“Universidad de las Ciencias Informáticas”
UCI

Implementación del Módulo de Diagnóstico del Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas.

Tesis de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas.

Autores: Dayner Merayo Rodríguez
Maykel Isaac Pacheco

Tutores: M.Sc. Yoan Martínez Márquez.
Ing. Reinier Castillo González.
Lic. Moisés A. Mayet Solano.

La Habana, Cuba.
2010

DECLARACIÓN DE AUTORÍA

Declaramos ser los autores del presente trabajo de diploma y autorizamos a la Universidad de las Ciencias Informáticas para que hagan uso pertinente de este trabajo.

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

_____ _____

Firma de Autor Firma de Autor

_____ _____

Firma Tutor Firma Tutor

_____

Firma Tutor

AGRADECIMIENTOS

- ***A mi familia**: como una muestra de mi cariño y agradecimiento, por todo el amor y el apoyo brindado y porque hoy veo llegar a su fin una de las metas de mi vida, le agradezco la orientación que siempre me han otorgado.*
- ***A mis amigos**: al término de esta etapa de mi vida, les quiero expresar un profundo agradecimiento por su ayuda, apoyo y comprensión, me alentaron a lograr esta hermosa realidad.*
- ***A mis tutores**: como un testimonio de gratitud y eterno reconocimiento, por el apoyo que siempre me han brindado y con el cual he logrado terminar mi carrera profesional.*
- ***A mi compañero de tesis**: no es fácil llegar, se necesita ahínco, lucha y deseo, pero sobre todo apoyo como el que he recibido durante este tiempo. Ahora más que nunca se acredita mi cariño, admiración y respeto. Gracias por lo que hemos logrado.*
- ***A mi novia**: por la oportunidad de existir, por su sacrificio en algún tiempo incomprendido, por su ejemplo de superación incansable, por su comprensión y confianza, por su amor incondicional, porque sin su apoyo no hubiera sido posible la culminación de mi carrera profesional.*

Dayner Merayo Rodríguez

- ***A mi familia** por contribuir a mi formación y por la confianza que depositaron en mí y en especial a **mi madre** porque a ella le debo todo lo que soy.*
- ***A mis tutores**, por su abnegada y constante preocupación por este trabajo, por sus experiencias y profesionalismo a la hora de inculcarme los conocimientos.*
- ***A todos los amigos** que han estado conmigo en las buenas y en las malas desde primer año.*
- ***A mi compañero de tesis** porque sin él, no hubiera sido posible el desarrollo de este trabajo.*
- ***A mi novia** por apoyarme en todo lo que necesité.*
- *A todos ellos desde el fondo de mi corazón quiero expresarles mi agradecimiento.*

A todos ellos desde el fondo de mi corazón quiero expresarles mi agradecimiento.

Muchas gracias,

***Maykel Isaac Pacheco**.*

DEDICATORIA

El resultado de dicho ***Trabajo de Diploma*** *se lo dedico* ***a toda mi familia*** *sabiendo que jamás existirá una forma de agradecer en esta vida de lucha y superación constante todo el esfuerzo derrochado para que se lograra esta meta, deseo expresarles que mis ideales, esfuerzos y logros han sido también suyos y constituye el legado más grande que pudiera recibir.*

Con cariño, admiración y respeto.

Dayner.

El resultado de dicho ***Trabajo de Diploma*** *se lo dedico* ***a mi madre*** *por haber estado presente en todo momento y por el apoyo que me ha brindado durante toda mi vida,* ***al resto de mi familia*** *por haber ayudado en mi formación como profesional y por haber confiado en mí.*

Maykel Isaac Pacheco.

RESUMEN

El proceso de formación ha sufrido cambios con el objetivo de mejorar muchos aspectos que son considerados de gran importancia en el proceso de aprendizaje. La inserción de las potencialidades de las Tecnologías de la Información y las Comunicaciones (TICs) a dado paso al surgimiento de nuevos métodos como el Auto-aprendizaje, para lo cual han surgido espacios virtuales, aulas virtuales, tutores inteligentes, entre otros.

La Universidad de las Ciencias Informáticas (UCI) es un centro que está a la vanguardia de todo el proceso de enseñanza y Auto-aprendizaje, 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 del diagnóstico de sus necesidades, estilos y estrategias de aprendizaje y nivel de idioma que poseen.

La presente investigación persigue como objetivo implementar el módulo gestión de diagnóstico del Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas para lo cual es usado Symfony 1.2.8 como framework para aplicaciones web, como lenguaje de programación empleado PHP 5.2.5 y como IDE NetBeans 6.8 M2. Además como SGBD se utilizó PostgreSQL- 8.3.4-1.

Para la implementación del módulo se realizó una valoración crítica del diseño, sobre lo propuesto por los analistas, se hizo un estudio acerca de 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 técnicas de prueba de software que permitieron comprobar la validez de la solución propuesta.

PALABRAS CLAVE

Tutor, Evaluación, Aprendizaje, Autónomo, Diagnóstico, Implementación.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Implementación del Módulo de Diagnóstico del Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas.

INTRODUCCIÓN

El uso irremplazable de las Tecnologías de la Información y las Comunicaciones (TIC) se está haciendo más evidente en las últimas décadas. El desarrollo de las mismas favorece que los procesos de la humanidad se realicen con más calidad, control y velocidad, siendo posible acortar innumerables situaciones complicadas en la sociedad y aumentar la fiabilidad y efectividad de los bienes y servicios que brindan en la actualidad el uso de las TIC.

El sistema educativo no puede quedar al margen de los nuevos cambios. Debe atender a la formación de los nuevos ciudadanos y la incorporación de las nuevas tecnologías ha de hacerse con la perspectiva de favorecer el aprendizaje y facilitar los medios que sustenten el desarrollo de los conocimientos y de las competencias necesarias para la inserción social y profesional de calidad, por tal motivo se han realizado diversas herramientas informáticas tales como: software educativos, multimedia, entornos virtuales de aprendizaje y sistemas tutores inteligentes (STI). Los STI utilizan técnicas de inteligencia artificial (IA) para la representación del conocimiento. Además, incorporan métodos y técnicas de enseñanzas con el fin de mejorar el rendimiento del estudiante. Se han desarrollado varios STI en el mundo desde su aparición en los años setenta, entre los más destacados se encuentra el *Scholar* (Carbonell, 1970) que es considerado como el primer STI, *Why* (Stevens, et al., 1977), *Sophie* (Brown, et al., 1982), *West* (Burton, 1982.), *Buggy* (Brown, et al., 1978), *Steamer* (Stevens, et al., 1977), *Meno* (Wolf, 1984), *Proust* (Johnson, 1986), *Sierra* (Vanlehn, 1988).

En nuestro país existe uno de estos sistemas, tal es el caso del *Sistema Tutorial Inteligente para Diagnóstico y Tratamiento de Infecciones de Transmisión Sexual (STIITS)*. El mismo fue desarrollado en la Universidad de Cienfuegos y presentado en el VII Congreso Internacional de Informática en Salud en febrero del 2009. Este sistema soluciona una necesidad social y tiene además un valor agregado por poder ser usado además en la docencia.

En la Universidad de las Ciencias Informáticas (UCI) se cuenta con disímiles herramientas que contribuyen al proceso de Auto-aprendizaje de los estudiantes. Idioma Extranjero es una de las materias en la Universidad vinculada con el uso de una de estas herramientas, la cual recibe el nombre de Entorno

Virtual de Aprendizaje (EVA). La misma permite a los estudiantes la realización de numerosos diagnósticos de ejercitación y consolidación de sus conocimientos y a la vez los estudiantes reciben una calificación de los mismos. A pesar de todas las facilidades que brinda esta herramienta, no es capaz de guiar y monitorear el avance durante el aprendizaje de los estudiantes en la asignatura, no logra reconocer el estilo de aprendizaje del estudiante ni el nivel académico. Como una variante para darle solución a las desventajas que presenta el EVA se había comenzado a desarrollar un Centro Virtual de Auto-aprendizaje de Lenguas Extranjeras (CEVALE) en la Facultad 9 pero debido a múltiples razones que no son objeto de esta investigación no se pudo finalizar.

Motivado por la necesidad de encontrar diferentes formas alternativas de evaluación del aprendizaje autónomo de idiomas extranjeros, se decidió poner en marcha el proyecto de Innovación Pedagógica: Metodología de Evaluación para el Aprendizaje Autónomo de Idiomas Extranjeros (VIRTEVALL), con el cual se tiene la intención de desarrollar un Sistema Tutor Inteligente que exhiba un comportamiento adaptativo, similar al de un tutor humano, que se adapte al desempeño del estudiante y sus estrategias de aprendizaje de acuerdo a sus estilos de aprendizaje, en lugar de ser un modelo rígido.

El proyecto de innovación pedagógica está estructurado sobre la base del sustento teórico provisto por una tesis doctoral sobre evaluación para aprendizaje autónomo de idiomas extranjeros. Como aporte práctico, ofrece el STI para implementar la metodología elaborada, el mismo está sustentado en el desarrollo de dos tesis de maestría que abordarán: la integración de las tecnologías en la gestión de la evaluación para aprendizaje autónomo de idiomas extranjeros y la integración de algoritmos de inteligencia artificial en la gestión de la evaluación para aprendizaje autónomo de idiomas extranjeros.

En una primera etapa, por la complejidad del sistema a desarrollar, se definió por la dirección del proyecto la división en nueve módulos estructurados en dependencia de las diferentes funcionalidades identificadas, entre ellos se encuentra el de Diagnóstico, el cual tiene como su principal objetivo gestionar los diagnósticos que serán aplicados a los estudiantes matriculados en el sistema.

Se hace necesario precisar el siguiente **problema a resolver**: Necesidad de desarrollar un módulo que permita la gestión de diagnósticos para evaluar el aprendizaje de lenguas extranjeras en el Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas.

Implementación del Módulo de Diagnóstico del Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas.

Siendo el **objeto de estudio**: el proceso de desarrollo de software para el módulo de diagnóstico.

Presentando como **campo de acción** en esta investigación: informatización de los procesos de Gestión de Diagnósticos para el Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas.

Teniendo en cuenta el problema mencionado anteriormente se puede plantear la **idea a defender**: si se logra implementar correctamente el módulo de gestión de diagnósticos entonces se potenciarán las funcionalidades del Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas.

El **objetivo general** de la investigación es: Implementar el Módulo Gestión de Diagnósticos del Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas.

Para darle cumplimiento al objetivo trazado se ha decidido desarrollar las siguientes **tareas de la investigación:**

- Valoración de la implementación de los diferentes tutores autónomos virtuales que existen en el mundo.
- Valoración de las diferentes reglas de inteligencia artificial que permitan darle solución al módulo.
- Caracterización de diferentes sistemas que permitan la gestión de diagnósticos.
- Diseño de la interfaz del módulo gestión de diagnósticos.
- Determinación de los estándares de codificación.
- Consulta de los diagramas correspondientes a la implementación.
- Implementación del módulo gestión de diagnóstico.
- Desarrollo de Casos de Prueba que certifiquen la veracidad de los algoritmos empleados.

Posibles resultados:

- Prototipo funcional del módulo Gestión de Diagnósticos del Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas.

La investigación está conformada con la siguiente estructuración.

Implementación del Módulo de Diagnóstico del Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas.

Capítulo 1. Fundamentación teórica:

El presente capítulo aborda el estudio de las principales tecnologías y otros elementos seleccionados para la correcta construcción del software. A su vez se hace un estudio del estado del arte acerca de las diferentes herramientas que existen para la implementación del módulo y se seleccionan las más adecuadas.

Capítulo 2. Descripción y análisis de la solución propuesta:

Se describen los algoritmos implementados y se analiza la complejidad de los mismos, así como las clases que modelan la solución y las principales funcionalidades de cada una de ellas.

Capítulo 3. Validación de la solución propuesta:

Se refiere al diseño de las pruebas que permitan validar la solución propuesta, detallando su objetivo, alcance y tipo, así como los valores utilizados para la ejecución de dichas pruebas.

Implementación del Módulo de Diagnóstico del Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

La calidad en el desarrollo y mantenimiento de software se ha convertido en uno de los principales objetivos del desarrollo de software a nivel mundial. La Universidad de las Ciencias Informáticas sigue muy de cerca esta tendencia, es por ello que para desarrollar de manera eficiente y con la calidad necesaria y requerida el módulo de gestión de diagnósticos del Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas, la dirección y el equipo de arquitectura del proyecto seleccionaron el lenguaje de programación, la metodología de desarrollo y las herramientas necesarias para obtener un sistema que cumpla con las exigencias del cliente.

En el presente capítulo se hará alusión a las principales características del lenguaje de programación, metodología de desarrollo y herramientas definidas.

1.2 Metodología de Desarrollo de Software

Una metodología es un conjunto de procedimientos que permiten producir y mantener un producto software, definiendo una serie de pasos a seguir para obtener un software de calidad. Un proceso define quién está haciendo qué, cuándo, y cómo alcanzar un determinado objetivo. Las metodologías se desarrollan con el objetivo de dar solución a los problemas existentes en la producción de software, que cada vez son más complejos. Estas abarcan procedimientos, técnicas, documentación y herramientas que se utilizan en la creación de un producto de software. (Jacaboson, et al., 2000).

La dirección del proyecto VIRTEVALL seleccionó para la creación del producto el Proceso Unificado de Desarrollo (en inglés Rational Unified Process, habitualmente resumido como RUP). A continuación se estudian sus principales características y se enfoca específicamente al flujo de trabajo implementación el cual aporta elementos importantes en este estudio.

1.2.1 RUP

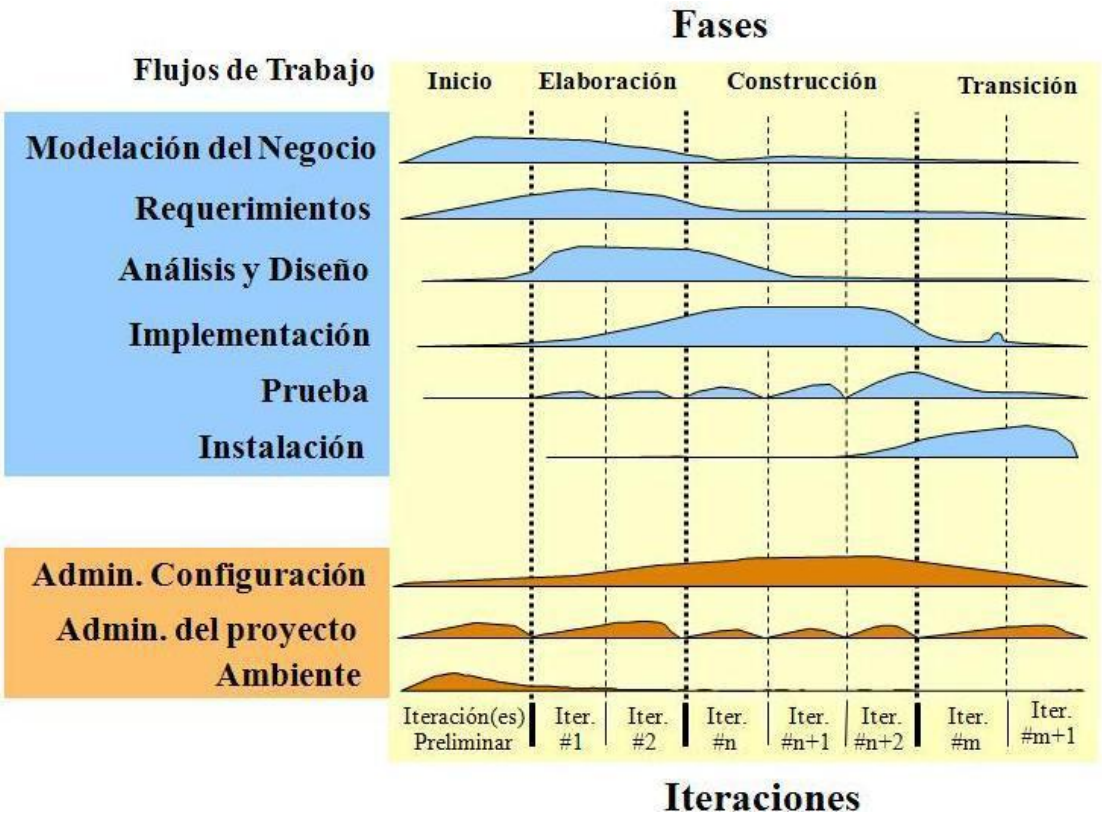


Figura 1: Ciclo de vida de RUP

RUP es una infraestructura flexible de procesos de desarrollo de software que ayuda proveyendo guías consistentes y personalizadas de procesos para todo el equipo de un proyecto. Cuenta con prácticas recomendadas y probadas que dan guía para conducir las actividades de desarrollo de un equipo y que son adoptadas en cientos de proyectos mundialmente y enseñadas en cientos de universidades. Usando esta metodología se podrán alcanzar resultados predecibles unificando el equipo con procesos comunes que optimicen la comunicación y creen un entendimiento común para todas las tareas, responsabilidades y artefactos. (Jacaboson, et al., 2000)

Esta metodología de desarrollo de software, en su modelación define como sus principales elementos:

- Trabajadores ("quién"): Define el comportamiento y responsabilidades (rol) de un individuo, grupo de individuos, sistema automatizado o máquina, que trabajan en conjunto como un equipo. Ellos realizan las actividades y son propietarios de elementos.
- Actividades ("cómo"): Es una tarea que tiene un propósito claro, es realizada por un trabajador y manipula elementos.
- Artefactos ("qué"): Productos tangibles del proyecto que son producidos, modificados y usados por las actividades. Pueden ser modelos, elementos dentro del modelo, código fuente y ejecutables.
- Flujo de actividades ("Cuándo") Secuencia de actividades realizadas por trabajadores y que produce un resultado de valor observable.

RUP presenta 3 características fundamentales, las mismas se mencionan a continuación: Es un proceso dirigido por casos de uso, centrado en la arquitectura e iterativo incremental. Esta metodología agrupa las actividades en grupos lógicos definiéndose 9 flujos de trabajo principales. En este caso sólo se expone el flujo de trabajo que es necesario conocer para el desarrollo de la investigación:

- **Implementación:** En este flujo de trabajo se implementan las clases y objetos en ficheros fuente, binarios, ejecutables y demás. Además se deben hacer las pruebas de unidad. El resultado final de este flujo de trabajo es un sistema ejecutable.

La implementación tiene entre sus propósitos definir la organización de código, en términos de sistemas de implementación organizados en capas. Implementar los elementos diseñados en términos de elementos de implementación y probar los componentes desarrollados como unidades. Por último integrar los resultados producidos por desarrolladores individuales o por equipos, en un sistema ejecutable. (Jacaboson, et al., 2000).

El artefacto más importante de este flujo es el **Modelo de Implementación** el cual incluye los siguientes elementos:

- Subsistemas de implementación y sus dependencias, interfaces y contenidos.
- Componentes, incluyendo componentes fichero y ejecutables, y las dependencias entre ellos. Los componentes son sometidos a pruebas de unidad.
- La vista de arquitectura del modelo de implementación, incluyendo sus elementos arquitectónicamente significativos.

El arquitecto es el responsable de la integridad, corrección, completitud y legibilidad del modelo de implementación de acuerdo a lo descrito en el modelo de diseño, pero el encargado de implementar los elementos de este modelo es el implementador.

El **implementador** escribe código fuente, reutiliza código, compila, realiza pruebas unitarias. Si encuentra defectos en el diseño, regresa a la fase de análisis y diseño y los corrige, también arregla defectos de código y realiza pruebas unitarias para verificar los cambios. (Jacaboson, et al., 2000).

El flujo de implementación está fuertemente determinado por el **lenguaje de programación** mediante el cual se llevará a cabo el desarrollo del producto de software.

1.3 Paradigmas de Programación

Un paradigma de programación es un modelo básico de diseño e implementación de programas. Un modelo que permite desarrollar programas conforme a ciertos principios o fundamentos específicos que se aceptan como válidos. En otras palabras, es una colección de modelos conceptuales que juntos modelan el proceso de diseño, orientan la forma de pensar y solucionar los problemas y, por lo tanto, determinan la estructura final de un programa. (Facultad Regional de Buenos Aires, 2005)

1.3.1 Programación Lógica

La programación lógica, junto con la funcional, forma parte de lo que se conoce como programación declarativa. En los lenguajes tradicionales, la programación consiste en indicar cómo resolver un problema mediante sentencias; en la programación lógica se trabaja de una forma descriptiva, estableciendo relaciones entre entidades, indicando no cómo, sino qué hacer. La ecuación de Robert Kowalski (Universidad de Edimburgo) establece la idea esencial de la programación lógica: algoritmos = lógica +

control. Es decir, un algoritmo se construye especificando conocimiento en un lenguaje formal (lógica de primer orden), y el problema se resuelve mediante un mecanismo de inferencia (control) que actúa sobre aquél. (Rossel, 2004)

Este paradigma permite comprobar hipótesis mediante la representación de un problema a partir de axiomas, se basa en el cálculo de predicados. Es muy utilizado en aplicaciones de Inteligencia Artificial como Sistemas de Expertos, en los que un sistema de información simula las sugerencias de un experto en un área de conocimiento, demostraciones de teoremas y programas para el reconocimiento del lenguaje natural, entre otros.

1.3.2 Programación Estructurada

La Programación Estructurada es un paradigma en el cual la escritura de las partes de un programa se realiza de la manera más clara posible a través de tres estructuras lógicas de control:

- Secuencia: esta estructura que las instrucciones se ejecutan secuencialmente siguiendo el mismo orden en que aparecen en el programa.
- Selección: se le conoce también como la estructura IF-TRUE-FALSE, supone la selección entre dos opciones según el resultado obtenido a partir de la evaluación de un predicado. Equivale a la instrucción IF de los lenguajes de programación.
- Iteración: conocida como la estructura DO-WHILE, consiste en la ejecución repetida de una instrucción mientras una condición previamente establecida continúe siendo válida.

1.3.3 Programación Funcional

La programación en un lenguaje funcional consiste en construir definiciones y en usar el computador para evaluar expresiones. El papel principal del programador es la definición de una función que permita resolver un problema dado. Esta función, que puede involucrar un cierto número de funciones auxiliares, se describe en una notación fundada en principios matemáticos comunes. (Bird, 2000)

Los programas escritos usando el paradigma funcional puro están formados solamente por definiciones de funciones, en ellos no existen las asignaciones de variables y las construcciones estructuradas como la secuencia o la iteración, por lo que para llevar a cabo operaciones repetitivas o bucles son necesarias las funciones recursivas.
Los lenguajes que usan este paradigma ofrecen al programador recursos expresivos que permiten solucionar problemas complejos a través de programas pequeños y robustos. Ofrecen la posibilidad de definir funciones que acepten funciones como argumento y devuelvan funciones como resultado y brindan facilidades para la manipulación de estructuras de datos infinitas.

1.3.4 Programación Modular

La programación modular es una técnica de programación, basada en la estrategia ¨divide y vencerás¨, que permite dividir la complejidad de un problema convirtiendo operaciones complejas en un subconjunto de problemas más simples y fáciles de implementar mediante la identificación de sub-problemas o tareas, este enfoque es denominado ¨diseño descendente por refinamientos sucesivos¨ y su consecuencia directa es la modularidad del código fuente resultante. Este paradigma permite la reutilización de código de un programa en cualquier momento de la ejecución del mismo.

Los módulos son comúnmente representados a través de procedimientos y funciones, poseen actividades bien definidas y se comunican entre sí a través de la implementación de interfaces.

Este tipo de programación es considerada como una forma evolucionada de la programación estructurada debido a su capacidad de solucionar problemas de mayor tamaño y complejidad auxiliándose de su estrategia de fragmentación.

1.3.5 Programación Orientada a Objetos

La Programación Orientada a Objetos (OOP según sus siglas en inglés) es un paradigma de programación que define los programas en términos de “clases de objetos”, objetos que son entidades que combinan estado (propiedades o datos), comportamiento (procedimientos o métodos) e identidad (propiedad del objeto que lo diferencia del resto). (Weitzenfeld, 2004)

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

Un objeto contiene toda la información que permite definirlo e identificarlo frente a otros objetos pertenecientes a otras clases e incluso entre objetos de una misma clase, al poder tener valores bien diferenciados en sus atributos. A su vez, dispone de mecanismos de interacción (los métodos) que favorecen la comunicación entre objetos de una misma clase o de distintas, y en consecuencia, el cambio de estado en los propios objetos. Esta característica lleva a tratarlos como unidades indivisibles, en las que no se separan, ni deben separarse, información (datos) y procesamiento (métodos). (Winblad, 1993)

Las clases son declaraciones o abstracciones de objetos, lo que significa, que una clase es la definición de un objeto. Cuando se programa un objeto y se definen sus características y funcionalidades, realmente se programa una clase.
A continuación se abordarán elementos de la POO.

- Herencia: es una característica específica de la programación orientada a objetos, donde una clase nueva se crea a partir de una clase existente. La herencia (a la que habitualmente se denomina subclases) proviene del hecho de que la subclase (la nueva clase creada) contiene los atributos y métodos de la clase primaria. La principal ventaja de la herencia es la capacidad para definir atributos y métodos nuevos para la subclase, que luego se aplican a los atributos y métodos heredados.
- Encapsulamiento: es el mecanismo básico de la programación orientada a objetos para ocultar los detalles internos del objeto de los demás objetos. El encapsulamiento permite distinguir entre la interface del objeto, o sea, los aspectos del objeto conocidos externamente, y su implementación, o sea, sus aspectos conocidos internamente. La interface de un objeto corresponde a las interfaces de sus funciones, mientras que la implementación corresponde a los datos y código, o implementación de sus funciones. (Weitzenfeld, 2004)
- Polimorfismo: es posiblemente el concepto más poderoso de la programación orientada a objetos, aunque también el más complicado. Mediante el polimorfismo se definen múltiples funciones con

> nombres e interfaces similares en distintas clases. Las funciones son implementadas de manera diferente en las clases. El polimorfismo es útil para extender la funcionalidad del sistema al definir nuevas clases aún desconocidas al momento de especificarlo. La idea general es definir un estándar de interfaces que deben seguir todas las clases, las existentes y las nuevas. (Weitzenfeld, 2004)

Por las características anteriormente expuestas y teniendo en cuenta las ventajas de seguridad y la fortaleza en los planteamientos lógicos en los que se basa y las facilidades que ofrece el trabajo con objetos es el paradigma seleccionado para el proceso de desarrollo.

1.4 Lenguaje de Programación

Un lenguaje de programación es aquel elemento dentro de la informática que permite crear programas mediante un conjunto de instrucciones, operadores y reglas de sintaxis; que pone a disposición del programador para que este pueda comunicarse con los dispositivos hardware y software existentes. (Definición.org).

El lenguaje de programación que seleccionó el equipo de arquitectura del proyecto VIRTEVALL para la construcción del software fue PHP5 5.2.5.

1.4.1 PHP5 5.2.5

PHP es un lenguaje interpretado de propósito general ampliamente usado y que está diseñado especialmente para desarrollo web y puede ser incrustado dentro de código HTML. Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida. Puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno. PHP se encuentra instalado en más de 20 millones de sitios web y en un millón de servidores, el número de sitios en PHP ha compartido algo de su preponderante sitio con otros nuevos lenguajes no tan poderosos desde agosto de 2005. (PHP:Hypertext)

El 13 de julio de 2004, fue sacado a la luz pública PHP 5, utilizando el motor Zend Engine II. En esta versión incorpora Programación Orientada a Objetos, lo que le convierte en un lenguaje aún más versátil.

Es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación. PHP, en el caso de estar montado sobre un servidor Linux o Unix, es más rápido que ASP, dado que se ejecuta en un único espacio de memoria y esto evita las comunicaciones entre componentes COM que se realizan entre todas las tecnologías implicadas en una página ASP. (PHP:Hypertext)

Este lenguaje de programación presenta disímiles ventajas: (PHP:Hypertext)

- Es un lenguaje multiplataforma.
- Completamente orientado a la web.
- Capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL y PostgreSQL.
- Capacidad de expandir su potencial utilizando la enorme cantidad de módulos (llamados ext's o extensiones).
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Permite las técnicas de Programación Orientada a Objetos.
- Biblioteca nativa de funciones sumamente amplia e incluida.
- No requiere definición de tipos de variables aunque sus variables se pueden evaluar también por el tipo que estén manejando en tiempo de ejecución.
- Tiene manejo de excepciones (desde PHP5).
- Rapidez. PHP generalmente es utilizado como módulo de Apache, lo que lo hace extremadamente veloz. Está completamente escrito en C, así que se ejecuta rápidamente utilizando poca memoria.

Desventajas que presenta php.

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

1.5 Entorno de Desarrollo Integrado

Un entorno de desarrollo integrado o, Integrated Development Environment (IDE), es un programa compuesto por un conjunto de herramientas para un programador que puede dedicarse en exclusiva a un sólo lenguaje de programación o bien, poder utilizarse para varios.

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. (SlideShare)

El entorno de desarrollo integrado que fue seleccionado por el equipo de arquitectura para desarrollar el software mediante el lenguaje de programación PHP 5.2.5 es NetBeans 6.8 M2.

1.5.1 Netbeans

NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 1000 socios en todo el mundo.
NetBeans comenzó como un proyecto estudiantil en República Checa (originalmente llamado Xelfi), en 1996 bajo la tutoría de la Facultad de Matemáticas y Física en la Universidad Carolina en Praga. La meta era escribir un entorno de desarrollo integrado (IDE) para Java parecida a la de Delphi. Xelfi fue el primer entorno de desarrollo integrado escrito en Java, con su primer pre-release en 1997. El proyecto atrajo suficiente interés, por lo que los estudiantes, después de graduarse, decidieron que lo podían convertir en un proyecto comercial. Prestando espacios web de amigos y familiares, formaron una compañía alrededor de esto. (netbeans.org)

1.5.2 Netbeans IDE 6.8 M2

El IDE Netbeans 6.8 M2 es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans. Es un producto de código abierto, con todos los beneficios del software disponible en forma gratuita, el cual ha sido examinado por una comunidad de desarrolladores.

Aparte de la filosofía de distribución y desarrollo que respalda a NetBeans, el IDE ofrece a los desarrolladores numerosas ventajas, en la creación de nuevas aplicaciones multiplataforma. En una era en la cual la arquitectura orientada al servicio (SOA) requiere servicios con cierta relación que manejen procesos específicos del negocio, NetBeans 6.8 M2 satisface los requisitos con conjuntos de herramientas independientes de la plataforma, modulares y orientadas al objeto. (IDE)

1.6 Framework

La palabra inglesa framework define, en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar.

En el desarrollo de software, un framework es una estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, en base a la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto. (Gutiérrez).

Representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

El framework que seleccionó el equipo de arquitectura del proyecto VIRTEVALL para la construcción del software fue Symfony 1.2.8.

1.6.1 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. (Fabien Potencier, 2008)

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 *nix (Unix, Linux, etc.) como en plataformas Windows. (symfony.es)

Symfony se diseñó para que se ajustara a los siguientes requisitos: (symfony.es)

- Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares).
- Independiente del sistema gestor de bases de datos. 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 los 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 base de datos, el controlador frontal y las acciones.
- Basado en la premisa de “convenir en vez de configurar”, en la que el desarrollador sólo debe configurar aquello que no es convencional.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Preparado para aplicaciones empresariales y adaptables a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- 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 ahorrando tiempo de trabajo.

1.6.2 Características de Symfony 1.2.8:

- **sfAction, Capa de Presentación**: Toma lo mejor de la arquitectura MVC y la implementa de forma que el desarrollo de aplicaciones sea rápido y sencillo, en el siguiente gráfico se puede apreciar como es el flujo de trabajo.

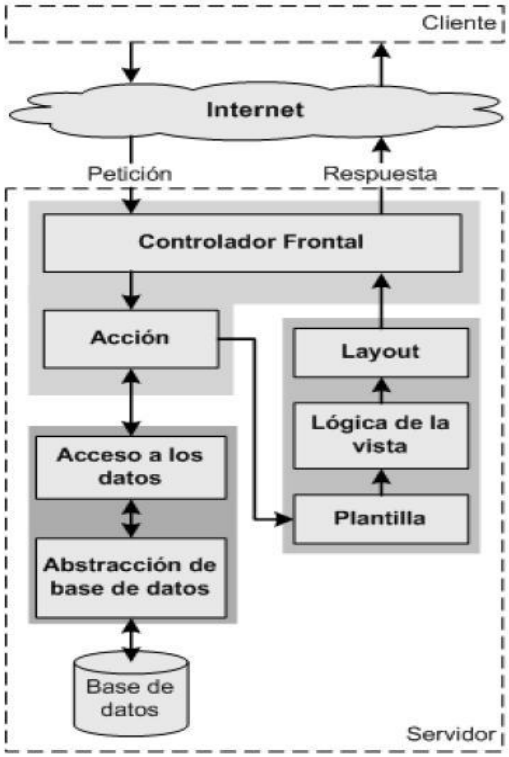


Figura 2: Flujo de Trabajo de Symfony

- **PROPEL, Acceso a Datos**: La lógica de negocio de las aplicaciones web depende casi siempre en su modelo de datos. El componente que se encarga por defecto de gestionar el modelo en Symfony es una capa de tipo ORM realizada mediante el proyecto Propel. En las aplicaciones Symfony, el acceso y la modificación de los datos almacenados en la base de datos se realiza mediante objetos; de esta forma nunca se accede de forma explícita a la base de datos. Este comportamiento permite un alto nivel de abstracción y permite una fácil portabilidad.

- **YALM, Configuración del Sistema**: YAML ("YAML Ain't Markup Language") es un lenguaje muy sencillo que permite describir los datos como en XML, pero con una sintaxis mucho más sencilla. YAML es un formato especialmente útil para describir datos que pueden ser transformados en

arrays simples y asociativos. Symfony utiliza el formato YAML como el lenguaje preferido para almacenar su configuración.

1.7 Sistemas Gestores de Base 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 base de datos, el usuario y las aplicaciones que la utilizan. La función general de un sistema gestor de base de datos es el de manejar de manera clara, sencilla y ordenada un conjunto de datos que posteriormente se convertirán en información relevante para una organización.

Existen distintos objetivos que deben cumplir los SGBD:

- **Abstracción de la información**: Los SGBD ahorran a los usuarios detalles acerca del almacenamiento físico de los datos. Da lo mismo si una base de datos ocupa uno o cientos de archivos, este hecho se hace transparente al usuario. Así, se definen varios niveles de abstracción.

- **Independencia:** La independencia de los datos consiste en la capacidad de modificar el esquema (físico o lógico) de una base de datos sin tener que realizar cambios en las aplicaciones que se sirven de ella.

- **Consistencia**: En aquellos casos en los que no se ha logrado eliminar la redundancia, será necesario vigilar que aquella información que aparece repetida se actualice de forma coherente, es decir, que todos los datos repetidos se actualicen de forma simultánea. Por otra parte, la base de datos representa una realidad determinada que tiene determinadas condiciones, por ejemplo que los menores de edad no pueden tener licencia de conducir. El sistema no debería aceptar datos de un conductor menor de edad. En los SGBD existen herramientas que facilitan la programación de este tipo de condiciones.

- **Seguridad:** La información almacenada en una base de datos puede llegar a tener un gran valor. Los SGBD deben garantizar que esta información se encuentre segura con la creación de grupos de usuarios que permitan otorgar diversas categorías de permisos.

- **Manejo de transacciones:** Una transacción es un programa que se ejecuta como una única operación. Esto quiere decir que luego de una ejecución en la que se produce una falla el resultado es el mismo que se obtendría si el programa no se hubiera ejecutado. Los SGBD proveen mecanismos para programar las modificaciones de los datos de una forma mucho más simple que si no se dispusiera de ellos.

- **Tiempo de respuesta**: Lógicamente, es deseable minimizar el tiempo que el SGBD tarda en dar la información solicitada y en almacenar los cambios realizados.

Existen varios gestores de bases de datos libres, entre ellos se encuentra el PostgreSQL 8.3.4.1 siendo elegido por la dirección del proyecto VIRTEVAL para llevar a cabo todo el trabajo y funcionamiento de la base de datos.

1.7.1 PostgreSQL 8.3.4.1

Es un servidor de base de datos relacional, libre. Tiene soporte total para transacciones, disparadores, vistas, procedimientos almacenados, almacenamiento de objetos de gran tamaño. Se destaca en ejecutar consultas complejas, sobre vistas, subconsultas y joins de gran tamaño. Permite la definición de tipos de datos personalizados e incluye un modelo de seguridad completo. PostgreSQL tiene una gran comunidad de desarrollo en Internet, su código fuente está disponible sin costo alguno y algo muy importante, dicha herramienta es multiplataforma. (postgresql.org)

Fue diseñado para ambientes de alto volumen. Escala muy bien al aumentar el número de CPUs y la cantidad de RAM. Soporta transacciones desde la versión 7.0, claves ajenas con comprobaciones de integridad referencial. Tiene mejor soporte para vistas y procedimientos almacenados en el servidor, además tiene ciertas características orientadas a objetos.

Algunas de sus múltiples características: (postgresql.org)

- **Alta concurrencia**: Mediante un sistema denominado MVCC (Acceso concurrente multiversión, 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, eliminando la necesidad del uso de bloqueos explícitos.

- **Atomicidad (Indivisible):** Propiedad que asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.

- **Consistencia:** Propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper la reglas y directrices de integridad de la base de datos.

- **Aislamiento**: Propiedad que asegura que una operación no puede afectar a otras.

- **Durabilidad**: Propiedad que asegura que una vez realizada la operación, persistirá y no se podrá deshacer aunque falle el sistema.

- Corre en casi todos los principales sistemas operativos: Linux, Unix, BSDs, Mac OS, Beos, Windows, etc.
- Documentación muy bien organizada, pública y libre, con comentarios de los propios usuarios.
- Comunidades muy activas, varias de ellas en castellano.
- Bajo “Costo de Propiedad Total” (TCO) y rápido “Retorno de la Inversión Inicial” (ROI).

Ofrece una potencia sustancial al incorporar los siguientes cuatro conceptos básicos: (clases, herencia, tipos, funciones) para que los usuarios puedan extender formalmente el sistema. Tiene mejor soporte para triggers y procedimientos en el servidor.

1.8 Librería utilizada: ExtJS 2.2

ExtJS es una librería Javascript que permite construir aplicaciones complejas en Internet, incluye:

- Componentes de interfaz de usuario de alto performance y personalizables.
- Modelo de componentes extensibles.
- Un API fácil de usar.
- Licencias Open source y comerciales.

1.8.1 ¿Qué tiene de bueno ExtJS 2.2?

Antes de poder entrar a examinar ExtJS es necesario hablar sobre RIA, acrónimo de Rich Internet Applications (Aplicaciones Ricas en Internet). Lo que RIA intenta proveer es aquello de lo que siempre ha adolecido la web, una experiencia de usuario muy parecida o igual a la que se tiene en las aplicaciones de escritorio.

Las aplicaciones web tradicionales tienen problemas en la recarga continua de las páginas cada vez que el usuario pide nuevo contenido, o la poca capacidad multimedia, para lo cual se han hecho necesarios plug-ins externos.

Junto con el reto de llevar la experiencia RIA a los usuarios comenzó el debate sobre cuál sería el mejor modo de atacar el problema. La historia de los últimos años ha traído diversas tecnologías, basadas en Flash (Adobe), Java (Sun), Silverlight (MS). Todas muy interesantes, pero con la desventaja de necesitar algún tipo de extensión en los navegadores que podría no estar presente. Ha sido esta limitante lo que le ha dado la victoria (al menos por el momento) al casi dejado de lado Javascript y la “nueva” tecnología conocida como AJAX (ExtJS).

ExtJS encaja dentro de este esquema como un motor que permite crear aplicaciones RIA mediante Javascript. Si se enmarca a ExtJS dentro del desarrollo RIA, éste sería el render de la aplicación que controla el cliente y se encarga de enviar y obtener información del servicio.

Una de las grandes ventajas de utilizar ExtJS es que permite crear aplicaciones complejas utilizando componentes predefinidos así como un manejador de layouts similar al que provee Java Swing, gracias a

esto provee una experiencia consistente sobre cualquier navegador, evitando el tedioso problema de validar que el código escrito funcione bien en cada uno (Firefox, IE, Safari, etc.). (ExtJS)

Además la ventana flotante que provee ExtJS es excelente por la forma en la que funciona. Al moverla o redimensionarla sólo se dibujan los bordes haciendo que el movimiento sea fluido lo cual representa una tremenda ventaja.

Usar un motor de render permite tener además estos beneficios: (ExtJS)

- **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.

ExtJS es muy flexible y además permite realizar de una manera muy rápida interfaces muy profesional, la comunidad que está detrás de esta herramienta es muy grande y la documentación cada vez es más extensa, además cuenta con varias licencias que se pueden utilizar de acuerdo al proyecto que se esté desarrollando, pueden usar la licencia “Open Source” o bien comprar una licencia “comercial” (ExtJS)

1.8.2 Desventajas que presenta ExtJS 2.2:

En el caso de ExtJS existe un factor que hay que tener en cuenta y que nace del hecho de ser una librería Javascript independiente de cualquier tecnología del lado servidor. No existe una forma fácil de realizar binding entre los componentes visuales con el respectivo modelo, lo cual genera que el programador tenga que escribir más código para validar y enlazar los formularios. (ExtJS)

Es difícil hacer que el servidor haga push de información desde el servidor web hacia el navegador, haciendo que el cliente tenga que constantemente hacer pooling para obtener datos actualizados del servidor.

La falta de un diseñador gráfico limita la difusión de la aplicación al no proveer una forma fácil y rápida de desarrollar en él.

1.9 Conclusiones

En este capítulo se abordó el tema referente a la metodología a utilizar, lenguaje de programación web, haciendo énfasis en las características y ventajas de cada uno de estos. Se seleccionó el lenguaje de programación web PHP 5 del lado del servidor por las disímiles ventajas que este brinda. Mientras que para el lado del cliente será empleado el Javascript. Además se escogió PostgreSQL 8.3.4.1 como Sistema Gestor de Base de Datos, Symfony 1.2.8 como framework para el desarrollo de la aplicación y NetBeans 6.8 como IDE para PHP.

CAPÍTULO 2: DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA

2.1Introducción

El modelo de implementación es el principal artefacto que se genera en el flujo de trabajo Implementación, según RUP, describe la forma en que los elementos del modelo de diseño, se implementan en términos de componentes, ficheros de código fuente, ejecutables etc. El modelo de implementación describe también como se organizan los componentes de acuerdo con los mecanismos de estructuración en el entorno de implementación y del lenguaje de programación utilizado y a demás describe la dependencia de los componentes unos con otros.

En este capítulo se realiza un análisis de la complejidad y funcionamiento de los algoritmos más importantes para la implementación del módulo, así como las estructuras de datos utilizadas para manejar los datos en el programa. Se presenta la descripción de las clases fundamentales que definen el comportamiento del sistema con sus atributos y métodos esenciales. Finalmente se definen los estándares de codificación a utilizar por el programador.

2.2 Diseño propuesto por el analista.

A partir del diseño propuesto por los analistas del proyecto VIRTEVALL para el módulo de diagnóstico, se pueden identificar las funcionalidades a desarrollar para que el mismo funcione correctamente, tomando como guía la descripción de los casos de uso. Se puede trazar una estrategia que permita una correcta implementación del módulo. Se designaron disímiles herramientas a usar para la implementación, las mismas fueron escogidas por las facilidades y ventajas que brindan. Entre las herramientas a usar están: gestor de base de datos PostgreSQL, este gestor es totalmente libre y además es muy factible para la realización de consultas a la base de datos, el framework para desarrollar el módulo será Symfony el cual hace uso del patrón de arquitectura Modelo Vista Controlador (MVC), permitiendo así una mejor organización de todas las clases e interfaces a implementar. Además de las herramientas mencionadas serán utilizas otras que permitirán hacer más fácil la implementación del módulo de diagnóstico del proyecto VIRTEVALL.

A continuación se muestran los diagramas de clases de diseño propuestos por los analistas. Los mismos están acorde con los requerimientos del sistema a implementar y con el patrón de arquitectura MVC implementado por el framework Symfony.

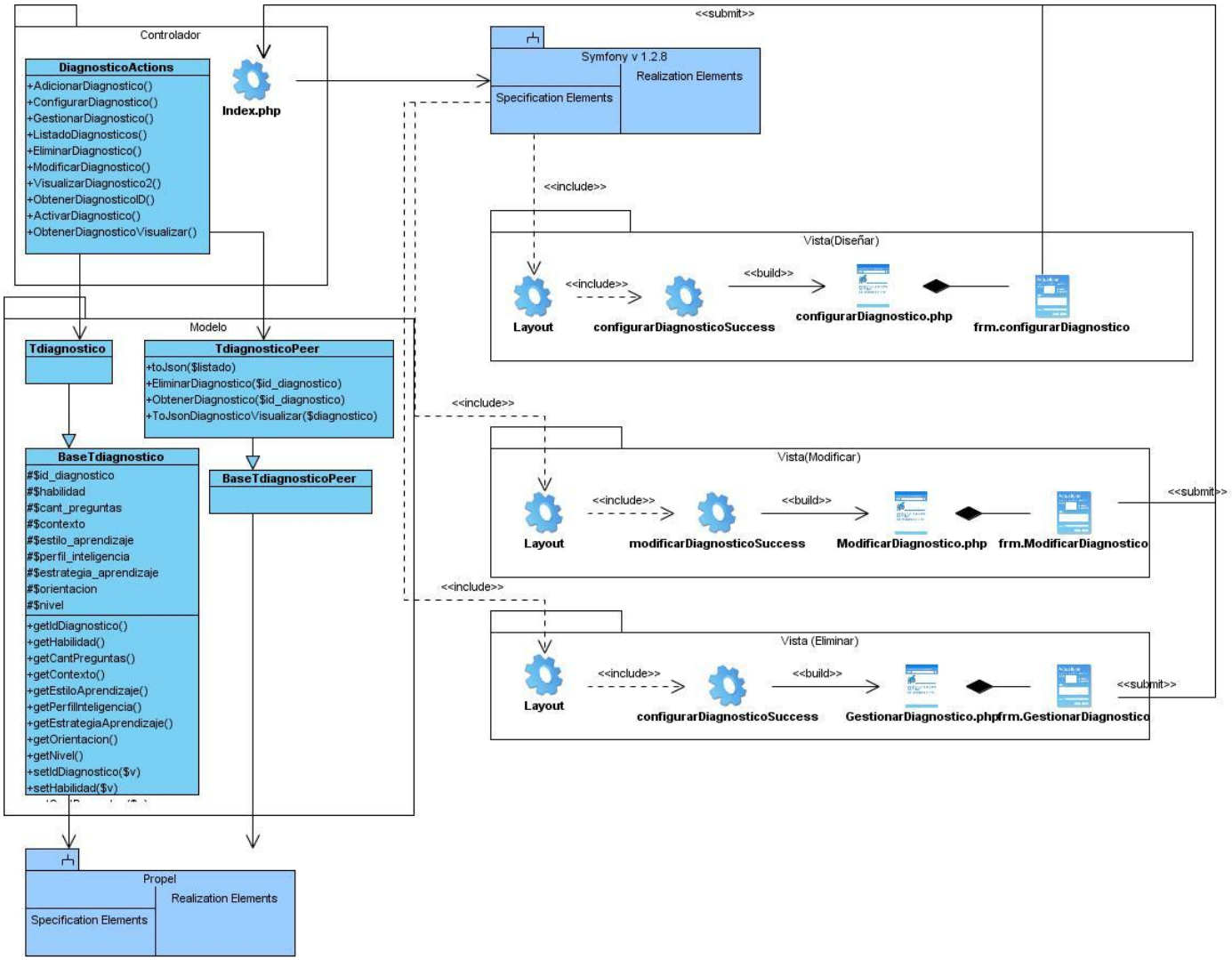


Figura 3: Diagrama de Clases de Diseño del CU Gestionar Diagnóstico

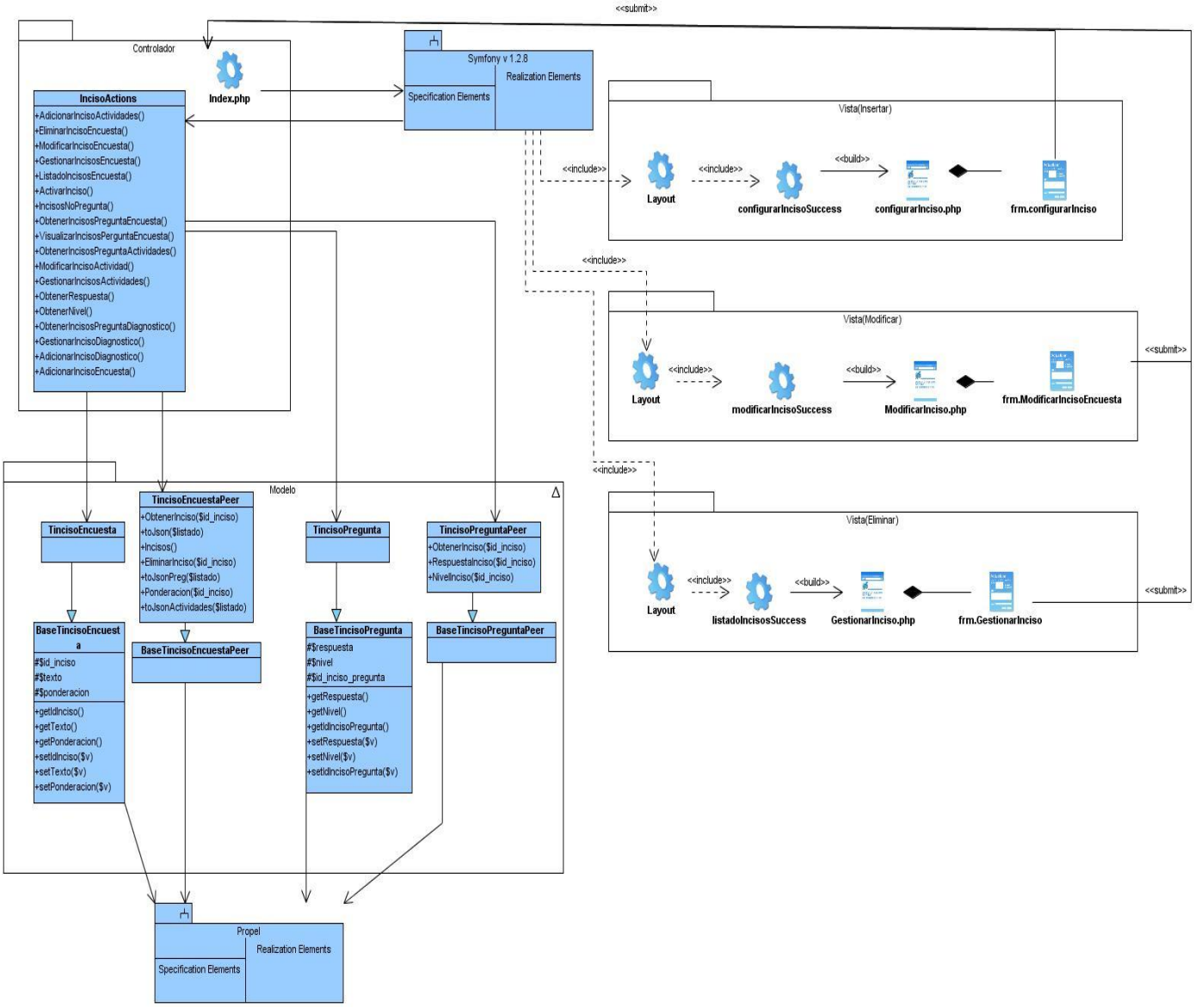


Figura 4: Diagrama de Clases de Diseño del CU Gestionar inciso

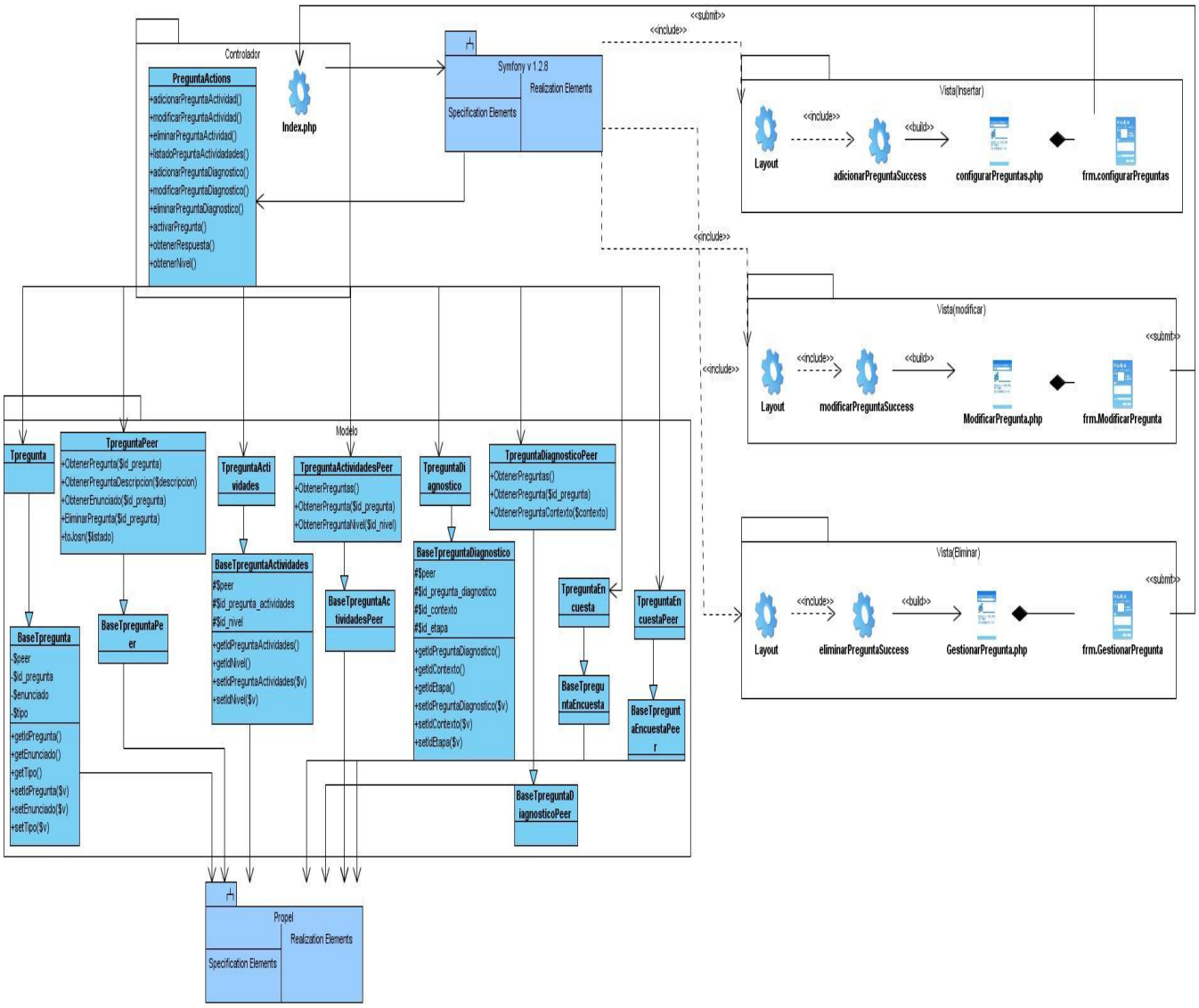


Figura 5: Diagrama de Clases de Diseño del CU Gestionar Pregunta

2.3 Patrones de software utilizados.

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

- **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.

- **Experto:** La implementación que realiza Symfony de la arquitectura MVC incluye varias clases, en las cuales se evidencia el uso de este patrón:

    - SfController: Clase perteneciente al controlador que se encarga de decodificar la petición y transferirla a la acción correspondiente.

    - SfRequest: Almacena los elementos que forman la petición (parámetros, cookies, cabeceras, etc.).

    - SfResponse: Contiene las cabeceras de la respuesta y los contenidos. El contenido del objeto se transforma en la respuesta HTML que se envía al usuario.

- **Alta Cohesión:** La clase nombreActions define las acciones para las plantillas y colabora con otras para realizar diferentes operaciones. Esta clase está formada por diferentes funcionalidades estrechamente relacionadas proporcionándole flexibilidad al software ante grandes cambios.

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

- **Controlador:** Las peticiones Web son manejadas por el controlador frontal, el cual es el único punto de entrada de toda la aplicación en un determinado entorno. El controlador frontal, al recibir una petición, utiliza el sistema de enrutamiento para enviar la acción al controlador encargado de darle solución.

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 el controlador frontal hay una llamada a sfContext::getInstance(). En una acción, el método getContext(), un objeto muy útil que guarda una referencia a todos los objetos del núcleo de Symfony.

- **Abstract Factory** (Fábrica abstracta): Permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando. Cuando el framework necesita por ejemplo crear un nuevo objeto para una petición, busca en la definición de la factoría el nombre de la clase que se debe utilizar para esta tarea.

En la categoría **Estructurales**:

- **Decorator** (Envoltorio): Añade funcionalidad a una clase, dinámicamente. El archivo layont.php, que también se denomina plantilla global, almacena el código HTML que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en el layout, o si se mira desde el otro punto de vista, el layout decora la plantilla.

- **Composite** (Objeto compuesto): Permite tratar objetos compuestos como si de un simple se tratase. Sirve para construir objetos complejos a partir de otros más simples y similares entre sí, gracias a la composición recursiva y a una estructura en forma de árbol. Esto simplifica el tratamiento de los objetos creados, ya que al poseer todos ellos una interfaz común, se tratan todos de la misma manera.

2.4 Representación gráfica del diagrama de componentes.

Un diagrama de componentes representa un sistema de software dividido en componentes y las dependencias entre estos. Los componentes físicos incluyen archivos, cabeceras, librerías compartidas, módulos, ejecutables, o paquetes. Los diagramas de componentes prevalecen en el campo de la arquitectura de software pero pueden ser usados para modelar y documentar cualquier arquitectura de sistema. (Hommel, 1999)

Debido a que estos son más parecidos a los diagramas de casos de usos, son utilizados para modelar la vista estática de un sistema, que representa la organización y dependencia que habría entre los componentes físicos que se necesitan para ejecutar la aplicación. No es necesario que un diagrama incluya todos los componentes del sistema, normalmente se realizan por partes. (Kruchten, 200)
A continuación se representa el diagrama de componentes referente al módulo diagnóstico.

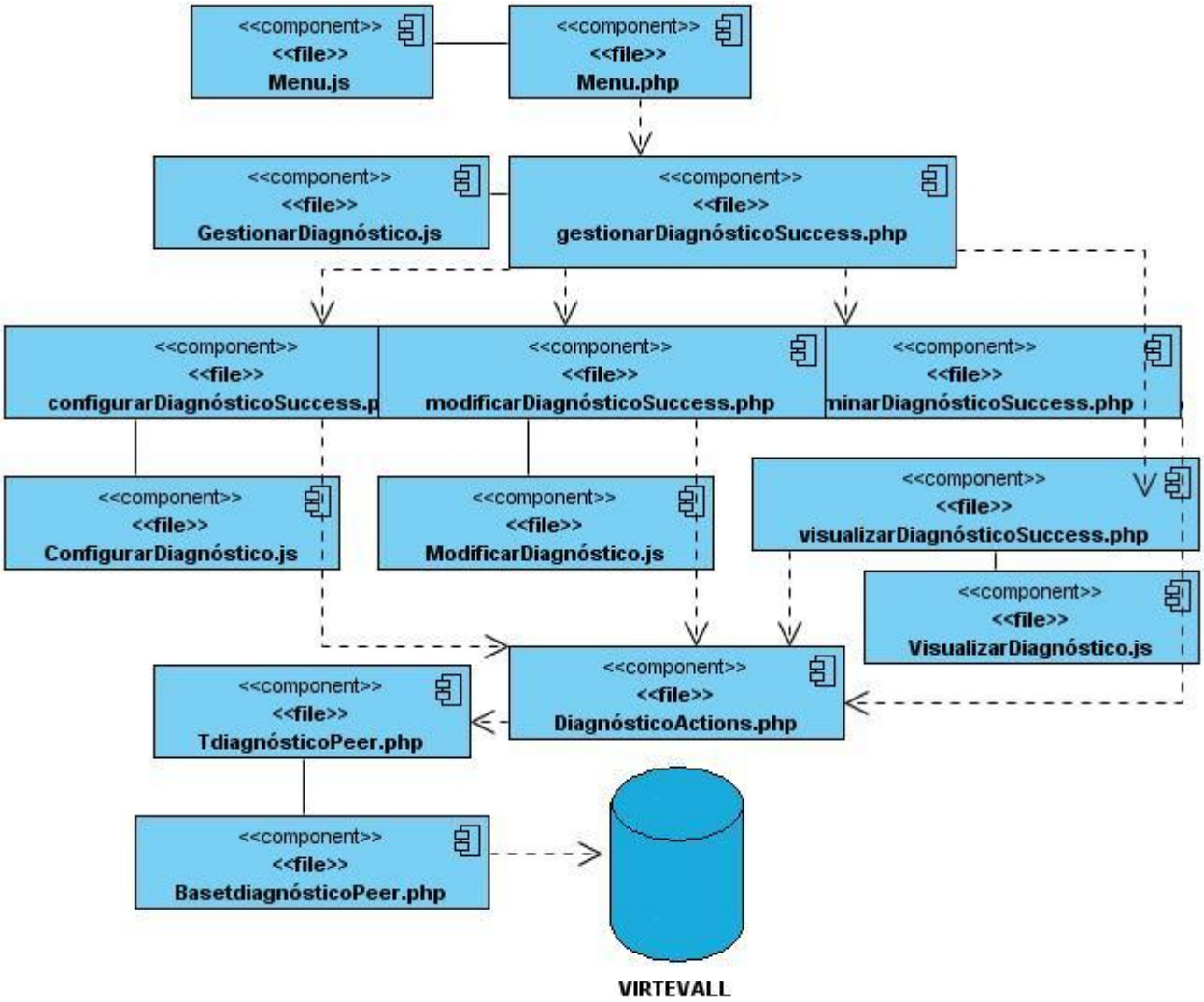


Figura 6: Diagrama de componentes del CU Gestionar Diagnóstico

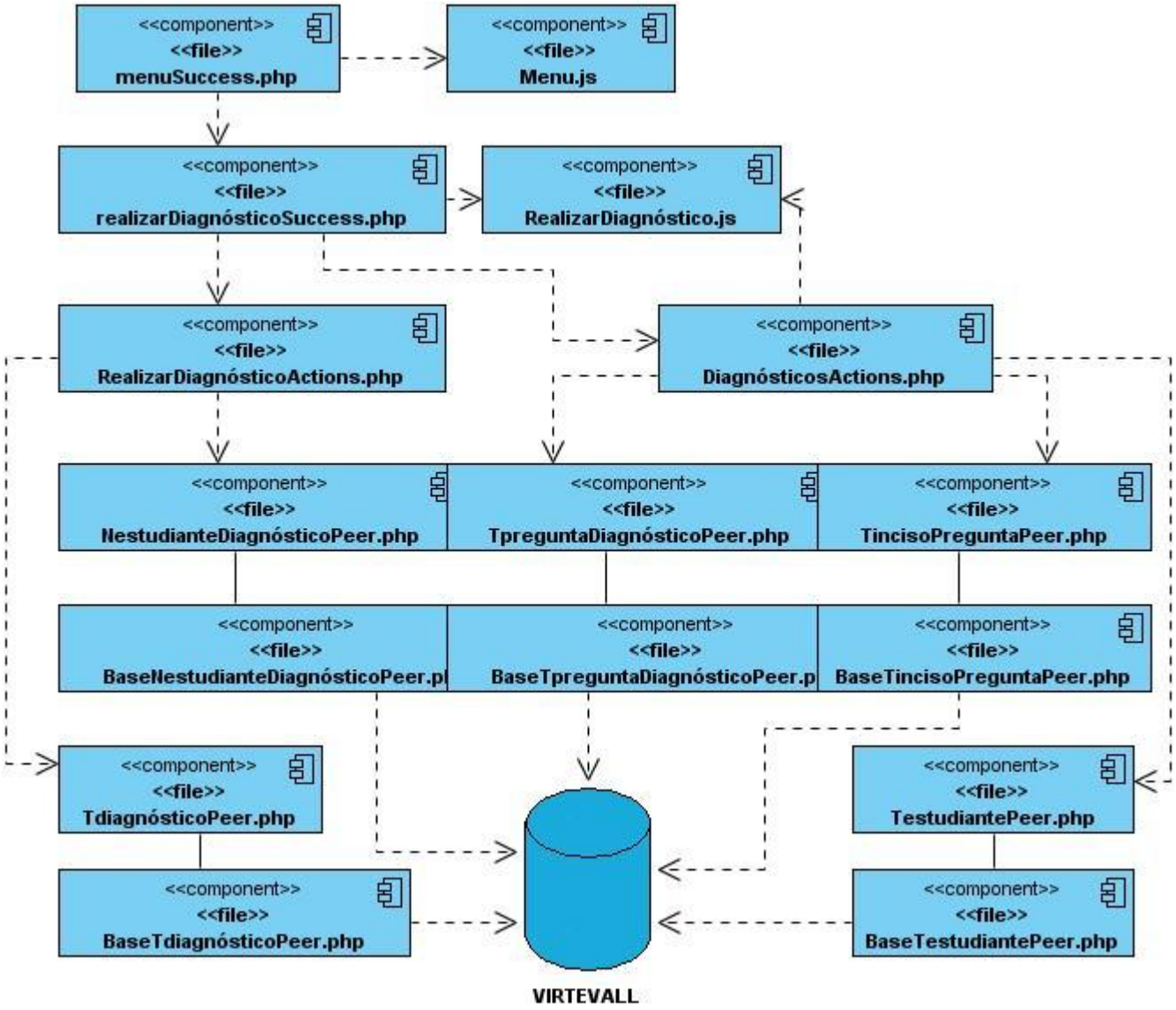


Figura 7: Diagrama de componente del CU Realizar Diagnóstico

2.5 Descripción de los algoritmos más importantes a implementar.

En esta parte se realizará un análisis de los algoritmos que son importantes para el correcto desarrollo del módulo o aquellos que su codificación resulte compleja. Se reflejará el código fuente de los mismos y a su vez se hará una descripción de su funcionamiento.

2.5.1 Adicionar diagnósticos.

Dicho algoritmo está contenido en la función executeAdicionarDiagnostico de la clase DiagnosticoActions. Permite adicionar diagnósticos que serán aplicados a los estudiantes.

```
class DiagnosticoActions extends sfActions
{
      ///***Gestionar Diagnostico***\\\
    public function executeAdicionarDiagnostico()
    {
        if ($this->getRequest()->getMethod () != sfRequest::POST){
            return sfView::SUCCESS;
        }else {
            $diagnostico = new Tdiagnostico();
            $diagnostico->setHabilidad($this->getRequest()->getParameter('habilidad'));
            $diagnostico->setCantPreguntas($this->getRequest()->getParameter('cant_preg'));
            $diagnostico->setContexto($this->getRequest()->getParameter('contexto'));
            $diagnostico->setEstiloAprendizaje($this->getRequest()->getParameter('estilo_aprendizaje'));
            $diagnostico->setPerfilInteligencia($this->getRequest()->getParameter('perfil_aprendizaje'));
            $diagnostico->setEstrategiaAprendizaje($this->getRequest()->getParameter('estrategia_aprendizaje'));
            $diagnostico->setOrientacion($this->getRequest()->getParameter('orientacion'));
            $diagnostico->setNivel($this->getRequest()->getParameter('nivel'));
            $diagnostico->save();
```

```
$TdiagnosticoTipo = new NdiagnosticoTtipoPreguntaTetapa();

$lista = $this->getRequest()->getParameter('lista');
$selecc = split("&", $lista);
$list = "";
$aux = "";
$j = 0;
for ($i = 8; $i < count($selecc); $i++){
    $aux = split("=on", $selecc[$i]);
    $list[$j] = $aux[0];
    $j++;
}
for($i = 0; $i < count($list); $i ++){
    $TdiagnosticoTipo = new NdiagnosticoTtipoPreguntaTetapa();
    $TdiagnosticoTipo->setIdDiagnostico($diagnostico->getIdDiagnostico());
    $TdiagnosticoTipo->setIdTipoPregunta($list[$i]) ;
    $TdiagnosticoTipo->save();
}
$id_diagnostico = $diagnostico->getIdDiagnostico();
```

Pasos del algoritmo adicionar diagnóstico de la función executeAdicionarDiagnostico:

1. Se comprueba al ejecutar la función executeAdicionarDiagnostico si la información procedente de la vista está siendo enviada por el método Post.
2. En caso de no cumplirse lo anterior se mostrará la vista con los campos del formulario.
3. Si la información que se obtiene fue enviada por el método Post entonces se crea un nuevo diagnóstico procesando la misma.
4. Se crea un objeto de tipo la clase NdiagnosticoTtipoPreguntaTetapa.
5. Se procede a adicionar los tipos de pregunta que va a contener este diagnóstico.
6. Se procede a insertar la información en la base de datos.

2.5.2 Eliminar diagnósticos.

Dicho algoritmo está contenido en la función executeEliminarDiagnostico de la clase DiagnosticoActions. Permite eliminar los diagnósticos.

```
public function executeEliminarDiagnostico()
{
    $diagnostico = TdiagnosticoPeer::ObtenerDiagnostico($this->getRequest ()->getParameter ('id_diagnostico'));
    if ($diagnostico != null)
    {
        TdiagnosticoPeer::EliminarDiagnostico($this->getRequest ()->getParameter ('id_diagnostico'));
        die();
    }

}
```

Pasos del algoritmo eliminar diagnóstico de la función executeEliminarDiagnostico:

1. Se realiza una búsqueda en la base de datos con el objetivo de comprobar que el *id* del diagnóstico que se desea eliminar es válido.
2. En caso de cumplirse lo anterior se procede a eliminar el diagnóstico seleccionado.

2.5.3 Modificar diagnósticos.

Dicho algoritmo está contenido en la función executeModificarDiagnostico de la clase DiagnosticoActions. Permite modificar los diagnósticos.

```
public function executeModificarDiagnostico()
{
    if ($this->getRequest ()->getMethod () != sfRequest::POST){
        return sfView::SUCCESS;
    }
    else{
            $diagnostico = TdiagnosticoPeer::ObtenerDiagnostico($this->getUser()->getAttribute('id_diag_act'));
            if ($diagnostico != null){

                $diagnostico->setHabilidad($this->getRequest()->getParameter('habilidad'));
                $diagnostico->setCantPreguntas($this->getRequest()->getParameter('cant_preg'));
                $diagnostico->setContexto($this->getRequest()->getParameter('contexto'));
                $diagnostico->setEstiloAprendizaje($this->getRequest()->getParameter('estilo_aprendizaje'));
                $diagnostico->setPerfilInteligencia($this->getRequest()->getParameter('perfil_aprendizaje'));
                $diagnostico->setEstrategiaAprendizaje($this->getRequest()->getParameter('estrategia_aprendizaje'));
                $diagnostico->setOrientacion($this->getRequest()->getParameter('orientacion'));
                $diagnostico->setNivel($this->getRequest()->getParameter('nivel'));
                $diagnostico->save();

                $lista = $this->getRequest()->getParameter('lista');
                $selecc = split("&", $lista);
                $list = "";
                $aux = "";
                $j = 0;
                for ($i = 8; $i < count($selecc); $i++){
                    $aux = split("=on", $selecc[$i]);
    $TdiagnosticoTipo = NdiagnosticoTtipoPreguntaTetapaPeer::ObtenerDiagnosticoID($diagnostico->getIdDiagnostico());
    for ($h = 0; $h < count($TdiagnosticoTipo); $h++){
        NdiagnosticoTtipoPreguntaTetapaPeer::Eliminar($TdiagnosticoTipo[$h]->getIdDiagnostico());
    }
    for($i = 0; $i < count($list); $i ++){
        $TdiagnosticoTipo = new NdiagnosticoTtipoPreguntaTetapa();
        $TdiagnosticoTipo->setIdDiagnostico($diagnostico->getIdDiagnostico());
        $TdiagnosticoTipo->setIdTipoPregunta($list[$i]) ;
        $TdiagnosticoTipo->save();
    }

}
else{
    $mensaje = "";
    if ($this->getRequest ()->getMethod () == sfRequest::POST){
            $mensaje = "Ese diagnostico no existe";
    }
    $this->mensaje = $mensaje;
}
```

Pasos del algoritmo modificar diagnóstico de la función executeEliminarDiagnostico:

1. Se comprueba al ejecutar la función executeAdicionarDiagnostico si la información procedente de la vista está siendo enviada por el método Post.
2. En caso de no cumplirse lo anterior se mostrará la vista con los campos del formulario.
3. Se realiza una búsqueda en la base de datos con el objetivo de comprobar que el *id* del diagnóstico que se desea modificar es válido.
4. Se procede a modificar la información en la base de datos y se redirecciona a la página donde se muestra un listado actualizado de todos los diagnósticos existentes.
5. Si el *id* seleccionado no es válido, se muestra un mensaje al usuario informándole la no existencia del diagnóstico seleccionado.

2.5.4 Visualizar diagnósticos.

Dicho algoritmo está contenido en la función executeObtenerDiagnosticoVisualizar de la clase DiagnosticoActions.

```
public function executeObtenerDiagnosticoVisualizar()
{
    $diagnostico = TdiagnosticoPeer::ObtenerDiagnostico($this->getUser()->getAttribute('id_diag_act'));
    return $this->renderText(TdiagnosticoPeer::ToJsonDiagnosticoVisualizar($diagnostico));
}
```

Pasos del algoritmo que permite visualizar diagnóstico de la función executeObtenerDiagnosticoVisualizar:

1. Se realiza una búsqueda en la base de datos con el objetivo de comprobar que el *id* del diagnóstico que se desea visualizar es válido.
2. En caso de cumplirse lo anterior se procede a obtener la información del diagnóstico seleccionado y la misma es retornada.

2.5.5 Adicionar preguntas.

Dicho algoritmo está contenido en la función executeAdicionarPreguntaDiagnostico de la clase PreguntaAction. Permite adicionar las preguntas que van a conformar los diagnósticos.

```
// Gestionar Pregunta Diagnostico
public function executeAdicionarPreguntaDiagnostico()
{
if ($this->getRequest()->getMethod () != sfRequest::POST){
        return sfView::SUCCESS;
    }
    else {
            $enunciado = $this->getRequest()->getParameter('enunciado');
            $enunciado = str_replace('$', '*', $enunciado);
            $tipo = $this->getRequest()->getParameter('tipo');
            $this->executeAdicionarPregunta($enunciado, $tipo);

            $preguntaD = new TpreguntaDiagnostico();
            $pregunta = TpreguntaPeer::ObtenerPregunta($this->getUser()->getAttribute( 'id_pregunta_activa' ));
            $preguntaD->setIdPreguntaDiagnostico($pregunta->getIdPregunta());
            $preguntaD->setIdContexto($this->getRequest()->getParameter('id_contexto'));
            $preguntaD->setIdEtapa($this->getRequest()->getParameter('id_etapa'));
            $preguntaD->save();
            $pregunta->setTpreguntaDiagnostico($preguntaD);
            $pregunta->save();

    }
}
```

Pasos del algoritmo que permite adicionar las preguntas que van a conformar los diagnósticos:

1. Se comprueba al ejecutar la función executeAdicionarPreguntaDiagnostico si la información procedente de la vista está siendo enviada por el método Post.
2. En caso de no cumplirse lo anterior se mostrará la vista con los campos del formulario.
3. Se procede a adicionar en la base de datos el enunciado y el tipo de pregunta.
4. Se crea un objeto de tipo la clase TpreguntaDiagnostico.
5. Se procede a insertar la información en la base de datos y se redirecciona a la página donde se muestra un listado actualizado de todas las preguntas que van a conformar los diagnósticos existentes.

2.5.6 Obtener el listado de preguntas.

Dicho algoritmo está contenido en la función executeGestionarPreguntaDiagnostico de la clase PreguntaAction. Permite obtener las preguntas que van a conformar los diagnósticos.

```
public function executeGestionarPreguntaDiagnostico()
{
    $lista = TpreguntaDiagnosticoPeer::ObtenerPreguntas( );
    return $this->renderText(TpreguntaPeer::toJsonPreguntaDiagnostico($lista));
}
```

Pasos del algoritmo que permite obtener las preguntas que van a conformar los diagnósticos:

1. Se realiza una búsqueda en la base de datos con el objetivo de obtener todas las preguntas que pueden conformar un diagnóstico.
2. Se convierte la información a Json para enviarla hacia el Js.

2.5.7 Adicionar incisos.

Dicho algoritmo está contenido en la función executeAdicionarIncisoDiagnostico de la clase IncisoAction. Permite adicionar incisos a las preguntas que conforman los diagnósticos.

```
public function executeAdicionarIncisoDiagnostico()
{
    $pregunta = TpreguntaDiagnosticoPeer::ObtenerPregunta($this->getUser()->getAttribute ( 'id_pregunta_activa' ));
    if($pregunta != null){
        $inciso = TincisoEncuestaPeer::ObtenerInciso($this->getRequest ()->getParameter ( 'id_inciso' ));
        if($inciso != null){
            $inc_preg = new NincisoEncuestaPregunta();
            $inc_preg->setIdincisoencuesta($inciso->getIdInciso());
            $inc_preg->setIdpregunta($pregunta->getIdPregunta());

            $inc_preg->save();
            die();
        }
```

```
        else{
            $inc = new TincisoEncuesta();
            $inc->setTexto( $this->getRequest ()->getParameter ( 'texto' ) );
            $inc->setPonderacion( $this->getRequest ()->getParameter ( 'ponderacion' ) );
            $inc->save ();
            $inc_act = new TincisoPregunta();
            $inc_act->setNivel($this->getRequest()->getParameter('id_nivel'));
            $inc_act->setRespuesta($this->getRequest()->getParameter('respuesta'));
            $inc_act->setIdIncisoPregunta($inc->getIdInciso());
            $inc_act->save();
            $inc->setTincisoPregunta($inc_act);
            $inc->save ();
            $id_inciso = $inc->getIdInciso();
            $this->getUser()->setAttribute('id_inciso_activa', $id_inciso);
            $inc_preg = new NincisoEncuestaPregunta();
            $inc_preg->setIdincisoencuesta($id_inciso);
            $inc_preg->setIdpregunta($pregunta->getIdPreguntaDiagnostico());
            $inc_preg->save();
            die();
        }
    }
    else{
        return renderText('false');
        die();
    }
}
```

Pasos del algoritmo que permite adicionar incisos a las preguntas que conforman los diagnósticos:

1. Se realiza una búsqueda de la pregunta a la cual se desea agregar el inciso.
2. Cuando la pregunta es encontrada se procede a adicionarle el inciso.
3. Si el inciso ya existe se crea un objeto de tipo la clase NincisoEncuestaPregunta en la cual se va a guardar el *id* del inciso adicionado y el *id* de la pregunta a la cual se le adicionó dicho inciso.
4. Si el inciso no existe, se procede a crearlo y adicionarlo a la pregunta seleccionada.
5. Se procede a salvar la información en la base de datos.

2.5.8 Obtener incisos.

Dicho algoritmo está contenido en la función executeObtenerIncisosPreguntaDiagnostico de la clase IncisoAction. Permite obtener los incisos de las preguntas que conforman los diagnósticos.

```
public function executeObtenerIncisosPreguntaDiagnostico()
{
    $preguntas = null;
    $incisos = null;
    $lista = null;
    $d = 0;
    $preguntas = TpreguntaDiagnosticoPeer::ObtenerPreguntas();
        for ($b = 0; $b < count($preguntas); $b++){
            //Obtengo los incisos de cada pregunta
            $incisos = NincisoEncuestaPreguntaPeer::ObtenerIncisosIdPregunta($preguntas[$b]->getIdPregunta());
            for ($c = 0; $c < count($incisos); $c++){
                $lista[$d++] = $incisos[$c];
            }
        }
    return $this->renderText(NincisoEncuestaPreguntaPeer::toJsonIncisoDiagnostico($lista));
    die();
}
```

Pasos del algoritmo que permite obtener los incisos de las preguntas que conforman los diagnósticos:

1. Obtiene todas las preguntas que pueden conformar diagnósticos.
2. Obtiene los incisos de cada una de dichas preguntas y crea una lista en la cual se almacenará el *id* de la pregunta con todos sus incisos.
3. Se convierte la información a Json para enviarla hacia el Js.

2.6 Archivos y Carpetas.

La aplicación Web que se creará para el desarrollo del sistema será "VIRTEVALL". Todas las clases, ficheros YML, páginas JS y otros ficheros del sistema estarán agrupados en una estructura de carpetas y paquetes. Haciendo uso de las convenciones de nombrado de paquetes y adaptándolas al sistema quedaría como paquete principal: Source Files. Las clases y ficheros de recursos se encontrarán dentro de la carpeta apps.

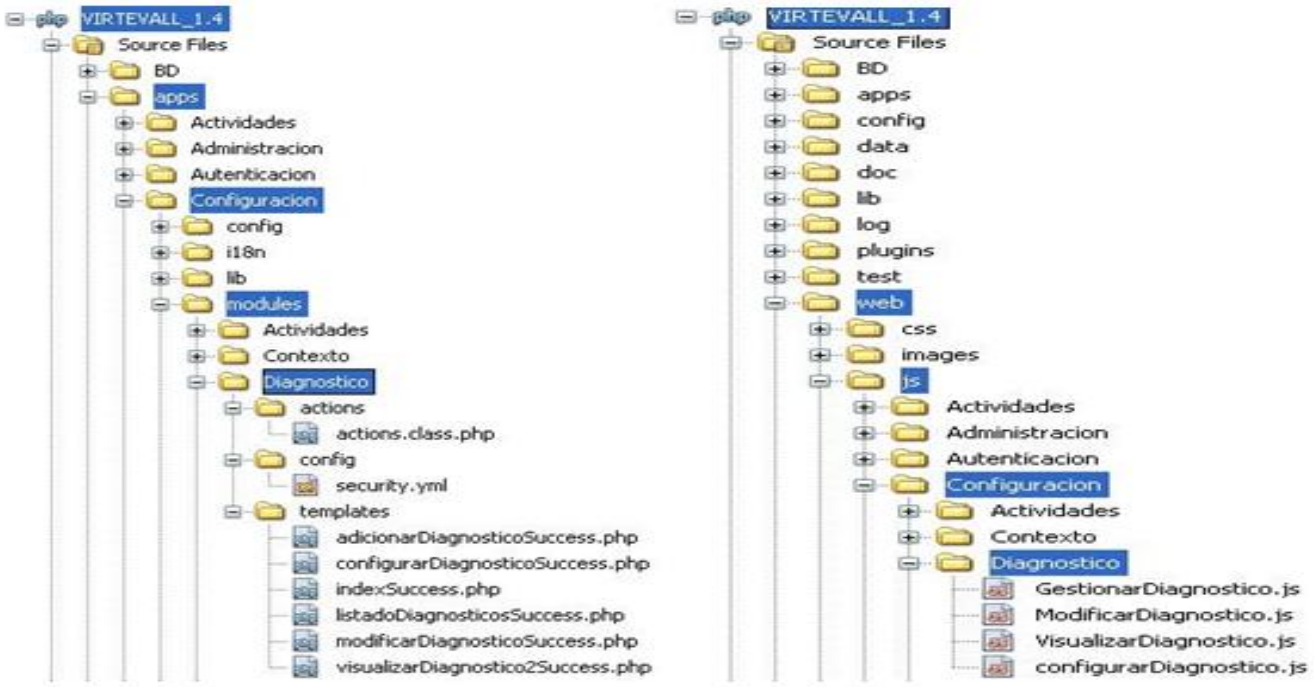


Figura 8: Organización de paquetes y subsistemas (módulo diagnóstico).

Figura 9: Organización de paquetes y subsistemas (módulo diagnóstico).

2.7 Descripción de las clases u operaciones necesarias.

En esta parte se hará una breve descripción de las principales clases que se utilizaron para la implementación del módulo de diagnóstico del proyecto VIRTEVALL.

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

1. **Nombre:** En el campo se escribirá el nombre o identificador de la clase.
2. **Tipo de clase:** En el campo se especifica el tipo de clase que representa la misma para el diseño de la solución.
3. **Nombre (Para cada responsabilidad):** En el campo se escribe el nombre o identificador de cada función o método de la clase.
4. **Descripción (Para cada responsabilidad):** En el campo se escribe una breve descripción del objetivo o acción principal de la función correspondiente unida a alguna observación oportuna que se requiera de la misma.

Tabla 1: Descripción de la clase DiagnosticoActions

Nombre: DiagnosticoActions	
Tipo de clase: Controladora	
Funciones:	
Nombre:	executeAdicionarDiagnostico()
Descripción:	Permite adicionar un Diagnóstico.
Nombre:	executeListadoDiagnosticos()
Descripción:	Muestra un listado con los Diagnósticos existentes.
Nombre:	executeModificarDiagnostico()
Descripción:	Permite modificar los elementos de un Diagnóstico
Nombre:	executeEliminarDiagnostico()
Descripción:	Permite eliminar un Diagnóstico.
Nombre:	executeActivarDiagnostico()
Descripción:	Permite activar un Diagnóstico.
Nombre:	executeObtenerDiagnosticoID()
Descripción:	Permite obtener un Diagnóstico dado el identificador del mismo.
Nombre:	executeVisualizarDiagnostico2()
Descripción:	Permite mostrar la vista dónde se visualizará el diagnóstico.
Nombre:	executeObtenerDiagnosticoVisualizar()
Descripción:	Permite obtener un Diagnóstico determinado que se quiera visualizar.

Tabla 2: Descripción de la clase PreguntaActions

Nombre: PreguntaActions	
Tipo de clase: Controladora	
Funciones:	
Nombre:	executeAdicionarPreguntaDiagnostico()
Descripción:	Permite adicionar una pregunta para los diagnósticos.
Nombre:	executeListadoPreguntasDiag()
Descripción:	Muestra la vista donde se muestra un listado con todas las preguntas que puede tener un diagnóstico.
Nombre:	excuteModificarPreguntaDiagnostico()
Descripción:	Modifica los elementos de una pregunta para los diagnósticos.
Nombre:	executeEliminarPreguntaDiagnostico()
Descripción:	Permite eliminar una pregunta de un diagnóstico.
Nombre:	executeActivarPregunta()
Descripción:	Permite activar una pregunta
Nombre:	executeGestionarPreguntaDiagnostico()
Descripción:	Muestra un listado con todas las preguntas que puede tener un diagnóstico.

Tabla 3: Descripción de la clase IncisoActions

Nombre: IncisoActions
Tipo de clase: Controladora
Funciones:

Nombre:	executeAdicionarIncisoDiagnostico()
Descripción:	Permite adicionar un inciso a una pregunta para los Diagnósticos.
Nombre:	executeListadoIncisosDiagnostico()
Descripción:	Muestra la vista donde se muestra un listado con todos los incisos de una pregunta que puede tener un diagnóstico.
Nombre:	executeModificarIncisoDiagnostico()
Descripción:	Modifica los elementos de un inciso de una pregunta para los diagnósticos
Nombre:	executeEliminarIncisoDiagnostico()
Descripción:	Permite eliminar un inciso de una pregunta para los diagnósticos.
Nombre:	executeActivarInciso()
Descripción:	Permite activar un inciso
Nombre:	executeGestionarIncisoDiagnostico()
Descripción:	Muestra un listado con todos los incisos que puede tener una pregunta para los diagnósticos.

Tabla 4: Descripción de la clase TdiagnosticoPeer

Nombre: TdiagnosticoPeer	
Tipo de clase: Modelo	
Funciones:	
Nombre:	ObtenerDiagnostico($id_diagnostico)
Descripción:	Permite obtener los datos de un diagnóstico.
Nombre:	EliminarDiagnostico($id_diagnostico)

Descripción:	Permite eliminar un Diagnóstico.
Nombre:	ToJsonDiagnosticoVisualizar($diagnostico)
Descripción:	Convierte y devuelve en formato Json un diagnóstico a visualizar.
Nombre:	toJson($listado)
Descripción	Convierte y devuelve en formato Json un listado de Diagnósticos.
Nombre:	toJsonOne($diagnostico)
Descripción	Convierte y devuelve en formato Json un Diagnósticos.

Tabla 5: Descripción de la clase BaseTdiagnostico

Nombre: BaseTdiagnostico	
Tipo de clase: Modelo	
Funciones:	
Nombre:	getIdDiagnostico() y setIdDiagnostico($v)
Descripción:	Permiten obtener el id de un diagnóstico y modificarlo.
Nombre:	getHabilidad() y setHabilidad($v)
Descripción:	Permiten obtener la habilidad de un diagnóstico y modificarla.
Nombre:	getCantPreguntas() y setCantPreguntas($v)
Descripción:	Permiten obtener la cantidad de preguntas de un diagnóstico y modificarla.
Nombre:	getContexto() y setContexto($v)
Descripción:	Permiten obtener el contexto de un diagnóstico y modificarlo.

Nombre:	getEstiloAprendizaje() y setEstiloAprendizaje($v)
Descripción:	Permiten obtener el estilo de aprendizaje de un diagnóstico y modificarlo.
Nombre:	getPerfilInteligencia() y setPerfilInteligencia($v)
Descripción:	Permiten obtener el perfil de inteligencia de un diagnóstico y modificarlo.
Nombre:	getEstrategiaAprendizaje() y setEstrategiaAprendizaje($v)
Descripción:	Permiten obtener la estrategia de aprendizaje de un diagnóstico y modificarla.
Nombre:	getOrientacion() y setOrientacion($v)
Descripción:	Permiten obtener la orientación de un diagnóstico y modificarla.
Nombre:	getNivel() y setNivel($v)
Descripción:	Permite obtener el nivel de un diagnóstico y modificarlo.

Tabla 6: Descripción de la clase TpreguntaDiagnosticoPeer

Nombre: TpreguntaDiagnosticoPeer	
Tipo de clase: Modelo	
Funciones:	
Nombre:	ObtenerPreguntaContexto($contexto)
Descripción:	Permite obtener los datos de todas las preguntas con el contexto especificado.
Nombre:	ObtenerPreguntas()
Descripción:	Permite obtener todas las preguntas

Nombre:	ObtenerPregunta($idpregunta)
Descripción:	Permite obtener los datos de una pregunta especificada.

2.8 Estándares de codificación

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

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

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

2.8.1 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.

1. **Clase**: Para las clases se escogió el UpperCamelCase.

```
class DiagnosticoActions extends sfActions
```

2. **Función:** Para las funciones se escogió el lowerCamelCase.

```
public function getIdDiagnostico()
```

3. **Variable**

```
protected $id_diagnostico;
```

2.8.2 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 define que la indentación se hará agregando un *tab* al inicio de la línea que se desee escribir. Se escribirá 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.

```
for (n = 0; n < P.count - 2; n++){
    for (m = n+1; m < P.count - 1; m++){
        if(P.preguntas[n].id_etapa > P.preguntas[m].id_etapa){
            aux=P.preguntas[m];
            P.preguntas[m]=P.preguntas[n];
            P.preguntas[n]=aux;
        }
    }
}
```

2.8.3 Llaves

Existe diversidad de criterios en cuanto a la ubicación de las llaves que delimitan el cuerpo de los bloques de código en los lenguajes que contienen este tipo de estructuras. Algunos programadores prefieren

hacerlo ubicando la llave de apertura inmediatamente detrás de la línea cabecera del bloque mientras otros apuestan por ubicarlas de forma solitaria en la línea siguiente a la línea cabecera. Para este último estilo existen además diferencias en cuanto al nivel de indentación de las mismas. Algunos lo hacen al nivel de la línea cabecera y otros al nivel de las líneas del cuerpo del bloque. Las llaves de apertura se colocarán al nivel de la línea cabecera. 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.

```
public function executeEliminarDiagnostico()
{
    $diagnostico = TdiagnosticoPeer::ObtenerDiagnostico($this->getRequest ()->getParameter ('id_diagnostico'));
    if ($diagnostico != null){
        TdiagnosticoPeer::EliminarDiagnostico($this->getRequest ()->getParameter ('id_diagnostico'));
        die();
    }

}
```

2.8.4 Líneas y espacios en blanco

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

- Entre funciones

```
public function executeAdicionarDiagnostico()
{
    //codigo
}

public function executeEliminarDiagnostico()
{
    //codigo
}
```

Se colocarán **espacios en blanco**:

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

```
protected $id_diagnostico;
```

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

```
handler: function(datos, numeros)
{
    //codigo
}
```

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

```
for (i = 0; i < P.count; i++){
```

2.8.6 Comentarios

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

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

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

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

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

```
//Organizar las preguntas por etapas de menor a mayor
for (n = 0; n < P.count - 2; n++){
    for (m = n+1; m < P.count - 1; m++){
        if(P.preguntas[n].id_etapa > P.preguntas[m].id_etapa){
            aux=P.preguntas[m];
            P.preguntas[m]=P.preguntas[n];
            P.preguntas[n]=aux;
        }
    }
}
```

2.9 Técnica de Inteligencia Artificial utilizada: Lógica Difusa.

En Inteligencia artificial (IA), la lógica difusa, o lógica borrosa se utiliza para la resolución de una variedad de problemas, principalmente los relacionados con control de procesos industriales complejos y sistemas de decisión en general entre los que se enmarca el sistema desarrollado. Los sistemas de lógica difusa están también muy extendidos en la tecnología cotidiana, por ejemplo en cámaras digitales, sistemas de aire acondicionado, lavarropas, etc. Los sistemas basados en lógica difusa imitan la forma en que toman decisiones los humanos a la hora de manipular la incertidumbre. Para darle solución a la respuesta que le brinda la implementación del Módulo Diagnóstico al sistema que sería el nivel académico actual del estudiante se hizo uso de esta técnica de IA, la cual cuenta con tres bloques fundamentales: el Difusor, Mecanismo de Inferencia y Desdifusor.

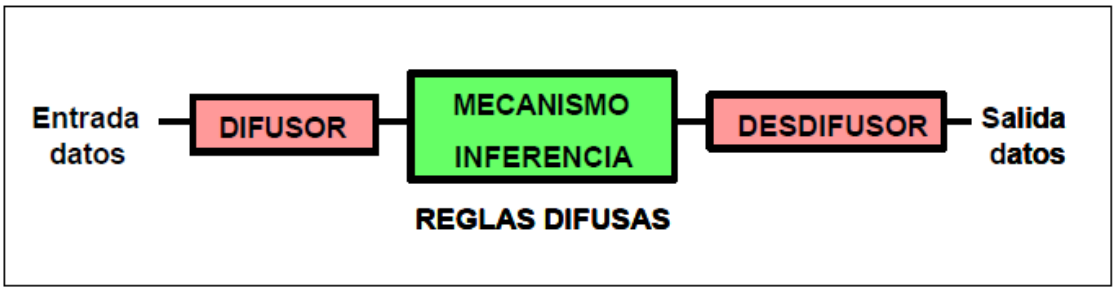


Figura 10: Esquema general de un sistema basado en lógica difusa.

Bloque Difusor: Bloque en que a cada variable de entrada se le asigna un grado de pertenencia a cada uno de los conjuntos difusos que se ha considerado, mediante las funciones características asociadas a estos conjuntos difusos. Las entradas a este bloque son valores concretos de las variables de entradas y las salidas son grados de pertenencias a los conjuntos difusos considerados. (difusa)

Bloque de Inferencia: Mediante los mecanismos de inferencia relaciona conjuntos difusos de entrada y de salida y representa a las reglas que definen el sistema. (difusa)

Bloque Desdifusor: Bloque en el cual a partir del conjunto difuso obtenido en el mecanismo de inferencia y mediante el método del Centroide (COG), se obtiene un valor concreto de la variable de salida, es decir, el resultado. (difusa)

En el sistema estos tres bloques están implementados en funciones separadas siendo la variable de entrada al bloque difusor las calificaciones obtenidas por preguntas al terminar el diagnóstico el estudiante, luego de obtener el grado de pertenencia correspondiente en dicho bloque se pasa a la inferencia haciendo uso de la reglas borrosas y luego al bloque Desdifusor donde se obtiene el resultado numérico mediante el método del Centroide (COG). Este resultado numérico obtenido no es más que el por ciento de cumplimiento de cada objetivo evaluado en el diagnóstico realizado para poder decidir el nivel académico (Básico, Intermedio, Avanzado) y la respectiva fase (A1, A2, B1, B2, C1, C2) de cada nivel en el que se encuentra el estudiante. A partir de este momento el sistema ya se encuentra en condiciones de comenzar el seguimiento al estudiante con información más precisa del nivel de aprendizaje del mismo para mostrarle las actividades a realizar de acuerdo a su nivel alcanzado en el Diagnóstico.

2.10 Conclusiones

En este capítulo se expuso el diseño entregado por el equipo de análisis el cual fue muy necesario en el proceso de implementación para obtener los resultados deseados. Se logró establecer una línea base sencilla que permita la programación eficiente por parte del desarrollador. Se establecieron los estándares de codificación pertinentes para que el código generado tenga la legibilidad necesaria al trabajar con algoritmos críticos y de grandes volúmenes de código. Se propusieron ejemplos sencillos incluyendo figuras para su comprensión. Se explicó la implementación de aquellas partes que pudieran resultar confusas, ambiguas o complicadas.

CAPÍTULO 3: VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

3.1 Introducción

Una vez terminado la implementación del Módulo de Diagnóstico y para validar la calidad y correspondencia con los requisitos funcionales y no funcionales para los cuales fue desarrollado, se presentan diferentes métodos y procedimientos que se emplearon para lograr este objetivo. Garantizar la calidad de los artefactos obtenidos en cada fase ayuda a minimizar los errores y obtener un producto con mayor calidad, que cumpla con todas las expectativas del cliente.

A continuación se muestran los procedimientos y métodos utilizados para medir la factibilidad de los artefactos obtenidos en el diseño a través de métricas de diseño de clases. Además son validadas las funcionalidades que fueron implementadas para el Módulo de Diagnóstico, mediante pruebas de caja blanca y pruebas de caja negra.

3.2 Pruebas de software

En la cadena de valor del desarrollo de un software específico, el proceso de prueba es clave a la hora de detectar errores o fallas. Conceptos como estabilidad, escalabilidad, eficiencia y seguridad se relacionan a la calidad de un producto bien desarrollado. El proceso de prueba es un proceso técnico especializado de investigación que requiere de profesionales altamente capacitados en lenguajes de desarrollo, métodos y técnicas de pruebas y herramientas especializadas.

La prueba no es una actividad sencilla, no es una etapa del proyecto en la cual se asegura la calidad, sino que la prueba debe ocurrir durante todo el ciclo de vida: poder probar la funcionalidad de los primeros prototipos; probar la estabilidad, cobertura y rendimiento de la arquitectura; probar el producto final. Lo que conduce al principal beneficio de la prueba: proporcionar feedback mientras hay todavía tiempo y recursos para hacer algo. (Juristo, y otros, 2006)

La prueba es un proceso que se enfoca sobre la lógica interna del software y las funciones externas, es un proceso de ejecución de un programa con la intención de descubrir un error. Un buen caso de prueba es

aquel que tiene alta probabilidad de mostrar un error no descubierto hasta entonces. Una prueba tiene éxito si descubre un error no detectado hasta entonces. (E.T.S.I)

Un proceso de ingeniería puede ser probado usando una de dos formas:

- Se pueden llevar a cabo pruebas que demuestren que cada función es completamente operativa.
- Se pueden desarrollar pruebas que aseguren que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada.

3.2.1 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)

Las pruebas de caja blanca son un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivar casos de prueba que garanticen que:

- Se ejercitan todos los caminos independientes de cada módulo.
- Se ejercitan todas las decisiones lógicas.
- Se ejecutan todos los bucles.
- Se ejecutan las estructuras de datos internas.

Para la realización de estas pruebas de caja blanca se utilizará la técnica de camino básico o cobertura de caminos, la misma propuesta inicialmente por Tom McCabe. El método del camino básico permite al diseñador de casos de prueba obtener una medida de la complejidad lógica de un diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución. En este tipo de pruebas se elaboran grafos de flujo, se determina su complejidad ciclomática para obtener el conjunto

básico de caminos linealmente independientes y de esta forma los casos de prueba del conjunto básico con el objetivo de ejecutar al menos una vez cada sentencia del programa.

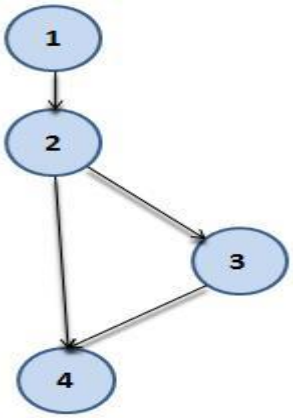
Dada la complejidad y extensión de las pruebas requeridas, en el presente capítulo se hará alusión a la prueba realizada al caso de uso eliminar diagnóstico del Módulo Gestionar Diagnóstico. A continuación se muestra la misma.

3.2.1.1 Prueba de caja blanca del caso de uso eliminar diagnóstico.

1. Función a probar:

```
public function executeEliminarDiagnostico()
   {
      $diagnostico = TdiagnosticoPeer::ObtenerDiagnostico($this->getRequest ()->getParameter
('id_diagnostico'));1
      if ($diagnostico != null){2
         TdiagnosticoPeer::EliminarDiagnostico($this->getRequest ()->getParameter ('id_diagnostico'));3
         die();3
      }
   }4
```

2. Grafo de complejidad



3. Complejidad ciclomática

   La complejidad ciclomática (VG) de un grafo de flujo está dada por la cantidad de aristas menos la cantidad de nodos más dos.

VG = (4 – 4) + 2 = 2

4. Caminos linealmente independientes.
   - 1 – 2 – 3 – 4
   - 1 – 2 – 4

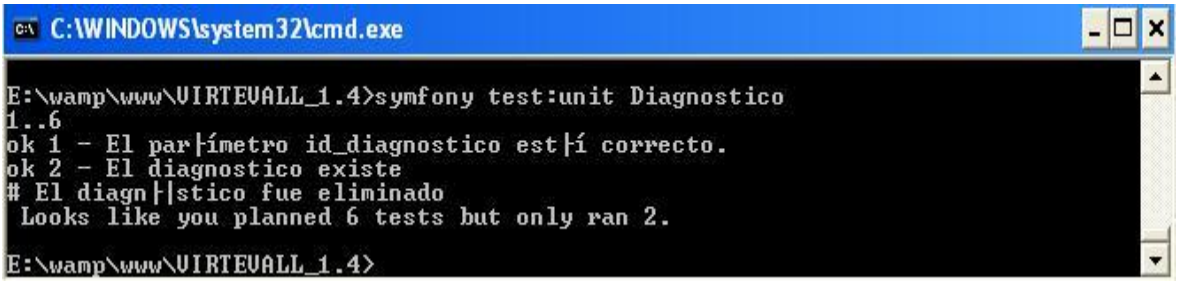
5. Caso de prueba

```
public function EliminarDiagnostico($diagnostico)
 {
  if ($this->obj->ok($diagnostico['id_diagnostico'] != '', 'El parámetro id_diagnostico está correcto.')) {
    if($this->obj->ok($this->existeDiagnostico($diagnostico['id_diagnostico']) != false, 'El diagnostico existe')) {
     TdiagnosticoPeer::EliminarDiagnostico($diagnostico['id_diagnostico']);
     $this->obj->diag('El diagnóstico fue eliminado');
    }
    else
     {
       $this->obj->diag('El diagnóstico no fue eliminado');
     }
  }

 }
```

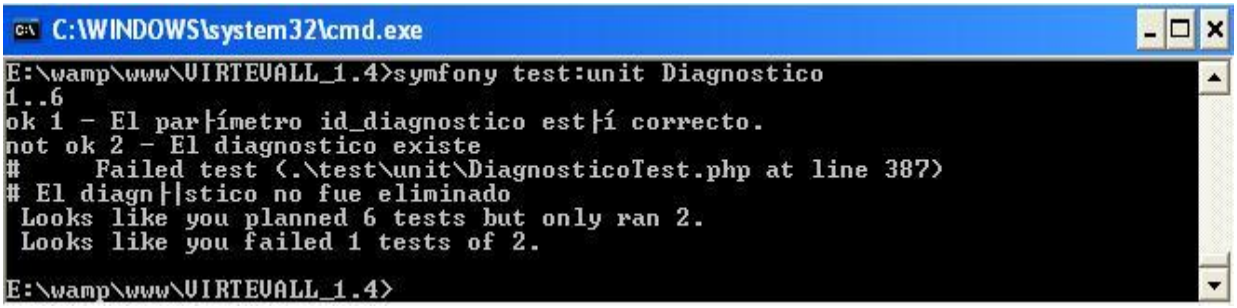
6. Juego de datos y resultado de la prueba

```
//Datos camino independiente
$juego_camino_independiente_6 = array (
'id_diagnostico' => '22');
```




```
//Datos camino independiente
$juego_camino_independiente_6 = array (
```

'id_diagnostico' => ' ');



3.2.2 Pruebas de caja negra

Se denomina caja negra a aquel elemento que es estudiado desde el punto de vista de las entradas que recibe y las salidas o respuestas que produce, sin tener en cuenta su funcionamiento interno. En otras palabras, de una caja negra interesará su forma de interactuar con el medio que le rodea (en ocasiones, otros elementos que también podrían ser cajas negras) entendiendo qué es lo que hace, pero sin dar importancia a cómo lo hace. Por tanto, de una prueba de caja negra deben estar muy bien definidas sus entradas y salidas, es decir, su interfaz; en cambio, no se precisa definir ni conocer los detalles internos de su funcionamiento. (ISO)
Los casos de prueba de la caja negra pretenden demostrar que:

- Las funciones del software son operativas.
- La entrada se acepta de forma adecuada.
- Se produce una salida correcta y la integridad de la información externa se mantiene.

La prueba de la caja negra intenta encontrar errores de las siguientes categorías: (E.T.S.I)

- Funciones incorrectas o ausentes.
- Errores de interfaz.
- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación.

Las pruebas de caja negra ejercitan los requisitos funcionales desde el exterior del módulo.

3.2.2.1 Diseño de casos de pruebas

Nombre del caso: Eliminar Diagnóstico de entrenamiento.

Condiciones de ejecución.

- Se debe identificar y autenticar ante el sistema y además debe tener los permisos para ejecutar esta acción.
- Se debe seleccionar el Módulo de Diagnóstico.
- Se debe seleccionar la opción Diagnóstico de Entrenamiento.
- Se debe seleccionar el diagnóstico que se desea eliminar.
- Se debe seleccionar la opción Eliminar.

Requisito a probar:

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
Eliminar Diagnóstico de entrenamiento	El sistema debe permitir eliminar un Diagnóstico de Entrenamiento existente.	EP 1.1: Eliminar un diagnóstico de entrenamiento correctamente.	– Se debe seleccionar el diagnóstico que se desea eliminar. – Se debe presionar el botón **Eliminar**. – Se debe presionar el botón **Aceptar**.
		EP 1.2: Cancelar la operación	– Se debe seleccionar el diagnóstico que se desea eliminar. – Se debe presionar el botón **Eliminar**. – Se debe presionar el botón **Cancelar**.

Descripción de variables.

Este caso de uso no presenta variables.

Id del escenario	Escenario	Respuesta del sistema	Resultado de la prueba
EP 1.1	Eliminar un diagnóstico de entrenamiento correctamente.	Elimina el diagnóstico seleccionado de la lista de diagnósticos y muestra la lista con los diagnósticos restantes.	Los resultados de la prueba fueron satisfactorios, la respuesta del sistema fue la esperada.
EP 1.2	Cancelar la operación.	Se cancela la operación y se cierra la ventana.	Los resultados de la prueba fueron satisfactorios, la respuesta del sistema fue la esperada.

Juego de datos a probar.

Registro de defectos y dificultades detectados

No se encontraron dificultades para este caso de uso.

3.3 Tabla de no conformidades

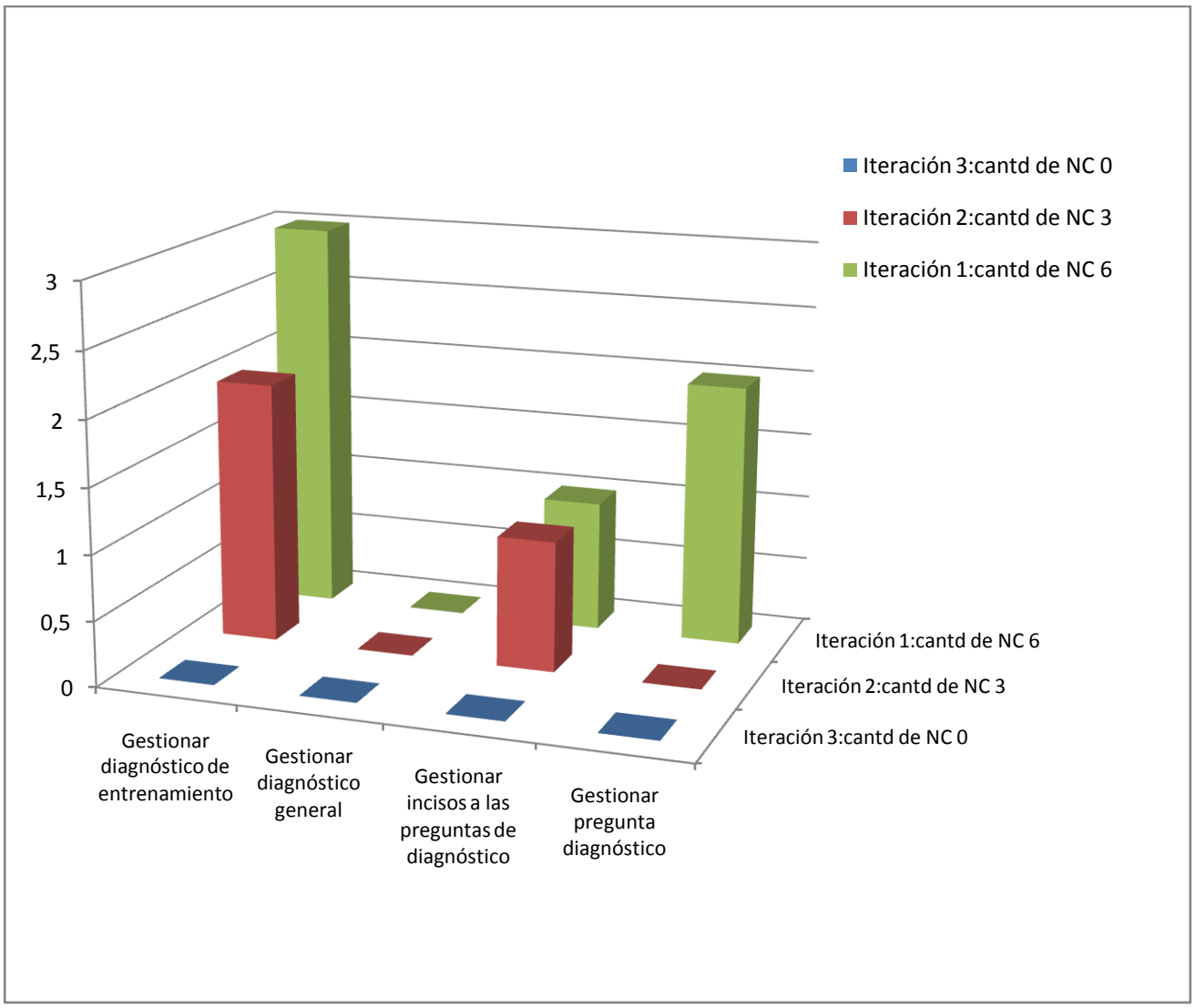


Tabla 7: No conformidades

3.4 Conclusiones

En este capítulo se hace referencia al proceso de pruebas de software. Se hace una síntesis de la descripción y el funcionamiento de las pruebas utilizadas. Se diseñaron casos de pruebas que permitieron, a partir de los resultados obtenidos, comprobar el correcto funcionamiento del software. Se realizó una tabla donde se reflejan las iteraciones realizadas durante el proceso de prueba y la cantidad de no

conformidades detectadas en cada caso de uso. Por último el sistema fue entregado al proyecto de calidad de la facultad 15 y fue liberado satisfactoriamente. (Para mayor información ver anexo_1)

CONCLUSIONES GENERALES

Al finalizar el presente trabajo se arribó a las siguientes conclusiones:

- Se realizó una revisión bibliográfica de las tecnologías actuales para el desarrollo de sistemas web, enfatizando en las tecnologías y herramientas libres para la implementación del sistema y de esta forma se decidió utilizar como lenguaje de programación PHP 5 en su versión 5.2.5, con el framework Symfony versión 1.2.8, sobre el entorno de desarrollo NetBeans 6.8 M2, utilizando el Sistema Gestor de Base de Datos PostgreSQL en su versión 8.3.4.1 y la librería ExtJS versión 2.2.
- Para lograr una correcta implementación del módulo de diagnóstico se consultaron los diagramas de clases del diseño y de componentes realizados por el equipo de análisis y diseño del proyecto, los cuales brindaron una fácil y mejor comprensión de las relaciones e interacciones entre los componentes.
- Con la implementación de la técnica de inteligencia artificial lógica difusa se logró obtener el por ciento de cumplimiento de cada objetivo evaluado en el diagnóstico realizado para poder decidir el nivel académico y la respectiva fase de cada nivel en el que se encuentra el estudiante.
- Con la aplicación de estándares de codificación internacionales como lo son el lowerCamelCase y el UpperCamelCase, se logró un mejor entendimiento y organización del código.
- Las pruebas realizadas para validar las funcionalidades que fueron implementadas, demuestran que el módulo Diagnóstico cumple satisfactoriamente con los requisitos que garantizan su correcto funcionamiento.
- Se logró una correcta implementación del módulo de diagnóstico, el cual le brinda la potencialidad al Tutor Virtual de Evaluación para el Aprendizaje Autónomo de Idiomas de realizar la gestión de diagnósticos y de esta forma le permite conocer el nivel de posicionamiento de los estudiantes para poder llevar a cabo su seguimiento en las futuras actividades a realizar en el sistema. De esta manera se le da solución al problema científico y cumplimiento al objetivo general planteado en la investigación del presente trabajo.

RECOMENDACIONES

- Profundizar más en las técnicas de inteligencia artificial para lograr la implementación de la funcionalidad “Reconocimiento de voz”, con el fin de realizar diagnósticos basados en la habilidad de expresión oral.
- Agregar al sistema un módulo de reporte con el fin de mostrar gráficamente a los profesores el avance de cada uno de los estudiantes y a su vez que estos también puedan observar su evolución a lo largo de su trayectoria en el sistema.

TRABAJOS CITADOS

Bird, Richard. 2000. *Introducción a la Programación Funcional con Haskell.* 2000.

Brown, J.S. and Burton, R.R. 1978. Diagnostic models for procedural bugs in basic mathematical skills. 1978.

Brown, J.S. y Burton, R.R... and Kleer, de. 1982. Pedagogical, natural language and knowledge engineering. [book auth.] D. Sleeman and J. S. Brown. 1982.

Burton, R. 1982.. Diagnosing bugs in a simple procedural skill. En Sleeman, D. y Brown, J. (Eds), Intelligent . 1982.

Carbonell, J.R. 1970. *AI in CAI: An artificial intelligence approach to computer assisted instruction. IEEE transaction on Man Machine System.* 1970. Vol 11, No 4.

Definición.org. Definición. [En línea] [Citado el: 10 de 2 de 2010.] http://www.definicion.org/lenguaje-de-programacion.

difusa, Lógica. Conceptos fundamentales de lógica difusa. [En línea] [Citado el: 16 de 6 de 2010.] http://www.tdr.cesca.es/TESIS_UPC/AVAILABLE/TDX-0207105-105056//04Rpp04de11.pdf.

E.T.S.I. Departamento de Lenguajes y sistema Informáticos. [En línea] Universidad de Granada.

ExtJS. Desarrollo en Web. [En línea] [Citado el: 10 de 2 de 2010.] http://blogs.antartec.com/desarrolloweb/2008/10/extjs-lo-bueno-lo-malo-y-lo-feo/.

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

Facultad Regional de Buenos Aires. 2005. *Fundamentos teóricos de los Paradigmas de Programación.* Buenos Aires : s.n., 2005.

Gutiérrez, J.J.,. *¿Qué es un framework web?*

Hommel, S. 1999. *Convenciones de Código para el lenguaje de programación Java TM.* 1999.

IDE, NetBeans. Desarrollo de aplicaciones multiplataforma con NetBeans IDE. [En línea] [Citado el: 10 de 2 de 2010.] http://www.sun.com/emrkt/innercircle/newsletter/latam/0207latam_feature.html.

ISO. in2test.lsi.uniovi.es. [En línea] [Citado el: 10 de 2 de 2010.] http://in2test.lsi.uniovi.es/gt26/.

Jacaboson, I, Booch, G and Rumbaugh J. 2000. *El Proceso Unificado de Desarrollo de Software.* s.l. : Addison Wesley, 2000.

Johnson, W. L. 1986. *Intention-based diagnosis of novice programming errors. Morgan-Kauffman.* 1986.

Juristo, N, Moreno, A y Vegas, S. 2006. *TÉCNICAS DE EVALUACIÓN DE SOFTWARE.* 2006.

Kruchten, P. 200. *The Rational Unified Process: An Introduction.* s.l. : Addison Wesley , 200.

netbeans.org. netbeans. [En línea] [Citado el: 9 de 2 de 2010.] http://www.netbeans.org/about/history.html.

PHP:Hypertext. PHP. [En línea] [Citado el: 9 de 2 de 2010.] http://www.php.net/.

postgresql.org. postgresql. [En línea] [Citado el: 10 de 2 de 2010.] http://www.postgresql.org/.

Rossel, Gerardo. 2004. Amzi! Prolog. *sitio Web Amzi!* [En línea] 2004. http://www.amzi.com/.

SlideShare, 2009. SlideShare. [En línea] [Citado el: 9 de febrero de 2010.] http://www.slideshare.net/remitos/sw-magazine-mx1.

Stevens, A. and Collins, A. 1977. The goal structure of a Socratic tutor. In Proceedings of the National ACM. Conference. New York: ACM. : s.n., 1977.

symfony.es. symfony. [En línea] [Citado el: 9 de 2 de 2010.] http://www.symfony.es.

Vanlehn, K. 1988. Student Modelling. M. Polson. Foundations of Intelligent Tutoring systems. Hillsdale. N.J. : Lawrence Erlbaum Associates, 1988.

Weitzenfeld, Alfredo. 2004. *Ingeniería de software orientada a objetos con UML, Java e Internet.* 2004.

Winblad, Ann L. 1993. *Software Orientado a Objetos.* 1993.

Wolf, B. 1984. *Context Dependent Planning in a Machine Tutor. Ph.D. Dissertation, University of Massachusetts.* Amherst, Massachusetts. : s.n., 1984.

BIBLIOGRAFÍA

Cataldi, Z y Salgueiro, F y Lage, F J y García, R. Sistemas Tutores Inteligentes: los estilos del estudiante para selección del tutorizado. http://www.itba.edu.ar/archivos/secciones/57wicc2005-SISTEMAS-TUTORES-INTELIGENTES.pdf

Corredor, M. Sistemas Tutoriales Inteligentes.
http://www.colombiaprende.edu.co/html/mediateca/1607/articles-126382_archivo.pdf

Delgado Lechuga, Gustavo. Modelado de un entorno virtual de aprendizaje interactivo para fomentar el desempeño académico en la educación en línea.
http://www.comie.org.mx/congreso/memoria/v10/pdf/area_tematica_07/ponencias/1345-F.pdf

Del Valle Barrientos, F. J. y Muñoz, J. y Pérez, C. Tutor Inteligente en un Ambiente Virtual de Experimentación (TIAVE).
http://ingsw.ccbas.uaa.mx/sitio/images/publicaciones/ANIEI04%28Valle_Munoz%29.pdf

Duran, Elena B. 2006 Modelo del alumno para sistemas de Aprendizaje Colaborativo. Universidad Nacional de Santiago del Estero.

González, C.S. Sistemas Inteligentes en la Educación: Una revisión de las líneas de investigación actuales. Revista Electrónica de Investigación y Evaluación Educativa, ISSN 1134-4032, Vol. 10, Nº. 1, 2004 http://dialnet.unirioja.es/servlet/oaiart?codigo=864059

González, Héctor M.Sc. y Duque, Néstor y Ovalle, Demetrio 2008 Modelo del Estudiante para Sistemas Adaptativos de Educación Virtual. Universidad Nacional de Colombia.
http://pisis.unalmed.edu.co/avances/archivos/ediciones/Edicion%20Avances%202008%201/22.pdf

Huapaya, C.R. Y Arona, G.M. Y Lizarralde, F.A. 2005 Sistemas Tutoriales Inteligentes Aplicados a Dominios de la Ingeniería. Universidad Nacional de Mar del Plata.
http://cs.uns.edu.ar/jeitics2005/Trabajos/pdf/22.pdf

Jiménez, Jovani A. M.Sc. Un modelo de integración de sistemas tutoriales inteligentes y ambientes colaborativos de aprendizaje bajo el esquema de universidad virtual. Universidad Nacional de Colombia, Colombia. http://medusa.unimet.edu.ve/programacion/actasCite2002/pdf/jjimenezColombia.pdf

Lage, Z. C y Fernando J. Sistemas Tutores Inteligentes Orientados a la Enseñanza para la Comprensión.
http://edutec.rediris.es/Revelec2/revelec28/edutec28_sistemas_tutores_inteligentes.html.

Mora, B. X. y Soto, F. A. Sistemas Tutores Inteligentes. http://www.slideshare.net/xavieraracely/sistemas-tutores-inteligentes

Ortega, M. y Sánchez, P.P. y Peces, C. AWLA y Eduwebman: Una propuesta para el desarrollo de Sistemas de e-learning colaborativos en web.
http://campus.usal.es/~teoriaeducacion/rev_numero_06_2/n6_02_art_ortega_sanchez_peces.htm

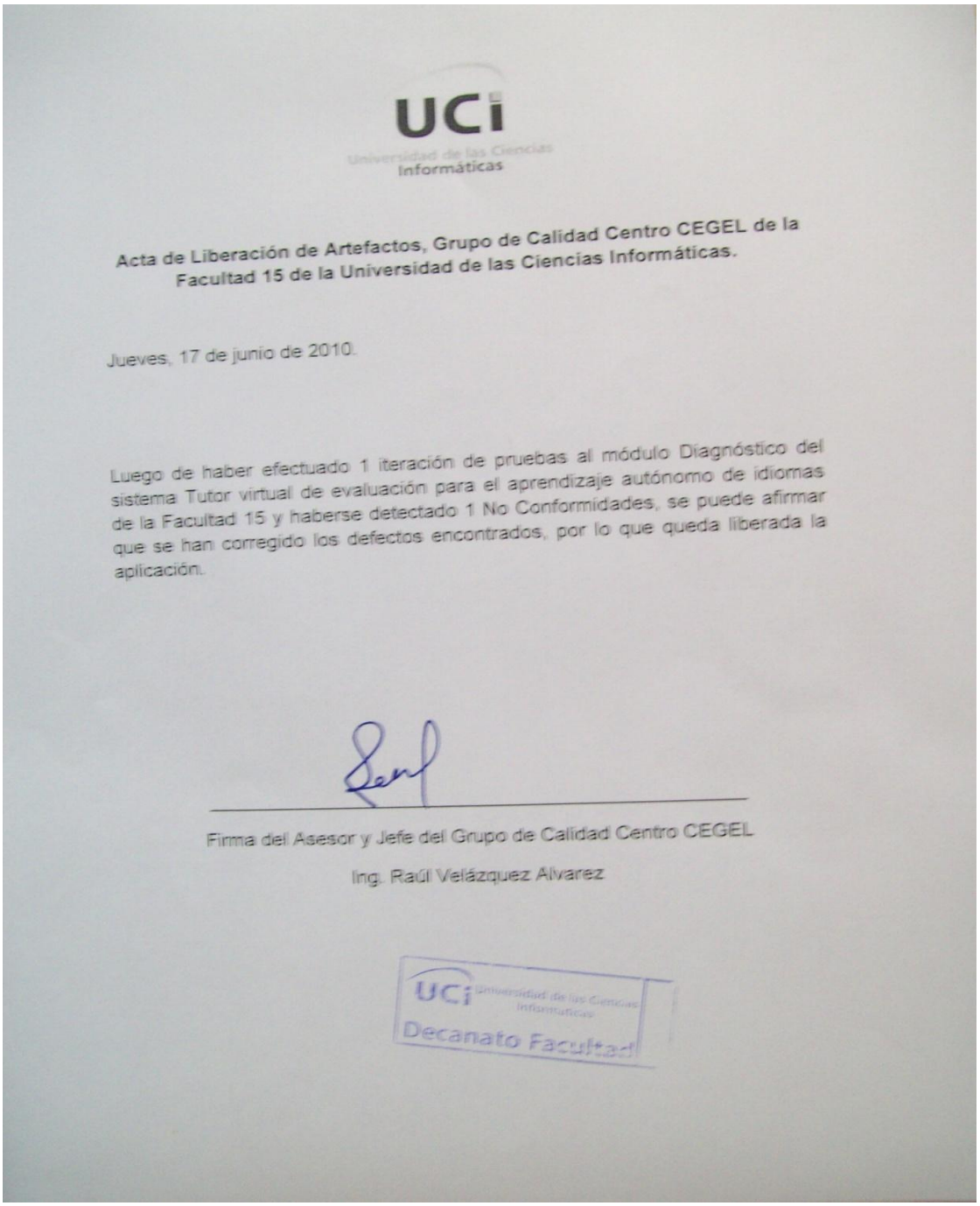
OVALLE, D.A. y JIMÉNEZ, J. A. Entorno Integrado de Enseñanza / Aprendizaje basado en Sistemas Tutoriales Inteligentes & Ambientes Colaborativos. http://www.iiisci.org/Journal/CV$/risci/pdfs/P554466.pdf

Reynoso, David A. 2009 Desarrollo de un Sistema Tutorial Inteligente –STI- utilizando Visual Basic 6.0 para la fundamentación teórica de la operación unitaria de lixiviación inducida (Extracción sólido-líquido) orientado a materiales biológicos (Metabolitos). Universidad de San Carlos, Guatemala. http://biblioteca.usac.edu.gt/tesis/08/08_8963.pdf

Urretavizcaya, M. Sistemas Inteligentes en el ámbito de la Educación. http://aepia.lcc.uma.es/index.php/ia/article/viewFile/703/695

Zavaleta, J. J. y Vasconcelos, L. C. Sistemas Tutores Inteligentes. http://www.cos.ufrj.br/~ines/courses/cos740/leila/cos740/STImono.pdf

ANEXOS



UCi
Universidad de las Ciencias Informáticas

Acta de Liberación de Artefactos, Grupo de Calidad Centro CEGEL de la Facultad 15 de la Universidad de las Ciencias Informáticas.

Jueves, 17 de junio de 2010.

Luego de haber efectuado 1 iteración de pruebas al módulo Diagnóstico del sistema Tutor virtual de evaluación para el aprendizaje autónomo de idiomas de la Facultad 15 y haberse detectado 1 No Conformidades, se puede afirmar que se han corregido los defectos encontrados, por lo que queda liberada la aplicación.

Firma del Asesor y Jefe del Grupo de Calidad Centro CEGEL

Ing. Raúl Velázquez Alvarez

UCi Universidad de las Ciencias Informáticas
Decanato Facultad