Universidad de las Ciencias Informáticas

FACULTAD # 3

Título: Optimización del acceso a datos del Tutor Virtual de Evaluación del Aprendizaje Autónomo de Idiomas.

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor: René Rubio Cruz

Tutor(es): MSc. Yoan Martínez Márquez
Ing. Reinier Castillo González

Co-tutor: Ing. Yusmary Companioni Sardiña
Lic. Moisés A. Mayet Solano

Ciudad de La Habana, Junio de 2011
Año 53 de la Revolución

DECLARACIÓN DE AUTORÍA

Declaro ser 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 _____.

_____

René Rubio Cruz
Autor

_____

Ing. Reinier castillo González
Tutor

_____

MSc. Yoan Martínez Márquez
Tutor

_____

Ing. Yusmary Companioni Sardiña
Co-Tutor

_____

Lic. Moisés Alain Mayet Solano
Co-Tutor

DATOS DE CONTACTO

Tutor: MSc. Yoan Martínez Márquez

Especialidad de graduación:

- Licenciado en Educación. Especialidad: Lengua Inglesa. 2004.
- Máster en Ciencias de la Educación. Especialidad: Tecnologías en los Procesos Educativos. 2007

Correo electrónico: yoanm@uci.cu

Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba

Tutor: Ing. Reinier Castillo González

Especialidad de graduación: Ingeniería en Ciencias Informáticas

Correo electrónico: rcgonzalez@uci.cu

Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba

Co-Tutor: Ing. Yusmary Companioni Sardiña

Especialidad de graduación: Ingeniería en Ciencias Informáticas

Correo electrónico: ycompanioni@uci.cu

Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba

Co-Tutor: Lic. Moisés Alain Mayet Solano

Especialidad de graduación: Licenciado en Ciencia de la Computación

Correo electrónico: mmayets@uci.cu

Universidad de las Ciencias Informáticas, Ciudad de la Habana, Cuba

DEDICATORIA

A toda mi familia, en especial a mis padres, mis hermanas y mis abuelos.

A la memoria de mi abuelo René.

A mis amigos.

AGRADECIMIENTO

A mi mamá y a mi papá por ser mi apoyo en todo momento y lo más grande en mi vida.

A mis abuelos por ser los mejores abuelos del mundo.

A mis hermanas por su amor y ternura.

A Odelaisy por ser como mi otra madre y a Kiki por ser como mi padre.

A mis tías que con su amor me han servido de apoyo en todo momento.

A mis primos por compartir conmigo cada momento de mi vida.

A toda mi familia en general.

A Mairelys y a Yusmary por ser como mis hermanas desde hace muchos años.

A Tahimi por ser mi negrita linda a lo largo de todo este tiempo.

A Yaimara, Yelina, Julio y Alejandro que aunque llegaron tiempo después también son como hermanos para mi.

A mi grán amigo Yidian, por estar conmigo en las buenas y malas.

A Reinier, a Alejandro y a Aníbal por apoyarme en momentos difíciles.

A mis compañeros de VIRTEVALL por su ayuda incondicional en el desarrollo de la tesis.

A mis tutores, por ser los mejores… Eternamente Agradecido…

Al resto de mis compañeros a lo largo de la vida universitaria… a los del antiguo grupo de la facultad 4 como los del nuevo grupo de la facultad 3.

A mis compañeros de 1er año, Liván, Adolfo, Rodolfo, Lian por los grandes momentos que vivimos juntos.

A mis profesores a lo largo de estos años, que me han ayudado a formarme como profesional.

A Belkis y Laura por ser como madres para mi dentro de la escuela, Gracias por su amor, ayuda y comprensión.

A todos los que han estado a mi lado en todos estos años, a los que aún están y a los que ya no están también.

RESUMEN

El proceso de formación 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 aprendizaje. La inclusión de las potencialidades de las Tecnologías de la Información y las Comunicaciones (TICs) ha dado paso al surgimiento de nuevos métodos como el autoaprendizaje, para lo cual han surgido recursos como los espacios virtuales, aulas virtuales, tutores virtuales, entre otros.

La UCI es un centro que está a la vanguardia de todo el proceso de enseñanza y autoaprendizaje es por ello que surge el Proyecto de Innovación Pedagógica: Metodología de Evaluación para el Aprendizaje Autónomo de Idiomas (VIRTEVALL), con el cual se prevé crear un Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas que funcione como un ambiente alternativo de evaluación para aprendizaje autónomo de Idiomas, que le permite a los estudiantes trabajar de manera independiente en sus debilidades y profundizar en los temas que son de su interés, a través de actividades evaluativas personalizadas.

La presente investigación persigue como objetivo optimizar el acceso a datos del Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas, para lo cual se ha realizado un estudio sobre las principales deficiencias que presentaba el sistema en cuanto al acceso a datos y las diferentes formas de solucionarlas.

Para darle cumplimiento al objetivo de la investigación fue necesario realizar una valoración crítica del diseño propuesto por los analistas, teniendo en cuanta los requisitos de la aplicación, se hizo un estudio acerca de los componentes a reutilizar y los algoritmos no triviales a implementar, para finalmente describir las nuevas clases y operaciones necesarias para darle solución al objetivo trazado en la presente investigación. Para la validación, se aplicaron métricas al diseño y técnicas de prueba de software que permitieron comprobar la validez de la solución propuesta. Se detallaron los casos de prueba, que permitieron detectar la ocurrencia de errores en la aplicación y la posterior solución de los mismos.

PALABRAS CLAVE

“Tutor Inteligente, Acceso a Datos, Base de Datos, Optimización.”

ÍNDICE

ÍNDICE DE FIGURAS

INTRODUCCIÓN

En la actualidad la información es uno de los elementos fundamentales de la vida diaria, y la necesidad de almacenarla y recuperarla es una constante casi invariable. Con el decursar de los años los mecanismos utilizados para transferir la información desde su origen hasta la aplicación, o viceversa, han sido diversos. Con el advenimiento de las nuevas tecnologías, el más empleado ha sido la base de datos, encargada de almacenar un conjunto de datos relacionados con un asunto, tema o actividad específicos. Las bases de datos son utilizadas para cosas tan sencillas como mantener un registro de una colección de discos de música, hasta llevar toda la gestión de una gran empresa u organización. Así, fue surgiendo la necesidad de almacenar grandes volúmenes de información y con ello la creación de Sistemas Gestores de Bases de Datos (SGBD), los cuales permiten el almacenamiento de datos de forma organizada, accesible y segura.

En la Universidad de Ciencias Informáticas existen proyectos productivos organizados por centros de producción, en los cuales se trata constantemente con grandes volúmenes de datos. Tal es el caso del Centro de Gobierno Electrónico (CEGEL), el cual cuenta con numerosos proyectos productivos. Estos proyectos se caracterizan por construir sistemas que manipulan numerosa cantidad de datos. Para ello hacen uso de Sistemas Gestores de Base de Datos tales como PostgresSQL, extrayendo los datos de las diferentes bases de datos a las que se encuentra conectado cada sistema y de las cuales necesita la información adquirida durante el proceso productivo.

En el Centro de Gobierno Electrónico se desarrolla el Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas (VIRTEVALL), el cual es un Sistema Tutor Inteligente (STI) que está estructurado sobre la base del sustento teórico provisto por una tesis doctoral sobre evaluación para aprendizaje autónomo de idiomas extranjeros. Con la puesta en marcha de este proyecto se prevé crear un ambiente alternativo de evaluación para el aprendizaje autónomo de idiomas extranjeros, que les permita a los estudiantes trabajar de manera independiente en sus debilidades y profundizar en temas que sean de su interés en el aprendizaje autónomo de idiomas.

Un aspecto fundamental de los Sistemas Tutores Inteligentes es el tratamiento de la información que se va obteniendo con el desarrollo de la aplicación. Actualmente el Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas almacena la información que se construye en el proceso de evaluación del aprendizaje de los usuarios del sistema en una base de datos. De esta forma hace uso de este recurso para almacenar la información referente al proceso de evaluación del aprendizaje de los usuarios, además de la información correspondiente a cada uno de ellos. Esta es accedida a través

de la capa de acceso a datos del sistema mediante sentencias SQL previamente elaboradas en el modelo, constituyendo esto una dificultad para la optimización de la velocidad de tratamiento de información, dado a que las respuestas a las peticiones de los usuarios no dependen solo del control del gestor de bases de datos. Lo que trae consigo que el procesamiento de requisitos del manejador de base de datos sea más lento, dificulta la modularización de la aplicación, además de que los datos pueden ser alterados por el uso de sentencias del lenguaje defectuosas o erróneas. En la base de datos se almacenan además información referente a los contextos o recursos didácticos de la aplicación los que son utilizados en las actividades propuestas al usuario o estudiante. Este aspecto es considerado como una dificultad del sistema debido a que la información referente a dichos recursos no está sujeta a cambios, o sea, es estática, y se hace necesario acceder a la misma haciendo constantes peticiones al motor de bases de datos, produciendo la ineficiencia en el tiempo de respuesta de la base de datos así como la sobrecarga de la misma.

A partir de lo expuesto anteriormente se presenta como **problema a resolver**: ¿Cómo optimizar el manejo de datos del Tutor Virtual de Evaluación del Aprendizaje Autónomo de Idiomas a partir de la ineficiencia de los tiempos de respuesta y la sobrecarga de la base de datos de dicho sistema?

Por tanto, se define como **objeto de estudio:** Gestión de la información almacenada en las bases de datos.

Con la finalidad de dar solución al problema expuesto se plantea el siguiente **objetivo general**:

Desarrollar funciones, mecanismos de persistencia y serialización de datos que contribuyan a la optimización del manejo de contextos e información del Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas.
Definiendo como **campo de acción**: Optimización del acceso a la información en bases de datos para STI.

Para darle cumplimiento al objetivo propuesto se plantea la siguiente **idea a defender**: Si se desarrollan las funciones y se implementan los mecanismos de serialización y persistencia de datos entonces se eliminará la ineficiencia en el tiempo de respuesta del Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas.

Para alcanzar el objetivo propuesto y darle solución a dicha problemática se trazan las siguientes **tareas de investigación**:

1. Análisis de la bibliografía en lo referente a los temas de bases de datos, gestores de base de datos, implementación de funciones y de mecanismos de persistencia y serialización de datos para la fundamentación del marco teórico de la investigación.
2. Estudio y selección de las diferentes herramientas de desarrollo y lenguaje de programación para el desarrollo de la solución propuesta.
3. Definición de las funciones a implementar en la base de datos del STI-VIRTEVAL.
4. Diseño y modelado arquitectónico de las funcionalidades necesarias para la implementación de mecanismos de serialización y persistencia de datos.
5. Implementación de las funciones a partir de la definición anterior.
6. Implementación de las funcionalidades basadas en el diseño propuesto para los mecanismos de serialización y persistencia de datos.
7. Realización de pruebas para la validación del correcto funcionamiento de las funcionalidades implementadas en la aplicación.

Métodos teóricos:

Analítico-sintético: Con el objetivo de obtener las características, principales ventajas y buenas prácticas para la implementación de funciones y mecanismos de persistencia y serialización de datos, para lo cual se analizan las distintas herramientas existentes, y documentos acerca del tema, luego seleccionar los elementos más importantes y que están más relacionados con el objeto de estudio planteado.

Análisis histórico lógico: Con el fin de investigar sobre la optimización de las bases de datos e implementación de los mecanismos de persistencia y serialización de datos. Además de analizar el avance que han alcanzado, así como las ventajas y desventajas del empleo de estas prácticas.

Inductivo-Deductivo: En la revisión, estudio y justificación de las prácticas utilizadas en la gestión y almacenamiento de información del Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas para así determinar las posibles mejoras y métodos a emplear.

Métodos empíricos:

Entrevista a especialista: En procesos de almacenamiento de datos para recoger toda la información posible acerca de las diferentes formas de almacenamiento de datos y manipulación de los mismos.

EL Trabajo de Diploma estará estructurado de la forma siguiente:

- ✓ **Capítulo 1:** Fundamentación teórica: Se plantean todos los elementos que sustentan el

desarrollo y solución del problema a resolver. Se describen las principales características de las bases de datos, gestores de base de datos, consultas SQL, funciones, además se definen los principales conceptos asociados a las herramientas a utilizar en el desarrollo de las aplicaciones y se exponen sus características como por ejemplo las tecnologías a utilizar, la metodología, el lenguaje de programación entre otras.

- ✓ **Capítulo 2:** Descripción de la solución: Basándose en el análisis de herramientas, tecnologías y tendencias de desarrollo actuales, se realiza la propuesta de solución para dar cumplimiento al objetivo planteado, definiendo el alcance de la aplicación. En este capítulo se hace la propuesta del diseño para las nuevas funcionalidades a incorporar en la aplicación, se describen las principales clases que serán incorporadas así como una evaluación del diseño propuesto a partir de las métricas propuestas por Calisoft. También se plantean los patrones de diseño, los estándares de codificación, el modelo de datos de la aplicación.

- ✓ **Capítulo 3:** Validación de la solución propuesta: Este capítulo contendrá todos los resultados obtenidos de las diferentes comprobaciones que se le realizaran a la solución que se obtuvo. En el mismo se mostrarán los diferentes casos de pruebas a realizar en la aplicación y los resultados de los mismos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción.

En el capítulo se realiza un análisis de los aspectos relacionados con el Sistema Tutor Inteligente de Evaluación para el Aprendizaje Autónomo de Idiomas haciendo mayor énfasis en la capa de acceso a datos y la base de datos. Se analizan los principales conceptos y definiciones asociados al dominio del problema que son necesarios para una mejor comprensión de los términos que se emplean a lo largo del trabajo de diploma. Además se hace una caracterización de las herramientas, entorno de desarrollo, lenguaje de programación, framework y alternativas para la optimización de la capa de acceso a datos.

1.1 Conceptos asociados al dominio del problema.

Para lograr una mejor comprensión y desenvolvimiento de las áreas temáticas que se abordarán en el presente y los posteriores capítulos, se mostrarán una serie de términos identificados durante la investigación.

1.1.1 Aprendizaje Autónomo.

Es la facultad de una persona para dirigir, controlar, regular y evaluar su forma de aprender de manera consciente e intencionada haciendo uso de Estrategias de Aprendizaje. (EEES, 2009)

1.1.2 Gestión del Aprendizaje.

Es la actividad que permite generar, compartir o distribuir y utilizar el conocimiento táctico (know-how) y explícito (formal) existente en un determinado espacio, para que los individuos y las comunidades lo apliquen cuando deben enfrentar sus necesidades de desarrollo. Específicamente se pudiera decir que es la acción que una persona realiza cuando desea cumplir una meta determinada dentro del proceso de aprendizaje autónomo. (Castillo, 2008)

1.1.3 Estilos de Aprendizaje.

Es el término que se refiere al hecho de que cada persona utiliza su propio método o estrategias a la hora de aprender. Aunque las estrategias varían según lo que se quiere aprender, cada uno tiende a desarrollar ciertas preferencias o tendencias globales. Esas preferencias o tendencias a utilizar, constituyen nuestro estilo de aprendizaje. Se resumen en tres estilos de aprendizaje: estilo visual, estilo auditivo y estilo táctil kinestésico. (Villanueva., 1997)

1.1.4 Sistema de bases de datos.

Los sistemas de BD constituyen una de las herramientas más difundidas en la actual sociedad de la información, son utilizadas para la recuperación y almacenamiento de la información que manejan las empresas y diversos organismos ya sean dedicado a la cultura, educación o las ciencias en general.

Surgen con el objetivo de resolver los problemas que planteaban los sistemas de ficheros, tales como la redundancia de datos y la dependencia entre programas y datos. Una BD puede definirse como: "un conjunto exhaustivo no redundante de datos estructurados organizados independientemente de su utilización y su implementación en máquinas accesibles en tiempo real". (J., 2002)Para facilitar el trabajo con la BD existen los Sistemas de Gestión de Base de Datos (SGBD), estos sistemas representan: "un tipo de *software* muy específico, dedicado a servir de interfaz entre la BD, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta". (VALDÉS., 2000).

Un SGBD es un software que permite definir, crear y mantener las BD, además proporciona un acceso controlado a estas y un lenguaje de manejo de datos para la inserción, actualización, eliminación y consulta de información en estas BD. Los SGBD se dividen en tres generaciones. La primera generación de SGBD comprende a los sistemas jerárquicos y de red. El modelo relacional da surgimiento a la segunda generación, los sistemas de bases de datos relacionales, siendo este el más utilizado y extendido en la actualidad. La tercera generación de los SGBD se encuentra representada por el modelo relacional extendido y el modelo orientado a objetos. Estos sistemas presentan una serie de ventajas. Algunas de estas ventajas son el control de la redundancia, la consistencia de datos, la mejora en los aspectos de seguridad y la integridad. Algunos de sus inconvenientes son su coste al requerir más capacidad de almacenamiento donde en la mayoría de los casos se dedica exclusivamente una computadora para el SGBD; al estar todo la información centralizada hace que presente vulnerabilidad ante los fallos que puedan producirse. (VALDÉS., 2000).

Un Sistema Gestor de Base de Datos está compuesto por:

- ✓ **Un lenguaje de definición de datos** (DDL, por sus siglas en inglés): Mediante este lenguaje el SGBD identifica las descripciones de los elementos contenidos en los esquemas y almacena dicha descripción en los catálogos del sistema.
- ✓ **Lenguaje de Manipulación de Datos** (DML, por sus siglas en inglés): Permite la manipulación de las operaciones de Inserción, Eliminación y Modificación.

- ✓ **Un Lenguaje de Consulta Estructurado** (SQL, por sus siglas en inglés): Es un lenguaje declarativo de acceso a base de datos relacionales que permite especificar diversos tipos de operaciones con las mismas. Una de sus características es el manejo del Álgebra Relacional y Cálculo Relacional, permitiendo lanzar consultas con el fin de recuperar de forma sencilla información de interés en una base de datos, así como realizar cambios sobre las mismas.

- ✓ **Un Lenguaje de Control de Datos** (DCL, por sus siglas en inglés): Con el nombre de lenguaje de control de datos se hace referencia a la parte del lenguaje SQL que se ocupa de los apartados de seguridad y de la integridad en el procesamiento concurrente. Definición de usuarios mediante un mecanismo determinado, que puede ser específico de cada SGBD. En sistemas multiusuario puede usarse el propio usuario del Sistema Operativo, o bien definir usuarios propios del SGBD.

1.1.5 Consultas SQL.

Una consulta SQL[1] es una expresión que define un comando de SQL. Las instrucciones o cadenas de SQL se utilizan en consultas y en funciones de agregado.

El lenguaje SQL consta de unas treinta sentencias. Cada sentencia demanda una acción específica por parte del SGBD tal como la creación de una nueva tabla, la recuperación de datos o la inserción de nuevos datos en la BD. Todas las sentencias SQL tiene la misma forma básica. Puede utilizar el Lenguaje de Consulta Estructurado (SQL o Structured Query Language) para consultar, actualizar y administrar bases de datos relacionales.

Todas las sentencias comienzan con un verbo, (una palabra clave que describe lo que la sentencia hace: CREATE, INSERT, DELETE, SELECT.). La sentencia continua con una o más cláusulas. Una cláusula puede especificar los datos sobre los que debe actuar la sentencia, o proporcionar más detalles acerca de lo que la sentencia se supone que hace. Todas las cláusulas también comienzan con una palabra clave (ejemplos: WHERE, FROM, INTO). Algunas cláusulas son necesarias, otras opcionales.

Las consultas permiten trabajar directamente con las tablas del servidor en lugar de hacer que el motor de base de datos procese los datos. (J., 2002)

1.1.6 Funciones.

Es un programa (o procedimiento) el cual es almacenado físicamente en una base de datos. Su implementación varía de un manejador de bases de datos a otro. El uso de las funciones tiene como

[1] *Lenguaje Estructurado de Consultas(Structured Query Lenguaje)(SQL), por sus siglas en inglés*

ventaja que al ser ejecutada, en respuesta a una petición de usuario, es ejecutada directamente en el motor de bases de datos, el cual usualmente corre en un servidor separado. Como tal, posee acceso directo a los datos que necesita manipular y sólo necesita enviar sus resultados de regreso al usuario, deshaciéndose de la sobrecarga resultante de comunicar grandes cantidades de datos salientes y entrantes y reduciendo el tiempo de respuesta de la base de datos.

Las funciones pueden ser ventajosas, cuando una base de datos es manipulada desde muchos programas externos. Al incluir la lógica de la aplicación en la base de datos utilizando funciones, la necesidad de embeber la misma lógica en todos los programas que acceden a los datos es reducida. Esto puede simplificar la creación y, particularmente, el mantenimiento de los programas involucrados datos (Frabicio).

1.1.7 Capa de Acceso a datos.

Una capa de acceso a datos es una capa de un programa de ordenador que proporciona simplificado el acceso a los datos almacenados en el almacenamiento persistente de algún tipo, como una entidad relacional de bases de datos. Además, es esta capa la que vincula directamente a la aplicación con la BD y es la base de todas las demás capas superiores. (Katiuska, 2005)

1.1.8 ORM (object-relational mapping).

El ORM es un componente de software que permite trabajar con los datos persistidos como si ellos fueran parte de una base de datos orientada a objetos (en este caso virtual). Debido a que se ha estandarizado el trabajo con las bases de datos relacionales, se deben realizar operaciones que permitan transformar un registro en objeto y viceversa. A esta funcionalidad se la llama Mapeo objeto-relacional (ORM).

1.1.9 Persistencia de los datos almacenados.

Se le denomina persistencia de datos a la propiedad de los datos que sobreviven a la ejecución del programa que los ha creado. Es la vía que tiene el programador para que sus datos se conserven al finalizar la ejecución de un proceso, de forma que se puedan reutilizar en otros procesos Sin esta capacidad, los datos solo existen en memoria RAM, y se pierden cuando la memoria pierde energía, como cuando se apaga el computador, o se interrumpe la conexión a la base de datos donde se encuentran almacenados.

1.1.10 Datos persistentes.

Los Datos persistentes es la información guardada de manera permanente, que perdura en el tiempo, almacenada en algún medio físico, para en un momento determinado poder consultar esa información. Los datos persistentes son de gran ayuda a la hora de almacenar gran volumen de información en cualquier formato que luego nos permita extraerla de forma eficiente.

1.1.11 Serialización de datos.

La Serialización es el proceso de convertir una estructura de datos o un objeto en una secuencia de bits que pueden ser almacenados en un archivo, un buffer de memoria, o transmitir a través de un enlace de conexión de red y ser utilizado más adelante en el mismo u otro entorno informático.

La Serialización tiene una serie de ventajas:

- Un método de persistencia de objetos que es más conveniente que escribir sus propiedades a un archivo de texto en el disco, y volver a montar a la lectura de este.
- Un método de emisión de llamadas a procedimiento remoto, por ejemplo, como en SOAP.
- Un método para detectar cambios en variables en el tiempo los datos.

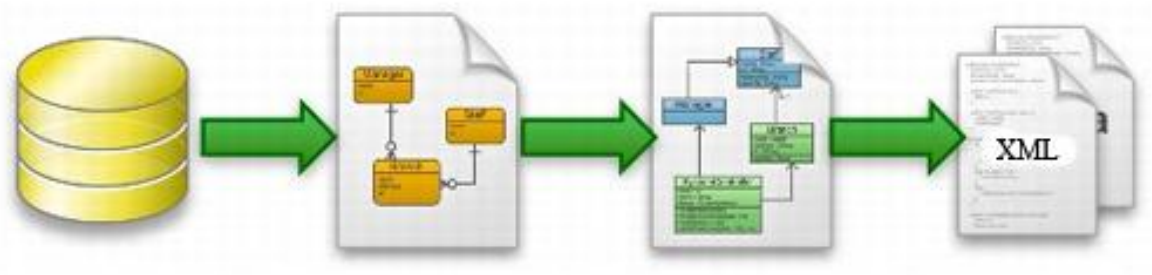


Ilustración 1: Esquema de persistencia de datos

1.1.12 Motores de persistencia.

La filosofía de imponer un único modelo teórico (formato de datos) a toda la aplicación, sufre un gran número de inconvenientes. En el caso de que toda la aplicación siga el modelo relacional, se perderían las ventajas de la orientación a objetos. En el caso de que toda la aplicación siga el modelo orientado a objetos, implica que las BD sean inmaduras y tengan un bajo nivel de estandarización. Por otro lado, si la aplicación sigue la lógica orientada a objetos y la BD es relacional, lo que, en principio, constituye la

opción más natural, plantea el problema de cómo conseguir que dos componentes con formatos de datos muy diferentes puedan comunicarse y trabajar conjuntamente. La solución se basa en encontrar un elemento intermedio que permita comunicar aplicación y BD, a este elemento se le conoce como "capa de persistencia" o "motor de persistencia".

El motor de persistencia traduce entre los dos formatos de datos: de registros a objetos y de objetos a registros. La situación se ejemplifica en la **Ilustración 1**. Cuando el programa necesita guardar un objeto llama al motor de persistencia, que traduce el objeto a registros y llama a la BD para que guarde estos registros. De la misma manera, cuando el programa necesita recuperar un objeto, la BD recupera los registros correspondientes, los cuales son traducidos en formato de objeto por el motor de persistencia.

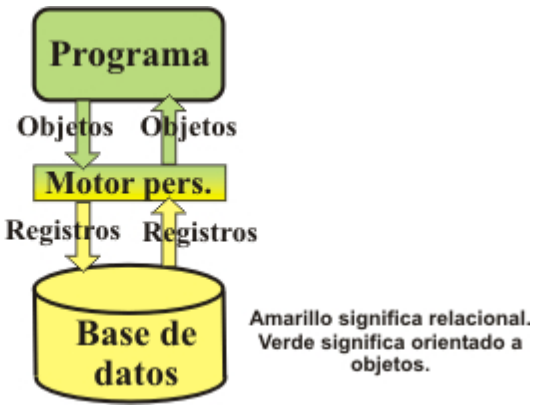


Ilustración 2: Motores de persistencia.

El programa sólo ve que puede guardar objetos y recuperar objetos, como si estuviera programado para una BD orientada a objetos. La BD sólo ve que guarda registros y recupera registros, como si el programa estuviera dirigiéndose a ella de forma relacional. Es decir, cada uno de los dos componentes trabaja con el formato de datos que le resulta más natural y es el motor de persistencia el que actúa de traductor entre los dos modelos, permitiendo que los dos componentes se comuniquen y trabajen conjuntamente.

1.1.13 XML.

XML, siglas en inglés de Extensible Markup Language ('lenguaje de marcas extensible'), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto XML no es

realmente un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML.

XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil. (O'Reilly, 2010)

Ventajas

- Es extensible: Después de diseñado y puesto en producción, es posible extender XML con la adición de nuevas etiquetas, de modo que se pueda continuar utilizando sin complicación alguna.
- El analizador es un componente estándar, no es necesario crear un analizador específico para cada versión de lenguaje XML. Esto posibilita el empleo de cualquiera de los analizadores disponibles. De esta manera se evitan bugs y se acelera el desarrollo de aplicaciones.
- Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarla. Mejora la compatibilidad entre aplicaciones. Podemos comunicar aplicaciones de distintas plataformas, sin que importe el origen de los datos, es decir, podríamos tener una aplicación en Linux con una base de datos Postgres y comunicarla con otra aplicación en Windows y Base de Datos MS-SQL Server.
- Si se quiere almacenar pequeños volúmenes de información resulta más eficiente hacerlo en ficheros XML que en base de datos pues es más rápido el acceso.
- Transformamos datos en información, pues se le añade un significado concreto y los asociamos a un contexto, con lo cual tenemos flexibilidad para estructurar documentos. (Bourred, 2005)

1.2 Conceptos asociados al STI VIRTEVALL.

A continuación se exponen las características de la aplicación a la cual se le optimizará la capa de acceso a datos. Detalles y conceptos que ayudan a comprender las necesidades de la misma.

1.2.1 Sistema Tutor Inteligente.

(Giraffa, 1997) Los delimita como: “un sistema que incorpora técnicas de IA (Inteligencia Artificial) a fin de crear un ambiente que considere los diversos estilos cognitivos de los alumnos que utilizan el programa”.

1.2.2 VIRTEVALL.

No es más que el aporte práctico del proyecto de innovación pedagógica Metodología de evaluación para aprendizaje autónomo de Idiomas Extranjeros, el cual es un STI encargado de la integración de las tecnologías en la gestión de la evaluación para aprendizaje autónomo de Idiomas Extranjeros con la utilización de algoritmos de inteligencia artificial.

1.2.3 Contextos.

Recursos didácticos utilizados por la aplicación los cuales son utilizados en la propuesta de actividades al estudiante o usuario del sistema, ya sean imágenes, videos, documentos, o archivos de sonido.

1.2.4 Deficiencias de la aplicación.

La aplicación cuenta con una base de datos, en la cual se almacena toda la información referente a los usuarios y al proceso de enseñanza-aprendizaje.

Estos datos son accedidos desde la aplicación desde la capa de acceso a datos mediantes consultas SQL previamente elaboradas en el modelo, constituyendo esto una dificultad para la optimización de la velocidad de tratamiento de información, dado a que las respuestas a las peticiones de los usuarios no dependen solo del control del gestor de bases de datos. Lo que trae consigo que el procesamiento de requisitos del manejador de base de datos sea más lento, dificulta la modularización de la aplicación, además de que los datos pueden ser alterados por el uso de sentencias del lenguaje defectuosas o erróneas.

En la base de datos se almacenan además información referente a los contextos. Este aspecto es considerado como una dificultad del sistema debido a que la información referente a dichos recursos no está sujeta a cambios, o sea, es estática, y se hace necesario acceder a la misma haciendo constantes peticiones al motor de bases de datos, produciendo la ineficiencia en el tiempo de respuesta de la base de datos así como la sobrecarga de la misma.

1.3 Tecnologías utilizadas.

Anteriormente, en la tesis de Arquitectura del proyecto VIRTEVALL se realizó un estudio profundo de los diferentes lenguajes de programación, entornos de desarrollo, gestores de bases de datos, herramientas CASE y Frameworks de desarrollo para darle cumplimiento al proyecto de innovación pedagógica. A continuación se muestra un resumen de la selección de las tecnologías con sus principales características.

1.3.1 Lenguaje de programación.

Un lenguaje de programación, no es más que un lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar. Permiten crear programas mediante un conjunto de instrucciones y operadores. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Por lo tanto, un lenguaje de programación es un modo práctico para que los programadores puedan dar instrucciones a un equipo. (Frabicio). La arquitectura de VIRTEVALL definió como lenguaje a emplear para desarrollar el proyecto, PHP 5.2.5.

PHP 5.2.5

Fue creado en el año 1994 por Rasmus Lerdorf. Se conoce originalmente como Personal Home Pages y su primera versión salió en los comienzos de 1995. Es un lenguaje de "código abierto" interpretado, de alto nivel, embebido en páginas HTML y ejecutado en el servidor. Dentro de las operaciones que pueden realizarse con este lenguaje se encuentran: procesar la información de formularios, generar páginas con contenidos dinámicos o enviar y recibir cookies, entre otras. (.PHP.net, 2009)

Existen tres campos en los que se usan scripts escritos en PHP.

- Scripts del lado del servidor. Este es el campo más tradicional y el principal foco de trabajo. Se necesitan tres cosas para que esto funcione. El intérprete PHP (CGI o módulo), un servidor web y un navegador.
- Scripts en la línea de comandos. Puede crear un script PHP y correrlo sin ningún servidor web o navegador.
- Escribir aplicaciones de interfaz gráfica. Probablemente PHP no sea el lenguaje más apropiado para escribir aplicaciones gráficas, pero si conoce bien PHP, y quisiera utilizar algunas

características avanzadas en programas clientes, puede utilizar PHP-GTK para escribir dichos programas.

Ventajas

- Puede ser utilizado en cualquiera de los principales sistemas operativos del mercado, incluyendo Linux, muchas variantes Unix (incluyendo HP-UX, Solaris y OpenBSD), Microsoft Windows, Mac OS X, RISC OS, entre otros.
- Soporta la mayoría de servidores web de hoy en día, incluyendo Apache, Microsoft Internet Information Server, Personal Web Server, Netscape e iPlanet, Oreilly Website Pro server, Caudium, Xitami, OmniHTTPd y muchos otros.
- Posee una amplia documentación en su página oficial entre la cual se destaca que todas las funciones del sistema están explicadas y ejemplificadas en un único archivo de ayuda.
- Aunque no todas las características estándar de la programación orientada a objetos están implementadas en la versión actual de PHP, muchas bibliotecas y aplicaciones grandes (incluyendo la biblioteca PEAR) están escritas íntegramente usando programación orientada a objetos.
- No se encuentra limitado a resultados en HTML. Entre las habilidades de PHP se incluyen: creación de imágenes, archivos PDF y películas Flash. También puede presentar otros resultados, como XHTM y archivos XML. PHP puede autogenerar estos archivos y almacenarlos en el sistema de archivos en vez de presentarlos en la pantalla.
- No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.
- La característica más potente y destacable de PHP es su soporte para una gran cantidad de bases de datos, entre ellas: mSQL, Direct MS-SQL, MySQL, ODBC, Oracle (OCI7 y OCI8), Ovrimos, PostgreSQL, y muchas más.

Desventajas

- No es escalable, debido a que no tiene medios propios de hacer programación distribuida.

1.3.2 Framework de desarrollo.

Se define un framework, como un conjunto de herramientas destinadas a la construcción de un determinado tipo de aplicaciones de manera generalista; es, en el ámbito informático, la estructura básica de una BD, proceso o programa. Su utilización simplifica el desarrollo de una aplicación mediante la automatización de algunos de los patrones utilizados para resolver las tareas comunes. Además, proporciona estructura al código fuente, forzando al desarrollador a crear código más legible y más fácil de mantener. Por último, el empleo de un framework facilita la programación de aplicaciones, ya que encapsulan operaciones complejas en instrucciones sencillas.

La gran facilidad, ahorro de tiempo y código, que actualmente brindan los frameworks, es sin dudas un aspecto a considerar para llevar a cabo el desarrollo de una aplicación web. La utilización de estos está dada por el lenguaje de programación seleccionado. Al ser PHP el lenguaje a utilizar para la implementación del sistema, se realizó un análisis exhaustivo de los diferentes framework de desarrollo existentes para dicho lenguaje de programación, donde el seleccionado fue Symfony 1.2.8.

Symfony 1.2.8

Symfony es un completo framework diseñado para optimizar el desarrollo de las aplicaciones web mediante algunas de sus principales características. 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. (Potencier, 2005)

Symfony está desarrollado completamente con PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y Microsoft SQL Server. Se puede ejecutar tanto en plataformas Linux, como en plataformas Windows.

Ventajas

- Funciona correctamente en cualquier tipo de plataforma.

- Independiente del SGBD. Su capa de abstracción y el uso de Propel, permiten cambiar con facilidad de SGBD en cualquier fase del proyecto.
- Utiliza programación orientada a objetos, de ahí que sea imprescindible PHP 5.
- Sencillo de usar en la mayoría de casos, está más indicado para grandes aplicaciones Web que para pequeños proyectos.
- Aunque utiliza MVC (Modelo vista controlador), tiene su propia forma de trabajo en este punto, con variantes del MVC clásico como la capa de abstracción de BD, el controlador frontal y las acciones.
- 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 las bibliotecas de otros fabricantes.
- Una potente línea de comandos que facilitan generación de código.

Desventajas

- Para su utilización se requiere de la generación de grandes cantidades de código y una configuración exhaustiva de antemano.
- Precisa de una estructuración estricta de sus directorios que se establece por consola a la hora de crear módulos.

1.3.3 Frameworks del lado del cliente.

Un framework del lado del cliente ofrece funciones JavaScript para enviar peticiones al servidor. Algunos framework son muy robustos y proveen una biblioteca completa para construir aplicaciones web. El seleccionado por el equipo de arquitectura de VIRTEVALL para el desarrollo de la aplicación fue ExtJS en su versión 2.2 teniendo en cuenta que sus componentes visuales, permitirán desarrollar una aplicación agradable y en muy poco tiempo.

ExtJS 2.2

ExtJS es una biblioteca de JavaScript para el desarrollo de aplicaciones web interactivas usando tecnologías como AJAX, DHTML y DOM. (extjses, 2010)

Originalmente construida como una extensión de la biblioteca YUI, en la actualidad puede usarse como extensión para las bibliotecas jQuery y Prototype.

Sus características principales son:

- Gran desempeño.
- Componentes de interfaz de usuario personalizables.
- Variada documentación.
- Es una librería ligera y de alto rendimiento compatible con la mayoría de los navegadores.

Ventajas

- Permite crear aplicaciones complejas utilizando componentes predefinidos así como un manejador de layouts similar al que provee Java Swing.
- Existe un balance entre Cliente – Servidor. La carga de procesamiento se distribuye, permitiendo que el servidor, al tener menor carga, pueda manejar más clientes al mismo tiempo.
- Comunicación asíncrona. En este tipo de aplicación el motor de render puede comunicarse con el servidor sin necesidad de estar sujeta a un clic o una acción del usuario, dándole la libertad de cargar información sin que el cliente se dé cuenta.
- Eficiencia de la red. El tráfico de red puede disminuir al permitir que la aplicación elija que información desea transmitir al servidor y viceversa, sin embargo la aplicación que haga uso de la pre-carga de datos puede que revierta este beneficio por el incremento del tráfico.

Desventajas

- Necesita una plataforma. En este caso se depende de ExtJS para mostrar los componentes y hacer el render de la aplicación.
- El JavaScript no es tan rápido como se quisiera, sin embargo con la entrada de Google Chorme y el nuevo motor de Mozilla las cosas van mejorando.

- Problemas con los motores de búsqueda. Los motores de búsqueda indexan el contenido web estático por lo que los textos cargados de manera dinámica no serán encontrados.
- Accesibilidad. Estas aplicaciones tienen problemas con los programas de accesibilidad pues, al igual que los motores de búsqueda, no trabajan bien con texto cargado dinámicamente.

1.3.4 Entorno de Desarrollo.

Un entorno de desarrollo integrado o IDE (acrónimo en inglés de Integrated Development Environment), es un programa informático compuesto por un conjunto de herramientas de programación.

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). (Luciano., 2009)

El entorno de desarrollo definido por la arquitectura de VIRTEVALL es el NetBeans 6.8 M2, debido a sus características, ventajas y soporte que brinda este potente Entorno de Desarrollo y sobre todo por ser su licencia “open source”.

NetBeans 6.9

NetBeans es una herramienta de código abierto para desarrolladores, que contiene las herramientas para que estos puedan crear aplicaciones desktop, enterprise, web, y aplicaciones móviles, con el lenguaje Java, así como también C/C++, PHP, JavaScript, Groovy, and Ruby. (newcomers.php., 2006)

Ventajas

- Brinda completamiento automático de código PHP, así como coloreado de código sintáctico y semántico.
- Permite depurar el código usando Xdebug.
- Permite crear e importar Proyectos Symfony.
- Soporte para crear Aplicaciones añadiéndoles flags o parámetros.
- Soporte para PHPDoc.

- Consola de comandos de Symfony, muy útil para los usuarios de Windows, donde es un poco más complejo utilizar la consola.
- Soporta todos los tipos de aplicación Java y es el primero en soportar Java EE 6 Platform.
- Es un entorno de desarrollo multiplataforma que está disponible para las plataformas Windows, Linux, Mac OS X y Solaris.
- Tiene soporte para FTP y Subversion
- Producto libre y gratuito sin restricciones de uso.

1.3.5 Sistema Gestor de Bases de Datos.

Los sistemas de gestión de bases de datos o SGBD (en inglés Database Management System, abreviado DBMS) son un tipo de software muy específico, dedicado a servir de interfaz entre la BD, el usuario y las aplicaciones que la utilizan.

El SGBD definido para el manejo de los datos del Sistema Tutor Inteligente de Evaluación para el Aprendizaje Autónomo de Idiomas es el PostgreSQL- 8.3.4-1 debido a sus características y ventajas que se muestran a continuación.

PostgreSQL- 8.3.4-1

PostgreSQL es un SGBD relacional orientada a objetos y libre, publicado bajo la licencia BSD.

Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una sola empresa sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo. Dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group).

Algunas de sus principales características son, entre otras:

- ✓ Alta concurrencia:

Mediante un sistema denominado MVCC (Multiversion Concurrency Control, por sus siglas en inglés) PostgreSQL permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos. Cada usuario obtiene una visión consistente de lo último a lo que se le hizo

commit. Esta estrategia es superior al uso de bloqueos por tabla o por filas común en otras bases, eliminando la necesidad del uso de bloqueos explícitos.

- ✓ Amplia variedad de tipos nativos

PostgreSQL provee nativamente soporte para:

- Números de precisión arbitraria.
- Texto de largo ilimitado.
- Figuras geométricas (con una variedad de funciones asociadas).
- Direcciones IP (IPv4 e IPv6).
- Bloques de direcciones estilo CIDR.
- Direcciones MAC.
- Arrays.

Adicionalmente los usuarios pueden crear sus propios tipos de datos, los que pueden ser por completo indexables gracias a la infraestructura GiST de PostgreSQL. Algunos ejemplos son los tipos de datos GIS creados por el proyecto PostGIS.

1.3.6 Metodología de desarrollo de Software.

La Metodología se define como un conjunto de métodos orientados a seguir una meta propuesta. Son los procesos que organizados en conjunto proporcionan una secuencia a seguir para la obtención del producto final. (Barrio, 2005) Las metodologías de desarrollo de software son un factor importante en la obtención de productos con calidad en el tiempo requerido. En la actualidad existen diversas metodologías de desarrollo, con características comunes pero también con significativas diferencias, recomendándose un estudio para la correcta selección de la misma, garantizando el éxito de un proyecto de software. (Cleger Despaigne) Luego del estudio realizado por el equipo de arquitectura del proyecto VIRTEVALL se llegó a la conclusión de que la más indicada para guiar el desarrollo del sistema es RUP, gracias a las características que se muestran a continuación.

RUP

Rational Unified Process, o RUP, es una plataforma flexible de procesos de desarrollo de software que ayuda brindando guías consistentes y personalizadas de procesos para todo el equipo de proyecto. RUP describe cómo utilizar de forma efectiva reglas de negocio y procedimientos comerciales probados en el desarrollo de software para equipos de desarrollo de software, conocidos como "mejores prácticas". Captura varias de las mejores prácticas en el desarrollo moderno de software en una forma que es aplicable para un amplio rango de proyectos y organizaciones. Es una guía de cómo utilizar de manera efectiva UML (Unified Modeling Language). Provee a cada miembro del equipo fácil acceso a una base de conocimiento con guías, plantillas y herramientas para todas las actividades críticas de desarrollo. Crea y mantiene modelos, en lugar de enfocarse en la producción de una gran cantidad de papeles de documentación. (Brito Acuña, 2009)

Los autores de RUP destacan que el proceso de software propuesto por RUP tiene tres características esenciales (Gómez, 2007)

- Proceso dirigido por casos de uso.
- Está centrado en la arquitectura.
- Es iterativo e incremental.

Una de las mejores prácticas centrales de RUP es la noción de desarrollar iterativamente. RUP organiza los proyectos en términos de disciplinas y fases, consistiendo cada una en una o más iteraciones. Con esta aproximación iterativa, el énfasis de cada flujo de trabajo variará a través del ciclo de vida. (Fragoso Vázquez, 2005)
La metodología RUP divide en 4 fases el desarrollo del software (Wesley, 1995):

- Inicio: Determinarla visión del proyecto.
- Elaboración: Determina la arquitectura óptima.
- Construcción: Obtiene la capacidad operacional inicial.
- Transmisión: Obtiene el release[2] del proyecto.

Cada una de estas fases se desarrolla mediante el ciclo de iteraciones, las cuales tienen sus objetivos en función de la evaluación de las iteraciones precedentes. El ciclo de vida que se desarrolla por cada iteración es llevada bajo dos disciplinas: (Wesley, 1995)

[2] *Release: liberación, puesta en circulación.*

1. Disciplina de Desarrollo: Modelamiento del Negocio, Requisitos, Análisis y Diseño, Implementación, Pruebas.
2. Disciplina de Soporte: Despliegue, Administración de configuración y cambios, Administración de Proyecto, Ambiente.

1.3.7 Herramienta CASE.

En las últimas décadas se han creado varias herramientas para el desarrollo de la Ingeniería de Software con el objetivo de desarrollar programas. Las herramientas CASE (Computer Aided Software Engineering) son utilizadas para la automatización del desarrollo de software, contribuyendo así a elevar la productividad y la calidad en el desarrollo de los sistemas informáticos, desde la planificación pasando por el análisis y diseño, hasta la generación del código fuente de los programas y la documentación. (informática, 2000) Dentro de las herramientas CASE podemos encontrar: Visual Paradigm, Rational Rose y ArgoUML, siendo Visual Paradigm la seleccionada por el equipo de Arquitectura de VIRTEVALL para el trabajo durante el desarrollo del proyecto.

Visual Paradigm

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado ayuda a una construcción más rápida de aplicaciones de calidad, mejores y a un menor coste, además permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación, así como una serie de tutoriales, demostraciones interactivas y proyectos. (Inffiet, 2007) Es importante señalar que la herramienta de modelado Visual Paradigm no es gratuito, pero la compañía Visual Paradigm UML Comunity, tiene disponible distintas versiones y facilita licencias especiales para fines académicos, sin interés de lucro. (Learning) Las razones más importantes por la que una organización debe seleccionar VP-UML sobre otras herramientas de modelado son: (Technologies., 2010)

- Permite incorporar dibujos de Visio en cualquier diagrama UML: A diferencia de otras herramientas CASE-UML, VP-UML prolonga esta limitación permitiendo incorporar dibujos de Visio en cualquier diagrama de UML, pudiendo modelar un dominio específico de hardware, software, redes, componentes, etc.

- Completa integración con Microsoft Office: VP-UML soporta la copia de diagramas como objetos OLE, para poder pegar el diagrama a cualquier documento Word, Excel, Power Point. El contenido del diagrama se puede editar directamente, sin embargo no hay que preocuparse de perder el diagrama original de la fuente.

- Herramientas efectivas y costeables: Son las más asequibles con las actuales funciones de las herramientas CASE-UML. Cuestan la octava parte del precio de otras herramientas de modelado con funcionalidades similares. Sus diversas ediciones se encuentran disponibles y se puede elegir de acuerdo a su presupuesto y necesidades.

- Soporta múltiples plataformas: No importa el Sistema Operativo que esté en uso, VP-UML se puede ejecutar en equipo y está disponible en muchas plataformas como Windows, Linux, Mac OS X y Java Desktop.

- Diseño automático del diagrama: Con su diseño automático se pueden ordenar los diagramas desordenados con tan sólo unos pocos clics con el ratón. Se puede elegir el trazado ortogonal, la disposición jerárquica o el árbol del diseño de acuerdo a la naturaleza del diagrama. Cada estilo del diseño puede afinarse con un conjunto de parámetros configurables.

Visual Paradigm es una herramienta de modelado multiplataforma que brinda una interfaz de usuario amigable, permite integrarse con varios IDE de desarrollo. También posibilita la aplicación de Ingeniería inversa, permitiendo entregar al cliente un producto bien documentado. Además tiene incluido un panel para el diseño de la interfaz de usuario lo cual permite no tener que integrarlo con Visio.

Conclusiones

En este capítulo se desarrolla un estudio sobre los principales aspectos relacionados con el Sistema Tutor Inteligente de Evaluación para el Aprendizaje Autónomo de Idiomas haciendo mayor énfasis en la capa de acceso a datos y la base de datos, así como se abordan los principales conceptos relacionados al dominio del problema para lograr un mejor entendimiento del mismo. Además, se realiza un análisis de las tecnologías utilizadas durante el desarrollo del presente trabajo de diploma, tales como lenguaje de programación, IDE de desarrollo, framework de desarrollo, SGBD, metodología de desarrollo de software y herramienta CASE , dándole cumplimiento al marco teórico de la investigación.

CAPÍTULO 2: DESCRIPCIÓN DE LA SOLUCIÓN.

Introducción

En el presente capítulo, después de haber realizado un estudio, se obtendrá una propuesta de solución según el problema a resolver planteado y con ello una visión más general de la optimización que se realizará en la capa de acceso a datos del Sistema Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas. Además se muestran de forma precisa las características de la capa de acceso a datos así como lo que se debe hacer en la misma para dar solución al problema planteado. Se aplicarán métricas al diseño y se reflejará el uso de patrones.

El objetivo de todos estos resultados es definir la vía óptima y el orden en que se van a realizar las actividades facilitando la comprensión y rapidez sobre la BD, logrando un mejor desarrollo de la solución.

2.1 Propuesta de solución.

Luego del análisis llevado a cabo, se puede plantear una propuesta que consiste en desarrollar un módulo encargado de la serialización de los contextos de la aplicación, mediante el cual se podrá leer y escribir cualquier información referente a los mismos, en ficheros XML. Además se migrarán diferentes tablas existentes en la base de datos, con información constante, a ficheros XM; evitando acceder mediante consultas a la base de datos de forma innecesaria.
Como otra propuesta para optimizar el acceso a datos del STI VIRTEVALL se plantea la migración de las consultas SQL previamente elaboradas en el modelo a implementarlas en funciones en la base de datos, mejorando así el tiempo de respuesta de la base de datos y por ende el de la aplicación.

2.2 Alcance.

El Sistema Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas debido a sus características propias, es un sistema que maneja numerosos volúmenes de información, lo cual es de vital importancia para su funcionamiento, por lo que se hace necesario optimizar el acceso a datos del mismo para lograr un mejor rendimiento de la aplicación, objetivo que persigue el presente trabajo de diploma. Actualmente la información referente a los contextos de la aplicación se encuentra almacenada en la base de datos, con la implementación del módulo de Serialización de contextos se pretende almacenar dicha información en ficheros XML, dicho módulo permitirá leer y escribir esta información sin tener que acceder a la base de datos mejorando así el rendimiento de la misma,

puesto que dicha información no se encuentra sujeta a cambios y se hace más factible almacenarla en ficheros que en base de datos. También se migrará a ficheros XML la información de algunas tablas que actualmente se encuentran en la base de datos, no sujeta a cambios, garantizando una mayor rapidez en el tiempo de respuesta de la aplicación debido a que reducirán la peticiones al motor de base de datos. También un aspecto que influirá positivamente en la mejora del tiempo de respuesta de la aplicación es la implementación de las funciones en la base de datos, dichas funciones estarán encargadas de sustituir las consultas que se hacen a la base de datos desde el modelo del sistema, implicando los principales módulos de la aplicación, como son los módulos de Autenticación, Actividades, Administración, Configuración, Presentación, Diagnóstico, Estudiante y Encuesta.

2.3 Modelo de diseño.

El modelo de diseño es una abstracción del modelo de implementación y su código fuente, el cual fundamentalmente se emplea para representar y documentar su diseño. Es usado como entrada esencial en las actividades relacionadas a la implementación. El modelo de diseño puede contener: los diagramas, las clases, subsistemas, cápsulas, protocolos, relaciones, colaboraciones, atributos y los prototipos de interfaces de usuario, entre otros que se puedan considerar para el sistema en desarrollo.

El Diseño de Sistemas se define como el proceso de aplicar ciertas técnicas y principios con el propósito de definir un dispositivo, un proceso o un Sistema, con suficientes detalles como para permitir su interpretación y realización física.
El Diseño del Sistema encierra cuatro etapas:
Trasforma el modelo de dominio de la información, creado durante el análisis, en las estructuras de datos necesarios para implementar el Software.

1. El diseño de los datos.

Define la relación entre cada uno de los elementos estructurales del programa.

2. El Diseño Arquitectónico.

Describe como se comunica el Software consigo mismo, con los sistemas que operan junto con él y con los operadores y usuarios que lo emplean.

3. El Diseño de la Interfaz.

4. El Diseño de procedimientos.

Transforma elementos estructurales de la arquitectura del programa. La importancia del Diseño del Software se puede definir en una sola palabra, *calidad*; dentro del diseño es donde se fomenta la calidad del Proyecto. El Diseño es la única manera de materializar con precisión los requerimientos del cliente. (JACOBSON, 2004).

2.3.1 Modelado mediante estereotipos web.

Para la realización de los Diagramas de Clases del Diseño será utilizada la extensión de UML para el modelado de aplicaciones Web (Laram, 1999).

Esta extensión presenta como elementos más significativos a tres clases UML: Server Page (Página Servidora), Client Page (Página Cliente) y Form (Formulario) permitiendo representar ficheros contenedores de sentencias script, empleadas para el código servidor, código cliente y formularios respectivamente. A continuación se brinda una explicación de cómo son usados estos estereotipos en el diseño de la propuesta del componente y qué representa cada cual:

<<Server Page>>: Representa la clase que tiene código que se ejecuta en el servidor, la cual se encarga de construir (build) o generar el resultado HTML y/o realizar peticiones a la capa inferior.

<<Client Page>>: Es una página Web con formato XHTML. Mezcla de datos, presentación y lógica. Son interpretadas por el navegador. Sus atributos son las variables declaradas dentro del script que son accesibles para cualquier función dentro de la página. Cada página cliente es construida por una sola página de servidor.

<<FormHTML>>: Es una colección de elementos de entrada que están contenidos en la página cliente. Sus atributos son los elementos de entrada del formulario (input boxes, radio buttons, check boxes, hidden fields, entre otros). No tienen operaciones, el método para el paso de los parámetros es $_POST y se comunican con las páginas servidores mediante submit.

2.3.2 Relaciones entre los elementos del diseño.

Entre las páginas servidoras pueden existir relaciones de inclusión (<<include>>).

Las páginas servidoras construyen el resultado XHTML que conforma el código cliente (<<build>>).

Los formularios forman parte del resultado XHTML (<<aggregation/ aggregation by).

Los formularios envían los datos al código servidor para su procesamiento (<<submit>>).

Entre las páginas clientes pueden existir vínculos (<<link>>) o redirecciones (<<redirect>>).

Las páginas clientes pueden incluir ficheros script (<<include>>).

DESDE/HASTA	Client Page	Form	Server Page
Client Page	<<link>> <<redirect>>	aggregation	<<link>>
Form	aggregation by		<<submit>>
Server Page	<<build>>		<<include>>

Ilustración 3 Relaciones entre los elementos del diseño.

2.4 Diagramas de clases del diseño.

Los diagramas de clases del diseño son diagramas estáticos que describen la estructura de un sistema mostrando sus clases, atributos y las relaciones entre ellos. Los diagramas de clases son utilizados durante el proceso de diseño de un sistema para crear el diseño conceptual de la información que se manejara en el mismo y los componentes que se encargaran de su funcionamiento, así como la relación entre ellos, como se muestra en la **Ilustración 3**. (Laram, 1999).

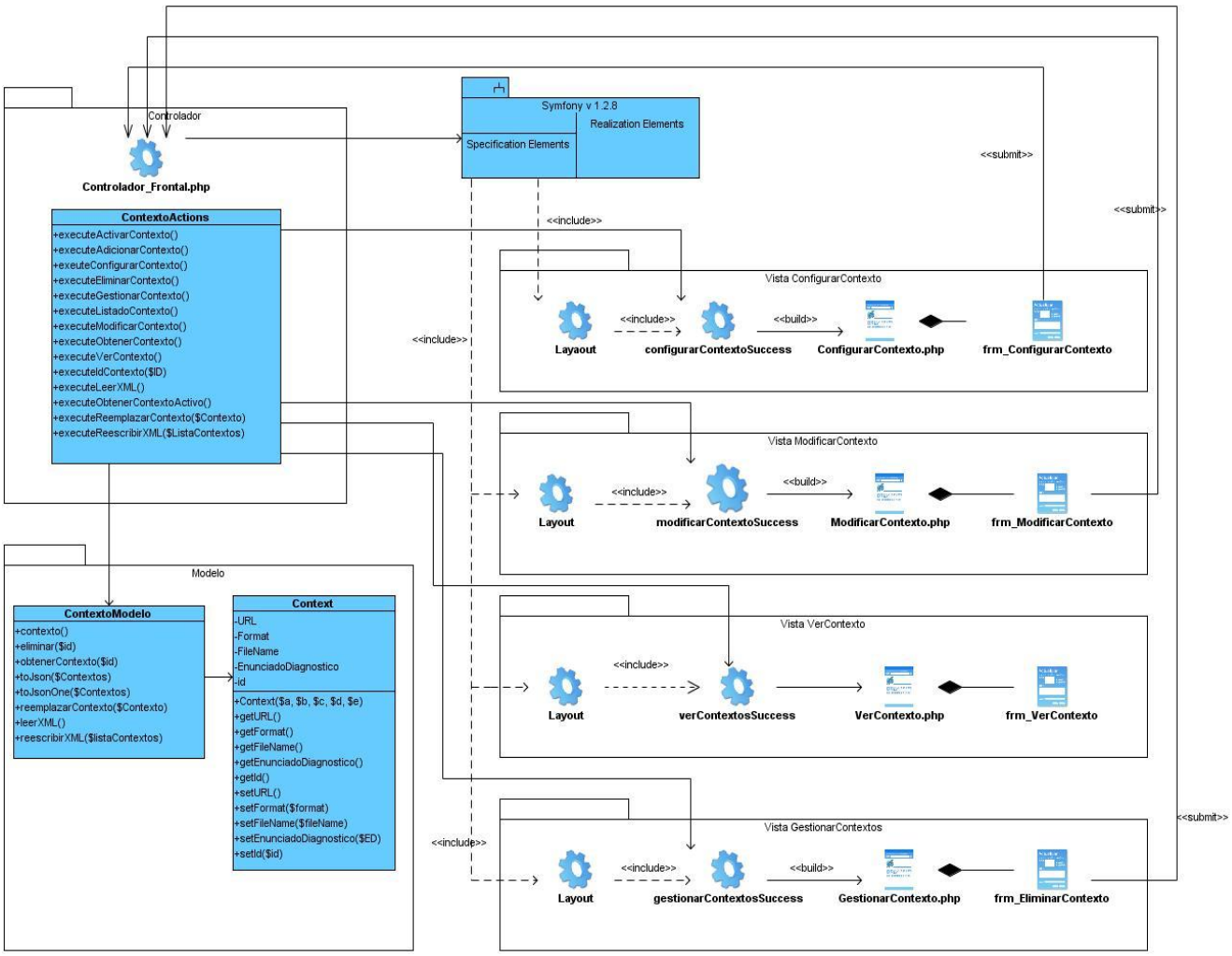


Ilustración 4: Diagrama de clases del diseño con estereotipos Web (Gestionar Contexto).

El diseño debe ser suficiente para que el sistema pueda ser implementado sin ambigüedades, y su principal objetivo es la elaboración de los diagramas de clases de diseño, que muestra las clases participantes en la realización en un caso de uso con todos sus atributos. Los restantes diagramas de clases del diseño con estereotipos Web, se encuentran anexos en la investigación en cuestión.

2.5 Modelo de Datos.

Cuando se empieza a desarrollar cualquier proyecto se debe tener en cuenta los modelos de datos, estos constituyen uno de los elementos más importantes en el instante en que se inicie el mismo, ya que puede determinar si el proyecto va a cumplir con su verdadero objetivo. Se puede decir que es la estructura sobre la que realmente reside la verdadera esencia de la aplicación. Se puede aludir entonces al **modelo de datos** como una definición lógica, independiente y abstracta de los objetos, operadores y demás que en conjunto constituyen la máquina abstracta con la que interactúan los

usuarios. Los objetos permiten modelar la estructura de los datos. Los operadores nos permiten modelar su comportamiento. **(Date 2001)**

2.5.1 Diseño de la base de datos.

El diseño de una base de datos es el proceso por el que se determina la organización de la misma, incluido su estructura, contenido y las aplicaciones que se han de desarrollar. El rendimiento adecuado del sistema se deberá a una correcta selección de las tablas que formarán parte de la base de datos, así como de la información que contendrá cada una de ellas, las relaciones que se establezcan entre las mismas y del correcto uso de formas de normalización. Las clases del diseño y las realizaciones de los casos de usos son artefactos de entradas para realizar el artefacto principal que se genera en esta actividad: **el modelo de datos,** el cual describe la representación lógica y física de los datos persistentes.

2.5.2 Modelo Entidad Relación de la BD.

El resultado de la fase del diseño conceptual, que no es más que el diagrama entidad relación, constituyó la entrada para la realización del modelo entidad relación de la base de datos propuesta, del cual, un fragmento se muestra a continuación:

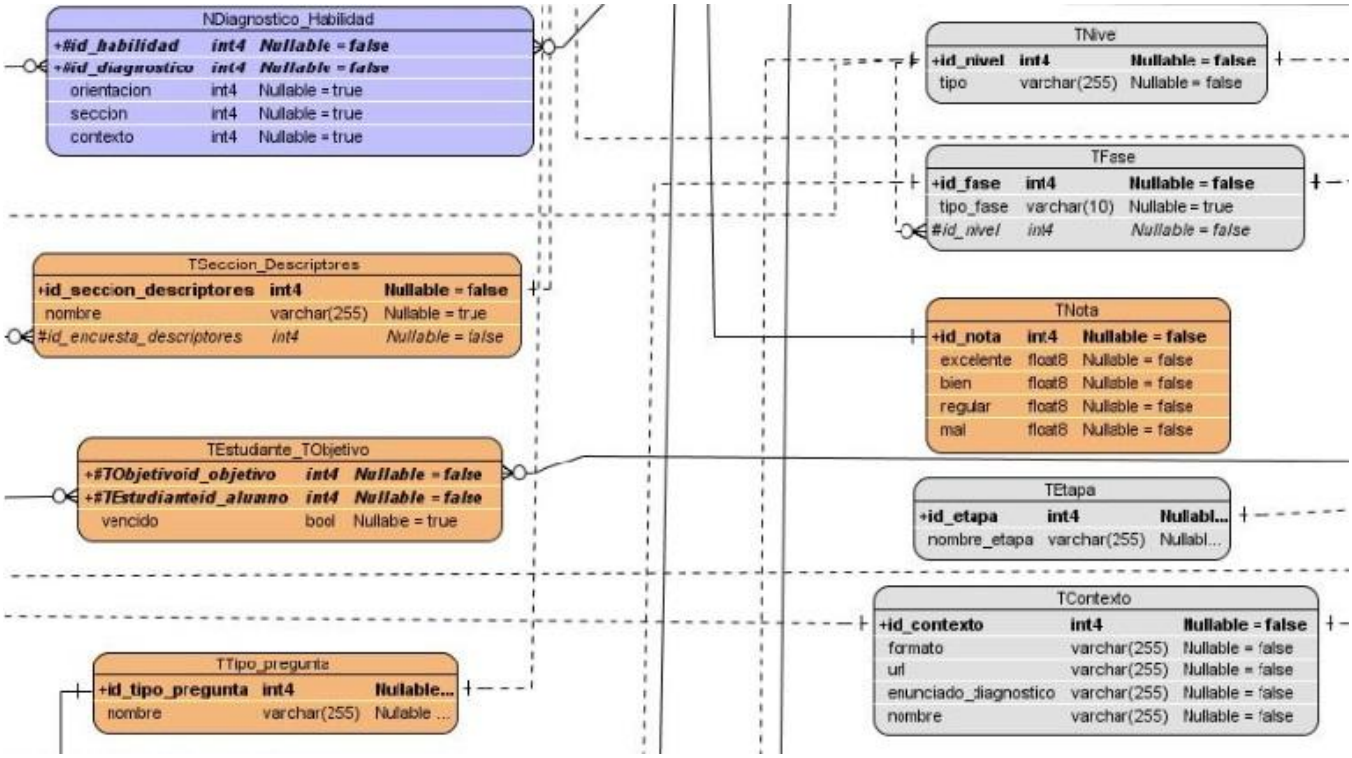


Ilustración 5: Fragmento del modelo de datos del STI VIRTEVALL.

El modelo entidad-relación generado para el sistema contiene las tablas (y las relaciones entre ellas) que guardarán la información del STI.

El modelo que muestra en el **anexo 1**, cuenta con 72 tablas, de las cuales 18 son nomencladoras, fundamentales en el proceso de completamiento de la información. Las tablas de color amarillo, contienen valores persistentes de toda la aplicación y son las más importantes arquitectónicamente dentro del modelo. Las entidades de color gris se migraron a ficheros XML, con el objetivo de optimizar el acceso a datos fundamentado en la investigación.

UML soporta tanto herencia simple como herencia múltiple. Todas en su conjunto conforman un modelo de entidad-relación, normalizado en Tercera Forma Normal (3FN), según la propuesta original de (Cood, 1992) que plantea que una relación está en tercera forma normal si todos los atributos de la relación dependen funcionalmente sólo de la clave, y no de ningún otro atributo.

La normalización de los datos puede considerarse como un proceso durante el cual los esquemas de relación que no cumplen las condiciones se descomponen repartiendo sus atributos entre esquemas de relación más pequeños que cumplen las condiciones establecidas. Un objetivo del proceso de normalización realizado para el modelo antes presentado, es garantizar que no ocurran anomalías de actualización. (Cood, 1992)

2.5.2.1 Descripción de algunas de las principales tablas de la base de datos.

Nombre: TUsuario		
Tipo: Tabla		
Atributos	**Tipo**	**Descripción**
Id_aumno	Serial	Almacena el id del usuario.
nombre	varchar	Almacena el nombre del usuario.
apellido1	varchar	Almacena el primer apellido del usuario.
apellido2	varchar	Almacena el segundo apellido del usuario.
correo	varchar	Almacena el correo del usuario.
contraseña	varchar	Almacena la contraseña del usuario.
ocupacion	varchar	Almacena la ocupación id del usuario.
id_tipo_usuario	varchar	Almacena el id del tipo de usuario, el cual es foráneo de la tabla TTipo_Usuario

Nombre: TDiagnostico		
Tipo: Tabla		
Atributos	**Tipo**	**Descripción**
id_diagnostico	Serial	Almacena el id del diagnóstico.
perfil_inteligencia	varchar	Almacena el perfil de inteligencia.
estrategia_aprendizaje	varchar	Almacena la estrategia de aprendizaje.
nivel	varchar	Almacena el nivel del diagnóstico.
fase	integer	Almacena la fase del diagnóstico.
id_tipo_usuario	varchar	Almacena el id del tipo de usuario, el cual es foráneo de la tabla TTipo_Usuario
id_tipo_diagnostico	integer	Almacena el id del tipo de diagnóstico, el cual es foráneo de la tabla TTipo_Diagnostico

Nombre: NEstudiante_Examen		
Tipo: Nomenclador		
Atributos	**Tipo**	**Descripción**

id_alumno	integer	Almacena el id del usuario, el cual es foráneo de la tabla TUsuario.
id_examen	integer	Almacena el id del examen, el cual es foráneo de la tabla TExamen.
nota	integer	Almacena la nota que obtuvo el usuario en el correspondiente examen.
resultado	varchar	Almacena el resultado que obtuvo el usuario en el correspondiente examen.

Nombre: NEstudiante_Encueta		
Tipo: Nomenclador		
Atributos	**Tipo**	**Descripción**
id_alumno	integer	Almacena el id del usuario, el cual es foráneo de la tabla TUsuario.
id_encuesta	integer	Almacena el id de la encuesta, el cual es foráneo de la tabla TEcuesta.
resultado	varchar	Almacena el resultado que obtuvo el usuario en la correspondiente encuesta.

2.5.2.2 Descripción de algunas de las principales funciones.

Nombre	**Descripción**
fn_objetivo_fase_est(id_est)	Dado el id de un estudiante, devuelve un listado con los objetivos de la fase en que se encuentra dicho estudiante.
fn_objetivosnocumplidos(id)	Dado el id de un estudiante, devuelve un listado con los objetivos que no ha cumplido de la fase en que se encuentra dicho estudiante.
fn_ocupacion_usuario(id)	Devuelve la ocupación de un usuario dado el id del usuario.
fn_registro_actividades(id)	Devuelve el registro de todas las actividades registradas de un usuario dado su id.
fn_usuario_alumno(id)	Devuelve todos los datos de un alumno dado su id.

fn_usuario_estudiante(id)	Devuelve todos los datos de un estudiante dado su id.
fn_usuario_profesor	Devuelve todos los datos de un profesor dado su id.
fn_usuario_trabajador	Devuelve todos los datos de un trabajador dado su id.
fn_datos_tipo_usuario	Dado el id de un usuario devuelve todos los datos referente a él según el tipo de usuario que sea.
f_insert_encuesta	Inserta una encuesta nueva en la base de datos.
f_insert_estudiante	Inserta una actividad nueva en la base de datos.
f_delete_encuesta	Elimina una encuesta de la tabla TEncuesta.
f_delete_estudiante	Elimina un estudiante de la tabla TEstudiante.
f_delete _inciso_encuesta	Elimina un inciso de encuesta de la tabla Tlinciso_Encuesta.
f_delete_profesor	Elimina un profesor de la tabla TProfesor.
f_delete_tipo_encuesta	Elimina un tipo de encuesta de la tabla TTipo_Encuesta.
f_delete_tipo_pregunta	Elimina un tipo de pregunta de la tabla TTipo_Pregunta.
f_delete_trabajador	Elimina un trabajador de la tabla TTrabajador.
f_delete_usuario	Elimina un usuario de la tabla TUsuario.

2.6 Patrones de Arquitectura.

En el capítulo anterior se menciona que el framework para PHP a utilizar en el desarrollo de la aplicación es Symfony en su versión 1.2.8 debido a que el mismo posee características y funcionalidades necesarias para lograr un desarrollo de forma eficiente. Una de las características fundamentales del uso del patrón de arquitectura Modelo-Vista-Controlador (MVC) es que separa la lógica del negocio de la presentación, haciendo más sencillo el mantenimiento de las aplicaciones.

El **modelo** representa la información con la que trabaja la aplicación, es decir, su lógica de negocio. Típicamente las clases de modelo contendrán funciones encargadas de recuperar, insertar y actualizar información en la BD.

La **vista** transforma el modelo en una página web que permite al usuario interactuar con ella.

El **controlador** se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

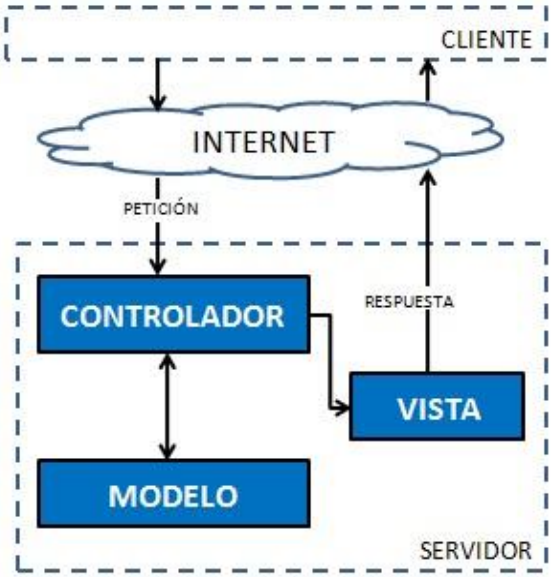


Ilustración 6: Patrón Modelo-Vista-Controlador (MVC).

2.6.1 La implementación del MVC que realiza Symfony.

El controlador frontal y el layout son comunes para todas las acciones de la aplicación. Se pueden tener varios controladores y varios layouts, pero solamente es obligatorio tener uno. El controlador frontal es un componente que sólo tiene código relativo al MVC, por lo que no es necesario crear uno, ya que Symfony lo genera de forma automática. (Potencier, 2005)

Ilustración 7: Flujo de trabajo de Symfony.

2.7 Patrones de Diseño utilizados en la implementación.

Durante el diseño del Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas, la dirección el proyecto junto al equipo de diseño decidió hacer uso de Patrones de Software.

2.7.1 Patrones GRASP.

Para la Asignación general de Responsabilidad (o patrones GRASP por sus siglas en ingles), los mismos constituyen principios básicos a tener en cuenta cuando se quiere construir eficazmente un software orientado a objetos. Entre los más evidentes en el diseño propuesto se encuentran los patrones: Creador, Experto, Alta Cohesión, Bajo Acoplamiento y Controlador.

- **Experto**: En la solución propuesta este es uno de los patrones más importantes ya que existen en el modelo clases referentes a los contextos que encapsulan toda la lógica y funcionalidades de los mismos.

- **Creador:** Las acciones definidas para cada módulo se encuentran en la clase nombremoduloActions. Las acciones contienen toda la lógica de la aplicación. En las acciones se crean los objetos de las clases que representan las entidades y de los objetos de otras clases que intervienen en la lógica del módulo.

- **Alta Cohesión**: En cada clase Actions se definen las acciones para las plantillas, además estas colaboran con otras para realizar diferentes operaciones, se instancian objetos, se acceden a las properties, es decir en una Actions se utilizan diferentes funcionalidades estrechamente relacionadas entre sí, lo que proporciona un software flexible ante los cambios.

- **Controlador:** Todas las peticiones Web son manejadas por un solo controlador frontal (sfActions), que es el punto de entrada único de toda la aplicación en un entorno determinado.

- **Bajo Acoplamiento:** A cada clase Symfony se le asigna una responsabilidad, de forma tal que mantiene pocas dependencias entre las mismas.

2.7.2 Patrones GOF.

Para contribuir a una implementación eficiente se hizo necesario el estudio de los patrones del grupo de los cuatro (GOF, Gang Of Four), de ellos se seleccionaron cuatro: Singleton, Abstract Factory, Decorator, Composite, los cuales se explicarán a continuación.

- ✓ En la categoría Creacionales:

Singleton (Instancia única): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. En las acciones se usan los métodos ->getRequest (), ->getUser (), esto se debe a que, en la acción, el método getContext () guarda una referencia a todos los objetos del núcleo de Symfony, estos métodos pueden ser accedidos desde la vista y desde el controlador, solo varía la forma de llamarlos. (Elers Pérez, 2009)

- ✓ En la categoría Estructurales:

Decorator (Envoltorio): Añade funcionalidad a una clase, dinámicamente. En cada archivo layout.php se define el código HTML común de cada vista, evitando que este sea repetido en cada página.

2.8 Patrones de acceso a datos.

2.8.1 Patrón de diseño DAO.

En la tarea de optimizar el acceso a datos del Sistema Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas existe un patrón de diseño que su utilización se ha hecho indispensable, este es el patrón de diseño DAO (Data Access Object, por sus siglas en inglés) es uno de los patrones de diseño de mayor uso en el desarrollo del acceso a datos de un sistema informático.
Se utiliza para separar las operaciones de bajo nivel de acceso a datos, de las operaciones de alto nivel de la lógica del negocio. Una implementación típica de DAO contiene:

- ➢ Una interfaz DAO. Que expone operaciones de acceso a datos, estas operaciones no están vinculadas con alguna tecnología de persistencia. Esta característica permite la abstracción del mecanismo que se emplee para acceder a los datos, permitiendo desarrollar diversas implementaciones a una misma interfaz DAO.

- ➢ Una clase concreta que implemente la interfaz DAO. Esta clase contiene la lógica de acceso a datos a una fuente específica de información.
- ➢ Una clase fabricadora. Responsable de instanciar las interfaces DAO.
- ➢ Objetos de transferencias de datos. Posibilitan transferir la información que se obtiene desde una fuente determinada.
- ➢ Este patrón permite implementar las operaciones definidas por una interfaz DAO de diversas maneras utilizando diferentes tecnologías, sin producir cambios en el código que utiliza las

operaciones expuestas por esta interfaz, ya que cada implementación concreta de una interfaz DAO es transparente a quien utilice dicha interfaz. (LAGO, 2007)

2.8.2 Patrón CRUD.

Acrónimo de Create-Read-Update-Delete. Conocido como el padre de todos los patrones de capa de acceso. Es usado para referirse a las funciones básicas en BD y a la capa de persistencia en un sistema de software. Describe que cada objeto debe ser creado en la BD para que sea persistente. Una vez creado, la capa de acceso debe tener una forma de leerlo para poder actualizarlo o simplemente borrarlo. Teóricamente el borrado de objetos debería quedar a cargo de la misma BD. Pero un recolector de objetos "basura" (garbage collector) en una BD gigante afecta en gran medida el rendimiento. Por ello es que la tarea de borrado queda delegada al programador. (Pizarro, 2011)

2.9 Modelo de Implementación.

La etapa de implementación surge con el resultado del diseño e implementado el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ficheros de código binario, ejecutables y similares. El objetivo principal de esta etapa es desarrollar la arquitectura y el sistema como un todo, de la forma más específica. Teniendo en cuenta los propósitos de la implementación:

- Definir la organización del código.
- Planificar las integraciones del sistema necesario en cada iteración.
- Implementar las clases y subsistemas encontrados durante el diseño.

El modelo de Implementación es la unión de varios artefactos que se generan de gran valor, pues denota la implementación actual del sistema en términos de componentes y subsistemas de implementación, es natural mantener el mismo a lo largo de todo el ciclo de vida del software. (Luciano., 2009)

2.9.1 Diagrama de componentes.

Un componente es el empaquetamiento físico de los elementos de un modelo.

En los diagramas de componentes se muestran los elementos de diseño de un sistema de software. Un diagrama de componentes permite visualizar con más facilidad la estructura general del

sistema y el comportamiento del servicio que estos componentes proporcionan y utilizan a través de las interfaces. (Microsoft, 2011)

Puede usar un diagrama de componentes para describir un diseño que se implemente en cualquier lenguaje o estilo. Solo es necesario identificar los elementos del diseño que interactúan con otros elementos del diseño a través de un conjunto restringido de entradas y salidas. Los componentes pueden tener cualquier escala y pueden estar interconectados de cualquier manera. (Microsoft, 2011)
A continuación se muestra el diagrama de componentes propuesto. El mismo está acorde con los requerimientos de las nuevas mejoras a implementar y con el MVC propuesto por el framework Symfony.

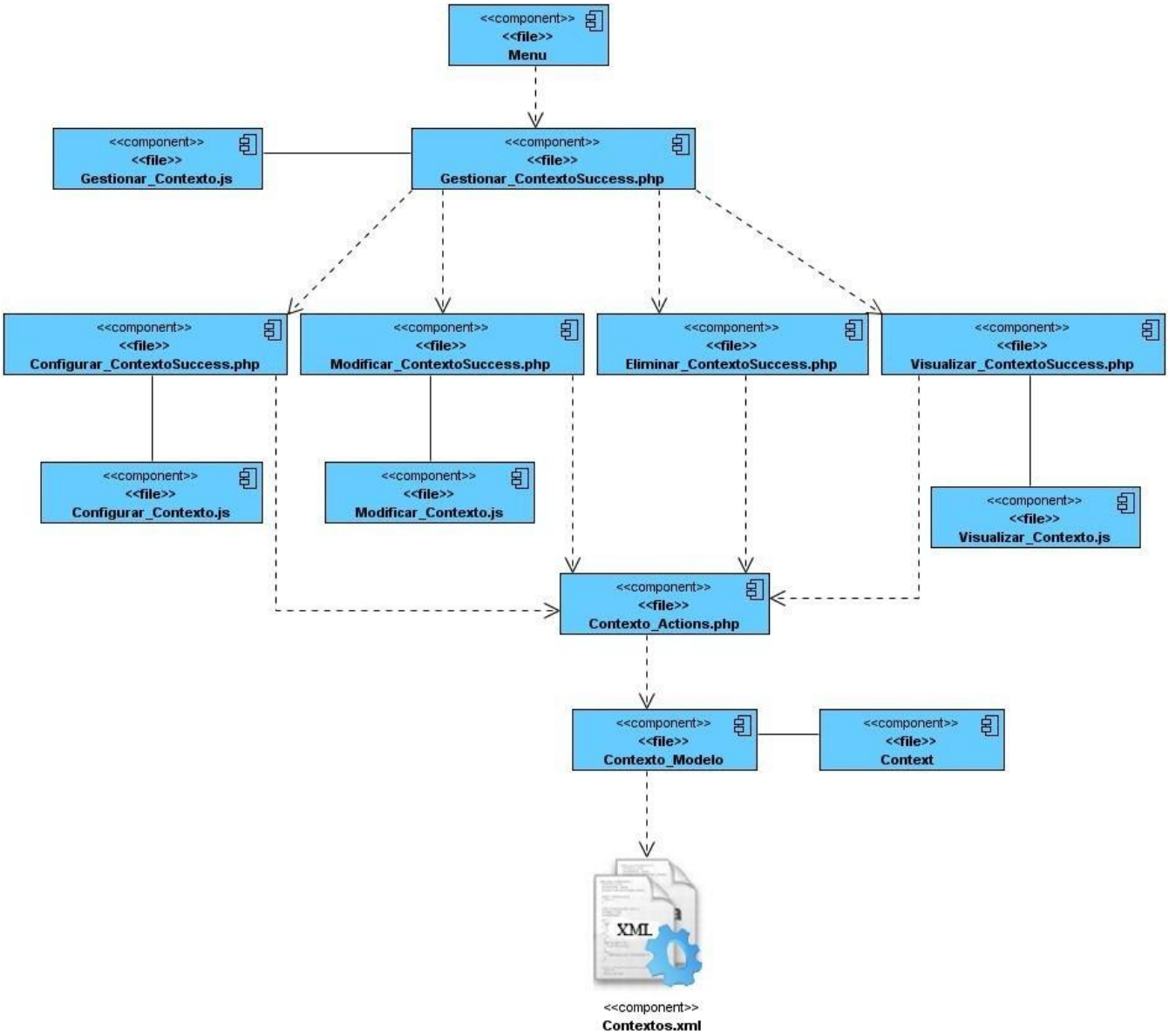


Ilustración 8: Diagrama de componentes Gestionar Contextos.

2.9.2 Estándares de codificación.

Los estándares de codificación consisten en estilos de codificación a la hora de escribir el código. Es necesario establecer estándares de codificación para todos los programadores para lograr un mayor entendimiento entre ellos. 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 cuáles 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.

2.9.2.1 Identificadores.

En el caso de los identificadores existen estilos definidos mundialmente como el lowerCamelCase y el UpperCamelCase. 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 UpperCamelCase.
- Clase: Para las clases se escogió el UpperCamelCase.

```
class ContextoActions extends sfActions
```

- Función: Para las funciones se escogió el lowerCamelCase.

```
public function executeActivarContexto()
```

- Variable

```
private $EnunciadoDiagnostico
```

2.9.2.2 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 dos 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.

```
1   <?php
2
3   /**...
11
12  class ContextoActions extends sfActions {
13
14      public function executeAdicionarContexto() {...}
56
57      public function executeGestionarContexto() {...}
61
62      public function executeListadoContextos() {...}
65
66      public function executeConfigurarContexto() {...}
69
70      public function executeObtenerContexto() {...}
74
75      public function executeEliminarContexto() {...}
```

Ilustración 9: Ejemplo de indentación.

2.9.2.3 Llaves.

Existen varios criterios en cuanto a la ubicación de las llaves que delimitan el cuerpo de los bloques de código. Algunos programadores ubican la llave de apertura inmediatamente detrás de la línea cabecera del bloque mientras otros lo hacen de forma solitaria en la línea siguiente a la línea cabecera. Para este último estilo existen además diferencias en cuanto al nivel de indentación 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 se definió anteriormente: Las llaves de apertura de las funciones se colocarán solitarias en la línea siguiente e indentadas al nivel de la línea cabecera del bloque y las llaves de aperturas de las estructuras (for, if, while, else,) se colocarán inmediatamente detrás 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 y entendimiento del código.

```
79      public function executeleerXML()
80      {
81          $archivo = 'uploads/CONTEXTOS.xml';
82          $listaContextos = array();
83          if (file_exists($archivo)) {
84              $xml = simplexml_load_file($archivo);
85              if ($xml) {
86                  foreach ($xml->Contexto as $Cxt) {
87                      $listaContextos[] = new Context();
88                  }
89              }
90          }
91          return $listaContextos;
92      }
```

Ilustración 10: Ejemplo de uso de llaves.

2.9.2.4 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.

```
95      public function executeGestionarContexto()
96      {
97          return $this->renderText(TcontextoPeer::toJson($this->leerXML()));
98      }
99
100     public function executeListadoContextos()
101     {
102         return sfView::SUCCESS;
103     }
```

Ilustración 11: Ejemplo 1 de espacios en blanco.

- Entre declaraciones de variables e implementaciones dentro del cuerpo de las funciones e colocarán espacios en blanco:

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

    private **$URL**;

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

```
26      public function Context($a, $b, $c, $d, $e)
27      {
```

Ilustración 12: Ejemplo 2 de espacios en blanco.

  - Después de cada punto y coma (;) en las estructuras for.

```
172         for ($i = 0; $i < count($ListaContextos); $i++){
173             $Ct = $c->addChild('Contexto');
```

Ilustración 13: Ejemplo 3 de espacios en blanco.

2.10 Descripción de las nuevas clases u operaciones necesarias.

Nombre: ContextoActions	
Tipo de clase: Controladora	
Atributos:	**Tipo:**
-	-
Funciones:	
Nombre:	activarContexto

Descripción:	Selecciona un contexto.
Nombre:	adicionarContexto
Descripción:	Adiciona un contexto.
Nombre:	configurarContexto
Descripción:	Permite configurar un contexto.
Nombre:	eliminarContexto
Descripción:	Permite eliminar un contexto.
Nombre:	gestionarContexto
Descripción:	Permite buscar todos los contextos para luego ser mostrados.
Nombre:	listadoContexto
Descripción:	Permite mostrar un listado de contextos.
Nombre:	modificarContexto
Descripción:	Permite modificar un contexto.
Nombre:	obtenerContexto
Descripción:	Permite obtener un contexto.
Nombre:	verContexto
Descripción:	Visualiza un contexto.
Nombre:	leerXML
Descripción:	Permite leer el XML donde se almacena la información de los contextos.
Nombre:	obtenerContextoActivo
Descripción:	Permite obtener el contexto que se encuentre seleccionado.

Nombre:	reemplazarContexto($Contexto)
Descripción:	Reemplaza un contexto que se encuentra almacenado por uno nuevo que se le pasa por parámetro.
Nombre:	reescribierXML($ListaContextos)
Descripción:	Reescribe el fichero XML pasándole uno nuevo por parámetro.

Tabla 1: Clase Control ContextoActions.

Nombre: Contexto	
Tipo de clase: Modelo	
Atributos:	**Tipo:**
URL	String
Format	String
FileName	String
EnunciadoDiagnostico	String
Id	Integer
Funciones:	
Nombre:	Contex($URL, $Format, $FileName, $EnunciadoDiagnostico, $Id)
Descripción:	Crea un nuevo contexto pasándoles los datos por parámetro.
Nombre:	getURL
Descripción:	Permite obtener el URL.
Nombre:	getFormat
Descripción:	Permite obtener el formato.
Nombre:	getFileName
Descripción:	Permite obtener el FileName.
Nombre:	getEnunciadoDiagnostico
Descripción:	Permite obtener el EnunciadoDiagnostico.

Nombre:	getId
Descripción:	Permite obtener el ID.
Nombre:	setURL($URL)
Descripción:	Permite cambiar el URL por otro que se le entre por parámetro.
Nombre:	setFormat($Format)
Descripción:	Permite cambiar el Format por otro que se le entre por parámetro.
Nombre:	setFileName($FileName)
Descripción:	Permite cambiar el FileName por otro que se le entre por parámetro.
Nombre:	setEnunciadoDiagnostico($enunciadoDiagnostico)
Descripción:	Permite cambiar el EnunciadoDiagnostico por otro que se le entre por parámetro.
Nombre:	setId($Id)
Descripción:	Permite cambiar el Id por otro que se le entre por parámetro.

Tabla 2: Clase Modelo Contex.

Nombre: ContextoPeer	
Tipo de clase: Modelo	
Atributos:	**Tipo:**
-	-
Funciones:	
Nombre:	eliminar($id_Contexto)
Descripción:	Permite eliminar un contexto
Nombre:	obtenerContexto($id_contexto)
Descripción:	Permite obtener un contexto pasándole el ID.

Nombre:	toJson($Contextos)
Descripción:	Permite convertir y obtener en formato Json todos los contextos.
Nombre:	toJsonOne($Contexto)
Descripción:	Permite convertir y obtener en formato Json un contexto pasado por parámetro.

Tabla 3: Clase Modelo ContextoPeer.

Anexos a esta investigación se encuentran el resto de las tablas con los datos y especificaciones de las nuevas clases.

2.11 Evaluación del Modelo de diseño propuesto.

En la mayoría de los desafíos técnicos, las métricas nos ayudan a entender tanto el proceso técnico que se utiliza para desarrollar un producto, como el propio producto. El proceso para intentar mejorarlo, el producto se mide para intentar aumentar su calidad. (PRESMAN, 1998)

Entre los tipos de Métricas que se utilizan para evaluar el diseño están:

- **Tamaño operacional de clase (TOC):** está dado por el número de métodos asignados a una clase.
- **Relaciones entre clases (RC):** está dado por el número de relaciones de uso de una clase con otras.

Para evaluar el diseño se empleó la métrica Tamaño operacional de clase (TOC), la cual mide la calidad de acuerdo a los atributos Responsabilidad, Complejidad de implementación y Reutilización de las clases.

Atributos de calidad que se abarcan en la métrica TOC:

- **Responsabilidad:** Consiste en la responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- **Complejidad de implementación:** Consiste en el grado de dificultad que tiene implementar un diseño de clases determinado.
- **Reutilización:** Consiste en el grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.

2.11.1 Evaluación de las métricas.

La calidad del software puede ser entendida como el grado con el cual el usuario percibe que el software satisface sus expectativas. El tipo y número de actividades de garantía de calidad que es necesario adoptar en un proyecto o en una organización depende del tamaño y complejidad de los productos software que se estén desarrollando. También influyen otros factores, como pueden ser el tipo de proceso de desarrollo de software o los métodos y herramientas utilizados, la estructura organizativa de la organización, la motivación del personal, entre otros. Según el modelo de calidad ISO 9126, la calidad de un proceso contribuye a mejorar la calidad del producto, y, a su vez, la calidad del producto contribuye a mejorar la calidad en uso. La finalidad de la calidad en uso es medir la efectividad, productividad, seguridad y la satisfacción de los usuarios (pertenecientes a perfiles determinados) que interactúan con el producto en escenarios específicos de uso. (Sicilia, 2009)

En este epígrafe sólo se mostrarán las gráficas y las valoraciones que resultaron después de aplicar las métricas, para las cuales se tuvieron en cuenta el número de métodos de las clases Controladoras y Modelos reflejados en el diseño de acuerdo a la propuesta planteada por la arquitectura.

2.11.2 Métricas de Software.

Según Pressman en su libro Ingeniería de software, un enfoque práctico, plantea lo siguiente: “El proceso del software y las métricas del producto son una medida cuantitativa que permite a la gente del software tener una visión profunda de la eficacia del proceso del software y de los proyectos que dirigen utilizando el proceso como un marco de trabajo. Se reúnen los datos básicos de calidad y productividad. Estos datos son entonces analizados, comparados con promedios anteriores, y evaluados para determinar las mejoras en la calidad y productividad. Las métricas son también utilizadas para señalar áreas con problemas de manera que se puedan desarrollar los remedios y mejorar el proceso del software” (Pressman, 2005).

2.11.3 Métricas de usabilidad.

Se pueden definir las mismas como aquellos criterios o variables que son medibles de forma objetiva. Mientras que la interpretación de una opinión es un análisis cualitativo o subjetivo por parte del experto, la interpretación de datos objetivos responde a un análisis cuantitativo. (Métricas, 2010).

De acuerdo con lo anterior el siguiente gráfico representa los resultados generales obtenidos de la aplicación de las métricas Tamaño operacional de clases, de acuerdo a la cantidad de métodos.

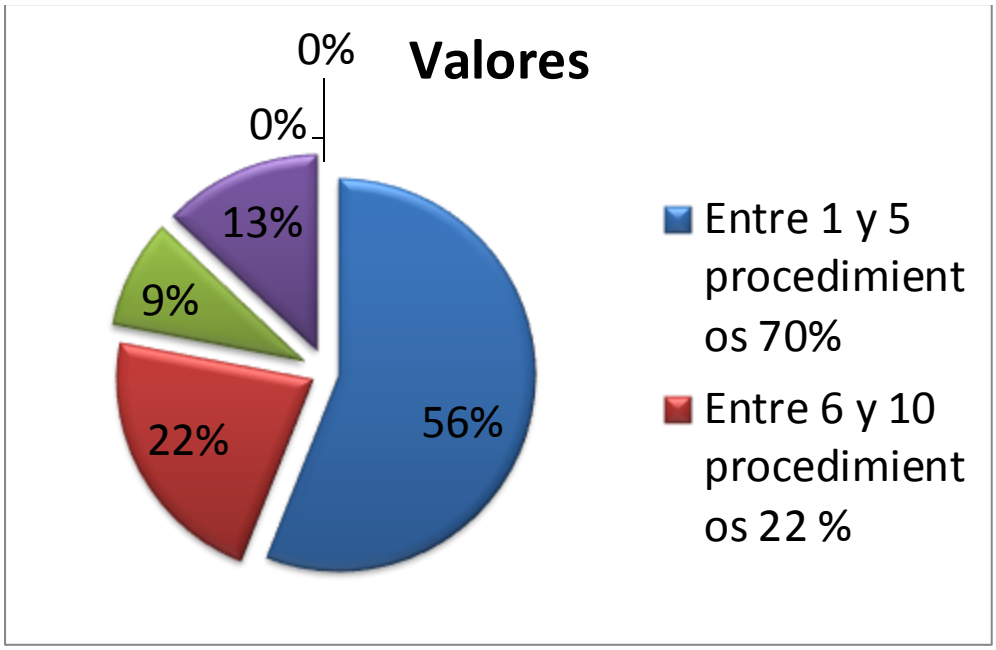


Ilustración 14: Gráfico de los resultados generales de acuerdo a la cantidad de procedimientos.

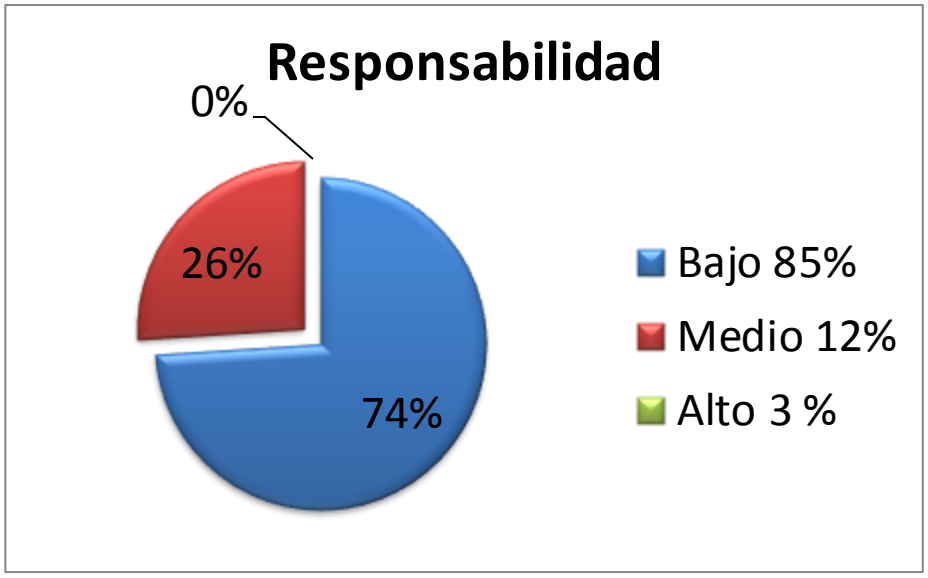


Ilustración 15: Gráfico de la Responsabilidad por clases.

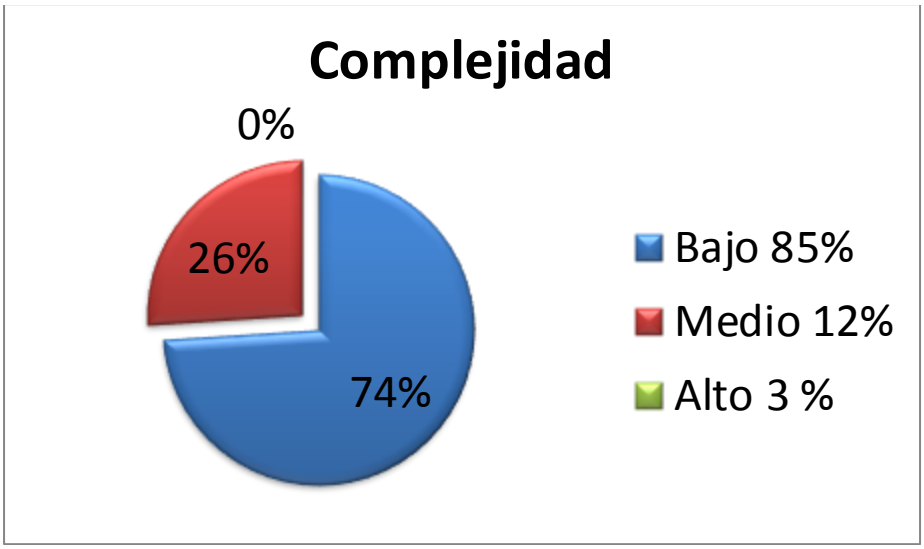


Ilustración 16: Gráfico de la Complejidad de implementación por clases.

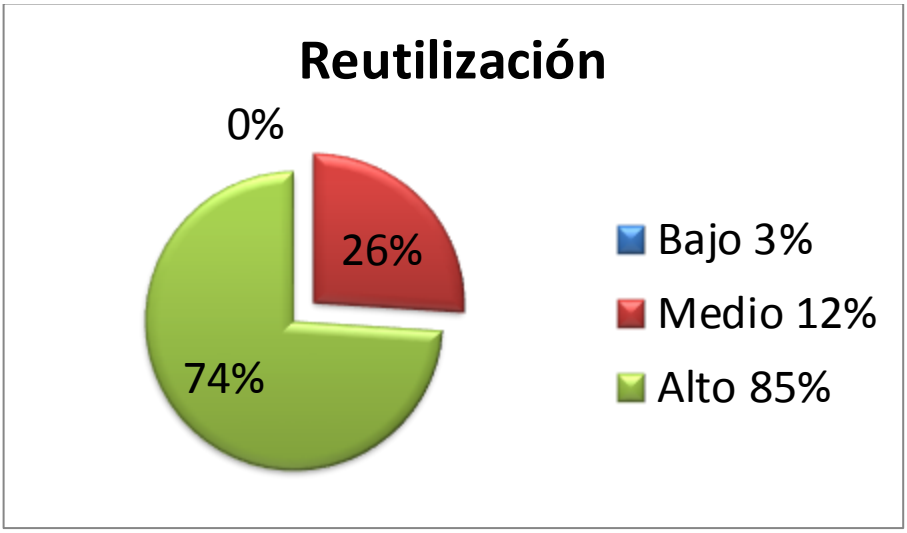


Ilustración 17: Gráfico del nivel de Reutilización de las clases.

Analizando los resultados obtenidos en la evaluación de la métrica TOC se puede concluir que el diseño tiene una calidad aceptable teniendo en cuenta que el 56% de las clases utilizadas posee menos cantidad de operaciones que la mitad del valor máximo registrado en las mediciones. Además el 74% de las clases poseen evaluaciones positivas en los atributos de calidad (Responsabilidad, Complejidad de implementación y Reutilización).

Conclusiones

Con el desarrollo de este capítulo se obtuvo la solución que tributa a la optimización del acceso a datos del Sistema Tutor Virtual de Aprendizaje de Evaluación para el Aprendizaje Autónomo de Idiomas. Se implementaron las funciones en el sistema gestor de bases de datos y se implementó el módulo encargado de la persistencia y serialización de los contextos siguiendo el patrón arquitectónico MVC y los diferentes patrones de diseño logrando resolver el problema de la forma más óptima.

CAPÍTULO 3: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

Introducción

Durante el proceso de desarrollo de un software, los errores pueden comenzar a aparecer incluso desde el mismo momento en que fueron definidos los objetivos y estos a su vez especificados de forma errónea e imperfecta; de la misma forma en los posteriores pasos del diseño y desarrollo. A raíz de la incapacidad humana de trabajar y comunicarse de forma perfecta, el desarrollo del software debe ser acompañado de una actividad que garantice su calidad (Quesada, 2005).

En el desarrollo de este capítulo se valida mediante pruebas si la solución brindada ha cumplido el objetivo planteado con la calidad requerida.

3.1 Modelo de pruebas.

Las pruebas son el proceso de ejercitar un programa con la intención de especificar y encontrar errores previos a la entrega del sistema al usuario final. Estas van encaminadas a garantizar la calidad del software en todo momento del desarrollo. Durante este flujo de trabajo se verifica el resultado de la implementación, planificando, diseñando e implementando los casos de prueba. (Reina, 2006). (Reina)

3.1.1 Objetivos de las pruebas.

Probar, es el proceso de ejecutar un programa con el fin de encontrar errores o fallas. Comprende los objetivos siguientes (Quesada, 2005):

- La prueba es el proceso de ejecución de un programa con la intención de descubrir un error.
- Un buen caso de prueba es aquel que tiene una alta probabilidad de descubrir un error no encontrado hasta entonces.
- Una prueba tiene éxito si descubre un error no detectado hasta entonces.

Las pruebas son una tarea tanto o más creativa que el desarrollo de software. Sin embargo, ha sido subvalorada por considerarse erróneamente como una tarea destructiva y rutinaria. No debe recurrirse a ella en una última etapa del desarrollo del software.

Un **proceso de pruebas** no es más que la sucesión de pasos y decisiones que se siguen para realizar una determinada actividad o tarea (Pons, 2006). Por tanto, puede definirse como proceso de pruebas de manera general, al grupo de actividades y entregables que se siguen para realizar pruebas a un

software, garantizando un producto confiable, sin errores y con la mayor calidad posible. Se recomienda su integración dentro del propio desarrollo del producto y no de manera aislada.

3.2 Casos de Prueba.

La ejecución de las pruebas conlleva a la realización de una serie de actividades, descritas por (Vegas, 2005) como:

1. Diseño de las pruebas: Comprende la identificación de la técnica o técnicas de pruebas que se utilizarán para probar el software. Distintas técnicas de prueba ejercitan diferentes criterios como guía para realizar las pruebas.

2. Generación de los casos de prueba: Consiste en la confección de los distintos casos de prueba según la técnica o técnicas identificadas previamente. La generación de cada caso de prueba debe ir acompañada del resultado que ha de producir el software al ejecutar dicho caso para detectar un posible fallo en el programa. Los casos de prueba determinan un conjunto de entradas, condiciones de ejecución y resultados esperados para un objetivo particular. Cada técnica de pruebas proporciona unos criterios distintos para generar estos casos o datos de prueba.

3. Definición de los procedimientos de la prueba: Conlleva a una especificación de cómo se va a llevar a cabo el proceso, quién lo va a realizar y cuándo.

4. Ejecución de la prueba: Es el momento de aplicar los casos de prueba generados previamente e identificar los posibles fallos producidos al comparar los resultados esperados con los resultados obtenidos.

3.2.1 Estrategias de Pruebas aplicadas.

Para el desarrollo de las pruebas de la capa de datos se escogen las de caja negra. Éstas se denominan:

- Pruebas de caja opaca
- Pruebas funcionales
- Pruebas de entrada/salida
- Pruebas inducidas por los datos

3.3 Pruebas de caja negra.

Son aquellas que trabajan con la interfaz del sistema. Verifican los dominios de entrada y salida del programa o programas para descubrir errores de funcionalidad, comportamiento y rendimiento. . El objetivo es demostrar que las funciones del software son operativas, que las entradas se aceptan de forma adecuada y se produce un resultado correcto, y que la integridad de la información externa se mantiene. Fundamentado en el estudio de la especificación de las funciones, la entrada y la salida para derivar los casos. La prueba consiste en probar todas las posibles entradas y salidas del programa sin considerar el código en lo absoluto.

Ilustración 18: Estrategia de Prueba Aplicada.

Las pruebas de caja negra se aplican al sistema con el objetivo de encontrar posibles errores en la optimización del Acceso a Datos. Se trata de hallar una " tasas de errores" para dar una medida del número de requisitos que se han probado. Los resultados han determinado márgenes específicos de errores, demostrando la valides de las pruebas aplicada, permitiendo corregir los errores lanzados.

El conjunto de datos posibles suele ser muy amplio (por ejemplo, la Gestión de Contextos).

Caso de Uso: Adicionar Contexto

En la siguiente tabla se muestra un fragmento de ejecución a un Requisito Funcional, desarrollado en el documento Diseño de casos de pruebas anexo a este trabajo de diploma.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Adicionar Contexto	Permite adicionar un contexto	EP 1.1: Adicionar Contexto	Seleccionar opción adicionar “El usuario selecciona la opción de Adicionar y se carga la opción de Configurar

			Contexto.
		EP 1.2: Seleccionar el contexto a almacenar	Carga un fichero, para guardar su URL de ubicación, permitiéndole al usuario la especificación del enunciado del contexto.
		EP 1.3: Validar Datos Enunciado del contexto Dirección del fichero	Comprueba que los datos insertados sean correctos.
		EP 1.4: Aceptar	Se guarda la información referente al contexto en el fichero (contexto.XML)
		EP 1.5 Devolver Resultados de la Acción.	Retorna al usuario a la pantalla Gestionar Contexto, mostrándole el listado de Contextos.

Id del escenario	Escenario	Ubicación de Contexto	Descripción de Contexto	Respuesta del sistema	Resultado de la prueba
EP 1.1	Adicionar Contexto	V	V	El sistema muestra un mensaje operación exitosa.	Mensaje "Contexto Adicionado Correctamente".
		I	V	El sistema emite un mensaje de error.	Mensaje "Datos Incorrectos"
		V	I	El sistema emite un mensaje de error.	Mensaje "Datos Incorrectos"

[Las celdas de la tabla contienen **V, I.** V indica válido**, I** indica inválido.]

Tabla Juegos de datos a probar (Adicionar Contexto).

3.4 Pruebas de caja blanca.

Se denomina cajas blancas a un tipo de pruebas de software que se realiza sobre las funciones internas de un módulo. Las pruebas de caja blanca están dirigidas a las funciones internas. Entre las técnicas usadas se encuentran; la cobertura de caminos (pruebas que hagan que se recorran todos los posibles caminos de ejecución), pruebas sobre las expresiones lógico-aritméticas, pruebas de camino de datos (definición-uso de variables), comprobación de bucles (se verifican los bucles para 0,1 y n iteraciones, y luego para las iteraciones máximas, máximas menos uno y más uno. (E.T.S.I., 2000)

3.4.1 Herramientas para la realización de pruebas de caja blanca.

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: (Angel, 2009)

- Ejecuta los archivos de prueba en un entorno independiente para evitar interferencias entre las diferentes pruebas.
- Las pruebas son fáciles de leer y sus resultados también lo son. En los sistemas operativos que lo soportan, los resultados 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.

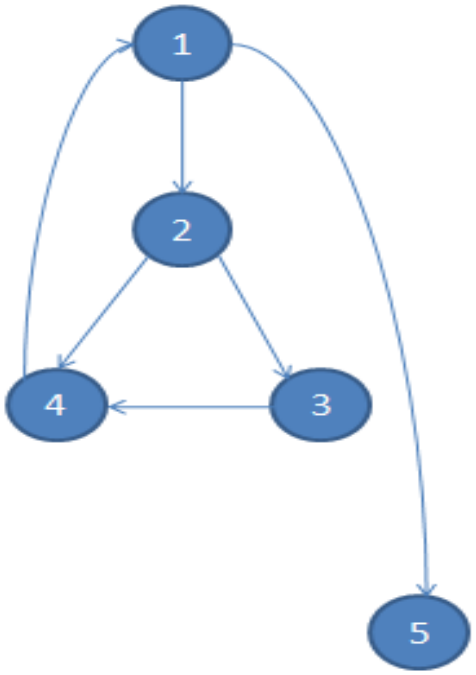
A continuación se muestran algunos ejemplos de las pruebas de caja blanca realizadas al sistema.

- "Obtener Contexto": Permite obtener un contexto dado su id.

```
public  function obtenerContextoByID($id_contexto, $contextos) {
     $contexto= null;                                   //1
     foreach ($lista as $index => $obj) {               //1
         if ($obj->getId() == $id_contexto)             //2
             $contexto = $obj;                          //3
     }                                                  //4
     return $contexto;                                  //5
}
```

Ilustración 19: Método obtenerContextoByID($id_contexto, $contextos).

En la ilustración 19 se muestra el código correspondiente al método obtenerContextoByID ($id_contexto, $contextos) en el cual, pasándole por parámetro el id de un contexto y el listado de contextos hace una búsqueda dentro de la lista y devuelve el contexto con dicho id.

Complejidad ciclomática V (G) = A – N + 2 V (G) = 6 – 5 + 2 V (G) = 3	
Posibles caminos - 1-2-3-4-5 - 1-2-4-5 - 1-5	

Juegos de datos definidos para la prueba.

```
$arr = array(array("uploads/Contextos/Modelo de datos.jpg", ".jpg", "Modelo de datos.jpg",
             "este es el modelo de datos", "1"), array("uploads/Contextos/001.jpg" , ".jpg",
             "001.jpg", "Imagen" , "2"));

$contexto= null;
foreach ($arr as $key => $val) {
        $c= new Contexto();
        $c->setURL($val[0]);
        $c->setFormat($val[1]);
        $c->setFileName($val[2]);
        $c->setEnunciado($val[3]);
        $c->setId($val[4]);
        $contexto[]= $c;
}


$Cont = new Contexto_Modelo();
$var1 = $Cont->obtenerContextoByID(1, $contexto);
$t->is($var1, "Modelo de datos.jpg", 'Correcto');

$var2 = $Cont->obtenerContextoByID(4, $contexto);
$t->is($var2, false, 'Incorrecto, No se encontró ningun contexto con dicho id');
```

Ilustración 20: Juegos de datos para la prueba del método obtenerContextoByID().

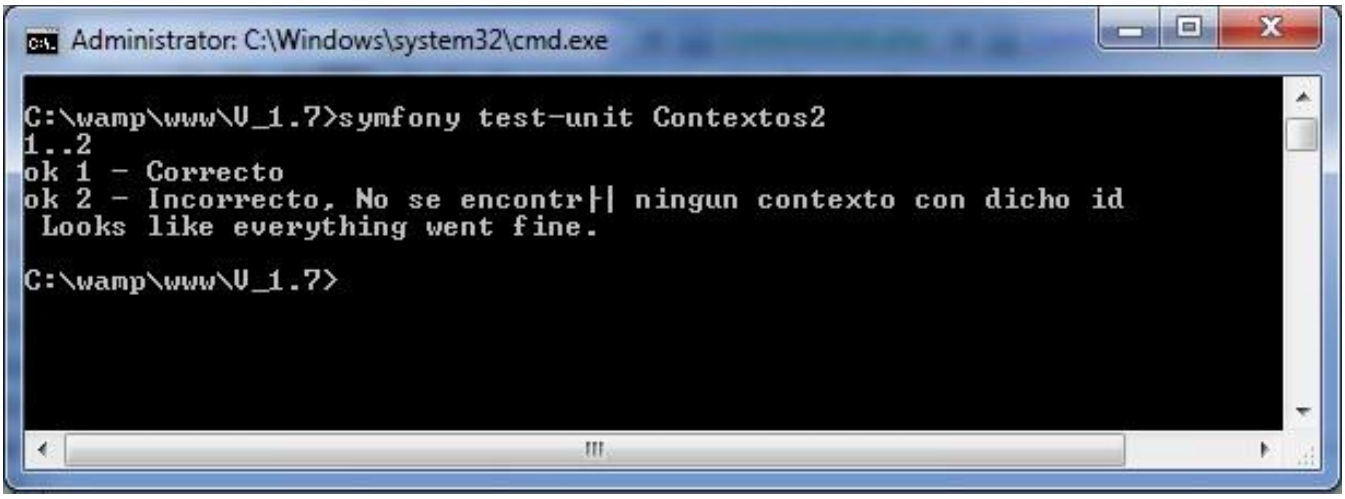


Ilustración 21: Resultados de la prueba del método obtenerContextoByID() con la herramienta Lime.

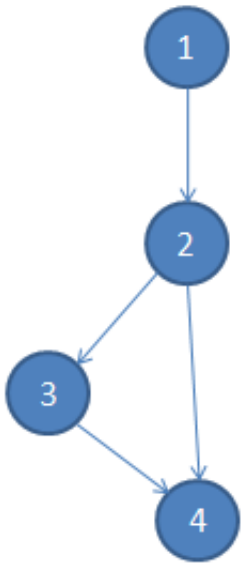
Como se muestra en la Ilustración 21, y dados los juegos de datos introducidos como se muestra en la ilustración 20, se valida el correcto funcionamiento del metodo obtenerContextoByID(), mostrado en la Ilustración 19.

- "Obtener nombre de un contexto": Permite obtener el nombre de un contexto dado su id.

```
public function getNombre($id, $listado) {
    $name = false;
    foreach ($listado as $value) { //1
        if ($value->getId() == $id) {//2
            $name = $value->getFileName(); //3
            break;
        }
    }
    return $name; //4
}
```

Ilustración 22: Método getNombre($id, $listado).

En la ilustración 22 se muestra el código correspondiente al método getNombre ($id_contexto, $contextos) en el cual, pasándole por parámetro el id de un contexto y el listado de contextos hace una búsqueda dentro de la lista y devuelve el nombre del contexto cuyo dicho id coincida con el recibido.

Complejidad ciclomática V (G) = A – N + 2 V (G) = 4 – 4 + 2 V (G) = 2	
Posibles caminos - 1-2-3-4 - 1-2-4	

Juegos de datos definidos para la prueba.

```
$arr = array(array("uploads/Contextos/Modelo de datos.jpg", ".jpg", "Modelo de datos.jpg",
             "este es el modelo de datos", "1"), array("uploads/Contextos/001.jpg" , ".jpg",
             "001.jpg", "Imagen" , "2"));
$contexto= null;
foreach ($arr as $key => $val) {
        $c= new Contexto();
        $c->setURL($val[0]);
        $c->setFormat($val[1]);
        $c->setFileName($val[2]);
        $c->setEnunciado($val[3]);
        $c->setId($val[4]);
        $contexto[]= $c;
}

$Cont = new Contexto_Modelo();
$var1 = $Cont->getNombre(2, $contexto);
$t->is($var1, "001.jpg", 'esta ok');

$var2 = $Cont->getNombre(4, $contexto);
$t->is($var2, false, 'No se encontró');
```

Ilustración 23: Juegos de datos para la prueba del método getNombre().

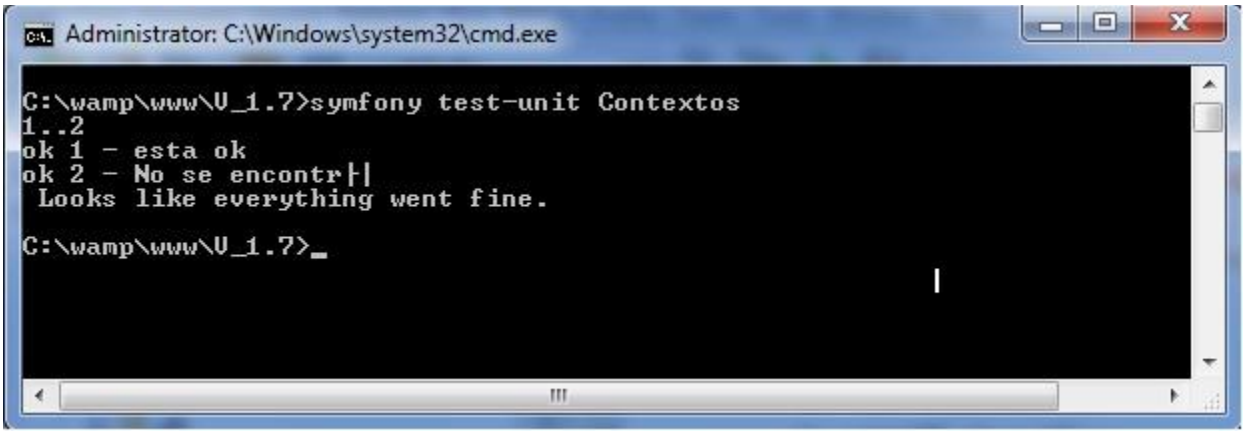


Ilustración 24: Resultados de la prueba del método getNombre() con la herramienta Lime.

Como se muestra en la Ilustración 24, y dados los juegos de datos introducidos como se muestra en la ilustración 23, se valida el correcto funcionamiento del metodo obtenerContextoByID(), mostrado en la Ilustración 22.

3.5 Prueba de Rendimiento o Carga.

Objetivo

Verificar el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado y así como la verificación de la capacidad del sistema para manejar volúmenes de datos extremos de acuerdo a la velocidad que se especifique para el sistema.

Metas

- Comprobar los tiempos de respuesta del sistema en una cantidad limitada de escenarios de trabajo (a nivel de número de usuarios y número de transacciones), bajo una configuración de hardware y software constante.
- Comprobar el tiempo de respuesta al realizar una función.
- Comprobar el tiempo de respuesta al realizar accesos concurrentes a una determinada información.
- Atender múltiples solicitudes de parte de los actores que acceden a un mismo recurso.

  En las siguientes ilustraciones se establece una comparación de las pruebas de rendimiento o cargas aplicadas al Sistema Tutor Inteligente, VIRTEVALL, en el módulo Perfil de Estudiantes.

GET btn-arrow-light.gif	200 OK	localhost	916 B	459ms
GET right-btn.gif	200 OK	localhost	871 B	458ms
GET glass-bg.gif	200 OK	localhost	873 B	458ms
GET btn.gif	200 OK	localhost	4.2 KB	444ms
GET tab-strip-bg.gif	200 OK	localhost	835 B	439ms
GET tabs-sprite.gif	200 OK	localhost	2.1 KB	441ms
GET corners-sprite.gif	200 OK	localhost	1.4 KB	447ms
GET top-bottom.gif	200 OK	localhost	875 B	448ms
GET left-right.gif	200 OK	localhost	815 B	449ms
GET tool-sprites.gif	200 OK	localhost	4.3 KB	291ms
GET perfil?_dc=13072466	200 OK	localhost	174 B	576ms
GET objetivosNCumplidos'	200 OK	localhost	381 B	2.53s
GET ObjetivosFase?_dc=1	200 OK	localhost	1.3 KB	3.07s
GET objetivosNCumplidos'	200 OK	localhost	381 B	2.79s
GET registros?_dc=13072	200 OK	localhost	41 B	1.82s
GET ActPorObjetivo?_dc=	200 OK	localhost	28 B	2.07s
32 pedidos			913.5 KB (882.2 KB desde la caché)	13.35s (onload: 9.67s)

Ilustración 25: Pruebas de rendimiento o carga aplicado al software antes de la optimización del acceso a datos

Los resultados expuestos en la Ilustración 19 muestran el tiempo de respuesta de la aplicación en el proceso de gestión del Perfil de Estudiante, antes de establecer la optimización del Acceso a datos del sistema.

La cual en la versión establecida en este trabajo de diploma reduce el tiempo de respuesta para el mismo módulo, bajo las mismas condiciones de ejecución. Demostrando los resultados de optimización obtenidos en la Ilustración 20. Determinando la factibilidad de la prueba de rendimiento o carga del software y su efectividad.

GET btn-arrow-light.gif	200 OK	localhost	916 B	445ms
GET right-btn.gif	200 OK	localhost	871 B	444ms
GET glass-bg.gif	200 OK	localhost	873 B	507ms
GET btn.gif	200 OK	localhost	4.2 KB	499ms
GET tab-strip-bg.gif	200 OK	localhost	835 B	416ms
GET tabs-sprite.gif	200 OK	localhost	2.1 KB	417ms
GET corners-sprite.gif	200 OK	localhost	1.4 KB	406ms
GET top-bottom.gif	200 OK	localhost	875 B	408ms
GET left-right.gif	200 OK	localhost	815 B	410ms
GET tool-sprites.gif	200 OK	localhost	4.3 KB	251ms
GET perfil?_dc=13072	200 OK	localhost	175 B	508ms
GET objetivosNCumplic	200 OK	localhost	381 B	1.99s
GET ObjetivosFase?_d	200 OK	localhost	1.3 KB	3.04s
GET objetivosNCumplic	200 OK	localhost	381 B	2.46s
GET registros?_dc=13	200 OK	localhost	41 B	2.81s
GET ActPorObjetivo?_	200 OK	localhost	28 B	2.26s
31 pedidos			885 KB (853.5 KB desde la caché)	5.97s (onload: 2.34s)

Ilustración 26: Pruebas de rendimiento o carga aplicado al software después de la optimización del acceso a datos

3.6 Criterios de validación:

Para la certificación del éxito o no de una prueba se tiene en cuenta el análisis de los resultados esperados contenidos en la descripción de los casos de prueba con los resultados reales. Se considera:

- Prueba Satisfactoria: Si el resultado esperado es idéntico al resultado obtenido.
- Prueba Insatisfactoria: Si el resultado esperado es diferente al resultado obtenido.

El equipo encargado de las pruebas determinó las no conformidades en los fallos en la validación del sistema y las erradicó, determinado continuar con los restantes casos de prueba y dar por finalizada la validación del sistema.

Conclusiones

Una correcta estrategia de validación y verificación ayuda a mejorar y asegurar la calidad de su desarrollo y permite reducir los costos de corrección de errores. Las ventajas son claras: un mayor control sobre el proceso, una identificación temprana de errores, problemas y una reducción drástica de los costos para subsanar dichos errores, por eso la atención esmerada que se le presta a esta actividad.

Las pruebas realizadas dieron la medida de cómo el acceso a datos iba mejorando en toda su totalidad, donde los tiempos de respuesta de la aplicación fueron inferiores, comparados con la versión anterior del software.

CONCLUSIONES GENRALES

Se realizó un profundo análisis de los temas referentes a gestores de bases de datos, acceso a datos y mecanismos de persistencia y serialización, imprescindibles para darle cumplimiento al objetivo de la investigación. Además se realizó un estudio y selección de herramientas y tecnologías en el marco teórico de la investigación lo cual tributó al desarrollo de la solución propuesta. Con el empleo de la metodología de desarrollo y las herramientas estudiadas, se obtuvo un diseño y se implementaron los mecanismos de persistencia y serialización de datos además de las funciones en el gestor de bases de datos, para darle cumplimiento a la propuesta de solución. Mediante el uso de pruebas y la aplicación de métricas, se pudo validar la propuesta presentada obteniendo resultados satisfactorios. Se concluye que la puesta en práctica de los mecanismos desarrollados para la optimización del acceso a datos, disminuye el tiempo de respuesta de la aplicación cumpliéndose así los objetivos planteados en el presente trabajo de diploma.

RECOMENDACIONES

Con vista a dar continuidad al desarrollo de este proyecto se recomienda:

- Implementar nuevas funciones en la base de datos que ayuden al mejor funcionamiento del sistema.
- Probar el sistema a gran escala.
- Aplicar técnicas de optimización para la base de datos con el objetivo de lograr un rendimiento óptimo de la misma.
- Presentar el presente trabajo en futuros eventos científicos.

BIBLIOGRAFÍA

.PHP.net. 2009. www. *www.PHP.net.* [En línea] 29 de diciembre de 2009. [Citado el: 24 de enero de 2011.] http://www.php.net/manual/es/intro-whatcando.php..

Barrio, López. 2005. Metodología de Desarrollo: Programación Extrema. Tesis (Doctorado en Ingeniería de Sistemas Electrónicos para Entornos Inteligentes. *Metodología de Desarrollo: Programación Extrema. Tesis (Doctorado en Ingeniería de Sistemas Electrónicos para Entornos Inteligentes.* [En línea] Dpto. Ingeniería Electrónica, 25 de noviembre de 2005. [Citado el: 10 de abril de 2011.] http://www-lsi.die.upm.es/~ca.

Bourred, Ronald. 2005. XML and Data bases. [En línea] Septiembre de 2005. [Citado el: 11 de Noviembre de 2010.] http://www.rpbourret.com/xml/XMLAndDatabases.htm#datavdocs.

Brito Acuña, Karenny. 2009. Selección de Metodologías de Desarrollo para Aplicaciones Web en la Facultad de Informática de la Universidad de Cienfuegos. *Selección de Metodologías de Desarrollo para Aplicaciones Web en la Facultad de Informática de la Universidad de Cienfuegos.* [En línea] 2009. www.eumed.net/libros/2009c/584/..

Castillo, ReinierGonzález. 2008. Módulo Gestión del Aprendizaje del Centro Virtual de Autoaprendizaje de lenguas Extranjeras. [En línea] 2008. [Citado el: 12 de 10 de 2010.] http//biblioteca.uci.cu.

Cleger Despaigne, Eliober y Tornés Montes de Oca, Annarella María. Análisis y Diseño de una herramienta interactiva de simulación de procesos: Nodo Virtual de Procesos. Segunda Iteración. Ciudad de la Habana : s.n.

E.T.S.I. 2000. s.l. : Departamento de Lenguajes y sistema Informáticos. [En línea] Universidad de Granada., 2000.

EEES, Alfin. 2009. mariapinto. *mariapinto.* [En línea] 2009. [Citado el: 4 de 11 de 2010.] http://www.mariapinto.es/alfineees/autonomo/que.htm .

Elers Pérez, Alberto y Sariol Pérez. 2009. Implementación del Módulo Gestión de Actividades del Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas. 2009.

extjses. 2010. [En línea] 2010. [Citado el: 13 de enero de 2010.]] http://extjses.com. .

Frabicio. www.definición.org. *www.definición.org.* [En línea] [Citado el: 18 de diciembre de 2010.] http://www.definicion.org/lenguaje-de-programacion..

Fragoso Vázquez, Amado Víctor y Alférez Salinas, Germán Harvey. 2005. *Un Enfoque para la Aplicación de las Tácticas de la Arquitectura con el Fin de Ampliar la Documentación de sus Calidades.* 2005.

Giraffa, L. M. M. 1997. *Selección y adopción de estrategias de educación en Sistemas Tutores Inteligentes.* Porto Alegre CPGCC/UFRGS : s.n., 1997.

Gómez, Omar Salvador. 2007. Evaluando Arquitecturas de Software. *Evaluando Arquitecturas de Software.* [En línea] 2007. http//:biblioteca.uci.cu.

Inffiet. 2007. Sitio de Descargas de Software. *Sitio de Descargas de Software.* [En línea] 5 de mayo de 2007. http://www.freedownload manager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/ .

informática, Instituto Nacional de Estadística. 2000. *Herramientas Case.* 2000.

J., HYDE. 2002. *Base de datos.* 2002.

Katiuska, Losada León Efraín y Cuba García. 2005. Capas de acceso a datos. *"Sistema de Control y Gestión de las Interrupciones y Servicios de las Tecnologías de la Información y las Comunicaciones".* 2005.

LAGO. 2007. proactiva-calidad. [En línea] 2007. http://www.proactiva-calidad.com/java/patrones/index.html.

Learning, CEDS y. Nauta. *Nauta.* [En línea] [Citado el: 11 de marzo de 2011.] http://ceds.nauta.es/informes/case04.htm .

Luciano. 2009. Entornos de Desarrollo Integrado para Java. [En línea] 2009. [Citado el: 19 de diciembre de 2011.] http://luauf.com/2008/05/13/entornos-de-desarrollo-integrado-para-java..

newcomers.php. 2006. newcomers.php. *newcomers.php.* [En línea] 2006. [Citado el: 4 de febrero de 2011.] http://www.eclipse.org/home/newcomers.php..

O'Reilly. 2010. XML.com: A technical introduction to XML. [En línea] 2010. http://www.xml.com/pub/a/98/10/guide0.html.

Pizarro. 2011. Datos y acceso a datos. [En línea] 2011. [Citado el: 22 de enero de 2011.] http://msdn.microsoft.com/es-es/vbasic/ms789183.

Potencier, Fabien y Zaninotto, François. 2005. Symfony la guía definitiva. 2005.

Potencier, Fabien y Zaninotto, François. 2005. *Symfony la guía definitiva.* 2005.

Quesada. 2005. 2005.

Reina. *2006.*

Sicilia, Miguel-Angel. 2009. *Métricas de la Calidad del Diseño Orientado a Objetos del Software.* 2009.

Technologies., Boost Productivity with Innovative and Intuitive. 2010. visual-paradigm. *visual-paradigm.* [En línea] 2010. [Citado el: 11 de 11 de 2010.] http://www.visual-paradigm.com/aboutus/10reasons.jsp .

VALDÉS. 2000. ¿Qué son las bases de datos? *¿Qué son las bases de datos?* [En línea] 2000. http://www.maestrosdelweb.com/principiantes/%C2%BFque-son-las-bases-de-datos/.

Villanueva., Mª Luisa. 1997. *Los Estilos de aprendizaje de Lenguas. .* 1997.

Wesley, Addison. 1995. *Elements of Reusable Object-Oriented Software.* 1995.

GLOSARIO

STI: Sistema Tutor Inteligente. Giraffa (1997) (Inffiet, 2007) los delimita como: “un sistema que incorpora técnicas de IA (Inteligencia Artificial) a fin de crear un ambiente que considere los diversos estilos cognitivos de los alumnos que utilizan el programa”.

HTML: HyperText Markup Language.

JavaScript: lenguaje de scripts, interpretado, multiplataforma y parcialmente orientado a objetos.

Ajax: Asynchronous JavaScript And XML.

API: Conjunto de funciones y procedimientos (o métodos si se refiere a programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Herramienta: Aplicación empleada para la construcción de otros programas o aplicaciones.

IDE: Entorno de Desarrollo Integrado.

Plataforma: sistema operativo o sistemas complejos, ya sea de hardware o software, sobre el cual un programa pueda ejecutarse. Ejemplos típicos incluyen: arquitectura de hardware, lenguajes de programación y sus librerías de tiempo de ejecución. Ejemplos de plataformas son PC (Windows) y Macintosh (Mac).

BD: Base de Datos.

1

2