

Fuente: Autores

Tabla IV. LXIX. Iteración 2. Historia 8

Historia de Usuario	
Número: 8	***Usuario**: Planificación, Directores de Unidades Académicas y Administrativas, Rector.*
***Nombre historia:**Gestión Financiamiento*	
Prioridad en negocio: Alta	***Riesgo en desarrollo:** Alto*
Esfuerzo: Alto	***Iteración asignada:** 2*
***Programador responsable:** Carlos López / Noe Remache*	
***Descripción**: Los usuarios una vez registrados en el sistema, en el cual deberá escoger el rol INGRESO DE META Y FINANCIAMIENTO, deberán ingresar la meta a cumplir del proyecto.*	
***Observaciones**: Se debe tener ingresado previamente el proyecto.*	

Fuente: Autores

Tabla IV. LXX. Iteración 2. Historia 9

Historia de Usuario	
Número: 9	***Usuario**: Planificación, Directores de Unidades Académicas y Administrativas, Rector.*
***Nombre historia**: Gestión Indicador*	
Prioridad en negocio: Alta	***Riesgo en desarrollo**: Alto*
Esfuerzo: Alto	***Iteración asignada**: 2*
***Programador responsable**: Carlos López / Noe Remache*	
***Descripción**: El usuario una vez registrado en el sistema, en el cual deberá escoger el rol INGRESO DE INDICADOR Y PROGRAMACION, deberá seleccionar la meta del proyecto para asignar los indicadores.*	
***Observaciones**:*	

Fuente: Autores

Tabla IV. LXXI. Iteración 2. Historia 10

Historia de Usuario	
Número: 10	**Usuario:** *Planificación, Directores de Unidades Académicas y Administrativas, Rector.*
Nombre historia:*Gestión Programación*	
Prioridad en negocio:*Alta*	**Riesgo en desarrollo:** *Alto*
Esfuerzo: *Alto*	**Iteración asignada:***2*
Programador responsable: *Carlos López / Noe Remache*	
Descripción: *El usuario una vez registrado en el sistema, en el cual deberá escoger el rol INGRESO DE INDICADOR Y PROGRAMACION, deberá seleccionar el indicador a programar para luego ingresar la programación de acuerdo a lo configurado en la Historia 2.*	
Observaciones:	

Fuente: Autores

Iteración 3

Tabla IV. LXXII. Iteración 2. Historia 11

Historia de Usuario	
Número: 11	**Usuario**: *Rector*
Nombre historia:*Control de Priorización de Proyectos*	
Prioridad en negocio: Alta	**Riesgo en desarrollo**: *Alto*
Esfuerzo: Alto	**Iteración asignada**: *3*
Programador responsable: *Carlos López / Noe Remache*	
Descripción: *El usuario debe autenticarse en el sistema. En el cual deberá escoger el rol PRIORIZAR PROYECTOS, en el mismo q se encuentra la opción Priorizar proyectos en la cual deberá escoger la Facultad / Unidad y luego escoger el proyecto a priorizar.*	
Observaciones: *Proyecto previamente ingresado.*	

Fuente: Autores

Tabla IV. LXXIII. Iteración 3. Historia 12

Historia de Usuario	
Número: 12	**Usuario:***Planificación*
Nombre historia:*Determinación de Proyectos*	
Prioridad en negocio: Alta	**Riesgo en desarrollo***: Alto*
Esfuerzo: Alto	**Iteración asignada**: *3*
Programador responsable: *Carlos López / Noe Remache*	
Descripción:*El usuario debe autenticarse en el sistema. En el cual deberá escoger el rol DETERMINACION DE PROYECTOS, en el mismo que se encuentra la opción Clasificar Proyecto en la cual deberá escoger el proyecto para determinar el tipo de proyecto al cual será contemplado.*	
Observaciones:*Proyecto previamente ingresado y priorizado.*	

Fuente: Autores

Tabla IV. LXXIV. Iteración 3. Historia 13

Historia de Usuario	
Número: 13	**Usuario**:*Planificación, Directores de Unidades Académicas y Administrativas, Rector.*
Nombre historia:*Formulación POA y PAI*	
Prioridad en negocio: *Alta*	***Riesgo en desarrollo:*** *Alto*
Esfuerzo:*Alto*	***Iteración asignada:*** *3*
Programador responsable: *Carlos López / Noe Remache*	
Descripción: *Los usuarios deben autenticarse en el sistema. En el cual deberá escoger el rol INGRESO DE POA Y PAI, en el mismo q se encuentra la opción Crear PAI en la cual deberá escoger el proyecto para anexar y luego ingresar la información restante con forme al formulario 5 de SEMPLADES.*	
Observaciones: *Proyecto previamente ingresado y priorizado.*	

Fuente: Autores

Tabla IV. LXXV. Iteración 3. Historia 14

Historia de Usuario	
Número: 14	**Usuario**: *Planificación, Directores de Unidades Académicas y Administrativas, Rector.*
Nombre historia: *Control de Reportes*	
Prioridad en negocio: *Alta*	**Riesgo en desarrollo:** *Alto*
Esfuerzo: Alto	***Iteración asignada:*** *3*
Programador responsable: *Carlos López / Noe Remache*	
Descripción: *Los usuarios deben autenticarse en el sistema con sus cuentas respectivamente podrán visualizar los reportes correspondientes a Plan Nacional, Plan Estratégico, Proyectos, Priorización, POA, PAI.*	
Observaciones:	

Fuente: Autores

12.2.6. Actividades

Las actividades de nuestro sistema fueron divididas en varios procesos reflejados en los siguientes diagramas o flujos de procesos:

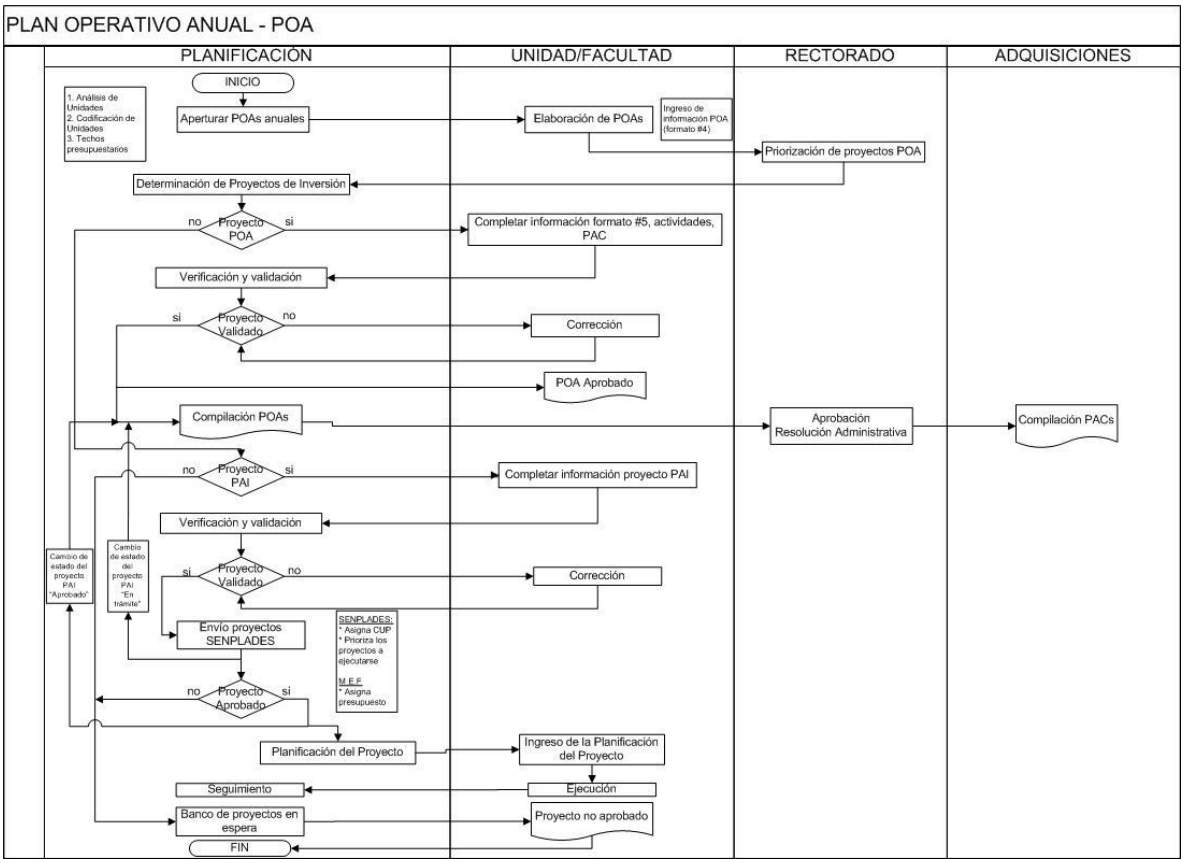


Figura IV. 86. Diagrama de procesos

Fuente: Autores

- Proceso1: Ingreso de Proyectos, proceso en el cual se desarrolla todo lo que conlleva a los datos de un proyecto como se muestra en la figura 86, ingreso de datos generales, específicos y la vinculación con el PEDI.
- Proceso 2: Elaboración del POA, proceso en el cual se desarrolla todo lo que conlleva a la elaboración del POA como se indica en la figura 86, asignar los proyectos de cada facultad / unidad que irán al POA.

- Proceso 3: Priorización de Proyectos, proceso en el cual se desarrolla todo lo que conlleva a la priorización de los proyectos como se muestra en la figura 86, indicando el valor y la semaforización del proyecto.
- Proceso 4: Determinación de Proyectos, donde se detalla todo lo que conlleva a la especificación del proyecto indicando a qué tipo de Plan corresponde como se muestra en la figura 86, asignación del tipo de proyecto (POA / PAI).
- Proceso 5: Completar formulario 5, donde se muestra el ingreso de los demás datos correspondientes a un proyecto como contempla el formulario 5 del formato SEMPLADES como se muestra en la figura 86

Tambien se destacará el cronograma de actividades en la figura 40:

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	Recolección de Datos	5 días	lun 21/05/12	vie 25/05/12	
2	**Gestión del Proyecto**	**10 días**	**lun 28/05/12**	**vie 08/06/12**	**1**
3	Planificación Inicial	2 días	lun 28/05/12	mar 29/05/12	
4	Iteración 1	2 días	mié 30/05/12	jue 31/05/12	3
5	Iteración 2	2 días	vie 01/06/12	lun 04/06/12	4
6	Iteración 3	2 días	mar 05/06/12	mié 06/06/12	5
7	Diario de Actividades	2 días	jue 07/06/12	vie 08/06/12	6
8	Implementación	218 días	lun 11/06/12	mié 10/04/13	2
9	Pruebas	27 días	jue 11/04/13	vie 17/05/13	8

Figura IV. 87. Cronograma de Actividades

Fuente: Autores

12.3. Implementación

En esta sección se detallará todo lo utilizado con respecto a la implementación del sistema como los esquemas de base de datos, el código fuente, los prototipos de interfaces del usuario final.

12.3.1. Base de datos

Nuestra base de datos consta de nueve esquemas:

- ✓ **CONFIGURACIÓN:** En este esquema se maneja todo lo concerniente con la configuración del sistema, usuarios, fechas de apertura y cierre de los procesos, asignación de techos presupuestarios de cada una de las facultades / unidades
- ✓ **PND:** en este esquema se encuentran las tablas informativas del Plan Nacional de Desarrollo el cual no posee interfaz.
- ✓ **PEDI:** en este esquema se encuentra todo lo referente a la gestión del Plan Estratégico de Desarrollo Institucional.
- ✓ **PROYECTO:** en este esquema se encuentran las tablas para el almacenamiento de la información de los proyectos con sus datos principales.
- ✓ **PRY_PRIORIZACION:** en este esquema se encuentra todo lo referente a la gestión de la priorización de los proyectos.
- ✓ **PRY_BENEFICIARIO:** en este esquema se encuentran las tablas para elegir a los beneficiarios que pertenecen a cada uno de los proyectos.
- ✓ **PRY_UBICGEO:** en este esquema se encuentran las tablas para elegir la ubicación geográfica a la cual va dirigido el proyecto.
- ✓ **POA:** en este esquema se encuentra todo lo referente a la gestión del Plan Operativo Anual (POA) para posteriormente ocupar en los reportes.
- ✓ **PAI:** en este esquema se encuentra todo lo referente a la gestión del Plan Anual de Inversión (PAI) además de la información del formulario 5.

El diseño de los esquemas de la Base de Datos podrá ser apreciado de mejor manera en el Anexo 4.

12.3.2. Interfaces de usuario finales

Con la descripción detallada de las historias de usuario y con los diagramas de procesos se definirá las interfaces de usuario finales (Ver figura 88 a la figura 45) las cuales serán implantadas en el sistema.

Figura IV. 88. Control de Acceso de Usuarios

Fuente: Autores

- En las figuras 89 - 91 se muestra la pantalla de FORMULACIÓN de PROYECTO la cual se divide en tres secciones que son: Datos Generales, Datos Específicos, Vinculación PEDI

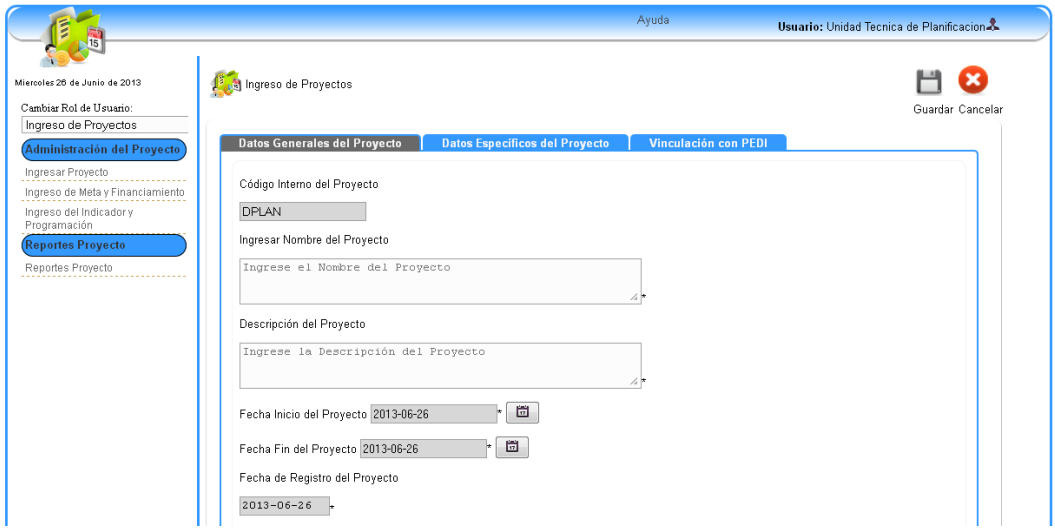


Figura IV. 89. Ingreso Proyecto: Datos Generales

Fuente: Autores

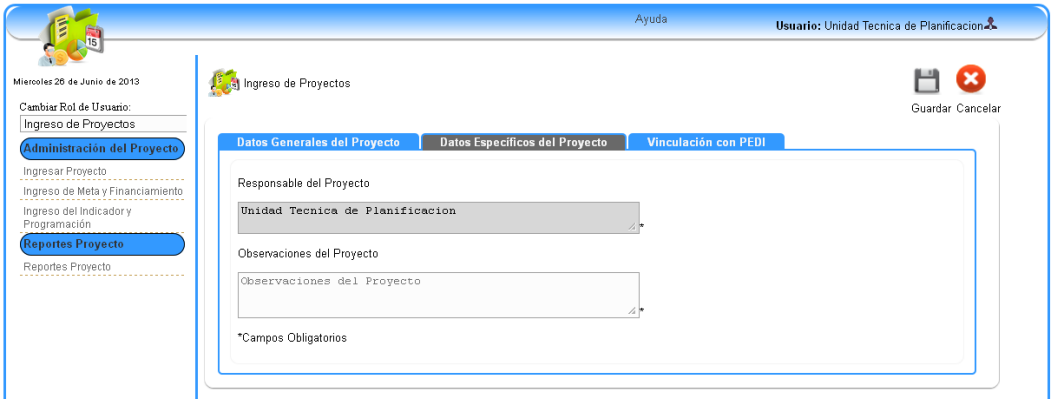


Figura IV. 90. Ingreso Proyecto: Datos Específicos

Fuente: Autores

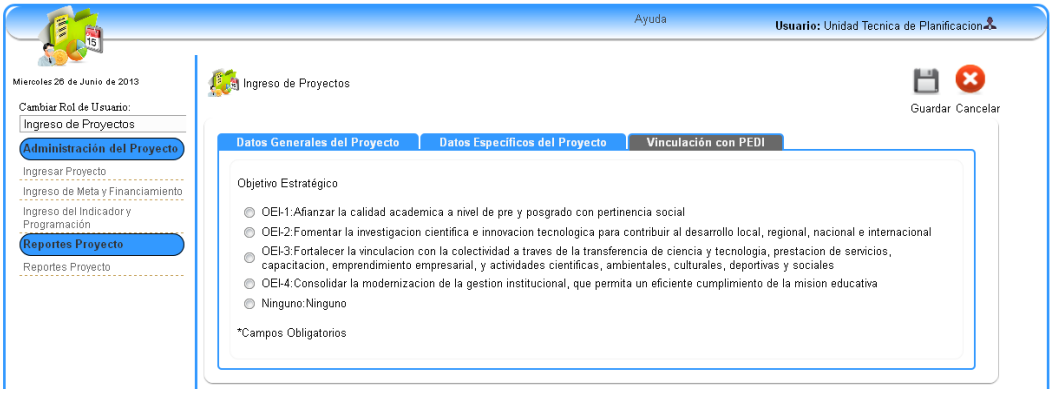


Figura IV. 91. Ingreso Proyecto: Vinculación PEDI

Fuente: Autores

- En la figura 92 se muestra la pantalla de FORMULACIÓN de POA

Figura IV. 92. Formulación POA

Fuente: Autores

- En la figura 93 se muestra la pantalla de PRIORIZACIÓN de PROYECTOS

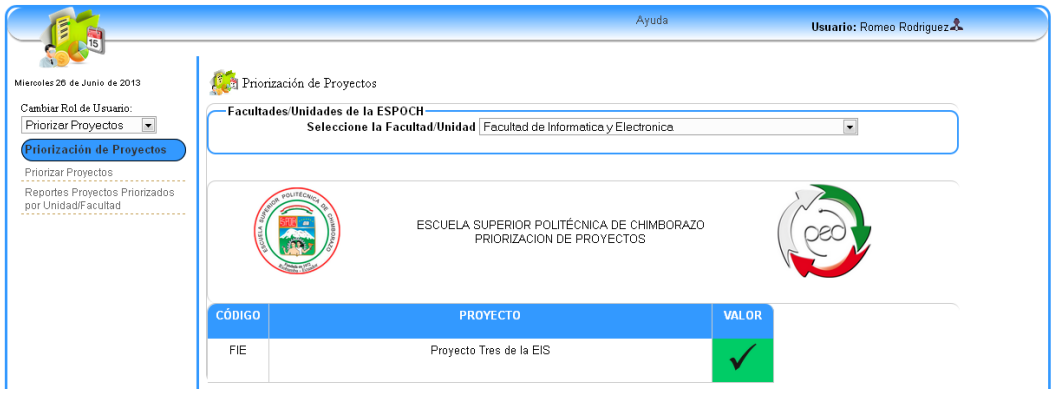


Figura IV. 93. Priorización de Proyectos

Fuente: Autores

➢ En la figura 94 se muestra la pantalla de DETERMINACIÓN de PROYECTOS

Figura IV. 94. Determinación de Proyectos
Fuente: Autores

➢ En la figura 95 se muestra la pantalla de FORMULACIÓN de PAI

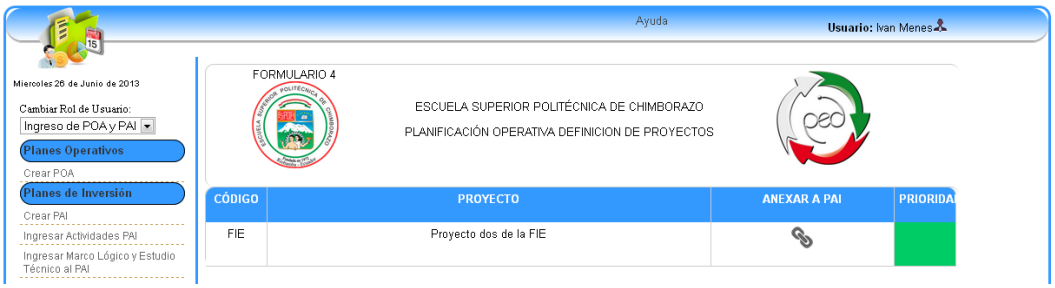


Figura IV. 95. Formulación PAI
Fuente: Autores

12.3.3. Código fuente

El código fuente se ha tomado de la solicitud, puesto que todas y cada una de las clases, funciones controladores y páginas JSP son similares en estructura.

12.3.4. Pruebas

Las pruebas se convierten en una herramienta de desarrollo, no un paso de verificación que puede despreciarse si a uno le parece que el código está bien. Las pruebas son creadas a partir de las historias de usuario. Durante una

iteración la historia de usuario seleccionada en la planificación de iteraciones se convertirá en una prueba de aceptación. El usuario especifica los aspectos a testear cuando una historia de usuario ha sido correctamente implementada. Una historia de usuario puede tener más de una prueba de aceptación, tantas como sean necesarias para garantizar su correcto funcionamiento.

Cada una de ellas representa una salida esperada del sistema. Una historia de usuario no se considera completa hasta que no supera sus pruebas de aceptación.

A continuación se presenta las pruebas realizadas a cada una de las historias de usuarios del sistema y cada una de estas consta con su tabla de pruebas respectivamente.

Historia 1: Control de Acceso a Usuarios

Tabla IV. LXXVI. Pruebas. Historia 1.

Fecha	Descripción	Autor
11/04/2013	Pruebas y Modificaciones	Carlos López / Noe Remache

Fuente: Autores

Descripción

Hay tres tipos de usuarios: Planificación, Rector, Directores de Unidades Académicas y Administrativas con distintos permisos de acceso a los menús dependiendo de las funciones que les corresponden.

Condiciones de ejecución

Cada uno de los usuarios deberá constar en la base de datos previamente registrados.

Entrada

✓ El usuario introducirá su cuenta y clave.

✓ El proceso de control de Acceso a Usuarios finaliza.

Resultado esperado

Tras ingresar el usuario su cuenta y clave respectiva, debe aparecer automáticamente el menú principal para cualquier caso de los tipos de usuarios.

Evaluación de la prueba

Prueba satisfactoria.

Historia 2: Configuración del Sistema

Tabla IV. LXXVII Pruebas. Historia 2.

Fecha	Descripción	Autor
15/04/2013	Pruebas y Modificaciones	Carlos López / Noe Remache

Fuente: Autores

Descripción

El usuario Planificación debe autenticarse en el sistema. En el cual deberá escoger el rol CONFIGURACIONES DEL SISTEMA, en el mismo que se encuentra la opción Fechas de apertura y cierre de los procesos, asignación de techos presupuestarios de cada unidad, elección de la programación de los indicadores de los proyectos.

Condiciones de ejecución

El usuario Planificación debe constar en la base de datos del Sistema.

Entrada

✓ El usuario Planificación una vez que ingresa al sistema, escoge el rol CONFIGURACIONES DEL SISTEMA.

✓ Escoge el submenú FECHAS: se escoge el año para verificar si existen las

fechas de apertura y cierre del sistema, caso contrario realizar el respectivo ingreso (año y fechas según los procesos), ingreso de fechas de inicio y fin de cada uno de los procesos.

- ✓ Escoge el submenú FECHAS FACULTAD: se escoge el año y la dependencia para verificar si existen las fechas de apertura y cierre del sistema y su posterior modificación.
- ✓ Escoge el submenú TECHOS: se escoge el año para verificar si existen los techos de cada una de la dependencias, caso contrario se escoge el año, dependencia y se asignación de presupuesto.
- ✓ Escoge el submenú PROGRAMACIÓN DEL INDICADOR: se escoge entre mensual, bimensual, trimestral, cuatrimestral, semestral.

Resultado esperado

Tras ingresar el usuario Planificación realiza la configuración del sistema exitosamente insertando las fechas, los techos presupuestarios y usuarios.

Evaluación de la prueba

Prueba satisfactoria.

Historia 3: Gestión PEDI, Objetivo Estratégico, Política, Estrategia

Tabla IV. LXXVIII. Pruebas. Historia 3.

Fecha	Descripción	Autor
17/04/2013	Pruebas y Modificaciones	Carlos López / Noe Remache

Fuente: Autores

Descripción

El usuario Planificación debe autenticarse en el sistema. En el cual deberá escoger el rol ADMINISTRACION DEL PND Y PEDI, en el mismo que se encuentra la opción Administrar Plan Estratégico Institucional en el cual nos presenta las opciones: Ingresar Plan Estratégico Institucional, Objetivo Estratégico, Política, Estrategia.

Condiciones de ejecución

Los usuarios previamente deben estar registrados en la base de datos.

Entrada

- ✓ El usuario Planificación una vez que ingresa al sistema, escoge el rol ADMINISTRACION DEL PND Y PEDI
- ✓ Escoge el submenú ADMINISTRAR PLAN NACIONAL: donde se gestiona el Plan Nacional, Objetivo Nacional, Política Nacional y Meta Nacional.
- ✓ Escoge el submenú VER PLAN NACIONAL: donde se puede visualizar el Plan Nacional completamente.
- ✓ Escoge el submenú INGRESAR PLAN ESTRATÉGICO INSTITUCIONAL: donde se gestiona el Plan Estratégico, Objetivo Estratégico, Objetivo Operativo, Vinculación del Programa.

Resultado esperado

Tras ingresar el usuario Planificación realiza el ingreso del PEDI exitosamente para posteriormente escoger el objetivo institucional el cual se vinculará un proyecto.

Evaluación de la prueba

Prueba satisfactoria.

Historia 4: Gestión Función, Objetivo Operativo, Meta, Indicadores

Tabla IV. LXXIX. Pruebas. Historia 4.

Fecha	Descripción	Autor
19/04/2013	Pruebas y Modificaciones	Carlos López / Noe Remache

Fuente: Autores

Descripción

El usuario Planificación debe autenticarse en el sistema. En el cual deberá escoger el rol ADMINISTRACION DEL PND Y PEDI, en el mismo que se encuentra la opción Administrar Plan Estratégico Institucional en el cual nos presenta las opciones: Ingresar Función, Objetivo Operativo, Meta, Indicadores.

Condiciones de ejecución

Los usuarios previamente deben estar registrados en la base de datos.

Entrada

- ✓ Escoge el submenú INGRESAR DATOS ANEXOS DEL PLAN ESTRATÉGICO INSTITUCIONAL: donde se gestiona la política, función, estrategia, indicador nacional, meta.
- ✓ Escoge el submenú VER PLAN ESTRATÉGICO INSTITUCIONAL: donde se puede visualizar el Plan Estratégico Institucional.

Resultado esperado

Tras ingresar el usuario Planificación realiza el ingreso del PEDI exitosamente para posteriormente escoger el objetivo institucional el cual se vinculará un proyecto.

Evaluación de la prueba

Prueba satisfactoria.

Historia 5: Gestión Programa, Proyecto del PEDI

Tabla IV. LXXX. Pruebas. Historia 5.

Fecha	Descripción	Autor
23/04/2013	Pruebas y Modificaciones	Carlos López / Noe Remache

Fuente: Autores

Descripción

El usuario Planificación debe autenticarse en el sistema. En el cual deberá escoger el rol ADMINISTRACION DEL PND Y PEDI, en el mismo que se encuentra la opción Administrar Plan Estratégico Institucional en el cual nos presenta las opciones: Ingresar Programa. Para el Proyecto del PEDI deberá escoger el rol INGRESO DE PROYECTOS, el cual se detallará más adelante.

Condiciones de ejecución

Tras ingresar el usuario Planificación realiza el ingreso del PEDI exitosamente para posteriormente escoger el objetivo institucional el cual se vinculará un proyecto.

Entrada

- ✓ Escoge el submenú INGRESAR PROGRAMA: donde se gestiona el programa al cual será vinculado el proyecto.
- ✓ Para el proyecto PEDI se explicará en las historia 6-10 detalladamente.

Resultado esperado

Tras ingresar el usuario Planificación realiza el ingreso del Programa exitosamente para posteriormente poder escogerlo en la creación del proyecto.

Evaluación de la prueba

Prueba satisfactoria.

Historia 6: Gestión Proyectos

Tabla IV. LXXXI. Pruebas. Historia 6.

Fecha	Descripción	Autor
25/04/2013	Pruebas y Modificaciones	Carlos López / Noe Remache

Fuente: Autores

Descripción

Los usuarios una vez registrado en el sistema, en el cual deberá escoger el rol INGRESO DE PROYECTOS, se deberá escoger el Submenú Ingresar Proyecto con sus Datos Generales, Específicos y Vinculación con PEDI

Condiciones de ejecución

Tras ingresar los usuarios que realizan el ingreso de los proyectos deberán estar registrados en la base de datos.

Entrada

- ✓ Los usuarios una vez que ingresa al sistema, escoge el rol INGRESO DE PROYECTOS.
- ✓ Escoge el submenú INGRESAR PROYECTO: donde se gestiona el Proyecto ingresando los datos generales, específicos y su vinculación con el PEDI.

Resultado esperado

Tras ingresar los usuarios podrán gestionar sus proyectos de cada dependencia a fin de llegar a consolidar un POA para mejorar su funcionamiento.

Evaluación de la prueba

Prueba satisfactoria.

Historia 7: Gestión Meta

Tabla IV. LXXXII. Pruebas. Historia 7.

Fecha	Descripción	Autor
29/04/2013	Pruebas y Modificaciones	Carlos López / Noe Remache

Fuente: Autores

Descripción

Los usuarios una vez registrado en el sistema, en el cual deberá escoger el rol INGRESO DE META Y FINANCIAMIENTO, deberán ingresar la meta a cumplir del proyecto.

Condiciones de ejecución

Los usuarios deberán tener el proyecto previamente ingresado.

Entrada

✓ Escoge el submenú INGRESAR META Y FINANCIAMIENTO: donde se anexa la Meta del Proyecto escogiendo previamente.

Resultado esperado

Tras ingresar los usuarios podrán especificar las metas que desean cumplir con sus proyectos.

Evaluación de la prueba

Prueba satisfactoria.

Historia 8: Gestión Financiamiento

Tabla IV. LXXXIII. Pruebas. Historia 8.

Fecha	Descripción	Autor
01/05/2013	Pruebas y Modificaciones	Carlos López / Noe Remache

Fuente: Autores

Descripción

Los usuarios una vez registrados en el sistema, en el cual deberá escoger el rol INGRESO DE META Y FINANCIAMIENTO, deberán ingresar la meta a cumplir del proyecto.

Condiciones de ejecución

Los usuarios deberán tener ingresado el proyecto y las metas que desean realizar el financiamiento.

Entrada

- ✓ Escoge el submenú INGRESAR META Y FINANCIAMIENTO: donde se anexa el Financiamiento en la Meta del Proyecto escogiendo previamente.

Resultado esperado

Tras ingresar los usuarios podrán especificar las metas que desean cumplir con sus proyectos.

Evaluación de la prueba

Prueba satisfactoria.

Historia 9: Gestión Indicador

Tabla IV. LXXXIV. Pruebas. Historia 9.

Fecha	Descripción	Autor
03/05/2013	Pruebas y Modificaciones	Carlos López / Noe Remache

Fuente: Autores

Descripción

El usuario una vez registrado en el sistema, en el cual deberá escoger el rol INGRESO DE INDICADOR Y PROGRAMACION, deberá seleccionar la meta del proyecto para asignar los indicadores.

Condiciones de ejecución

Los usuarios deberán tener el proyecto previamente ingresado.

Entrada

✓ Escoge el submenú INGRESAR INDICADOR Y PROGRAMACION: donde se anexa el Indicador en la Meta del Proyecto escogiendo previamente.

Resultado esperado

Tras ingresar los usuarios podrán especificar los indicadores que poseen las metas para ayudarlas a cumplir con el objetivo del proyecto.

Evaluación de la prueba

Prueba satisfactoria.

Historia 10: Gestión Programación

Tabla IV. LXXXV. Pruebas. Historia 10.

Fecha	Descripción	Autor
07/05/2013	Pruebas y Modificaciones	Carlos López / Noe Remache

Fuente: Autores

Descripción

El usuario una vez registrado en el sistema, en el cual deberá escoger el rol INGRESO DE INDICADOR Y PROGRAMACION, deberá seleccionar el indicador a programar para luego ingresar la programación de acuerdo a lo configurado en la Historia 2.

Condiciones de ejecución

Los usuarios deberán tener previamente ingresados los indicadores a programar.

Entrada

✓ Escoge el submenú INGRESAR INDICADOR Y PROGRAMACION: donde se

anexa la Programación en el Indicador escogiendo previamente.

Resultado esperado

Los usuarios podrán programar cada uno de los indicadores de un proyecto para asegurar la mayor eficiencia y eficacia del mismo.

Evaluación de la prueba

Prueba satisfactoria.

Historia 11: Control de Priorización de Proyectos

Tabla IV. LXXXVI. Pruebas. Historia 11.

Fecha	Descripción	Autor
09/05/2013	Pruebas y Modificaciones	Carlos López / Noe Remache

Fuente: Autores

Descripción

El usuario debe autenticarse en el sistema. En el cual deberá escoger el rol PRIORIZAR PROYECTOS, en el mismo q se encuentra la opción Priorizar proyectos en la cual deberá escoger la Facultad / Unidad y luego escoger el proyecto a priorizar.

Condiciones de ejecución

El usuario Rector deberá constar registrado en la base de datos y las dependencias deberán haber ingresado sus proyectos para priorizar.

Entrada

- ✓ El usuario Rector una vez que ingresa al sistema, escoge el rol PRIORIZAR PROYECTOS.
- ✓ Escoge el submenú PRIORIZAR PROYECTOS: donde se selecciona la dependencia para visualizar los proyectos que posee, luego se procede a priorizar de acuerdo a la ponderación previamente establecida por el

departamento de Planificación.

Resultado esperado

Tras ingresar el usuario Rector podrá priorizar y semaforizar los proyectos de cada una de las dependencias a fin de establecer de mayor a menor prioridad los proyectos para su ejecución.

Evaluación de la prueba

Prueba satisfactoria.

Historia 12: Determinación de Proyectos

Tabla IV. LXXXVII. Pruebas. Historia 12.

Fecha	Descripción	Autor
13/05/2013	Pruebas y Modificaciones	Carlos López / Noe Remache

Fuente: Autores

Descripción

El usuario debe autenticarse en el sistema. En el cual deberá escoger el rol DETERMINACION DE PROYECTOS, en la misma q se encuentra la opción Clasificar Proyecto en la cual deberá escoger el proyecto para determinar el tipo de proyecto al cual será contemplado.

Condiciones de ejecución

El usuario Planificación debe constar en la base de datos de la aplicación y los proyectos deberán estar priorizados.

Entrada

- ✓ El usuario Planificación una vez que ingresa al sistema, escoge el rol DETERMINACIÓN DE PROYECTOS.
- ✓ Escoge el submenú CLASIFICAR PROYECTO: donde se selecciona la dependencia para visualizar los proyectos que posee, luego se procede a

determinar para que plan (operativo, inversión) se asignará el proyecto.

Resultado esperado

Tras ingresar el usuario Planificación puede visualizar los proyectos que ya han sido determinados o clasificados.

Evaluación de la prueba

Prueba satisfactoria.

Historia 13: Formulación POA y PAI

Tabla IV. LXXXVIII. Pruebas. Historia 13.

Fecha	Descripción	Autor
15/05/2013	Pruebas y Modificaciones	Carlos López / Noe Remache

Fuente: Autores

Descripción

Los usuarios deben autenticarse en el sistema. En el cual deberá escoger el rol INGRESO DE POA Y PAI, en el mismo q se encuentra la opción Crear PAI en la cual deberá escoger el proyecto para anexar y luego ingresar la información restante con forme al formulario 5 de SEMPLADES.

Condiciones de ejecución

Los usuarios deberán tener determinados sus proyectos de POA y PAI para poder anexarlos.

Entrada

- ✓ Los usuarios una vez que ingresa al sistema, escoge el rol INGRESO POA Y PAI.
- ✓ Escoge el submenú CREAR POA: donde se selecciona los proyectos que constan en el PEDI y los proyectos creados, priorizados y determinados,

también se ingresará un código interno que manejará el proyecto en un futuro.

- ✓ Escoge el submenú CREAR PAI: donde se selecciona los proyectos creados, priorizados y determinados.

Resultado esperado

Tras los usuarios podrán visualizar su POA y PAI que han creado para su posterior aprobación.

Evaluación de la prueba

Prueba satisfactoria.

Historia 14: Control de Reportes

Tabla IV. LXXXIX. Pruebas. Historia 14

Fecha	Descripción	Autor
17/05/2013	Pruebas y Modificaciones	Carlos López / Noe Remache

Fuente: Autores

Descripción

Los usuarios deben autenticarse en el sistema con sus cuentas respectivamente podrán visualizar los reportes correspondientes a Plan Nacional, Plan Estratégico, Proyectos, Priorización, POA, PAI.

Condiciones de ejecución

Los usuarios deberán tener ingresados los datos previamente antes de ejecutar los reportes de los diferentes módulos.

Entrada

- ✓ Los usuarios una vez registrados en el sistema, podrán visualizar los reportes sólo en los

Resultado esperado

Tras ingresar los usuarios podrán acceder a los datos de los reportes exitosamente en la base de datos.

Evaluación de la prueba

Prueba satisfactoria.

CONCLUSIONES

- ✓ El análisis permitió destacar las ventajas de cada método de reutilización de código en Java, destacándose el método de reutilización *"Herencia"* ya que su utilización en el desarrollo del Sistema Informático de la Unidad Técnica de Planificación de la ESPOCH permitió obtener una disminución en su tiempo de desarrollo, esfuerzo, dificultad y líneas de código.
- ✓ Al utilizar el método de reutilización de código *"herencia"*, el tiempo de desarrollo de una aplicación informática disminuye en un 9% de manera apreciable respecto al tiempo total de desarrollo. Este resultado se obtuvo de la comparación con el prototipo sin reutilización de código, mientras que el método de reutilización de código *"composición"* el tiempo de desarrollo se incrementa un 3%.
- ✓ El esfuerzo requerido para desarrollar una aplicación informática disminuye al utilizar el método de reutilización de código *"herencia"* en un 9,01% respecto al esfuerzo requerido por el prototipo *"sin reutilización de código"*, mientras que con el método de reutilización de código *"composición"* el esfuerzo se incrementa un 3%.
- ✓ El número de líneas de código utilizadas para desarrollar los prototipos disminuye en un 8,04% al utilizar el método de reutilización de código *"herencia"*, y al utilizar el método de reutilización de código *"composición"* el número de líneas de código se incrementa en un 3,11%.
- ✓ La dificultad al momento de desarrollar los prototipos es pequeña al utilizar el método de reutilización de código *"herencia"* y es media al utilizar el método de reutilización de código *"composición"*.

- ✓ El porcentaje de uso del CPU en el servidor disminuye un 25,7% al utilizar el método de reutilización de código *“herencia”*, y al utilizar el método de reutilización de código *“composición”* solo disminuye un 0,18%.
- ✓ El porcentaje de uso de la memoria RAM del servidor se incrementa un 51,12% al utilizar el método de reutilización de código *“herencia”* y al utilizar el método de reutilización de código *“composición”* disminuye un 4,14%.
- ✓ La herencia y la composición ofrecen ventajas que sirven de ayuda para el desarrollo de las aplicaciones, pero sus desventajas es el uso de más objetos y en el aumento de complejidad de ciertas partes del código.
- ✓ Se desarrolló un sistema web para la Unidad Técnica de Planificación de la ESPOCH que le permitirá automatizar los procesos de:
  - Gestión de Proyectos del Plan Operativo Anual
  - Gestión de Proyectos del Plan de Inversión Pública.

RECOMENDACIONES

- ✓ Se recomienda utilizar el método de reutilización de código Herencia para implementaciones de futuros sistemas web.
- ✓ El sistema web para la Unidad Técnica de Planificación de la ESPOCH, debe ser mejorado con la implementación de un nuevo módulo en el sistema que permita la reformulación del Plan Operativo Anual y también del Plan de Inversión Pública.
- ✓ A partir del presente estudio se puede realizar uno nuevo que compare los métodos de reutilización de código, en otras plataformas de desarrollo y con otras tecnologías.
- ✓ Se recomienda realizar un análisis detallado del método de reutilización de código *"Delegación"*, el cual se basa en la creación de un objeto que contiene referencias a otros objetos y les delega comportamiento a cada uno de ellos.

RESUMEN

“Análisis de métodos de reutilización de código java para optimizar el desarrollo de aplicaciones web. Caso Práctico: Unidad Técnica de Planificación de la Escuela Superior Politécnica de Chimborazo”

El Método Inductivo se usó para el análisis e implementación del sistema permitiendo investigar los métodos de reutilización de código java necesarios para la implementación del mismo. El Método Deductivo fue útil para tratar temas generales analizados de tal forma que permitió tener una percepción global del Sistema de Gestión de Planes Operativos e Inversión (SIGPOI).

Los materiales usados para el desarrollo del (SIGPOI) fueron: Sistema Operativo Windows7, motor de Base de Datos Postgres 9.1, IDE Netbeans 7.3, dos laptops, un servidor conjunto facilitado por la FIE integrando el Servidor de Base Datos Postgres 9.1 y Servidor Web Glassfish Server Open Source Edition 3.1.1 con Windows server 2008 Enterprise Edition Sp2.

Se realizó el estudio, análisis, comparación de los métodos de reutilización de código herencia y composición; y su respectiva interpretación de valores cuantitativos obteniéndose los siguientes puntajes: Herencia disminuye el 9% en tiempo de desarrollo, 9,01% esfuerzo, 8,04% líneas de código, 0,18% uso del CPU e incrementa el 4,14% uso de la memoria RAM, mientras que la Composición incrementa el 3% en tiempo de desarrollo, 3% esfuerzo, 3,11% líneas de código, 25,7% uso del CPU y 51,12% uso de la memoria RAM. Se determinó así que el método de reutilización de código más adecuada es la

Herencia porque mejora la productividad en el desarrollo de aplicaciones web, por los parámetros mencionados anteriormente.

SUMMARY

“Java code reuse methods analysis to optimize the development of web applications. Case Study: Technical Planning Unit in the Polytechnic of Chimborazo”.

The inductive method was used for the system analysis and implementation to research java code reuse methods needed for its implementation. The deductive method was useful to address general issues analyzed in a what that permitted an overall perception about the Operating and Investment Plans Management System (SIGPOI).

The used materials for the (SIGPOI) development were: Windows 7, database engine Postgres 9.1, NetBeans IDE 7.3, two laptops, a set server provided by the FIE integrating the Database Server Postgres 9.1 and Glassfish Server Open Source Edition 3.1.1 with Windows Ser)08 Enterprise Edition Sp2.

The study, analysis, composition and legacy code reuse comparison methods were conducted; and their respective interpretation of quantitative values obtained the following scores: transmission decreases 9% in time development, 9.01% effort, 8.04% code lines, 0.18% CPU use and increases 4.14% RAM memory usage while composition increases 3% time development, 3% effort, 3.11% code lines, 25.7% CPU usage and 51.12% RAM memory use. Thus, it was determined that the most appropriate code reuse method is Transmission because it improves productivity in the web applications development for the above mentioned parameters.

It is concluded that in developing the SIGPOI, worded out at the Technical Planning Unit (UTP), the information will have a proper management, administration and control.

It is recommended to apply the SIGPOI technical and user manual for the system correct use and thereby the Technical Planning Unit will keep an excellent performance at the institutional level.

GLOSARIO

IDE: Llamado también entorno de desarrollo integrado, es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios.

JAVA SERVER PAGES: Es una tecnología que ayuda a los desarrolladores de software a crear páginas web dinámicas basadas en HTML, XML entre otros tipos de documentos. JSP es similar a PHP pero usa el lenguaje de programación Java.

Java: Lenguaje de Programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90.

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

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

Servidor de Base Datos: Es un programa que provee servicios de base de datos a otros programas u otras computadoras, como es definido por el modelo cliente-servidor.

Servidor Web: También conocido como servidor HTTP, es un programa informático que procesa una aplicación del lado del servidor realizando una

conexión con el cliente generando o cediendo una respuesta en cualquier lenguaje o Aplicación del lado del cliente. El código recibido por el cliente suele ser compilado y ejecutado por un navegador web. Para la transmisión de todos estos datos suele utilizarse algún protocolo.

Motor de Base de Datos: Es un programa o aplicación que permite administrar una base de datos, existen muchos actualmente, los hay libres y de pago.

BIBLIOGRAFÍA

1. REUTILIZACIÓN DE CÓDIGO

http://www.alegsa.com.ar/Dic/reutilizacion%20de%20codigo.php

http://www.emagister.com/curso-programacion-avanzada/reutilizacion-codigo

http://www.sc.ehu.es/sbweb/fisica/cursoJava/fundamentos/clases1/clases.htm

http://wwwdi.ujaen.es/asignaturas/progav/progav-tema2.pdf

http://www.slideshare.net/senaticscesar/programacion-orientada-a-objetos-4540712

http://www.desarrolloweb.com/articulos/499.php

http://www.alegsa.com.ar/Dic/programacion%20generica.php

http://www.zator.com/Cpp/E4_12.htm

http://www.eumed.net/libros/2006c/216/1j.htm

http://www.slideshare.net/ajmae28/programacin-dinmica-3742041

[2012/09/26]

2. REUTILIZACIÓN DE CÓDIGO JAVA

http://www.sicuma.uma.es/sicuma/Formacion/documentacion/Total-Java.pdf

http://java-spain.com/en/node/5

http://javacafesv.blogspot.com/2009/05/polimorfismo-en-java.html

http://danubuntu.wordpress.com/2008/07/30/conceptos-sobre-polimorfismo-y-programacion-orientada-a-objetos/

http://dis.um.es/~lopezquesada/documentos/IES_1011/DFSI/curso/UT6/java

/cap9/index.html

http://www.programacion.com/articulo/tutorjava_nivel_basico_97/21

http://artemisa.unicauca.edu.co/~dparedes/java/objetos/subclass.html

http://www.labcom.upcomillas.es/poo/itig/apuntes/Java03.pdf

[2012/10/01]

3. COMPOSICIÓN

http://www.sc.ehu.es/sbweb/fisica/cursoJava/fundamentos/clases1/composicion.htm

http://www.slideshare.net/techmi/curso-java-inicial-5-relaciones-entre-objetos

http://www.cristalab.com/tutoriales/programacion-orientada-a-objetos-asociacion-vs-composicion-c89337l/

http://docs.kde.org/stable/es/kdesdk/umbrello/uml-elements.html

https://sites.google.com/site/tutoriasdeingenieria/programacion-o-o/uml-relaciones-entre-clases-poo-java

http://www.dsi.fceia.unr.edu.ar/downloads/informatica/info_II/mapeoUMLJava3.doc

http://arco.esi.uclm.es/~david.villa/pensar_en_C++/vol1/ch14s05s02.html

http://www.fdi.ucm.es/profesor/jpavon/poo/01ConceptosOO.pdf

[2012/10/15]

4. HERENCIA

http://www.slideshare.net/mejiaff/java-orientado-a-objetos

http://sophia.javeriana.edu.co/~acarrillo/POO/Material/CursoPOOHerencial.pdf

http://www.emagister.com/curso-java/herencia-java-1

http://www.fdi.ucm.es/profesor/gmendez/docs/prog0607/Tema4-Herencia.pdf

http://dis.um.es/~lopezquesada/documentos/IES_1011/DFSI/curso/UT6/java/cap9/index.html

http://www.proyectoleonardo.net/index.php/leonardo/article/download/36/42

http://javacafesv.blogspot.com/2009/03/herencia-multiple-en-java-conveniente-o.html

http://www.fdi.ucm.es/profesor/gmendez/docs/prog0607/Tema4-Herencia.pdf

http://www.fdi.ucm.es/profesor/gmendez/docs/prog0607/Tema4-Herencia.pdf

http://profesores.fi-b.unam.mx/carlos/java/java_basico4_8.html

[2012/11/05]

5. PLAN OPERATIVO ANUAL (POA)

http://definicion.de/plan-operativo/

http://www2.uacj.mx/transparencia/Plan/documentos/9_III_Documentos_Programaci%C3%B3n/1_Que_es_el_POA.pdf

http://www.icf.gob.hn/files/tramites/Servicios%20Prestados%20Plan%20Operativo%20Anual.pdf

http://es.scribd.com/doc/68443440/Plan-Operativo-Anual-Concepto-y-Ejemplos

[2013/11/19]

6. PLAN ANUAL DE INVERSIÓN (PAI)

http://www.quindio.gov.co/home/docs/items/item_100/P-PLA-04PlanOperativoanualdeInversion.pdf

http://www.planificacion.gob.ec/wp-content/uploads/downloads/2012/08/Inversi%C3%B3n-P%C3%BAblica.pdf

http://www.antioquia.gov.co/Fantioquia-1/Forganismos/Fplaneacion/Fdescargas/Ffinanzas/F08planoperativoanualdeinversiones.doc&ei=IwicUOWGJYP69QSRjoC4Dw&usg=AFQjCNEWmVvwrFy3CG-VwTzfNZ0BCRZAUg

[2013/12/17]

ANEXOS

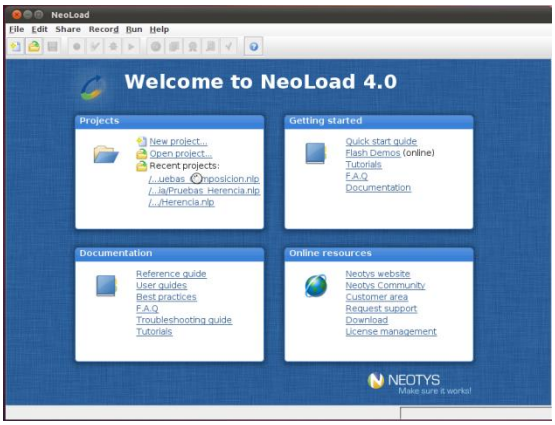
ANEXO 1

USO DE NEOLOAD

Splash herramienta NeoLoad v4.0.4

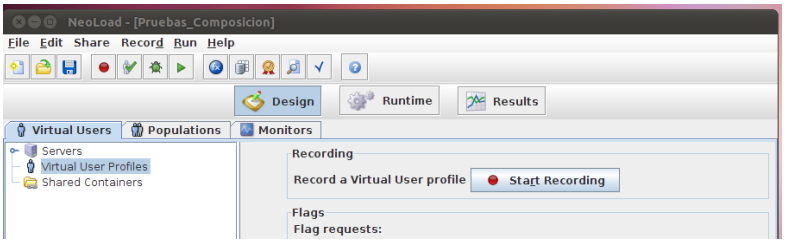
Para realizar la medida del desempeño de una aplicación web debemos como primer paso realizar la captura del sistema web que vamos a medir, al referirnos a capturar el sistema nos referimos a grabar todo el proceso que hará nuestro sistema, en este caso la herramienta grabará los pasos que realizamos desde el ingreso al Sistema hasta cuando finalicemos la sesión de trabajo, en este caso para la medición de los prototipos se especificara los pasos de uno de ellos ya que el procedimiento es similar para ambos.

1.- Ingresamos a la herramienta y lo primero que observaremos es el menú de Proyectos, seleccionamos **New Project** y esperamos a que la herramienta cargue los componentes necesarios para iniciar el trabajo:



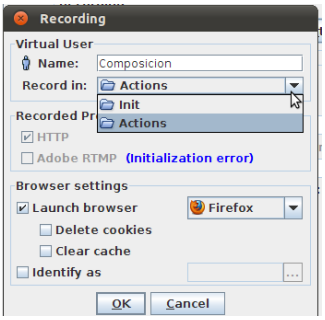
Opciones de Herramienta NeoLoad

Una vez cargada la herramienta, seleccionamos la opción **StartRecording:**



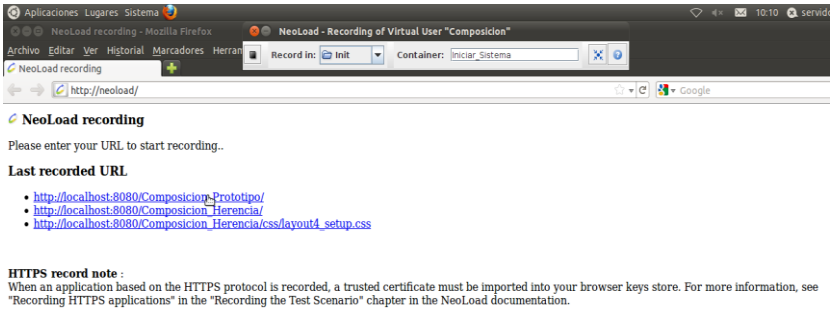
Botón de inicio de Grabación.

Aparecerá una ventana donde especificaremos el nombre de nuestro entorno virtual y en **Record in** seleccionaremos **Init** que se refiere a la parte inicial del sistema que es por donde se empieza la captura, este tipo de diferenciaciones la herramienta lo utiliza para clasificar cada parte del sistema, así por ejemplo la gestión de los datos del usuario y del proyecto serán almacenados en **Actions**. Presionamos **OK.**



Ventana de Propiedades de la grabación.

Al finalizar la Configuración se abrirá nuestro navegador predeterminado, el cual nos indicará todas las direcciones que hayamos abierto recientemente y pulsamos sobre el enlace que deseemos grabar, en la parte superior central de la página observaremos que la herramienta se encuentra grabando desde el inicio, en la casilla **Container** se indicará un nombre que nos servirá para poder identificar que parte del sistema se encuentra en funcionamiento.

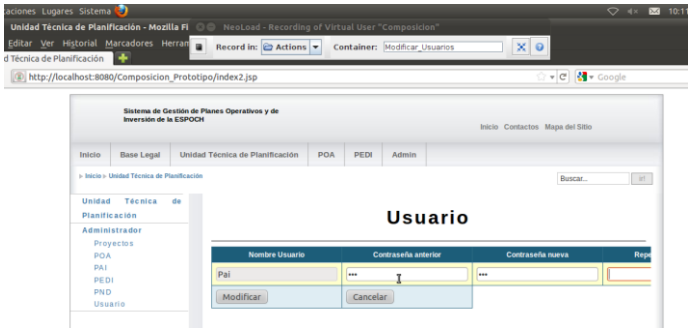


Selección de enlace a grabar.

De igual manera al Autenticarnos en el sistema indicaremos en Container el nombre del paso que hacemos.

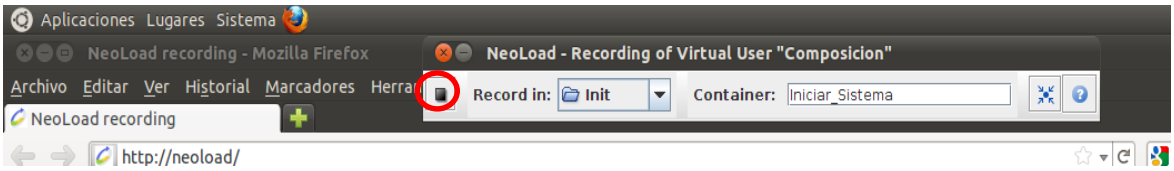
Autenticación del Sistema.

Al momento de realizar la gestión de usuarios y de proyectos debemos especificar que parte estamos grabando, es decir en esta parte se procederá a grabar el ingreso, modificación y eliminación de datos por lo que en **Record in** vamos a seleccionar **Actions**y en **Container** ingresaremos un nombre que identifique el proceso a realizar. De igual forma especificaremos el proceso referente a proyectos.



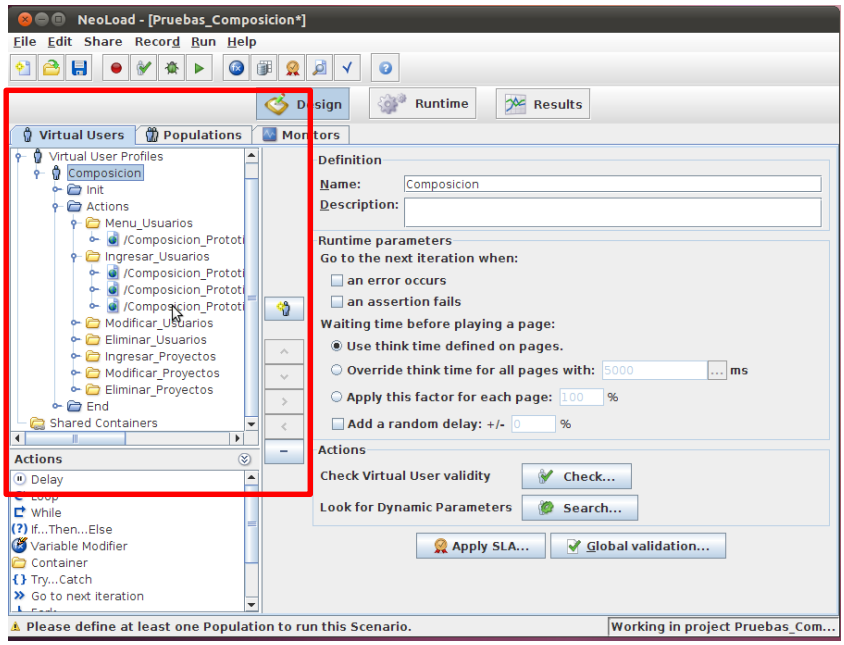
Grabación de Gestión de Usuarios

Una vez hayamos terminado de grabar los procesos que deseemos procederemos a cerrar la grabación haciendo clic en el botón **Stop** que se encuentra en la ventana superior central.



Botón de alto a grabación del sistema.

Una vez detenida la grabación esperaremos a que termine la creación del entorno virtual que se empieza a ejecutar una vez hayamos cerrado la grabación, al finalizar este paso se nos mostrará en la ventana que aparece a continuación los pasos que grabamos separados de acuerdo a la posición como hayamos decidido ubicarlos, se puede apreciar que se verán las páginas del sistema por las que navegamos o que tomaron parte durante la grabación.

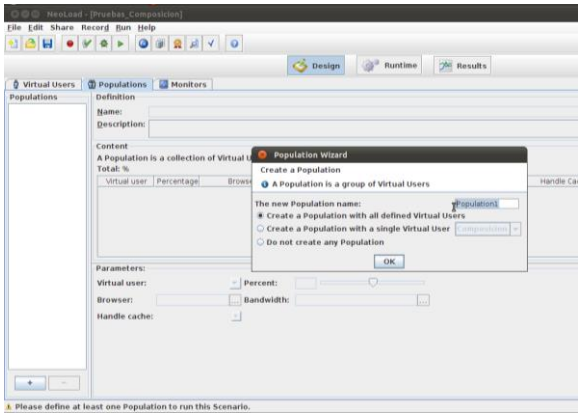


Pasos capturados del Sistema.

ANEXO 2

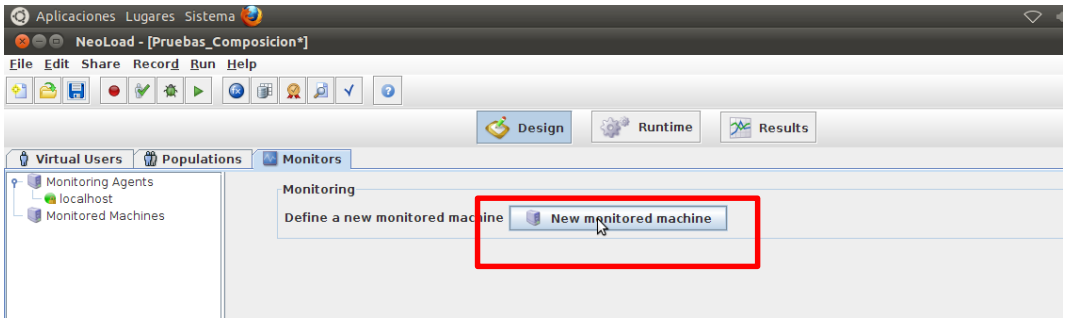
CONFIGURACION DE LA MEDIDA DE DESEMPEÑO

Ahora que ya hemos capturado los pasos de nuestro sistema para la medición del desempeño, procedemos a la creación de población, es decir el número de usuarios virtuales para utilizar el sistema durante la ejecución de la simulación de uso. Para ellos seleccionamos la pestaña **Populations,** hacemos click en el botón **+** ubicado en la parte inferior izquierda de la ventana, aparecerá una ventana donde indicaremos un nombre para nuestros usuarios virtuales, dejamos marcada la primera opción de y pulsamos **OK**, por defecto la población que se crea es de 10 usuarios aunque podemos incrementar esta cantidad.



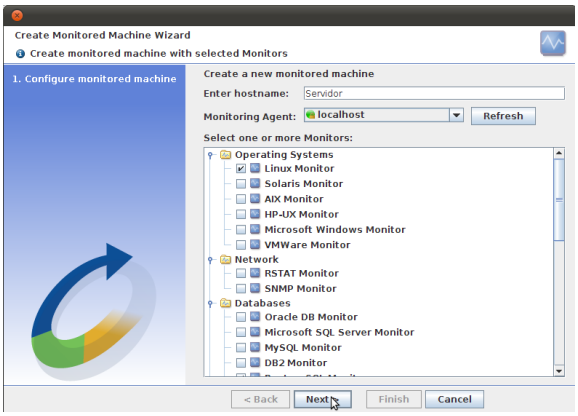
Ingreso de población para la medición del desempeño.

Ahora vamos a configurar los monitores de medida para el desempeño, estos monitores nos permiten saber el desempeño de la aplicación en lo que respecta a uso de CPU, memoria, acceso a bases de datos, ancho de banda utilizado y otras opciones, para desplegarlo nos vamos a la pestaña llamada **Monitors** y pulsamos la opción **New Monitor Machine.**



Ingreso de un nuevo Monitor.

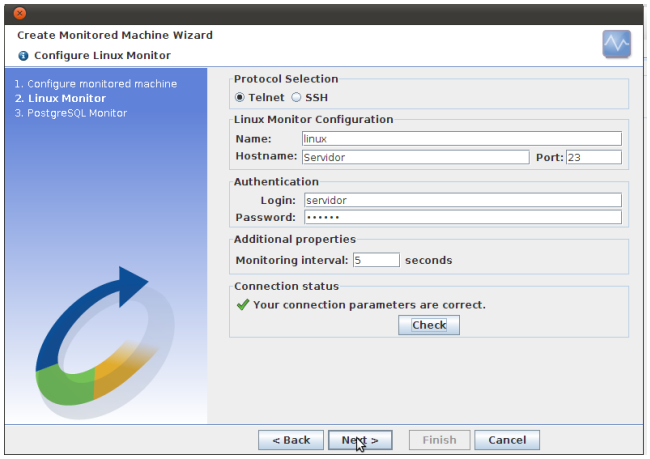
Aparece una ventana en la cual podemos indicar los monitores que vamos a utilizar, podemos escoger entre monitores para el Sistema Operativo, Red y Bases de Datos, seleccionamos los que necesitamos y pulsamos **Next**.



Monitores para medición de Desempeño.

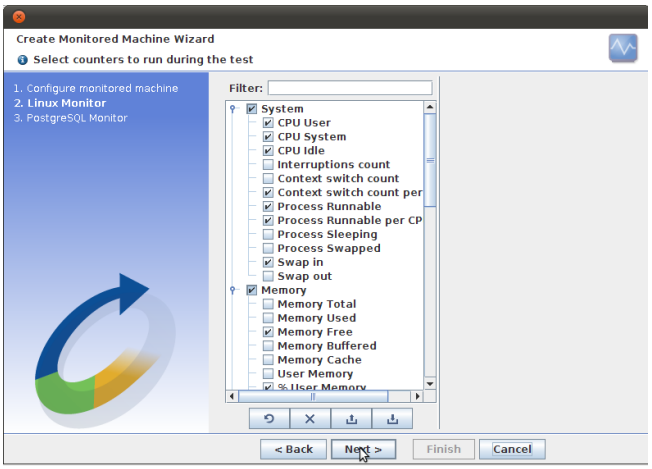
Ahora una vez seleccionado los monitores para medición, procedemos a configurar los monitores que necesitamos, empezamos con el Monitor del Sistema Operativo, en este caso se utiliza Ubuntu 12.04 para lo cual se escogió un monitor referente a Linux, esta herramienta utiliza para el monitoreo el servicio de Telnet, ingresamos los datos necesarios para la conexión y comprobamos que la conexión sea correcta para ello basta con hacer click en **Check** y nos mostrará un

aviso que nos indicará si la conexión es válida o debemos corregir algo, una vez configurado lo que necesitamos pulsamos **Next**.



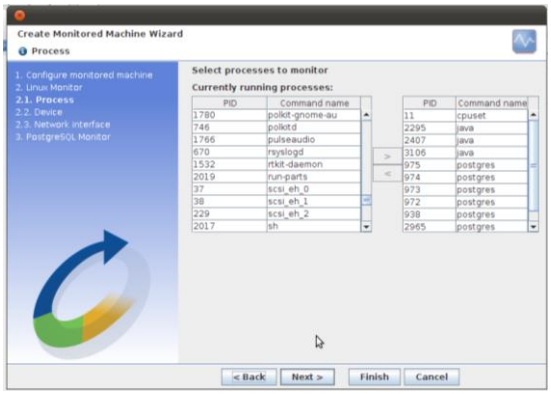
Configuración Monitor del Sistema Operativo.

Ahora indicaremos que contadores queremos que sean analizadas en lo referente a utilización del CPU, memoria RAM, etc.; Marcamos los filtros que necesitamos y pulsamos **Next**.



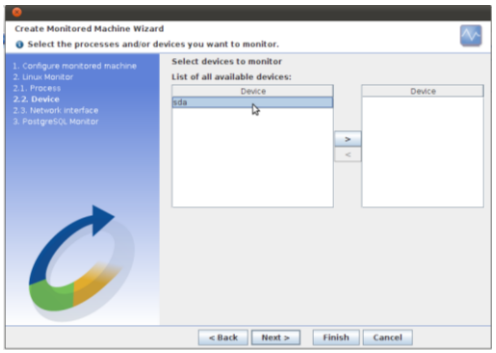
Contadores del Sistema Operativo.

Ahora seleccionamos los procesos que queremos sean medidos, estos procesos son los que se medirán durante la simulación, una vez seleccionados pulsamos **Next**.



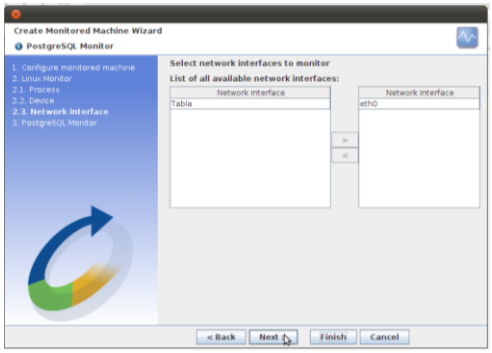
Procesos del Sistema a medir.

Para la medición de porcentaje de uso del disco duro, debemos indicar el disco donde nuestro sistema se encuentra alojado, una vez seleccionado pulsamos **Next**.



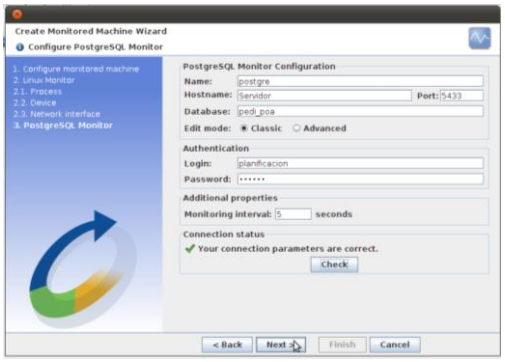
Selección de Disco.

Ahora se procede a indicar la interfaz de red que se utilizará durante las pruebas de desempeño del sistema, seleccionamos y pulsamos **Next**.



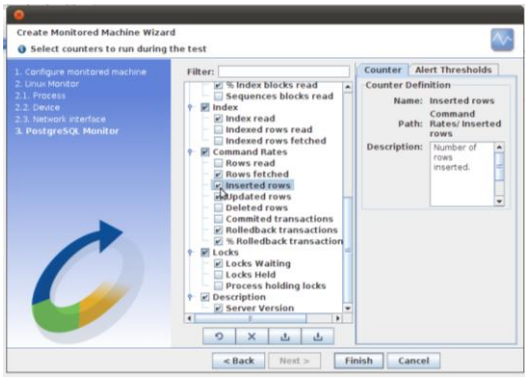
Selección de interfaz de Red.

Una vez terminado la selección de Indicadores para el Sistema Operativo, procedemos a seleccionar los indicadores para las pruebas de la Base de Datos, se utiliza Postgres ya que es el motor de base de datos usado para la elaboración del sistema. De igual manera se procede a configurar la conexión con la base datos, se puede comprobar si la conexión es correcta haciendo click en **Check**, una vez finalizado la configuración pulsamos **Next**.



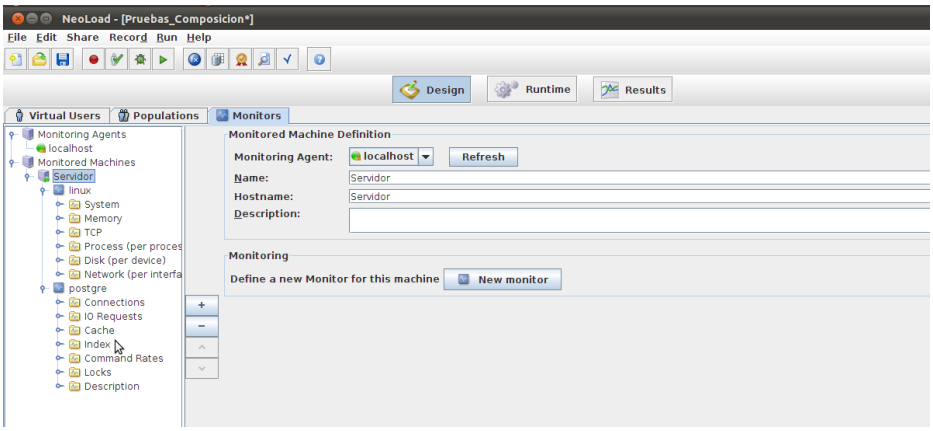
Configuración del Monitor de Bases de Datos.

Una vez configurada la conexión con el motor de Base de Datos, se muestra los diferentes indicadores que se pueden medir durante la simulación, seleccionamos aquellos que creamos relevantes para la investigación y pulsamos **Finish**.



Indicadores medidor de Bases de Datos

Al finalizar la configuración de los Monitores, volvemos a la ventana de Diseño de la Prueba y podremos apreciar que los indicadores que hemos seleccionado para la prueba aparecen en el menú de la izquierda.

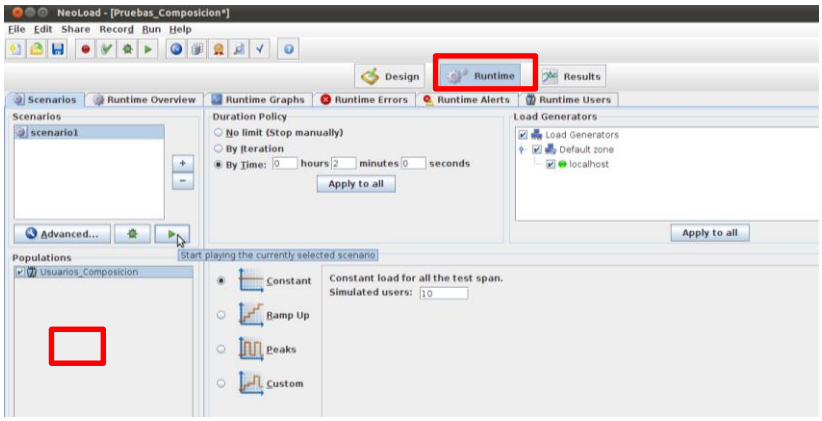


Indicadores de los Monitores del Sistema.

ANEXO 3

CONFIGURACION DE EJECUCION DE PRUEBAS

Una vez terminamos con el diseño de la Prueba, nos moveremos al botón **Runtime**, al presionarlo se observará el escenario, la población de usuarios que por defecto será de 10 para la ejecución de las Pruebas, para ello pulsamos el botón **Run** ubicado en la parte izquierda de la ventana

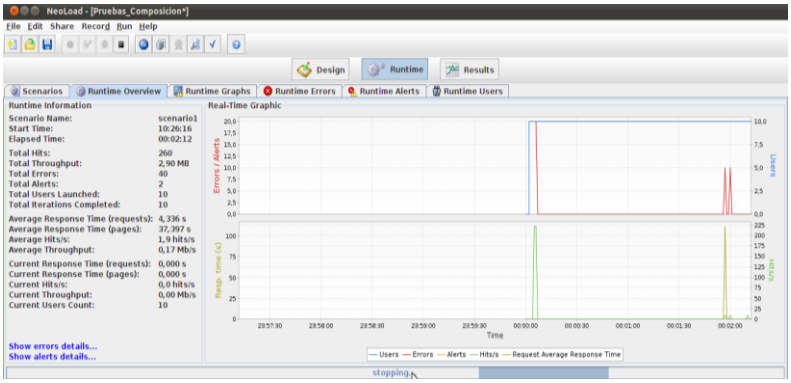


Runtime de las pruebas.

Al pulsar **Run** se muestra una ventana emergente que nos pide ingresar un nombre del Reporte de Resultados que por defecto será la fecha y una descripción de la misma, una vez ingresado estos datos pulsamos **OK**.

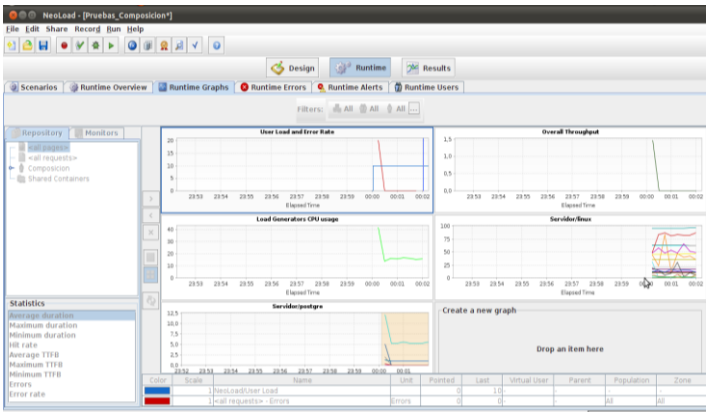
Ingreso de Nombre y Descripción de la Prueba.

A continuación podremos ver como se inicia las pruebas de desempeño de la aplicación, esta herramienta muestra resultados en tiempo real.



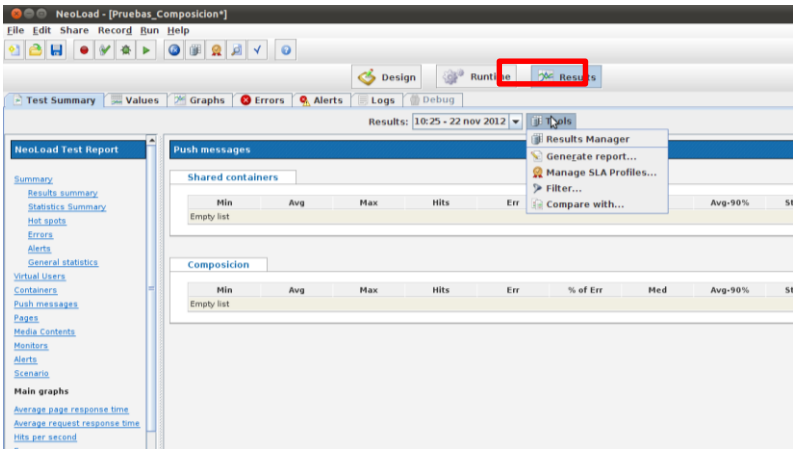
Ejecución de Pruebas de Desempeño.

De igual manera se al finalizar la prueba se puede generar gráficos que permiten apreciar de manera más correcta los resultados de la simulación.



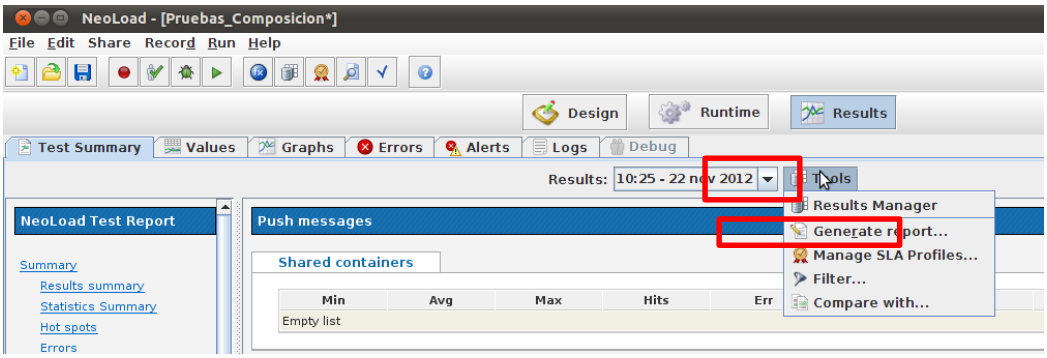
Gráficas generadas tras la prueba.

Una vez terminada la Prueba, al pulsar en el Botón **Results**, la herramienta genera de manera automática un informe detallado acerca de los resultados obtenidos durante la simulación realizada.



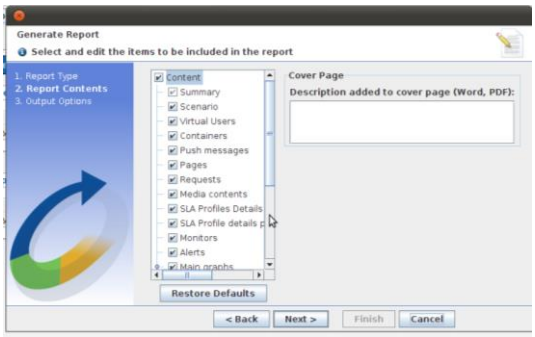
Informe de Resultados de la Prueba.

Se puede generar además un informe en formato PDF en caso de que fuera necesario, para ello pulsamos en el botón llamado **Tools** y se mostrará varias opciones que se pueden tratar con el resultado obtenido, hacemos clic en **Generate Report……** .



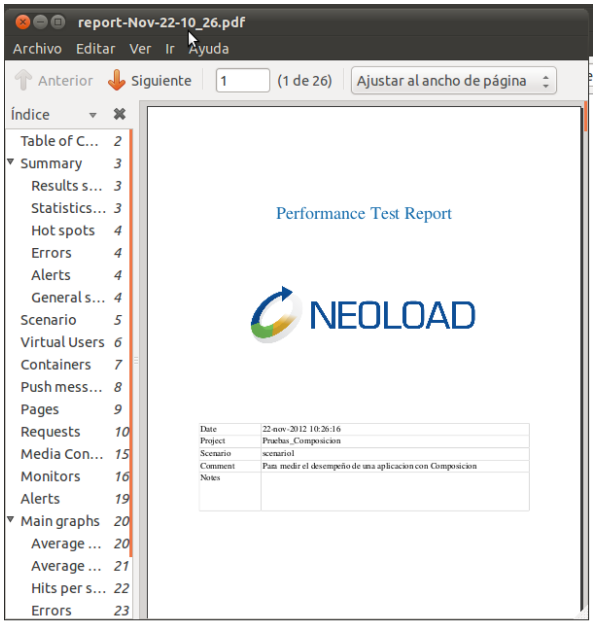
Opciones de Generación del reporte de Resultados.

Una vez elegido la Generación del reporte, se muestra una pantalla en la que se indicará que partes de los resultados deseamos que se muestren en el informe, pulsamos **Next**.



Opciones del Reporte.

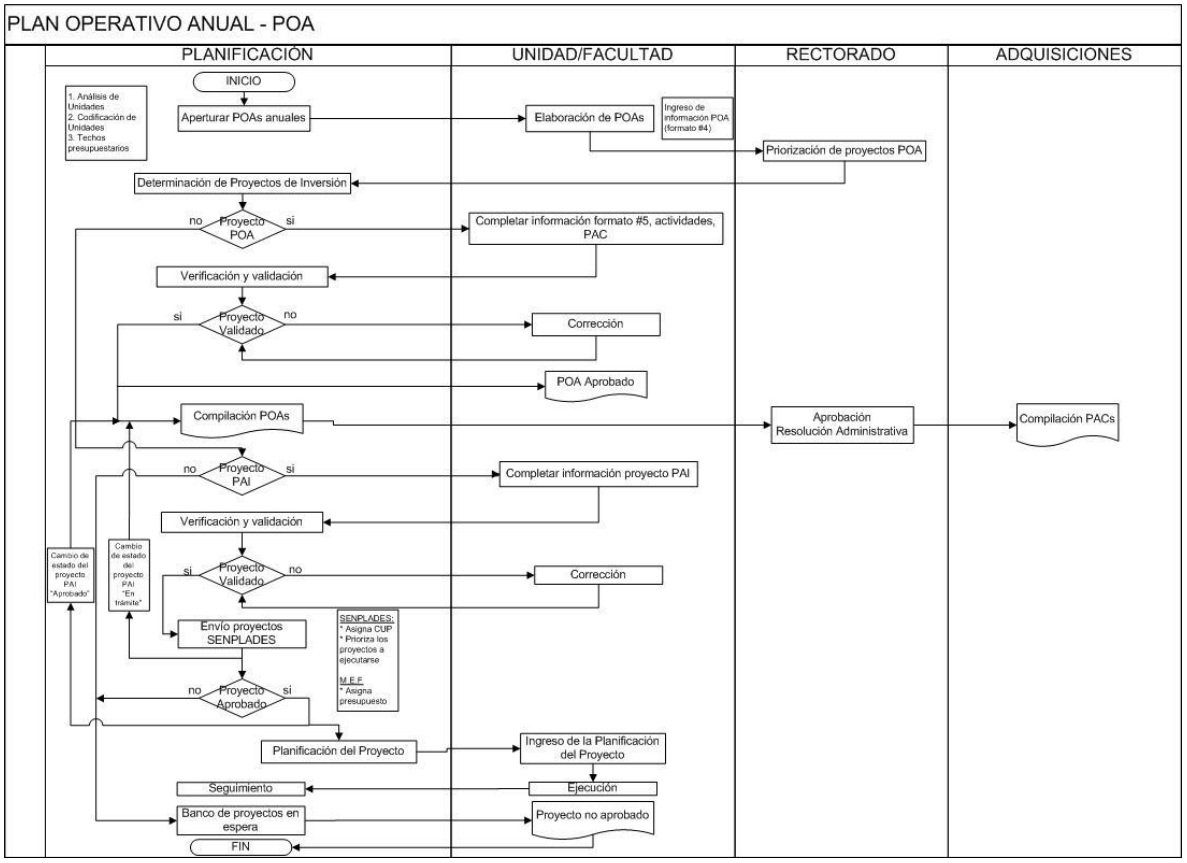
El informe se generará de acuerdo a los componentes que deseamos ver en el mismo, por defecto se crea un informe en formato PDF.



Reporte Generado.

ANEXO 4

DIAGRAMA DE PROCESOS DE LA UNIDAD TÉCNICA DE PLANIFICACIÓN



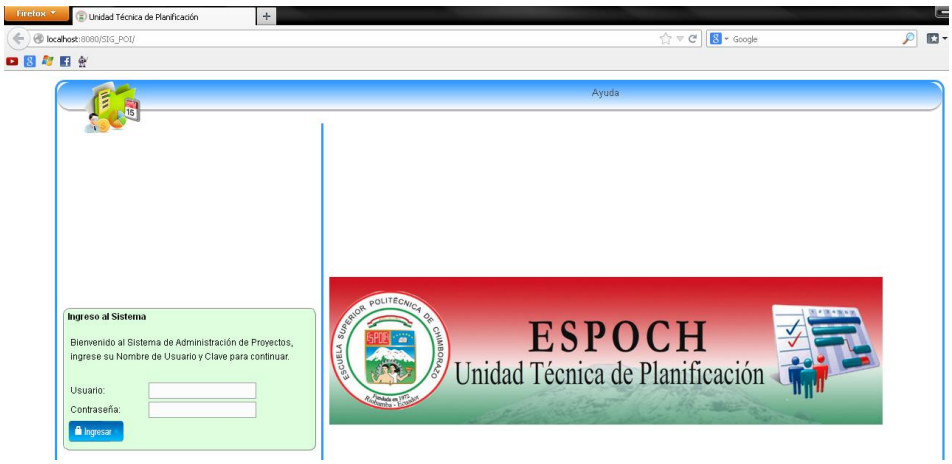
ANEXO 5

MANUAL DE USUARIO

INTRODUCCIÓN

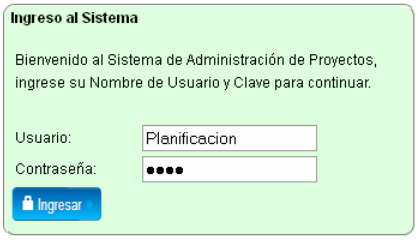
El Sistema de Gestión de Planes Operativos e Inversión (SIG POI) es un sistema que gestiona proyectos, genera Planes Operativos y Planes de Inversión y cuenta con lo siguiente:

Pantalla de Inicio: la cual aparecerá al momento de escribir la dirección 172.30.34.226/SIG_POI



INICIAR SESIÓN

Luego se deberá iniciar sesión mediante un nombre de usuario y una contraseña



CERRAR SESIÓN

Para cerrar la sesión el usuario debe dar clic en la figura que se encuentra en la parte derecha del nombre del usuario como en la figura siguiente.

Si el usuario no es válido o sea no se pudo autentificar no podrá continuar, es decir, se mostrará la pantalla de inicio. Caso contrario aparecerá las diferentes opciones según el tipo de usuario.

TIPOS DE USUARIOS

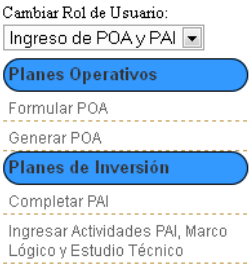
Tenemos usuario de tipo **ADMINISTRADOR, RECTOR, DECANO** los cuales poseen las siguientes funciones:

TIPO DE USUARIO	ALCANCE
DECANO	Tendrá la generación de POA y PAI.
RECTOR	Tendrá el alcance del tipo de usuario DECANO más una opción para PRIORIZACIÓN DE LOS PROYECTOS de las facultades.
ADMINISTRADOR	Tendrá el alcance del tipo de usuario DECANO más la opción de DETERMINACIÓN DE PROYECTOS, ADMINISTRACIÓN DEL PND y PEDI, CONFIGURACIONES DE SISTEMA

USUARIO TIPO DECANO

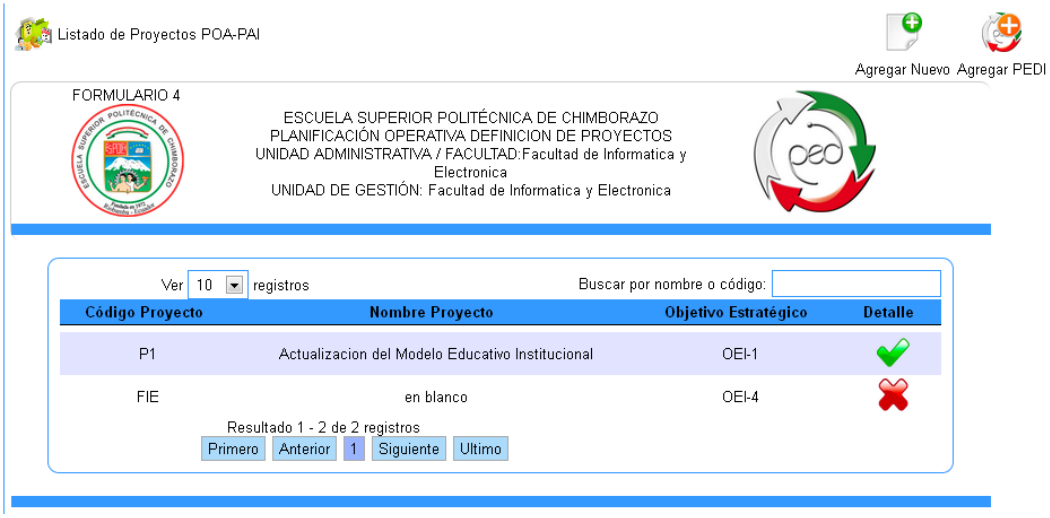
Ejemplo de Usuario tipo decano que aparece en la parte superior derecha

Posee el siguiente menú para la generación del POA y PAI ubicado en la parte superior izquierda de la pantalla



Formular POA

Es donde se puede añadir proyectos existentes en el PEDI o añadir proyectos nuevos

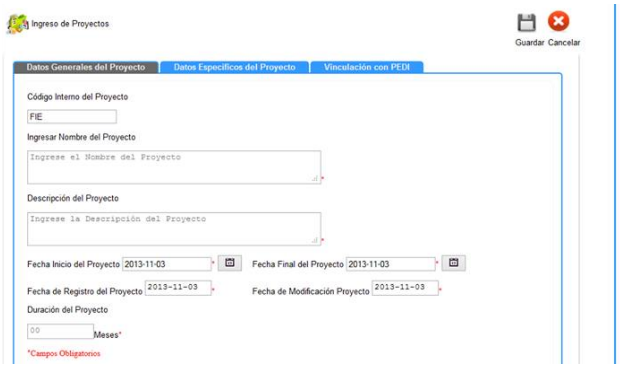


Los proyectos del PEDI se mostrarán como en la figura anterior para añadir al POA y para ingresar proyectos nuevos, ubicados en la parte superior derecha.

En la opción AGREGAR NUEVO es donde se puede crear un nuevo proyecto a partir de datos bases para su posterior aprobación, priorización y determinación si es POA o PAI.

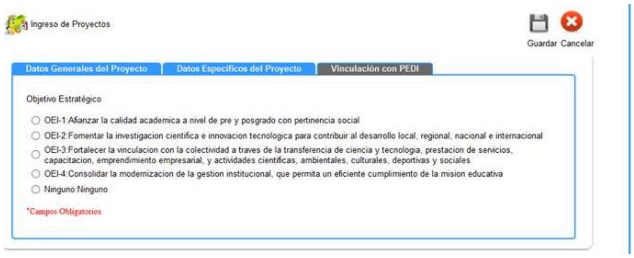
Esta opción consta de 3 pestañas:

DATOS GENERALES: la cual posee datos base de todo proyecto

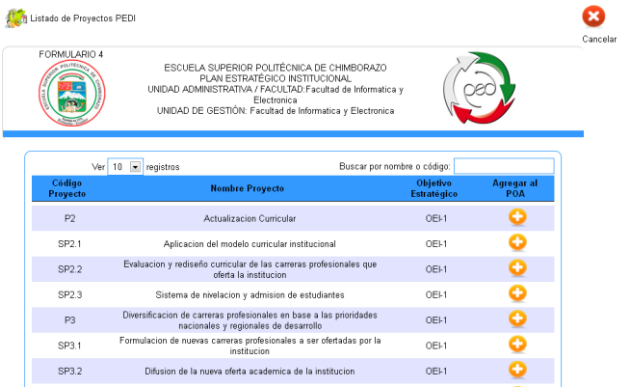


DATOS ESPECÍFICOS: donde se encuentra la opción de escoger el responsable del proyecto dentro de la dependencia que lo ingresa y las observaciones del proyecto.

VINCULACIÓN CON PEDI: donde se selecciona el Objetivo Estratégico el cual forma parte el proyecto



En la opción AGREGAR PEDI es donde se agregan proyectos que están definidos en el PEDI, los cuales solo los agregaremos al POA.

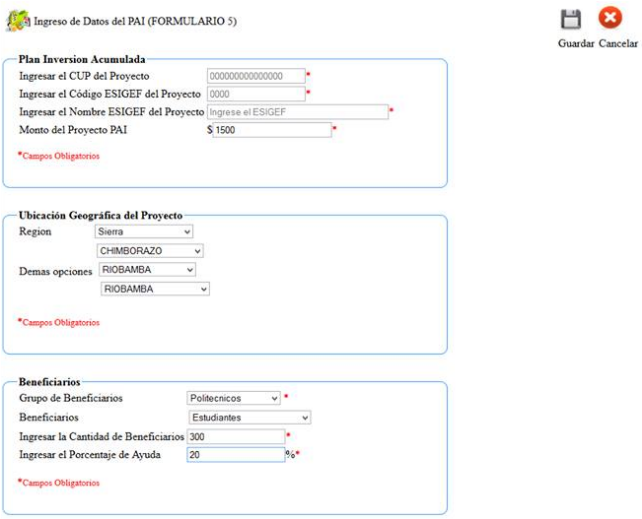


Generar POA

Es donde se visualiza los proyectos nuevos o proyectos del PEDI añadidos al POA.

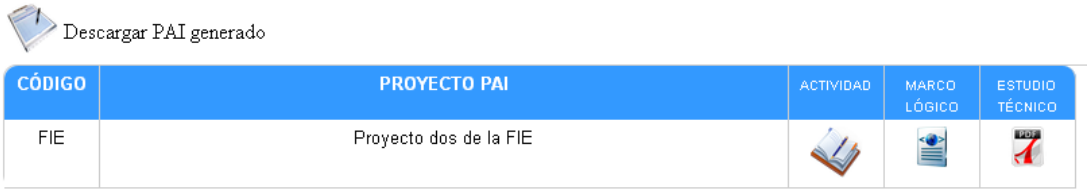
Completar PAI

Es donde se completa la información requerida por SEMPLADES para el tratamiento de proyectos de Inversión.

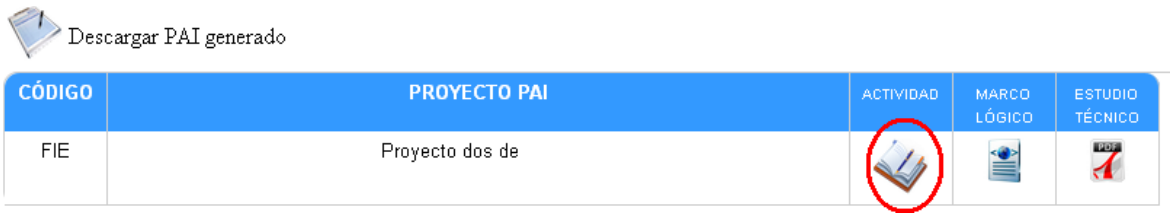


Ingresar Actividades, Marco Lógico y Estudio Técnico

El menú se detalla a continuación:

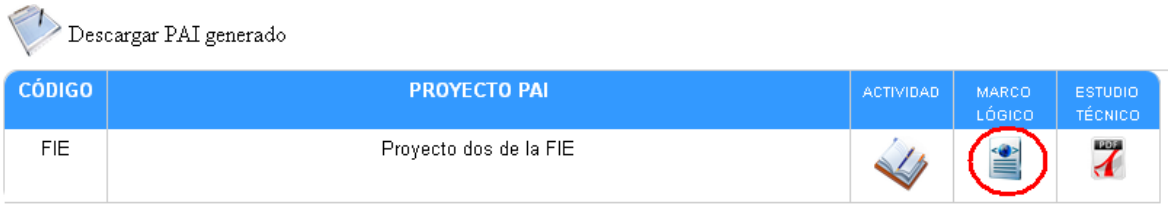


Para ingresar ACTIVIDADES escogemos el vínculo señalado a continuación para ingresar a la opción

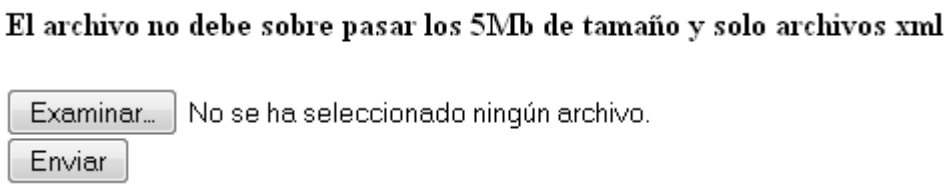


Luego aparece la pantalla para el ingreso de actividades o recursos a utilizar en el proyecto, detallando la cantidad, el costo y unos casilleros para indicar en que meses se utilizarán dichos recursos, los responsables y observaciones. En caso que hubiera actividades o recursos ingresados estos se mostrarán en la parte inferior de esta opción.

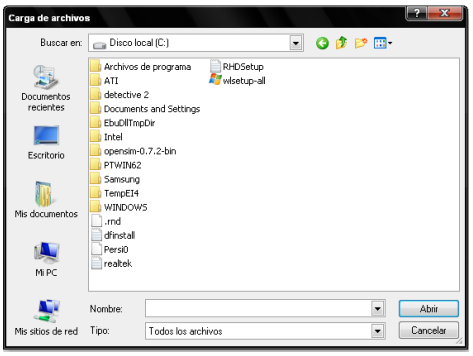
Para ingresar el MARCO LÓGICO escogemos el vínculo señalado a continuación para ingresar a la opción



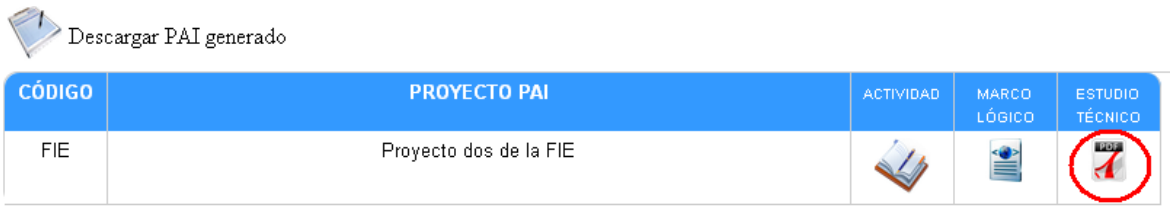
Para luego la opción para subir un archivo de extensión XML generado por el programa SIP offline el cual hace la gestión de marco lógico



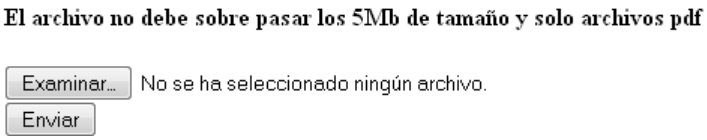
Donde se presiona el botón EXAMINAR… y nos aparece una pantalla para seleccionar el archivo XML



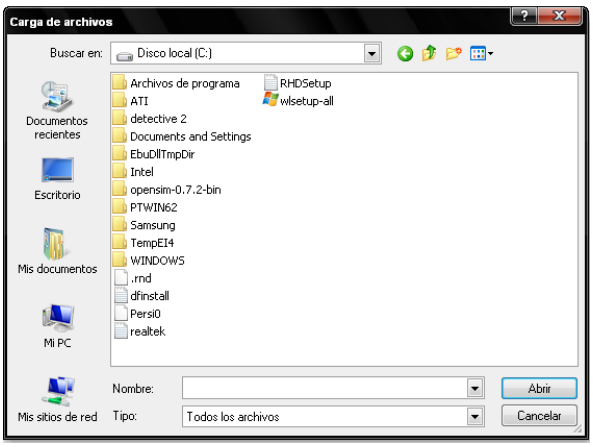
Para ingresar el ESTUDIO TÉCNICO escogemos el vínculo señalado a continuación para ingresar a la opción



Para luego la opción para subir un archivo de extensión PDF el cual contendrá todo el estudio técnico del proyecto.



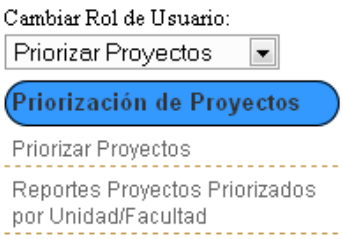
Donde se presiona el botón EXAMINAR… y nos aparece una pantalla para seleccionar el archivo PDF



USUARIO TIPO RECTOR

Ejemplo de Usuario tipo rector que aparece en la parte superior derecha

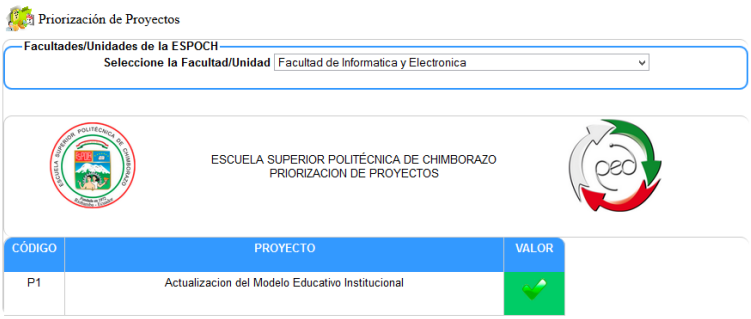
Posee el siguiente menú para la generación del POA y PAI ubicado en la parte superior izquierda de la pantalla que se explicó anteriormente y el nuevo menú que es para PRIORIZAR PROYECTOS



Priorizar Proyectos

Se debe escoger la facultad o dependencia de la cual se priorizará los proyectos.

El cual nos aparecerá los proyectos priorizados y no priorizados, también nos aparecerá semaforizados los proyectos priorizados.



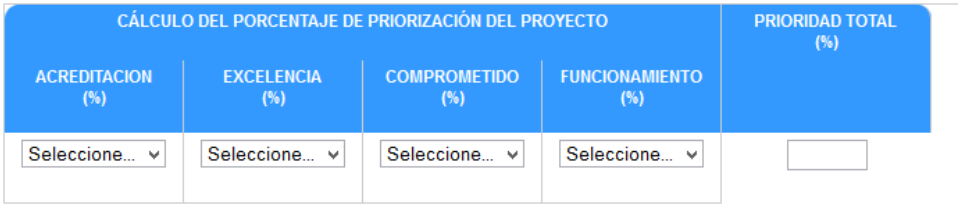
Para priorizar los proyectos se deberán hacer clic en el ícono de la columna

VALOR

NO PRIORIZADO **PRIORIZADO**

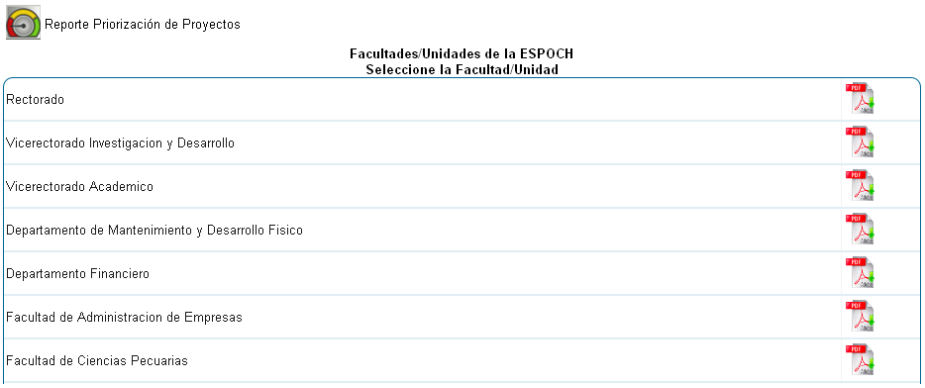
El cual si se encuentra un **VISTO** estará priorizado y se podrá modificar, caso contrario se mostrará un indicador de colores para priorizar el proyecto.

Se seleccionará los valores de BAJA, MEDIA, ALTA, de los parámetros Acreditación, Excelencia, Compromiso y Funcionamiento



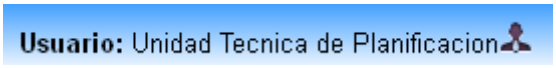
Reportes de Proyectos Priorizados

Saldrá una lista de todas las facultades o dependencias con un vínculo de un archivo PDF en la parte derecha del nombre para escoger y visualizar los reportes por cada una de las facultades o dependencias



USUARIO TIPO ADMINISTRADOR

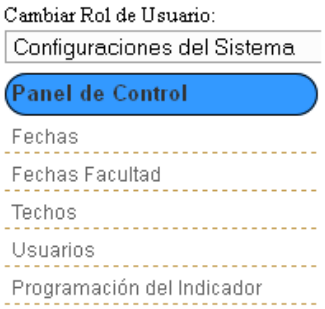
Ejemplo de Usuario tipo rector que aparece en la parte superior derecha



Posee el siguiente menú para la generación del POA y PAI ubicado en la parte superior izquierda de la pantalla que se explicó anteriormente y el nuevo menú que es para CONFIGURACIONES DEL SISTEMA, DETERMINACIÓN DEL PROYECTOS y ADMINISTRACIÓN DEL PND y PEDI.

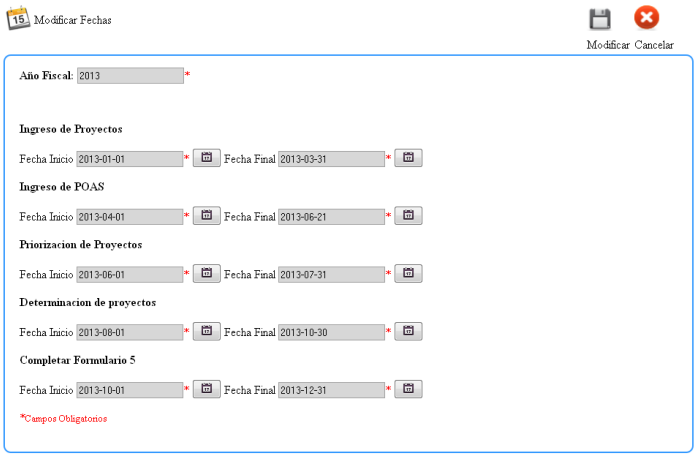
Configuraciones del Sistema

Es donde sea administra las fechas de apertura y cierre del sistema y de cada facultad, los techos presupuestarios, usuario y los tipos de programación del indicador.



Fechas: el cual permite escoger el año fiscal para mostrar los tiempos establecidos en cada una de las fases del sistema

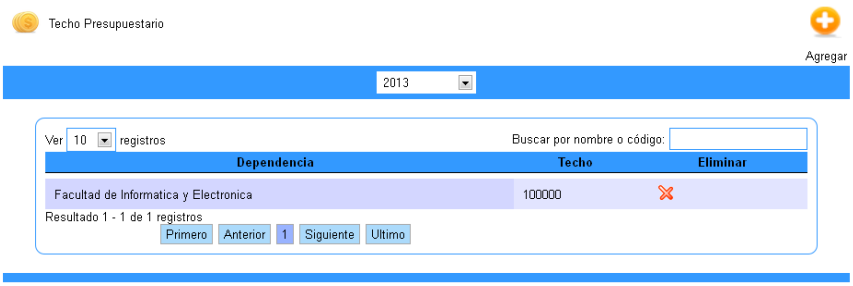
Para modificar las mismas hay que hacer un clic en cualquier fase y aparecerá detalladas las fases con su fecha inicial y final para cambiar mediante un botón con icono de calendario.



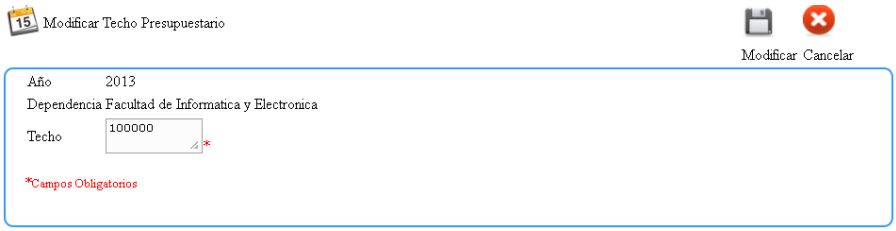
Fechas Facultad: en esta opción se escogerá el año fiscal y la facultad o dependencia que se desea extender o contraer los plazos establecidos.

Y de igual manera se modificarán las fechas establecidas para esa facultad o dependencia.

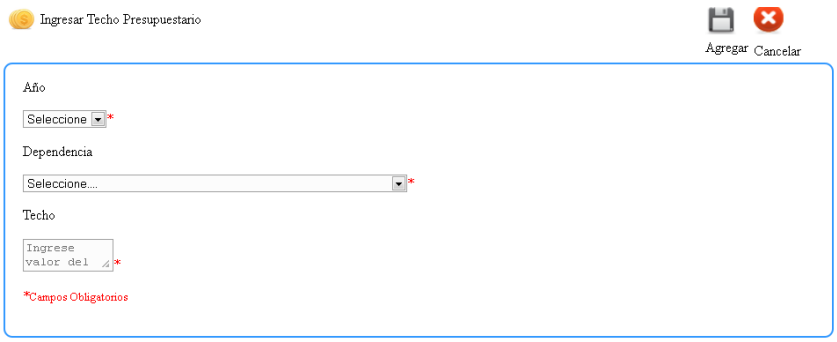
Techos: Se debe escoger el año fiscal y aparecerán todas las facultades o dependencias que ya se hayan asignado un techo.



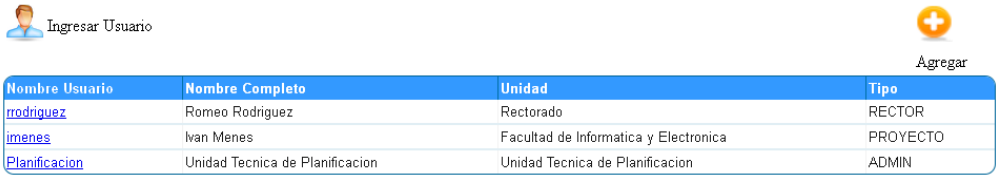
Para modificar algún techo ya ingresado se deberá dar clic en el nombre de la facultad o dependencia y aparecerá el sitio donde se podrá modificar el techo.



Para ingresar un nuevo techo se deberá escoger el año fiscal, la facultad o dependencia e ingresar el valor del techo presupuestario.

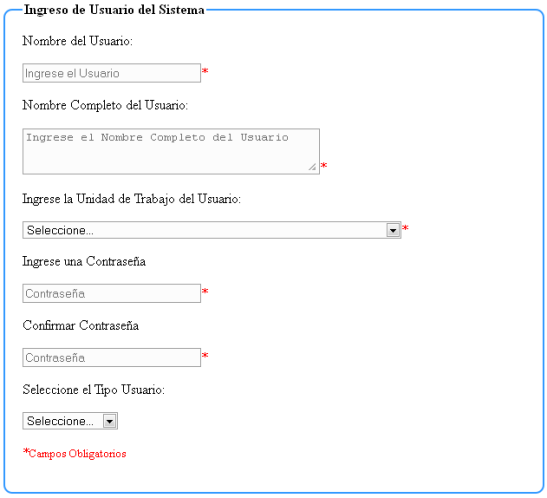


Usuarios: Donde se gestiona los usuarios existentes y nuevos para el sistema

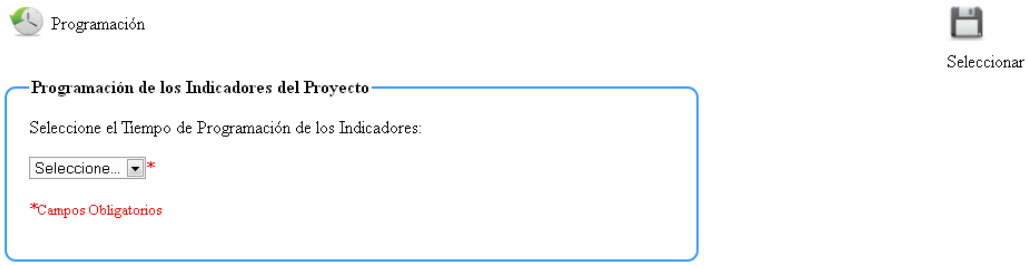


Para modificar los un usuario se deberá dar clic en el nombre del usuario a modificar, para actualizar los datos de dicho usuario.

Para ingresar un nuevo usuario es el mismo formulario de ingreso pero sin datos, donde escribiremos un nombre de usuario, un nombre completo del usuario, la facultad o dependencia, la contraseña y su confirmación y por último el tipo.

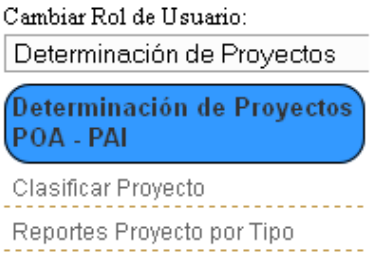


Programación del indicador: Donde se escoge la metodología de programar el indicador, puede ser Mensual, Bimensual, Trimestral, Cuatrimestral y Semestral.

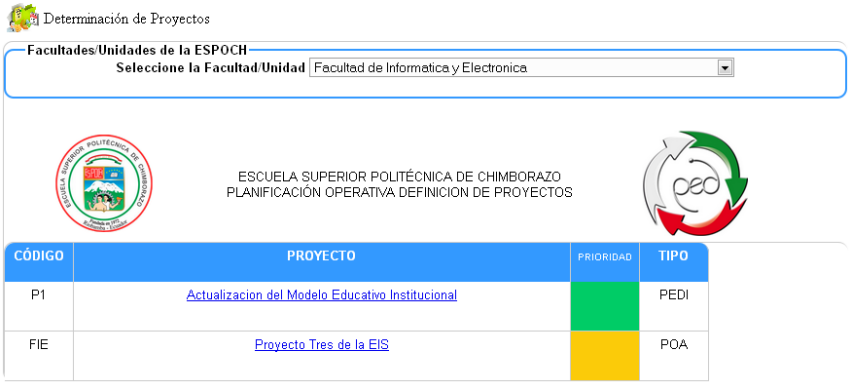


Determinación de Proyectos

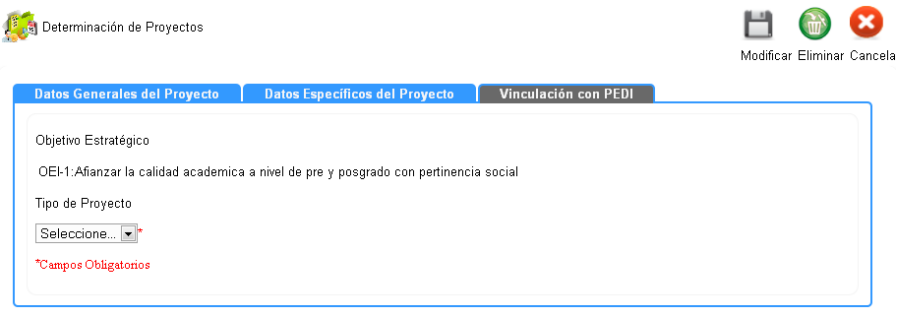
Que posee el siguiente menú



Clasificar Proyecto: donde se deberá escoger la facultad o dependencia y se mostrará los proyectos y su tipo al cual están dirigidos.



Para modificar el tipo se deberá hacer clic en el nombre del proyecto.

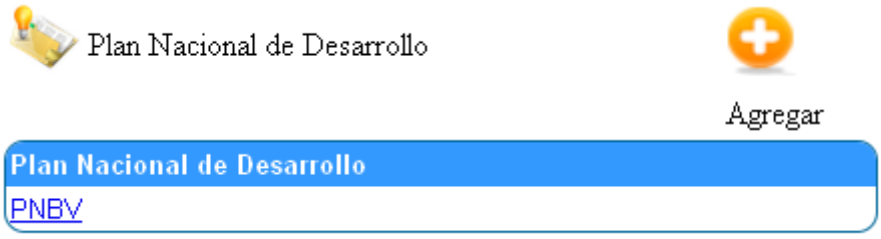


Reportes Proyecto por Tipo: aparecerá un listado de las facultades o dependencias con un ícono de archivo PDF para mostrar en dicho formato un resumen de todos los proyectos agrupados por tipo.

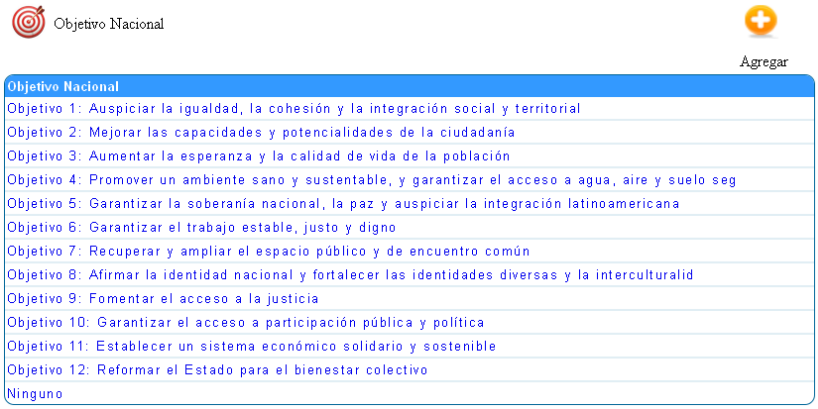
Administración del PND y PEDI

Administrar Plan Nacional, es la sección donde se visualizará toda la gestión correspondiente al Plan Nacional.

Ingresar Plan Nacional de Desarrollo: Es donde se muestra el plan vigente o se ingresa un nuevo plan



Ingresar Objetivo Nacional: Es donde se muestran los objetivos estipulados por el gobierno en el plan del Buen Vivir.



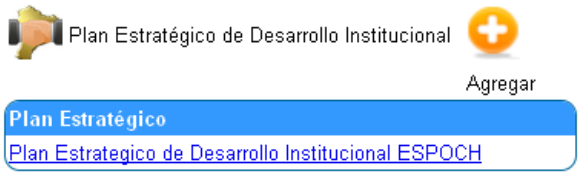
Ingresar Política Nacional: Es donde se muestran las políticas nacionales según los objetivos que tenemos.

Ingresar Meta Nacional: De la misma forma se escogerá la política nacional y se mostrará las metas nacionales a partir de la política a la que pertenece.

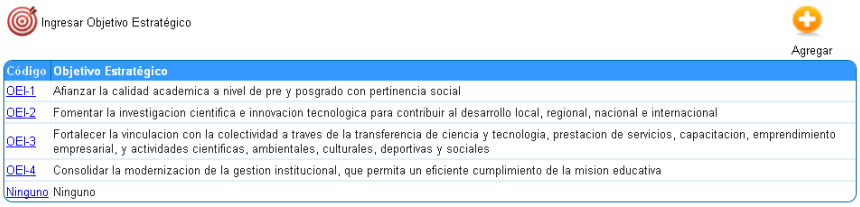
Ver Plan Nacional, es donde se visualizará de forma completa el Plan Nacional ingresado previamente, con opción a descargarlo en archivo de formato PDF.

Plan Estratégico Institucional, es la sección donde se visualizará toda la gestión correspondiente al Plan Estratégico Institucional de la ESPOCH.

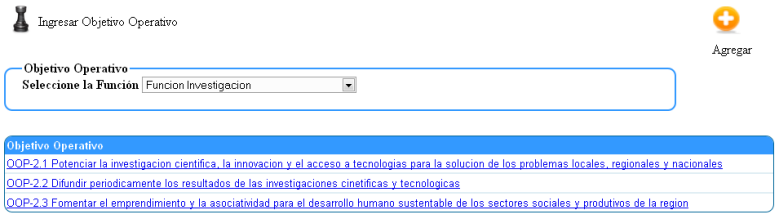
Ingresar Plan Estratégico Institucional: Es donde se muestra el plan estratégico vigente o se ingresa un nuevo plan.



Ingresar Objetivo Estratégico: Es donde se muestra los objetivos estratégicos que posee el plan estratégico.



Ingresar Objetivo Operativo: Es donde se muestran los objetivos operativos que corresponden a una función la cual es vinculada solo a un objetivo estratégico.



Datos Anexos, es la sección donde se visualizará toda la gestión correspondiente a los datos que faltan en el Plan Estratégico Institucional.

Ingresar Política: Es donde se muestran las políticas a seguir según el objetivo estratégico que corresponda.

Ingresar Función: Es donde se muestran las funciones que posee cada objetivo operativo.

Ingresar Estrategia: Es donde se muestran las estrategias que posee cada objetivo estratégico.

Ingresar Meta: Es donde se muestran las metas que posee cada objetivo operativo.

Ver Plan Estratégico Institucional, es la sección donde se visualizará toda la información correspondiente al Plan Estratégico Institucional ingresado.

Evaluación POA y PAI

Es donde se mostrará reportes del seguimiento de los proyectos mediante gráficos estadísticos generados automáticamente.