UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 9

Implementación del Módulo Gestión del Aprendizaje del Centro Virtual de Autoaprendizaje de Lenguas Extranjeras.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN INFORMÁTICA

Autor: Arturo Edel Regueiro García

Tutor: Msc. Yaneisis Pérez Heredia

Co-Tutor: Ing. Reinier Castillo González

Ciudad de la Habana, 2009
"Aniversario 50 del Triunfo de la Revolución"

DEDICATORIA

- A nuestro **Comandante Fidel Castro**, por haber sido el creador, de este Proyecto Futuro, donde he tenido la oportunidad de graduarme como Ingeniero en Ciencias Informáticas.

- A mis padres y mi hermana, por estar presentes en todo momento y por el apoyo que me han brindado durante toda mi vida.

- Al resto de mi familia por haber contribuido de una forma u otra en mi formación

- A todos mis compañeros por estar siempre presentes y darme su apoyo.

- A todos las personas que confiaron en mí y que de una forma u otro contribuyeron a que obtuviera este resultado.

AGRADECIMIENTOS

- A toda mi familia por su apoyo y preocupación constante.
- A Rolando Toledo Fernández por trasmitirme sus conocimientos y por ayudarme en los momentos más difíciles durante estos meses.
- A mi tutora Yaneisis Pérez Heredia por ser la mejor tutora que uno pueda tener.
- A Pimienta por su ayuda y por crear en mi ese hábito de estudio con sus consejos.
- A mi co-tutor por los buenos consejos y la ayuda que siempre me dio.
- A Yoandris S. Pacheco Jerez por toda su ayuda que me sirvió de mucho.
- A mi novia por su paciencia durante toda esta etapa.
- A mis **amigos**, que siempre confiaron en mí y me brindaron su apoyo constante en todo momento durante estos 5 años.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de la presente tesis y reconozco 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 _____.

_____ Arturo Edel Regueiro García
[**Autor**]

_____ Yaneisis Pérez Heredia
[**Tutora**]

La inteligencia consiste no sólo en el conocimiento, sino también en la destreza de aplicar los conocimientos en la práctica.

Aristóteles

"Lo que sabemos es una gota de agua; lo que ignoramos es el océano."

Isaac Newton

OPINIÓN DEL TUTOR

RESUMEN

En Cuba la educación es una esfera que a lo largo de los años ha sufrido cambios con el objetivo de mejorar muchos aspectos que son considerados de gran importancia en el proceso de enseñanza. La necesidad de buscar nuevas formas de enseñanza a dado paso a la aparición de nuevos métodos como el autoaprendizaje, para lo cual han surgidos recursos como los espacios virtuales, aulas virtuales, que tienen como complemento la realización de laboratorios, seminarios, áreas de investigación, salas de encuentro y discusión.

La Universidad de las Ciencias Informáticas (UCI) es un centro que está a la vanguardia de todo el proceso de enseñanza y autoaprendizaje es por ello que surge el proyecto CEVALE conocido como (Centro Virtual de Auto-aprendizaje de Lenguas Extranjeras) con el cual se prevé crear un ambiente alternativo de aprendizaje, que les permite a los estudiantes trabajar de manera independiente en sus debilidades y profundizar en los temas que son de su interés. Un entorno virtual de aprendizaje que facilite el proceso de gestión del aprendizaje mediante un proceso de adaptación a los estilos de aprendizaje de los estudiantes y que sea capaz de generar posibles rutas de aprendizaje para los distintos niveles y fases del aprendizaje de lenguas extranjeras. En el siguiente trabajo se propone la implementación del entrono virtual de aprendizaje para lo cual es usado Symfony como framework, como lenguaje de programación empleado php 5 y como Entorno Integrado de Desarrollo (IDE) Eclipse. Además el Sistema Gestor de Bases de Datos que se utilizó es PostgreSQL.

TABLAS Y FIGURAS

ÍNDICE

INTRODUCCIÓN

En la actualidad Cuba está inmersa en un proceso de informatización y desarrollo el cual requiere que cada uno de los profesionales que participan en él sean personas que cuenten con una preparación integral, lo cual los obliga a llevar a cabo un proceso de superación, con el objetivo de que estos tengan una mejor formación y así puedan desarrollar sus habilidades y ampliar su marco de conocimientos contando con la posibilidad del intercambio con los distintos ámbitos sociales.

El conocer lenguas extranjeras es uno de los principales requisitos en este proceso de formación que es realizado por cada uno de estos profesionales, pues su estudio está determinado por necesidades económicas, políticas y sociales, y a la vez está condicionado por las posibilidades objetivas y subjetivas existentes, las cuales incluyen el poder realizar intercambios culturales y profesionales con distintas instituciones internacionales que a su vez pueden ser de gran ayuda para fomentar el desarrollo tanto económico como intelectual a través de la formación de lazos de amistad y cooperación entre los diferentes sectores.

La Universidad de Ciencias Informáticas (UCI) es una institución que no está exenta en todo este proceso, ya que juega un papel primordial en el ámbito del desarrollo e informatización del país. Por lo que la preparación de cada uno de los que forman parte de su comunidad es considerado como una tarea primordial. Es por ello que hoy en día se incluye en la vanguardia del proceso de enseñanza-aprendizaje donde van incluido las lenguas extranjeras, para lo cual tiene como una de las tareas más urgentes en la actualidad la creación de métodos y sistemas que ayuden a solucionar el problema de encontrar formas óptimas y regímenes de trabajo en el aprendizaje de idioma, es decir, resolver el problema de la efectividad y la intensificación en el aprendizaje. Por lo tanto para lograr el dominio de la lengua extranjera en la comunicación es necesario intensificar el proceso de enseñanza, elevar el coeficiente de productividad y efectividad tanto en clases como fuera de ellas.

La diversidad y complejidad del conocimiento actual exige que los planes de estudio de las universidades incluyendo la UCI estén constantemente evolucionando a formas superiores de enseñanza donde reviste especial atención la actividad cognoscitiva de los estudiantes y su habilidad para ampliar independientemente sus conocimientos. Es por ello que una de las formas de reforzar este proceso es el empleo de los medios y tecnologías creados con este fin.

Tal es el caso de los Centros de Auto-aprendizaje y Servicios de Idiomas Extranjeros (CASIE), entendida este como un sistema que se combina en dos modalidades, el espacio físico que lo constituyen los diferentes CASIE creados en la Universidad de las Ciencias Informáticas y donde se promueve la autogestión del aprendizaje. El espacio virtual conocido como Centro de Autoaprendizaje Virtual de Idiomas Extranjeros (CAVIE), el cual es utilizado como complemento de los CASIE y se puede acceder a través de la plataforma de teleformación [1].

El proyecto CEVALE conocido como (Centro Virtual de Auto-aprendizaje de Lenguas Extranjeras) se comienza a gestar en la Universidad de las Ciencias Informática (UCI), a raíz del re-diseño de la estrategia curricular de la disciplina de Idioma Inglés, expresándose en el área de la Vice-rectoría de Formación, en el Departamento Docente Central de Idiomas Extranjeros. Con la puesta en marcha de este proyecto se prevé crear un ambiente alternativo de aprendizaje, que les permite a los estudiantes trabajar de manera independiente en sus debilidades y profundizar en los temas que son de su interés. Un entorno virtual de aprendizaje que facilite el proceso de gestión del aprendizaje mediante un proceso de adaptación a los estilos de aprendizaje de los estudiantes [1].

Por lo que, se hace necesario precisar el siguiente **problema a resolver**: La no existencia de una aplicación que permita la gestión del aprendizaje de idiomas extranjeros de manera automática en los CASIE de la UCI.

Para darle solución al problema planteado se define como **objeto de estudio:** Gestión del aprendizaje.

El **campo de acción:** Sistemas de gestión de aprendizaje.

A partir del problema referenciado anteriormente se puede enunciar la siguiente **idea a defender:** Con un sistema que automatice la gestión del aprendizaje de idiomas extranjeros en los CASIES de la UCI se logrará una mejor gestión del aprendizaje personal.

El **objetivo general** de esta investigación está dirigido a: Implementar el módulo de Gestión del aprendizaje para los CASIE en la UCI.

Del objetivo general mencionado anteriormente se derivan los siguientes **objetivos específicos:**

1. Actualizar el conocimiento en torno a la gestión del aprendizaje de idiomas extranjeros en la tesis de análisis y diseño y del modelo planteado en la tesis de maestría del CASIE.
2. Valorar las tendencias actuales que existen en cuanto a la gestión del aprendizaje y la determinación de estas para el desarrollo de la aplicación que se va a implementar.
3. Implementar el módulo de gestión del aprendizaje.

Para darle cumplimiento a los objetivos trazados se ha decidido desarrollar las siguientes **Tareas de la investigación:**

1. Actualizar el conocimiento en torno a la gestión del aprendizaje de idiomas extranjeros.
2. Evaluar el contenido de la información obtenida sobre la gestión de aprendizaje del idioma inglés a partir del modelo planteado en una tesis doctoral.
3. Diseñar la interfaz de la aplicación a desarrollar.
4. Determinar los estándares de codificación.
5. Realizar los diagramas correspondientes a la implementación.
6. Implementar módulos de gestión del aprendizaje.
7. Desarrollo de Casos de Prueba que certifiquen la veracidad de los algoritmos empleados.

Se utilizaron como métodos de investigación científica:

Métodos Teóricos:

- **Histórico lógico:** Este método es de suma importancia ya que permitirá realizar un análisis de cómo ha evolucionado la gestión del aprendizaje durante los años, en los distintos momentos sujetos a las distintas corrientes del pensamiento o desarrollo científico técnico. Para de esta forma poder obtener una tendencia de cómo se debe comportar en la actualidad y así actualizar el conocimiento en torno ha dicho proceso centrándonos en la gestión del aprendizaje de idiomas extranjeros.

- **Modelación:** Este método permitirá tener en cuenta todas y cada una de las relaciones que se establecen dentro del proceso de enseñanza-aprendizaje, así como la creación de diferentes modelos que permitirán obtener una perspectiva de la realidad investigada.

- **Analítico-Sintético:** Este método permitirá después de realizar un análisis de las tendencias actuales en torno a la investigación para la gestión del aprendizaje, determinar los principales métodos y algoritmos que son usados a nivel mundial y que nos pueden servir en la implementación del modulo.

Métodos Empíricos:

- **Entrevista a especialistas del Departamento de Idiomas:** Con este método se han realizado entrevistas no formales para lograr obtener la información necesaria respecto a las características que debe tener el sistema.

- **Encuesta:** Con este método se ha aplicado una encuesta a los profesores del departamento de Idiomas Extranjeros para definir las características y facilidades que debe tener el sistema de gestión del aprendizaje.

Posibles resultados:

- Prototipo funcional del módulo de gestión del aprendizaje.

El contenido del trabajo se encuentra distribuido en el documento de la siguiente manera:

Capítulo 1. Fundamentación teórica:

Se realiza un análisis valorativo del estado del arte correspondiente a las técnicas de programación y lenguajes usados en el mundo, en Cuba y en la UCI. Se analizan y explican las técnicas de programación, y lenguajes usados para la implementación del sistema, así como algunas de las herramientas empleadas tales como el sistema gestor de base de datos, el framework sobre el cual se realizó la aplicación, el tipo de arquitectura con el cual se trabajo y el IDE empleado para la programación.

Capítulo 2. Descripción y análisis de la solución propuesta:

En este capítulo se realiza una valoración del diseño que propone el analista. Se describen algunos algoritmos a implementar y se analiza la complejidad de los mismos, así como las clases que modelan la solución y las principales funcionalidades de cada una de ellas. Finalmente se establecen los estándares de codificación a utilizar.

Capítulo 3. Validación de la solución propuesta:

En este capítulo se aborda el tema referente a la automatización de pruebas, se analizan algunas de las características que presentan las pruebas unitarias y funcionales que son empleadas para la validación de la solución propuesta y se propone el empleo de una herramienta con la cual se ejecuto un caso de prueba lo que permitió que se comprobaran los resultados obtenidos para de esta forma llegar a conclusiones sobre lo logrado en el trabajo.

CAPÍTULO 1: Fundamentación Teórica.

1.1 Introducción

En este capítulo se abordan las características de las diferentes técnicas y lenguajes de programación tanto web como de escritorio, utilizados actualmente en el mundo y en la UCI. Se analiza el paradigma de programación orientado a objetos y los principales lenguajes exponentes del mismo así como las plataformas que los soportan. Se exponen algunas de las herramientas de desarrollo que más se ajustan a los propósitos de ese trabajo tales como el framework, gestor de base de datos, servidor web e IDE empleados para el desarrollo del mismo. Finalmente se realiza un estudio crítico y valorativo de las librerías usadas en la programación de la herramienta.

1.2 Técnicas de Programación

Con el tiempo, los investigadores en ciencias de la computación observaron que, más allá del propósito para el que fueron creados, los lenguajes podían diferenciarse por la forma de trabajo que presentan al programador, ofreciendo diversas formas de ver y pensar un programa antes de escribirlo. Así comenzaron a surgir distintas técnicas de programación**,** cada una de ellas representada por una familia de lenguajes. A continuación se hace referencia a cada una de estas técnicas.

1.2.1 Programación no Estructurada.

Cuando las personas empiezan a aprender, a programar, lo hacen escribiendo programas pequeños y sencillos consistentes en un solo programa principal. En la programación no estructurada se denomina "programa principal" a una secuencia de comandos o instrucciones que modifican datos que son a su vez globales en el transcurso de todo el programa. Este tipo de programación consiste en secuencias de instrucciones donde los saltos y el fin de programa no siguen ninguna estructura. Los saltos pueden apuntar a cualquier punto del código lo que ocasiona que el algoritmo termine siendo un ovillo indescifrable. Tampoco se puede saber cuándo termina. Este tipo de programación persiste en lenguajes como el Assembler [7].

Esta técnica de programación tiene como desventaja que una vez que el programa se hace suficientemente grande, si es necesario emplear la misma secuencia de instrucciones en diferentes situaciones dentro del programa, la secuencia debe ser repetida. Esto ha conducido a la idea de extraer estas secuencias, darles un nombre y ofrecer una técnica para llamarlas y regresar desde estos procedimientos.

Este tipo de programación se caracteriza por ser secuencias de instrucciones donde los saltos y el fin de programa no seguían ninguna estructura. Los saltos podían apuntar a cualquier punto del código lo que ocasionaba que el algoritmo terminara siendo un ovillo indescifrable. Uno de los problemas principales de este tipo de programación es la sentencia, **goto** que permite saltar a cualquier línea del programa, por ejemplo, si existe una línea 40 (que dice **if a<300 then goto 70**) se puede traducir como "si la variable **a** tiene un valor mayor a 300 entonces hay que saltar a la línea 70), es decir si el programa contiene algunas sentencias **goto** esto complica bastante la lectura del algoritmo por muy simple que este sea. También es muy difícil determinar cuándo terminará el programa, si bien la sentencia **end,** por ejemplo se encuentra en la línea 120, no podemos saber cuándo es que el algoritmo llegará (y si llegará realmente) a esa línea [7].

1.2.2 Programación Procedimental

La programación procedimental permite combinar las secuencias de instrucciones repetibles en un solo lugar. Una *llamada de procedimiento* se utiliza para invocar al procedimiento. Después de que la secuencia es procesada, el flujo de control procede exactamente después de la posición donde la llamada fue hecha.

Con el empleo de esta tecnología al introducir *parámetros,* así como procedimientos de procedimientos (*subprocedimientos*) los programas pueden ser escritos en forma más estructurada y libres de errores.

De este modo, un programa puede ser visto como una secuencia de llamadas a procedimientos. El programa principal es responsable de pasar los datos a las llamadas individuales, los datos son procesados por los procedimientos y una vez que el programa ha terminado, los datos resultantes son presentados. El programa principal coordina las llamadas a procedimientos y pasa los datos apropiados en forma de parámetros como muestra la Ilustración 1.

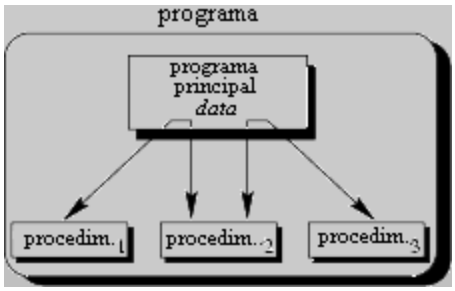

Ilustración 1 Funcionamiento de la Programación Procedimental.

Con esta metodología el programa se divide en partes independientes, cada una de las cuales ejecuta una única actividad o tarea y se codifican independientemente de los demás. Cada una de ellas se analiza, codifica y ponen a punto por separado. Siguiendo un método ascendente o descendente de desarrollo se llegará a la descomposición final del problema en módulos en forma jerárquica. Las descomposiciones resultantes reciben luego el refinamiento progresivo del repertorio de instrucciones que van a formar parte de cada pieza del programa [17].

Algunas de las desventajas que presenta este tipo de programación es su rigidez e inflexibilidad de los programas, la pérdida excesiva de tiempo en la corrección de errores, documentación deficiente e insuficiente, incluso nula además de la imposibilidad de reutilizar el programa o fragmentos suyos en proyectos futuros.

1.2.3 Programación Modular

La programación modular consiste en dividir un programa en módulos ó subprogramas con el fin de hacerlo más legible y manejable. En la programación modular, los procedimientos con una funcionalidad común son agrupados en módulos separados. Un programa por consiguiente ya no consiste solamente de una sección, ahora está dividido en varias secciones más pequeñas que interactúan a través de llamadas a procedimientos y que integran el programa en su totalidad.

Cada módulo puede contener sus propios datos. Esto permite que cada módulo maneje un estado interno que es modificado por las llamadas a procedimientos de ese módulo. Sin embargo, solamente hay un estado por módulo y cada módulo existe cuando más una vez en todo el programa. Además puede ser que algunos necesiten 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.

El programa principal coordina las llamadas a procedimientos en módulos separados y pasa los datos apropiados en forma de parámetros como se muestra en la Ilustración 2.

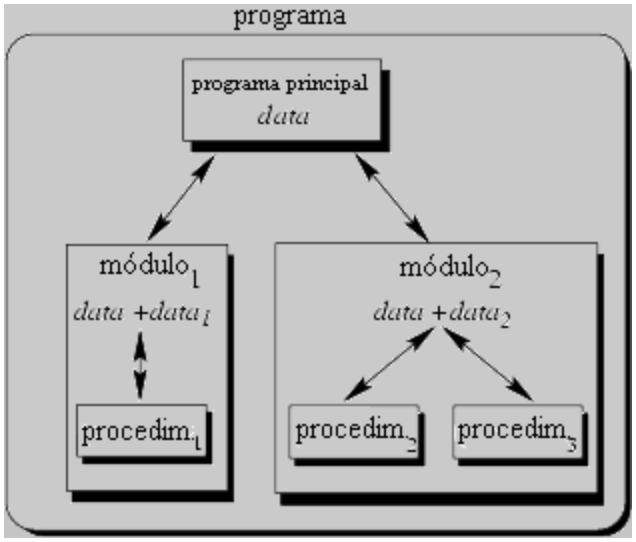

Ilustración 2 Funcionamiento de la Programación Modular.

La programación modular 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 [17].

1.2.4 Programación Orientada a Objetos

La Programación Orientada a Objetos (POO u OOP según sus siglas en inglés) es el paradigma de programación que usa objetos y sus interacciones para diseñar aplicaciones y programas de computadora. Está basado en varias técnicas, incluyendo herencia, modularidad, polimorfismo y encapsulamiento [5].

Esta a su vez expresa un programa como un conjunto de objetos que colaboran entre ellos para realizar tareas, lo cual permite hacer los programas y módulos más fáciles de escribir, mantener y reutilizar.

De esta forma, 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, permitiendo a su vez el cambio de estado en los propios objetos. Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separan, ni deben separarse el estado y el comportamiento.

Los métodos (comportamiento) y atributos (estado) están estrechamente relacionados por la propiedad de conjuntos. Esta destaca que una clase requiere de métodos para poder tratar los atributos con los que cuenta.

La POO es un concepto avanzado en la programación, que en sus inicios hizo que antiguos programadores cambiaran su forma de pensar a la hora de programar, pero una vez que se domina y se aprovecha al máximo se obtienen grandes ventajas.

En la programación orientada a objetos las clases tienen características que son propias y pudieran servir en muchos otros contextos; de ahí que se pueden reutilizar tal como son o quizás con alguna pequeña adaptación. Los elementos en instrucciones están mejor organizados y por esto encontrar errores es más sencillo. La herencia es otra de las grandes ventajas al poder derivar clases de otras con todas sus propiedades y características, ampliarlas o rediseñarlas en las nuevas clases que se denominan hijas de las primeras. Esta última posibilidad ahorra bastante tiempo en la implementación de clases.

Después de haber realizado un análisis acerca de las distintas técnicas de programación existentes las ventajas de cada una de estas expuestas anteriormente, se ha decidido emplear la Programación Orientada a Objetos, por sus características y ventajas que ofrece, las cuales hacen que sea la técnica empleada en Cuba y específicamente en la UCI.

1.3 Aplicaciones Web

Las aplicaciones web son aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador. Estas exponen una interfaz Web, pero mantienen un procesamiento lógico o proceso de negocio del lado del servidor. En caso de que esta lógica o proceso de negocio del lado del servidor no existiera, entonces se conoce como Sitio Web.

Estas 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 los webmails, wikis, weblogs, tiendas en línea y muchos otros ejemplos que constituyen aplicaciones web.

En ellas se generan dinámicamente una serie de páginas en un formato estándar, soportado por navegadores Web comunes como HTML o XHTML. Se utilizan lenguajes interpretados del lado del cliente, tales como JavaScript, para añadir elementos dinámicos a la interfaz de usuario. A continuación se realiza una caracterización de los lenguajes de programación.

1.4 Lenguajes de Programación Web

En todo el mundo existen diferentes lenguajes de programación que de acuerdo a las características que estos presentan y las que deben cumplir el producto que se desea desarrollar son escogidos por los programadores que finalmente harán uso de los mismos. De esta forma es que existen lenguajes de programación que son del lado del cliente y otros que son del lado del servidor.

Cada uno de ellos tiene sus ventajas e inconvenientes:

- Un lenguaje del lado cliente es totalmente independiente del servidor, lo cual permite que la página pueda ser albergada en cualquier sitio. El código, tanto del hipertexto como de los scripts, es accesible a cualquiera y ello puede afectar a la seguridad.

- Un lenguaje del lado servidor es independiente del navegador utilizado. Además, el código de los script puede ocultarse al terminal cliente, que sólo verá el código HTML terminado. Por otro lado, la prueba y depuración de un script deberá hacerse desde el servidor.

A continuación se aborda sobre los principales lenguajes de programación, tanto del lado del cliente como del lado del servidor, siguiendo este mismo orden.

1.4.1 Lenguaje JavaScript

El leguaje de programación JavaScript es precisamente un lenguaje que es empleado del lado del cliente en la programación de páginas web.

Fue creado por Brendan Eich en la empresa Netscape Communications[1]. Este es un lenguaje interpretado, es decir, no requiere compilación. Utilizado principalmente en páginas web. Es similar a Java, aunque no es un lenguaje orientado a objetos, si no que es orientado a pseudo-objetos. El mismo no dispone de herencias, aunque se puede conseguir clonando una clase y añadiendo más métodos o propiedades. La mayoría de los navegadores en sus últimas versiones interpretan código JavaScript ya que es de hecho un estándar.

Este lenguaje puede ser integrado dentro de nuestras páginas web. Para evitar incompatibilidades el World Wide Web Consortium (W3C) diseñó un estándar denominado DOM (en inglés Document Object Model, en su traducción al español Modelo de Objetos del Documento)[2] cuyos objetos modelan tanto la ventana del navegador como el historial, el documento o página web, y todos los elementos que pueda tener dentro la propia página, como párrafos, divisiones, tablas, formularios y sus campos entre otros. Así de esta manera a través del DOM se puede acceder, por medio de Javascript, a cualquiera de estos elementos, es decir a sus correspondientes objetos para alterar sus propiedades o invocar a sus métodos lo que permite a los programadores de Javascript acceder a cualquier elemento de la página, para modificarlo, suprimirlo, e incluso crear nuevos elementos. A continuación se describen algunas de las ventajas y desventajas que presenta el JavaScript.

Ventajas

- Lenguaje de scripting seguro y fiable.
- Los script tienen capacidades limitadas, por razones de seguridad.

[1] La compañía fue fundada como Mosaic Communications Corporation el 4 de abril de 1994 por Marc Andreessen y Jim Clark y fue una de las primeras compañías en trabajar con la naciente World Wide Web. La compañía cambió de nombre a "Netscape Communications Corporation" el 14 de noviembre de 1994.

[2] El DOM es una estructura de objetos que representa absolutamente todos los elementos que componen una web, y mediante él conseguiremos acceder a información de la página web, añadir nuevos elementos, o modificarlos.

- El código JavaScript se ejecuta en el cliente, lo que hace que sea totalmente independiente del servidor siendo más rígido respecto al cambio de navegador a otro o respecto a las versiones del mismo.

Desventajas

- Código visible por cualquier usuario.
- El código debe descargarse completamente.
- Puede poner en riesgo la seguridad del sitio, con el actual problema llamado XSS (significa en inglés Cross Site Scripting renombrado a XSS por su similitud con las hojas de estilo CSS).

1.4.2 CSS

CSS (Cascading Style Sheets), en español Hojas de estilo en Cascada, es una tecnología que permite la creación de páginas web de una manera más exacta. Estas son usadas con el objetivo de definir el estilo específico de cada uno de los atributos de la página sin tener que modificar las etiquetas dentro de la página HTML. Además brindan la posibilidad de realizar una serie de acciones que no podían hacerse con HTML. Tal es el caso de incluir márgenes, tipos de letra, fondos, colores e incluso definir estilos propios en un archivo externo a las páginas; de esta manera si en algún momento se desea cambiar alguno de ellos, automáticamente se actualizarán todas las páginas vinculadas al sitio.

CSS es un lenguaje de estilo pensado para dar al navegador detalles sobre el aspecto de un objeto, su principal objetivo es separar el documento (estructura y datos) en sí del aspecto del mismo.

1.4.3 HTML

HTML (HyperText Markup Language), en español Lenguaje de Marcas de Hipertexto), es el lenguaje de marcado predominante para la construcción de páginas web, es decir, diseñado para estructurar textos y presentarlos en forma de hipertexto. Es el formato estándar de las páginas web, además de que permite complementar dicho texto con objetos tales como imágenes. Gracias a Internet y a los navegadores como Internet Explorer, Opera, Firefox, Netscape o Safari, el HTML se ha convertido en uno de los formatos más populares y fáciles de aprender que existen para la elaboración de documentos para Web. HTML le indica corno hacer un documento interactivo a través de ligas especiales de hipertexto, las cuales

conectan diferentes documentos en una o varias computadoras, así como otros recursos de Internet, como FTP [8].

HTML se escribe en forma de "etiquetas", rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta un cierto punto, la apariencia de un documento y puede incluir un script (por ejemplo Javascript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML.

La estructura básica de HTML son los elementos. Estos tienen dos propiedades básicas: atributos y contenido. Cada atributo y contenido tiene ciertas restricciones para que se considere válido al documento HTML. Un elemento generalmente tiene una etiqueta de inicio ejemplo: (<nombre-de-elemento>) y una etiqueta de cierre (p.ej. </nombre-de-elemento>).

Los atributos del elemento están contenidos en la etiqueta de inicio y el contenido está ubicado entre las dos etiquetas ejemplo: (<nombre-de-elemento atributo="valor">Contenido</nombre-de-elemento>). Algunos elementos, tales como
, no tienen contenido ni llevan una etiqueta de cierre. La mayoría de los atributos de un elemento son pares nombre-valor, separados por un signo de igual "=" y escritos en la etiqueta de comienzo de un elemento después del nombre de éste. El valor puede estar rodeado por comillas dobles o simples, aunque ciertos tipos de valores pueden estar sin comillas en HTML (pero no en XHTML). [9] De todas maneras, dejar los valores sin comillas es considerado poco seguro. En contraste con los pares nombre-elemento, hay algunos atributos que afectan al elemento simplemente por su presencia.

Después de realizar un análisis de las principales características y ventajas que presentan cada uno de los lenguajes antes vistos, se llega a la conclusión que es necesario emplearlos en el desarrollo de la aplicación propuesta siguiendo la estructura que cada uno plantea asegurando obtener el resultado esperado.

1.4.4 Lenguaje PHP

Al igual que existen diversos lenguajes para el lado del cliente también los hay del lado del servido tal es el caso del PHP.

PHP es un lenguaje creado por una gran comunidad de personas. El sistema fue desarrollado originalmente en el año 1994 por Rasmus Lerdorf como un CGI escrito en C que permitía la interpretación de un número limitado de comandos. El sistema fue denominado Personal Home Page Tools y adquirió relativo éxito gracias a que otras personas pidieron a Rasmus que les permitiese utilizar sus programas en sus propias páginas. Es un lenguaje de programación utilizado para la creación de sitios web. PHP es un acrónimo recursivo que significa “PHP Hypertext Pre-processor”. Dada la aceptación del primer PHP y de manera adicional, su creador diseñó un sistema para procesar formularios al que le atribuyó el nombre de FI (Form Interpreter) y el conjunto de estas dos herramientas, sería la primera versión compacta del lenguaje: PHP/FI.

PHP es un lenguaje de script interpretado en el lado del servidor utilizado para la generación de páginas web dinámicas, embebidas en páginas HTML y ejecutadas en el servidor. PHP no necesita ser compilado para ejecutarse. Para su funcionamiento necesita tener instalado Apache o IIS con las librerías de PHP. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas. Los archivos cuentan con la extensión (php).

1.4.4.1 Lenguaje PHP 5

El 13 de julio de 2004, fue sacado a la luz pública PHP 5, utilizando el motor Zend Engine II (o Zend Engine 2). La versión más reciente de PHP es la 5.2.6 sacada el 1 de Mayo de 2008. En su última versión PHP 5 incorpora Programación Orientada a Objetos, lo que le convierte en un lenguaje aún más versátil. Es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación. PHP, en el caso de estar montado sobre un servidor Linux o Unix, es más rápido que ASP, dado que se ejecuta en un único espacio de memoria y esto evita las comunicaciones entre componentes COM que se realizan entre todas las tecnologías implicadas en una página ASP.

Esta última versión incluye todas las ventajas que provee el nuevo Zend Engine 2 [6] como:

- Mejor soporte para la Programación Orientada a Objetos, que en versiones anteriores era extremadamente rudimentario, con PHP Data Objects.
- Mejoras de rendimiento.
- Mejor soporte para MySQL con extensión completamente reescrita.

- Mejor soporte a XML (XPath, DOM, etc.).
- Soporte nativo para SQLite.
- Soporte integrado para SOAP.
- Iteradores de datos.
- Manejo de excepciones.

1.4.5 Lenguaje ASP

Es una tecnología del lado de servidor desarrollada por Microsoft para el desarrollo de sitios web dinámicos. ASP significa en inglés (Active Server Pages), fue liberado por Microsoft en 1996. Para las páginas web desarrolladas bajo este lenguaje es necesario tener instalado Internet Information Server (IIS). Es una tecnología de páginas activas que permite el uso de diferentes scripts y componentes en conjunto con el tradicional HTML para mostrar páginas generadas dinámicamente.

ASP no necesita ser compilado para ejecutarse. Existen varios lenguajes que se pueden utilizar para crear páginas ASP. El más utilizado es VBScript, nativo de Microsoft. ASP se puede hacer también en Perl and Jscript (no JavaScript). Los archivos cuentan con la extensión (asp). El ASP es una tecnología dinámica que funciona del lado del servidor, lo que significa que cuando el usuario solicita un documento ASP, las instrucciones de programación dentro del script son ejecutadas para enviar al navegador únicamente el código HTML resultante. La ventaja principal de las tecnologías dependientes del servidor radica en la seguridad que tiene el programador sobre su código, ya que éste se encuentra únicamente en los archivos del servidor que al ser solicitado a través del web, es ejecutado, por lo que los usuario no tienen acceso más que a la página resultante en su navegador. A continuación se abordan algunas de las ventajas y desventajas que presenta el ASP.

Ventajas

- Usa Visual Basic Script, siendo fácil para los usuarios.
- Comunicación óptima con SQL Server.
- Soporta el lenguaje JScript (JavaScript de Microsoft).

Desventajas

- Código desorganizado.
- Se necesita escribir mucho código para realizar funciones sencillas.
- Tecnología propietaria.
- Hospedaje de sitios web costosos.

1.4.6 Lenguaje ASP.NET

Este es un lenguaje comercializado por Microsoft y usado por programadores para desarrollar entre otras funciones, sitios web. ASP.NET es el sucesor de la tecnología ASP, fue lanzada al mercado mediante una estrategia de mercado denominada .NET.

El ASP.NET fue desarrollado como sucesor más completo de ASP. Creado para desarrollar web sencillas o grandes aplicaciones. Para el desarrollo de ASP.NET se puede utilizar C#, VB.NET o J#. Los archivos cuentan con la extensión (aspx). Para el funcionamiento de las páginas se necesita tener instalado IIS con el Framework .Net. Microsoft Windows 2003 incluye este framework, solo es necesario instalarlo en versiones anteriores. A continuación se abordan algunas de las ventajas y desventajas de ASP.NET

Ventajas

- Completamente orientado a objetos.
- Controles de usuario y personalizados.
- División entre la capa de aplicación o diseño y el código.
- Facilita el mantenimiento de grandes aplicaciones.
- Incremento de velocidad de respuesta del servidor.
- Mayor velocidad.
- Mayor seguridad.

Desventajas

- Mayor consumo de recursos.

1.4.7 Lenguaje JAVA

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems[3] a principios de los años 90. Su sintaxis toma mucho de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Las aplicaciones Java están típicamente compiladas en un bytecode[4]. En el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible.

La implementación original y de referencia del compilador, la máquina virtual y las bibliotecas de clases de Java fueron desarrolladas por Sun Microsystems en 1995. Desde entonces, Sun ha controlado las especificaciones, el desarrollo y evolución del lenguaje a través del Java Community Process[5].

Entre noviembre de 2006 y mayo de 2007, Sun Microsystems liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL[6], de acuerdo con las especificaciones del Java Community Process, de tal

---

3 Sun Microsystems es una empresa informática de Silicon Valley, fabricante de semiconductores y software. Fue constituida en 1982 por el alemán Andreas von Bechtolsheim y los norteamericanos Vinod Khosla, Bill Joy, Scott McNealy y Marcel Newman.

4 El bytecode es un código intermedio más abstracto que el código máquina. Habitualmente es tratado como un fichero binario que contiene un programa ejecutable similar a un módulo objeto, que es un fichero binario producido por el compilador cuyo contenido es el código objeto o código máquina. El bytecode recibe su nombre porque usualmente cada código de operación tiene una longitud de un byte.

5 El Proceso de la Comunidad Java, o Java Community Process, establecido en 1998, es un proceso formalizado el cual permite a las partes interesadas a involucrarse en la definición de futuras versiones y características de la plataforma Java.

6 La Licencia Pública General de GNU o más conocida por su nombre en inglés GNU General Public License, es una licencia creada por la Free Software Foundation a mediados de los 80, y está orientada

forma que prácticamente todo el Java de Sun es ahora software libre (aunque la biblioteca de clases de Sun que se requiere para ejecutar los programas Java todavía no es software libre). A continuación se exponen algunas de las ventajas y desventajas de JAVA.

Ventajas

- **Simple:** Elimina la complejidad de los lenguajes como "C" y da paso al contexto de los lenguajes modernos orientados a objetos.
- **Familiar:** Como la mayoría de los programadores están acostumbrados a programar en C o en C++, la sintaxis de Java es muy similar a la de estos.
- **Robusto:** El sistema de Java maneja la memoria de la computadora por el usuario e impide que este se preocupe por la memoria que no se esté utilizando. Elimina el uso de apuntadores.
- **Seguro:** El sistema de Java tiene ciertas políticas que evitan se puedan codificar virus con este lenguaje. Existen muchas restricciones, especialmente para los applets, que limitan lo que se puede y no puede hacer con los recursos críticos de la computadora.
- **Portable:** Como el código compilado de Java (conocido como bytecode) es interpretado, un programa compilado de Java puede ser utilizado por cualquier computadora que tenga implementado el interprete de Java.
- **Independiente a la arquitectura:** Al compilar un programa en Java, el código resultante es un tipo de código binario conocido como bytecode. Este código es interpretado por diferentes computadoras de igual manera, solamente hay que implementar un intérprete para cada plataforma. De esa manera Java logra ser un lenguaje que no depende de una arquitectura computacional definida.
- **Multithreaded:** Un lenguaje que soporta múltiples threads, es un lenguaje que puede ejecutar diferentes líneas de código al mismo tiempo.
- **Interpretado:** Java corre en máquina virtual, por lo tanto es interpretado.

---

principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

- **Dinámico**: Java no requiere que compiles todas las clases de un programa para que este funcione, es decir, si se realiza una modificación a una clase, Java se encarga de realizar un Dynamic Bynding o un Dynamic Loading para encontrar las clases.

Desventajas

- Dado que la máquina virtual de java es un intérprete y redunda en una falta de rendimiento con relación a aplicaciones equivalentes escritas en código máquina nativo.
- El poder reducir los problemas de acceso a memoria y liberación automática hacen de java un lenguaje poco apropiado para desarrollar aplicaciones de base como Sistemas Operativos.
- Los programas hechos en Java no tienden a ser muy rápidos, supuestamente se está trabajando en mejorar esto. Como los programas de Java son interpretados nunca alcanzan la velocidad de un verdadero ejecutable.

1.4.8 Comparación de PHP 5 con Java y ASP.NET

A continuación se expone una comparación del PHP 5 frente a otros lenguajes del lado del servidor, como son Java y ASP.NET, utilizando sus principales características, las cuales han sido empleadas por diferentes programadores para realizar disímiles tareas y en condiciones muy diferentes.

Ventajas frente a Java y ASP.NET

- Mejor soporte para la Programación Orientada a Objetos.
- Mejoras de rendimiento.
- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL con extensión completamente reescrita.
- Mejor soporte a XML (XPath, DOM, etc.).
- Soporte nativo para SQLite.
- Soporte integrado para SOAP.
- Iteradores de datos.
- Manejo de excepciones.
- Es un lenguaje multiplataforma.

- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Biblioteca nativa de funciones sumamente amplia e incluida.
- Es libre por lo que se presenta como una alternativa de fácil acceso para todos.

A partir de lo antes expuesto se puede plantear que el PHP 5 es el lenguaje que más se adapta a las características que requiere nuestro sistema por todas las ventajas que presenta, lo cual lo hace el mejor candidato entre todos los analizados, además de que este cuenta con gran disponibilidad de recursos en Internet. Cuenta con un motor de plantillas como SMARTY, una capa de abstracción de datos como ADOdb. Por ello se propone su uso como lenguaje del lado del servidor.

1.5 Eclipse como IDE de PHP

Eclipse es un Entorno Integrado de Desarrollo (IDE) de código abierto, para todo tipo de aplicaciones libres. Fue desarrollado originalmente por IBM como el sucesor de su familia de herramientas para VisualAge, y actualmente es desarrollado por la Fundación Eclipse: una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios.

Esta plataforma típicamente ha sido usada para desarrollar entornos de desarrollo integrados, como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse y que son usados también para desarrollar el mismo Eclipse. También se puede usar para otros tipos de aplicaciones cliente, como BitTorrent Azureus [1].

Eclipse es también una comunidad de usuarios, extendiendo constantemente las áreas de aplicación cubiertas. Un ejemplo es el recientemente creado Eclipse Modeling Project, cubriendo casi todas las áreas de Model Driven Engineering.

La versión actual de Eclipse dispone de las siguientes características:

- Editor de texto
- Resaltado de sintaxis
- Compilación en tiempo real

- Pruebas unitarias con JUnit.
- Control de versiones con CVS.
- Integración con Ant.
- Asistentes (wizards): para creación de proyectos, clases.
- Refactorización.
- Multiplataforma (GNU/Linux, Solaris, Mac OSX, Windows).
- Soportado para distintas arquitecturas (x86, 64).

Asimismo, a través de "plugins" libremente disponibles es posible añadir:

- Control de versiones con Subversión.
- Integración con Hibernate.

Ventajas

- Editor visual con sintaxis coloreada.
- Compilación incremental de código.
- Modifica e inspecciona valores de variables.
- Avisa de los errores cometidos mediante una ventana secundaria.
- Depura código que resida en una máquina remota.

Después de haber realizado un análisis sobre Eclipse como IDE de PHP y teniendo en cuenta sus características y ventajas, se ha decidido emplearlo en el desarrollo de la aplicación garantizando de esta forma obtener un mejor resultado, además de estar considerado como el mejor entorno de desarrollo integrado web para PHP 5 existente.

1.6 Apache como servidor Web

Apache es un servidor web que es considerado uno de los mayores triunfos del software libre. En Diciembre de 1997 tenía una cuota de mercado cercana al 45% y en Julio del 2000 ya estaba por encima del 60%. Esta es la primera cifra que hace que cualquier responsable de la estrategia Internet de una empresa tenga que tomar a Apache como el servidor de referencia. Apache era inicialmente unos parches al servidor que de WWW de National Center for Supercomputing Applications (NCSA) conocido como httpd (principios de 1995). Al igual que GNU/Linux, fue un proyecto que atrajo a muchas personas por el

gran interés de su objetivo: lograr el servidor web más rápido, más eficiente y con mayor funcionalidad desde el enfoque del software libre [15].

Características

Apache es un servidor web flexible, rápido y eficiente, continuamente actualizado y adaptado a los nuevos protocolos (HTTP 1.1). Entre sus características destacan:

- Multiplataforma.
- Es un servidor de Web conforme al protocolo HTTP/1.1.
- Modular: Puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona, y con la API de programación de módulos, para el desarrollo de módulos específicos.
- Basado en hebras en la versión 2.0.
- Incentiva la realimentación de los usuarios, obteniendo nuevas ideas, informes de fallos y parches para la solución de los mismos.
- Se desarrolla de forma abierta.
- Extensible: gracias a ser modular se han desarrollado diversas extensiones entre las que destaca PHP, un lenguaje de programación del lado del servidor.

1.7 Los frameworks como ayuda en el desarrollo de software

El concepto framework se emplea en muchos ámbitos del desarrollo de sistemas software, no solo en el ámbito de aplicaciones Web. Se puede encontrar frameworks para el desarrollo de aplicaciones médicas, de visión por computador, para el desarrollo de juegos, entre otros muchos.

Los objetivos principales que persigue un framework son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones. Un framework Web, por tanto, puede ser definido como un conjunto de componentes (por ejemplo clases en java y descriptores y archivos de configuración en XML) que componen un diseño reutilizable que facilita y agiliza el desarrollo de sistemas Web. [11]

Los Frameworks ayudan en el desarrollo de software, proporcionan una estructura definida la cual ayuda a crear aplicaciones con mayor rapidez, además de que ayuda a la hora de realizar el mantenimiento del sitio gracias a la organización durante el desarrollo de la aplicación.

Los Frameworks son desarrollados con el objetivo de brindarles a los programadores y diseñadores una mejor organización y estructura a sus proyectos.

1.7.1 Symfony

Symfony es un framework completo diseñado para optimizar de acuerdo a sus características, el desarrollo de las aplicaciones Web. Para empezar separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación Web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación Web compleja. Además automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación Web [12].

Symfony emplea el tradicional patrón de diseño MVC (modelo-vista-controlador) para separar las distintas partes que forman una aplicación web. El modelo representa la información con la que trabaja la aplicación y se encarga de acceder a los datos. La vista transforma la información obtenida por el modelo en las páginas web a las que acceden los usuarios. El controlador es el encargado de coordinar todos los demás elementos y transformar las peticiones del usuario en operaciones sobre el modelo y la vista.

¿Por qué utilizar Symfony?

Symfony está desarrollado completamente con PHP 5, es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y SQL Server de Microsoft. Se puede ejecutar tanto en plataformas (Unix, Linux, etc.) como en plataformas Windows.

El diseño de Symfony permite que se ajuste a los siguientes requisitos:

- Fácil de instalar y configurar en la mayoría de plataformas.
- Independiente del sistema gestor de bases de datos.

- Sencillo de usar en la mayoría de casos, y lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador solo debe configurar aquello que no es convencional.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la Web.
- Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.

Además de todo lo antes expuesto también se debe tener en cuenta que:

- Su código fuente incluye más de 8.000 pruebas unitarias y funcionales.
- Se trata del framework PHP mejor documentado.
- Se puede controlar hasta el último acceso a la información e incluye por defecto protección contra ataques XSS y CSRF.
- Symfony utiliza una licencia MIT[7], con la que puedes hacer aplicaciones web comerciales, gratuitas y/o de software libre.
- Symfony ha sido probado con éxito durante varios años en aplicaciones muy diferentes. Desde sitios web con millones de usuarios (del.icio.us, Yahoo Bookmarks, Yahoo Answers) hasta otros miles de sitios pequeños y medianos.
- Symfony es infinitamente escalable si se disponen de los recursos necesarios.

1.8 Arquitectura MVC

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. Este patrón se ve

[7] La licencia MIT es una de tantas licencias de software que ha empleado el MIT (Massachusetts Institute of Technology) a lo largo de su historia, El texto de la licencia no tiene copyright, lo que permite su modificación.

frecuentemente en aplicaciones web, con el cual se consigue un mantenimiento más sencillo de dichas aplicaciones. Aquí la vista es la página HTML y el código que provee de datos dinámicos a la página. El modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio y el controlador es el responsable de recibir los eventos de entrada desde la vista.

Si por ejemplo una misma aplicación debe ejecutarse tanto en un navegador estándar como un navegador de un dispositivo móvil, solamente es necesario crear una vista nueva para cada dispositivo; manteniendo el controlador y el modelo original. El controlador se encarga de aislar al modelo y a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email, etc.). El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes del tipo de gestor de bases de datos utilizado por la aplicación [16].

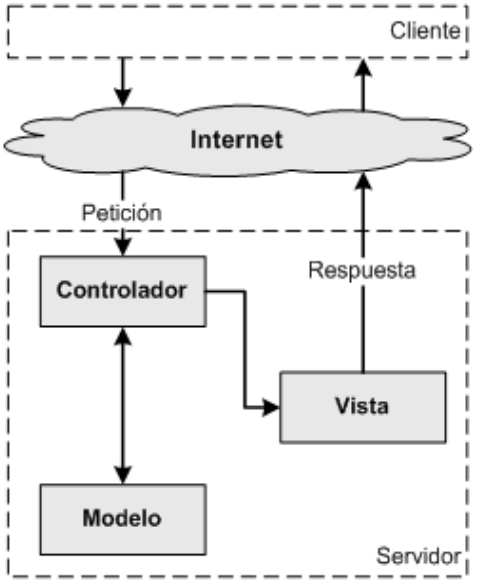

Ilustración 3 Funcionamiento del Patrón Modelo Vista Controlador.

- **Modelo.** El modelo administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).

- **Vista.** Maneja la visualización de la información.

- **Controlador.** Interpreta las acciones del ratón y el teclado, informando al modelo y/o a la vista para que cambien según resulte apropiado.

Ventajas del patrón de arquitectura MVC.

- **Soporte de vistas múltiples**. La vista se encuentra separada del modelo y no hay dependencia directa del modelo con respecto a la vista, la interfaz de usuario puede mostrar múltiples vistas de los mismos datos simultáneamente. Por ejemplo, múltiples páginas de una aplicación Web pueden utilizar el mismo modelo de objeto, mostrado de maneras diferentes.

- **Adaptación al cambio**. Los requerimientos de interfaz de usuario tienden a cambiar con mayor rapidez que las reglas de negocios. Los usuarios pueden preferir distintas opciones de representación o requerir soporte para nuevos dispositivos como teléfonos celulares o PDAs, dado que el modelo no depende de las vistas. Agregar nuevas opciones de presentación generalmente no afecta al modelo. Este patrón sentó las bases para especializaciones ulteriores, tales como Page Controller y Front Controller.

1.9 Sistemas Gestores de Bases de Datos (SGBD).

Los Sistemas Gestores de Bases de Datos son un tipo de software muy específico, dedicado a servir de interfaz entre las bases de datos y las aplicaciones que la utilizan. En la actualidad existe una gran variedad de SGBD, dentro de las principales funciones que satisfacen se encuentran la creación y mantenimiento de la base de datos, el control de accesos, la manipulación de datos de acuerdo con las necesidades del usuario, el cumplimiento de las normas de tratamiento de datos, evitar redundancias e inconsistencias y mantener la integridad [13].

Objetivos de los Sistemas Gestores de Bases de Datos

- **Abstracción de la información**: Los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos.
- **Independencia**: Consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.

- **Consistencia**: En aquellos casos en los que no se ha logrado eliminar la redundancia, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea.
- **Seguridad** (también llamada protección): garantizar el acceso autorizado a los datos como forma de interrumpir cualquier intento de acceso no autorizado, ya sea por error del usuario o por mala intención.
- **Integridad**: Se trata de adoptar las medidas necesarias para garantizar la validez de los datos almacenados. Es decir, se trata de proteger los datos ante fallos de hardware, datos introducidos por usuarios descuidados o cualquier otra circunstancia capaz de corromper la información almacenada.
- **Respaldo**: Los SGBD deben proporcionar una forma eficiente de realizar copias de respaldo de la información almacenada en ellos y de restaurar a partir de estas copias los datos que se hayan podido perder.

1.9.1 POSTGRESQL

PostgreSQL es un Sistema Gestor de Bases de Datos que ofrece un gran número de facilidades de uso por lo cual ha sido escogido para el desarrollo de la aplicación en cuestión. El mismo presenta características tales como: es un sistema bien conocido por su conformidad a los estándares, ha agregado sintaxis para varias características introducidas en la especificación ANSI SQL: 2003, incluyendo funciones de agregación estadística, sentencias VALUE con múltiples registros, UPDATE RETURNING y funciones de agregación de múltiples columnas. Posee una amplia licencia BSD (*Berkeley Software Distribution*), esta licencia básicamente consiste en que el código puede ser redistribuido y modificado. Posee una estabilidad y confiabilidad legendaria nunca ha presentado caídas en varios años de operación de alta actividad. Fue diseñado para ambientes de alto volumen intentando estar a la altura de Oracle, Sybase o Interbase. Tiene mejor soporte para triggers, vistas y procedimientos almacenados en el servidor, además tiene ciertas características orientadas a objetos [14].

¿Por qué utilizar PostgreSQL?

- PostgreSQL es un motor de bases de datos:
- Avanzado.

- De código abierto.
- RDBMS[8].

Soporta:

- Querys complejos, incluyendo subselects.
- Integridad referencial (Foreign Keys).
- Triggers.
- Vistas (Views).
- Integridad Transaccional (ACID).
- Control de versionado concurrente (MVCC).

Ventajas:

- Soporta transacciones y desde la versión 7.0, llaves foráneas (integridad referencial).
- Soporta un subconjunto de SQL92 MAYOR que el que soporta MySQL.

1.10 Conclusiones Parciales

En este capítulo se abordó el tema referente a las distintas técnicas de programación, así como los diferentes lenguajes tanto de programación web como de escritorio, haciendo énfasis en las características y ventajas de cada uno de estos. Todo esto permitió seleccionar la programación orientada a objetos como técnica a emplear ya que esta ha ido evolucionando y cada vez son más las posibilidades que brinda en el desarrollo de productos informáticos. Dentro de ella se selecciono el lenguaje de programación web PHP 5 del lado del servidor por las disímiles ventajas que este brinda. Mientras que para el lado del cliente será empleado el JavaScript. Se hará uso también de CCS para lograr un estilo bien definido en cada una de las páginas de la aplicación. Además se escogió PostgresSQL como Sistema Gestor de Base de Datos, Symfony como framework para el desarrollo de la aplicación y Eclipse como IDE para PHP.

[8] Sistema Administrador de Bases de Datos Relacionales

CAPÍTULO 2: Descripción y análisis de la solución propuesta.

2.1 Introducción

En este capítulo se realiza un análisis crítico y valorativo del diseño entregado por el analista y se explican los cambios necesarios que se realizaron para mejorar el proceso de implementación. Se hace un análisis de la complejidad y funcionamiento de los algoritmos más importantes para la implementación del sistema, así como las estructuras de datos utilizadas para manejar los datos en el programa. Se presenta la descripción de las clases fundamentales que definen el comportamiento del sistema con sus atributos y métodos esenciales**.** Finalmente se definen los estándares de codificación a utilizar por el programador.

2.2 Valoración crítica del diseño propuesto por el analista.

Del diseño propuesto por los analistas se pueden identificar las funcionalidades a implementar para que el sistema funcione correctamente, basándose en la descripción detallada de los casos de uso, se puede establecer una estrategia de trabajo que permita la implementación de la aplicación. Tomando como base la propuesta de las herramientas a usar, cabe puntualizar que como servidor de base de datos será utilizado en lugar de MySQL server V5.0.22 PostgreSQL por las facilidades que brinda el mismo además de ser totalmente libre, es decir, no se necesita pagar licencia para su uso, además de que al emplear Symfony como framework este hace uso del patrón de arquitectura Modelo Vista Controlador (MVC), y para el acceso a la aplicación por parte de los usuarios se emplea los servicios web que brinda la Universidad, los cuales permiten comprobar si el usuario y la contraseña con el cual están accediendo a la aplicación es válido .

2.3 Descripción de los algoritmos no triviales a implementar.

En esta sección se hará un análisis de aquellos algoritmos que resultan importantes para la implementación de la herramienta o que su codificación es compleja. Se mostrará el código fuente de los mismos y se describirán los pasos principales en qué consiste el algoritmo.

2.3.1 Algoritmo que permite al usuario loguearse en el sistema.

Este algoritmo está contenido en la función Login de la clase adminActions del módulo Admin. Su objetivo consiste en crear la sección de usuario en el momento en que este se registre, asignando a cada uno de los atributos de la misma el valor que le corresponde de acuerdo a los datos obtenidos de la consulta

realizada a la base de datos general del centro. De esta forma se comprueba la validez de dicha información lo que permite otorgarle acceso al usuario al sistema. Su importancia está dada porque permite que los usuarios accedan a la aplicación con los mismos datos con que acceden a distintos servicios que brinda el centro tales como correo, akademos, teleformación, entre otros.

```
public function executeLogin()
{
    $ldap = new Login();
    $this->nombre='';
    $this->usuario='';
    $this->correo='';
    $this->categoria='';

    if( $this->getRequestParameter('nombre')!='' &&
    $ldap->Logearse('uci.cu',$this->getRequestParameter('nombre'),$this->getRequestParameter('pass')))
    {
        $criteria = new Criteria();
        $criteria->add(UsuarioPeer::USUARIO,$this->getRequest()->getParameter('nombre'));
        $result = UsuarioPeer::doSelectOne($criteria);

        $this->getUser()->setAuthenticated(true);
        $this->getUser()->setAttribute('nombre', $ldap->getName());
        $this->getUser()->setAttribute('correo', $ldap->getMail());
        $this->getUser()->setAttribute('usuario', $ldap->getMailnickname());
        $this->getUser()->setAttribute('categoria', $ldap->getTitle());

        if($result != null)
        {
            $this->getUser()->setAttribute('idUser',$result->getIdUsuario());
            $c = new Criteria();
            $c->add(UsuarioRolPeer::ID_USUARIO,$result->getIdUsuario());
            $rol = UsuarioRolPeer::doSelectOne($c);
```

```
    if(count($rol))
    {
        if($rol->getRol()->getTipoRol() === "administrador")
        {
            $this->getUser()->addCredential('administrador');
            $this->redirect('admin/perfil');
        }
        else
        {
        if($rol->getRol()->getTipoRol() === "asesor_disennador")
            {
                $this->getUser()->addCredential('asesor_disennador');
                $this->redirect('admin/perfil');
            }
            else
            {
                if ($rol->getRol()->getTipoRol() === "asesor")
                {
                    $this->getUser()->addCredential('asesor');
                    $this->redirect('admin/perfil');
                }
            }
        }
        $this->redirect('admin/perfil');
    }
    else
    {
        $this->getUser()->addCredential('usuario');
        $this->redirect('admin/perfil');
    }
}
```

```
        else
        {
            $user = new Usuario();
            $user->setUsuario($this->getRequest()->getParameter('nombre'));
            $user->setNombre($this->getUser()->getAttribute('nombre'));
            $user->save();
            $criteria = new Criteria();
            $criteria->add(UsuarioPeer::USUARIO,$this->getRequest()->getParameter('nombre'));
            $usuario = UsuarioPeer::doSelectOne($criteria);
            $this->getUser()->setAttribute('idUser',$usuario->getIdUsuario());
            $this->getUser()->addCredential('usuario');
            $this->redirect('admin/perfil');
        }
    }
    else {
        $mensaje="";
        if ($this->getRequest()->getMethod() == sfRequest::POST) {
            $mensaje="Rectifique sus Datos";
        }
        $this->mensaje=$mensaje;
    }
}
```

Ilustración 4 Función Login de la clase adminActions perteneciente al módulo Admin.

Pasos del algoritmo de la función ***Login*:**

1. Se crea un objeto de la clase ldap y se declaran las variables que serán usadas posteriormente.
2. Se comprueba que el campo usuario no esté vacio y se hace una llamada a la función Logearse de la clase ldap pasándole como parámetros los datos del usuario y de esta forma comprobar si los mismos son correctos.
3. En caso que lo anterior se cumpla se realiza una búsqueda en la base de datos con el objetivo de comprobar si el usuario ya existe en la misma y se procede a asignar como valor a cada variable declarada anteriormente la información correspondiente obtenida de la consulta realizada a la base de datos del centro.
4. En caso que el usuario exista se realiza una nueva búsqueda para comprobar si tiene asignado algún rol con el objetivo de asignarle el mismo como credencial y se redirecciona a la pagina perfil.
5. Si el usuario no existe en la base de datos, se procede a insertar el mismo y se redirecciona a la pagina perfil.

6. En caso que el usuario inserte datos incorrectos en el momento de registrase se le muestra un mensaje al mismo informándole que debe comprobar sus datos

2.3.2 Algoritmo que permite asignar a un usuario determinado un rol.

Este algoritmo está contenido en la función Roles de la clase adminActions del módulo Admin. El objetivo de este consiste en asignar a un usuario determinado un rol. De esta forma al acceder el usuario al sistema tendrá asignado una serie de permisos que estarán en correspondencia con el tipo de rol. Su importancia está dada porque permite que los usuarios accedan solo a las funcionalidades que se correspondan con el rol que desempeña lo cual ayuda a la seguridad e integridad de la información que se maneja en el sistema.

```
public function executeRoles()
{
    $this->form = new RolesForm();
if ($this->getRequest()->getMethod() != sfRequest::POST)
    {
        return sfView::SUCCESS;
    }
 else
    {
        $criteria = new Criteria();
        $criteria->add(UsuarioPeer::NOMBRE,$this->getRequest()->getParameter('auto_nombre'));
        $usuario = UsuarioPeer::doSelectOne($criteria);
        $c = new Criteria();
        if ($usuario)
        {
            $c->add(UsuarioRolPeer::ID_ROL,$this->getRequest()->getParameter('rol'));
            $c->add(UsuarioRolPeer::ID_USUARIO,$usuario->getIdUsuario());
        }
        $id = UsuarioRolPeer::doSelectOne($c);
        if ($usuario && !$id)
         {
            $usuario_rol = new UsuarioRol();
            $usuario_rol->setIdUsuario($this->getRequest()->getParameter('nombreId'));
            $usuario_rol->setIdRol($this->getRequest()->getParameter('rol'));
            $usuario_rol->save();
            $this->getUser()->setFlash('confirmacion','El rol ha sido asignado satisfactoriamente');
            $this->redirect('admin/roles');
         }
```

```
        elseif (!$usuario)
            {
                $this->getUser()->setFlash('confirmacion','Este usuario no existe');
                return sfView::SUCCESS;
            }
            else
            {
                $this->getUser()->setFlash('confirmacion','Este usuario ya tiene este rol');
                return sfView::SUCCESS;
            }
    }
}
```

Ilustración 5 Función Roles de la clase adminActions perteneciente al módulo Admin.

Pasos del algoritmo de la función ***Roles*:**

1. Se comprueba al ejecutar la función *Roles* si la informacion procedente de la vista está siendo enviada por el método Post.
2. En caso de no cumplirse lo anterior solo se mostrará la vista con los campos del formulario.
3. Si la informacion que se obtiene fue enviada por el método *Post* entonces se procesa la misma.
4. Inicialmente se realiza una búsqueda en la base de datos con el objetivo de comprobar que el nombre de usuario al que se le asignara un rol determinado es válido.
5. En caso de ser valido el nombre de usuario se comprueba que el mismo no tenga asignado el rol seleccionado.
6. Después de haber realizado ambas comprobaciones si el usuario es válido y no tiene asignado el rol seleccionado se procede a asignarle el mismo informando al usuario de la correcta ejecución de la acción.
7. En caso de no ser valido el nombre del usuario introducido esto es informado.
8. Si ya el usuario tiene asignado el rol seleccionado esto es informado.

2.3.3 Algoritmo que permite eliminar un usuario determinado.

Este algoritmo está contenido en la función Eliminar Usuario de la clase adminActions del módulo Admin. Su objetivo consiste en eliminar un usuario determinado.

```
public function executeEliminarUsuario()
{
    $this->form = new RolesForm();
    if ($this->getRequest()->getMethod() != sfRequest::POST)
    {
        return sfView::SUCCESS;
    }
 else
    {
        $c = new Criteria();
        $c->add(UsuarioPeer::NOMBRE,$this->getRequest()->getParameter('auto_nombre'));
        $usuario = UsuarioPeer::doSelectOne($c);
        if ($usuario)
         {
            $criteria = new Criteria();
            $criteria->add(UsuarioPeer::ID_USUARIO,$this->getRequest()->getParameter('nombreId'));
            UsuarioPeer::doDelete($criteria);
            $this->getUser()->setFlash('confirmacion','El usuario ha sido eliminado satisfactoriamente');
            $this->redirect('admin/eliminarUsuario');
         }
         else
         $this->getUser()->setFlash('confirmacion','El usuario es incorrecto');
    }
}
```

Ilustración 6 Función Eliminar Usuario de la clase adminActions perteneciente al módulo Admin.

Pasos del algoritmo de la función ***Eliminar Usuario*:**

1. Se comprueba al ejecutar la función Eliminar Usuario si la información procedente de la vista está siendo enviada por el método Post.
2. En caso de no cumplirse lo anterior solo se mostrará la vista con los campos del formulario.
3. Si la información que se obtiene fue enviada por el método *Post* entonces se procesa la misma.
4. Se realiza una búsqueda en la base de datos con el objetivo de comprobar que el nombre de usuario que se desea eliminar es válido.
5. En caso de cumplirse lo anterior se procede a eliminar el usuario informando del correcto funcionamiento de la acción.
6. Si el usuario no es válido esto es informado.

2.3.4 Algoritmo que permite adicionar una encuesta.

Este algoritmo está contenido en la función Adicionar Encuesta de la clase gaprendizajeActions del módulo GAprendizaje. Su objetivo es brindar la posibilidad al administrador del sistema o al asesor diseñador de insertar nuevas encuestas, para posteriormente ser publicadas.

```
public function executeAdicionarEncuestas()
{
    if ($this->getRequest()->getMethod() != sfRequest::POST)
    {

        return sfView::SUCCESS;
    }
else
    {
        $criteria = new Criteria();
        $criteria->add(EncuestaPeer::TIPO,$this->getRequest()->getParameter('nombre'));
        $result = EncuestaPeer::doSelectOne($criteria);

        if($result != null)
        {
            $this->getUser()->setFlash('confirmacion','Ya existe una encuesta con este nombre');
            $this->redirect('gaprendizaje/encuestas');
        }
        else
        {
            $encuesta = new Encuesta();
            $encuesta->setTipo($this->getRequest()->getParameter('nombre'));
            $encuesta->save();
            $this->redirect('gaprendizaje/adicionarPreguntas');
        }
    }
}
```

Ilustración 7 Función Insertar Encuesta de la clase gaprendizajeActions perteneciente al módulo GAprendizaje.

Pasos del algoritmo de la función ***Insertar Encuesta*:**

1. Se comprueba al ejecutar la función Insertar Encuesta si la informacion procedente de la vista está siendo enviada por el método Post.
2. En caso de no cumplirse lo anterior solo se mostrará la vista con los campos del formulario.

3. Si la información que se obtiene fue enviada por el método *Post* entonces se procesa la misma.
4. Se realiza una búsqueda en la base de datos con el objetivo de comprobar si la encuesta que se desea insertar ya existe en la base de datos.
5. En caso de existir se informa y se redirecciona a la misma página.
6. Si no existe se procede a insertar la encuesta y se redirecciona a la pagina donde se insertaran las preguntas de dicha encuesta.

2.3.5 Algoritmo que permite adicionar preguntas a una encuesta determinada.

Este algoritmo está contenido en la función Adicionar Preguntas de la clase gaprendizajeActions del módulo GAprendizaje. Tiene como objetivo brindar la posibilidad al administrador del sistema o al asesor diseñador seleccione a que encuesta desea insertar las preguntas.

```
public function executeAdicionarPreguntas()
{
    if ($this->getRequest()->getMethod() != sfRequest::POST)
    {
        $this->form = new AdicionarPreguntasForm();
        return sfView::SUCCESS;
    }
else
    {
        $this->form = new AdicionarPreguntasForm();
        $this->cantidad = $this->getRequest()->getParameter('cantidad');
        $this->preguntas = $this->getRequest()->getParameter('preguntas');
        if ($this->preguntas)
        {
            foreach ($this->preguntas as $pregunta) {
                $preguntas = new Preguntas();
                $preguntas->setIdEncuesta($this->getRequestParameter('encuestas'));
                $preguntas->setPregunta($pregunta);
                $preguntas->save();
            }
        }
    }
}
```

Ilustración 8 Función Insertar Preguntas de la clase gaprendizajeActions perteneciente al módulo GAprendizaje.

Pasos del algoritmo de la función ***Insertar Preguntas*:**

1. Se comprueba al ejecutar la función Insertar Preguntas si la información procedente de la vista está siendo enviada por el método Post.
2. En caso de no cumplirse lo anterior solo se mostrará la vista con los campos del formulario.
3. Si la información que se obtiene fue enviada por el método *Post* entonces se procesa la misma.
4. Se crea una variable donde será guardado el arreglo de preguntas.
5. Se comprueba que el arreglo no esté vacío.
6. En caso de contener elementos se crea un ciclo para recorrer el mismo.
7. Se insertan cada una de las preguntas junto con el identificador de la encuesta a la que pertenecen.

2.4 Descripción de las nuevas clases u operaciones necesarias.

A continuación se hará una descripción de las clases que se utilizaron para modelar el problema.

Las clases serán descritas en una tabla que consta de los siguientes campos:

- **Nombre:** En este campo se escribirá el nombre o identificador de la clase.
- **Tipo de clase:** En este campo se especifica el tipo de clase que representa la misma para el diseño de la solución.
- **Atributo:** En este campo se escribe el nombre o identificador para cada atributo que contenga la clase en cuestión.
- **Tipo:** En este campo se especifica el tipo de dato al que pertenece el atributo cuyo identificador aparece a la izquierda.
- **Nombre (Para cada responsabilidad):** En este campo se escribe el nombre o identificador de cada función o método de la clase seguido de los parámetros que la misma debe recibir (con su tipo de dato e identificador encerrados entre paréntesis).
- **Descripción (Para cada responsabilidad):** En este campo se escribe una breve descripción del objetivo o acción principal de la función correspondiente unida a alguna observación oportuna que se requiera de la misma.

Tabla 1 Descripción de la clase adminActions.

Nombre: adminActions

Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	executeIndex()
Descripción:	Ejecuta la página Índex del módulo.
Nombre:	executePerfil()
Descripción:	Ejecuta la página Perfil donde se muestra información personal al usuario.
Nombre:	executeLogin()
Descripción:	Ejecuta la función Login que permite al usuario registrarse y acceder a la aplicación.
Nombre:	executeLogout()
Descripción:	Ejecuta la función Logout que se encarga de cerrar la sección del usuario cuando este seleccione la opción salir.
Nombre:	executeContrato()
Descripción:	Ejecuta la página Contrato donde aparece el formulario del contrato que el usuario debe llenar.
Nombre:	executeRoles()
Descripción:	Ejecuta la página Roles donde aparece el formulario correspondiente a la función asignar roles la cual tiene un campo donde se inserta el nombre del usuario al que se le asignara un rol determinado y otro donde aparecen los roles del sistema.
Nombre:	executeEliminarUsuario()
Descripción:	Ejecuta la página Eliminar Usuario donde aparece el formulario correspondiente a la función eliminar usuario la cual tiene un campo donde se inserta el nombre del usuario que se desea eliminar.
Nombre:	executeNoticias()
Descripción:	Ejecuta la página Noticias donde estarán publicadas noticias referentes a los distintos temas que sean de interés para los usuarios.
Nombre:	executeCursos()
Descripción:	Ejecuta la página Cursos donde se publicarán los cursos disponibles para los

	usuarios.
Nombre:	executeAsignarIdioma()
Descripción:	Ejecuta la página Asignar Idioma donde aparece el formulario correspondiente a la función asignar idioma la cual tiene un campo donde se inserta el nombre del usuario al que se le asignara un idioma determinado y otro donde aparecen los idiomas.
Nombre:	executeEliminarRol()
Descripción:	Ejecuta la página Eliminar Rol donde aparece el formulario correspondiente a la función eliminar rol la cual tiene un campo donde se inserta el nombre del usuario al que se le eliminará el rol mostrando además todos los roles que tenga asignado dicho usuario junto con la opción de eliminar.
Nombre:	executeCambiarIdioma()
Descripción:	Ejecuta la página Eliminar Idioma donde aparece el formulario correspondiente a la función eliminar idioma la cual tiene un campo donde se inserta el nombre del usuario al que se le eliminará el idioma mostrando además todos los idiomas que tenga asignado dicho usuario junto con la opción de eliminar.

Tabla 2 Descripción de la clase gaprendizajeActions.

Nombre: gaprendizajeActions	
Tipo de clase: Controladora	
Para cada responsabilidad:	
Nombre:	executeIndex()
Descripción:	Ejecuta la página Índex del módulo.
Nombre:	executeEncuestasingles()
Descripción:	Ejecuta la página Encuestas Inglés donde se muestran los vínculos a cada una de las encuestas publicadas.
Nombre:	executeGestionarencuestas()
Descripción:	Ejecuta la página Gestionar Encuesta donde están disponible cada una de las acciones que se pueden realizar a las encuestas como son: Modificar, Insertar, Eliminar.
Nombre:	executeAdicionarEncuestas()

Descripción:	Ejecuta la pagina Adicionar Encuesta donde aparece el formulario correspondiente a esta función el cual tiene un campo donde se inserta el nombre de la encuesta que se desea insertar.
Nombre:	executeAdicionarPreguntas()
Descripción:	Ejecuta la página Adicionar Preguntas, donde aparece el formulario correspondiente a esta función el cual tiene un campo donde se muestran las encuestas que han sido insertadas brindando la posibilidad de escoger una e insertar las preguntas deseadas.
Nombre:	executeModificarEncuestas()
Descripción:	Ejecuta la pagina Modificar Encuesta donde aparece el formulario correspondiente a dicha función el cual tiene un campo donde se muestran las encuestas existentes brindando la posibilidad de escoger la que se desee modificar.
Nombre:	executeEliminarEncuestas()
Descripción:	Ejecuta la página Eliminar Encuesta donde aparece el formulario correspondiente a dicha función la cual muestra un campo con el listado de las encuestas existentes brindando la posibilidad de seleccionar la que se desea eliminar .
Nombre:	executeGetEncuesta()
Descripción:	Ejecuta la página Get Encuesta donde se muestran todas las preguntas de la encuesta junto con las opciones de respuesta de cada una de ellas, de forma que se les pueda dar respuesta a cada una de ellas.
Nombre:	executeMostrarUsuario()
Descripción:	Ejecuta la página Mostrar Usuario donde el formulario correspondiente a dicha función el cual muestra un campo de búsqueda junto con las distintas opciones de búsqueda.

2.5 Estándares de codificación

Es prudente establecer estándares de codificación para todos los programadores. Estos estándares consisten en estilos de codificación a la hora de escribir el código. Los aspectos para los que generalmente se establecen estándares son los siguientes:

- Identificadores.
- Indentación.
- Líneas y espacios en blanco.
- Comentarios.

En cada grupo de desarrollo se definen cuales serán los aspectos a estandarizar y que estilos se aplicarán a cada uno de ellos. El cumplimiento de estándares hace que todo el código lleve el sello personal del programador y en caso de ser varios los programadores pues se busca que todo el código parezca que ha sido implementado por la misma persona. De esta manera se consigue mayor legibilidad y facilidad de mantenimiento. Los estándares deben responder además a acciones prácticas que acomoden al programador. A continuación se definirá que estándares usar para la actividad de implementación correspondiente a este trabajo. Cabe destacar que estos estándares se definen teniendo en cuenta el estilo personal del programador, las características propias del lenguaje de programación, los recursos que se utilizarán y el tipo de programa que se debe implementar.

2.5.1 Identificadores

En el caso de los identificadores existen estilos definidos mundialmente como el *lowerCamelCase*[9] y el *UperCamelCase*. Cada palabra interna en identificadores compuestos comienza con mayúsculas para ambos estilos, además ocurre que no se colocan caracteres de separación entre las palabras que conforman un identificador compuesto en ninguno de los dos casos. Para el primero, el identificador comienza con minúscula y para el segundo, el identificador comienza con mayúscula.

- **Espacio de nombre**: Para los espacios de nombres se escogió el *UperCamelCase*.
- **Clase**: Para las clases se escogió el *UperCamelCase*.

[9] CamelCase es un estilo de escritura que se aplica a frases o palabras compuestas. El nombre CamelCase se podría traducir como Mayúsculas/Minúsculas Camello, aunque no es correcto en todos los contextos ya que la palabra inglesa Case no tiene traducción literal. El nombre se debe a que las mayúsculas a lo largo de una palabra en CamelCase se asemejan a las jorobas de un camello.

```
class pruebaActions extends sfActions
```

Ilustración 9 Uso del estilo UperCamelCase para el nombre de las clases.

- **Variable**: Para las variables se establece el *UperCamelCase*.

```
private $nombreUsuario;
```

Ilustración 10 Uso del estilo UperCamelCase para el nombre de las variables.

- **Función**: Para las funciones se establece el *UperCamelCase*.

```
public function getName()
```

Ilustración 11 Uso del estilo UperCamelCase para el nombre de las funciones.

Los identificadores que sean empleados sin importar su tipo, deben ser lo suficientemente descriptivos. Con esto se evita escribir abreviaturas que pueden confundir a otros programadores por desconocer su significado. Además se deben utilizar palabras que no den un significado ambiguo.

2.5.2 Indentación

La Indentación es una práctica de programación que consiste en comenzar a escribir cada línea de código a diferentes distancias desde el borde izquierdo del área de texto del editor. Esta distancia está determinada por la jerarquía que se forma al introducir sentencias dentro de bloques de estructuras. Esto brinda mayor legibilidad y entendimiento para el programador e igualmente depende del propio estilo de cada persona, del lenguaje y tipo de programa que se implementa. Se definió que la *indentación* se hará agregando un *tab* al inicio de la línea que se desee escribir. Se escribirá solo una sentencia por línea de código y en el caso de cortar las líneas, se hará luego de una coma o antes de un operador. La sección de la derecha de la línea que se corte se ubicará en la línea siguiente indentada al nivel de la expresión correspondiente en la línea superior.

```
10    class Login
11    {
12    private $displayname;
13    private $mail;
14    private $mailnickname;
15    private $title;
16
17    public function getName(){
20    public function getMail(){
23    public function getMailnickname(){
26    public function getTitle(){
29
30    public function Logearse($domain,$usuario,$contrasenna)
61
62    public function search_user($domain,$user,$password,$usrer_2)
85
86    }
```

Ilustración 12 Ejemplo de Indentación.

2.5.3 Llaves

Existe diversidad de criterios en cuanto a la ubicación de las llaves que delimitan el cuerpo de los bloques de código en los lenguajes que contienen este tipo de estructuras. Algunos programadores prefieren hacerlo ubicando la llave de apertura inmediatamente detrás de la línea cabecera del bloque mientras otros apuestan por ubicarlas de forma solitaria en la línea siguiente a la línea cabecera. Para este último estilo existen además diferencias en cuanto al nivel de Identación de las mismas. Algunos lo hacen al nivel de la línea cabecera y otros al nivel de las líneas del cuerpo del bloque.

Para este trabajo: Las llaves de apertura se colocarán solitarias en la línea siguiente e indentadas al nivel de la línea cabecera del bloque. Las llaves de cierre se colocarán solitarias en la línea que sigue a la última línea dentro del bloque e indentadas al nivel de la línea cabecera del bloque. Es prudente señalar que este estilo agrega más líneas de código al programa al ubicar las llaves solitarias en una línea pero a su vez se gana en legibilidad del código.

```
public function getMail()
{
return $this->mail;
}
```

Ilustración 13 Ubicación de las llaves.

2.5.4 Líneas y espacios en blanco

Para mejorar la legibilidad y organización del código muchas veces se utilizan líneas en blanco para separar segmentos de código que pueden corresponder a clases, funciones, declaraciones, implementaciones, comentarios, bloques o sencillamente secciones críticas que se deseen despejar. Así mismo sucede con los espacios en blanco cuando se utilizan para separar elementos dentro de las sentencias de código. En ocasiones se separan con espacios cada operador de su respectivo operando, paréntesis, identificadores, símbolos y algunos lenguajes exigen que se separen las palabras propias del vocabulario de las adyacentes para ser comprendidas por los compiladores. En este trabajo se ha definido emplear **líneas en blanco:**

- Entre funciones.

```
public function getMail()
{
return $this->mail;
}

public function getMailnickname()
{
return $this->mailnickname;
}
```

Ilustración 14 Uso de las líneas en blanco.

- Entre las declaraciones de interfaces, espacios de nombre, estructuras y clases.
- Entre declaraciones de variables e implementaciones dentro del cuerpo de las funciones.

```
$this->categoria='';

if( $this->getRequestParameter('nombre')!=''&&$ldap->Logearse
```

Ilustración 15 Uso de las líneas en blanco.

Se colocarán **espacios en blanco:**

- Entre las palabras reservadas y los elementos adyacentes a las mismas.

```
    private $displayname;
```

Ilustración 16 Uso de espacios en blanco.

- Después de las comas en la lista de argumentos de las funciones.

```
    public function Logearse($domain,$usuario,$contrasenna)
```

Ilustración 17 Uso de espacios en blanco.

- Entre los operadores binarios y los elementos adyacentes a los mismos.
- Después de cada *punto-y-coma* (—;||) en las estructuras *for*.

2.5.5 Comentarios

El uso de comentarios durante la codificación ha demostrado que es beneficiosa por varias razones:

- Ayuda al programador a entender cada elemento o sección de código.
- Hace más fácil el proceso de adaptación del código durante su reutilización.
- Sirve de guía en los casos en que varios programadores trabajen sobre las mismas secciones del código.
- Disminuye el esfuerzo de análisis ya que el lenguaje natural es más legible que cualquier lenguaje de programación.

Todo esto se aprecia claramente cuando se escribe gran cantidad de código en largos intervalos de tiempo donde generalmente el o los programadores olvidan lo que se pensó en un momento. Los comentarios se pueden utilizar para varios fines:

- Para explicar el propósito de las funciones.

- Para explicar las características fundamentales de las clases.
- Para sintetizar las acciones de los algoritmos complejos.
- Para aclarar los datos que representan las variables.
- Para dividir secciones de código en dependencia de los diferentes contextos y funciones.
- A veces se usan comentarios temporales para recordar cosas que faltan, cosas que se deben modificar o analizar en otro momento.

Es necesario tener en cuenta algunos detalles al escribir **comentarios**:

- La capacidad de síntesis.
- El uso de lenguaje técnico.
- No repetir exactamente paso por paso lo que hace el algoritmo sino expresar un resumen de su propósito.
- Usar un estilo uniforme de comentario definido en estándares para todo el equipo.
- Escribir el comentario solo donde sea necesario.

En este trabajo se decidió colocar los comentarios encima de la línea a la que se le quiera aplicar y encima de la línea cabecera de los bloques. La Indentación se hará al nivel de la línea en cuestión. Se utilizarán en funciones y clases. Se puede utilizar en algoritmos no triviales y secciones de diferentes contextos dentro de los métodos.

2.6 Conclusiones

En este capítulo se realizó un análisis del diseño entregado por el analista donde se hicieron algunas modificaciones y aportes necesarios en el proceso de implementación para obtener los resultados deseados. Se hizo un esbozo de los componentes utilizados en la solución de las funcionalidades principales. Se logró establecer una línea base sencilla que permita la programación eficiente por parte del desarrollador. Se establecieron los estándares de codificación pertinentes para que el código generado tenga la legibilidad necesaria al trabajar con algoritmos críticos y grandes volúmenes de código. Se propusieron ejemplos sencillos incluyendo figuras para su comprensión. Se explicó la implementación de aquellas partes que pudieran resultar confusas, ambiguas o complicadas.

CAPÍTULO 3: Validación de la solución propuesta.

3.1 Introducción

En este capítulo se hace una búsqueda de las técnicas de prueba de software que permitan comprobar la validez de la solución obtenida. Se detallan los casos de prueba con sus objetivos y alcance de los mismos. Se aborda el tema relacionado con las herramientas empleadas en el proceso de desarrollo y se analizan los resultados obtenidos.

3.2 Pruebas de software

La calidad de un sistema está determinada, entre otras cosas, por la coincidencia entre lo que se programó y los requisitos establecidos en la primera fase. Para comprobar el grado de cumplimiento de estos requisitos se usan las pruebas del sistema. Estas definen un conjunto amplio de acciones de comprobación que abarcan todas las características que determinan la calidad de un software. Se comprueban las funcionalidades diseñando casos de prueba que definen cómo proceder. Estos casos de prueba incluyen los juegos de datos a usar que son los válidos o esperados y los no válidos o no esperados por el programa. Además establecen los resultados a alcanzar en correspondencia de la lógica del programa y los datos ingresados. Describen las condiciones generales en las que se debe aplicar las pruebas para obtener los objetivos propuestos. El objetivo de los casos de prueba es forzar al máximo el sistema en los puntos críticos para encontrar fallos y detectar defectos. Las pruebas se deben aplicar durante todo el ciclo de vida del software e invariablemente se le debe dedicar una gran parte del esfuerzo total del desarrollo. Se deben planificar correctamente desde el inicio y establecer qué hacer, cómo hacer, quién va a hacer y en qué condiciones hacer las comprobaciones. Es beneficioso que los desarrolladores prueben su producto pero que no falte la mano de terceras personas que no intervinieron en el proyecto directamente ya que así se detecta mayor cantidad de fallas [18].

Se debe escoger los tipos de prueba que se adapten mejor al sistema que se va a probar. Para esto se debe tener en cuenta el lenguaje de programación, el proceso de desarrollo, las características de los desarrolladores, el tipo de funcionalidad que se implementa, la plataforma en que se ejecutan los procesos, los errores más importantes, si la aplicación es de escritorio o web, si realiza conexiones a bases de datos entre otras observaciones.

3.3 Automatización de pruebas

En el desarrollo de aplicaciones web el proceso de probar la aplicación de una manera correcta supone un gran esfuerzo. Esto está dado por el hecho de que los requisitos de la aplicación están sujetos a cambios constantes, lo que implica un gran número de versiones de la aplicación y con ello la aparición de nuevos errores.

Este es el motivo por el que la automatización de pruebas es una recomendación, aunque no una obligación, útil para crear un entorno de desarrollo satisfactorio. Las pruebas automatizadas permiten garantizar que los cambios no introducen incompatibilidades en el funcionamiento de la aplicación. Además, este tipo de pruebas obligan a los programadores a crear pruebas en un formato estandarizado y muy rígido que pueda ser procesado por un framework de pruebas [12].

El tener el proceso de prueba automatizado permite ilustrar el funcionamiento de la aplicación y al realizar un gran número de estas. Se puede mostrar la salida que produce la aplicación para una serie de entradas de prueba, brindando la posibilidad de entender el propósito de cada método. El proceso de automatización de pruebas es la parte del ciclo de calidad en la que el software de automatización es utilizado para controlar la ejecución de pruebas, comparación de resultados, preparación de precondiciones y realización de informes. Los dos grandes grupos de pruebas unitarias existentes son las pruebas de Caja Negra y las pruebas de Caja Blanca.

3.4 Pruebas de Caja Blanca

Las pruebas de Caja Blanca se nombran de esta forma porque a diferencia de las pruebas de Caja Negra que actúan sobre la interfaz, estas revisan la parte interna del software, específicamente sobre el código fuente. Se basan en el examen minucioso de los detalles procedimentales. Se comprueban los caminos lógicos del sistema generando casos de prueba que ejerciten las estructuras condicionales y los bucles. Es por esto que las pruebas unitarias se basan en las Técnicas de Pruebas de Caja Blanca. Existen varios métodos que analizan diferentes partes del programa y se complementan entre sí para garantizar la calidad del sistema. La técnica del camino básico se utiliza para comprobar la complejidad lógica de un diseño procedimental, permite diseñar casos de prueba para cubrir todas las sentencias de un programa a partir de la obtención de un conjunto de caminos independientes. La complejidad ciclomática, como resultado fundamental de estas pruebas, acota la cantidad mínima de casos de prueba que se deben

ejecutar. La prueba de las Condiciones es un método que se encamina hacia la ejercitación de las condiciones. Se basa en el principio de que si un conjunto de casos de prueba es capaz de ejercitar todas las condiciones contenidas en un bloque de código, este mismo conjunto serviría para encontrar más errores en el programa que no tengan que ver directamente con las condiciones. La prueba del Flujo de Datos verifica la validez en el uso de las variables para manipular los datos de la aplicación. Selecciona los casos de prueba atendiendo a las definiciones y los usos de las variables. El procedimiento indica que se debe encontrar las sentencias donde se define cada variable y las sentencias donde se hace uso de las mismas. Luego se encuentran las —cadenas de definición - uso que representan el ciclo de vida de las variables y se diseñan casos de prueba que las ejecuten en su totalidad. La prueba de los Bucles se centra en la validez de las estructuras cíclicas o bucles. El objetivo es probar el comportamiento de estas estructuras en sus valores límites de iteración. Los bucles se clasifican en cuatro tipos: Bucles simples, Bucles anidados, Bucles concatenados y Bucles no estructurados. Aunque la esencia es la misma, cada tipo se prueba de forma diferente. De lo anterior pudiera pensarse que las pruebas de Caja Blanca logran enmendar todos los errores del programa. La desventaja de estas es entonces de tipo logística ya que resulta imposible abarcar todo el código fuente de un sistema medianamente grande. El tiempo necesario para realizarlas sería considerable y se torna compleja su aplicación sobre algoritmos críticos.

3.5 Pruebas de Caja Negra

Las Pruebas de Caja Negra deben su nombre a los elementos que estas revisan y las condiciones en que se hace la revisión. Estas se basan en los requerimientos funcionales del sistema y se llevan a cabo desde el exterior de la aplicación. Este tipo de prueba es importante a la hora de medir el grado de cumplimiento de los requerimientos solicitados por el cliente y se aplican sobre la interfaz de la aplicación observando las respuestas del sistema antes determinadas acciones y los datos de salida para determinados datos de entrada [18].

Al analizar los métodos de prueba de Caja Blanca y Caja Negra se llega a la conclusión de que es más factible aplicar las pruebas de Caja Negra para comprobar la validez en las respuestas del programa ante las acciones del usuario y la calidad de las salidas en dependencia de las entradas. Para probar la validez de la solución propuesta en la implementación del sistema se decidió probar los elementos fundamentales de la interfaz de usuario mediante casos de prueba de Caja Negra así como la validez de las salidas del sistema en dependencia de las entradas del usuario.

3.6 Pruebas unitarias y funcionales

Las pruebas unitarias permiten probar, como su nombre lo indica, cada unidad independiente del software. Actúan esencialmente sobre el código fuente y sobre los elementos básicos de la interfaz de cada módulo. Pressman (18) plantea que los casos de prueba que se generan durante las pruebas de unidad deben estar encaminados a verificar los siguientes elementos:

- **Interfaz:** "Se prueba la interfaz del módulo para asegurar que la información fluye de forma adecuada hacia y desde la unidad de programa que está siendo probada".
- **Estructuras de datos locales:** "Se examinan las estructuras de datos locales para asegurar que los datos que se mantienen temporalmente, conservan su integridad durante todos los pasos de ejecución del algoritmo".
- **Condiciones límites:** "Se prueban las condiciones límites para asegurar que el módulo funciona correctamente en los límites establecidos como restricciones de procesamiento".
- **Caminos independientes:** "Se ejercitan todos los caminos independientes (caminos básicos) de la estructura de control con el fin de asegurar que todas las sentencias del módulo se ejecutan por lo menos una vez".
- **Caminos de manejo de errores:** "Se prueban todos los caminos de manejo de errores".

Con las pruebas unitarias es posible aislar una parte del código de manera que pueda ser analizado. Un ejemplo de esto es evaluar las funciones o métodos, a los cuales se les realiza una entrada de datos para obtener los datos de salida correctos. Este tipo de pruebas validan la forma en la que las funciones y métodos trabajan en cada caso particular.

Las pruebas funcionales no solo validan la transformación de una entrada en una salida, sino que validan una característica completa. Por ejemplo un sistema de cache solamente puede ser validado por una prueba funcional, ya que comprende más de 1 solo paso: la primera vez que se solicita una página, se produce su código; la segunda vez se obtiene directamente de la cache. De modo que las pruebas funcionales validan procesos y requieren de un escenario. En Symfony se deberían crear pruebas funcionales para todas las acciones.

Estas pruebas simulan la navegación del usuario, realizan peticiones y comprueban los elementos de la respuesta tal y como lo haría manualmente un usuario para validar que una determinada acción hace lo

que se supone que debe hacer. En las pruebas funcionales se ejecuta un escenario correspondiente a lo que se denomina un "caso de uso" [12].

3.7 Herramientas para la realización de pruebas

Cuando se emplea php como lenguaje de programación existe diversidad de frameworks para crear pruebas unitarias y funcionales, siendo los más usados *PhpUnit* y *SimpleTest*. Symfony incluye su propio framework para pruebas unitarias llamado *Lime*. El mismo utiliza la librería Test::More de Perl y es compatible con TAP. Lo que significa que los resultados de las pruebas se muestran con el formato definido en el "Test Anything Protocol", creado para facilitar la lectura de los resultados de las pruebas. El *Lime* presenta además una serie de ventajas dentro de las cuales resaltan por su importancia las siguientes:

- Ejecuta los archivos de prueba en un entorno independiente para evitar interferencias entre las diferentes pruebas.
- Las pruebas de Lime son fáciles de leer y sus resultados también lo son. En los sistemas operativos que lo soportan, los resultados de Lime utilizan diferentes colores para mostrar de forma clara la información más importante.
- Está escrito con PHP, es muy rápido y está bien diseñado internamente. Consta únicamente de un archivo llamado lime.php y no tiene ninguna dependencia.

A pesar de tener todas estas ventajas *Lime* no resulta un framework suficiente para el proceso de prueba en muchas aplicaciones, ya que en caso que se necesite realizar validaciones del lado del cliente este framework de prueba no brinda la posibilidad de simular el comportamiento de Javascript, el cual es un lenguaje que es empleado en el proceso de validación.

Al no poder emplear Lime como framework de prueba debido a no permitir iteraciones con Javascript se decidió usar una nueva herramienta para desarrollar el proceso de prueba, es el caso de *Selenium*. La cual consiste en una herramienta para realizar pruebas funcionales. Simula la acción que puede realizar cualquier usuario desde un navegador con la ventaja de realizarla de una manera más rápida evitando el trabajo repetitivo de probar varias veces lo mismo. Además de ser capaz de ejecutar todo el código JavaScript de la página, incluidas las interacciones creadas con Ajax.

Las pruebas que se realizan con la herramienta Selenium se escriben en HTML, por lo que cada caso de prueba consiste en una página HTML, con una tabla de 3 columnas: comando, destino y valor. Esto se convierte en un proceso un poco tedioso pues habría que escribir todo el código. Para solucionar esto se emplea una extensión para Firefox denominada Selenium IDE, la cual consiste en un entorno de desarrollo integrado para las pruebas de *Selenium* que permite grabar, editar y depurar las pruebas. Algunas de las características de esta herramienta son:

- Fácil reproducción y grabación
- Guardar como HTML.
- Soporta extensiones Javascript.

3.8 Diseño de prueba

Con el objetivo de probar el correcto funcionamiento de la aplicación se realizaron algunas pruebas funcionales empleando la herramienta Selenium IDE. A continuación se relacionan algunos de los casos de prueba implementados los cuales consisten en:

Chequear el funcionamiento de la autenticación del sistema.
Chequear la validación de uno de los formularios del sistema.

Tabla 3 Caso de Prueba 1

Caso de prueba	Autenticar Usuario
Condiciones	El usuario debe estar registrado en la base de datos general. Los datos usuario y contraseña introducidos deben ser los mismos que los registrados en la base de datos general.
Datos de entrada	Usuario y contraseña.
Resultados esperados	1. Si los datos introducidos coinciden con los registrados en la base de datos general, el sistema debe mostrar la página de bienvenida mostrando los datos del usuario que acaba de registrarse.
	2. Si alguno de los datos introducidos es incorrecto el sistema debe mostrar un texto de error y brindar la posibilidad de volver a ingresar los datos.
Resultados obtenidos	1. Los resultados obtenidos fueron satisfactorios. Al introducir datos correctos el sistema redirecciona a la página de bienvenida mostrando la información referente

	al usuario que se registró.
	2. Los resultados obtenidos fueron satisfactorios. Al introducir datos incorrectos, el sistema muestra un mensaje de error donde le informa al usuario que por favor verifique sus datos brindando la posibilidad de ingresarlos nuevamente.
Observaciones	El empleo de Selenium permitió realizar pruebas al sistema de forma satisfactoria obteniendo en todos los casos los resultados esperados.

Tabla 4 Caso de Prueba 2

Caso de prueba	Asignar Rol
Condiciones	El usuario debe estar registrado en la base de datos, al igual que los roles deben estar definidos.
Datos de entrada	Nombre de usuario y rol.
Resultados esperados	1. Si el nombre del usuario es correcto y a este no se le ha asignado con anterioridad el rol seleccionado el sistema debe permitir asignar el rol y mostrar un mensaje que indique la correcta ejecución de la acción.
	2. Si el nombre del usuario es incorrecto el sistema debe mostrar un mensaje donde indique esto. 3. Si el usuario ya tiene asignado el rol seleccionado el sistema debe mostrar un mensaje donde informa sobre esto.
Resultados obtenidos	1. Los resultados obtenidos fueron satisfactorios. Al introducir datos correctos el sistema muestra el mensaje donde informa la correcta ejecución de la función.
	2. Los resultados obtenidos fueron satisfactorios. Al introducir un nombre de usuario incorrecto el sistema muestra un mensaje donde informa que dicho usuario no es correcto. 3. Los resultados obtenidos fueron satisfactorios. Al tratar de asignar al usuario un rol que ya tiene el sistema muestra un mensaje donde informa que dicho usuario ya tiene asignado el rol.
Observaciones	El empleo de Selenium permitió realizar pruebas al sistema de forma satisfactoria obteniendo en todos los casos los resultados esperados.

Tabla 5 Caso de prueba 3

Caso de prueba	Realizar Encuesta
Condiciones	El usuario debe estar registrado en la base de datos, y no puede haber realizado con anterioridad la encuesta.
Datos de entrada	Respuesta de cada pregunta de la encuesta.
Resultados esperados	1. Si el usuario analizó cada pregunta de la encuesta y respondió correctamente el sistema le informa al usuario cual es su clasificación en cuanto al estilo de aprendizaje.
	2. Si el usuario escoge la misma respuesta para cada una de las preguntas el sistema le informa que es incorrecto y brinda la posibilidad de hacer nuevamente la encuesta. 3. Si el usuario envía la misma respuesta igual cantidad de veces en cada uno de los grupos en los que se divide la encuesta el sistema informa que es incorrecto y brinda la posibilidad de realizar nuevamente la encuesta.
Resultados obtenidos	1. Los resultados obtenidos fueron satisfactorios. Al realizar correctamente la encuesta el sistema informa al usuario cual es su clasificación en cuanto a estilo de aprendizaje.
	2. Los resultados obtenidos fueron satisfactorios. Al enviar la encuesta con la misma respuesta en cada una de las preguntas el sistema informa al usuario que es incorrecto brindando la posibilidad de realizar nuevamente dicha encuesta. 3. Los resultados obtenidos fueron satisfactorios. Al enviar la encuesta con la misma respuesta igual número de veces en cada uno de los grupos en los que se divide la encuesta el sistema informa al usuario que es incorrecto brindando la posibilidad de realizar nuevamente dicha encuesta.
Observaciones	El empleo de Selenium permitió realizar pruebas al sistema de forma satisfactoria obteniendo en todos los casos los resultados esperados.

3.9 Conclusiones

En este capítulo se hace referencia al proceso de automatización de pruebas. Se aborda el tema de las pruebas unitarias y funcionales, así como los elementos más importantes de cada una de estas. Además se exponen algunas de las características del framework para pruebas unitarias con que dispone Symfony, explicando el por qué no se emplea este y en su lugar se escoge Selenium, con el cual se

realizaron dos casos de pruebas que permitieron a partir de los resultados obtenidos comprobar el correcto funcionamiento del sistema.

CONCLUSIONES

Durante el desarrollo del presente Trabajo de Diploma se han cumplido los objetivos trazados y las tareas previstas, obteniéndose los siguientes resultados. Antes de la implementación del módulo Gestión del Aprendizaje, de este trabajo se hizo necesario realizar un estudio valorativo y crítico del diseño precedente y a partir del mismo hacer las modificaciones pertinentes. Para implementar las funcionalidades se estudiaron los contenidos teóricos referentes a las características de las posibles herramientas a usar, así como los Estándares de Codificación, lo cual permitió realizar una selección que se ajustara a las características del sistema a desarrollar. Además se validaron las funcionalidades en el producto para comprobar que cumpliera con los requerimientos funcionales, lográndose una aplicación de fácil manejo, con una interfaz visual amigable y sencilla. La ejecución de las tareas en el sistema obtenido ocurre de forma aceptable. Mediante el análisis de los resultados se llega a la conclusión de que la puesta en práctica del sistema permitirá facilitar el proceso de autoaprendizaje en los estudiantes, aumentando la gestión del mismo.

Recomendaciones

En el transcurso de este trabajo se cumplió con cada uno de los objetivos propuestos logrando una implementación adecuada para los módulos desarrollados, lo que permitió enriquecer el sistema por lo cual el autor de este trabajo recomiendo algunas actividades para darle continuidad al mismo:

- Continuar con la implementación del resto de los módulos existentes con el fin de aumentar las posibilidades del sistema.
- Aplicar tecnologías novedosas sobre la base de los controles personalizados en el perfeccionamiento de los elementos de la interfaz visual del sistema propuesto.
- Agregar nuevas funcionalidades que puedan surgir a partir de la realización de estudios referentes a las características del actual sistema educacional.

REFERENCIAS BIBLIOGRÁFICAS

1. **Castillo González, Reinier y De la Rosa Rodríguez, José M. "***Módulo Gestión del Aprendizaje del Centro Virtual de Autoaprendizaje de Lenguas Extranjeras***".** Trabajo para optar por el título de Ingeniero en Ciencias Informáticas. Universidad de las Ciencias Informáticas (UCI). Ciudad de la Habana, 2008. 131 pág.
2. **Sanz-Lumbier, S**., et al. (2002). *Hezinet: The Hypermedia System that Makes the Basque Language Easy to Learn. 5th International Conference: Computer and Advance Technology in Education (LASTED)*. Cancún (México). Acta Press.
3. **VanLehn, K.** *The Behavior of Tutoring Systems*. International Journal of Artificial Intelligence in Education, 16:227-265, 2006.
4. **CELLA.** Comprehensive English Language Learning Assessment. Consultado en: http://www.ets.org (23-01-2009)
5. **Archer, Tom y Whitechapel, Andrew.** *LA BIBLIA DE MICROSOFT VISUAL C++ .NET***.** s.l.: ANAYA, 2003. 84-415-1487-9.
6. **SOLANO, R**. *Teoría de Sistemas.*
7. **Müller, Peter.** *Introducción a la Programación Orientada a Objetos Empleando C++.* Globewide Network Academy (GNA), Berlin, Germany.

   Consultado en: http://www.desy.de/gna/html/cc/Tutorial/Spanish/tutorial.html (17-02-2009).
8. **Musciano Chuck y Kemedy Bill.** *HTML La Guía Completa.* McGRAW-HILL INTERAMERICANA EDITORES, S.A. de C.V. ISBN 970-10-2141-X. Consultado en: http://bibliodoc.uci.cu/pdf/reg01313.pdf (19-03-2009).
9. *On SGML and HTML.* Disponible en*:* www.w3.org/TR/html401/intro/sgmltut.html#h-3.2.2.
10. *Differences with HTML 4.*Disponible en: www.w3.org/TR/xhtml1/diffs.html#h-4.4.
11. **Gutiérrez, J.J**., *¿Qué es un framework web?*
12. **Fabien Potencier, F.Z**., *Symfony la guía definitiva*. 2008. 435.

13. **TRAMUÑAS, J**. Introducción a la Documática. *Los sistemas de bases de datos y los SGBD*. DL HU-159-2002. ed. Zaragoza: KRONOS, Disponible en: http://tramullas.com/documatica/2-4.html
14. **CaFeCONF**, *Migración a PostgreSQL desde otras bases de datos.* 2005.
15. **Figueredo Guzmán, Yulien y Sao Aballí Yenisleydis**. "*SUBSISTEMA DE NÓNIMA, EVALUACIÓN DE DESEMPEÑO Y ENTREGA DE MATERIALES EN LA FACULTAD#9*". Trabajo para optar por el título de Ingeniero en Ciencias Informáticas. Universidad de las Ciencias Informáticas (UCI). Ciudad de la Habana, 2008. 114 pág.
16. **Potencier, Fabien y Zaninotto, François.** *Symfony la guía definitiva***.** 2008. 435 pág.
17. **Díaz, J. F.** *Técnicas de Diseño de Programas. 2008.*
    Disponible en: http://www.galeon.com/neoprogramadores/t_dis_sw.htm#Modular
18. **Pressman, Roger S**. *Ingeniería de Software. Un enfoque práctico.* s.l: Mc Graw-Hill/Interamericana de España, S. A, 2002.

BIBLIOGRAFÍAS

1. **López Cuadrado, Javier. Sanz, Sara. Villamañe, Maikel. Sanz, Silvia**. *Utilización de Tests Adaptativos para la Evaluación en un Sistema Educativo*. Departamento de Lenguas y Sistemas Informáticos. España. 6 pág.
2. **Castillo González, Reinier y De la Rosa Rodríguez, José M. "***Módulo Gestión del Aprendizaje del Centro Virtual de Autoaprendizaje de Lenguas Extranjeras***".** Trabajo para optar por el título de Ingeniero en Ciencias Informáticas. Universidad de las Ciencias Informáticas (UCI). Ciudad de la Habana, 2008. 131 pág.
3. **García Ruiz, Ángel Miguel.** *Aplicaciones de la Realidad Virtual e Inteligencia Artificial en la Enseñanza del Idioma Inglés***.** Facultad de Telemática, Universidad de Colima. México. 6 pág.
4. **Archer, Tom y Whitechapel, Andrew.** *LA BIBLIA DE MICROSOFT VISUAL C++ .NET***.** s.l.: ANAYA, 2003. 84-415-1487-9.
5. **Cuena, J**. *Lógica Informática*. Editorial Alianza, 1985.
6. **Alpuente Prentice, P. Julián, M. y Hall, Pearson**. *Programación Lógica. Teoría y Práctica.* 2007.
7. **Gosling James, Joy Bill, Steele Guy, y Bracha Gilad.** *The Java language specification.* Tercera edición**.** Addison-Wesley, 2005**.** ISBN 0-321-24678-0.
8. **Müller, Peter.** *Introducción a la Programación Orientada a Objetos Empleando C++.* Globewide Network Academy (GNA), Berlin, Germany.
9. **Lindholm Tim y Yellin Frank.** *The Java Virtual Machine specification***.** Segunda edición**.** Addison-Wesley, 1999. ISBN 0-201-43294-3**.**
10. **Peter Abel.** *IBM PC Assembly Language and Programming***.** Fourth Edition. Prentice Hall. 1997.
11. **Musciano Chuck y Kemedy Bill.** *HTML La Guía Completa.* McGRAW-HILL INTERAMERICANA EDITORES, S.A. de C.V. ISBN 970-10-2141-X. Consultado en:

    http://bibliodoc.uci.cu/pdf/reg01313.pdf (19-03-2009).
12. *On SGML and HTML.* Disponible en*:* www.w3.org/TR/html401/intro/sgmltut.html#h-3.2.2.
13. *Differences with HTML 4.*Disponible en: www.w3.org/TR/xhtml1/diffs.html#h-4.4.
14. **Gutiérrez, J.J**., *¿Qué es un framework web?*

15. **Fabien Potencier, F.Z**., *Symfony la guía definitiva.* 2008. 435.

16. **TRAMUÑAS, J**. Introducción a la Documática. *Los sistemas de bases de datos y los SGBD*. DL HU-159-2002. ed. Zaragoza: KRONOS, Disponible en: http://tramullas.com/documatica/2-4.html

17. **CaFeCONF**, *Migración a PostgreSQL desde otras bases de datos.* 2005.

18. **Figueredo Guzmán, Yulien y Sao Aballí Yenisleydis**. "*SUBSISTEMA DE NÓNIMA, EVALUACIÓN DE DESEMPEÑO Y ENTREGA DE MATERIALES EN LA FACULTAD#9*". Trabajo para optar por el título de Ingeniero en Ciencias Informáticas. Universidad de las Ciencias Informáticas (UCI). Ciudad de la Habana, 2008. 114 pág.

19. **Potencier, Fabien y Zaninotto, François.** *Symfony la guía definitiva***.** 2008. 435 pág.

20. **Díaz, J. F.** *Técnicas de Diseño de Programas. 2008.* Disponible en: http://www.galeon.com/neoprogramadores/t_dis_sw.htm#Modular