Universidad de las Ciencias Informáticas

Facultad 1

“Desarrollo del módulo Configuración del subsistema Núcleo para el Sistema de Gestión Universitaria”.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO

EN CIENCIAS INFORMÁTICAS

Autores: Ramón Rodríguez Usatorrez

Ronald Rey Leyva Cala

Tutor: Ing. Joe Del Toro Domínguez

Cotutora: Ing. Kilmeny Acuña Crespo

Ciudad de La Habana, 2010.

Año del 52 aniversario de la Revolución

"Si el presente es de lucha, el futuro es nuestro"

Ernesto Che Guevara

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____ _____

Firma del Autor **Firma del Autor**

Ramón Rodríguez Usatorrez Ronald Rey Leyva Cala

_____

Firma del Tutor

Ing. Joe Del Toro Domínguez

AGRADECIMIENTOS

Le agradezco especialmente a mis padres por su dedicación, amor, educación, ejemplo y apoyo que siempre me han dado y de lo que estaré agradecido de por vida. Gracias por estar siempre conmigo. A mi hermana a quien quiero mucho y nunca ha dejado de apoyarme y aconsejarme.

A mi familia que nunca ha dejado de guiarme y ayudarme durante el transcurso de la carrera.

Al tutor Joe Del Toro Domínguez y la cotutora Kilmen Acuña Crespo por su paciencia, ayuda y dedicación durante el desarrollo de la investigación.

A mis amistades dentro y fuera de la universidad por brindarme apoyo incondicional cuando lo necesité.

A todos aquellos que de una forma u otra han ayudado en mi formación como profesional y como persona. A todos gracias.

Ronald

A mi mamá que ha sido mi amiga, hermana, que me ha guiado en la vida, que me ha ayudado a salir de los malos momentos, te quiero un montón.

A mi papá que me ha apoyado en toda la vida.

A mi hermana Yolennis, que ha sido un ejemplo a seguir y me ha brindado su amistad, tiempo y dedicación como si fuese mi otra madre.

A mis familiares que siempre se han preocupado por mí, los quiero.

A mis amistades que desde primer año estuve molestando, Danay Guará, Adita y a Yeny Milanés, que en esta última etapa han sido de gran ayuda.

A mis colegas del IPU y de la academia, Mario (portuondo), J avier Olivera (la poli), Liosbel (diamante), Gerardo, Yoankis, Yolier, por haberme brindado su amistad incondicionalmente.

Al tutor Joe Del Toro Domínguez y la cotutora Kilmen Acuña Crespo por su paciencia, ayuda y dedicación durante el desarrollo de la tesis.

Ramón Usatorres

DEDICATORIA

A mi mamá y papá por estar siempre atentos a mí, por sus consejos que han hecho de mí cada día una persona mejor. Por la entrega y sacrificio, por sus ejemplos y amor que siempre me han proporcionado. Por todo lo anterior y por muchas más.
A mi hermana por su amor y cariño.

Ronald

A mi madre, mi padre, mi hermana y mi bichito María Belén.

Ramón Usatorres

RESUMEN

La Universidad de las Ciencias Informáticas (UCI) tiene como una de sus principales tareas realizar la gestión de la información universitaria, para la cual se está desarrollando un sistema llamado Sistema de Gestión Universitaria. Éste tiene como objetivo integrar todas las áreas de procesos existentes en la UCI como son: Pregrado, Investigación, Laboratorio entre otras que enriquecen la gestión de la información en la universidad. Las áreas de procesos actualmente están divididas provocando que no se tenga mejor información sobre las acciones que se desarrollan en la universidad, principalmente en la docencia que es de gran importancia para la preparación de futuros profesionales, esto conlleva que la gestión de la información universitaria no sea totalmente completa. Por tanto, es necesario implementar un módulo Configuración dentro del subsistema Núcleo de dicho sistema para que todas las áreas de proceso que se incluyan posean de forma centralizada y estándar, un módulo Configuración que les servirá para adaptar y configurar sus funcionalidades independientemente de las características de cada área de proceso.

PALABRAS CLAVE

- Gestión de la información.
- Sistema de Gestión Universitaria.
- Áreas de proceso.
- Configuración.
- Subsistema Núcleo.
- Módulo.

ÍNDICE DE CONTENIDOS

Introducción.

El avance de las tecnologías de la informática y las comunicaciones (TIC) ha impactado positivamente dentro de la sociedad, con grandes logros y ha sido tema de estudio y reflexión durante mucho tiempo por especialistas e investigadores.

Este continuo crecimiento en el desarrollo de sus técnicas ha estado acompañado por un gran avance en la gestión de la información y el conocimiento. La gestión de la información tiene vital importancia en la esfera educacional, ya que constituye una actividad fundamental a la hora de organizar y controlar todo el proceso docente. Esto abarca documentos, informes, metodologías y publicaciones además está relacionada con la significación que adquieren los datos como resultado de su contextualización, categorización y procesamiento.

En nuestro país con el desarrollo de las TIC, se han estado llevando a cabo nuevos proyectos en diferentes esferas de la sociedad, con el fin de elevar los aportes de la investigación a la producción y comercialización de productos y servicios informáticos.

En la Universidad de las Ciencias Informáticas (UCI) como parte de las tareas para la informatización de los servicios docentes, se desarrolla un Sistema de Gestión Universitaria donde las distintas áreas de proceso que posee como pregrado, las investigaciones y la extensión universitaria conjuntamente con la producción, la superación posgraduada, la residencia, los laboratorios, la biblioteca, la plataforma virtual de aprendizaje y la cooperación tienen gran importancia en la formación de los futuros profesionales por lo que no se pueden ver como procesos aislados dentro de la gestión universitaria.

Debido a que estos procesos son vitales en el desarrollo de la universidad y que unos no pueden existir sin la presencia de otros es necesario que todos converjan en una misma solución, motivo por el cual es necesario el desarrollo de un Sistema de Gestión Universitaria que agrupe y lleve un control en todos los procesos mencionados anteriormente.

El Departamento de Gestión Universitaria de la UCI tiene la necesidad de desarrollar un módulo de configuración en el subsistema Núcleo para el Sistema de Gestión Universitaria. Éste módulo contribuirá a mejorar los procesos de configuración de las áreas de proceso, permitiendo que su utilización y funcionamiento sea de forma centralizada y estándar para cada una de estas áreas.

Por lo que se plantea como **problema científico** a lo antes expuesto lo siguiente: ¿Cómo desarrollar el módulo Configuración del subsistema Núcleo para el Sistema de Gestión Universitaria?

El **objeto de estudio** de la investigación está enmarcado en los procesos de configuración para las áreas de procesos de la UCI.

Campo de acción: Se centra en los procesos de configuración en el subsistema Núcleo del Sistema de Gestión Universitaria.

El **objetivo general** está encaminado a: Desarrollar el módulo Configuración para el Sistema de Gestión Universitaria, derivándose los siguientes **objetivos específicos:**

- Analizar las técnicas de programación, plataformas, herramientas y metodologías de desarrollo de software que faciliten la implementación de dichos módulos.
- Realizar el análisis de la arquitectura propuesta para el desarrollo de la aplicación.
- Desarrollar dicho módulo para el Sistema de Gestión Universitaria.

Tareas:

- Desarrollar habilidades en el uso del lenguaje de programación PHP, del Marco de trabajo CodeIgniter y del gestor de base de datos PostgreSQL.
- Describir las tecnologías y herramientas a utilizar en la realización de la propuesta de solución.
- Estudiar la arquitectura propuesta por el arquitecto principal del Centro de Gestión Universitaria.
- Realizar las tareas de ingeniería para las historias de usuario definidas por el analista.
- Implementar la propuesta de solución.
- Realizar pruebas internas.

Con el cumplimiento de las tareas anteriores se tiene como **Idea a Defender** que si se desarrolla el módulo de Configuración, entonces se podrá contar con un mejor control en la realización de los cambios en cada área de proceso dentro del Sistema de Gestión Universitaria.

Los siguientes **métodos teóricos** sustentan la investigación:

Histórico – lógico

Su empleo nos permitió un mejor conocimiento de la evolución del proceso de control docente dándonos una base para el fundamento de la realización de un nuevo módulo y su importancia así como sus aplicaciones más recientes en el ámbito docente, y nos aporto conocimiento de las mejores técnicas y herramientas para el desarrollo del módulo Configuración.

Analítico-sintético

Se realizó un análisis profundo y continuo de la gestión de la información en su proceso intelectual y por otra parte realizamos el análisis en cuanto a los distintos documentos que están vinculados con este tema y se llegó a la conclusión de que integrando nuevas funcionalidades se obtenían soluciones más prácticas, lográndose así la realización de un sistema mucho más eficaz.

Los siguientes **métodos empíricos** sustentan la investigación:

Observación

A través de este método podemos darnos cuenta de la necesidad de realizar nuevos módulos que brinden nuevas funcionalidades y lograr enriquecer la informatización universitaria, para el uso inmediato docente el cual será de gran importancia en el trabajo educativo tanto profesional como del estudiantado.

Revisión de documentos

La revisión de documentos posee gran importancia ya que contienen toda la información de lo que se realice, explicándose detalladamente cada proceso, en fin este método nos dio una visión más detallada de lo que se quiere, se necesita hacer y de cómo hacerlo.

El desarrollo y objetivo de la investigación quedaría estructurado de la siguiente manera:

Capítulo 1. Fundamentación Teórica: En este capítulo se exponen los elementos teóricos que soportan el trabajo realizado, su importancia, además de las diferentes técnicas de programación que existen a nivel internacional, nacional y de la Universidad. Además, se realiza un estudio de los lenguajes, herramientas, marco de trabajo, entorno de desarrollo, plataforma de desarrollo y metodología entre otras, empleadas durante el desarrollo de la propuesta de solución, acompañadas de una argumentación de su selección.

Capítulo 2. Análisis y Descripción de la Solución Propuesta: Se brinda una descripción de la implementación del sistema; como componentes que puedan ser rehusados así también clases que modelan la solución y las principales funcionalidades de cada una de ellas.

Capítulo 3. Validación de la Solución Propuesta: En este capítulo se analizan algunas de las características que presentan las pruebas funcionales que son empleadas para la validación de la solución propuesta, y finalmente se realizan pruebas para la validación de la solución y dar una valoración final de lo desarrollado.

CAPÍTULO 1 Fundamentación Teórica.

1.1 Introducción.

En este capítulo se explica los procesos de configuración, se hace un estudio de los sistemas similares que existen en la universidad y se describen las técnicas de programación que existen a nivel internacional, nacional y de la universidad. También se especifica y describe la plataforma de desarrollo, el marco de trabajo, herramientas y gestor de base de datos, además de la metodología y entorno de desarrollo que se utilizan para el desarrollo del módulo.

1.2 Procesos de configuración.

Son un conjunto de actividades o eventos que se realizan dentro de un software para resolver las necesidades específicas de un usuario o del propio software. Es una tarea esencial antes de trabajar con cualquier nuevo elemento. La tendencia actual es a reducir las necesidades de configuración y facilitar al máximo el proceso de configuración.

Los elementos de configuración más importantes en el Sistema de Gestión Universitarias son todos aquellos que serán usados por otros módulos y áreas tales son los casos de:

- Atributos globales: Es objetivo configurar los nomencladores que serán usados por las áreas de procesos como son los tipos de áreas y áreas político administrativas.
- Conexiones: Será objeto de configuración la base de datos, archivos y otros que sean de importancia para el funcionamiento del sistema.
- Operaciones de actualizar: Tal es el caso de exportar, importar, actualizar y desinstalar módulos o subsistemas que están incluidos o serán incluidos en el sistema.
- Preferencias: Tales como los temas e idiomas que serán adaptados al sistema para su utilización en un momento determinado.
- Configuraciones generales del sistema: Dígase dominios, entidades, y otros que son usados en módulos como el de Administración para el Sistema de Gestión Universitaria.

Los elementos de configuración pueden variar en dependencia a las necesidades del cliente y a las características del sistema a implementar.

1.3 Sistemas similares.

La mayoría de los sistemas informáticos tanto nacionales como internacionales tienen complejos procesos que varían día a día en función de su situación particular, por lo que se hace necesario que cada sistema tenga un mecanismo de configuración que le provea ayuda a la hora de realizar cambios o definir nuevas funcionalidades dentro de este. Un enfoque objetivo de la configuración es que puede mejorar el buen funcionamiento y control en un sistema informático.

En la Universidad de Ciencias Informáticas (UCI) es objetivo y de gran importancia que cada sistema o aplicación web contenga un módulo configuración que les permita realizar cambios de forma tal que se satisfagan las necesidades del cliente. Un ejemplo actual de sistemas que utilizan módulos de configuración en la Universidad de Ciencias Informáticas, es el producto de ERP (Planificador de Recursos Económicos) cubano llamado Cedrux (Sistema Integral de Gestión) que contiene dentro de su estructura un módulo configuración que le permite gestionar los procesos de cambios. Este mismo módulo de configuración implementado por el ERP es utilizado por otro sistema informático llamado GeForza (Sistema Unificado de Fuerza de Trabajo Calificada), el cual se apoya en dicho módulo para dar solución a sus necesidades de configuración.

1.4 Técnicas de Programación.

Las técnicas de programación constituyen parte fundamental en el proceso de desarrollo e Ingeniería del Software dentro del ámbito informático.

Cada técnica tiene sus propias características, y distintos métodos de resolución de problemas, así como la implementación de estándares de ciertas compañías o instituciones, y es de gran importancia aprender a implementarlas a la hora de adentrarse en la evolución de cualquier proyecto de desarrollo de software.

En este tema, se describen varias técnicas de programación que son aplicadas con frecuencia para segmentar un proyecto a mediana, pequeña o gran escala, en problemas más simples que un grupo de trabajo o incluso un sólo programador, puede resolver.

1.4.1 Programación modular.

La programación modular es un paradigma de programación que consiste en dividir un programa en módulos o subprogramas con el fin de hacerlo más legible y manejable. Se presenta históricamente como una evolución de la programación estructurada para solucionar problemas de programación más grandes y complejos de lo que ésta puede resolver.

Al aplicar la programación modular, un problema complejo debe ser dividido en varios subproblemas más simples, y estos a su vez en otros subproblemas más simples. Esto debe hacerse hasta obtener subproblemas lo suficientemente simples como para poder ser resueltos fácilmente con algún lenguaje de programación. Esta técnica se llama refinamiento sucesivo, divide y vencerás o análisis descendente.

Un módulo es subprograma que resuelve uno de los subproblemas en que se divide el problema complejo original. Cada uno de estos módulos tiene una tarea bien definida y algunos necesitan de otros para poder operar. En caso de que un módulo necesite de otro, puede comunicarse con éste mediante una interfaz de comunicación que también debe estar bien definida.

Si bien un módulo puede entenderse como una parte de un programa en cualquiera de sus formas y variados contextos, en la práctica es común representarlos con procedimientos y funciones. También pueden considerarse módulos las librerías que se incluyen en un programa o en la programación orientada a objetos.

1.4.2 Programación estructurada.

La programación estructurada es una forma de escribir programas de ordenador (programación de computadora) de manera clara. Para ello utiliza únicamente tres estructuras: secuencia, selección e iteración; siendo innecesario el uso de la instrucción o instrucciones de transferencia incondicional (GOTO, EXIT FUNCTION, EXIT SUB o múltiples RETURN).

Hoy en día las aplicaciones informáticas son mucho más ambiciosas que las necesidades de programación existentes en los años 1960, principalmente debido a las aplicaciones gráficas, por lo que las técnicas de programación estructurada no son suficientes. Ello ha llevado al desarrollo de nuevas técnicas, tales como la programación orientada a objetos y el desarrollo de entornos de programación que facilitan la programación de grandes aplicaciones.

1.4.3 Programación orientada a objetos.

La programación orientada a objetos o POO (OOP según sus siglas en inglés) es un paradigma de programación que usa objetos y sus interacciones, para diseñar aplicaciones y programas de ordenador. Está basado en varias técnicas, incluyendo herencia, abstracción, polimorfismo y encapsulamiento. Su uso se popularizó a principios de la década de 1990. En la actualidad, existe variedad de lenguajes de programación que soportan la orientación a objetos.

La programación orientada a objetos, expresa un programa como un conjunto de estos objetos, que colaboran entre ellos para realizar tareas. Esto permite hacer los programas y módulos más fáciles de escribir, mantener, y reutilizar.

Un objeto contiene toda la información que permite definirlo e identificarlo frente a otros objetos pertenecientes a otras clases e incluso frente a objetos de una misma clase, al poder tener valores bien diferenciados en sus atributos. A su vez, los objetos disponen de mecanismos de interacción llamados métodos, que favorecen la comunicación entre ellos. Esta comunicación favorece a su vez el cambio de estado en los propios objetos. Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separa el estado y el comportamiento.

La POO difiere de la programación estructurada tradicional, en la que los datos y los procedimientos están separados y sin relación, ya que lo único que se busca es el procesamiento de unos datos de entrada para obtener otros de salida. La programación estructurada anima al programador a pensar sobre todo en términos de procedimientos o funciones, y en segundo lugar en las estructuras de datos que esos procedimientos manejan. En la programación estructurada sólo se escriben funciones que procesan datos. Los programadores que emplean POO, en cambio, primero definen objetos para luego enviarles mensajes solicitándoles que realicen sus métodos por sí mismos.

1.4.4 Programación lógica.

La programación lógica, junto con la funcional, forma parte de lo que se conoce como programación declarativa. En los lenguajes tradicionales, la programación consiste en indicar cómo resolver un problema mediante sentencias; en la programación lógica, se trabaja de una forma descriptiva, estableciendo relaciones entre entidades, indicando no cómo, sino qué hacer. La ecuación de Robert Kowalski (Universidad de Edimburgo) establece la idea esencial de la programación lógica: algoritmos = lógica + control. Es decir, un algoritmo se construye especificando conocimiento en un lenguaje formal (lógica de primer orden), y el problema se resuelve mediante un mecanismo de inferencia (control) que actúa sobre aquél.

La programación lógica consiste en la aplicación del corpus de conocimiento sobrelógica para el diseño de lenguajes de programación. La programación lógica comprende dos paradigmas de programación: la programación declarativa y la programación funcional. La programación declarativa gira en torno al concepto de predicado, o relación entre elementos. La programación funcional se basa en el concepto defunción (que no es más que una evolución de los predicados), de corte más temático. (1)

La programación lógica construye base de conocimientos mediante reglas y hechos. Un lenguaje de programación lógica por excelencia es el Prolog.

1.4.5 Programación concurrente.

Se conoce por programación concurrente a la rama de la informática que trata de las notaciones y técnicas de programación que se usan para expresar el paralelismo potencial entre tareas y para resolver los problemas de comunicación y sincronización entre procesos. Un programa concurrente es aquel que ejecuta más de una actividad simultáneamente. En la programación concurrente se supone que hay un procesador utilizable por cada tarea; no se sabe si el procesador será una unidad independiente para cada uno de ellos o si será un solo CPU que se comparte en el tiempo entre las tareas. Independientemente de cómo se vaya a ejecutar realmente el programa, en un sistema monoprocesador o multiprocesador, el resultado debe ser el correcto. Por ello, se supone que existe un conjunto de instrucciones primitivas que son, o bien parte del sistema operativo o bien parte de un lenguaje de programación, y cuya correcta implementación y corrección está garantizada por el sistema.

1.4.6 Programación funcional.

La programación funcional es un paradigma de la programación declarativa basado en la utilización de funciones matemáticas. (2)

Los lenguajes funcionales ofrecen al programador un buen número de recursos expresivos que permiten resolver problemas complejos mediante programas pequeños y robustos. La programación funcional apareció como un paradigma independiente a principio de los 60. Su creación es debida a las necesidades de los investigadores en el campo de la inteligencia artificial y en sus campos secundarios del cálculo simbólico, pruebas de teoremas, sistemas basados en reglas y procesamiento del lenguaje natural. Estas necesidades no estaban cubiertas por los lenguajes imperativos de la época.

La característica principal de la programación funcional es que los cálculos se ven como una función matemática que hacen corresponder entradas y salidas. No hay noción de posición de memoria y por tanto, no hay necesidad de una instrucción de asignación. Los bucles se modelan a través de la recursividad ya que no hay manera de incrementar o disminuir el valor de una variable. Como aspecto práctico casi todos los lenguajes funcionales soportan el concepto de variable, asignación y bucle. Estos elementos no forman parte del modelo funcional “puro”.

Existen dos grandes categorías de lenguajes funcionales: los funcionales puros y los híbridos. La diferencia entre ambos estriba en que los lenguajes funcionales híbridos son menos dogmáticos que los puros, al admitir conceptos tomados de los lenguajes imperativos, como las secuencias de instrucciones o la asignación de variables. En contraste, los lenguajes funcionales puros tienen una mayor potencia expresiva, conservando a la vez su transparencia referencial, algo que no se cumple siempre con un lenguaje funcional híbrido. (3)

Haskell es un lenguaje funcional puro, no es un lenguaje muy rápido, pero se prioriza el tiempo del programador sobre el tiempo de computación.

1.5 Software libre.

El software libre es una cuestión de la libertad de los usuarios de ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. Más precisamente, significa que los usuarios de programas tienen las cuatro libertades esenciales. (4)

- La libertad de ejecutar el programa, para cualquier propósito (libertad 0).
- La libertad de estudiar cómo trabaja el programa, y cambiarlo para que haga lo que usted quiera (libertad 1). El acceso al código fuente es una condición necesaria para ello.
- La libertad de redistribuir copias para que pueda ayudar al prójimo (libertad 2).
- La libertad de distribuir copias de sus versiones modificadas a terceros (la 3ª libertad). Si lo hace, puede dar a toda la comunidad una oportunidad de beneficiarse de sus cambios. El acceso al código fuente es una condición necesaria para ello.

El desarrollo del software libre se ha ido incrementando cada vez más por las siguientes razones:

- La motivación *ética*, abanderada por la Fundación de Software libre (*Free Software Foundation*), heredera de la cultura hacker, y partidaria del apelativo libre, que argumenta que el software es conocimiento y debe poderse difundir sin trabas. Su ocultación es una actitud antisocial y la posibilidad de modificar programas es una forma de libertad de expresión.
- La motivación pragmática, abanderada por la Iniciativa Código Abierto (*Open Source Initiative*) y partidaria del apelativo *abierto*, que argumenta ventajas técnicas y económicas.

Aparte de estas dos grandes motivaciones, la gente que trabaja en software libre suele hacerlo por muchas otras razones, que van desde la diversión a la mera retribución económica, que es posible debido a modelos de negocio sustentables.

Entre las ventajas que posee el Software Libre están:

- Bajo costo de adquisición: Se trata de un software económico ya que permite un ahorro de grandes cantidades en la adquisición de las licencias.

- Innovación tecnológica: Esto se debe a que cada usuario puede aportar sus conocimientos y su experiencia y así decidir de manera conjunta hacia donde se debe dirigir la evolución y el desarrollo del software. Es un gran avance en la tecnología mundial.
- Independencia del proveedor: Al disponer del código fuente, se garantiza una independencia del proveedor que hace que cada empresa o persona pueda seguir contribuyendo al desarrollo y los servicios del software.
- Escrutinio público: Esto hace que la corrección de errores y la mejora del producto se lleven a cabo de manera rápida y eficaz por cada uno de los usuarios que lleguen a utilizar el producto.
- Adaptación del software: Esta cualidad resulta de gran utilidad para empresas e industrias específicas que necesitan un software personalizado para realizar un trabajo específico y con el software libre se puede realizar y con costes mucho más razonables.
- Lenguas: Aunque el software se cree y salga al mercado en una sola lengua, el hecho de ser software libre facilita en gran medida su traducción y localización para que usuarios de diferentes partes del mundo puedan aprovechar estos beneficios.

1.6 Aplicaciones Web.

Se denomina aplicación web a aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador. En otras palabras, es una aplicación software que se codifica en un lenguaje soportado por los navegadores web en la que se confía la ejecución al navegador.

Las aplicaciones web son populares debido a lo práctico del navegador web como cliente ligero, así como a la facilidad para actualizar y mantener aplicaciones web sin distribuir e instalar software a miles de usuarios potenciales. Existen aplicaciones como las tiendas en línea entre otras.

Las aplicaciones web son de gran importancia para el uso empresarial por tener una estrategia que está emergiendo para las empresas proveedoras de software que consiste en proveer acceso vía web al software. Para aplicaciones previamente distribuidas, como las aplicaciones de escritorio, se puede optar por desarrollar una aplicación totalmente nueva o simplemente por adaptar la aplicación para ser usada con una interfaz web. Estos últimos programas permiten al usuario pagar una cuota mensual o anual para usar la aplicación, sin necesidad de instalarla en el ordenador del usuario. A esta estrategia de uso se la denomina Software como servicio y a las compañías desarrolladoras se les denomina Proveedores de Aplicaciones de Servicio (ASP por sus siglas en inglés), es un modelo de negocio que está atrayendo la atención de la industria del software.

Sus ventajas radican en lo siguiente:

Ahorra tiempo: Se pueden realizar tareas sencillas sin necesidad de descargar ni instalar ningún programa.

No hay problemas de compatibilidad: Basta tener un navegador actualizado para poder utilizarlas.

No ocupan espacio en nuestro disco duro.

Actualizaciones inmediatas: Como el software lo gestiona el propio desarrollador, cuando nos conectamos estamos usando siempre la última versión que haya lanzado.

Inmediatez de acceso. Las aplicaciones basadas en web no necesitan ser descargadas, instaladas y configuradas. Usted accede a su cuenta en línea (online) y están listas para trabajar sin importar cuál es su configuración o su hardware. (5)

Menos requerimientos de memoria. Las aplicaciones basadas en web tienen muchas más razonables demandas de memoria RAM de parte del usuario final que los programas instalados localmente. Al residir y correr en los servidores del proveedor, a esas aplicaciones basadas en web usa en muchos casos la memoria de las computadoras que ellos corren, dejando más espacio para correr múltiples aplicaciones del mismo tiempo sin incurrir en frustrantes deterioros en el rendimiento. (6)

Multiplataforma: Se pueden usar desde cualquier sistema operativo porque sólo es necesario tener un navegador.

Portables: Es independiente del ordenador donde se utilice (un PC de sobremesa, un portátil) porque se accede a través de una página web (sólo es necesario disponer de acceso a Internet). La reciente tendencia al acceso a las aplicaciones web a través de teléfonos móviles requiere sin embargo un diseño específico de los ficheros CSS para no dificultar el acceso de estos usuarios.

La disponibilidad suele ser alta porque el servicio se ofrece desde múltiples localizaciones para asegurar la continuidad del mismo.

Los virus no dañan los datos porque éstos están guardados en el servidor de la aplicación.

Colaboración: Gracias a que el acceso al servicio se realiza desde una única ubicación es sencillo el acceso y compartición de datos por parte de varios usuarios. Tiene mucho sentido, por ejemplo, en aplicaciones en línea de calendarios u oficina.

1.7 Herramientas, Lenguajes, Gestores de base de datos y Tecnologías asociadas al desarrollo del sistema.

Para el desarrollo de la aplicación se utilizarán un grupo de herramientas, tecnologías, lenguajes definidos con las cuales se asegurará la calidad y entrega en tiempo del módulo a desarrollar.

1.7.1 Entornos integrados de desarrollo (IDE).

Un entorno de desarrollo integrado o IDE, es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios.

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (*GUI*). Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. El lenguaje Visual Basic, por ejemplo, puede ser usado dentro de las aplicaciones de Microsoft Office, lo que hace posible escribir sentencias *Visual Basic* en forma de macros para *Microsoft Word*.

Los IDE proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como *PHP, C++, Python, Java, C#, Delphi, Visual Basic*, etc. En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto.

1.7.1.1 Netbeans.

Es un producto que posee doble licencia, *Common Development y Distribution License (CDDL)* y *GNU General Public License* versión 2 con *Classpath exception (GPL2).*

El NetBeans IDE es un entorno de desarrollo para Windows, Mac, Linux y Solaris. El proyecto NetBeans consiste en un IDE de código abierto y una plataforma de aplicaciones que permiten a los desarrolladores crear rápidamente web, empresa, escritorio y aplicaciones móviles utilizando la plataforma Java, así como JavaFX, PHP, JavaScript y Ajax, Ruby y Ruby on Rails ,Groovy y Grails, y C / C + +. Permite crear a los desarrolladores rápidamente una Web, aplicaciones de escritorio y móviles utilizando la plataforma Java, así como JavaFX, PHP, JavaScript y Ajax, C / C + +, entre otros.

El proyecto NetBeans es compatible con una vibrante comunidad de desarrolladores, ofrece una amplia documentación y recursos de capacitación, así como una amplia selección de plug-ins de terceros.

1.7.2 Marco de Trabajo (*Framework*).

Es una estructura de soporte definida mediante la cual otro proyecto de software puede ser desarrollado y organizado. Puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto. Representa una arquitectura de software que modela las relaciones generales de las entidades del

dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

Son diseñados con el intento de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional. Es el esqueleto sobre el cual varios objetos son integrados para una solución dada. No es más que una base de programación que atiende a sus descendientes (manejado de una forma estructural y/o en cascada) posibilitando cualquier respuesta ante las necesidades de sus miembros, o secciones de una aplicación web.

1.7.2.1 CodeIgniter.

Como marco de trabajo está CodeIgniter 1.7.2 que es un entorno de desarrollo PHP para la creación rápida de aplicaciones web. Es un producto de código libre, libre de uso para cualquier aplicación. Contiene una serie de librerías que sirven para el desarrollo de aplicaciones web y además propone una manera de desarrollarlas que se debe seguir para obtener provecho de la aplicación.

Este marco de trabajo utiliza el patrón de diseño de software MVC, lo que facilita mucho estructurar la aplicación y su mantenimiento, además posee helpers, bibliotecas, etc. que facilitan y agilizan mucho el desarrollo. (7)

Contiene muchas ayudas para la creación de aplicaciones PHP avanzadas, que hacen del proceso de desarrollo más rápido. A la vez, define una arquitectura de desarrollo que hará que se programe de una manera más ordenada y contiene diversas herramientas que ayudan a hacer aplicaciones más versátiles y seguras.

CodeIgniter ayuda a dar un salto definitivo a los desarrolladores PHP, creando aplicaciones Webs más profesionales y con código más reutilizable, Está creado para que sea fácil de instalar en cualquier servidor y de utilizar. Además, muchas de sus utilidades y modos de funcionamiento son opcionales, lo que hace que goces de mayor libertad a la hora de desarrollar sitios web.

Algunos de los puntos más interesantes sobre este marco de trabajo, en comparación con otros productos similares, son los siguientes:

Compatibilidad: CodeIgniter es compatible con la versión PHP 4, lo que hace que se pueda utilizar en cualquier servidor, incluso en algunos antiguos. Por supuesto, funciona correctamente también en PHP 5.

Versatilidad: Quizás la característica principal de CodeIgniter, en comparación con otros marcos de trabajo PHP, es capaz de trabajar en la mayoría de los entornos o servidores, incluso en sistemas de

alojamiento compartido, donde sólo tenemos un acceso por FTP para enviar los archivos al servidor y donde no tenemos acceso a su configuración.

Facilidad de instalación: No es necesario más que una cuenta de FTP para subir CodeIgniter al servidor y su configuración se realiza con apenas la edición de un archivo.

Ligereza: El núcleo de CodeIgniter es bastante ligero, lo que permite que el servidor no se sobrecargue interpretando o ejecutando grandes porciones de código. La mayoría de los módulos o clases que ofrece se pueden cargar de manera opcional, sólo cuando se van a utilizar realmente.

Flexibilidad: CodeIgniter es bastante menos rígido que otros marcos de trabajo. Define una manera de trabajar específica, pero en muchos de los casos podemos seguirla o no y sus reglas de codificación muchas veces nos las podemos saltar para trabajar como más a gusto encontremos.

Sin duda, lo más destacable de CodeIgniter es su accesibilidad, ya que podemos utilizarlo en la mayor gama de entornos. Es el marco de trabajo ideal si lo que se busca es un rendimiento excepcional. Al usar este maco de trabajo se evita la complejidad, a favor de soluciones simples y se cuenta con la documentación completa.

1.7.3 JQuery.

JQuery es un nuevo tipo de biblioteca o Marco de desarrollo de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, permitiendo manejar eventos, desarrollar animaciones y agregar interacción con la tecnología AJAX al sistema. JQuery, al igual que otras librerías, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código. Es decir, con las funciones propias de esta librería se logran grandes resultados en menos tiempo y espacio. La gran ventaja de JQuery es que permite cambiar el contenido de nuestra página web sin necesidad de recargarla, utilizando DOM y AJAX de manera extremadamente sencilla gracias a su sintaxis.

1.7.4 Gestores de bases de datos.

Se trata de un conjunto de programas no visibles al usuario final que se encargan de la privacidad, la integridad, la seguridad de los datos y la interacción con el sistema operativo. Proporciona una interfaz entre los datos, los programas que los manejan y los usuarios finales. (8)

Un sistema gestor de base de datos se define como el conjunto de programas que administran y gestionan la información contenida en una base de datos. Ayuda a realizar las siguientes acciones:

- Definición de los datos.
- Mantenimiento de la integridad de los datos dentro de la base de datos.

- Control de la seguridad y privacidad de los datos.
- Manipulación de los datos.

1.7.4.1 Postgresql.

Posee una licencia PostgreeSQL, un liberal de licencia de código abierto, similar a las licencias BSD o MIT. PostgreeSQL ofrece muchas ventajas para su compañía o negocio respecto a otros sistemas de bases de datos, entre las que tenemos:

Instalación Ilimitada: Es frecuente que las bases de datos comerciales sean instaladas en más servidores de lo que permite la licencia. Algunos proveedores comerciales consideran a esto la principal fuente de incumplimiento de licencia. Se puede usar PostgreeSQL, pues no estaría violando acuerdos de licencia, puesto que no hay costo asociado a la licencia del software.

Esto tiene varias ventajas adicionales:

- Modelos de negocios más rentables con instalaciones a gran escala.
- No existe la posibilidad de ser auditado para verificar cumplimiento de licencia en ningún momento.
- Flexibilidad para hacer investigación y desarrollo sin necesidad de incurrir en costos adicionales de licenciamiento.

PostgreSQL está ampliamente considerado como el sistema de bases de datos de código abierto más avanzado del mundo. Posee muchas características que tradicionalmente sólo se podían ver en productos comerciales de alto calibre.

1.7.5 PgAdmin III.

PgAdmin III es una aplicación gráfica para gestionar el gestor de bases de datos PostgreSQL, siendo la más completa y popular con licencia Open Source. Es capaz de gestionar versiones a partir de la PostgreSQL 7.3. Está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. El interfaz gráfico soporta todas las características de PostgreSQL y facilita enormemente la administración. La aplicación también incluye un editor SQL con resaltado de sintaxis.

1.7.6 Metodología de desarrollo.

La metodología de desarrollo de software es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. A lo largo del tiempo, una gran cantidad

de métodos han sido desarrollados diferenciándose por su fortaleza y debilidad, esto dio surgimiento a los tipos de metodologías Ágiles y Robustas.

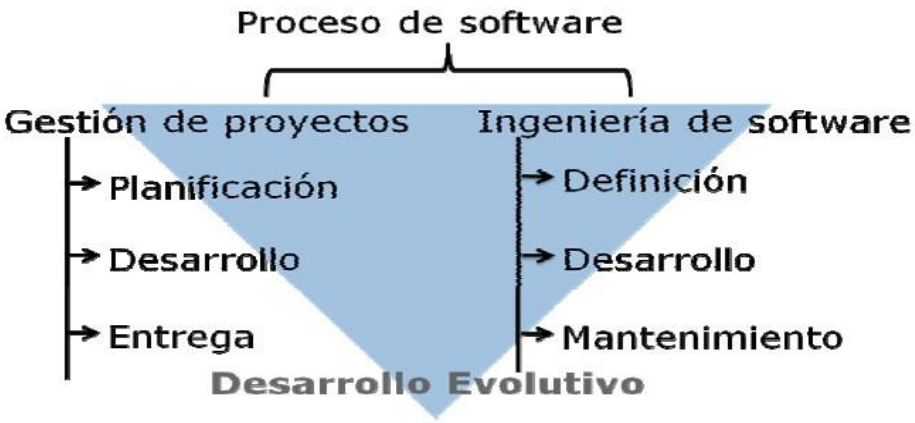
Los métodos ágiles enfatizan las comunicaciones cara a cara en vez de la documentación. La mayoría de los equipos ágiles están localizados en una simple oficina abierta, a veces llamadas "plataformas de lanzamiento" (Bullpen en inglés). La oficina debe incluir revisores, escritores de documentación y ayuda, diseñadores de iteración y directores de proyecto. Los métodos ágiles también enfatizan que el software funcional es la primera medida del progreso. Combinado con la preferencia por las comunicaciones cara a cara, generalmente los métodos ágiles son criticados y tratados como "indisciplinados" por la falta de documentación técnica.

Los métodos robustos se basan en procesos predefinidos con documentación muy precisa, y una detallada planificación inicial que debe seguirse estrictamente. Esta forma de trabajar surgió naturalmente hace unos cincuenta años como una adaptación del manejo de proyectos de ingeniería, que era lo más parecido a desarrollar programas que se conocía en ese momento, y funcionó razonablemente bien en un comienzo. Los métodos robustos eliminan muchos de los problemas que son difíciles de superar con las metodologías habituales. La ambigüedad, lo incompleto y la inconsistencia se descubren y se corrigen más fácilmente, mediante la aplicación del análisis matemático.

1.7.6.1 SXP.

La metodología SXP hace uso de dos metodologías ágiles, que se describirán más adelante, muy utilizadas actualmente en el desarrollo del software. La misma, por lo tanto, constituye un híbrido entre SCRUM y XP incorporando de ambas las mejores prácticas para lograr el objetivo final de desarrollar un software de calidad. Esta metodología divide el Proceso de Software en dos grandes ramas: Gestión de Proyectos e Ingeniería de Software como se muestra en la siguiente figura:

Ilustración 1 Estructura de la Metodología de Desarrollo SXP.



Para el proceso de Gestión de Proyectos se utilizan las técnicas más significativas de SCRUM y para la Ingeniería de Software se usan las características más destacadas de XP. A continuación se ofrece una descripción de ambas metodologías.

SCRUM

Más que una metodología es una herramienta de autogestión del equipo de desarrollo. Los propios programadores deciden qué hacer (tareas) y en qué tiempo lo van a completar. Permite ver claramente el progreso de las tareas por lo que los jefes pueden percibir diariamente que se ha hecho y que falta, es decir, tener una idea general del estado del proyecto acorde a las tareas a cumplir. Se basa en iteraciones o Sprints. Su principio básico es que desde un inicio es muy difícil tener una lista de funcionalidades, que no se vayan a modificar, bien definidas del sistema ya que los usuarios y el propietario de la aplicación van haciendo sucesivas aportaciones. Por esto se plantea el desarrollo de versiones ampliadas, todas ellas usables y puestas a prueba por el usuario final. Además, está orientada a pequeños equipos de desarrollo y plazos de entrega muy cortos

Entre los beneficios que brinda SCRUM están:

- Incluye la gestión de estructuras de datos como listas enlazadas, arrays dinámicos, tablas de búsqueda, árboles binarios, o la gestión de Entrada/Salida asíncrona.
- Aumenta la conformidad del cliente mediante la entrega periódica de resultados tangibles además de integrarlo en el ciclo de desarrollo, permitiendo que el producto final se adecue más a sus necesidades.

- Potencia la formación de equipos autosuficientes y multidisciplinarios, reduciendo la carga de gestión y creando un ambiente propicio para que los programadores desarrollen sus potencialidades al máximo.
- Se centra en el producto y las personas y hace énfasis en la eliminación pre-activa de las trabas e impedimentos que puedan influir en el correcto desarrollo del software. Permitiendo un aumento en la productividad con mayor calidad.
- SCRUM es iterativo e incremental lo cual significa que liberamos código constantemente que representa una mejora ampliada y mejor de la versión anterior del producto.
- La medida del avance del proyecto es un producto que funciona correctamente, no las horas o recursos dedicados a una tarea o grupos de tareas.
- Todo en SCRUM tiene un límite de tiempo. Al cabo de ese límite se entrega lo que esté hecho, no se alargan los plazos de entrega para cumplir con los planes. Se cambia la concepción: "recursos" + "funcionalidades deseadas" = "fecha de entrega" y lo convierte en "recursos" + "fecha de entrega" = "funcionalidades que podremos entregar". Si hay una funcionalidad de la lista que no pudimos implementar no importa pero entregamos.
- El SCRUM Máster no es un jefe tradicional sino un siervo-líder cuya función no es distribuir trabajo ni determinar que funcionalidades se implementan y en qué orden sino cerciorarse de que el proceso de SCRUM siga en marcha y eliminar los impedimentos que pueda tener el equipo de desarrollo en su camino. En SCRUM el equipo decide y el cliente evalúa.

XP: Programación Extrema

Es utilizada en proyectos a corto plazo por equipos pequeños de desarrollo. Es un proceso ligero de bajo riesgo, flexible, predecible, científico y divertido de desarrollar software. Su utilidad se mide en valores como la simplicidad, comunicación, retroalimentación y coraje.

La metodología consiste en una programación rápida o extrema que incluye al usuario final como parte del equipo de desarrollo ya que es uno de los requisitos para llevar a buen término el proyecto. Tiene entregas frecuentes y una refactorización continua lo que nos permite mejorar el diseño cada vez que se le haga una nueva funcionalidad. Sustenta la propiedad colectiva, la programación en parejas y pequeños encuentros diarios donde cada uno de los integrantes del equipo cuenta que ha hecho, que problemas tiene y que hará más adelante.

Lo que propone XP es:

- Empieza en pequeño y añade funcionalidades con retroalimentación continua.

- La gestión de los cambios se convierte en pieza clave del proceso de desarrollo.
- El costo del cambio no depende de la fase o la etapa.
- No se programan funcionalidades que no sean necesarias antes de tiempo.
- El cliente o usuario forma parte del equipo.

Esta metodología ágil centra su éxito en fomentar las buenas relaciones interpersonales en el equipo de desarrollo logrando un mayor trabajo en conjunto y preocupándose por la superación continua de los integrantes del proyecto. Es clave la participación del cliente como parte del proceso de desarrollo, la simplicidad de las soluciones implementadas así como el coraje para enfrentar los cambios.

1.7.7 Lenguajes de programación.

Un lenguaje de programación es un idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado de un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Al proceso por el cual se escribe, se prueba, se depura, se compila y se mantiene el código fuente de un programa informático se le llama programación.

También la palabra programación se define como el proceso de creación de un programa de computadora, mediante la aplicación de procedimientos lógicos.

1.7.7.1 PHP.

PHP (acrónimo de "PHP: Hypertext Preprocessor") es un lenguaje de "código abierto" interpretado, de alto nivel, embebido en páginas HTML y ejecutado en el servidor. Con PHP no se encuentra limitado a resultados en HTML. Entre las habilidades de PHP se incluyen: creación de imágenes, archivos PDF y películas Flash (usando libswf y Ming) sobre la marcha. También puede presentar otros resultados, como XHTM y archivos XML. PHP puede autogenerar estos archivos y almacenarlos en el sistema de archivos en vez de presentarlos en la pantalla. Quizás la característica más potente y destacable de PHP es su soporte para una gran cantidad de bases de datos. Escribir un interfaz vía web para una base de datos es una tarea simple con PHP.

1.7.7.2 CSS.

Hojas de Estilo en Cascada (Cascading Style Sheets), es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser

pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos. (9)

1.7.7.3 JavaScript.

JavaScript es un lenguaje de programación interpretado, es decir, que no requiere compilación, es utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y lenguaje C. Es un lenguaje orientado a objetos, ya que dispone de Herencia, la cual se realiza siguiendo el paradigma de programación basada en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad. JavaScript se ejecuta en el cliente al mismo tiempo que las sentencias van descargándose junto con el código HTML.

1.7.7.4 HTML.

Es un lenguaje estático para el desarrollo de sitios web (*Lenguaje de Marcas Hipertextuales*). Desarrollado por el World Wide Web Consortium (W3C).

HTML es sencillo, permite describir hipertexto, el texto es presentado de forma estructurada y agradable, no necesita de grandes conocimientos cuando se cuenta con un editor de páginas web, sus archivos son pequeños, permite un despliegue rápido, es fácil aprendizaje y lo admiten todos los exploradores.

También podemos decir que por ser un lenguaje estático, no permite la creación de páginas web dinámicas, la interpretación de cada navegador puede ser diferente, guarda muchas etiquetas que pueden convertirse en “basura” y esto dificulta la corrección, además de que las etiquetas son muy limitadas.

1.7.7.5 JavaScript asíncrono y XML (AJAX).

Es una técnica de desarrollo web para crear aplicaciones interactivas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma, es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

Ajax es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores, dado que está basado en estándares abiertos como JavaScript y Modelo de Objetos del Documento (*DOM*).

1.7.8 Sistemas de Control de Versiones (CVS).

Un sistema de control de versiones es un sistema de gestión de archivos y directorios, cuya principal característica es que mantiene la historia de los cambios y modificaciones que se han realizado sobre ellos a lo largo del tiempo. De esta forma, el sistema es capaz de "recordar" las versiones antiguas de los datos, lo que nos permite examinar el histórico de cambios o recuperar versiones anteriores de un fichero, incluso aunque haya sido borrado.

1.7.8.1 Subversión.

Subversión es un software de sistema de control de versiones diseñado específicamente para reemplazar al popular CVS, el cual posee varias deficiencias. Es software libre bajo una licencia de tipo Apache/BSD y se le conoce también como SVN por ser ese el nombre de la herramienta de línea de comandos. Una característica importante de Subversión es que, a diferencia de CVS, los archivos versionados no tienen cada uno un número de revisión independiente. En cambio, todo el repositorio tiene un único número de versión que identifica un estado común de todos los archivos del repositorio en cierto punto del tiempo.

1.7.9 Servidor Web.

Un servidor web es un programa que está diseñado para transferir hipertextos, páginas web o páginas HTML (HyperText Markup Language): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música. El programa implementa el protocolo HTTP (HyperText Transfer Protocol).

El Servidor web se ejecuta continuamente en un ordenador, manteniéndose a la espera de peticiones por parte de un cliente (un navegador web) y que responde a estas peticiones adecuadamente, mediante una página web que se exhibirá en el navegador o mostrando el respectivo mensaje si se detectó algún error.

1.7.9.1 Apache.

Apache es un servidor web de código libre robusto cuya implementación se realiza de forma colaborativa, con prestaciones y funcionalidades equivalentes a las de los servidores comerciales. El

proyecto está dirigido y controlado por un grupo de voluntarios de todo el mundo que, usando Internet y la web para comunicarse, planifican y desarrollan el servidor y la documentación relacionada. Estos voluntarios se conocen como el *Apache Group*. Además del *Apache Group*, cientos de personas han contribuido al proyecto con código, ideas y documentación. (10)

El servidor Apache es un servidor web HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras.

Apache está diseñado para ser un servidor web potente y flexible que pueda funcionar en la más amplia variedad de plataformas y entornos. Las diferentes plataformas y entornos, hacen que a menudo sean necesarias diferentes características o funcionalidades. Apache se ha adaptado siempre a una gran variedad de entornos a través de su diseño modular. Este diseño permite a los administradores de sitios web elegir que características van a ser incluidas en el servidor seleccionando que módulos se van a cargar, ya sea al compilar o al ejecutar el servidor.

1.7.10 Plataforma de desarrollo.

Una plataforma de desarrollo es el entorno de software común en el cual se desenvuelve la programación de un grupo definido de aplicaciones. Comúnmente se encuentra relacionada directamente a un sistema operativo; sin embargo, también es posible encontrarla ligada a una familia de lenguajes de programación o a una Interfaz de programación de aplicaciones (API por sus siglas en inglés). (11)

1.7.10.1 GNU/Linux.

GNU/Linux es un sistema operativo de software libre que cumple las normas POSIX, su base es un núcleo o Kernel monolítico llamado Linux combinado con un grupo de librerías y herramientas. Su estructura general es la típica de cualquier sistema UNIX (núcleo, "intérprete de comandos", aplicaciones). GNU/Linux tiene todas las características que se pueden esperar de un moderno y flexible sistema operativo. Incluye multitarea real, memoria virtual, librerías compartidas, dirección y manejo propio de memoria. Es sin lugar a dudas uno de los ejemplos más prominentes del software libre y del desarrollo del código abierto.

Varias son las distribuciones de GNU/Linux que se han creado a nivel mundial gracias al trabajo constante de los desarrolladores y promotores del Software Libre en el mundo entero. Ejemplo de estas: Debian, Mandriva, Ubuntu, Novel/Suse, Red Hat y Gentoo.

En el desarrollo del módulo se decidió por parte del Departamento de Gestión Universitaria utilizar la plataforma de desarrollo Ubuntu 9.10.

1.8 Fundamentación de las tecnologías y herramientas a utilizar.

Una vez abordado los lenguajes de programación, plataformas, herramientas y metodologías de desarrollo de software, se especifica la selección por parte del Departamento de Gestión Universitaria de la UCI el uso del paradigma de programación orientada a objetos, lenguaje de programación PHP 5, sobre la plataforma GNU/Linux, distribución Ubuntu 9.10, sistema gestor de base de datos Postgresql 8.4.1, Marco de desarrollo CodeIgniter 1.7.1, entorno de desarrollo NetBeans 6.7.1, regido por la metodología ágil SXP.

1.9 Conclusiones parciales.

Durante el desarrollo de este capítulo número uno, se hizo un análisis de las tendencias nacionales e internacionales de lenguajes de programación, di igual forma se mencionaron sistemas existentes en la Universidad de Ciencias Informáticas que posean dentro de su estructura un módulo configuración. Por otra parte, se brinda una descripción de las herramientas, gestores de base de datos, Marco de desarrollo, IDE, Plataformas y metodología para el desarrollo en el módulo.

Teniendo en cuenta las tendencias actuales hacia el software libre el Departamento de Gestión Universitaria de la Universidad de las Ciencias Informáticas ha seleccionado las herramientas antes mencionadas por la flexibilidad, potencia y las variadas posibilidades que ofrecen. De este modo damos por concluido el capítulo número uno (Fundamentación Teórica) de este trabajo de diploma y damos continuidad al capítulo número dos que trata del análisis de la solución propuesta.

CAPÍTULO 2 Análisis y Descripción de la solución propuesta.

2.1 Introducción.

Para el desarrollo de este capítulo realizamos un análisis del documento de análisis y diseño entregado por los analistas del proyecto Sistema de Gestión Universitaria para el módulo de Configuración. Además, se explican los estándares de codificación usados en la implementación. Seguidamente se tratan más a fondo componentes e implementaciones que se usarán en el desarrollo de la aplicación.

2.2 Características del módulo Configuración del subsistema Núcleo del Sistema de Gestión Universitaria.

El módulo Configuración posee como principales características la configuración de una variedad de funcionalidades comunes por otros módulos, funcionalidades tales como gestionar módulos o subsistemas, tipos de entidades, áreas político-administrativas, activar/desactivar funcionalidad, entre otros las cuales estarán dentro del módulo de forma centralizadas y estándar para ser usadas.

2.3 Propuesta de solución.

En el proceso de análisis y diseño los analistas tomaron como metodología a desarrollar Scrum – XP en la misma se generan una serie de artefactos donde a los programadores se le entrega la historia del usuario (HU) en forma de requisitos funcionales en la cual se brinda una descripción del sistema a implementar con una secuencia de pasos lógicos a seguir por el cliente a la hora de interactuar con el sistema. Estas emplean términos del cliente sin lenguaje técnico. Se realiza una por cada característica principal del sistema, estas deben proporcionar sólo el detalle suficiente y difieren de los casos de uso. Cada HU debe contar con las interfaces correspondientes para la solución de las mismas.

Para guiar la implementación del módulo Configuración se han definido las siguientes HU:

1. **Configuración General del Sistema**
   - Gestionar Tipo de Área Político Administrativa
   - Gestionar Área Político Administrativa
2. **Configuración Módulo Estructura y Composición**
   - Configurar Tipos de Entidades

- Configurar Indicadores

3. **Configurar Sistema**
    - Actualizar Subsistema o Módulo
    - Desinstalar Subsistema o Módulo
    - Mostrar Subsistema
    - Mostrar Módulo
    - Editar Funcionalidad

En el diseño propuesto por los analistas se identificaron las funcionalidades a implementar, a través de la descripción detallada de las HU. Además, se toma como base la propuesta de los prototipos de interfaz, ya que cumple con las exigencias del cliente, estos hacen más fácil y cómodo el trabajo y sirven de guía para la implementación de la solución propuesta.

2.4 Descripción de las Historias de Usuarios (HU).

Se mostrará las descripciones de cada HU para la implementación de la solución propuesta.

2.4.1 HU Configuración General del Sistema.

Se recogen las historias de usuario para las funcionalidades de configuración del Sistema, dígase: Gestionar Tipo de Área Político Administrativa y Gestionar Área Político Administrativa que gestiona el módulo Administración para el Sistema de Gestión Universitaria. Se especifica los requisitos de software para las funcionalidades de Configuración del Sistema.

2.4.1.1 Gestionar Tipo de Área Político Administrativa.

El tipo de Área Política Administrativa define como está dividido un país política y administrativamente, en el caso de Cuba tenemos Provincias, Municipios y un Municipio Especial. En la pantalla de Configuración de los Tipos de Área Político Administrativas se mostrará una tabla con los mismos y de ellos se mostrará Nombre, Descripción y las acciones a ejecutar sobre los mismos, desde esta pantalla se podrá acceder a crear Tipos de Área Político Administrativas, y sobre los listados se podrá realizar las siguientes acciones: activar o desactivar, editar o eliminar.

- Para Agregar un nuevo Tipo de Área Político Administrativa se mostrará una pantalla con los siguientes datos a llenar: Nombre, Descripción, Subordinada a, y permitir dejar Activo o no, además se podrá añadir algún atributo nuevo que requiera.

- Para Editar un Tipo de Área Político Administrativa se mostrará una pantalla con los datos modificables: Nombre, Descripción, Subordinada a, agregar o quitar atributos y cambiar a Activo o Desactivar.
- Para Eliminar un Tipo de Área Político Administrativa se mostrará un mensaje de verificación, si este Tipo de Área Político Administrativa está asignado a alguna Área Político Administrativa no se podrá eliminar, por lo que se mostrará un mensaje y si ocurre con éxito se mostrará otro mensaje indicándolo.

2.4.1.2 Gestionar Área Político Administrativa.

El Área Política Administrativa define cuales son las áreas para cada uno de los tipos de Área Política Administrativa en un país, en el caso de Cuba en las provincias tenemos Ciudad Habana, Matanzas, Pinar del Río, etc. Por cada provincia tenemos un conjunto de municipios (Varadero, Cárdenas, etc.) además de La Isla de la Juventud, que es el Municipio Especial. En la pantalla de Configuración de las Áreas Político Administrativas se mostrará una tabla con las mismas y de ellas se mostrará Nombre, Descripción y las acciones a ejecutar sobre los mismos, desde esta pantalla se podrá acceder a Crear Tipos de Área Político Administrativas, y sobre los listados se podrá realizar las siguientes acciones: activar o desactivar, editar o eliminar.

- Para Agregar una nueva Área Político Administrativa se mostrará una pantalla con el Tipo de Área Político Administrativa, según el tipo que se seleccione se mostrarán los atributos asociados a este tipo (Nombre, Abreviatura, Letra que la identifica, Código Estándar, Subordinada a, y permitir dejar Activo o no.
- Para Editar un Área Político Administrativa se mostrará una pantalla con los datos modificables: Tipo de Área Político Administrativa, Nombre, Abreviatura, Letra que la identifica, Código Estándar, Subordinada a, y cambiar a Activo o Desactivar.
- Para Eliminar un Área Político Administrativa se mostrará un mensaje de verificación, si de esta depende alguna otra área no se podrá eliminar, por lo que se mostrará un mensaje y si ocurre con éxito se mostrará otro mensaje indicándolo.

2.4.2 HU Configuración módulo Estructura y Composición.

Se recogen las historias de usuario para las funcionalidades de configuración del Sistema, dígase: Configurar Entidad, Configurar Tipos de Entidades y Configurar Indicadores para el Sistema de Gestión Universitaria. Se especifica los requisitos de software para las funcionalidades de Configuración del Sistema.

2.4.2.1 Configurar Tipos de Entidades.

La Configuración de los Tipos de Entidades debe permitir Crear Tipos de Entidades, Editarlas y Eliminarlas, para esto se mostrará una pantalla con el listado de todos los Tipos de Entidades, posibilidad para Crear Nuevos, y sobre los listados posibilidades para editar y eliminar.

- Para Crear un Tipo de Entidad se deberán introducir los datos: Nombre, Descripción y Seleccionar los Tipos de Entidades a los cuales se podrá subordinar.
- Para Editar un Tipo de Entidad se podrán modificar todos los datos de la misma (Nombre, Descripción y Tipos de Entidades a los cuales se podrá subordinar).
- En mostrar detalles se podrá ver el Nombre, Descripción y los hijos en caso de tener alguno.

2.4.2.2 Configurar Indicadores.

La historia de usuario permite gestionar los indicadores cuantitativos de Promoción, Eficiencia Limpia, Eficiencia Vertical, Índice de Progreso, Índice de Éxito y Matrícula Responsable. Incluye las actividades de Crear Indicador, Mostrar Indicador ubicadas en la barra flotante y las opciones Modificar Indicador y Ver detalles de indicador que se llevan a cabo a partir del Mostrar Indicador. Es objetivo fundamental de esta historia de usuario especificar en el orden correcto los operadores matemáticos con los que se formará el indicador.

- Para Crear Indicador se introducen los campos nombre, descripción, se selecciona la opción de estado activo y se seleccionan los operadores matemáticos que son propios del indicador que se esté creando. Se muestran los botones aceptar y cancelar, si selecciona aceptar se guardan los datos y se muestra un mensaje notificando la acción, si selecciona cancelar se muestra Listar Indicador.
- Para Listar Indicador se muestra un listado con todos los indicadores creados anteriormente con los datos nombre y la descripción como detalle ampliado en forma de globo (comentario). A partir del listado se muestra un conjunto de opciones: Ver Detalles de Indicador y Modificar Indicador.
- Para Modificar un Indicador se selecciona en Mostrar Indicador la opción de editar. Se muestran los datos que se pueden modificar de forma editable. Se muestran los botones guardar cambios y cancelar, si selecciona guardar cambios se guardan los cambios como su nombre lo indica y se muestra un mensaje notificando la acción, si selecciona cancelar se muestra Listar Indicador.

2.4.3 HU Configurar Sistema.

Se recoge las historias de usuario para las funcionalidades de configuración de subsistemas o módulo, dígase: Actualizar, Desinstalar, Mostrar y editar funcionalidad, para el Sistema de Gestión Universitaria. Se especifica los requisitos de software para las funcionalidades de configuración de subsistemas.

2.4.3.1 Actualizar Subsistema o Módulo.

En el proceso de Actualización de un subsistema o módulo este será reemplazado por la nueva actualización, por tanto, se deberá desinstalar la versión anterior e instalar la nueva. Se le mostrará al usuario la opción para que examine y seleccione la ruta donde se encuentra la nueva actualización. Una vez definido esto por el usuario, el sistema deberá reemplazar la carpeta del subsistema o módulo según corresponda. Luego de esto se deberá realizar una ejecución del script de BD y actualizar los ficheros de configuración de la misma.

2.4.3.2 Desinstalar Subsistema o Módulo.

En el proceso de Desinstalación de un subsistema o módulo este será eliminado del sistema, aunque quedará en BD, para volver a utilizar dicho subsistema deberá instalarlo nuevamente mediante la importación del mismo. Se le mostrará al usuario la opción para continuar o cancelar la desinstalación, si el usuario decide continuar se deberá eliminar la carpeta de dicho subsistema/módulo y actualizar los ficheros de configuración de la BD. Si la desinstalación ocurre con éxito se le mostrará un mensaje.

2.4.3.3 Mostrar Subsistema.

En esta pantalla se mostrará al usuario lo referido al subsistema, dígase Nombre, Descripción, si está Activo o no y opción para cambiarlo, y todos los módulos que lo conforman con sus respectivas funcionalidades y posibilidad para la ejecución de acciones sobre estos

2.4.3.4 Mostrar Módulo.

En esta pantalla se mostrará al usuario lo referido al módulo, dígase Nombre, Descripción, y opción para cambiarlo, y todas las funcionalidades que lo conforman y posibilidad para activar o desactivar las mismas.

2.4.3.5 Editar Funcionalidad.

De una Funcionalidad se podrá editar las URL que están asociadas a la misma, a través de esta se controlará el acceso y permisos de los Roles de Usuarios. Se mostrará una pantalla con el Nombre de

la Funcionalidad, Descripción, posibilidad para Activar o Desactivar y un área de texto para introducir todas las URL asociadas a esta funcionalidad.

2.5 Descripción de las nuevas clases u operaciones necesarias.

2.5.1 HU Configuración Módulo Estructura y Composición.

Tabla 1 Clase controladora: Tipo Área Político Administrativa.

Nombre: tipo_area_politico_administrativa	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	index()
Descripción:	Carga la vista de listar los Tipos de Áreas Político Administrativas.
Nombre:	obtenerTipoAreaPoliticoAdministrativa()
Descripción:	Obtiene todas los Tipos de Áreas Político Administrativas que estén creadas y que tengan valor falso del atributo eliminado en la base de datos y los muestra.
Nombre:	activar($tipoArea)
Descripción:	Dado el id de un Tipo de Área, da la opción de activarlo o no.
Nombre:	agregarTPA()
Descripción:	Crea un Tipo de Área Político Administrativa.
Nombre:	eliminarTipodeArea($id_TipoArea)
Descripción:	Dado el id del Tipo Área, eliminarla.
Nombre:	cargarModificar($id)
Descripción:	Dado un id del Tipo de Área, modificarla.

Tabla 2 Clase base: Tipo Área Político Administrativa.

Nombre: tb_ntipo_area_politico_administrativa_base	
Tipo de clase: Base	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	__construct()
Descripción:	Permite que la clase modelo herede las funcionalidades de ella.
Nombre:	obtenerTbNtipoAreaPoliticoAdministrativaPorPagina($inicio=null,$limite=null,$elementoOrdenar=null,$dirOrdenar='asc')
Descripción:	Obtiene todos los Tipos de Áreas.
Nombre:	registrarTbNtipoAreaPoliticoAdministrativa($params=array())
Descripción:	Registra en la base de datos todos los datos del Tipo de Área insertada.
Nombre:	modificarTbNtipoAreaPoliticoAdministrativa($arrIds,$params=array())
Descripción:	Modifica los datos del Tipo de Área dado el id.
Nombre:	obtenerTbNtipoAreaPoliticoAdministrativaDadoIdTbNtipoAreaPoliticoAdministrativa($arrIds)
Descripción:	Obtiene el Tipo de Área dado el id.
Nombre:	obtenerCantidadTbNtipoAreaPoliticoAdministrativa()
Descripción:	Obtiene la cantidad de Tipo de Área.

Tabla 3 Clase controladora: Área Político Administrativa.

Nombre: area_politico_administrativa	

Tipo de clase: Controladora		
Atributo		**Tipo**
Para cada responsabilidad:		
Nombre:	index()	
Descripción:	Carga la vista de listar las Áreas Político Administrativas.	
Nombre:	obtenerAreaPoliticoAdministrativa()	
Descripción:	Obtiene todas las Áreas Político Administrativas que estén creadas y que tengan valor falso del atributo eliminado en la base de datos y los muestra.	
Nombre:	activar($tipoArea)	
Descripción:	Dado el id de un Área, da la opción de activarlo o no.	
Nombre:	crearAreaPoliticoAdministrativa()	
Descripción:	Crea un Área Político Administrativa.	
Nombre:	cargarModificar()	
Descripción:	Modifica el Área que esta seleccionada en ese momento.	
Nombre:	eliminarArea($id_area)	
Descripción:	Dado el id del Área, eliminarla.	

Tabla 4 Clase base: Área Político Administrativa.

Nombre: tb_narea_politico_administrativa_base	
Tipo de clase: Base	
Atributo	**Tipo**

Para cada responsabilidad:	
Nombre:	__construct()
Descripción:	Permite que la clase modelo herede las funcionalidades de ella.
Nombre:	obtenerTbNareaPoliticoAdministrativaPorPagina($inicio=null,$limite=null,$elementoOrdenar=null,$dirOrdenar='asc')
Descripción:	Obtiene todas las Áreas.
Nombre:	registrarTbNareaPoliticoAdministrativa($params=array())
Descripción:	Registra en la base de datos todos los datos del Área insertada.
Nombre:	modificarTbNareaPoliticoAdministrativa($arrIds,$params=array())
Descripción:	Modifica los datos del Área dado el id.
Nombre:	obtenerTbNareaPoliticoAdministrativaDadoIdTbNareaPoliticoAdministrativa($arrIds)
Descripción:	Obtiene el Área dado el id.
Nombre:	obtenerCantidadTbNareaPoliticoAdministrativa()
Descripción:	Obtiene la cantidad de Área.

2.5.2 HU Configurar Sistema.

Tabla 5 Clase controladora: Tipo Entidad.

Nombre: tipo_entidad	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	index()

Descripción:	Carga la vista de listar los Tipos de Entidades.
Nombre:	obtenerTiposEntidades()
Descripción:	Obtiene todos los Tipos de Entidades que estén creadas y los muestra.
Nombre:	crearTipoEntidad()
Descripción:	Crea un Tipo de Entidad.
Nombre:	modificarTipoEntidad()
Descripción:	Modifica la Entidad que esta seleccionada en ese momento.
Nombre:	detalles($idTipoEntidad)
Descripción:	Dado el id del Tipo de Entidad da una descripción de su contenido.
Nombre:	eliminar($idAtributo)
Descripción	Dado un id eliminar el Tipo de Entidad.

Tabla 6 Clase base: Tipo Entidad.

Nombre: tb_ntipo_entidad_base	
Tipo de clase: Base	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	__construct()
Descripción:	Permite que la clase modelo herede las funcionalidades de ella.
Nombre:	obtenerTbNtipoEntidadPorPagina($inicio=null,$limite=null,$elementoOrdenar=null,$dirOrdenar='asc')
Descripción:	Obtiene todas las Entidades.

Nombre:	registrarTbNtipoEntidad($params=array())
Descripción:	Registra en la base de datos todos los datos de la Entidad insertada.
Nombre:	modificarTbNtipoEntidad($arrIds,$params=array())
Descripción:	Modifica los datos de la Entidad dado el id.
Nombre:	obtenerTbNtipoEntidadDadoIdTbNtipoEntidad($arrIds)
Descripción:	Obtiene la Entidad dado el id.
Nombre:	obtenerCantidadTbNtipoEntidad()
Descripción	Obtiene la cantidad de Entidades.

Tabla 7 Clase controladora: Funcionalidad.

Nombre: funcionalidad	
Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	index()
Descripción:	Carga la vista de listar las Funcionalidades.
Nombre:	listarFuncionalidades()
Descripción:	Obtiene todas las Funcionalidades que estén creadas y los muestra.
Nombre:	editar($id_funcionalidad)
Descripción:	Dado un id se edita las URL asociadas a dicha Funcionalidad.

Tabla 8 Clase base: Funcionalidad.

Nombre: tb_dfuncionalidad_base	

Tipo de clase: Base		
Atributo		**Tipo**
Para cada responsabilidad:		
Nombre:	__construct()	
Descripción:	Permite que la clase modelo herede las funcionalidades de ella.	
Nombre:	obtenerTbDfuncionalidadPorPagina($inicio=null,$limite=null,$elementoOrdenar=null,$dirOrdenar='asc')	
Descripción:	Obtiene todas las Funcionalidades.	
Nombre:	modificarTbDfuncionalidad($arrIds,$params=array())	
Descripción	Modifica las URL de la Funcionalidad dado el id.	
Nombre:	obtenerCantidadTbDfuncionalidad()	
Descripción:	Obtiene la cantidad de Funcionalidades.	

2.6 Estilos de programación.

Los estilos de programación tienen gran importancia pues establecen una seria de reglas de la forma en que se escribe el código logrando así un mejor entendimiento de este a la hora de ser modificado o consultado por otros programadores.

Cuando se trabaja sobre un proyecto escrito por otro, es mejor adaptarse al estilo en que está escrito en vez de modificarlo o mezclarlo.

Muchas veces se escribe el código pensando que la única persona que lo modificará es el mismo programador. Y cuando llega alguien más, y comienza a revisar el código, comienzan los problemas. Peor aún es cuando se mezclan estilos de programación.

Se cree que la meta final del programador es construir programas, y no es así lo ideal es construir "buenos" programas. Hay diversas cualidades generalmente aceptadas de lo que es un buen programa por lo que siempre serán bien recibidas y aceptadas cualquier técnica, método o herramienta que nos ayude a mejorar esas cualidades.

Existen diversas cualidades que se ven beneficiadas por el uso de un buen estilo de programación entre las cuales se encuentran:

- **Extensibilidad**: La facilidad con que se adapta el software a cambios de especificación. Un buen estilo de código fomenta programas que no solo resuelven el problema, sino que también reflejan claramente la relación problema/solución. Esto tiene como efecto que muchos cambios simples en el problema reflejen de forma obvia los cambios a hacer en el programa.
- **Verificabilidad**: la facilidad con que pueden comprobarse propiedades de un sistema. Si el estilo de código hace obvia la estructura del programa, eso ayuda a verificar que el comportamiento sea el esperado.
- **Reparabilidad**: la posibilidad de corregir errores sin demasiado esfuerzo.
- **Capacidad de evolución**: la capacidad de adaptarse a nuevas necesidades.
- **Comprensibilidad**: la facilidad con que el programa puede ser comprendido.

Existen diferentes estilos de programación como son: Estilo K&R, Estilo Allman, Estilo BSD KNF, Estilo Whitesmiths y Estilo GNU.

Para el desarrollo del módulo Configuración se decidió utilizar el estilo de programación GNU**,** por generar grandes cualidades al código que lo hace más entendible.

2.6.1 Estilo GNU

El estilo GNU coloca una llave sobre la siguiente línea. Las llaves son indentadas por 2 espacios, y el código que contiene indentada por 2 espacios adicionales.

Ejemplo:

```
function saludar($val)
 {
    if($val == 1)
    {
        echo "HOLA";
    }
    else
    {
        echo "CHAO";
    }
```

```
}
```

2.7 Estándares de codificación.

Un estándar de codificación por lo general son reglas que se siguen para la escritura del código fuente, de tal manera que a otros programadores se les facilite entender el código.

2.7.1 Identación, llaves de apertura y cierre, y tamaño de las líneas.

Usar una identación sin tabulaciones, con un equivalente a 4 espacios, para mantener integridad en las revisiones. El uso de las llaves"{}" será en una nueva línea. La longitud de las líneas de código es aproximadamente de 75-80 caracteres. Para mantener la legibilidad del código.

Ejemplo:

Ilustración 2 Estándares de codificación (Identación).

```
1 ....$a = $b;
2
3 ....function ejemplo()
4 ....{
5     ....//BI
6 ....}
```

2.7.2 Conversión de nomenclatura.

Variables: Se rigen por la nomenclatura camelCase. Siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula.

Ejemplo:

Ilustración 3 Estándares de codificación (Conversión de nomenclatura - Variables).

```
1 ....$variable
2 ....$variableNombreCompuesto
```

Constantes: Siempre debe ser todo en mayúsculas, con caracteres de subrayado "_" para separar palabras en caso de nombres compuestos.

Ejemplo:

Ilustración 4 Estándares de codificación (Conversión de nomenclatura - Constantes).

```
1 ....define(CONSTANTE,valor);
2 ....define(CONSTANTE_COMPUESTO,valor);
```

Clases: siempre comienzan con mayúscula, en caso de nombre compuesto las palabras se separan con el carácter subrayado "_" y el resto en minúscula.

Ejemplo:

Ilustración 5 Estándares de codificación (Conversión de nomenclatura - Clases).

```
1 ....class Clase
2 ....{
3     ....//BI
4 ....}
5
6 ....class Clase_nombre_compuesto
7 ....{
8     ....//BI
9 ....}
```

Funciones: Se rigen por la nomenclatura camelCase. Siempre comienzan con minúscula y en caso de nombres compuestos la primera letra de cada palabra comienza con mayúscula. Los parámetros son separados por espacio luego de la coma que los separa.

Ejemplo:

Ilustración 6 Estándares de codificación (Conversión de nomenclatura - Funciones).

```
1 ....function funcion($parametro1,.$parametro2)
2 ....{
3     ....//BI
4 ....}
5
6 ....function funcionNombreCompuesto($parametro1,.$parametro2)
7 ....{
8     ....//BI
9 ....}
```

Ficheros: Todo siempre en minúscula y en caso de nombres compuestos se usa el carácter subrayado"_".

- **Vistas:** Intuitivo y relacionado con el formulario y/o vista que representa.
- **Modelos:** Con el mismo nombre de la clase que representa que contiene en el nombre el sufijo _mdl o _base en caso de ser modelos base.
- **Librerías:** Con el mismo nombre de la clase que representa que contiene en el nombre el sufijo _lib.
- **Controladoras:** Con el mismo nombre de la clase que representa.
- **Manager:** Con el mismo nombre de la clase que representa que contiene en el nombre el sufijo _mng.

2.7.3 Estructuras de control.

Se incluye if, for, foreach, while, switch, entre las estructuras de control y los paréntesis debe existir un espacio. Se recomienda utilizar siempre llaves de apertura y cierre, incluso en situaciones en las que técnicamente son opcionales. Esto aumenta la legibilidad y disminuye la probabilidad de errores lógicos.

Ejemplos:

Ilustración 7 Estándares de codificación (Estructuras de control).

```
1 ....if.(condicion)
2 ....{
3     ....//BI
4 ....}
5 ....elseif(condicion)
6 ....{
7     ....//BI
8 ...}
9 ....else
```

Si las condiciones son muy largas que sobrepasen el tamaño de la línea, estas se dividen en varias líneas.

Ejemplo:

Ilustración 8 Estándares de codificación (Estructuras de control).

```
1 ....if.(condicion1
2 ....|| condicion2)
3 ....||(condicion3
4 ....&& condicion4))
5 ....{
6     ....//BI
7 ....}
```

En el mejor de los casos cuando la condición es muy extensa, se puede dividir esta en variables y compararlas dentro de la estructura de control.

Ejemplo:

Ilustración 9 Estándares de codificación (Estructuras de control).

```
1 ....$variableCondicion1 = condicion1 || condicion2;
2 ....$variableCondicion2 = condicion3 && condicion4;
3
4 ....if.($variableCondicion1 || $variableCondicion2)
5 ....{
6     ....//BI
7 ....}
```

2.7.4 Documentación

Todos los archivos deben de tener la documentación asociada al mismo. Para esto debe de cumplir con el siguiente bloque al principio de cada clase.

Clase:

Ilustración 10 Estándares de codificación (Documentación - Clases).

```
/**
*Breve descripción de la clase
*
*PHP versión #
*
*@category Categoría de la clase implementada "Libreria,
* Controladora, Modelo"
*@package  Nombre del paquete o módulo al que pertence
*@author  Nombre y Apellidos del autor y  correo electrónico
*/
```

Funciones:

Ilustración 11 Estándares de codificación (Documentación - Funciones).

```
/**
*Breve descripción de la función
*
*@param tipo y nombre del paremetro(por cada parametro que
* recibe la función)
*@return  tipo que retorna
*@author  Nombre y Apellidos del autor y  correo electrónico
*/
```

2.7.5 Buenas prácticas

Los valores booleanos y nulos siempre se escriben con mayúscula, para facilitar la legibilidad del código usar un Enter antes de las estructuras de control y definición de las funciones.

Ilustración 12 Ilustración 12 Estándares de codificación (Buenas prácticas).

```
....$variableBooleana = FALSE;
....$variableNula = NULL;
....
....if(condicion)
....{
    ....//BI
....}
```

2.8 Pautas para la implementación de las HU.

El desarrollo en la metodología utilizada por el Sistema de Gestión Universitaria está guiado por las HU y a su vez el desarrollo de estas HU estará guiado y dirigido por las tareas de ingeniería que se generan de una HU.

Para la implementación de las estas tareas de ingeniería se definen algunos patrones a seguir:

1. Regirse por el estándar de código definido para la implementación del proyecto.
2. Todas las interfaces interactúan mediante Ajax con el servidor.
3. Las acciones se dividen generalmente en dos métodos en las controladoras para lograr una mayor legibilidad del código:
   - Para obtener los datos necesarios en caso de que lo necesite y mostrar la vista. El nombre usado para estos métodos se escribe en infinitivo, ejemplo: crear, listar, modificar, detallar, asociar.
   - Para obtener y validar los datos recibidos desde la vista, interactúa con la librería asociada y crear el mensaje que se envía a la vista. El nombre usado es el mismo infinitivo usado en el otro método con el o los elementos afectado en la acción ejemplo: crear área, modificar área, en el caso de que no sea necesario mostrar una vista, este se escribe en infinitivo ejemplo: eliminar, activar.
4. En los métodos implementados en las clases modelo, las consultas no se hacen con código SQL directamente, se realiza utilizando el active record de CodeIgniter.

   Ejemplo:

   ```
   $this->db->select ('sq_esquema.tb_ddatos.id_datos');
   $this->db->where ($key, $value);
   $this->db->get ('sq_esquema.tb_ddatos') ->result ();
   ```
5. Para la creación de las interfaces se utilizarán las funciones del helper form de CodeIgniter para garantizar homogeneidad en el html, ejemplo: form_open, form_dropdown, form_input.
6. Los mensajes de error en la vista los lanzan con un throw Exception (“mensaje”).
7. No se utiliza el .load de jQuery pues el envío de los datos los hace por get y para garantizar un poco de seguridad se deben de enviar por post siempre, para esto utilizar $.Ajax.
8. En los métodos de las controladoras que serán encuestados mediante Ajax, usar “echo” y no “die” para mostrar los datos o vista cargada, esto garantiza el buen funcionamiento de la programación orientada a aspectos.
9. Todas las implementaciones se realizan en español.

2.9 Tareas de ingeniería.

Las tareas de ingeniería son actividades que los programadores conocen que el sistema debe hacer. Deben ser estimables, y poder ser implementadas entre uno y tres días ideales. La mayoría de estas tareas se derivan directamente de las historias de usuario.

Cada historia de usuario puede ser dividida en varias tareas de ingeniería sencillas. Para determinar las tareas que componen a una historia de usuario se propone hacer una reunión con todos los miembros del equipo de desarrollo y obtener una buena lista con todas.

Cada tarea de ingeniería será comprobada a través de los casos de prueba y no tienen por qué ser comprendidas por el cliente.

2.9.1 Tareas de Ingeniería de la historia de usuario "Configuración General del Sistema".

Tabla 9 Tarea#1: Listar Tipo de Área Político Administrativa.

Tarea	
Número de tarea: 1	**Numero de HU:1**
Nombre de la tarea: Listar Tipo de Área Político Administrativa	
Tipo de tarea: Desarrollo	**Punto estimados:** 3 días
Fecha inicio: 02/05/2010	**Fecha fin:** 04/052010
Descripción: Se muestra en una vista los atributos nombre, descripción, subordinado a y si esta activo de la tabla Tipo de Área Político Administrativa.	

Tabla 10 Tarea#2: Crear Tipo de Área Político Administrativa.

Tarea	
Número de tarea: 2	**Numero de HU:1**
Nombre de la tarea: Crear Tipo de Área Político Administrativa	
Tipo de tarea: Desarrollo	**Punto estimados:** 3 días
Fecha inicio: 06/05/2010	**Fecha fin:** 08/052010
Descripción: Crear un formulario con los campos a llenar, donde se especifica el nombre,	

subordinado, descripción, si estará activo o no y atributos del tipo de área.

Tabla 11 Tarea#3: Modificar Tipo de Área Político Administrativa.

Tarea	
Número de tarea: 3	**Numero de HU:1**
Nombre de la tarea: Modificar Tipo de Área Político Administrativa	
Tipo de tarea: Desarrollo	**Punto estimados:**3 días
Fecha inicio: 09/05/2010	**Fecha fin:** 11/052010
Descripción: Se crea un formulario con los datos a modificar del tipo de área seleccionada y si los datos son correctos se procede a modificarlos.	

Tabla 12 Tarea#4: Eliminar Tipo de Área Político Administrativa.

Tarea	
Número de tarea: 4	**Numero de HU:1**
Nombre de la tarea: Eliminar Tipo de Área Político Administrativa	
Tipo de tarea: Desarrollo	**Punto estimados:**3 días
Fecha inicio: 12/05/2010	**Fecha fin:** 14/052010
Descripción: Se procede a desarrollar el código para eliminar un Tipo de Área.	

Tabla 13 Tarea#5: Listar Área Político Administrativa.

Tarea	
Número de tarea: 5	**Numero de HU:1**
Nombre de la tarea: Listar Área Político Administrativa	
Tipo de tarea: Desarrollo	**Punto estimados:** 3 días

Fecha inicio: 15/05/2010	**Fecha fin:** 17/052010
Descripción: Se muestra en una vista los atributos nombre, descripción, subordinado a y si esta activo, de la tabla Área Político Administrativa.	

Tabla 14 Tarea#6: Crear Área Político Administrativa.

Tarea	
Número de tarea: 6	**Numero de HU:1**
Nombre de la tarea: Crear Área Político Administrativa	
Tipo de tarea: Desarrollo	**Punto estimados:** 3 días
Fecha inicio: 18/05/2010	**Fecha fin:** 20/052010
Descripción: Crear un formulario con los campos a llenar, donde se especifica el nombre, subordinado, descripción y si estará activo o no, del área.	

Tabla 15 Tarea#7: Modificar Área Político Administrativa.

Tarea	
Número de tarea: 7	**Numero de HU:1**
Nombre de la tarea: Modificar Tipo de Área Político Administrativa	
Tipo de tarea: Desarrollo	**Punto estimados:**3 días
Fecha inicio: 21/05/2010	**Fecha fin:** 23/052010
Descripción: Se muestra en un formulario con los datos a modificar de la tabla Área Político Administrativa.	

Tabla 16 Tarea#8: Eliminar Área Político Administrativa.

Tarea	
Número de tarea: 8	**Numero de HU:1**

Nombre de la tarea: Eliminar Tipo de Área Político Administrativa	
Tipo de tarea: Desarrollo	**Punto estimados:**3 días
Fecha inicio: 24/05/2010	**Fecha fin:** 26/052010
Descripción: Crear el código que procede a eliminar el Área Político Administrativa.	

Nota:

Las tareas de usuarios correspondientes a las historias de usuario **Configuración Módulo Estructura y Composición,** junto con las de **Configurar Sistema** están presentes en el Anexo 1.

2.10 Patrones de Arquitectura.

Son aquéllos que expresan un esquema organizativo estructural fundamental para sistemas de software.

2.10.1 Patrón Modelo Vista Controlador (MVC).

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. Su principal finalidad es mejorar la reusabilidad y que las modificaciones en las vistas impacten en menor medida en la lógica de negocio o de datos.

El **Modelo** es el responsable de:

- Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
- Define las reglas de negocio (la funcionalidad del sistema).
- Lleva un registro de las vistas y controladores del sistema.
- Si estamos ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo.

El **Controlador** es responsable de:

- Recibir los eventos de entrada.
- Contiene reglas de gestión de eventos, estas acciones pueden suponer peticiones al modelo o a las vistas.

Las **Vistas** son responsables de:

- Recibir datos del modelo y los muestra al usuario.

- Tienen un registro de su controlador asociado.
- Pueden dar el servicio de "Actualización", para que sea invocado por el controlador o por el modelo cuando es un modelo activo.

¿Por qué MVC?

Porque es un patrón que convierte una aplicación en un paquete mantenible, modular y de desarrollo rápido. La modularidad y el diseño independiente permiten a los desarrolladores y diseñadores hacer cambios en alguna parte de la aplicación sin afectar a los demás.

2.11 Patrones de diseño.

Los patrones del diseño tratan los problemas del diseño que se repiten y que se presentan en situaciones particulares del diseño, con el fin de proponer soluciones a ellas. Por lo tanto, los patrones de diseño son soluciones exitosas a problemas comunes. Existen muchas formas de implementar patrones de diseño. Los detalles de las implementaciones son llamadas estrategias.

Los patrones de diseño son útiles ya que pueden acelerar el proceso de desarrollo. La reutilización de patrones de diseño ayuda a evitar problemas sutiles que pueden causar grandes problemas y mejora la legibilidad del código para los programadores y arquitectos familiarizados con los patrones.

No es aconsejable forzar el uso de patrones ni tampoco abusar del uso dado ya que puede ser un gran error. No es obligatorio su uso, solo es aconsejable en el caso de tener el mismo problema o similar que soluciona el patrón.

Para el diseño del módulo configuración el departamento de gestión universitaria acordó usar los patrones de asignación de responsabilidades (GRASP) y los patrones GOF (Gang-Of-Four) que a continuación se describen:

Experto: Es el principio básico de asignación de responsabilidades. Indica, por ejemplo, que la responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo. De este modo se obtiene un diseño con mayor cohesión y así la información se mantiene encapsulada.

Creador: El patrón creador ayuda a identificar quién debe ser el responsable de la creación (o instanciación) de nuevos objetos o clases.

Controlador: El patrón controlador es un patrón que sirve como intermediario entre una determinada interfaz y el algoritmo que la implementa, de tal forma que es la que recibe los datos del usuario y la que los envía a las distintas clases según el método llamado.

Alta cohesión: Nos dice que la información que almacena una clase debe de ser coherente y debe estar (en la medida de lo posible) relacionada con la clase.

Bajo acoplamiento: Es la idea de tener las clases lo menos ligadas entre sí que se pueda. De tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases.

Indirección: El patrón de indirección nos aporta mejorar el bajo acoplamiento entre dos clases asignando la responsabilidad de la mediación entre ellos a un tercer elemento (clase) intermedio.

Patrón Solitario (Singleton): Garantiza que una clase sólo tenga una única instancia y proporcionar un punto de acceso global a ella. Reduce el espacio de nombres, es una mejora sobre las variables globales.

2.12 Análisis de algoritmos.

El análisis de algoritmos estudia, desde el punto de vista teórico, los recursos computacionales que necesita la ejecución de un programa de ordenador: Su eficiencia.

¿Por qué estudiar la eficiencia de los algoritmos?

Porque nos sirve para establecer la frontera entre lo factible y lo imposible.

Cálculo de eficiencia de algoritmo.

La complejidad de procedimientos que llaman a otros procedimientos, pero no de forma recursiva, se calcula de la siguiente forma:

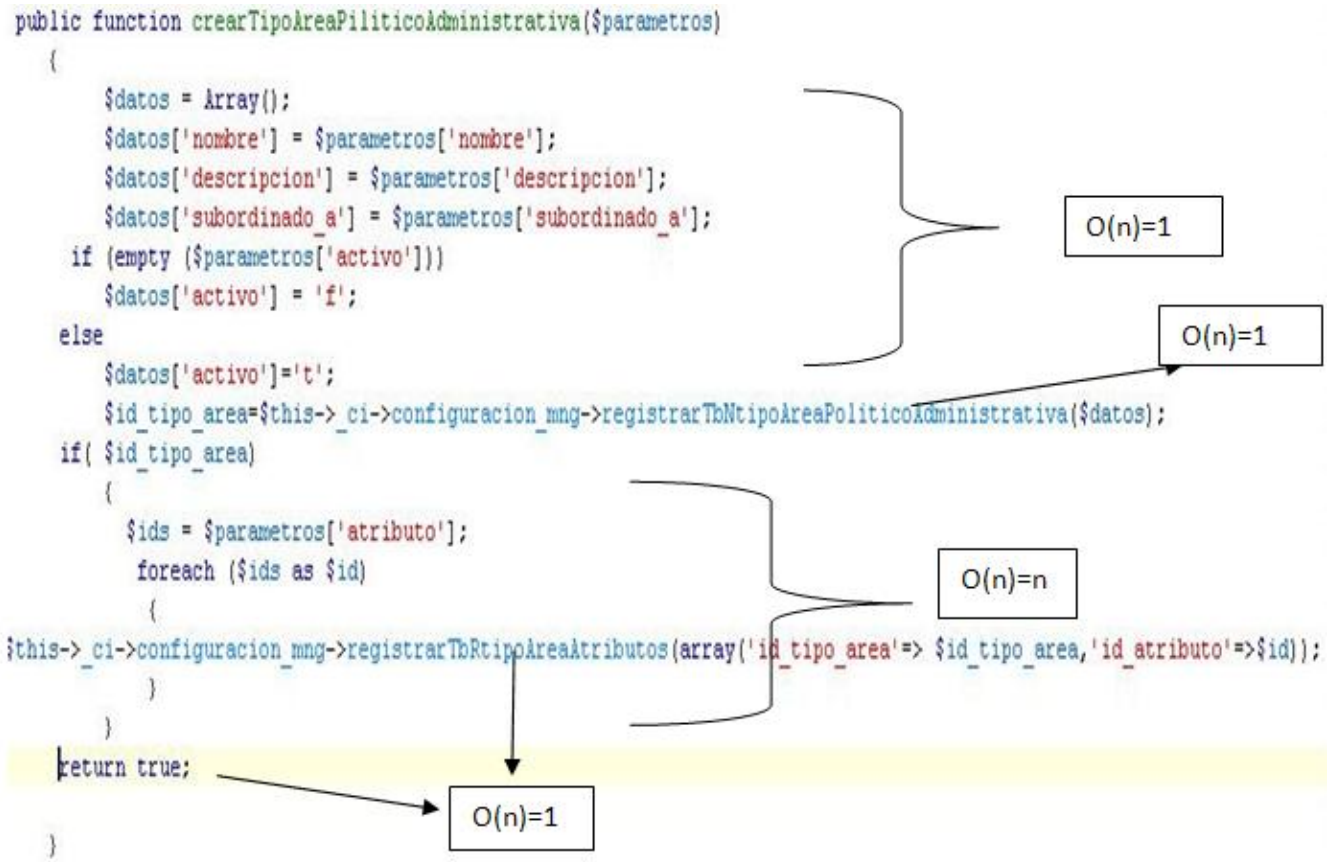
- Se determina la complejidad de los procedimientos que no llaman a otros.
- Se determina la complejidad de los procedimientos en que todos los procedimientos que él activa tienen calculada su complejidad.

Nota:

En el desarrollo del módulo Configuración no fue necesario el uso de algoritmos recursivos.

Para realizar el análisis de eficiencia se escogió el método **crearTipoAreaPoliticoAdministrativa** de la HU Gestionar Tipo Área.

Ilustración 13: Análisis de algoritmo (crearTipoAreaPoliticoAdministrativa).



El método **registrarTbNtipoAreaPoliticoAdministrativa** tiene una complejidad **O(n)=**1

El método **registrarTbRtipoAreaAtributos** tiene una complejidad **O(n)=**1

$$T(n)=\begin{cases}1 & n=1\\ n & n>1\end{cases}$$

T (1)=1

T (n)=n

O (n)=n

C1=O(n)

C1=n.

Para medir la eficiencia de un algoritmo se tiene la siguiente tabla

Tabla 17 Complejidad algorítmica.

Cuota superior	Descripción	Orden Eficiencia
$O(1)$	Orden constante	1
$O(\log n)$	Orden logarítmico	2
$O(n)$	Orden lineal	3
$O(n \log n)$	Orden cuasi-lineal	4
$O(n^2)$	Orden cuadrático	5
$O(n^3)$	Orden cúbico	6
$O(2^n)$	Orden exponencial	7

Entre menor es el orden de eficiencia mayor es la eficiencia del algoritmo por lo que se puede concluir que el método expuesto para el análisis es eficiente.

2.13 Tarjetas CRC.

Las tarjetas CRC son técnicas de modelado creadas para ayudar a los desarrolladores de software a crear diseños de clases orientados a responsabilidades. Dichas tarjetas constan de tres secciones, nombre de la clase, responsabilidades y colaboradores. La sección responsabilidades se listan cada una de las funciones o tareas que debe ser capaz cumplir un objeto de dicha clase mientras que la sección colaboradora contiene otras clases del diseño.

Tabla 18 Tarjeta CRC (Módulo Configuración).

Configuración	
Funcionalidades.	**Colaboraciones.**
_construct() index() listar() obtenerTipoAreaPoliticoAdministrativa() activar($tipoArea)	MY_Controller

agregarTPA() crearTipoAreaPoliticoAdministrativa() cargarModificar() modificar() eliminarTipoArea($id) obtenerTiposEntidades() crearTipoEntidad() modificarTipoEntidad() detalles($idTipoEntidad) obtenerSubordinado() obtenerAreaPoliticoAdministrativa() crearAreaPoliticoAdministrativa() eliminarArea() listarFuncionalidades() editar($id_funcionalidad) editarFuncionalidades() obtenerIndicadores() frmRegistro() frmModificar() registrarIndicador() eliminarIndicador($id) modificarIndicador() detalles() exportarModSubS() cargarExportar($llave)	

2.14 Integración del módulo Configuración con el subsistema Núcleo y el Sistema de Gestión Universitaria.

El Sistema de Gestión Universitaria tiene como principal objetivo integrar todas las áreas de procesos dentro del sistema, pero para lograrlo es necesario que el módulo configuración este integrado al subsistema Núcleo y al sistema en general, para esta integración se utiliza el servicio **IoC** (Inversión de Control) el cual es un archivo XML en el que están todas las funcionalidades creadas dentro del módulo y que otros métodos necesitan. También están los servicios Web para cuando los módulos no estén en la misma aplicación y necesiten utilizar funcionalidades del módulo Configuración, ya que el

IoC que tiene programado el sistema no permite la comunicación entre aplicaciones, solo entre módulos de una misma aplicación.

Un ejemplo conciso de esto es como interactúa con el módulo Personal y Secretaria durante los procesos de prematrícula, matrícula y ratificación de matrícula al utilizar las áreas políticos administrativas y los tipos de áreas políticos administrativas.

2.15 Distribución del sistema.

El diagrama de despliegue permite conocer la ubicación física de los nodos, así como los protocolos de conexión entre ellos.

Ilustración 14 Diagrama de despliegue

2.16 Conclusiones Parciales.

Durante el desarrollo de este capítulo se mostró las principales Historias de Usuarios definidas por el analista del departamento, de las cuales se hizo una descripción de cada una de ellas también las principales clases u operaciones necesarias para el desarrollo del módulo Configuración. Se crearon las tareas de Ingeniería para las Historias de Usuarios correspondientes a estas y se demostró las buenas prácticas de programación a través de los estilos y estándares utilizados. También se realizó el análisis de algoritmos para comprobar la eficiencia del código utilizado, además se hizo un estudio de los patrones de diseño y arquitectura para un mejor desempeño en el desarrollo del módulo. De este modo damos por concluido el capítulo número dos Análisis y Descripción de la solución propuesta de este trabajo de diploma y damos continuidad al capítulo número tres y final que trata de validar la solución propuesta.

CAPÍTULO 3 Validación de la Solución Propuesta.

3.1 Introducción.

Durante este capítulo se mostrarán los casos de pruebas que define la metodología de desarrollo por historia de usuarios para evaluar que la aplicación web este completamente funcional, así también se mostrarán a través de tablas los resultados obtenidos durante la realización de las pruebas de aceptación.

3.2 Pruebas de software.

Las pruebas de software, son los procesos que permiten verificar y revelar la calidad de un producto software. Son utilizadas para identificar posibles fallos de implementación, calidad, o usabilidad de un programa de ordenador o videojuego. Básicamente es una fase en el desarrollo de software consistente en probar las aplicaciones construidas. Las pruebas de software se integran dentro de las diferentes fases del ciclo del software dentro de la Ingeniería de software. Para determinar el nivel de calidad se deben efectuar unas medidas o pruebas que permitan comprobar el grado de cumplimiento respecto de las especificaciones iniciales del sistema.

Dentro de las pruebas de software que genera la metodología SXP se encuentran las pruebas unitarias y las de aceptación.

Pruebas unitarias: Uno de los métodos más utilizados para realizar pruebas a nuestro software es el llamado Prueba Unitaria (unit testing). La base de este método es hacer pruebas en pequeños fragmentos del programa. Estos fragmentos deben ser unidades estructurales encargados de una tarea específica, en programación orientada a objetos se puede afirmar que estas unidades son los métodos o las funciones que se tienen definidos.

Tras realizar estas pruebas sobre los elementos unitarios de un programa se puede eliminar gran parte de los errores de los que podría adolecer. Cualquier prueba demuestra no la ausencia de errores sino que revela la presencia de ellos. Las pruebas unitarias no revelan errores en la integración de las partes unitarias ni tampoco otros problemas como el bajo rendimiento de las aplicaciones o problemas derivados del sistema sobre el que está ejecutándose un programa.

El objetivo de las pruebas unitarias es el aislamiento de partes del código y la demostración de que estas partes no contienen errores.

Una vez creados el conjunto de pruebas unitarias sobre un fragmento de código los beneficios obtenidos, incluso antes de ejecutar ninguna otra prueba, son múltiples. Ejemplo:

Simplificación de la integración: Las pruebas unitarias eliminan las posibles incertidumbres y errores en lo que se espera de cada una de las unidades, ayudando a entender la integración de cada una de las partes.

Refactorización de código: Una vez refactorizado el código; las mismas pruebas unitarias pueden servir para probar el nuevo código asegurando que este sigue siendo válido bajo la nueva implementación.

Documentación: Las pruebas unitarias sirven como método de documentación mismo. Los desarrolladores pueden ver a través de las pruebas unitarias cual es el objetivo de las distintas partes del código de una manera básica.

Diseño: Cuando se desarrolla el software las pruebas unitarias pueden tomar el lugar del diseño formal. Cada prueba unitaria puede ser vista como un elemento de diseños que especifica las clases, los métodos y el comportamiento observable de la aplicación.

Prueba de aceptación: Son especificaciones de comportamiento y funcionalidad deseados para un sistema. Ellos nos informan cómo, para una determinada historia de usuario o caso de uso, el sistema trata determinadas condiciones y entradas. Una buena prueba de aceptación debe ser:

- Propiedad de los clientes
- Escrito en conjunto con los clientes, desarrolladores y analistas de prueba
- Sobre el Qué y no sobre el Cómo
- Expresada en lenguaje de dominio del problema
- Conciso, preciso y sin ambigüedades

El objetivo de las pruebas de aceptación es validar que un sistema cumple con el funcionamiento esperado y permitir al usuario de dicho sistema que determine su aceptación, desde el punto de vista de su funcionalidad y rendimiento.

3.3 Prueba de aceptación.

Las pruebas de aceptación se elaboran a lo largo de la iteración, en paralelo con el desarrollo del sistema, y adaptándose a los cambios que el sistema sufra. Son más importantes que las pruebas unitarias dado que significan la satisfacción del cliente con el producto desarrollado al final de una iteración. Por esta razón se decidió utilizar este tipo de prueba para garantizar el buen funcionamiento del módulo Configuración y así la satisfacción del cliente con el desarrollo del mismo.

3.3.1 Casos de prueba (CP) de aceptación.

Condiciones Generales de Ejecución:

1. El usuario debe haber sido autenticado en el sistema.

3.3.1.1 CP: Gestionar Tipo de Área Político Administrativa.

Escenario “Crear Tipo Área Político Administrativa”

Tabla 19 Prueba de aceptación: Crear Tipo Área Político Administrativa.

Clases Válidas	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
Introducción correcta del Tipo de Área Político Administrativa. **“Tipo de Área: Provincia”**	1. Tras la introducción de todos los datos de un Tipo Área Político Administrativa (Anexo 2 ilustración 15) , si el procesado ha sido correcto, en la tabla de la base de datos correspondiente a este aparecerán los datos del nuevo Tipo Área Político Administrativa y en la base de datos tipo área atributos estará el o los id de atributo que se le introdujeron al Tipo de Área junto con el id del Tipo de Área. 2. Se muestra un mensaje que indica que la acción ha sido realizada satisfactoriamente. (Anexo 2 ilustración 16).	El resultado de la prueba fue exitoso cumpliéndose el resultado esperado.	

Clases Inválidas	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
Introducción de campos obligatorios de un Tipo de Área sin llenar. **"Tipo de Área: Nombre"**	1. Se muestra un mensaje indicando que no se han introducidos todos los datos en los campos obligatorios (Anexo 2 ilustración 17). 2. El Tipo de Área no es introducido en la base de datos.	El resultado de la prueba fue exitoso cumpliéndose el resultado esperado.	
Evaluación de la Prueba:	Satisfactoria		

Escenario "Editar Tipo Área Político Administrativa "

Tabla 20 Prueba de aceptación: Editar Tipo Área Político Administrativa.

Clases Válidas	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
Editar correctamente el Tipo Área Político Administrativa. **"Tipo de área: Provincia"**	1. Tras la modificación de todos o algunos de los datos de un Tipo de Área (Anexo 2 ilustración 18), si el procesado ha sido correcto, en la tabla de la base de datos correspondiente a este aparecerán	El resultado de la prueba fue exitosa cumpliéndose el resultado esperado	

	los nuevos datos del Tipo Área Político Administrativa y en la base de datos tipo área atributos estará, en caso de ver modificado los atributos, el o los id de los atributos modificados con el id del Tipo de Área. 2. Se muestra un mensaje que indica que la acción ha sido realizada satisfactoriamente (Anexo 2 ilustración 19).		
Clases Inválidas	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
Introducción de campos obligatorios sin llenar de un Tipo de Área **"Tipo de Área: Nombre"**	1. Se muestra un mensaje indicando que no se han introducidos todos los datos en los campos obligatorios (Anexo 2 ilustración 20). 2. El Tipo de Área no es modificado en la base de datos.	El resultado de la prueba fue exitoso cumpliéndose el resultado esperado.	
Evaluación de la Prueba:	Satisfactoria		

Escenario "Eliminar Tipo Área Político Administrativa"

Tabla 21 Prueba de aceptación: Eliminar Tipo Área Político Administrativa.

Clases Válidas	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
Eliminar el Tipo Área Político Administrativa. **"Tipo de Área: Provincia"**	1. Se muestra un mensaje de verificación para eliminar el Tipo de Área (Anexo 2 ilustración 21). 2. Se muestra un mensaje que indica que la acción ha sido realizada satisfactoriamente (Anexo 2 ilustración 22).	El resultado de la prueba fue exitoso cumpliéndose el resultado esperado.	
Clases Inválidas	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
Eliminar el Tipo Área Político Administrativa con un área asignado. **"Tipo de Área: Provincia"**	1. Se muestra un mensaje indicando que no se puede eliminar el Tipo de Área (Anexo 2 ilustración 23). 2. El Tipo de Área no es eliminado por lo que no es modificado en la base de datos.	El resultado de la prueba fue exitoso cumpliéndose el resultado esperado.	
Evaluación de la Prueba:	Satisfactoria		

3.3.1.2 CP: Gestionar Área Político Administrativa.

Condiciones de Ejecución:

1. El usuario debe haber creado los Tipos de Área Político Administrativa a la que va a ser asignada el Área Político Administrativa.

Escenario "Crear Área Político Administrativa"

Tabla 22 Prueba de aceptación: Crear Área Político Administrativa.

Clases Válidas	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
Introducción correcta del Área Político Administrativa. **"Área: Municipio"**	1. Tras la introducción de todos los datos de un Área (Anexo 2 ilustración 24) , si el procesado ha sido correcto, en la tabla de la base de datos correspondiente a este aparecerán los datos del nuevo Área Político Administrativa. 2. Se muestra un mensaje que indica que la acción ha sido realizada satisfactoriamente (Anexo 2 ilustración 25).	El resultado de la prueba fue exitoso cumpliéndose el resultado esperado.	
Clases Inválidas	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**

Introducción de campos obligatorios sin llenar de un Área. **"Área: Nombre"**	1. Se muestra un mensaje indicando que no se han introducidos todos los datos en los campos obligatorios (Anexo 2 ilustración 26). 2. El Tipo de Área no es introducido en la base de datos.	El resultado de la prueba fue exitoso cumpliéndose el resultado esperado.	
Evaluación de la Prueba:	Satisfactoria		

Escenario "Editar Área Político Administrativa "

Tabla 23 Prueba de aceptación: Editar Área Político Administrativa.

Clases Válidas	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
Editar correctamente el Área Político Administrativa. **"Área: Municipio"**	1. Tras la modificación de todos o algunos de los datos de un Área (Anexo 2 ilustración 27), si el procesado ha sido correcto, en la tabla de la base de datos correspondiente a este aparecerán los nuevos datos del Área Político Administrativa.	El resultado de la prueba fue exitoso cumpliéndose el resultado esperado.	

	2. Se muestra un mensaje que indica que la acción ha sido realizada satisfactoriamente (Anexo 2 ilustración 28).		
Clases Inválidas	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
Introducción de campos obligatorios de un Área Político Administrativa sin llenar. **"Área: Nombre"**	1. Se muestra un mensaje indicando que no se han introducidos todos los datos en los campos obligatorios (Anexo ilustración 29). 2. El Tipo de Área no es modificado en la base de datos.	El resultado de la prueba fue exitoso cumpliéndose el resultado esperado.	
Evaluación de la Prueba:	Satisfactoria		

Escenario "Eliminar Área Político Administrativa"

Tabla 24 Prueba de aceptación: Eliminar Área Político Administrativa.

Clases Válidas	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
Eliminar Área Político Administrativa. **"Área: Municipio"**	1. Se muestra un mensaje de verificación para eliminar el Área (Anexo 2 ilustración 30). 2. Se muestra un	El resultado de la prueba fue exitoso cumpliéndose el resultado esperado.	

	mensaje que indica que la acción ha sido realizada satisfactoriamente (Anexo 2 ilustración 31).		
Clases Inválidas	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
Eliminar el Área Político Administrativa con alguna otra Área asignado. **"Área: Localidad"**	1. Se muestra un mensaje indicando que no se puede eliminar el Área (Anexo 2 ilustración 32). 2. El Área no es eliminado por lo que no es modificado en la base de datos.	El resultado de la prueba fue exitoso cumpliéndose el resultado esperado.	
Evaluación de la Prueba:	Satisfactoria		

3.3.1.3 CP: Configurar Tipos de Entidades.

Escenario "Crear Tipo de Entidad"

Tabla 25 Prueba de aceptación: Crear Tipo de Entidad.

Clases Válidas	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
Introducción correcta del Tipo de Entidad. **"Tipo de Entidad: UM"**	1. Tras la introducción de todos los datos de un Tipo de Entidad (Anexo 2 ilustración 33), si el procesado ha sido correcto, en la tabla de la base de datos correspondiente a este	El resultado de la prueba fue exitoso cumpliéndose el resultado esperado.	

	aparecerán los datos del nuevo Tipo de Entidad. 2. Se muestra un mensaje que indica que la acción ha sido realizada satisfactoriamente (Anexo 2 ilustración 34).		
Clases Inválidas	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
Introducción de campos obligatorios de un Tipo de Entidad sin llenar. **"Tipo de Entidad : Nombre"**	1. Se muestra un Mensaje de error indicando el campo obligatorio (Anexo 2 ilustración 35). 2. El Tipo de Entidad no es introducido en la base de datos.	El resultado de la prueba fue exitoso cumpliéndose el resultado esperado.	
Evaluación de la Prueba:	Satisfactoria		

Escenario "Editar Tipo de Entidad"

Tabla 26 Prueba de aceptación: Editar Tipo de Entidad.

Clases Válidas	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
Editar correctamente	1 Tras la	El resultado de la prueba	

del Tipo de Entidad. **"Tipo de Entidad: UM"**	modificación de todos o algunos de los datos del Tipo de Entidad (Anexo 2 ilustración 36), si el procesado ha sido correcto, en la tabla de la base de datos correspondiente a este aparecerán los nuevos datos del Tipo de Entidad. 2 Se muestra un mensaje que indica que la acción ha sido realizada satisfactoriamente (Anexo 2 ilustración 37).	fue exitoso cumpliéndose el resultado esperado.	
Clases Inválidas	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
Introducción de campos obligatorios de un Tipo de Entidad sin llenar. **"Tipo de Entidad : Nombre"**	1. Se muestra un Mensaje de error indicando el campo Obligatorio (Anexo 2 ilustración 38). 2. El Tipo de Entidad no es modificado en la base de datos.	El resultado de la prueba fue exitoso cumpliéndose el resultado esperado.	
Evaluación de la Prueba:	Satisfactoria		

Escenario “Eliminar Tipo de Entidad”

Tabla 27 Prueba de aceptación: Eliminar Tipo de Entidad.

Clases Válidas	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
Eliminar el Tipo de Entidad. ¨**Tipo de Entidad: UM**¨	1. Se muestra un mensaje de verificación para eliminar el Tipo de Entidad 2. Se muestra un mensaje que indica que la acción ha sido realizada satisfactoriamente.	El resultado de la prueba fue exitoso cumpliéndose el resultado esperado.	
Clases Inválidas	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
Eliminar un Tipo de Entidad que está asignada a alguna Entidad. **“Tipo de Entidad : Nombre”**	1. Se muestra un mensaje indicando que no se puede eliminar el Tipo de Entidad. 2. El Tipo de Entidad no es eliminado por lo que no es modificado en la base de datos.	El resultado de la prueba fue exitoso cumpliéndose el resultado esperado.	
Evaluación de la Prueba:	Satisfactoria		

3.3.1.4 Configurar Indicadores.

Escenario “Crear Indicadores”

Tabla 28 Prueba de aceptación: Crear Indicadores.

Clases Válidas	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
Introducción correcta de los datos. **“Indicador: Indicador 1”**	1. Al introducirse los datos correctos en los campos (Anexo 2 ilustración 39) se crea el nuevo Indicador y los datos son introducidos en la base de datos. 2. Se muestra un mensaje que indica que la acción ha sido realizada satisfactoriamente (Anexo 2 ilustración 40).	El resultado de la prueba fue exitoso cumpliéndose el resultado esperado.	
Clases Inválidas	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
Introducción incorrecta de los datos. **“Indicador : Nombre”**	1. Se muestra un mensaje de error indicando que hay campos vacíos, y señala el o los campos (Anexo 2 ilustración 41). 2. El Indicador no es creado y no se introducen los datos en base de datos.	El resultado de la prueba fue exitoso cumpliéndose el resultado esperado.	
Evaluación de la Prueba:	Satisfactoria		

Escenario “Modificar Indicadores”

Tabla 29 Prueba de aceptación: Modificar Indicadores.

Clases Válidas	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
Introducción correcta de los datos a modificar. **“Indicador: Indicador 1”**	1. Al introducirse los datos correctos en los campos, (Anexo 2 ilustración 42) se modifican los nuevos datos del Indicador y son cambiados en la base de datos. 2. Se muestra un mensaje que indica que la acción ha sido realizada satisfactoriamente (Anexo 2 ilustración 43).	El resultado de la prueba fue exitoso cumpliéndose el resultado esperado.	
Clases Inválidas	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
Introducción incorrecta de los datos. **“Indicador : Nombre”**	1. Se muestra un mensaje de error indicando que hay campos vacíos, y señala el o los campos (Anexo 2 ilustración 44). 2. El Indicador no es modificado y no se introducen los nuevos datos en base de datos.	El resultado de la prueba fue exitoso cumpliéndose el resultado esperado.	

Evaluación de la Prueba:	Satisfactoria

3.3.1.5 Actualizar Subsistema o Módulo.

Escenario “Actualizar Subsistema o Módulo”

Tabla 30 Prueba de aceptación: Actualizar Subsistema o Módulo.

Clases Válidas	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
Actualizar Subsistema o Módulo. ¨ Subsistema o Módulo **:Hola.xml**¨	Al actualizar el subsistema correctamente se muestra un mensaje donde indica que la acción se ha realizado satisfactoriamente.	El resultado de la prueba fue exitoso cumpliéndose el resultado esperado.	
Clases Inválidas	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
Actualizar incorrecta del Subsistema o Módulo. **“**Subsistema o Módulo **:Hola.html”**	Al no actualizarse correctamente ya sea que un subsistema no es el correcto o por dejar el campo en blanco, se muestra un mensaje de error.	El resultado de la prueba fue exitoso cumpliéndose el resultado esperado.	
Evaluación de la Prueba:	Satisfactoria		

3.3.1.6 Editar Funcionalidad.

Escenario “Editar Funcionalidad”

Tabla 31 Prueba de aceptación: Editar Funcionalidad.

Clases Válidas	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
Editar Funcionalidad **“**Funcionalidad **:Hola.uci.cu”**	Al editarse una Funcionalidad correctamente, se muestra un mensaje que indica que la acción ha sido realizada satisfactoriamente.	El resultado de la prueba fue exitoso cumpliéndose el resultado esperado.	
Clases Inválidas	**Resultado Esperado**	**Resultado de la Prueba**	**Observaciones**
Editar incorrectamente la Funcionalidad **“**Funcionalidad **:Hola”**	Al editarse incorrectamente la funcionalidad, dejando el campo del nombre vacío, se muestra un mensaje indicando el error.	El resultado de la prueba fue exitoso cumpliéndose el resultado esperado.	
Evaluación de la Prueba:	Satisfactoria		

3.4 Conclusiones parciales.

En el desarrollo de este capítulo se verifica el correcto funcionamiento del módulo a través de los casos de pruebas de aceptación por escenario y las pruebas de caja blanca. Queda plasmado el desarrollo del módulo propuesto obteniéndose resultados favorables, logrando centralizar y estandarizar los procesos de configuración para las áreas de proceso que estarán incluidas dentro del Sistema de Gestión Universitaria.

Conclusiones.

- Se mejora los procesos de configuración de las áreas de proceso del Sistema de Gestión Universitaria, a través de la centralización y estandarización del módulo Configuración.
- El empleo de los métodos teóricos y empíricos proporcionaron el estado actual del objeto de estudio.
- Las tareas de investigación brindaron una mejor organización durante la investigación del trabajo.
- La realización de la prueba de aceptación permitió comprobar el buen funcionamiento y eficiencia del módulo a partir de los requisitos existentes en las historias de usuarios para la implementación del módulo.
- El objetivo del trabajo fue cumplido, contribuyendo al desarrollo del Sistema de Gestión Universitaria.

A modo de conclusión se puede decir que los beneficios que brinda el desarrollo del módulo Configuración son los siguientes:

- Las áreas de proceso dentro del Sistema de Gestión Universitaria contarán con un módulo configuración dentro del subsistema Núcleo, que les facilitará el trabajo a la hora de realizar cambios así como adaptar o modificar funcionalidades que sean necesarias para cada una de estas áreas.
- Cualquier sistema informático que se rija por la arquitectura propuesta por el Departamento de Gestión Universitaria para el desarrollo de las aplicaciones, podrá utilizar dicho módulo Configuración, para gestionar sus procesos de configuración, ahorrándose tiempo en la implementación de un módulo propio.
- El Sistema de Gestión Universitaria contará con un módulo Configuración en el subsistema Núcleo que servirá de forma estándar y centralizada para todas las áreas de proceso que el sistema posea.

Recomendaciones.

Una vez terminado el objetivo de la investigación se recomienda que se continúe con el desarrollo del módulo, implementando las funcionalidades como Gestionar entidad, Importar módulo o subsistemas, configurar base de datos y otras funcionalidades que son objetivos de próximas iteraciones, para lograr un completo y mejor funcionamiento del módulo Configuración.

Bibliografía Consultada.

1. **Alexander Oré B.** CalidadySoftware.com. [En línea] [Citado el: 1 de Junio de 2010.] http://www.calidadysoftware.com/testing/pruebas_unitarias1.php.
2. **Alvarez, Sara.** desarrolloweb.com. [En línea] [Citado el: 11 de Abril de 2010.] http://www.desarrolloweb.com/articulos/2477.php.
3. atenas.cult.cu. [En línea] [Citado el: 16 de Abril de 2010.] http://www.atenas.cult.cu/rl/informatica/manuales/sl/introduccion_al_SL/sec-ide.html.
4. autorneto.com. [En línea] [Citado el: 11 de Abril de 2010.] http://autorneto.com/tecnologia/software/tecnicas-de-programacion/.
5. babylon. [En línea] [Citado el: 8 de Mayo de 2010.] http://www.babylon.com/definition/plataforma_de_desarrollo/Spanish.
6. chuidiang.com. [En línea] [Citado el: 19 de Abril de 2010.] http://www.chuidiang.com/ood/metodologia/scrum.php.
7. **Correa, Leonardo A.** desarrolloweb.com. [En línea] [Citado el: 15 de Abril de 2010.] http://www.desarrolloweb.com/articulos/aplicaciones-web-usabilidad-practica-exito.html.
8. desarrolloweb.com. [En línea] [Citado el: 17 de Abril de 2010.] http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html.
9. desarrolloweb.com. [En línea] [Citado el: 17 de Abril de 2010.] http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html.
10. desarrolloweb.com. [En línea] [Citado el: 14 de Abril de 2010.] http://www.desarrolloweb.com/articulos/aplicaciones-web-usabilidad-practica-exito.html.
11. Efectos JavaScript. [En línea] [Citado el: 25 de Abril de 2010.] http://www.efectosjavascript.com/javascript.html.
12. Enciclopedia. [En línea] [Citado el: 20 de Abril de 2010.] http://enciclopedia.us.es/index.php/Lenguaje_de_programaci%C3%B3n.
13. **Gerardo Fernández Escribano. 2002.** info-ab.uclm.es. [En línea] 9 de 12 de 2002. [Citado el: 20 de Abril de 2010.] http://www.info-ab.uclm.es/asignaturas/42551/trabajosAnteriores/Presentacion-XP.pdf.
14. GNU Operatign System. [En línea] [Citado el: 14 de Abril de 2010.] http://www.gnu.org/philosophy/free-sw.es.html.

15. GNU Operating System. [En línea] [Citado el: 8 de Mayo de 2010.] http://www.gnu.org/home.es.html.
16. GNU Operating System. [En línea] [Citado el: 12 de Abril de 2010.] http://www.gnu.org/philosophy/free-sw.es.html. 1.
17. Ivanex. [En línea] [Citado el: 5 de junio de 2010.] http://ivanex.wikidot.com/patron-arquitectura.
18. Java. [En línea] [Citado el: 29 de Mayo de 2010.] http://java.ciberaula.com/articulo/diseno_patrones_j2ee/.
19. jordisan.net. [En línea] [Citado el: 15 de Abril de 2010.] http://jordisan.net/blog/2006/que-es-un-framework.
20. jQuery. [En línea] [Citado el: 16 de Abril de 2010.] http://docs.jquery.com/.
21. La tecla de Escape. [En línea] [Citado el: 19 de Abril de 2010.] http://latecladeescape.com/w0/ingenieria-del-software/metodologias-de-desarrollo-del-software.html.
22. masadelante. [En línea] [Citado el: 4 de Mayo de 2010.] http://www.masadelante.com/faqs/servidor.
23. masadelante. [En línea] [Citado el: 21 de Abril de 2010.] http://www.masadelante.com/faqs/php.
24. masadelante. [En línea] [Citado el: 25 de Abril de 2010.] http://www.masadelante.com/faqs/ajax.
25. Mastermagazine. [En línea] [Citado el: 25 de Abril de 2010.] http://www.mastermagazine.info/termino/5286.php.
26. omitsis.com. [En línea] [Citado el: 19 de Abril de 2010.] http://www.omitsis.com/scrum-como-metodologia-de-desarrollo.
27. PostgreSQL. [En línea] [Citado el: 18 de Abril de 2010.] http://www.postgresql.org/about/.
28. **Ramirez, Ignacio. 2002.** [En línea] 15 de 10 de 2002. [Citado el: 28 de Abril de 2010.] http://iie.fing.edu.uy/~nacho/blandos/archivos/cvs.pdf.
29. Scribd. [En línea] [Citado el: 18 de Abril de 2010.] http://www.scribd.com/doc/2050925/metodologias-de-desarrollo-software.
30. solcre.com. [En línea] [Citado el: 15 de Abril de 2010.] http://www.solcre.com/files/ventajas_de_las_aplicaciones_web.pdf.
31. solcre.com. [En línea] [Citado el: 15 de Abril de 2010.] http://www.solcre.com/files/ventajas_de_las_aplicaciones_web.pdf.

32. solcre.com. [En línea] [Citado el: 15 de Abril de 2010.] http://www.solcre.com/files/ventajas_de_las_aplicaciones_web.pdf.
33. Soluciones Informáticas. [En línea] [Citado el: 4 de Mayo de 2010.] http://www.xolucionesinformaticas.com/index.php?option=com_content&view=article&id=77&Itemid=85.
34. Soluciones Informáticas. [En línea] [Citado el: 4 de Mayo de 2010.] http://www.xolucionesinformaticas.com/index.php?option=com_content&view=article&id=77&Itemid=85.
35. **Stallman, Richard.** gnu. [En línea] [Citado el: 8 de Mayo de 2010.] http://www.gnu.org/gnu/linux-and-gnu.es.html.
36. teleformacion.edu.aytolacoruna.es. *teleformacion.edu.aytolacoruna.es.* [En línea] http://teleformacion.edu.aytolacoruna.es/PASCAL/document/modular.htm#intro
37. W3C. [En línea] [Citado el: 21 de Abril de 2010.] http://www.w3c.es/divulgacion/guiasbreves/hojasestilo.
38. W3C. [En línea] [Citado el: 21 de Abril de 2010.] http://www.w3c.es/divulgacion/guiasbreves/hojasestilo.
39. Zona Linux. [En línea] [Citado el: 16 de Abril de 2010.] http://zonalinux.com.ar/codeigniter-un-poderoso-framework-open-source/.
40. Zona Linux. [En línea] [Citado el: 16 de Abril de 2010.] http://zonalinux.com.ar/codeigniter-un-poderoso-framework-open-source/.

Referencias Bibliográficas.

1. Scribd. [En línea] [Citado el: 9 de Mayo de 2010.] http://www.scribd.com/doc/2851433/La-programacion-logica-y-funcional.

2. Scribd. [En línea] [Citado el: 9 de Mayo de 2010.] http://www.scribd.com/doc/2851433/La-programacion-logica-y-funcional.

3. Scribd. [En línea] [Citado el: 9 de Mayo de 2012.] http://www.scribd.com/doc/2851433/La-programacion-logica-y-funcional.

4. GNU Operating System. [En línea] [Citado el: 10 de Mayo de 2012.] http://www.gnu.org/philosophy/free-sw.es.html.

5. Solcre. [En línea] [Citado el: 10 de Mayo de 2012.] http://www.solcre.com/files/ventajas_de_las_aplicaciones_web.pdf.

6. Solcre. [En línea] [Citado el: 9 de Mayo de 2012.] http://www.solcre.com/files/ventajas_de_las_aplicaciones_web.pdf.

7. Zona Linux. [En línea] [Citado el: 11 de Mayo de 2012.] http://zonalinux.com.ar/codeigniter-un-poderoso-framework-open-source/.

8. desarrolloweb. [En línea] [Citado el: 12 de Mayo de 2012.] http://www.desarrolloweb.com/articulos/sistemas-gestores-bases-datos.html.

9. w3c. [En línea] [Citado el: 14 de Mayo de 2012.] http://www.w3c.es/divulgacion/guiasbreves/hojasestilo.

10. Soluciones Informáticas. [En línea] [Citado el: 15 de Mayo de 2012.] http://www.xolucionesinformaticas.com/index.php?option=com_content&view=article&id=77&Itemid=85.

11. babylon. [En línea] [Citado el: 15 de Mayo de 2012.] http://www.babylon.com/definition/plataforma_de_desarrollo/Spanish.

Glosario de Términos.

- **AJAX:** Acrónimo de Asynchronous *JavaScript And XML** (en inglés JavaScript y XML asíncronos). Sirve para crear aplicaciones más interactivas y no necesita recargar completamente la Web ya que utiliza: HTML con hojas de estilos (CSS) para mostrar la Web, Document Object Model (DOM*) y JavaScript interactuando más dinámicamente con los datos XML y XSLT* sirviendo para intercambiar y manipular los datos.
- **CSS:** Son las siglas de *Cascading Style Sheets*, que significa Hojas de Estilo en Cascada. Es un lenguaje que describe la presentación de los documentos estructurados en hojas de estilo para diferentes métodos de interpretación. El lenguaje CSS se basa en una serie de reglas que rigen el estilo de los elementos en los documentos estructurados, y que forman la sintaxis de las hojas de estilo.
- **Javascript:** Es un lenguaje de programación que permite a los desarrolladores crear acciones en sus páginas Web. Permite crear diferentes efectos e interactuar con los usuarios. Gran parte de la programación en este lenguaje está centrada en describir objetos, escribir funciones que respondan a movimientos del mouse, aperturas, utilización de teclas, cargas de páginas, entre otros. Al ser un lenguaje del lado del cliente, este es interpretado por el navegador.
- **JQuery:** Es un nuevo tipo de biblioteca o Marco de desarrollo de JavaScript que permite simplificar la manera de interactuar con los documentos HTML, permitiendo manejar eventos, desarrollar animaciones y agregar interacción con la tecnología AJAX al sistema.
- **DOM** (*Document Object Model*): Es una representación interna estándar de la estructura de un documento, y proporciona una interfaz al programador (API) para poder acceder de forma fácil, consistente y homogénea a sus elementos, atributos y estilo. Es un modelo independiente de la plataforma y del lenguaje de programación.