Máster Universitario en Software Libre

Trabajo Final de Máster: EvaExam HTML Report

Consultor: Francisco Javier Noguera Otero

Tutor: Silvio Erwert

Presenta:

Mario Alberto Díaz Castellanos

10.01.2014

Licencia de Software

EvaExam HTML Report es software libre y se distribuye bajo la licencia GPLv3. El código fuente de la aplicación puede descargarse del siguiente repositorio:

https://github.com/codergolem/EvaExam-HTML-Report

Tabla de Contenido

Resumen

El siguiente documento consiste en la memoria de trabajo del proyecto de fin carrera titulado "*EvaExam HTML Report*" correspondiente al Máster en Software Libre de la Universidad Oberta de Catalunya.

El proyecto tuvo la finalidad de crear un *plugin* para la aplicación web *EvaExam que* permite presentar los resultados de exámenes en un informe *HTML*. El proyecto además formó parte del trabajo central de las prácticas de fin de carrera que se realizaron en la empresa E*lectric Paper Evaluationssysteme GmbH.*

EvaExam es un aplicación web desarrollada y comercializada por la empresa alemana Electric Paper Evaluationssysteme GmbH .La aplicación funciona como un asistente para la realización de evaluaciones en distintos ámbitos,cuenta con herramientas para realizar el diseño de formularios digitales de pruebas y exámenes , la presentación de los mismos en formato impreso o en línea a los participantes de la evaluación, la recolección de los datos y el análisis y presentación de los resultados.

El módulo *EvaExam HTML Report* que se desarrolló como producto final de este proyecto consiste en un componente que se instala como un *plugin*, es decir que no forma parte de las funcionalidades por defecto de la aplicación, y que muestra los resultados de los exámenes por medio de gráficos, filtros de contenido, menús de navegación, entre otras características implementadas por medio de P*HP, HTML y JavaScript*.

El documento presenta el contexto de realización del proyecto,las especificaciones técnicas del producto final y las etapas y actividades realizadas para su consecución.

1 Introducción al Proyecto

EvaExam HTML Report es un proyecto realizado como trabajo final de máster de aplicación profesional para el itinerario de especialización en "*Administración Web y Comercio Electrónico"*. El objetivo del proyecto se definió de la siguiente manera:

Crear un informe en formato HTML para la presentación de los resultados de exámenes en EvaExam.

El proyecto se realizó también como parte del trabajo de prácticas en la empresa alemana *Electric Paper Evaluationssysteme y* requirió para su realización aproximadamente 370 horas, distribuidas en un periodo que cubrió casi todo el año 2013.

EvaExam es una aplicación web con licencia privativa que la empresa *Electric Paper Evaluationssysteme* comercializa en Europa, Estados Unidos y Canadá. Su función es automatizar la mayoría de las tareas involucradas en la realización de evaluaciones. El *plugin* desarrollado añade a *EvaExam* un informe HTML para la presentación de los resultados de las evaluaciones.

El *plugin* se programó utilizando *PHP,JavaScript y HTML*, su arquitectura se basa en el paradigma modelo-vista-controlador y hace uso del *framework Zend*[1] como plataforma de desarrollo,la comunicación con el programa central se realiza por medio de una interfaz *SOAP*[2].

En este capítulo se presentará la descripción del proceso de selección del proyecto,la introducción al uso y funcionamiento de *EvaExam*, así como referencias técnicas y aspectos legales que se consideraron importantes para comprender mejor el resto del documento.

En los capítulos posteriores se documentan cada una de las etapas que se llevaron a cabo para la realización del proyecto, las cuales están basadas en los estándares para proyectos de desarrollo de software.

1 http://framework.zend.com/
2 siglas de *Simple Object Access Protocol*

1.1 El proceso de selección del proyecto

EvaExam es el segundo producto en importancia para la empresa que lo produce y se encuentra desde inicios del año 2013 en una nueva etapa de promoción como producto independiente, ya que anteriormente se había comercializado como un módulo para otro programa. Para impulsar las ventas del programa en esta nueva fase, se añadieron varias funciones y se mejoró la interfaz de usuario, además la búsqueda de ideas para mejorar el programa continua y se tiene planeada la liberación de una nueva versión del programa por lo menos una vez por año.

En el proceso de definir el tema del trabajo final del máster y de las prácticas, se consideró que un proyecto para implementar una nueva función o mejora en *EvaExam* cumplía con el perfil técnico necesario para ser el tema del trabajo final. Los alcances del proyecto se podían adaptar para hacerlos realizables en el número total de horas disponibles fijado por los requerimientos de la práctica. De este modo tanto la empresa como el estudiante obtenían un beneficio directo, la primera por obtener recursos humanos adicionales para el desarrollo de *EvaExam;* y el segundo por tener la posibilidad de aplicar los conocimientos adquiridos en el máster para desarrollar un producto comercial, utilizado por una amplia base de usuarios.

El siguiente paso consistió en discutir y analizar el tema del proyecto, para ello se consultó a todos los involucrados, entre ellos: el equipo de desarrollo, el jefe de implementación, el equipo de entrenamiento y soporte, y el gerente. Durante varias reuniones se recogieron las propuestas de cada uno de los participantes. Para tomar la decisión final se consideró la relevancia (desde la perspectiva del usuario) de cada una de las propuestas, los recursos estimados necesarios para llevarla a cabo y la viabilidad técnica de la misma.

Finalmente entre todas las propuestas se escogió la de mejorar la presentación de los resultados en *EvaExam*, puesto que esta parte del programa no había sido modificada sustancialmente en la última versión. Se consideró que el informe *PDF* presentaba un aspecto visual poco agradable, que además dificultaba la identificación de los datos más relevantes, también se tomaron en cuenta sugerencias de los clientes que solicitaban un informe que permitiera clasificar los resultados basándose en diversos criterios, como la nota obtenida, las preguntas con mejor puntuación, etc. Se determinó que por la dificultad técnica que implicaba, no era viable extender el informe *PDF* actual para incluir las ideas mencionadas anteriormente y se decidió que lo más sencillo era crear un informe en formato *HTML*, no como sustituto del informe *PDF* sino como una opción adicional para el usuario

1.2 Introducción a EvaExam

Para entender las características del proyecto es necesario hacer una breve introducción sobre el funcionamiento y uso de *EvaExam*[1], puesto que el objetivo del proyecto fue desarrollar un *plugin* para esta aplicación. A continuación se presenta una descripción general de la arquitectura, funciones y aspectos legales de este programa.

EvaExam es una aplicación cliente servidor que originalmente formaba parte del programa *EvaSys* y que en su versión más reciente opera como un componente que se puede instalar de manera independiente.

La aplicación funciona como un asistente para la realización y calificación automática de exámenes, los cuales pueden presentarse en formato impreso o en línea. Proporciona herramientas de software que cubren todas las etapas del proceso de evaluación, incluyendo el diseño del formulario de examen por medio de un editor web, la distribución de los formularios de prueba a los participantes, el procesamiento de dichos formularios y la presentación de los resultados.

1.2.1 Arquitectura y Componentes de *EvaExam*

EvaExam es una aplicación web basada en *php* que se instala sobre sistemas operativos *Windows*. La aplicación consta de cuatro componentes: el *frontend* que consiste en la interfaz del usuario, la base datos que contiene la información del sistema, el *VividForms Reader* que se encarga de la digitalización de las pruebas impresas y el *ScanStation* que se conecta a dispositivos de escáner. A continuación se describen brevemente cada uno de estos componentes.

El frontend está programado en *php* y hace uso intensivo de *JavaScript*, es la interfaz con la que el usuario interacciona, con él el administrador del sistema define los usuarios y roles del mismo, configura los parámetros del sistema, y por medio de él los examinadores crean los formularios de exámenes y tienen acceso a los resultados.

La base de datos de *EvaExam* contiene toda la información del sistema, incluyendo los usuarios, los datos de los formularios, los resultados de los exámenes, etc. En la versión actual de *EvaExam* es posible configurar la base de datos con los gestores *Microsoft SQL Server* y *MySQL*.

*El VivivForms Reader e*s una aplicación ejecutable para *Windows*, su función es digitalizar la información contenida en las imágenes escaneadas de los formularios de prueba, utiliza la técnica de reconocimiento óptico de marcas (*OMR* por sus siglas en ingles) para determinar las respuestas dadas por los examinados y transferir esta información a la base de datos.

El ScansStation es también una aplicación ejecutable para *Windows* que se conecta a los escáneres y su función es validar las imágenes escaneadas y luego transferirlas al *VividForms Reader*.

1.2.2 Especificación Técnica de *EvaExam*

Como ya se mencionó *EvaExam* es una aplicación que se puede instalar únicamente sobre plataformas *Window*s. Se puede configurar para funcionar con los servidores web Apache y Microsoft IIS, y con los gestores de bases de datos *MySQL* y *Microsoft SQL Server.* La siguiente tabla resume las características técnicas de *EvaExam.*

1 Se puede encontrar una descripción general del programa (en ingles) en: http://www.evasys.co.uk/products/evaexam.html

EvaExam 6.0 2000 (Última versión estable a Diciembre del 2014)					
Servidor Web	**Base de Datos**	**PHP**	**JavaScript Framework**	**Sistema Operativo**	**Navegadores**
-Apache 2.2 -Microsoft IIS 6.0 o superior	-MySQL -Microsoft SQL Server 2005 o superior	5.3	Dojo 1.9	Windows Server 2003, 2008, 2012	-Firefox 3.5+ -Safari 4+ -Chrome 28+ -Internet Explorer 8 o superior -Opera 12+

Tabla 1.1: Especificación técnica de EvaExam

1.2.3 Exámenes y Pruebas En Linea e Impresas

En cuanto al formato en que el formulario se presenta al examinado existen dos posibilidades: en formato impreso o en línea.

En el formato impreso como su nombre lo indica, el usuario tiene que imprimir los formularios de exámenes y distribuirlos a los examinados, estos tiene que anotar en cada formulario su número de matrícula o identificación para que los resultados puedan ser individualizados, los formularios contestados se digitalizan posteriormente por medio del *VividForms Reader*.

En el formato en línea los participantes reciben un código de acceso generado automáticamente por *EvaExam* con el que acceden, por medio de un navegador web, al formulario de examen en formato HTML, cuando el examinado termina de contestar el examen, el formulario se enviá de regreso al servidor y los resultados se transfieren a la base de datos.

1.2.4 La Presentación de Los Resultados

Una vez que los resultados se han almacenado en la base de datos, el examinador puede abrir el informe de examen. Los resultados en *EvaExam* se presentan en dos formatos: *PDF* y *CSV.* El primero es él más usado, consiste en un informe con los resultados individuales de los examinados, incluyendo los puntos obtenidos en cada pregunta y los puntos obtenidos en total, si el examinado aprobó o no el examen y la nota obtenida. Además contiene datos estadísticos que se obtienen a partir de los resultados globales de la prueba.

El informe *csv* consiste en un archivo en este formato (csv del ingles *comma separated values)* con los resultados individuales de la prueba, se utiliza principalmente para exportar los resultados a otros programas.

1.2.5 Licencia de *EvaExam*

El modelo de negocios de *EvaExam* se basa en la venta de licencias de uso acompañadas en muchos casos por un contrato de soporte y entrenamiento, acorde con este modelo de negocios el código de *EvaExam* se distribuye bajo los términos de una licencia privativa, sin embargo la aplicación hace

uso de componentes de software libre creados por terceros como son el servidor web *Apache,* la base de datos *MySQL* y la librería JavaScript *Dojo*.

1.3 El Software Libre y El Proyecto

Como se ha mencionado el proyecto es parte de las actividades del Máster en Software Libre de la UOC, por ello uno de los requerimientos invariables fue que el código generado se distribuyera bajo alguna licencia de software libre.

1.3.1 La Definición de Software Libre

El software libre es, según la definición[1] de la *Free Software Foundation* un programa de computadora que otorga al usuario las siguientes libertades:

- **Libertad 0:** La libertad de ejecutar el programa para cualquier propósito.
- **Libertad 1:** La libertad de estudiar cómo funciona el programa, y cambiarlo para que haga lo que usted quiera. El acceso al código fuente es una condición necesaria para ello.
- **Libertad 2:** La libertad de redistribuir copias para ayudar a su prójimo.
- **Libertad 3:** La libertad de distribuir copias de sus versiones modificadas a terceros. Esto le permite ofrecer a toda la comunidad la oportunidad de beneficiarse de las modificaciones. El acceso al código fuente es una condición necesaria para ello.

El software libre se asocia regularmente con el concepto de *código abierto* (del ingles o*pen source*), la definición de software de código abierto viene dada por los lineamientos de la *Open Source Initiative (Iniciativa de Código Abierto)* y se expresa en la *Open Source Definition*[2], que son diez condiciones bajo las cuales se debe distribuir un programa de computadora para ser etiquetado como de código abierto. Desde el punto de vista técnico casi todo el software considerado de código abierto otorga las cuatro libertades definidas por la *FSF* y es por lo tanto también software libre, por lo que muchas veces los términos se utilizan como sinónimos; en ingles se suele recurrir al acrónimo *FOSS* de *Free and Open Source Software (Software Libre y de Código Abierto)*. Sin embargo es importante mencionar que a pesar de que para efectos de este proyecto los conceptos son equivalentes, existen detrás de ellos movimientos que defienden diferentes filosofías, el movimiento del software libre reivindica aspectos morales, políticos y éticos en favor del software libre, mientras que la *Open Source Initiative* pone énfasis en los beneficios prácticos y comerciales de la utilización del software de código abierto.

En el resto del documento se dará preferencia al término software libre, y a la definición de la FSF, sin embargo se han tenido en cuenta las ventajas comerciales y técnicas derivadas del uso del software libre mencionadas por la OSI.

1.3.2 Ventajas del Software Libre

Preferir software libre sobre software no libre no es una decisión arbitraria, a la pregunta de cuales son las ventajas del software libre, *Richard Stallman* responde que el software no libre es sencillamente malo porque restringe la libertad del usuario, y por lo tanto el software que otorga las mayores libertades será siempre mejor (*The Advantages of free software*, http://www.gnu.org/philosophy/practical.html).

1 *véase http://www.gnu.org/philosophy/free-sw.es.htm*
2 *véase http://opensource.org/osd*

Si bien el argumento anterior es muy válido, existen además varios beneficios concretos del software libre y de código abierto sobre el software no libre o privativo,a continuación presentamos algunos de ellos tomados de dos publicaciones acreditadas en Internet[1] :

- **Seguridad:** El software libre permite ver y ejecutar el código de la aplicación al público en general y no sólo a los creadores del programa, y entre más personas analizan y realizan pruebas al código existen más probabilidades de encontrar errores y repararlos. En general el software libre es más seguro porque la auditoria del código puede llevarse a cabo por personas independientes del creador original.
- **Calidad:** Muchos proyectos de software involucran cientos cuando no miles de desarrolladores de software, lo cual acelerar el proceso de innovación, además el software libre permite que el usuario final se involucre en el desarrollo del producto si tiene los conocimientos necesarios, o en su defecto contrate a alguien que los tenga, esto regularmente implica una mayor satisfacción con el producto final.
- **Independencia del Proveedor (Lock-in):** El software libre reduce la dependencia del proveedor, ya que los estándares abiertos utilizados por mayoría del software libre facilitan el cambio de un software o proveedor a otro.
- **Reducción de Costos:** Si bien software libre y de código abierto no necesariamente significa software gratuito, es cierto que en muchos casos el software libre tiene un menor costo de producción, y que el usuario no tiene que pagar una licencia de uso por él.

La anterior no es una lista exhaustiva de las ventajas del software libre ya que existen otras ventajas que no fueron mencionadas. También es importante resaltar que algunas beneficios sólo aparecen cuando el software logra popularizarse y conseguir una comunidad de creadores y usuarios importante.

Las ventajas mencionadas hasta ahora no son benéficas solamente para los usuarios finales sino también para los productores de software quienes también de la comercialización y uso de software libre.

Una de las posibilidades que tienen las empresas para aprovechar las características del software libre es la creación compartida de productos secundarios. Las empresas tienen que utilizar y muchas veces crear software que no comercializan directamente, sino que sirve como herramienta para la producción del software que fundamenta su lógica de negocios, tal es el caso de software administrativo, de gestión de proyectos, *bugtracking*, base de datos, gestión de personal, etc. Este tipo de software se utiliza de manera similar en otras empresas que tienen iguales o parecidas necesidades. El software libre facilita el desarrollo conjunto de este tipo de herramientas entre todos aquellos que se benefician de su utilización y disminuye los costos de producción al repartirlos entre los involucrados.

1 Vease “10 Reasons Open Source is good for Business”, http://www.pcworld.com/article/209891/10_reasons_open_source_is_good_for_business.html y “Top10 Benefits of using Free Software, http://www.edutopia.org/blog/benefits-free-software-shahzad-saeed

1.3.3 El Proyecto y Los Modelos de Negocios en el Software Libre

El software libre también puede ser parte de la lógica de negocios de una empresa como producto central, y se pueden obtener ingresos directamente por su comercialización. Para ello existen varios modelos de negocio basados en software libre, en estos esquemas generalmente no se obtienen ganancias por la venta de licencias de uso, como en el caso del software privativo, sino que se comercializan servicios y productos asociados al software, algunos de estos servicios pueden ser:

- Contratos de soporte
- Programas de entrenamiento y certificación
- Venta de libros y documentación especializada
- Comercialización de un producto con licencia privativa como complemento o extensión de un programa de software libre.
- Programas de licencia dual, una libre y otra no libre.

Para el caso particular de este proyecto, el desarrollo es apoyado y en parte financiado por una empresa cuyo modelo de negocios esta totalmente basado en software privativo y que accedió a apoyar un proyecto basado en software libre dentro del contexto de un acuerdo académico entre el creador y la empresa.

Por lo anterior el proyecto no cuenta con estructura previa ni es parte de un plan de negocios propio del software libre, sin embargo se intenta abrir camino y plantear la posibilidad de que en el futuro la empresa pueda explotar más un modelo de negocios parcial o totalmente basado en software libre.

El objetivo mínimo en este caso es lograr que el módulo llegue a los clientes y se instale como complemento al producto principal. Podría esperarse que eventualmente los clientes aprovechen las libertades que les otorga el software libre y desarrollen sus propias extensiones basados en el código del módulo, y que después distribuyan estas extensiones de tal manera que exista un beneficio tanto para otros usuarios como para la empresa que podría integrar estos componentes dentro de su paquete de soluciones.

1.3.4 Propiedad Intelectual y Licencia del Proyecto

Debido a un acuerdo entre el creador, en este caso estudiante de la *UOC* que realiza el proyecto y la empresa donde se realiza la práctica, los derechos patrimoniales o de explotación del producto final pertenecen en su totalidad a la empresa. Este acuerdo sin embargo estipula que el creador tiene la libertad de distribuir el código de la aplicación bajo una o varias licencias de software libre y código abierto.

1.4 Aplicaciones Web con HTML5,PHP y JavaScript

El producto final de este proyecto es una aplicación web basada en *PHP,HTML y JavaScript*, tres tecnologías usadas en la mayoría de los proyectos de aplicaciones para Internet de la actualidad. En esta sección se presenta una breve introducción a algunos de los conceptos más relevantes de estas tecnologías en relación con el trabajo de este proyecto.

1.4.1 HTML5 y Canvas

HTML5 es la quinta revisión hecha por la *W3C(World Wide Web Consortium)* del lenguaje básico de la *World Wide Web*, cuenta con nuevos elementos y atributos que intentan responder a las necesidades de uso actuales de los sitios web. Uno de los objetivos implícitos en *HTML5* es buscar lo que se ha llamado la web semántica, es decir que las etiquetas que se utilizan como marcadores para el contenido sean capaces de brindar una información mínima sobre la naturaleza del contenido, y que esta información tenga algún significado en el lenguaje común humano. Este estándar aun no esta completo y su proceso de desarrollo continua, sin embargo la mayoría de los navegadores actuales son compatibles con varias de sus funciones.

Una de las nuevas características en *HTML5* y que se utilizó para el desarrollo del informe HTML fue el elemento <*canvas*>. *Canvas* se utiliza para dibujar gráficos sobre la marcha directamente en la página web, es solamente un contenedor y se tiene que utilizar junto con algún script para obtener los gráficos finales. Para el informe se usó la librería de *JavaScript jqPlot*[1] que utiliza *canvas,* para dibujar diagramas que representan los resultados de los exámenes.

1.4.2 JavaScript y Jquery

En las *aplicaciones de Internet enriquecidas* o *RIA (Rich Internet Applications)* se usa a menudo *JavaScript* para modificar el contenido de las páginas de forma dinámica y facilitar la interacción con el usuario. *JavaScript* es un lenguaje de programación interpretado que en el caso de las aplicaciones de Internet se almacena en el servidor y se enviá como parte de los contenidos de una página junto con el código *html para* ejecutarse en el navegador del cliente.

Jquery[2] es una biblioteca de software libre para *JavaScript* con la que se escribieron todos los *scripts* que se ejecutan en el navegador para el informe HTML. *Jquery* simplifica el trabajo de manipular el *DOM (Document Object Model)* y manejar eventos, lo cual agilizó el desarrollo de las funciones.

1.4.3 PHP, MVC y Zend Framework

PHP es un lenguaje script que se utiliza ampliamente para el desarrollo de aplicaciones web y su código se almacena del lado del servidor, es muy común encontrarlo en ambientes de trabajo junto con el servidor web *Apache*, para este proyecto todos los scripts del lado del servidor se programaron en *php*, y el *plugin* opera tanto con *Apache* como con el servidor *IIS*.

Para programar en *php* existen múltiples *frameworks* que buscan estandarizar la estructura del código, proporcionar librerías para las tareas más comunes,mejorar la seguridad de la aplicación, etc. *EvaExam* se programa utilizando el *framework* Zend[3] que se basa en el patrón de arquitectura de software conocido como *Modelo – Vista – Controlador o MVC. Zend Framework* se distribuye bajo la nueva licencia BSD y es por lo tanto software libre.

Puesto que el proyecto crea un *plugin* para *EvaExam* se decidió que lo más adecuado era también utilizar *Zend* como herramienta de desarrollo y basar la arquitectura de la aplicación el modelo MVC. De esta manera se facilitaba la integración del *plugin* y se mantenía un coherencia con la arquitectura del producto central. Además como la nueva licencia BSD es una licencia *FOSS* se adaptó de manera perfecta con la naturaleza del proyecto.

1 http://www.jqplot.com/
2 http://jquery.com/
3 http://framework.zend.com/

2 Metodología y Planificación del Proyecto

En esta sección se presenta de manera general la planificación del proyecto y la metodología que se utilizó para realizarlo. El ciclo de desarrollo del programa siguió el modelo en cascada. En este modelo el proyecto progresa siguiendo una secuencia ordenada de pasos, que van desde el concepto inicial del software hasta la validación del producto. Las fases que se tienen que llevar a cabo se muestran en la figura 2.1.

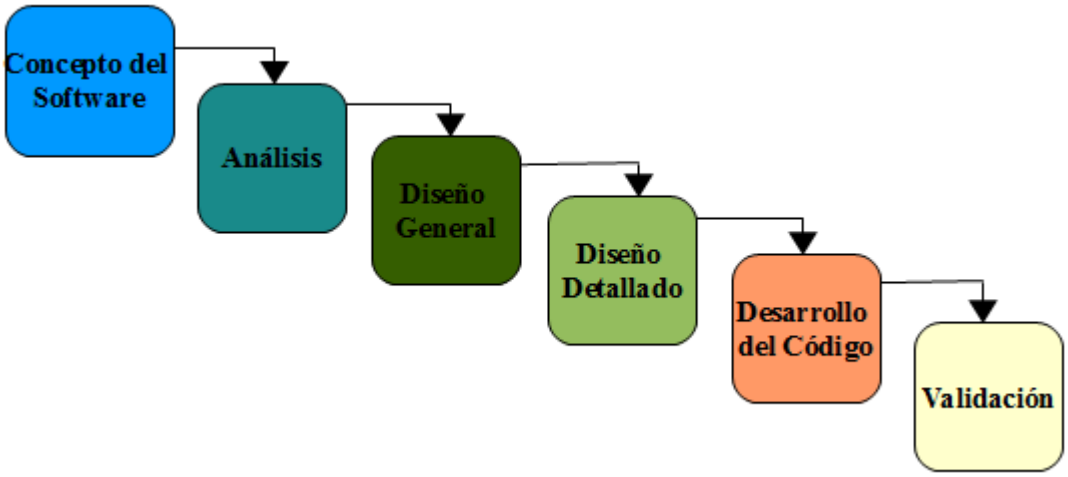


Figura 2.1: Fases del modelo en cascada

Para la realización del proyecto se consideraron un total de 364 horas en un periodo que cubriría prácticamente todo el año 2013. El proyecto se dividió en diferentes etapa, típicas de los proyectos de desarrollo de software, la distribución de horas por etapa y actividad que se planeó es la siguiente:

Número de horas por etapa	
Definición general del proyecto	10
Análisis	24
Diseño	40
Desarrollo	110
Test	140
Documentación	40
Total de horas:	**364**

Tabla 2.1: Distribución de horas por fase del proyecto

No.	Actividad	Horas
1	*Definición general del proyecto*	10
2	*Análisis de Los Requisitos*	24
3	*Diseño*	40
4	*Planes de prueba*	10
5	*Desarrollo del código y pruebas unitarias*	80
6	*Pruebas de Validación del Producto*	130
7	*Corrección de errores*	30
8	*Documentación*	40

Tabla 2.2: Horas por actividad

2.1 Hitos del Proyecto

Para llevar un control de la correcta realización del proyecto se establecieron una serie de hitos. Las figuras muestran los hitos del proyecto y su ubicación temporal respecto de las etapas, en ingles tal como se definieron para la empresa, además se se incluye una breve descripción del significado de cada hito.

Figura 2.2: Etapas e hitos del proyecto

Milestone	Deadline	Status
✓ General Definition of the Project	15/02/2013	- Open
✓ Requirements Definition	19/04/2013	- Open
✓ Requirements Analysis	30/04/2013	- Open
✓ General Concept	15/05/2013	- Open
✓ Fine Concept	30/05/203	- Open
✓ Test Plan	30/06/2013	- Open
✓ Feature Freeze	30/07/2013	- Open
✓ Feature Complete	08/08/2013	- Open
✓ Release Candidate	10/10/2013	- Open
✓ Release	01/11/2013	- Open

Figura 2.3: Hitos del Proyecto

- General Definition of the Project: Bosquejo del concepto del software.
- Requirements Definition: Definición general de los requisitos del proyecto.
- Requirement Analysis: Definición detallada de los requisitos del proyecto.
- General Concept: Diseño general del sistema.
- Fine Concept: Diseño detallado del sistema.
- Test Plan: Especificación del plan de pruebas.
- Feauture Freeze: Punto a partir del cual deja de agregar funciones, paquetes e interfaces a la definición del software.
- Feature Complete: Punto en el que todas las funciones del software han sido implementadas.
- Release Candidate: Versión del software candidata a convertirse en la versión final.
- Release: Liberación del software al usuario final.

3 Estudio de Viabilidad

En este capítulo se presenta el análisis que se llevó a cabo para determinar cual de las alternativas de solución planteadas originalmente era la más viable. Para ello se consideraron los aspectos técnicos, económicos y legales de cada una de ellas.

3.1 Estado Actual del Sistema

El primer paso en esta etapa es determinar el estado actual del sistema,este análisis es necesario para definir los cambios que se necesitan realizar para cumplir las nuevas especificaciones.

La versión estable más reciente de *EvaExam* es la 6.0 2000, en esta versión los resultados de las pruebas de *EvaExam* se muestran en dos tipos de informes: el informe *PDF* y el informe *CSV*.

El primero es como lo indica su nombre un documento en formato *PDF* que se genera de forma dinámica a partir de los resultados de las pruebas, el informe se divide en las siguientes secciones:

- Análisis de Preguntas: Es un análisis de los resultados para cada pregunta, que se calculan a partir del número de veces que una opción de respuesta fue escogida respecto del número total de respuestas dadas, se presenta utilizando gráficas de barras y tablas.
- Informe de Examen: Es una tabla que contiene los números de identificación de los estudiantes, los puntos que obtuvo el estudiante en cada pregunta y los puntos que obtuvo en total en el examen.
- Informe General: Contiene estadísticas generales como son mediana, promedio y desviación estándar; calculadas a partir de los los resultados de todos los participantes.
- Informe Resumido: Es una tabla que contiene el número de identificación de cada estudiante, el número de puntos que obtuvo en la prueba, y si para el examen se definió un criterio de asignación de notas, también contiene la nota que el estudiante obtuvo y una indicación de si aprobó o reprobó el examen.

El informe *csv* es un archivo en este formato que contiene los datos en bruto de la prueba, cada entrada en el archivo representa el resultado de uno de los participantes en el examen. Los resultados incluyen los puntos obtenidos en cada pregunta, los puntos en total, la nota obtenida y una indicación de aprobación o reprobación. El informe se utiliza principalmente cuando los resultados se quieren procesar fuera de *EvaExam*.

Desde un principio se sabía que el estado actual de sistema no satisfacía las nuevos requerimientos, sin embargo se determinó que que el proyecto no sustituiría la solución existente sino que la extendería para responder a las nuevas necesidades. Así se planteó no alterar el informe *PDF* ni el informe *CSV* sino añadir una tercera forma de presentar los resultados que incorporará los nuevos requerimientos.

3.2 Requisitos del Proyecto

Una vez que se cuenta con un diagnostico del estado actual del sistema se procede a establecer los requisitos del proyecto. Para esto se comenzó con reuniones con las personas involucradas en el desarrollo del producto, donde se recogieron las ideas de cada uno sobre las características que de debería tener la nueva solución. Tuvieron especial importancia en esta etapa las ideas aportadas por

el departamento de entrenamiento y por el departamento de soporte, ya que el personal de estos departamentos tiene contacto directo y continuo con los usuarios de *EvaExam* por lo que conoce cuales son las peticiones más comunes de los clientes.

Después de las reuniones con los involucrados se contaba con un documento que contenía las ideas para la nueva solución en forma de sentencias generales, el siguiente paso fue clasificar esas ideas por prioridad y tipo.

En este primer análisis los requerimientos se clasificaron en funcionales y no funcionales y se ordenaron por prioridad, donde 1 es la prioridad más alta y 4 la más baja, esta clasificación se presenta a continuación.

Requisitos Funcionales		
Prioridad	ID	Requisito
1	F1	El informe debe presentar una lista de los participantes en el examen así como los puntos en total obtenidos en el mismo.
1	F2	De existir un sistema de asignación de notas y un criterio de aprobación, el informe debe incluir la nota obtenida por cada estudiante así como si aprobó o no el examen.
1	F3	El informe debe incluir en su encabezado el nombre del instructor o examinador, el nombre del examen y el formato del mismo.
1	F4	El informe debe incluir un gráfico de distribución de las notas obtenidas.
1	F5	El informe debe ser compatible con la versión *EvaExam* 6.0 2000.
1	F6	El informe debe presentar los resultados en una interfaz moderna en formato HTML.
2	F7	El informe debe incluir los puntos obtenidos por cada participante en cada pregunta.
2	F8	El informe debe incluir una sección donde se presenten las estadística generales sobre las respuestas dadas a cada pregunta.
2	F9	El informe debe permitir al usuario filtrar los resultados del examen a partir de distintos criterios como pueden ser la nota obtenida,si el estudiante aprobó o no el examen, los puntos obtenidos, etc.
3	F10	El examinador debe poder crear un informe resumido basado en dos o más exámenes.
4	F11	Debe ser posible acceder a los detalles de las respuestas dadas por cada estudiante.
4	F12	Debe existir una presentación alternativa del informe donde se presenten de forma detallada los resultados individuales de cada estudiante.

Table 3.1: Requisitos Funcionales

Requisitos No Funcionales		
Prioridad	ID	Requisito
1	NF1	El informe no pude utilizar ninguna de las librerías internas o en general código de *EvaExam*.
1	NF2	El módulo debe comunicarse por medio de alguna interfaz pública de tal manera que se pueda instalar como un *plugin*.
1	NF3	El código del informe se debe distribuir bajo una licencia de software libre.
1	NF4	El informe debe ser accesible sólo para usuarios registrados en *EvaExam.*
2	NF5	La integración del módulo debe seguir los criterios definidos para proyectos externos de *EvaExam*.
2	NF6	Los colores elegidos para la interfaz deben ser coherentes con los usados en EvaExam para no causar la apariencia de que se trata de productos distintos.
2	NF7	El informe debe contar con una interfaz que pueda traducirse a varios idiomas de forma sencilla.
3	NF8	El sistema debe utilizar *Zend Framework* como plataforma de desarrollo.

Table 3.2: Requisitos no funcionales

Después de este primer análisis se contaba con una descripción muy general del sistema, se definieron 12 requisitos funcionales y 8 no funcionales. Como se puede observar se trata sentencias cortas que aun no son suficientes para construir el diseño de la nueva solución pero que nos sirven como punto de partida. En las siguientes etapas estos requisitos se irán dividiendo en requisitos más específicos, clasificando y algunos de ellos se descartarán por no ser viables.

3.3 Análisis de las Alternativas de Solución

Al hacer un análisis de los requisitos del sistema se plantearon dos alternativas de solución: utilizar un programa de generación de informes ya existente o crear un módulo propio específico para *EvaExam*. Para la primera opción se pensó en tomar el conjunto de herramientas de *reporting Jasper Report*[1] y construir la interfaz necesaria para integrar dicha aplicación con *EvaExam*. La segunda alternativa implicaba desarrollar enteramente una solución por cuenta propia..

Para determinar cual de los dos alternativas de solución era la más adecuada se realizó un análisis comparativo de las ventajas y desventajas de ambas. A continuación se presentan los detalles de este análisis, en él que se consideraron aspectos técnicos, económicos y legales.

3.3.1 Alternativa 1 : Jasper Report Library

Jasper Report es un conjunto de librerías y componentes, distribuidos bajo la licencia *LGPL* y programados en *Java*, para la creación de informes en distintos formatos. Cuenta con un editor

1 http://community.jaspersoft.com/project/jasperreports-library

llamado i*Report* con el que se puede definir la estructura y apariencia del informe.

Las características que se tomaron en cuenta para considerar a *JasperReport* como una posible alternativa de solución fueron las siguientes:

- El programa se distribuye bajo una licencia de software libre.
- Es posible generar informes en distintos formatos como *HTML,PDF,XML, Microsoft Excel*, entre otros.
- Los datos del informe pueden tomarse desde varias fuentes como son archivos estáticos, bases de datos, interfaces soap, etc.
- El diseño del informe se puede personalizar usando *iDesigner*.
- Cuenta con una comunidad de usuarios muy grande y dispone de amplia documentación tanto a nivel de usuario como sobre el código fuente.
- Se puede usar como aplicación web por medio de *Java y* un servidor de aplicaciones.

3.3.2 Alternativa 2: Desarrollo de una aplicación por cuenta propia.

Esta opción ofrecía la ventaja de poder construir un *plugin* a la medida de los requisitos del proyecto, teniendo control total sobre las características técnicas de la aplicación. Por otro lado probablemente implicaba un esfuerzo mayor de desarrollo que la alternativa 1.

3.3.3 Análisis de Los Aspectos Técnicos

Si se escogía como solución *Jasper Report* el esfuerzo de desarrollo se concentraría en construir la interfaz entre *EvaExam y Jasper Report* para alimentar el informe con los datos provenientes de *EvaExam*. La función de crear el informe en los distintos formatos sería realizada por las librerías de *Jasper Report.*

En el caso de la solución a la medida las tareas incluían tanto el desarrollo de la interfaz con *EvaExam* como el diseño y creación del informe. La siguiente es una comparación de las ventajas técnicas de cada una alternativa frente a la otra:

Ventajas de *JasperReport* sobre *EvaExam*

- Menor tiempo necesario para desarrollar el código de la aplicación por tener que crear solamente la interfaz entre los componentes.
- Variedad de formatos de presentación de *Jasper Report* frente al formato único en HTML de la solución a la medida.
- Facilidad de crear varios diseños para la presentación del informe con *JasperReport* haciendo uso de i*Designer*; frente a un único diseño con la solución a la medida.

Ventajas de la solución a la medida sobre *JasperReport*

- Integración fina con *EvaExam*, por poder realizar la programación con *PHP y Zend*, y construir su arquitectura específicamente para acoplarse con *EvaExam*.
- Poder utilizar el servidor web de *EvaExam* como contenedor del código de la aplicación, por el contrario *JasperReport* requiere instalar el servidor de aplicaciones *TomCat* para poder funcionar como aplicación web.

- En el caso de *JasperReport* se requería familiarizarse primero con el uso y programación de las librerías que lo conforman antes de comenzar el desarrollo.

3.3.4 Análisis de Los Aspectos Económicos

En lo económico no se observó diferencia entre ambos proyectos. *JasperReport* se distribuye sin ningún costo, y aunque existe una versión privativa que se comercializa por medio del pago de licencias de uso, no se consideró necesario utilizarla. También se pensó que la documentación disponible de manera gratuita era suficiente para poder realizar el desarrollo de la interfaz y que no sería necesario invertir en entrenamiento o soporte.

El número de horas para desarrollar la aplicación es fijo y está determinado por los requerimientos de la práctica, por lo que independientemente de la solución elegida el tiempo de desarrollo se tiene que ajustar a este requisito, de esta manera el costo total asociado al número de horas invertido es el mismo para cualquier de los dos casos.

3.3.5 Análisis de Los Aspectos Legales

En lo correspondiente al área legal, un requisito indispensable en el caso de tomar una aplicación existente era que esta se distribuyera bajo una licencia de software libre, puesto que después el producto derivado tendría que poder ser distribuido bajo esa misma licencia, o según el caso bajo alguna otra de software libre.

Para el caso de la solución a la medida se había establecido previamente que la empresa como propietaria de los derechos patrimoniales permitiría la distribución del código de la aplicación bajo los términos de una licencia de software libre.

Jasper Report cumple de manera general con los requisitos de licencia definidos anteriormente, pues como se mencionó se distribuye bajo la licencia de software libre *BSD* modificada.. Así en este aspecto no existe una diferencia relevante entre las dos alternativas.

3.3.6 Elección de La Solución.

Ya hemos descrito las características de ambas alternativas de solución y las ventajas de una frente a la otra, el siguiente paso fue decidir, basándose en esta información, cual seria la solución más apropiada.

En este análisis se encontró un factor determinante que llevo a preferir la solución a la medida sobre *JasperReport*. Después de consultar al equipo de soporte se concluyó que los requerimientos técnicos para hacer funcionar *JasperReport* no eran compatibles con el ambiente de trabajo de *EvaExam*. Si bien *JasperReport* puede operar en el mismo servidor que *EvaExam*, requiere del servidor de aplicaciones *TomCat* para funcionar, mientras que *EvaExam* por estar programado en *PHP* funciona con el servidor web *Apache* . Esta condición implicaba que los clientes tendrían que instalar software adicional, y abrir un puerto virtual más, ya que *Apache* operaria utilizando un puerto y *TomCat* otro. El equipo de soporte, que normalmente asiste a los clientes en el proceso de instalación, argumentó que era inviable solicitar a los clientes realizar cambios tan importantes en el ambiente de trabajo únicamente para instalar un *plugin.*

De esta manera se descartó la alternativa de solución uno y se optó por la solución número dos: hacer un desarrollo a la medida de las necesidades del proyecto, esta solución carecía de las posibilidades de personalización del diseño que tenía la primera alternativa y se limitaba a un sólo formato de presentación en *HTML*; sin embargo reducía la complejidad técnica del proyecto y podía

implementarse sin la necesidad de instalar software adicional.

3.3.7 Interfaz con EvaExam

Una vez que se eligió una de las alternativas de solución restaba evaluar y escoger el tipo de interfaz con la que el *plugin* se comunicaría con *EvaExam*. Las comunicación se podía realizar por medio de tres formas:

A) Obtener los datos directamente de la base de datos.

B) Obtener los datos vía la interfaz *SOAP.*

C) Obtener los datos vía la interfaz *Data Access Layer (DAL).*

A continuación describiremos las características y ventajas de cada una de estas opciones.

A) *Base de datos de EvaExam*

En este caso se tomarían los datos necesarios para generar el informe directamente de las tablas de la base de datos que utiliza *EvaExam*.

Ventajas:

- Sencillez técnica: Técnicamente solo sería necesario utilizar una conexión a la base de datos y luego ejecutar las secuencias SQL necesarias para obtener los datos.
- Flexibilidad: Al tener acceso directo a la base de datos se podrían tomar cualquier dato que fuera necesario y se podría agruparlos según las necesidades propias del informe.

Desventajas:

- Poca escalabilidad y estabilidad: Como se dependería directamente de la estructura de la base de datos para una versión concreta, y dicha estructura es susceptible de cambios con cada nueva versión del programa o por la instalación de parches , se corre el riesgo de que estos cambios provoquen fallos en el funcionamiento del *plugin* o que sencillamente lo hagan inoperable.

B) API SOAP

La interfaz soap de *EvaExam* ofrece métodos para ejecutar varias funciones de *EvaExam*, está diseñada precisamente para que aplicaciones externas se comuniquen con *EvaExam* de forma estandarizada.

Ventajas:

- Independiente de la estructura interna del programa y de la base de datos.
- Protocolo estándar para el intercambio de información.
- Amplia documentación para su uso.

Desventajas:

- Posible desempeño deficiente en casos de manejo de gran cantidad de datos.

C) DAL

La interfaz *DAL* (*Data Access Layer)* consiste en una serie de tablas especiales que se crean en la base de datos de *EvaExam*, no forman parte de la estructura por defecto de la misma y recogen los datos más relevantes correspondientes a los resultados del sistema,tomándolos desde otras tablas que contienen esta información. Las tablas que conforman la *DAL* se crean después de la instalación de *EvaExam* y su propósito es ofrecer acceso directo a los resultados pero con independencia de la estructura de la base de datos de *EvaExam.* Para poder instalar la interfaz *DAL* el cliente tiene pagar por una licencia de uso, la cual incluye la garantía de que las tablas serán actualizadas con cada nueva versión para así mantener la compatibilidad entre *EvaExam* y las aplicaciones que dependan de la interfaz *DAL*.

3.3.8 Elección de la interfaz con EvaExam

Al comparar las tres opciones primero se descartó la opción A ya que el equipo de desarrollo de *EvaExam* desaconsejó tomar los datos directamente desde la base de datos, por las desventajas ya mencionadas.

La opción C se descartó porque se consideró que no sería atractivo para el cliente pagar el alto costo de la interfaz *DAL* solo para contar con un *plugin.*

Así la opción B se consideró como una interfaz estable y estándar, probada ya muchas veces en otros proyectos, y se concluyó que los posibles efectos negativos que pudieran aparecer al manejar grandes cantidades de datos se podrían mitigar con un diseño apropiado de la aplicación.

4 Análisis de Los Requisitos

En este capitulo se presenta el análisis de los requisitos del proyecto,para el análisis se toman como base los requisitos definidos en el estudio de viabilidad y la solución seleccionada para el desarrollo del proyecto. El objetivo es conseguir una especificación detallada del proyecto.

Para comenzar es necesario definir de manera general las funciones del *EvaExam HMTL Report Plugin o Informe HTML*, estas funciones se expresan de la siguiente manera:

- Mostrar en el informe los resultados individuales de los participantes.
- Mostrar en el informe los puntos obtenidos en cada pregunta por cada participante.
- Mostrar gráficos que representen la distribución porcentual de las notas obtenidas en el examen.
- Incluir en el informe datos estadísticos de la prueba como son media, mediana, desviación estándar, máximo y mínimo número de puntos alcanzados.
- Implementar diferentes filtros para los datos.

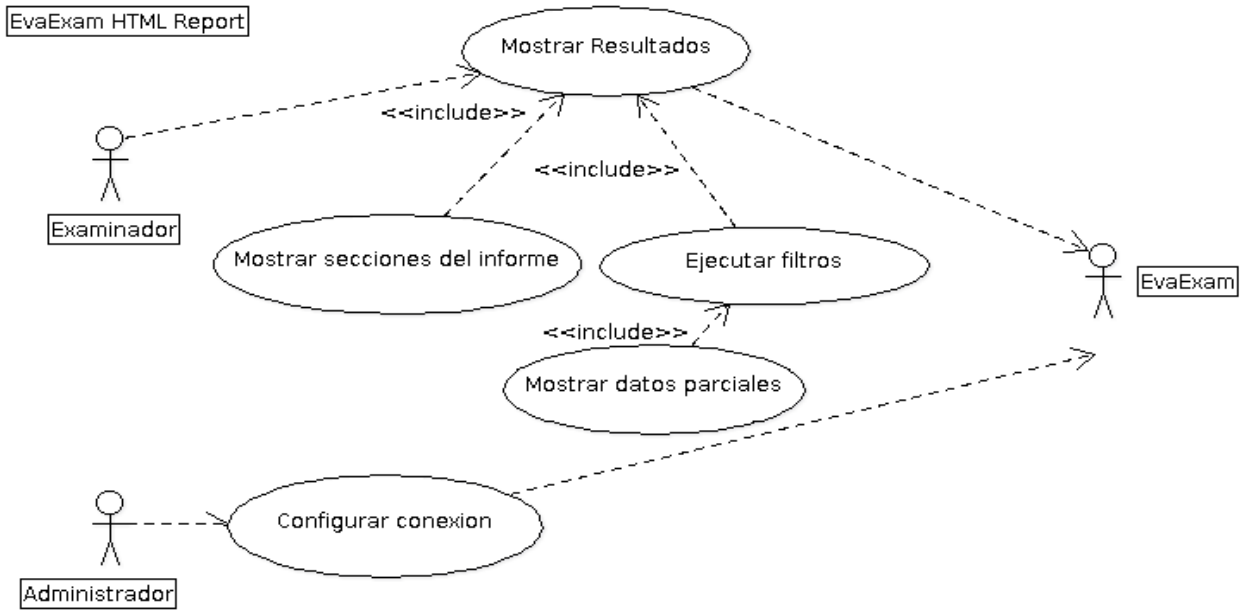


Figura 4.1: Casos de Uso del EHReport Plugin

El diagrama de casos de uso de la figura 4.1 se realizó para representar las funciones del sistema. En el diagrama participan tres actores: el examinador, el administrador y *EvaExam*. La función principal del informe es mostrar los resultados de los exámenes, dividiendo la información en varias secciones. El informe también dispondrá de filtros para generar subgrupos de datos a partir de distintos criterios. El actor principal en este diagrama es el examinador quien abre el informe y ejecuta las funciones, el papel de *EvaExam* es proporcionar la interfaz de acceso al informe y los datos que contendrá. El tercer actor es el administrador de *EvaExam* el cual se encarga de establecer los permisos necesarios que permiten conectar el *plugin* con *EvaExam* por medio de la interfaz *SOAP*.

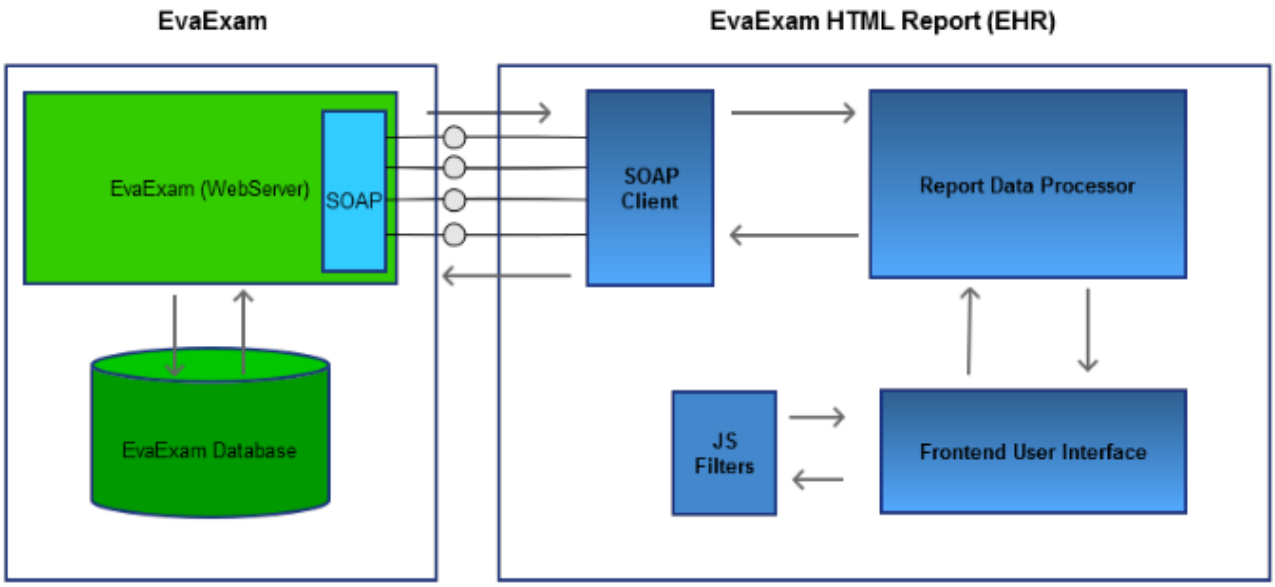


Figura 4.2: Arquitectura General del EHReport Plugin

Una vez que se definieron las funciones fue posible definir un prototipo del sistema. El diagrama de la figura 4.2 muestra un esquema general de los componentes que conforman el *EHReport plugin*, los componentes que podemos identificar son los siguientes:

- **EvaExam**: *EvaExam* proporciona, a través de la interfaz *SOAP*, los datos del informe almacenados en la base de datos. También sirve como interfaz de usuario para acceder al informe.
- **Cliente Soap** (Soap Client): Es el componente del *EHReport plugin* que se conecta directamente a *EvaExam* para obtener los datos del informe por medio de llamadas a los métodos del servidor *soap* de *EvaExam*.
- ***Procesador de Datos del Informe*** *(Report Data Processor):* Este componente recibe la información del cliente *soap* y construye las estructuras de datos que se transfieren a la interfaz del usuario donde obtienen su forma final para presentarse al usuario. También hace los cálculos necesarios para obtener datos derivados, estos datos no los proporciona directamente *EvaExam*, sino que se deben calcular a partir de otros datos contenidos en el informe, tal es el caso de los valores de la mediana, la media y la desviación estándar.
- **Interfaz de Usuario** (Frontend User Interface): Es el informe mismo en formato HTML, tal como lo ve el usuario, presenta los datos utilizando tablas, gráficas y otros elementos. Incluye los controles para ejecutar la funciones de los filtros de datos.
- **Filtros (Filters):** Los filtros del informe se implementan directamente en el navegador del cliente, y operan por lo tanto una vez que el informe ha sido generado, para la programación de estos filtros se utilizó *JavaScript.*

El diagrama de la figura 4.2 presenta la arquitectura general del *EHReport Plugin*, sin embargo no representa la arquitectura interna de la aplicación a nivel de código, con clases, interfaces y otros elementos, esta especificación se realizará en la etapa de diseño. El propósito de diseñar este esquema en esta etapa, es representar la estructura de la aplicación de lo forma más sencilla posible, de tal manera que pueda ser comprendida por todos los involucrados.

4.1 Especificación Detallada de Los Requisitos

Las secciones anteriores mostraron las funciones y estructura del *EHReport plugin* de forma general, en esta sección se tomaron los requisitos establecidos en la etapa de viabilidad y se detallaron y volvieron a clasificar para conseguir la especificación final que se utilizó en la etapa de diseño.

4.1.1 Usuarios del Sistema

Los actores que ya se mencionaron anteriormente se clasificaron en actores físicos y actores del sistema, los primeros corresponden a usuarios reales del sistema, y los segundos a representaciones de componentes virtuales, a continuación se define el rol de cada uno de ellos:

Actores Físicos:

- Examinador: Es la persona que lleva a cabo la evaluación y utiliza el informe para visualizar los resultados.
- Administrador de *EvaExam*: Es el usuario con más privilegios del sistema, y su función es configurar los parámetros de trabajo de *EvaExam*, en el contexto del proyecto debe establecer los permisos para que el cliente SOAP pueda conectarse.

Actores del Sistema:

- Cliente: Corresponde al sistema desde donde el informe se abre, representado en este caso por el navegador web.
- Servidor de *EvaExam:* Es el servidor donde se ejecuta el código de la aplicación.

4.1.2 Ambiente de Trabajo

El ambiente de trabajo en el que operará el *plugin* es el mismo que el de *EvaExam y se* define de la siguiente manera:

- Sistema Operativo: Windows Server 2003 o superior.
- Servidor Web: Microsoft IIS 6.0 o superior, Apache 2.2.
- Base de Datos: Microsoft SQL Server, MySQL.

4.1.3 Limitaciones de Diseño e Implementación

El plugin debe comunicarse con EvaExam exclusivamente por medio de la interfaz SOAP API.

4.1.4 Licencia del Proyecto

El código de la aplicación debe distribuirse bajo los términos de una licencia de software libre, la cual se habrá de definir en la siguientes etapas.

4.1.5 Métodos de La Interfaz Soap Utilizados

En está fase se identificó cuales serian los métodos de la interfaz soap de EvaExam que se utilizarían para obtener la información necesaria para construir el informe HTML. La tabla 4.1 muestra la lista de métodos que se utilizaron en el proyecto y una breve descripción de su uso.

Método	Descripción
GetGradingKey	Devuelve el criterio de asignación de notas
GetExamResults	Devuelve los resultados de los participantes
GetExam	Devuelve los datos del examen
GetAllExamsFolders	Devuelve todas las carpetas que contienen exámenes
GetAllExamsByFolderId	Devuelve todos los exámenes almacenados en una carpeta

Tabla 4.1: Métodos de la interfaz soap utilizados

4.1.6 Lista de Requisitos del Sistema

En esta parte del documento se presenta la lista de requisitos del sistema tal y como se encuentran definidos en el documento de requisitos que se entregó al supervisor del proyecto para su aprobación, junto al nombre del requisito en español se incluye el nombre del requisito en ingles por concordancia con el documento original. Para facilitar el entendimiento y trabajo con los requisitos se crearon nuevas categorías que representan distintas funciones o características del *EHReport Plugin*. Además cuando se consideró que la definición de los requisitos podía ser imprecisa se añadieron secuencias de estimulo-respuesta para ejemplificar el comportamiento esperado de la función descrita.

Clasificación de Los Requisitos

Se establecieron las siguientes categorías que para clasificar a los requisitos, las categorías de funcionales y no funcionales se mantuvieron como categorías generales.

Símbolo	Categoría	Descripción
C	Contenido del Informe	Se refiere a toda la información que se debe mostrar
F	Funciones de interacción con los datos	Funciones que permiten al usuario interaccionar con los datos del informe.
U	Interfaz de usuario	Requisitos de la interfaz gráfica de usuario.
P	Rendimiento	Condiciones de rendimiento que deben cumplirse
I	Interfaz con EvaExam	Interfaz de comunicación con EvaExam
O	Otros	Todos los requisitos no incluidos en otras categorías

Tabla 4.2: Clasificación de los requisitos

Prioridad de los Requisitos

El sistema para clasificar los requisitos basándose en su prioridad se mantiene como en la etapa anterior, sin embargo se construyó la siguiente tabla para especificar el significado de cada una de las categorías.

Prioridad	Significado	Descripción
1	Debe tener	Estos requisitos son esenciales para el proyecto.
2	Debería tener	Estos requisitos no son críticos para el proyecto, pero son casi tan importantes como los de categoría 1, por lo que se debe buscar implementarlos en cualquier caso.
3	Podría tener	Estos requisitos no son críticos, sin embargo si se es posible implementarlos a un bajo costo, incrementarían la satisfacción del cliente.
4	Tendría	Estos requisitos no se implementarán en este proyecto, sino en versiones futuras del producto.

Tabla 4.3: Prioridad de los requisitos

a)Requisitos Funcionales

REQ_001: Informe General de Examen (Exam General Report)
Prioridad: 1 Categoría: C
Descripción:

El informe HTML debe tener una sección que muestre la lista de los participantes y los puntos que cada uno obtuvo en el examen.

REQ_001.1: Visualización de Participantes (Display of large amount of participants)

Prioridad 1: Categoría: C

Descripción:

Se debe incluir una función para mostrar a todos los participantes en distintas páginas si el número de estos es demasiado grande para caber en el área del informe designado para ello, de tal manera que dicha área tenga siempre la misma dimensión.

REQ_001.2: Informe General Extendido (Exam General Report Extended)

Prioridad: 2 Categoría: C

Descripción:

El informe debe incluir una sección donde se muestren los puntos obtenidos en cada pregunta por cada uno de los participantes en el examen.

REQ_001.2.1: Visualización de preguntas (Display of large amount of questions)

Prioridad: 2 Categoría: C

Descripción:

Si el número de preguntas es demasiado grande para el área del informe designada para mostrarlas,se debe incluir una función para mostrar las preguntas en distintas páginas, de modo que el área designada mantenga siempre la misma dimensión.

REQ_001.3: Mostrar nota para cada participante (Display grade for each participant)
Prioridad 1: Categoría C

Descripción:

Si se definió un criterio para asignación de notas para el examen, el informe debe mostrar la nota obtenida por cada participante, así como si este aprobó o no el examen.

REQ_002: Sección de estadísticas generales (General Statistics Section)

Prioridad: 2 Categoría: C

Descripción:

El informe debe tener una sección donde se muestren los siguientes datos:

- Número de participantes en el examen.
- Número máximo de puntos alcanzables.
- Media de los puntos obtenidos.
- Número máximo de puntos obtenidos por un participante.
- Número mínimo de puntos obtenidos por un participante.
- Desviación estándar.

REQ_003: Informe de Notas (Grade Note Report)

Prioridad: 1 Categoría: C

Descripción:

Si para el examen se definió un criterio de asignación de notas,debe existir una sección que muestre la siguiente información:

- Por cada nota, el porcentaje mínimo del total de puntos en el examen que es necesario para obtener la nota en cuestión.
- Por cada nota, el número total de participantes que obtuvieron la misma.
- Por cada nota, el porcentaje de participantes que obtuvieron la misma respecto del total.
- Una indicación de cual es la nota mínima aprobatoria.
- En una tabla se debe presentar el valor porcentual y absoluto de participantes que aprobaron el examen, así como de los que no lo aprobaron.

REQ_003.1: Gráfica Aprobado-Reprobado (Passed Failed Graphic)

Prioridad 1 Categoría: C

Descripción:

El informe debe mostrar por medio de un gráfico circular, la proporción de participantes que aprobaron y reprobaron el examen. La proporción se debe indicar en términos absolutos y porcentuales. La proporción de participantes aprobados se debe representar en el gráfico con el color verde y la de participantes reprobados con color rojo.

REQ_003.2: Gráfica de Notas (Note-Grade Graphic)

Prioridad 1 Categoría : C

Descripción:

EL informe debe mostrar por medio de un gráfico circular la proporción de estudiantes que obtuvieron cada una de las notas posibles para un examen, este valor se debe indicar en términos absolutos y porcentuales. Cada una de las notas en el gráfico debe dibujarse con un color distinto al de las otras.

REQ_003.3: Visibilidad del Informe de Notas (Visibility of Grade Note Report)

Prioridad 1 Categoría: C

Si el examen no tiene asignado un sistema de asignación de notas, la sección definida en el requisito REQ_003 no se debe mostrar.

REQ_004: Informe del Participante (Student Report)

Prioridad 4 Categoría: C

Descripción:

Sebe mostrar un informe detallado de los resultados de cada participante, donde se incluya las respuestas dadas a cada pregunta, las respuestas que fueron correctas e incorrectas, los puntos obtenidos y la nota final.

REQ_004.1 : Respuestas datas y respuestas correctas (Answers given and correct answers)
Prioridad 4 Categoría: C

Descripción:

EL informe debe mostrar las respuestas correctas para cada una de las preguntas del examen. Para este requisito se debe considerar que las respuestas a preguntas abiertas no mostrarán ninguna información.

REQ_005: Análisis estándar de preguntas (Standard Item Analysis)

Prioridad 4 Categoría: C

Descripción:

El EHReport plugin debe mostrar un análisis de los resultados globales del examen para cada pregunta.

REQ_006: Función de búsqueda del informe del participante (Student Report Search Function)

Prioridad 3 Categoría: F

Descripción:

Se debe contar con una función que permita buscar a un estudiante en particular entre todos los participantes de exámenes y generar el informe del participante (REQ_005) para cada uno de los exámenes en que participó.

REQ_007: Filtro aprobado/reprobado (Passed/failed filter)

Prioridad 1 Categoría: F

Descripción:

El informe debe proporcionar una función para filtrar los resultados de los participantes a partir de del criterio aprobado/reprobado.

Secuencia Estimulo-Respuesta:

Condición previa: Debe existir un examen que tenga asignado un criterio de asignación de notas.

1.El usuario selecciona el criterio para el filtro entre dos posibles opciones: participantes aprobados o participantes reprobados.

2.El usuario ejecuta el filtro..

Resultado: En los resultados de los participantes se muestran sólo aquellos que aprobaron o reprobaron según haya sido el criterio elegido por el usuario.

Manejo de errores: Si el resultado de la ejecución del filtro no incluye a ningún participante, se debe informar al usuario de que no se encontró ningún resultado para la búsqueda.

REQ_08: Filtro mayor que (Filter more than)

Prioridad 4 Categoría: F

Descripción:

Se debe contar con una función para filtrar los resultados de los participantes a partir del número de puntos que obtuvieron, el usuario proporciona un número de puntos y después de la ejecución del filtro se muestran todos lo participantes que obtuvieron más puntos que el valor dado por el usuario.

REQ_09: Filtro menor que (Filter less than)

Prioridad 4 Categoría: F

Descripción:

Se debe contar con una función para filtrar los resultados de los participantes a partir del número de puntos que obtuvieron, el usuario proporciona un número de puntos y después de la ejecución del filtro se muestran todos lo participantes que obtuvieron menos puntos que el valor dado por el usuario.

REQ_010: Filtro por nota (Filter by note)

Prioridad 3 Categoría: F

Descripción:

El informe debe proporcionar una función para filtrar los resultados de los participantes a partir de la nota que obtuvieron en el examen.

Secuencia Estimulo-Respuesta:

Condición previa: Debe existir un examen que tenga asignado un criterio de asignación de notas.

1.El usuario selecciona una de las notas posibles en el examen.

2.El usuario ejecuta el filtro..

Resultado: Se muestran los resultados sólo para aquellos participantes que obtuvieron la nota elegida por el usuario.

Manejo de errores: Si el resultado de la ejecución del filtro no incluye a ningún participante, se debe informar al usuario de que no se encontró ningún resultado para la búsqueda.

REQ_011: Informe Resumido (Summary Reports)

Prioridad 4 Categoría: F

Descripción:

Se debe contar con una función para generar informes resumidos a partir de los resultados de dos o mas exámenes que utilicen el mismo formulario.

REQ_012: Página inicial (Initial Page)

Prioridad 1 Categoría: O

Descripción:

La primer sección que se debe mostrarse en el informe es la definida en el requisito REQ_001.

REQ_012.1 Encabezado del informe (Report Header)

Prioridad 1 Categoría : C

Descripción:

El encabezado del informe debe contener la siguiente información:el nombre del examinador, el nombre del examen, el nombre del formulario de examen y la tasa de respuesta.

REQ_013 : EvaExam Access GUI

Prioridad 1 Categoría: U

Descripción:

Debe existir un enlace en el menú izquierdo de la pantalla de bienvenida del examinador, este enlace debe mostrar la lista de exámenes con resultados disponibles, esta lista debe contener un enlace para abrir el informe HMTL correspondiente.

b) Requisitos no funcionales

REQ_014: Integración como proyecto PSE (Integration as PSE Project)

Prioridad 1 Categoría:I

Descripción:

La integración con *EvaExam* debe seguir el procedimiento establecido para proyectos tipo *PSE (Profesional Services).*

REQ_015: Interfaz SOAP (SOAP Interface)

Prioridad 1 Categoría: I

Descripción:

La comunicación con *EvaExam* debe realizarse únicamente por medio de la interfaz *SOAP*, y en ningún caso utilizando conexiones directas a la base de datos, o funciones internas de *EvaExam*.

REQ_016: Seguridad del acceso a traves de EvaExam (Evaexam Security Access)

Prioridad 1 Categoría: O

Descripción:

El informe debe ser accesible solamente para usuarios de cuentas de examinador, y se debe verificar la validez del código de sesión (*php session*).

REQ_017: Marcas EvaExam y ClassExam (EvaExa/ClassExam branding)

Prioridad 1 Categoría: O

El informe debe incorporar los logos y otros elementos distintivos de marca acorde al derivado del producto, ya sea EvaExam o ClassExam*.

REQ_018 : Idioma (Language)

Prioridad 1 Categoría: O

Descripción:

El informe debe tener una interfaz con textos en ingles, y debe proporcionar un método para incluir traducciones adicionales que no implique la alteración directa del código de la aplicación, por ejemplo archivos de texto para cada idioma, donde se definan el texto a mostrar en el informe.

REQ_019 : Colores en la interfaz de usuario (Colors and User Interface)

Prioridad 1 Categoría: U

Descripción:

La apariencia general del informe, colores y diseño debe ser coherente con la interfaz de *EvaExam* en su versión 6.0 2000.

REQ_020: Compatibilidad con navegadores (Browser Support)

Prioridad 1 Categoría: O

Descripción:

El informe debe ser compatible y mostrarse de igual manera en los siguientes navegadores: IE8+, FireFox 3.5+,Safari4+ ,Chrome 28+ y Opera 12+.

REQ_021: Logging Errors

Prioridad 1 Categoría: O

Descripción:

Todos los errores generados deben registrarse en un archivo de texto llamado “EHReport.log”, que debe ubicarse en el directorio root/evaexam/data/logs.

* *ClassExam* es el nombre con el que *EvaExam* se distribuye en los Estados Unidos, la interfaz de usuario difiere ligeramente en colores y logotipos.

4.2 Interfaces del Usuario

En la definición de requisitos se especificaron algunos características de la interfaz de usuario, sin embargo para lograr una especificación más completa se realizaron bosquejos de la apariencia que debería tener la interfaz. Los bosquejos presentados en esta sección no representan estrictamente el aspecto final de la interfaz, sin embargo sirvieron para tener una idea de las posibilidades de diseño.

4.2.1 Interfaz del *EHR Plugin* con *EvaExam*

El requisito *REQ_014* establece que la integración del *plugin* en *EvaExam* se debía realizar siguiendo el formato de proyectos *PSE*, este formato especifica que un *plugin* o extensión para el programa sólo puede insertar enlaces en el menú izquierdo de la pantalla principal del administrador o examinador según sea el caso.

Exam Name	Exam Id	Exam Count	HTML Report
Sample Exam 1	12	20	open(link)
Sample Exam 2	13	22	open(link) ...
Sample Exam 3	14	30	open(link) ...
Sample Exam 4	15	50	open(link)

Figura 4.3: Lista de Exámenes

De acuerdo a lo anterior el enlace que muestra la lista de exámenes con resultados disponibles *(REQ_013)* debe insertarse en el menú izquierdo del examinador. Este enlace representa el punto de entrada a las funciones del *plugin*. Cuando el usuario haga clic en este enlace se debe mostrar una lista con todos los exámenes con resultados disponibles para el examinador en cuestión. El diseño de como se vería esta lista se muestra en la figura 4.3.

4.2.2 Secciones del Informe HTML

El diagrama de la figura 4.4 muestra el bosquejo que se realizó de la vista principal del informe. En la figura es posible identificar las partes principales del informe: El encabezado donde se muestra la información del examen, un menú lateral que contiene enlaces a las distintas secciones del informe y una tabla donde se muestran los resultados del examen.

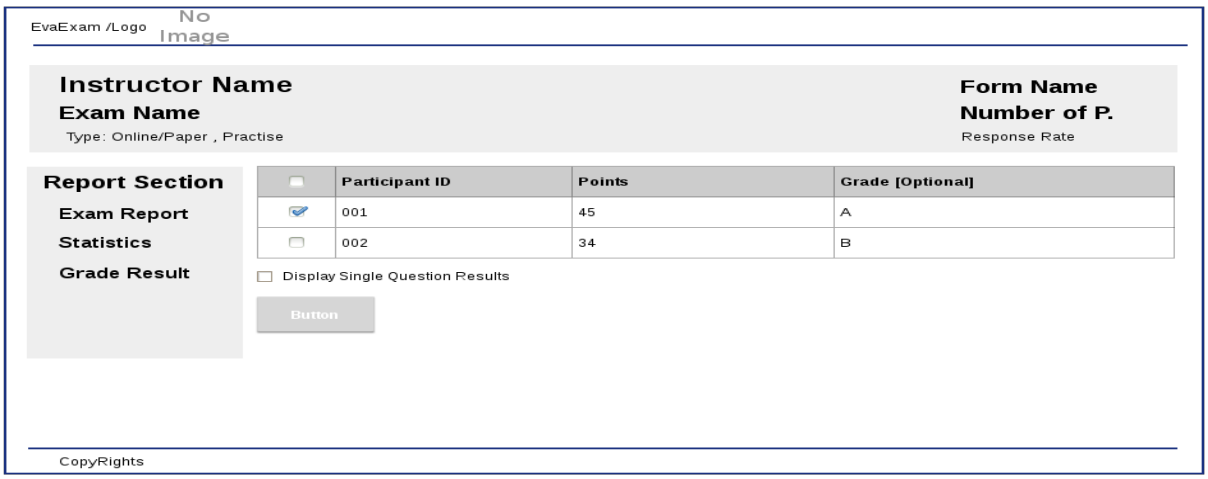


Figura 4.4: Vista principal del informe

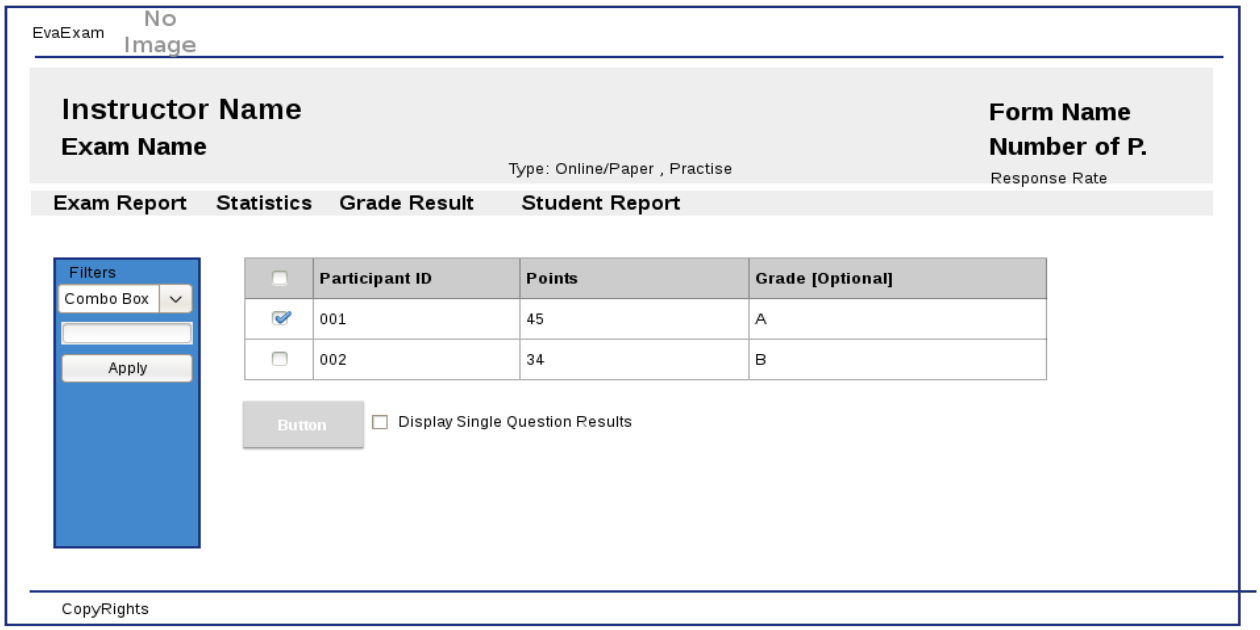


Figura 4.5: Vista principal (versión alternativa)

La figura 4.5 muestra un esquema alternativo para organizar los elementos de la vista principal. En este diagrama los enlaces a las secciones del informe se encuentran en un menú horizontal en la parte superior, justo debajo del encabezado, y en la parte lateral, se muestra el menú que contendría los filtros de datos.

4.3 Plan de Pruebas

En esta etapa se elaboró la primera versión del plan de pruebas, la versión final(en ingles) se puede encontrar en los anexos de este documento, en esta sección se incluye solamente una descripción de la estructura del plan y ejemplos de los casos de prueba.

El plan de pruebas se redactó tomando en cuenta las especificaciones del documento de requisitos. Cada uno de los casos de prueba verifica el correcto funcionamiento de alguna de las funciones definidas en dicho documento. También se incluye casos de prueba para verificar aspectos no funcionales como el rendimiento y la corrección de la marcas y los logotipos (*branding*). Las categorías en que se dividieron los casos en el plan de pruebas son las siguientes:

- Lista de exámenes: Corresponde a los casos de prueba de la interfaz donde se muestra la lista de exámenes con resultados y los enlaces para abrir el informe *HTML*.
- Secciones del informe : Estos casos de prueba se diseñaron para verificar la consistencia de los datos mostrados en las distintas secciones del informe.
- Exámenes con sistema de asignación de notas : Corresponde a los casos de prueba que verifican las funciones que se ejecutan solamente cuando el examen cuenta con un sistema de asignación de notas.
- Navegación e interfaz de usuario: Verifican las funciones de interacción entre el usuario y el informe. También se incluyen en esta parte del plan los casos para comprobar la correcta apariencia de la interfaz gráfica (tablas, gráficas, *layout*, etc).

- Marcas, logotipos e idioma: Se verifica que los logotipos y marcas correspondan con el derivado del producto principal, y que el idioma del informe coincida con el del examinador en *EvaExam*.
- Seguridad y permisos de usuario : Son los casos de prueba que se diseñaron para comprobar que ningún usuario no autorizado puede abrir el informe.
- Rendimiento: Estas pruebas buscan encontrar fallos en el informe cuando la cantidad de datos que se manejan es mayor a la estándar.

4.3.1 El Documento del Plan de Pruebas

El documento del plan de pruebas consiste en un hoja de cálculo donde se encuentran contenidos todos los casos de prueba, clasificados de acuerdo a las secciones mencionadas. Este documento tiene un formato que permite incluir los resultados para varias iteraciones, el encabezado del documento incluye, como se muestra en la figura 4.6 , campos para registrar los datos de cada iteración, estos campos corresponden a información sobre el ambiente de trabajo y la configuración de sistema,la versión de los componentes, información sobre la persona que ejecutó las pruebas, la duración de las mismas, la fecha en que se llevo a cabo y otros datos que se consideraron relevantes.

Los Casos de Prueba

Debajo del encabezado se encuentran las secciones del plan con sus correspondientes casos de prueba(figura 4.7). Cada caso de prueba esta formado por los siguientes campos:

- ID (TestId): Es el código de identificación del caso de prueba.
- Condición Previa(Precondition): Representa la condición o el estado que debe guardar el sistema antes de ejecutar el caso de prueba.
- Acción (Action): Las acciones en que consiste el caso de prueba mismo.
- Resultado Esperado (Expected Result): Es el resultado esperado o el estado posterior en que se tendría que encontrar el sistema después de la ejecución del caso de prueba.
- Resultado del Caso de Prueba: Es una nota que califica el resultado del caso y debe tener alguno de los siguientes resultados: *OK* si el resultado fue el esperado, *NOK* si el resultado no fue el esperado,*Not Possible* si por alguna razón no se pudo probar la función y *Not Implemented* si el caso de prueba no se ejecutó.

Version V0.1

EHR Version	Dev1
Test Result	**Failed**
Test Enviroment and Configuration	
VM Ort	QAVIM
Gast OS	W2K8
Client Java RE (JVM)	-
Webserver (Typ + Version)	Apache
Database	MySQL
Browser version	Chrome 29
Installation Type	-
Installation Path	Default
Webserver location	Default
Database	Default
	Default
	Default
zusätzliche Dateien (z.B. Skripte, Add-ons):	
Product Components	
VividForm Designer	-
EvaExam /ClassExam	-
VividForms Reader	-
Scanstation	-
Test Information	
Tester	mc
Test duration in h	8h
Date	21/09/2013
License Type	v
Update from old version	
Update from version	-
	-
	-
	-
zusätzliche Dateien	-
Result	
ok	20
nok	5
not possible	0
not implemented	0
Test coverage	**100,0**
Error Rate	**20,0**

Figura 4.6: Plan de Pruebas

5 Diseño

En este capitulo se presenta el diseño del sistema. El objetivo de esta etapa es definir con detalle la solución técnica que se dará a los requisitos presentados en las etapas anteriores. Para ello se debe diseñar el modelo arquitectónico de la aplicación e identificar los subsistemas que lo conforman,establecer los requisitos de integración , los estándares que se seguirán y los componentes de terceros que se utilizarán. Además se debe hacer una revisión de los casos de uso y los requisitos y realizar cambios de ser necesario.

5.1 Estándares de Documentación

El primero paso fue definir los estándares que se seguirían para la documentación del diseño. Los lineamientos de documentación para proyectos de desarrollo de *EvaExam*, como es el caso del *EHReport Pluigin,* establecen que para documentar las especificaciones de diseño de un proyecto, se debe crear un documento de concepto o *concept document*. El documento de concepto esta escrito originalmente en ingles,la información contenida en él se encuentra traducida al español y distribuida en las distintas secciones de este capítulo.

5.2 Integración con EvaExam

El requisito REQ_014 especifica que la integración del *plugin* con *EvaExam* debe seguir las reglas establecidas para proyectos *pse (profesional services)*, estas reglas definen que el punto de entrada para cualquier componente externo debe ser un enlace en el menú izquierdo de la página de inicio del usuario , las pantallas que conforman la aplicación deben ser accesibles únicamente desde este enlace. El *EHReport Plugin* es una extensión que estará disponible únicamente para los usuarios con el rol de examinador, por lo que el enlace de entrada debe mostrarse en la página de inicio de este tipo de usuarios, tal como se puede observar en la figura 5.1.

Para que el código del *EHR Plugin* sea reconocido adecuadamente por *EvaExam* y se pueda hacer uso del *framework Zend*, que *EvaExam* incluye por defecto, es necesario seguir algunos estándares respecto de los nombres y ubicación de los archivos. La siguiente tabla indica las carpetas(en el servidore de *EvaExam*) donde se deben ubicar los distintos componentes de la aplicación:

Componente	Ubicación
Las clases que conforman el modelo.	***htdocs/evaexam/application/modules/custom/models***
Controlador	***htdocs/evaexam/application/modules/custom/Controllers***
Scripts de las vistas	***htdocs/evaexam/application/modules/custom/views/scripts/EhtmlReport***
Layouts	***htdocs/evaexam/application/layouts/***
Archivos CSS	***htdocs/evaexam/application/layouts/css***
Archivos de JavaScript	***htdocs/evaexam/application/layouts/scripts***
Archivos de idiomas	***htdocs/evaexam/res/evaexam***

Tabla 5.1: Ubicación de los componentes en EvaExam

Para que *EvaExam* agregué un enlace de acceso en la interfaz del examinador es necesario crear un archivo con el nombre *customermenu.inc.php y* ubicarlo en la carpeta: *htdocs/evaexam/customer.* Este archivo es reconocido automáticamente por *EvaExam* y contiene los parámetros para la creación del enlace.

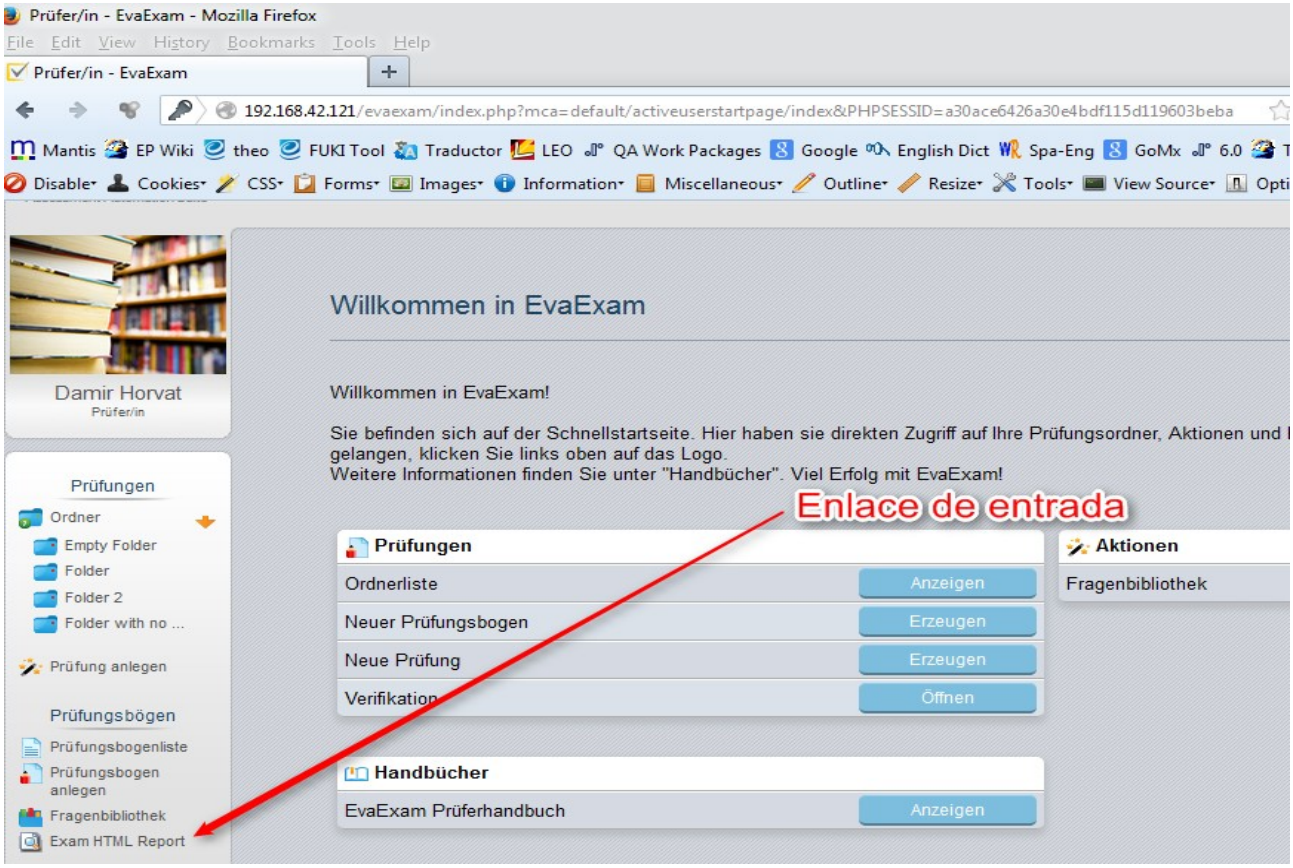


Figura 5.1: Enlace de entrada en EvaExam

El código de este archivo es muy sencillo, por lo que se definió antes de la etapa de desarrollo, y se muestra a continuación:

```
function addMenuItemActiveAccountExam()
{
	$aLink[] = array( 'imageSrc' => "bilder/filter.gif",
					'link' => "index.php?mca=custom/ehtmlreport/index&PHPSESSID=",
					'linkText' => "Exam HTML Report",
					'altText' => "Exam HTML Report Plugin",
					'cssClass' => "");

	return $aLink;
}
```

La función *addMenuItem...* le indica a *EvaExam* que debe insertar un enlace para el plugin en la interfaz del examinador, los parámetros de este enlace se definen en el arreglo *$alink* y son los siguientes:

- imageSrc: es la ubicación de la imagen que servirá como icono para el enlace.

- link: es la ruta a la que apunta el enlace, en este caso a la acción *index* del controlador ehtmlreport.
- linkText: es el texto que muestra el enlace.
- altText: es el texto que se muestra para el atributo alt.

5.2.1 Convenciones para el nombre de clases y archivos

Todas las clases creadas para el proyecto deben incluir en el nombre el prefijo *EH* que las identifica como componentes del *EvaExam HTML Report Plugin*. Las excepciones a esta regla son la clase que implementa el controlador, que se nombró *Custom_EhtmlReportController* y los scripts de las vistas que no son clases sino simple código php y html almacenado en archivos con la extensión .phtml.

5.3 Arquitectura del Sistema

La arquitectura del *EHReport Plugin* se diseñó siguiendo los principios del patrón de software modelo-vista-controlador, que separa los datos y la lógica de negocios de la interfaz del usuario. La implementación de esta arquitectura se realizó con la ayuda del *framework Zend* que proporciona herramientas y componentes que facilitan el desarrollo del código. Por su complejidad esta fuera del alcance de este documento incluir explicaciones extensas sobre la los fundamentos teóricos del patrón mvc, así como de los detalles conceptuales y técnicos del *framework Zend*, si se desea comprender mejor estos temas se pueden consultar las referencias bibliográficas de esta memoria*.

En el *EHReport Plugin* el modelo es una representación del informe mismo que contiene y gestiona el acceso a todos los datos que se mostrarán al usuario, la vista corresponde al *frontend* o interfaz de usuario y contiene el código html,css, y las rutinas en JavaScript. El controlador completa la estructura, responde a eventos generados por las acciones del usuario, modifica el modelo si es necesario y decide que vista será mostrada y que datos se necesitarán.

La figura 5.2 muestra la arquitectura conceptual de la aplicación, se puede observar la interacción entre el modelo, la vista y el controlador. El flujo de información muestra que los datos provienen de *EvaExam*, se obtienen por medio del cliente SOAP y se almacenan en el modelo. En el diagrama también se puede identificar al *EHReport Mapper*, este componente implementa el patrón de arquitectura de software conocido como *data mapper* y es una interfaz entre el modelo y el cliente *soap*, la función de este componente se detalla más adelante. La vista se comunica con el modelo para obtener los datos que se presentarán al usuario. El controlador se comunica tanto con el modelo como con la vista, cuando el usuario realiza una acción el controlador debe reaccionar actualizando los datos del modelo, modificando la vista o cargando una nueva vista.

* Se puede encontrar una breve introducción en ingles tanto al patrón de diseño mvc como al framewoek Zend en el siguiente enlace: http://framework.zend.com/manual/1.12/en/learning.quickstart.intro.html

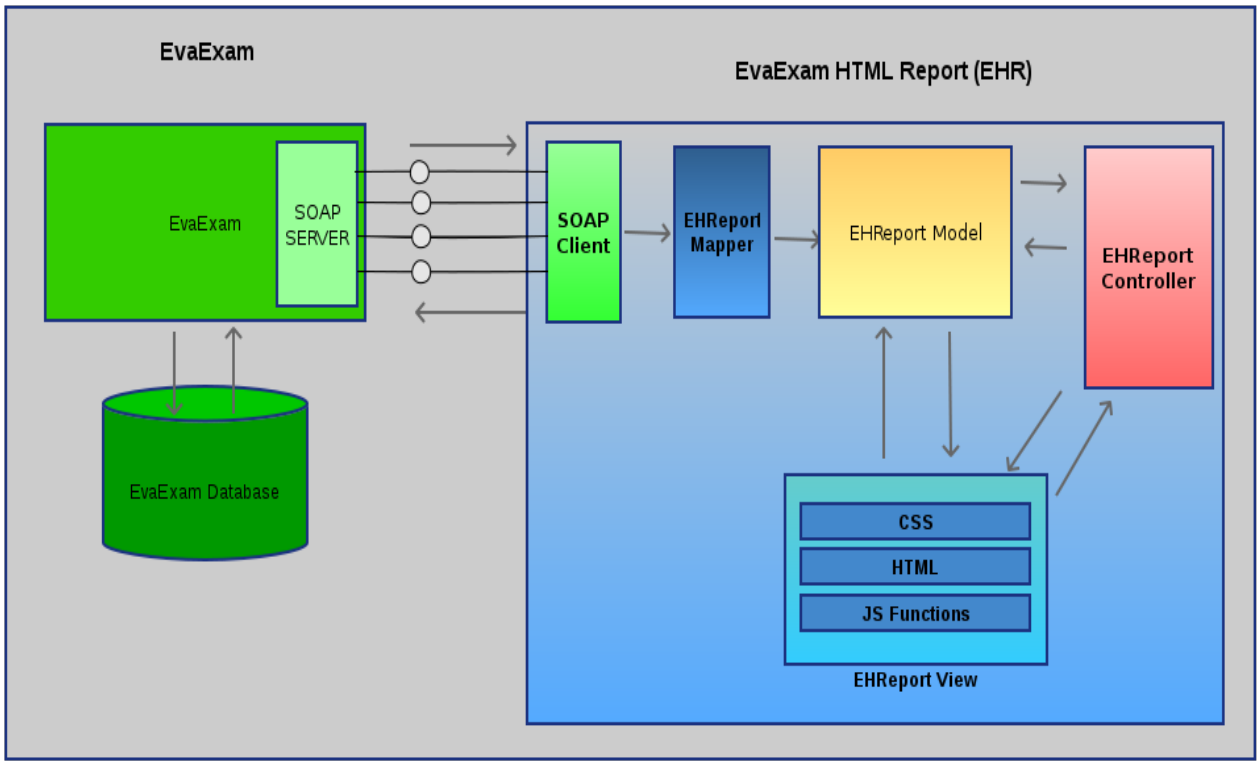


Figura 5.2: Arquitectura del EHReport Plugin

5.3.1 El Modelo Del Informe

La clase *EHReport* representa el modelo del informe. El modelo contiene la información que fundamenta lógica de negocios, e implementa rutinas para la gestión de los datos, en este caso el modelo almacena casi toda la información que se presenta al usuario en las vistas de la aplicación. *EHReport* es una clase del tipo *entidad de dominio o domain entity* en el paradigma: *domain driven design o* diseño dirigido por dominio, *y* es, al momento de su creación una plantilla vacía que no contiene información, los datos del informe se transfieren a una instancia de esta clase por medio del *EHReportSoapMapper*.

La figura 5.3 muestra el diagrama de clase del modelo. La información del examen se almacena en las variables miembro de la clase, para gestionar esta información la clase cuenta con una interfaz compuesta por métodos tipo *set* y *get* para cada variable. La interfaz además cuenta con métodos que proporcionan datos derivados, estos son datos compuestos generados a partir de otros datos del modelo. La información almacenada en las variables miembro se describe a continuación:

- *participants:* Almacena los resultados de los participantes, es un arreglo de instancias de la clase *EHParticipantResult*, cada instancia en este arreglo representa el resultado de un participante.
- p*laceholders:* Es un arreglo asociativo que contiene datos relativos al examen, algunos de los cuales se tienen que mostrar en la vista, como el nombre del examen o la tasa de respuesta.

- *examId:* Es el número de identificación del examen en la base de datos de *EvaExam*.
- *gradingKey:* Objeto que representa el criterio de asignación de notas.
- *reportName:* Nombre del informe.

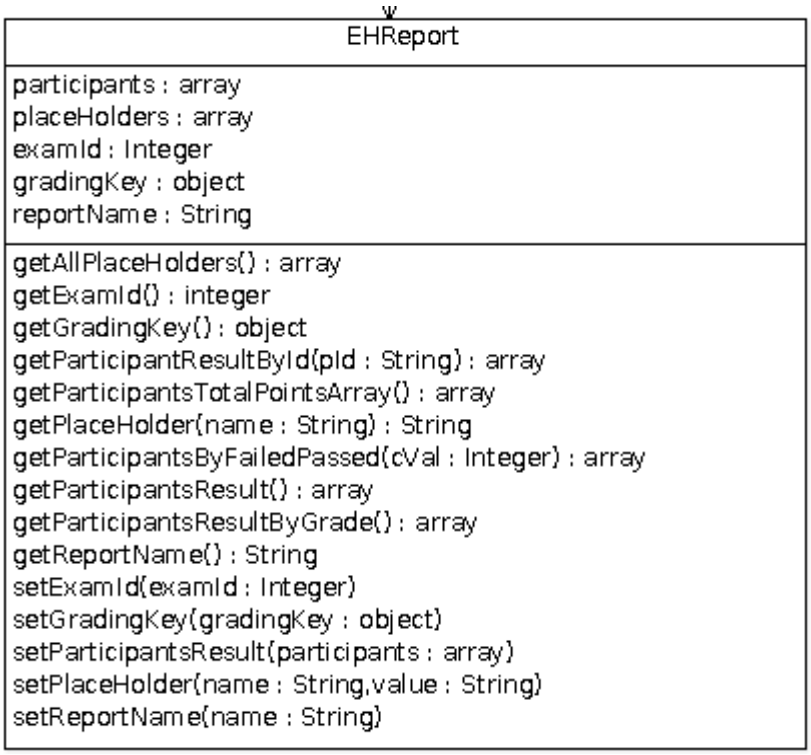


Figura 5.3: Diagrama de la clase EHReport

Algunos ejemplo de los métodos auxiliares que completan las rutinas de acceso a los datos son *getParticipantsByFailedPassed* que devuelve los resultados de los participantes separados a partir del resultado global en el examen, o *getParticipantResultById* que devuelve el resultado individual de un participante tomando como parámetro su número de identificación. La documentación completa sobre las clases y métodos se puede encontrar en el repositorio del proyecto[1].

PlaceHolders

La variable miembro *placeHolders* almacena constantes que normalmente se tienen que mostrar en alguna parte de la la vista final del informe en HMTL o bien que sirven para indicar alguna característica del examen. Se pueden agregar tantas constantes como se quiera y existen algunas que se definen al momento de crear la instancia de la clase y se les asigna un valor por defecto,el valor final que tendrán se establece cuando se transfiere la información del examen a la instancia de clase. La tabla 5.2 muestra las constantes por defecto que se definen al momento de crear una instancia de la clase EHReport.

1https://github.com/codergolem/EvaExam-HTML-Report

Tipo	Nombre	Descripción
String	examName	Nombre del examen
String	createDate	Fecha de creación del examen
Int	formId	Numero de identificación del formulario de examen
String	formShortName	Nombre del formulario de examen
Boolean	isOnline	Indica si el examen es en línea o en papel
float	maxReacheablePoints	Número máximo de puntos obtenibles en el examen
float	approvalThreshold	Mínimo número de puntos necesarios para aprobar el examen
int	approvedParticipants	Número de participantes que aprobaron el examen
boolean	isOpen	Indica si el examen aún continua abierto
int	examStatus	Indica el estado del examen

Tabla 5.2: Constantes en el arreglo PlaceHolders

5.3.2 El Cliente SOAP

El cliente *soap* es la interfaz entre *EvaExam* y el *EHReport plugin*, su función es realizar las llamadas a los métodos del servidor soap de *EvaExam* para obtener los datos del informe.

Este componente encapsula la clase *SoapClient* que viene integrada por defecto en *php* versión *5.0* y posteriores. La clase *SoapClient de php* proporciona métodos para realizar peticiones a cualquier servidor *soap*. El cliente *soap* de este proyecto se implementó en la clase *EHSoapClient y* agrega algunas operaciones especificas para facilitar la conexión con el servidor *de EvaExam*.

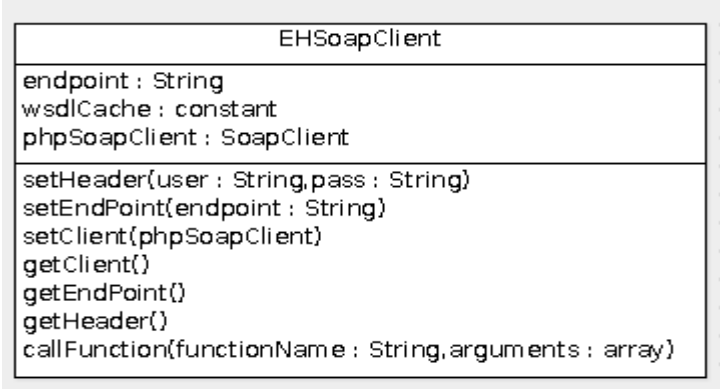


Tabla 5.3: Diagrama de la clase EHSoapClient

En la figura 5.3 se observa el diagrama de la clase *EHSoapClient*. Para realizar la conexión se necesita la URL del servidor *(endpoint)*, una instancia de la clase *SoapClient* y los datos para autenticarse (usuario y contraseña). Las llamadas a los métodos *soap* se hacen utilizando el método *"callFunction"*.

5.3.3 EHReportMapper

Ya se describió la clase que representa el modelo en la arquitectura mvc y se mencionó que esta clase es una plantilla vacía al momento de crearse, por lo que puede almacenar la información de cualquier examen. Para llenar esta plantilla y asignar valores a las variables miembro del modelo se utiliza la clase *EHReportSoapMapper*.

La clase *EHReportSoapMapper* implementa el patrón de arquitectura de software conocido como *data mapper*[1] y conforma la capa de acceso de datos *(Data Access Layer)*, es decir es una interfaz entre el origen de los datos y el modelo que contendrá dichos datos. *EHReportMapper* utiliza una instancia de la clase *EHSoapClient* para llamar a los métodos de la interfaz *soap* de *EvaExam*, el resultado de las llamadas a estos métodos son los datos del examen, luego transfiere estos datos a un instancia de la clase *EHReport*.

Para incrementar la flexibilidad del diseño y facilitar desarrollos futuros se diseñó la interfaz *EHReportMapper* que contiene un sólo método abstracto: *buildReport*, cuya función es construir el modelo del informe. La clase *EHReportSoapMapper* implementa el único método abstracto de la interfaz *EHReportMapper*. De esta manera si posteriormente se quiere construir el informe a partir de una fuente de datos distinta a la interfaz *soap*,tomando directamente la información de la base de datos por ejemplo, lo único que se tiene que hacer es remplazar la clase *EHReportSoapMapper* por una clase *mapper* diferente que implemente el método *buildReport*. La figura 5.4 muestra el diagrama de clases de la interfaz *EHReportMapper* y de la clase EHReportSoapMapper.

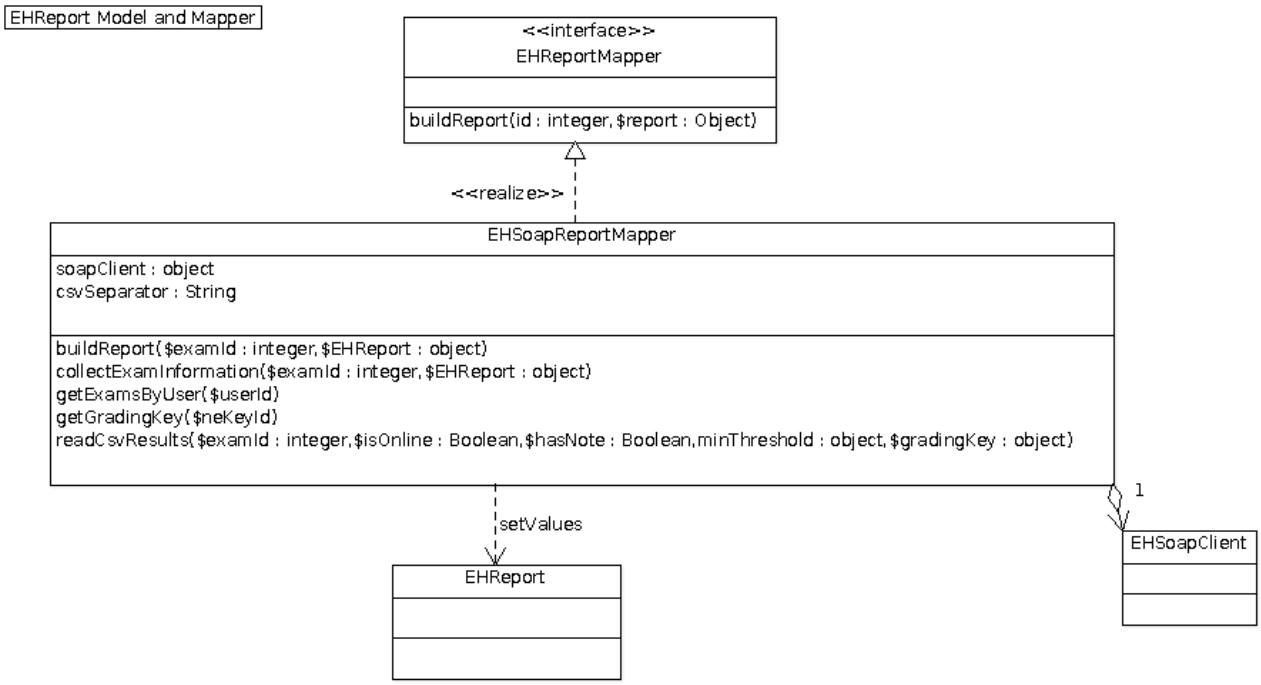


Tabla 5.4: Diagrama de las clases EHReportMapper y EHSoapReportMapper

5.3.4 Clases de Dominio EHGradingKey y EHParticipantResult

En el diagrama de clase del modelo se observa que las variables miembro *gradingKey* y *participants* almacenan objetos, para ambos casos se trata de instancias tipo *domain entity*.

La clase *EhGradingKey* representa el criterio de asignación de notas, el cual define las reglas que determinan la nota que un participante recibe a partir de los puntos que obtuvo en el examen. La nota obtenida por cada participante proviene del archivo *csv* que contiene los resultados, por lo que no es necesario calcularla usando el criterio de asignación de notas, sin embargo las reglas para la

1 Para entender mejor el patrón de diseño *data mapper* se pueden consultar los siguientes enlaces: http://msdn.microsoft.com/en-us/magazine/dd569757.aspx y http://martinfowler.com/eaaCatalog/dataMapper.html

asignación de notas se tienen que mostrar en la vista de examen, además de que se utilizan para determinar si el participante aprobó o no el examen.

Durante la asignación de datos al modelo, de existir un criterio de asignación de notas,el *EHReportSoapMapper* crea una instancia de la clase *EHGradingKey* que luego asigna al modelo. La información para crear el objeto de la clase *EHGradingKey* es proporcionada por el método *GetGradingKey* de la interfaz *soap* de *EvaExam*.

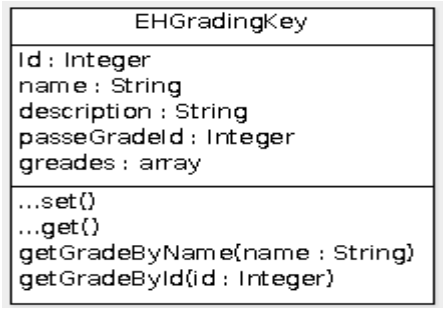


Tabla 5.5: Clase EHGradingKey

En la figura 5.5 se observa el diagrama UML de la clase *EHGradingKey*, con sus variables miembros y métodos. Los métodos *set* and *get* para cada una de las variables miembro se han abreviado para simplificar el diagrama. Cada instancia de esta clase contiene el id del criterio de asignación (id),el nombre(name), la descripción (description), el id de la nota mínima necesaria para aprobar el examen (passedGradeId) y las notas que conforman el criterio (grades). Adicionalmente a los métodos *set* and *get* para cada una de las variables miembro, la clase cuenta con los métodos *getGradeById* y *getGradeByName* que devuelven un objeto de la clase *EHGrade* tomando como parámetro el id o el nombre de la nota respectivamente.

La variable miembro *grades* es un arreglo que contiene la información de las notas. Cada elemento de este arreglo es una instancia de la clase *EHGrade*. La clase *EHGrade* es una clase muy sencilla también del tipo *domain entity, y* almacena el nombre, id y porcentaje asociado a la nota, como puede observarse en el diagrama de clase de la figura 5.6.

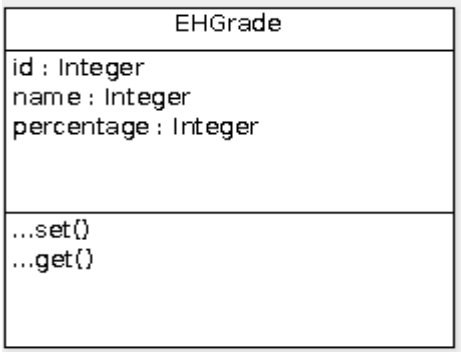


Tabla 5.6: Clase EHGrade

EHParticipantResults

La clase *EHParticipantResults* almacena los resultados individuales de los participantes, y proporciona métodos para acceder a los datos. En la figura 5.4 se muestra el diagrama de clase, como en los figuras anteriores los métodos *get* y *set* para cada variable miembro se abreviaron para simplificar el diagrama. Cada instancia de esta clase contiene los puntos totales obtenidos por el

participante, además de un arreglo donde cada elemento almacena los puntos obtenidos en una de las preguntas del examen. Esta información forma parte de los resultados que luego se muestra en la vista del informe HTML.

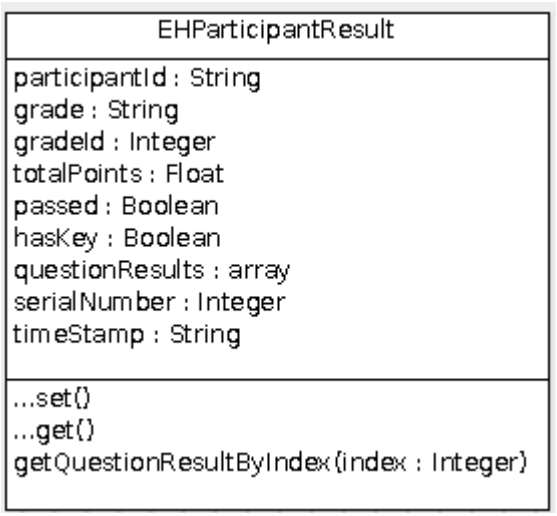


Figura 5.4: Diagrama de la clase EHParticipantResult

5.3.5 Los Datos del Examen

El EHReportSoapMapper cuenta con dos métodos para obtener los datos del examen: *collectExamInformation y readCsvResults.* En el primero se obtienen los datos más importantes acerca del examen: nombre, id del formulario de examen, tasa de respuesta, etc. En el segundo se lee un archivo *csv* que contiene los resultados del examen, los cuales se transfieren a un arreglo de objetos que representan los resultados individuales de los participantes.

El método *collectExamInformation* hace una llamada al método GetExam de la interfaz soap de EvaExam, utilizando como parámetro el id del examen. La respuesta del método GetExam es un objeto que contiene los datos del examen. Un ejemplo de esta respuesta (en formato xml) se muestra a continuación:

```
<SOAP-ENV:Body>
  <ns1:GetExamResponse>
    <Exam>
      <ExamId>454980</ExamId>
      <CreateDate>2013-04-28 13:27:07</CreateDate>
      <Status>1</Status>
      <Name>ExamRWTH</Name>
      <ExamCount>3</ExamCount>
      <FormId>51</FormId>
      <FolderId>8</FolderId>
      <ShortName>OExam2</ShortName>
      <Open>true</Open>
      <IsOnline>true</IsOnline>
      <GradinKeyId>34</GradinKeyId>
    </Exam>
```

Cada uno de los datos que se obtienen en la respuesta se almacena en el modelo como una constante en la variable miembro placeHolders.

El método *readCsvResults* realiza una llamada al método *GetExamResults*, que recibe como parámetro el id del examen y devuelve un enlace al archivo que contiene los resultados de los participantes, tal como se muestra a continuación:

```
<ns1:GetExamResultsResponse>
    <FilePath>http://localhost/evasys/data/tmp/tmp_1a43f0a3c2910ea_51b.csv</FilePath>
</ns1:GetExamResultsResponse>
```

En la figura 5.5 se muestra un ejemplo de como se presentan los resultados de los exámenes en el archivo csv.

Figura 5.5: Formato de los datos en el archivo csv

Cada renglón del archivo representa el resultado individual de un participante; una instancia de la clase *EHReportSoapMapper* lee celda por celda el contenido del archivo para obtener los puntos en cada pregunta , la nota y los puntos totales obtenidos por cada participante. Para cada participante en el archivo se crea una instancia de la clase *EHParticipantResult*,las instancias en conjunto se almacenan en el modelo en la variable *participants*.

Si el examen cuenta con un criterio de asignación de notas, durante la creación del modelo se llama al método *getGtradingKey* de la interfaz soap de EvaExam y se obtiene como respuesta un objeto que representa el criterio de asignación de notas. Una respuesta típica para este método es muy extensa para incluirla en esta sección pero en el anexo *EvaExam Soap Request Samples* se pueden encontrar ejemplos de las peticiones y respuestas para todos los métodos de la interfaz soap que se utilizaron en el *EHReportPlugin*.

5.3.6 Creación del Modelo

Para facilitar la compresión del proceso en el que se crea el modelo del informe y se asignan los datos del examen se creó un diagrama UML de secuencia que se muestra la figura 5.6, se puede observar que los comentarios llevan una numeración que corresponde a la secuencia de los eventos,a continuación se presenta una descripción basada en esta numeración:

1. Información del Examen: Se utiliza el método *callFunction* de la clase *EHSoapClient* para llamar a la operación *GetExam*.El resultado de esta llamada es un objeto que contiene la información del examen (nombre,cuestionario,tasa de respuesta,tipo, etc).

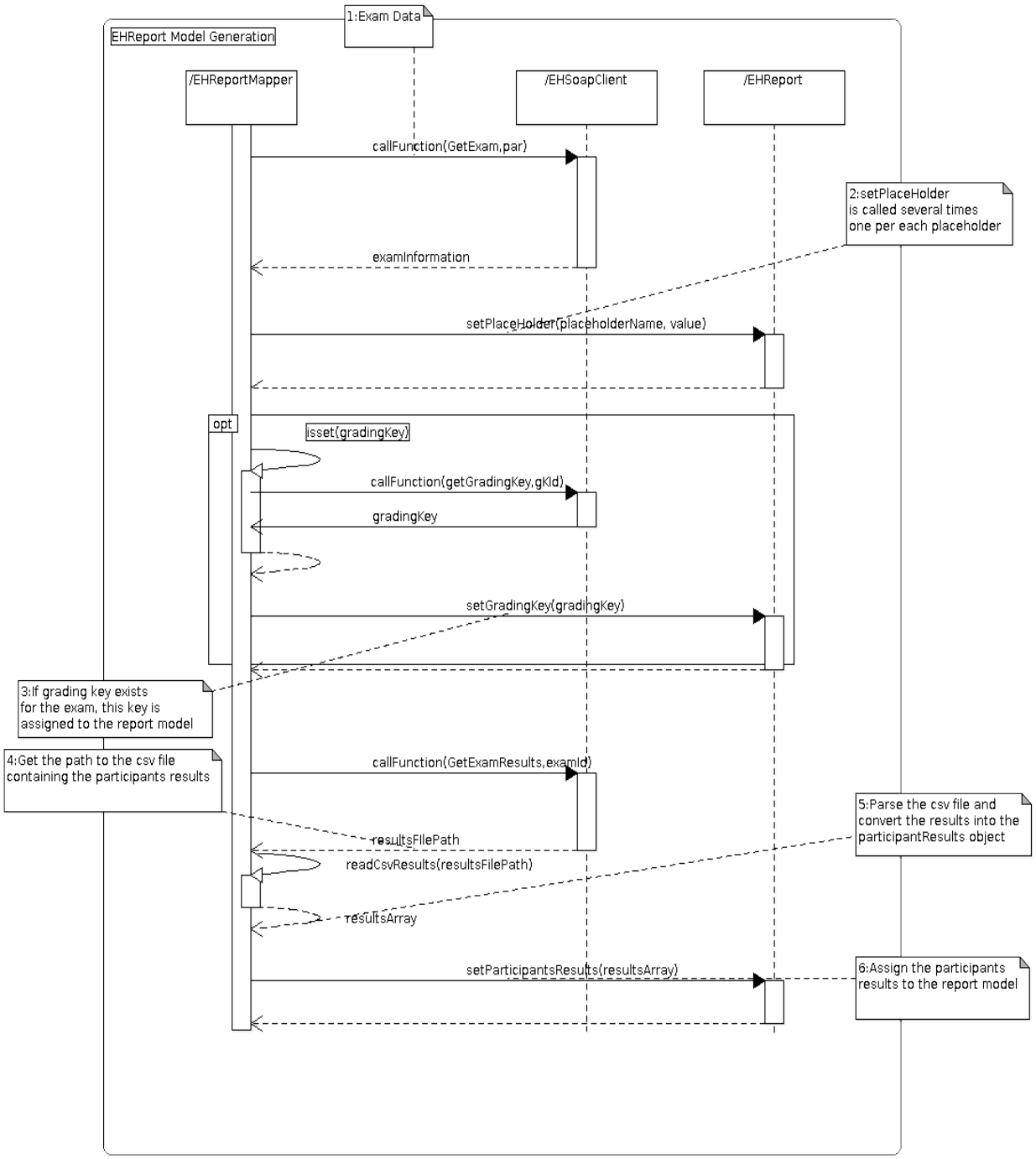


Figura 5.6: Diagrama de secuencia de la creación del modelo

2. *Place holders*: Con la información del examen obtenida en el paso anterior, la clase *EHReportMapper* llama al método *setPlaceHolder* de la clase *EHReport*. Cada llamada a este método incluye como parámetros el nombre del *placeholder* y el valor asignado al mismo. Aunque para simplificar el diagrama sólo aparece una llamada a este método, se realizan múltiples llamadas, una por cada *placeholder*.

3. Si el examen tiene definido un criterio de asignación de notas (*isset(gradingkey)*), se llama al método *getGradingKey* y se asigna el objeto resultante a la instancia de la clase *EHReport.*
4. Se realiza una llamada a la operación *GetExamResults* de la interfaz *soap* de *EvaExam.*El valor resultante es un enlace a un archivo CSV que contiene los resultados del examen.
5. El *EHReportMapper* llama a su propio método r*eadCsVresults* que obtiene el archivo *CSV* y convierte los datos contenidos en el archivo en un arreglo (*array*).
6. El arreglo que contiene los resultados de los participantes se asigna al objeto de la clase EHReport.

5.3.7 El modelo en la lista de exámenes

El método *getExamsByUser,* de la clase *EHReportSoapMapper* , se utiliza especialmente cuando se construye la vista donde se muestra la lista de los exámenes, y devuelve un arreglo que contiene la información de todos los exámenes disponibles para un examinador a partir de su id de usuario, sin incluir los resultados de los participantes. En la vista de la lista de exámenes no es necesario mostrar los resultados de los participantes, por lo que se evita llamar al método *readCsvResults.*

5.3.8 El Controlador

El controlador es el punto de entrada del *plugin*, el enlace que se muestra en la interfaz del examinador hace una llamada a este componente. El controlador del *EHR Plugin* se implementó en la clase *Custom_EHReportController*, que extiende a la clase *Zend_Controller_Action* del *framework Zend.*

El controlador esta basado en eventos o *actions* que se ejecutan como respuesta a una interacción del usuario. Cada *action* está asociada a una vista que se se carga después de que el código del controlador se ha ejecutado, la vista es el resultado final que el usuario recibe como respuesta a la acción que realizó.

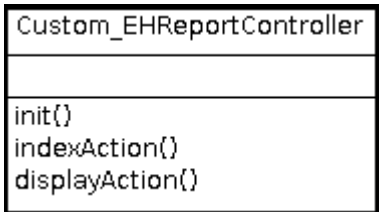


Figura 5.7: Diagrama de clase del controlador

Para este proyecto el controlador incluye dos *acciones* que responden a eventos generados por el usuario. La primera es la acción *index*, que muestra la lista de exámenes con resultados disponibles, esta se ejecuta cuando el usuario hace clic sobre el enlace de acceso al *EHReportPlugin.* La segunda es la acción *display* que muestra el informe HTML para un examen concreto, y se ejecuta cuando el usuario hace clic sobre alguno de los enlaces que se muestran en la lista de exámenes.

La figura 5.7 muestra el diagrama de clase del controlador. Se observa que se definen tres funciones: *init, indexAction y displayAction*.

La función *init* se ejecuta al momento de crear la instancia de la clase, antes de pasar a las acciones

index o *display*. En esta función se realizan las siguientes tareas:

- Crear el objeto de la clase *EHSoapClient* para realizar la conexión a EvaExam.
- Crear las instancias de la clase *EHSoapReportMapper* y *EHReport.*
- Obtener el id del usuario almacenado en la variable $_SESSION.
- Configura el idioma de la vista a partir del idioma del examinador.

En la función *indexAction* se obtienen los exámenes con resultados disponibles y se carga la vista que los muestra. Para ello se llama al método *getExamsByUser* de la clase *EHReportMapper*, este devuelve un arreglo que contiene los datos de los exámenes con resultados para el examinador en cuestión, esta información se pasa a la vista antes de cargarla.

La función *displayAction* se ejecuta cuando el usuario hace clic sobre alguno de los enlaces que se muestran en la lista de exámenes, como resultado se debe mostrar el el informe HTML para el examen correspondiente. Las tareas que esta función realiza son las siguientes:

- Obtener el id del examen que se enviá como parámetro POST en la petición http.
- Construir el modelo del informe utilizando el objeto de la clase *EHReporSoapMapper.*
- Asignar a la vista el nombre del examinador, apellido y id de usuario en *EvaExam*, esta información se obtiene de la variable $_SESSION .
- Definir el número máximo de preguntas y participantes que se mostrarán en una página.
- Definir si las paginas de preguntas se mostraran a partir del número total de preguntas.
- Definir si las paginas de participantes se mostraran a partir del número de participantes.
- Definir los valores del informe para el criterio de asignación de notas.
- Calcular los valores de media, mediana y desviación estándar utilizando la clase estática *EHMathUtils.*

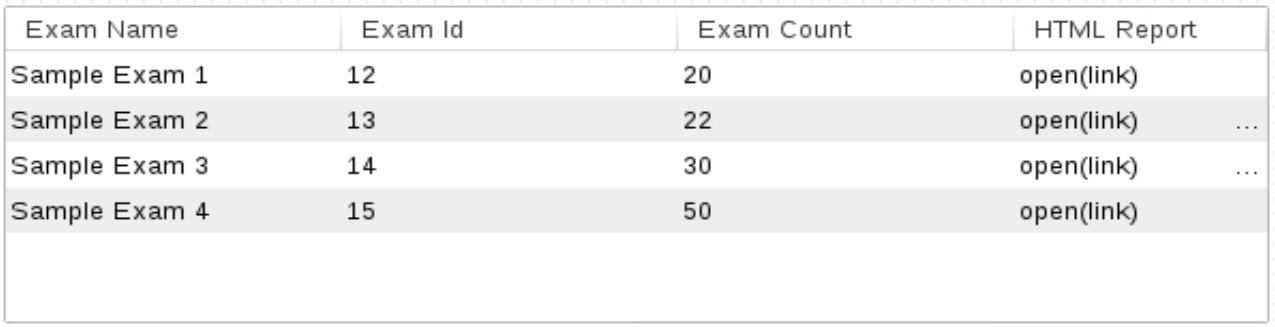
El controlador carga las vistas correspondientes de forma automática, en el *framework Zend* si no se configura otra cosa, para cada acción debe existir una vista, por defecto la vista se carga al terminarse de ejecutar una acción. El controlador busca el archivo de vista correspondiente a la acción basándose en el nombre de la acción y el nombre del controlador. El nombre del archivo que contiene la vista debe coincidir con el nombre de la acción y debe ubicarse en una carpeta con el mismo nombre del controlador. Por ejemplo para que el controlador encuentre la vista que corresponde a la acción *display (displayAction)* debe existir en el directorio de vistas: una carpeta con el nombre *EhtmlReport* que coincide con el nombre del controlador(el prefijo *Custom_* indica el nombre del módulo), y en esta carpeta debe existir un archivo con el nombre *display.phtml* que contiene el código de la vista.

5.3.9 La Vista

La vista es el componente del modelo mvc que representa la interfaz del usuario. La interfaz del EHReport Plugin cuenta con dos vistas: la lista de exámenes y el informe HTML. La vista de exámenes está asociada a la acción *index* del controlador y muestra una tabla con los nombres de los exámenes, id, tasa de respuesta y un enlace para abrir el informe HTML. La vista del informe HTML está asociada a la acción *display* y muestra los resultados del examen.

Vista de Exámenes

Esta vista es la interfaz de inicio del *plugin* y se muestra cuando el examinador hacer clic sobre el enlace de entrada (véase fig. 5.1). Para mostrar los exámenes disponibles se utiliza una tabla que contiene cuatro campos: nombre de examen, id, numero de respuestas y *HTML Report*,este último contiene un enlace para abrir el informe HTML. Cada renglón de la tabla representa un examen. El diseño de esta interfaz se muestra en la figura 5.8.



Exam Name	Exam Id	Exam Count	HTML Report
Sample Exam 1	12	20	open(link)
Sample Exam 2	13	22	open(link)
Sample Exam 3	14	30	open(link)
Sample Exam 4	15	50	open(link)

Figura 5.8: Vista de Exámenes

El enlace que abre el informe HTML para un examen se construye automáticamente utilizando el siguiente formato:

href="index.php?mca=custom/ehtmlreport/display&examid=<?php echo $exam->ExamId; ?>&PHPSESSID=<?php echo session_id(); ?> " target="_blank">Open</td>

se observa que se realiza una llamada a la acción *display* del controlador y se pasa como parámetros el id del examen y el número de sesión actual. En la acción *display* se obtiene la información de los exámenes disponibles y se transfiere a la vista. En la a vista se construye la tabla que lista los exámenes y se verifica que se incluyan unicamente aquellos con *status* 1, el resto son exámenes que aún no cuentan con resultados disponibles.

Vista del Informe HTML

Es la vista donde se muestran los resultados de los participantes en un examen y representa el informe HTML en si mismo. En esta etapa realizaron algunas modificaciones al diseño de la etapa anterior y se definió la apariencia final de esta vista. La información que se presenta en el informe se distribuyó en las siguientes secciones:*Exam Report ,Question Results ,Grade Result,General Statistics*.

Los elementos principales de esta vista se observan en la figura 5.9, a continuación se presenta una descripción de cada uno de ellos:

1. Encabezado-Logotipo de EvaExan: Es una barra en la parte superior de la pantalla donde se muestra el logotipo de EvaExam.
2. Encabezado del Informe: Muestra la información del examen al que corresponde el informe. Los datos que se muestran son: el nombre del examinador,el nombre del examen, el nombre del formulario del examen y el número de participantes.
3. Barra de Navegación: Contiene enlaces a cada una de las secciones del informe.
4. Contenido de la Sección: En este espacio se muestran las tablas, gráficas y otros elementos que corresponde a los resultados de cada sección.

5. Pie del Informe: Muestra la línea de copyright con el siguiente texto :*Copyright 2013 Electric Paper Evaluationssysteme GmbH*.

El logotipo de EvaExam, el encabezado, la barra de navegación y el pie del informe se muestran en todas las secciones. El contenido de la sección cambia a partir de la sección que se haya seleccionado. Para navegar a través de las secciones el usuario debe hacer clic sobre los enlaces que se encuentran en la barra de navegación.

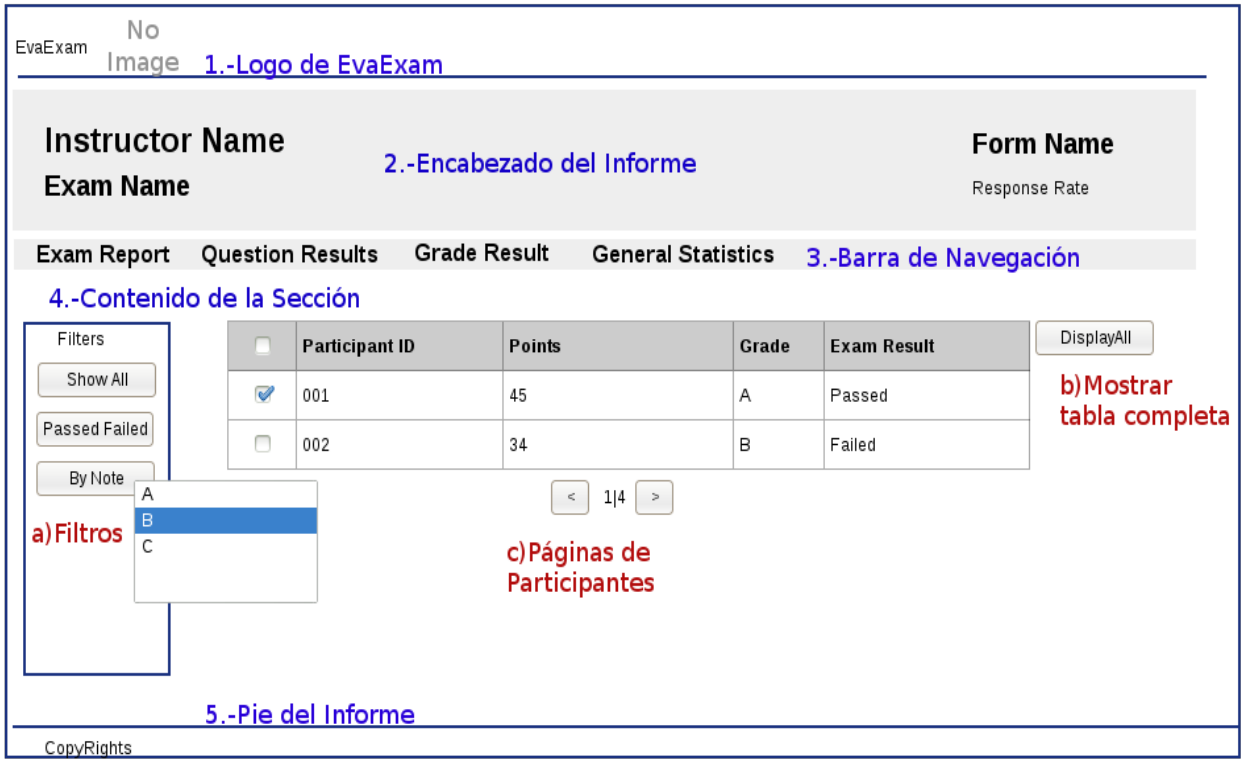


Figura 5.9: Vista del Infome HTML

En código que se enviá desde el servidor todas las secciones son visibles, al momento de cargar la página se utiliza JavaScript para ocultar todas las secciones excepto la sección *Exam Report*. A continuación se presenta una breve descripción del contenido de cada una de las secciones.

A) Sección Exam Report

Esta sección muestra los resultados generales de los participantes en una tabla que contiene los siguientes campos: id del participante, los puntos obtenidos en el examen, y si es el caso la nota obtenida por el participante y una indicación de aprobación o reprobación. El diseño de la tabla se muestra en la figura 5.9. Si el examen tiene un criterio de asignación de notas también se muestra la interfaz para ejecutar los filtros(fig 5.9 a)filtros), de otro manera este menú no debe aparecer.

B) Sección *Question Results*

En esta sección se muestran los puntos obtenidos por cada participante en cada pregunta. Los resultados se muestran en una tabla que contiene el número de identificación de cada participante, los puntos totales obtenidos en el examen, y una casilla que contiene los puntos obtenidos en cada pregunta del examen. El diseño final de esta sección se muestra en la figura 5.10.

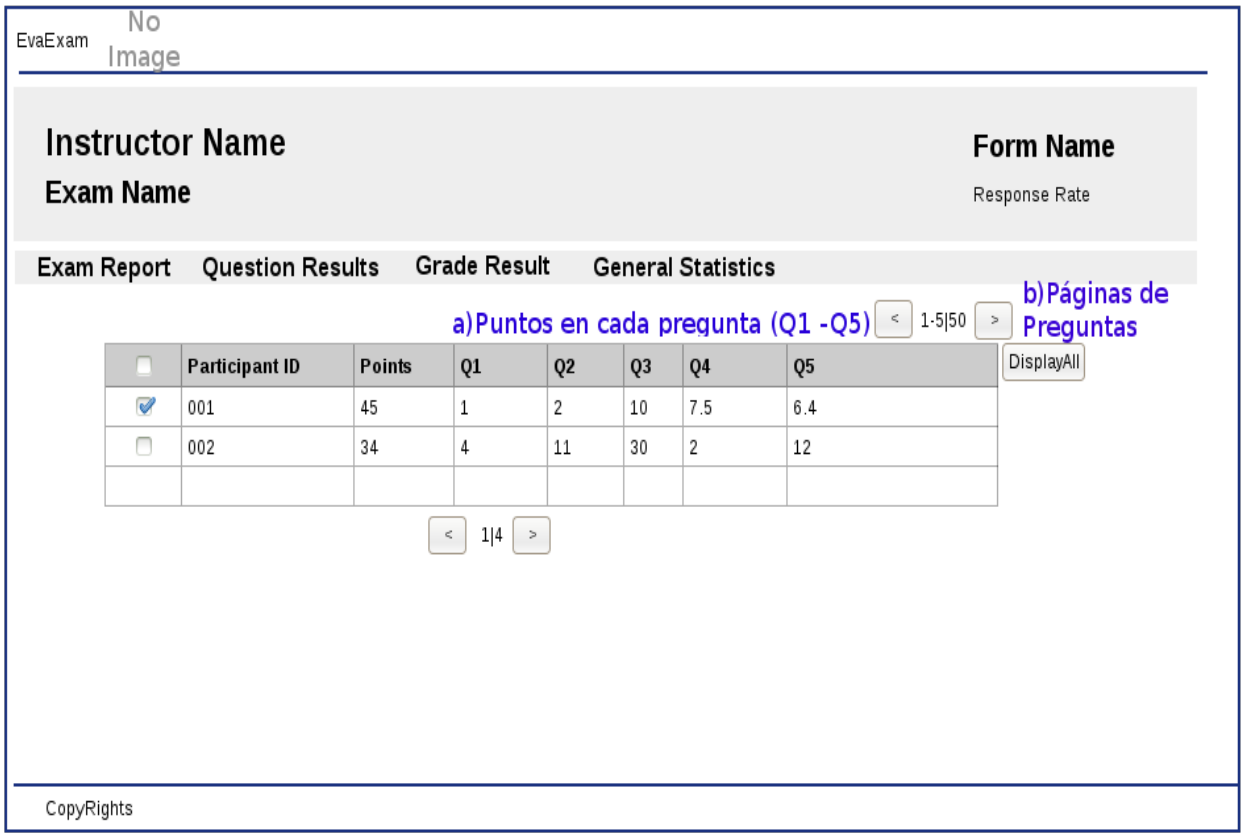


Figura 5.10: Sección Questions Results

C) *Sección Grade Result*

Si existe un criterio de asignación de notas para el examen, se debe mostrar la sección *Grade Result*, de otra manera sólo deben aparecer las otras tres secciones del informe. La sección Grade Result muestra los resultados relativos a las notas obtenidas por los participantes, estos consisten en el número y porcentaje de participantes aprobados y reprobados, y el número y porcentaje de estudiantes clasificados por nota obtenida. En la figura 5.11 se puede observar el diseño del contenido de esta sección, la información que se muestra en las tablas y gráficas es la siguiente:

- Nota mínima aprobatoria: Es la tabla de la parte superior y muestra los datos de la nota mínima aprobatoria; contiene tres campos: nombre,porcentaje y puntos mínimos para aprobar el examen.
- Participantes Aprobados: Las tabla y la gráfica de la parte inferior izquierda muestra en número y porcentaje los participantes que aprobaron el examen y los que lo reprobaron.
- Participantes por Nota: La tabla y la gráfica de la parte inferior derecha muestran la distribución de participantes por nota obtenida. La tabla contiene cuatro campos: el nombre de la nota, el porcentaje mínimo para obtener la nota, el número de participantes que obtuvo la nota en valor absoluto y el número de participantes que obtuvo la nota en valor porcentual. La gráfica muestra la distribución porcentual de participantes por nota.

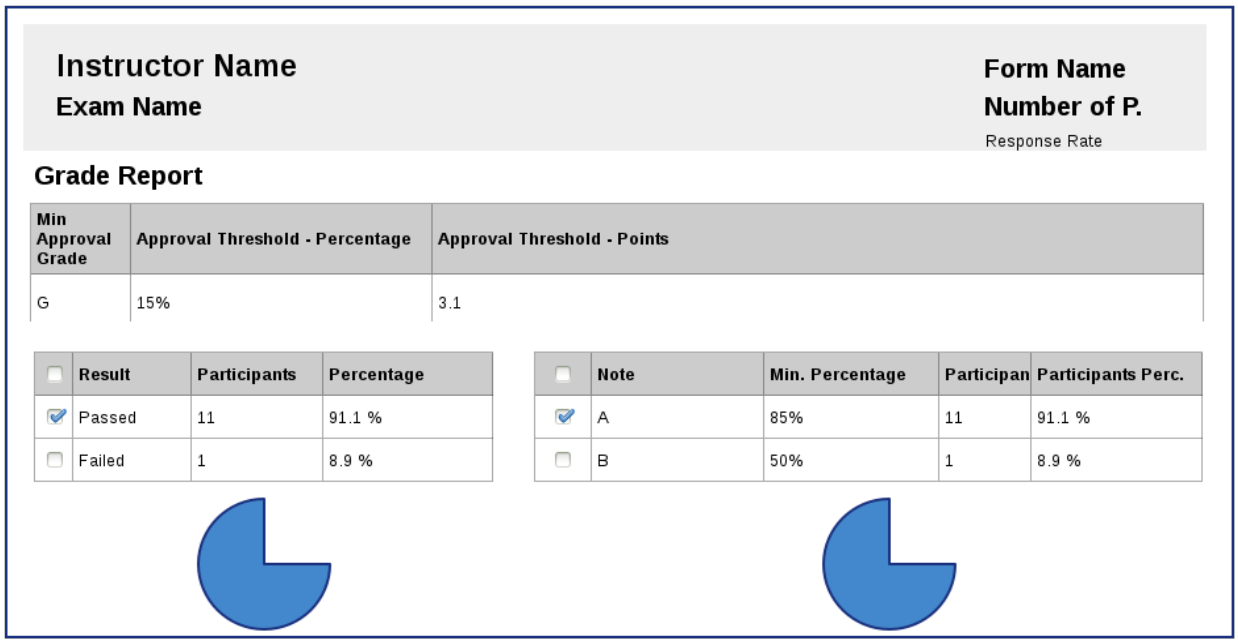


Figura 5.11: Sección Grade Report

D) *Sección de Estadísticas Generales*

El diseño final de la sección de estadísticas generales se muestra en la figura 5.12, esta sección presenta por medio de una tabla los siguientes datos:

- Numero de Participantes
- Numero de máximo de puntos posibles
- Media
- Mediana
- Máximo numero de puntos alcanzados por un participante
- Numero mínimo de puntos alcanzados por un participante
- Desviación estándar

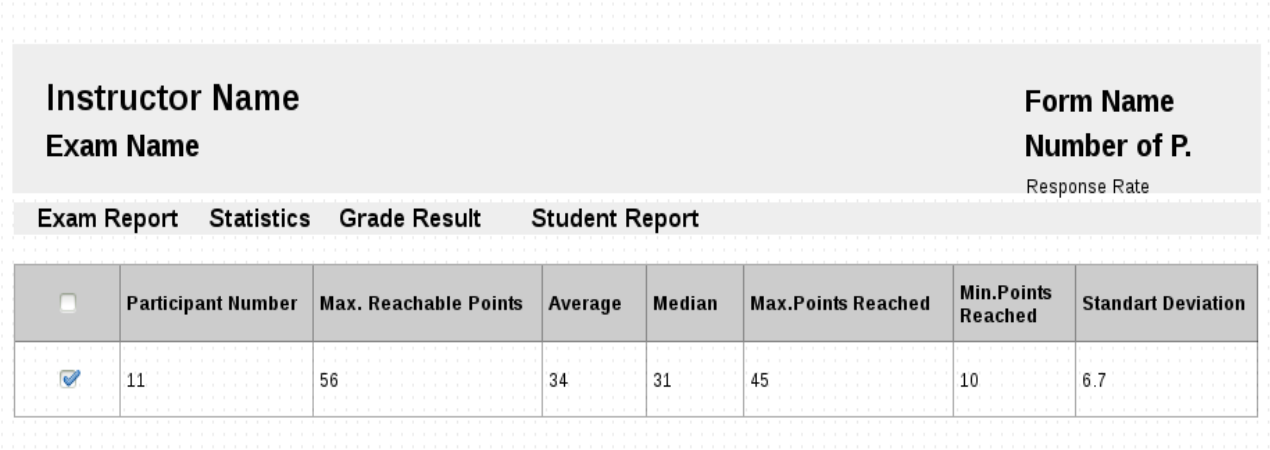


Figura 5.12: Sección General Statistics

Páginas de Participantes y Preguntas

Las páginas de participantes y preguntas son la solución que se encontró para los requisitos REQ_1.1 y REQ_1.2.1, estos establecen que tanto el área del informe donde se muestra los resultados generales de los participantes como el área donde se muestran los resultados de las preguntas deben tener un tamaño fijo independientemente del número de participantes o preguntas.

Los tablas que muestran los resultados de los participantes (en la sección *Exam Report)* y los resultados de las preguntas (en la sección *Question Results)* se dividieron en páginas o segmentos, cada uno de los cuales contiene un número fijo de participantes o preguntas según sea el caso. De estas páginas solamente una es visible por vez y cada página tiene una dimensión fija igual a la de las otras . Para que el usuario pueda ver los resultados de todas las preguntas o participantes debe navegar a través de las distintas páginas utilizando los controles de navegación creados para este propósito. Las divisiones en las tablas se establecieron del lado del servidor utilizando *php y* la navegación se realizó con JavaScript.

Paginas de Participantes

En la sección *Exam Report* las dimensiones de la tabla aumentan verticalmente hacia abajo entre más son los participantes que se deben mostrar; el número máximo de participantes por página que se estableció fue de 25, cuando se alcanza el número máximo de participantes y aún quedan participantes por mostrar se construye una nueva página que almacena nuevamente como máximo 25 participantes. Se crean tantas páginas como sean necesarias para incluir a todos los participantes en la prueba.

Como se mencionó sólo una de las páginas es visible a la vez, por lo que se proporciona al usuario controles de navegación para pasar de una página a la siguiente o para regresar a la página anterior. Los controles de navegación son las flechas que se observan en el diseño de la figura 5.9 (*c)Páginas de Participantes*) justo debajo de la tabla de resultados, en el espacio entre las flechas se indica textualmente, a la izquierda y a la derecha, la página actual y el número total de páginas respectivamente. Las páginas de participantes se utilizan también en la sección de *Question Results* de la misma forma que en la sección *Exam Report*. La figura 5.13 muestra como se verían las páginas de participantes.

En caso de que el examen tenga 25 participantes o menos, las flechas de navegación no se mostrarán dado que no es necesario utilizar las páginas de participantes.

Botón Display All

Uno de los casos de uso que se consideró es cuando el usuario desea eliminar las páginas para que los resultados se muestren en una sola página. Para cumplir con este caso se diseño el botón *Display All* que aparece en la parte superior derecha junto a la tabla de resultados (*fig 5.9 b)Mostrar tabla completa*), tanto en la sección *Exam Report* como en la sección *Question Result*s, al hacer clic sobre este botón se eliminan las páginas de participantes y la lista de participantes se muestra en una sola vista de la tabla. Además este evento hace visible el menú de filtros en la sección *Exam Report*, ya que solo tiene sentido aplicar filtros sobre todos los resultados y no sobre secciones parciales.

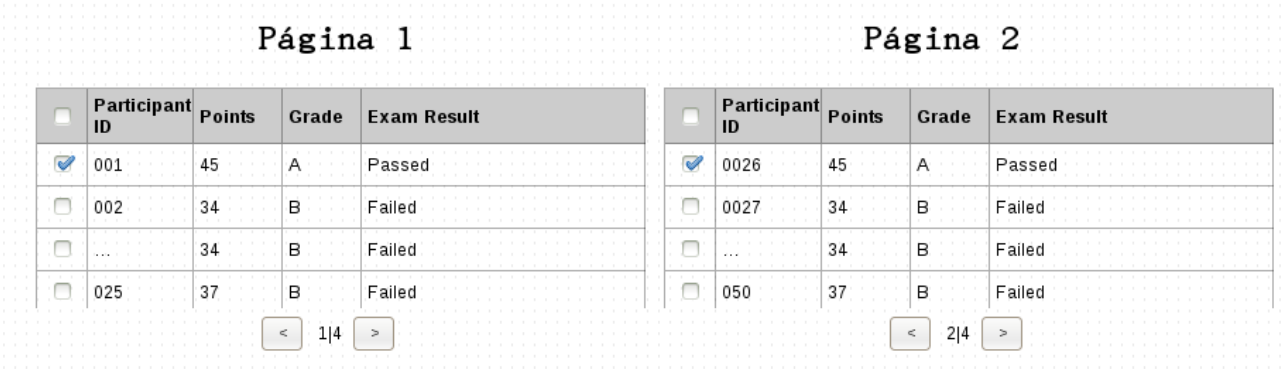


Figura 5.13: Ejemplo del uso de las páginas de participantes

Páginas de Preguntas

Las páginas de preguntas dividen a la tabla de resultados de la sección *Question Results* en segmentos de 30 preguntas de los cuales solo uno es visible a la vez. Para navegar entre las distintos páginas se utilizan las flechas que se ubican en la parte superior derecha de la tabla de resultados(fig 5.10). En el espacio que se encuentra entre las flechas se indica del lado izquierdo la página actual ,con el número de pregunta inicial y final del segmento, y de lado derecho el número total de preguntas. De forma análoga a las páginas de participantes, las páginas de preguntas no se muestran si el examen tiene 30 preguntas o menos. La figura 5.14 muestra una representación de las páginas de preguntas.

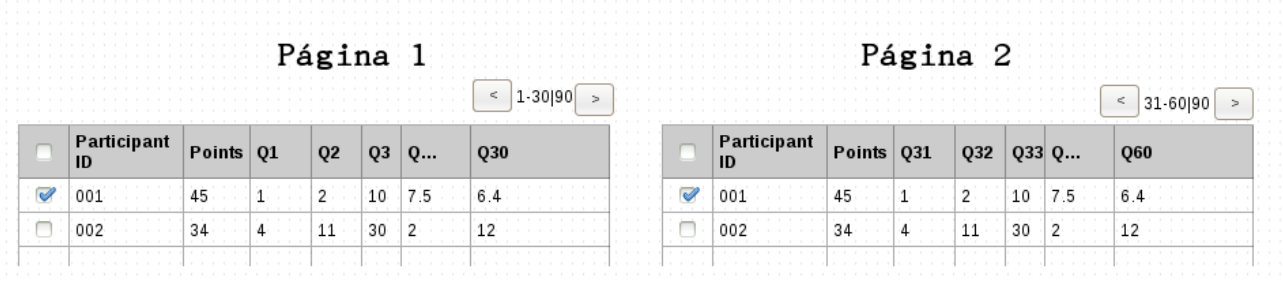


Figura 5.14: Ejemplo del uso de las páginas de preguntas

Filtros

Los filtros se aplican sobre la tabla de resultados de la sección *Exam Report*, su implementación se realizó utilizando JavaScript, por lo que se ejecutan del lado del cliente, una vez que la vista del informe se ha cargado. El menú de filtros esta ubicado en la parte izquierda de la tabla de resultados(*fig 5.9 a)filtros*) *y* contiene tres botones , dos de las cuales despliegan un menú adicional al hacer clic sobre ellos. El funcionamiento de los filtros es el siguiente:

- Filtro de notas: Al hacer clic sobre el botón *by note* se muestra un lista desplegable que contiene las notas posibles para el examen, cuando el usuario hace clic sobre una de las notas, la tabla de resultados cambia para mostrar únicamente los resultados de los participantes que hayan obtenido la nota seleccionada.
- Filtro aprobado/reprobado : Al hacer clic sobre el botón *passed/failed se m*uestra una lista desplegable que contiene dos opciones: aprobado(*passed*) o reprobado *(failed*), al hacer clic sobre una de ellas, la tabla muestra unicamente los resultados de los participantes que correspondan al criterio elegido.
- Mostrar todo : El botón *show all* tiene la función de mostrar los resultados de todos los participantes para restaurar la tabla a su estado original después de que se ha aplicado algún filtro sobre los resultados.

Visibilidad de Los Filtros

Como se puede notar los filtros están relacionados directamente con el criterio de asignación de notas, por lo que cuando el examen no cuenta con uno, el menú de filtros no se debe mostrar. Además si se utilizan páginas de participantes los filtros permanecen ocultos hasta que el participante cambia a la vista completa de los resultados en la sección *Exam Report (ver Botón Display All)*.

5.3.10 Zend Layouts

Las vistas en el modelo mvc son sólo fragmentos de código HTML y php, y no páginas completas. Para formar una página HTML completa es necesario contar con una platilla en donde se inserte el contenido de la vista, de esta manera una plantilla puede utilizarse para varias vistas, con lo que se mantiene un diseño coherente en todas las paginas de la aplicación. Estas plantillas son lo que se conoce como *layouts*.

Zend Framework proporciona herramientas para la utilización de *layouts*, los archivos correspondientes tienen la extensión .phtml al igual que las vistas, y deben colocarse en un directorio especial para que sean reconocidos. El archivo de *layout* debe tener el mismo nombre que la carpeta donde se almacenan las vistas,el *framework Zend* se encarga de insertar automáticamente el contenido de las vistas en el *layout* correspondiente.

El *layout* utilizado para las vistas del proyecto contiene el encabezado de la página donde se muestra el logotipo de EvaExam, y el pie de página donde se muestra la línea de copyright. Ademas también contiene las declaraciones para incluir los scripts y archivos css que se necesitarán.

La figura 5.15 muestra un bosquejo del código del *layout*, se puede observar que el código es realmente sólo el esqueleto de la página, en la linea *$this->layout()->content* se inserta el código correspondiente a la vista.

```
<!DOCTYPE HTML>
<html>
    <head>
        <title>Exam HTML Report</title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=9" />

        <link rel="stylesheet" href="application/layouts/css/bootstrap/css/bootstrap.css">
        <script type="text/javascript" src="application/layouts/scripts/ehreport.js"></script>

        <!--[if lt IE 9]><script language="javascript" type="text/javascript" src="application/layou

    </head>

     <body>
         <div id="contain-wrapper">
            <div id="page-content">
                <div id="headersite" class="clearfix">
                    <span class="inner"><img class="logo" src="images/logos/organization/system.png"
                </div>
                <?php echo $this->layout()->content ?>
            </div>
        </div>
        <div id="footer">Copyright 2013 Electric Paper Evaluationssysteme GmbH </div>

    </body>

</html>
```

Figura 5.15: Bosquejo del código del layout

5.4 Seguridad

El requisito REQ_016 especifica que soló los usuarios con una cuenta de examinador podrán acceder a las funciones del *plugin*, las comprobaciones necesarias para satisfacer este requisito son realizadas por *EvaExam*. El *plugin* se integra como un componente más del sistema por lo que para cada llamada al controlador del *plugin* EvaExam verifica que el usuario cuente con los permisos necesarios para acceder a las funciones solicitadas.

Con la información del archivo customermenu.inc se le indica a *EvaExam* que el enlace al *plugin*, el cual apunta al controlador de este, debe aparecer únicamente en la interfaz de los usuarios con el rol de examinador, de esta manera ningún otro tipo de usuario puede acceder a las funciones del *plugin* desde su interfaz. Existe también la posibilidad de realizar una llamada al controlador utilizando directamente la dirección URL del mismo,como se muestra a continuación:

http://192.168.42.121/evaexam/index.php?mca=custom/ehtmlreport/index&PHPSESSID=7e89008

una dirección URL con este formato se establece como parámetro *href* del enlace al *plugin* en la interfaz del examinador. Sin embargo incluso en el caso de que se realizara una llamada directa como la del ejemplo anterior, solamente los usuarios autorizados tendrían acceso al controlador.

Para entender como *EvaExam* comprueba los permisos del usuario cuando se realiza una llamada al controlador es necesario explicar las partes que componen la dirección URL, las cuales se listan a continuación:

- 192.168.42.121/evaexam: es la dirección URL base del servidor de *EvaExam*.
- index.php: es el punto de entra de EvaExam, cualquier llamada a algún componente del sistema tiene que pasar por este script, el cual se encarga de validar la llamada antes de pasar el control al componente en cuestión.

- mca: es el parámetro que indica el controlador al que se realiza la petición,en el ejemplo su valor es “custom/ehtmlreport/index”,donde *custom* indica el módulo al que pertenece el controlador, *ehtmlreport* el nombre del controlador, e *index* la acción del controlador a la que se realiza la llamada.
- PHPSESSID: es el id de sesión *php* para el usuario que realiza la petición.

El servidor de *EvaExam* está configurado para recibir solamente peticiones dirigidas al script *index.php*, cualquier petición directa a otro componete del sistema es rechazada. Cualquier llamada a una función del sistema debe realizarse a través de index.php incluyendo en el parámetro mca la información necesaria para identificar el componente al que se esta llamando. Index.php se encarga de comprobar la validez de los parámetros, entre otras tareas, y si todo es correcto dirige el flujo de las acciones al componete solicitado.

Ahora bien, todo el proceso descrito anteriormente también se realiza para el controlador del *EHReport Plugin*, por lo que el flujo de datos pasa primero por index.php, una de las comprobaciones que este script realiza es la validez de la sesión php (PHPSESSID), si el número de sesión no esta registrado en el servidor, se niega el acceso al usuario; para conseguir un número de sesión valido el usuario tiene que iniciar sesión con su nombre de usuario y contraseña, por lo que incluso haciendo una llamada directa al controlador, solamente un usuario con permisos válidos podría acceder a las funciones del *EHReport Plugin*.

5.5 Branding

Existen dos variantes de *EvaExam*: *EvaExam* propiamente, y *ClassExam* que se distribuye en el Estados Unidos y Canadá, estas dos variantes solo difieren en los colores y logotipos de la interfaz de usuario, desde el punto de vista funcional son exactamente iguales. El *EHReport Plugin* debe integrarse con ambas variantes y según el requisito REQ_017 la interfaz de usuario debe corresponder con la variante en cuestión. En las interfaces del *EHReport Plugin* el logotipo de *EvaExam* es el único elemento que difiere entre las dos variantes. Este logotipo se muestra en el encabezado de la vista del informe HTML y la imagen proviene del archivo: images/logos/organization/system.png almacenado en el servidor. El código de las interfaces de usuario del EHReport es exactamente el mismo para ambas variantes, ya que el paquete de instalación de *EvaExam* se encarga de instalar la imagen correcta a partir de la variante, ya sea *EvaExam* o *ClassExam*, el nombre y ubicación del archivo no difieren entre variantes, por lo que el logotipo en el encabezado del informe corresponde automáticamente con la variante del producto.

5.6 Idioma

Según el requisito REQ_018 la interfaz de usuario del EvaExam HTLM Report plugin debe estar disponible en lo mismos idiomas por defecto de EvaExam: alemán e ingles.

Zend framework proporciona la herramient *Zend_Translate*[1] que permite separar el código de las vistas, de las cadenas de texto que se muestran en ellas, para facilitar la presentación de la interfaz en distintos idiomas.

Cuando se utiliza *Zend_Translate* la vista no incluye directamente el texto que se muestra al usuario sino se definen únicamente constantes o marcadores que representan el texto final, el *framework Zend* sustituye en tiempo de ejecución los marcadores por el texto final que proviene de

1 Se puede consultar una introducción (en ingles) al componente Zend Tranlate en el siguiente enlace: http://framework.zend.com/manual/1.12/en/zend.translate.html

un archivo que almacena exclusivamente las cadenas de caracteres.

Para realizar este proceso en el controlador se crea una instacia de la clase *Zend_Translate* y se asigna a la vista de la siguiente manera:

```
$this->view->translator = new Zend_Translate (
        array(
           'adapter' => 'array',
           'content' => 'res/evaexam/ehreport.'.$lang.'.inc',
           'locale'  => 'en'
        )
);
```

El arreglo que se pasa como parámetro determina el idioma en que se mostrará la vista, el valor de la constante *content* es la ubicación del archivo que contiene el texto de la interfaz en un idioma concreto, locale define la abreviatura con que se identifica al idioma.

El archivo que contiene el texto final para cada idioma devuelve un arreglo asociativo que relaciona las constantes o marcadores con las cadenas de caracteres del texto de la interfaz. Se tiene que crear un archivo por cada idioma en que se va a presentar la interfaz. Un ejemplo muy sencillo del contenido de este archivo para el idioma ingles sería el siguiente:

```
<?
return $english = array(
   'S_GRADE_SECTION_NAME'  => 'Exam',
   'S_EXAMINER_NAME'  => 'Examiner',
```

En la vista se debe definir una constante para cada texto independiente que se vaya a mostrar en la interfaz de usuario, de la siguiente manera:

```
$this->translator->_('S_GRADE_SECTION_NAME')
```

en el ejemplo anterior ' S_GRADE_SECTION_NAME' representa el titulo de la sección *Grade Section,* al momento de cargar la vista *zend_translator* busca el nombre de cada constante en archivo que se definió en el controlador y la sustituye por el texto final asociado.

De esta manera las constantes se definen una sola vez en la vista, y esta no está asociada a ningún idioma específico. El controlador se encarga de decidir a partir de otros parámetros cual será el idioma en que se mostrará la interfaz del usuario. Si se desea añadir una traducción para la interfaz, lo único que se necesita hacer es crear un archivo que relacione las constantes de la vistas con los textos correspondientes.

En EvaExam el idioma se puede configurar específicamente para el examinador y puede ser distinto del idioma del sistema, el idioma de las interfaces del *EHReport Plugin* debe coincidir con el idioma del examinador que utiliza el *plugin,* por lo que antes de cargar la vista es necesario determinar el idioma de la interfaz del examinador. No es posible saber el idioma del examinador utilizando algún método de la interfaz soap, ni con los valores almacenados en la sesión de php, por lo que esta información se obtiene directamente de la base de datos. Los requisitos del proyecto establecen que la comunicación entre el *plugin* y EvaExam debe realizarse exclusivamente a través de la interfaz soap, sin embargo, dado que no existía otra alternativa se realizó una excepción en

este caso.

Para realizar la conexión con la base de datos de EvaExam se creó la clase *EhDbCon que i*nternamente utiliza la clase estática *Zend_db de ZendFramework* para obtener un objeto que realiza la conexión con la base de datos.

También se diseño la clase *EHCoreUtils* que cuenta con un sólo método *getLanguage*, el cuál encapsula la petición *sql* que devuelve la cadena de caracteres que representa el idioma del examinador.

En la base de datos de EvaExam la información sobre el idioma de la interfaz del examinador esta almacenada en la tabla “benutzer” que tiene el siguiente formato:

UserId	Login	Password	Sprache	....
34	test	test	en	....

Tabla 5.7: Representación de la tabla benutzer

El valor almacenado en la el campo “Sprache” representa el idioma del examinador. El método *getLanguage* recibe como parámetro el id del usuario y devuelve una constante de cadena que representa el idioma del examinador. El id del examinador se obtiene del arreglo de la sesión del usuario.

Cuando un examinador utiliza el *plugin* el controlador se encarga de determinar el idioma del usuario, y a partir de ello configura el idioma en que se mostrará la vista.

5.7 Cálculos Matemáticos

Los requerimientos del sistema especifican que el informe deberá mostrar la media,mediana y la desviación estándar de los puntos totales de los participantes en el examen. Para realizar estos cálculos se diseño la clase estática *EHMathUtils* , que contiene métodos que reciben como parámetros los resultados del examen y devuelven los datos especificados anteriormente.

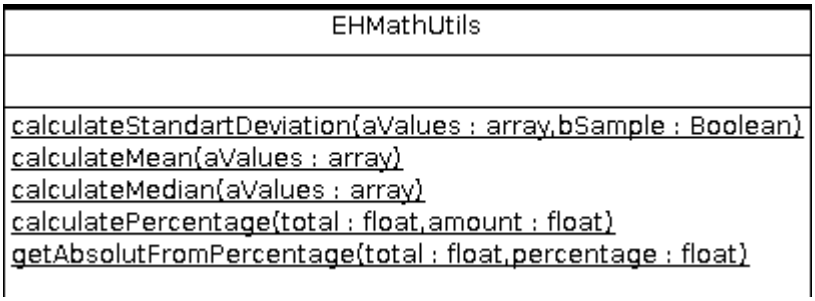


Figura 5.16: Clase EHMathUtils

El diagrama UML de la figura 5.16 muestra los métodos de esta clase, la función de cada uno de ellos es la siguiente:

- calculateStandartDeviation : Calcula la desviación estándar.
- CalculateMean: Calcula la media.
- calculateMedian: Calcula la mediana.

- calculatePercentage : Calcula el porcentaje que representa una cantidad respecto de otra.
- getAbsolutFromPercentage : Calcula el valor absoluto de un porcentaje respecto del total dado.

5.8 Componentes de Terceros

Para agilizar el desarrollo de la aplicación se buscaron componentes de software libre que cumplieran algunas de las tareas que se debían realizar. Los componentes de terceros que se utilizaron son los siguientes: boostrap como css framework, Jquery como framework para JavasScript y Jplot para dibujar las gráficas del informe.

5.8.1 Boostrap CSS

Bootstrap css[1] es una colección de herramientas para aplicaciones web, contiene una serie de plantillas css que definen la apariencia de los elementos de la página. Además cuenta con una librería de componentes que incluye: botones, menús desplegables,barras de navegación entre otros componentes de la interfaz de usuario.

El *EHReport Plugin* utiliza las plantillas css de boostrap para definir la apariencia de la mayoría de los elementos en la interfaz de usuario. Sin embargo en algunos casos fue necesario sobrescribir las reglas css de boostrap para dar una apariencia personalizada a algunos elementos. Las reglas css especificas para el proyecto se almacenaron en el archivo *report.css*.

5.8.2 Jquery y Jplot

Como framework para escribir el código de JavaScript se optó por *Jquery*[2]. *Jquery* es una biblioteca que simplifica la manera de interaccionar con los documentos HTML. Cuenta con una sintaxis propia para definir la manera en que JavaScript modifica el contenido de una página. En el proyecto se utilizó para las siguientes funciones: la navegación entre secciones, la ejecución de filtros, las páginas de participantes y preguntas y los menús desplegables.

Jquery además dispone de muchos *plugins* creados por terceros que facilitan las tareas más comunes para el desarrollo de una página. Uno de estos *plugins* es *JPlot*[3] , que se utilizó para dibujar las gráfica de la sección G*rading Key.*

5.9 Licencias del Código

La licencia elegida para el proyecto fue la *GPLv3*[4]. Una revisión de los componentes externos que se utilizan muestra que no existen problemas de compatibilidad. La tabla 5.8 muestra los componentes que conforman el *EHReport Plugin* y la versión y licencia de cada uno de ellos.

1 http://getbootstrap.com/css/
2 http://jquery.com/
3 http://www.jqplot.com/
4 Se puede consultar el texto original de la licencia(en ingles) en: http://www.gnu.org/licenses/gpl.html o una traducción no oficial al español en:http://hjmacho.github.io/translation_GPLv3_to_spanish/

Componente	Version	Licencia
EHReportPlugin	1.0	GPLV3
Zend Framework	1.2	BSD modificada
Jquery	1.10.2	MIT
Jplot	1.0.8r1250	MIT
Bootstrap CSS	2.3.2	Apache 2.0

Tabla 5.8: Licencia de los componentes

5.10 Aseguramiento de La Calidad

En la etapa de diseño se actualizó el plan de pruebas para incorporar las nuevas especificaciones del sistema. Para la fase de desarrollo se estableció como requisito realizar pruebas unitarias para verificar el correcto funcionamiento de los módulos y clases que integran la aplicación.

La calidad del producto final se determina durante la fase de validación, que consiste en realizar diversas pruebas al sistema para asegurar que cumple con los requisitos establecidos. Este etapa normalmente se realiza con un proceso iterativo de pruebas y corrección de errores encontrados hasta alcanzar la versión final del producto. Las pruebas que se planearon para la etapa de validación fueron las siguientes:

- Pruebas de Integración: Son las pruebas funcionales y no funcionales que verifican el funcionamiento del *plugin* una vez que se ha instalado en EvaExam.
- Pruebas de Derivados: Prueban el correcto funcionamiento del *plugin* en ClassExam.
- Pruebas de Desempeño o Performance: Son las pruebas que se realizan para garantizar que el plugin opera correctamente cuando el volumen de los resultados es muy grande.
- Pruebas Compatibilidad con los navegadores: Son pruebas funcionales que se realizaron en todos los navegadores web con los que debe ser compatible el *plugin*, sirven para verificar que el *plugin* proporciona el mismo funcionamiento en todos ellos.

6 Desarrollo y Validación

En esta sección se presenta un resumen de las actividades que se llevaron a cabo en las etapas de desarrollo y validación del producto. El objetivo de la fase desarrollo es escribir el código de la aplicación de acuerdo a las especificaciones de diseño. El objetivo de la fase de validación es evaluar la calidad del producto y determinar cuando esta listo para su liberación.

En esta fase se alcanzaron los hitos *feature freeze* y *feature complete* ,y al final de la misma, después de un proceso de un proceso de refinamiento del software, se consiguió la versión final del producto.

6.1 Proceso De Desarrollo y Validación

El proceso de desarrollo de software para el *EHReport Plugin* siguió un modelo en cascada como ya se ha mencionado anteriormente. En este modelo el ciclo de liberación de software incluye varias etapas y comienza normalmente con la liberación de las versiones iniciales del programa, cada versión de prueba pasa por un proceso de validación para encontrar posibles fallos, cada determinado tiempo se libera una nueva versión de prueba que contiene correcciones o *parches* para los fallos encontrados. Este proceso de perfeccionamiento del software se realiza de forma iterativa y concluye cuando una versión pasa satisfactoriamente el proceso de validación y se convierte en la versión final del producto.

Normalmente las fases en que se divide el ciclo de liberación dependen del estado de madurez del software, que va desde un estado inicial de desarrollo hasta la versión final. Para la mayoría de los proyectos de software, el ciclo de liberación incluye las siguientes fases:

- *Alpha*: Es esta fase el software es normalmente inestable y puede ocasionar perdida de datos. El tipo de pruebas usual para esta fase es de caja blanca o *white box test*.
- *Beta:* Generalmente comienza cuando se ha alcanzado el hito *Feature Complete*, es decir cuando se ha desarrollado el código al cien por ciento. El software aún contiene muchos errores, aunque es más estable que las versiones *alpha*.
- *Release Candidate* o *RC*: Es una versión que se considera lo suficientemente estable como para poder convertirse en la versión final. Esta versión aún pasa por el proceso de validación, de encontrarse errores significativos estos se tienen que corregir y se libera una nueva versión. Cuando una versión RC aprueba el proceso de validación se convierte en la versión final.
- *GoldMaster*: Es la versión final del software empaquetada y lista para su liberación al usuario final.

Para el desarrollo del *EHReport Plugin* se siguió en términos generales el procedimiento descrito anteriormente, con la excepción de que dada la sencillez del proyecto no existieron oficialmente versiones de prueba *alpha*, y la primera versión de prueba se etiquetó como Beta1.

La figura 6.1 muestra un diagrama que resume el proceso de desarrollo y validación para el *EHReport Plugin*. La etapa de desarrollo corresponde a la escritura del código e incluye las pruebas unitarias. La fase de validación comenzó con la liberación de la versión Beta1, incluyó las pruebas de integración y de compatibilidad, y concluyó con la versión final. La etapa de corrección de errores se realizó tanto en el desarrollo como en la etapa de validación, ya que después de las pruebas unitarias también existió una proceso de reparación de los fallos encontrados.

En la línea de tiempo también se puede observar la ubicación aproximada de los hitos del proyecto: *fetaure freeze y feature complete*. *Feauter freeze* que es el punto a partir del cual las especificaciones del producto deben permanecer sin cambios,y se alcanzó justo al comienzo de la fase de desarrollo. *Feature complete* se alcanza cuando todas las funciones del software han sido implementadas.

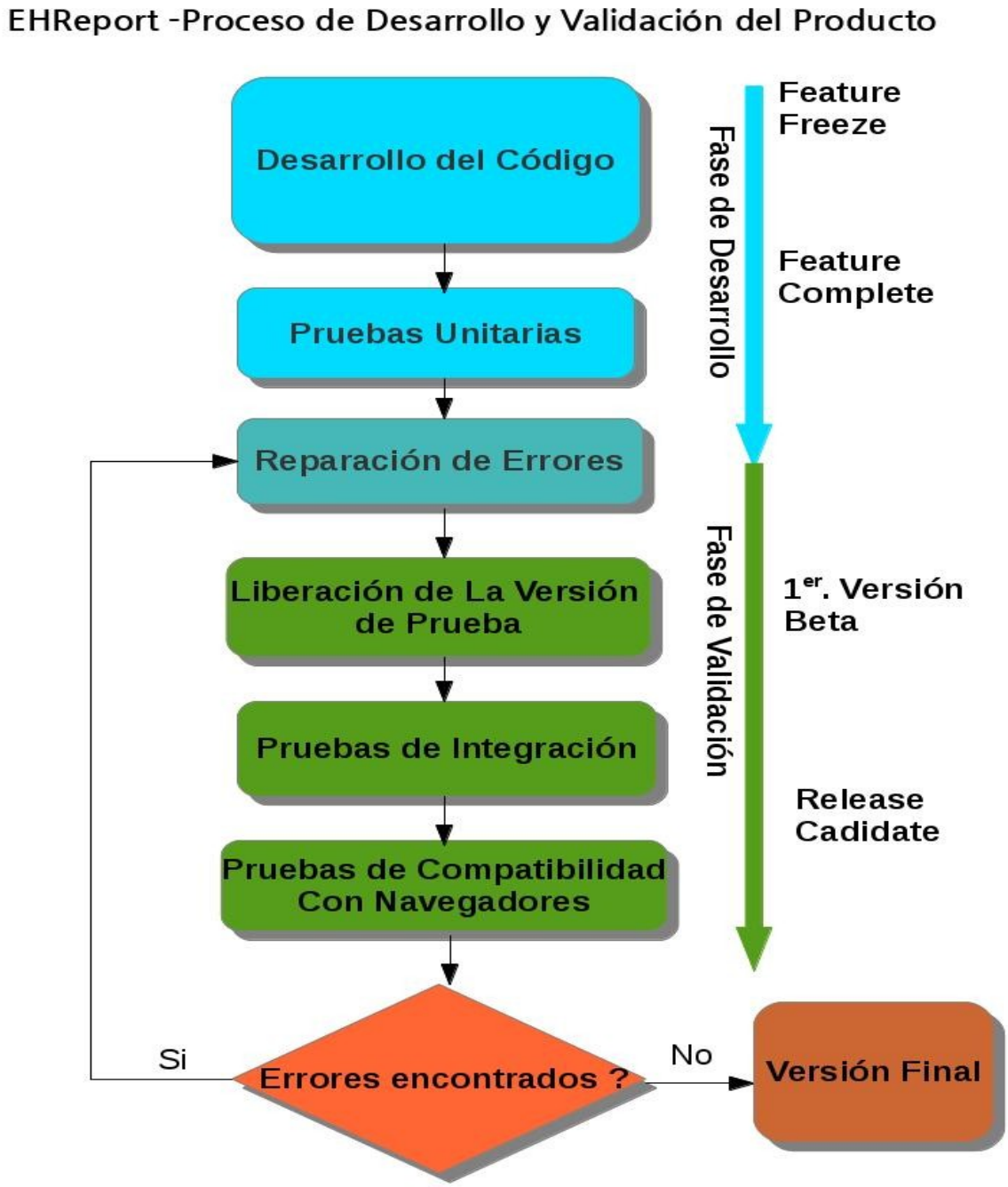


Figura 6.1: Proceso de desarrollo y validación

6.2 Herramientas de Trabajo

Para la escritura del código y la realización de las pruebas se utilizaron algunas herramientas de software que facilitaron el trabajo. Las herramientas utilizadas fueron las siguientes:

- Netbeans 7.1: Como entorno de desarrollo integrado para el desarrollo del código.
- PHPUnit 3.7: Como plataforma de pruebas unitarias.
- VMware ESXi 4.1: Como plataforma de virtualización para realizar pruebas en distintos ambientes de trabajo.
- Firebug 1.2: Como herramienta de depuración del código HTML y JavaScript.

6.3 Desarrollo del Código

El proceso de desarrollar el código se realizó a partir de las especificaciones de la etapa de diseño y siguió un proceso basado en la arquitectura del *plugin*. Primero se construyó la interfaz de datos y el modelo, luego las librerías auxiliares y finalmente la interfaz de usuario y el controlador. El proceso de depuración del código se realizo con las herramientas que proporciona *Netbeans* y con *Firebug* del lado del cliente.

Cuando se completó la implementación de todas las funciones, es decir al alcanzar el hito *Feature Complete*, se comenzó con la implementación y ejecución de las pruebas de unidad para todas las clases que componen el *plugin*,se corrigieron los errores encontrados y finalmente se liberó la primera versión de prueba Beta1.La tabla 6.1 muestra los archivos que componen el código fuente.

Componentes creados durante el proyecto	
Archivo	**Descripción**
EHCoreUtils.php	Libreria auxliar
EHGrade.php	Componente del modelo
EhGradingKey.php	Componente del modelo
EHMathUtils.php	Libreria auxliar
EHParticipantResult.php	Componente del modelo
EHReport.php	Componente del modelo
EHReportMapper.php	Componente del modelo
EHSoapClient	Cliente soap
EHReportSoapMapper.php	Componente del modelo
EhtmlReportController.php	Controlador
display.phtml	Vista
index.phtml	Vista
maResultTable.phtml	Componente de la vista
questionTableBody.phtml	Componente de la vista
questionsTable.phtml	Componente de la vista
report.css	Archivo css
ehreport.js	Script de la vista
ehreport.phtml	layout
ehreport.de_edu.inc	Archivo de idiomas
ehreport.en_edu.inc	Archivo de idiomas

Componentes de Terceros	
Archivo	**Descripción**
bootstrap.css	Archivo css de boostrap
jquery.jplot.css	Archivo css de jplot
bootstrap.js	Script de bootstrap
jplot.DonutRenderer.js	Script de jplot
jplot.pie.js	Scipt de jplot
Jquery-1.10.2.min.js	Jquery
Jquery-jqplot.min.js	Jplot

Tabla 6.1: Componentes de Software

6.3.1 Estándares para la Escritura del Código

Para mejorar la estructura y legibilidad del código generado se siguió el estándar para la escritura de código en PHP recomendado por los creadores del *framework Zend*, que es el mismo que se utiliza para el código de *EvaExam*[1].

6.3.2 Documentación del Código

En los comentarios del código se utilizó el estándar *PhpDoc*[2], que define reglas para la estructura y sintaxis de los comentarios. Para generar la documentación del código a partir de los comentarios se utilizo la librería *PHPDocumentatorv2*[3] , que reconoce el estándar *PhpDoc* y genera automáticamente la documentación en distintos formatos. La documentación del código generada con *PhpDocumentator* se puede encontrar en el repositorio oficial del proyecto:

https://github.com/codergolem/EvaExam-HTML-Report

6.3.3 Sistema de Control Versiones

Para llevar un control de versiones sobre el código se crearon dos repositorios, uno local proporcionado por la empresa y que utiliza *svn* como herramienta de versiones y el segundo en el sitio web *GitGub*[4] .

6.4 Pruebas de Validación

El objetivo de las pruebas de validación es garantizar que el producto cumpla con los requisitos de diseño y las condiciones generales establecidas para su liberación. El resultado de las pruebas de validación sirve para determinar la madurez del software y decidir cuando es el momento en que se puede poner a disposición del usuario final.

Las pruebas de validación buscan evaluar todos las características de un programa de computadora, incluyendo los aspectos funcionales y no funcionales, interfaz del usuario, desempeño o *performance*, idiomas, etc.

A continuación se presenta de forma resumida los aspectos que se evaluaron durante el proceso de validación del *EHReport Plugin*:

- Integración con *EvaExam*: Visibilidad del enlace al plugin, lista de exámenes para un examinador, funciones generales.
- Consistencia e integridad de los resultados del examen: puntos obtenidos por cada participante, puntos obtenidos en cada pregunta, nota obtenida, resultado final, estadísticas generales, estadísticas por nota.
- Consistencia de los datos del examen: nombre, examinador, tasa de respuesta, nombre del formulario.
- Filtros: Ejecución de filtros a partir de la nota y el resultado final.
- Marcas: Logotipo en el encabezado de acuerdo al derivado del producto, ya sea *EvaExam* o

1 Los lineamientos de este estándar se pueden consultar (en ingles) en el siguiente enlace: http://framework.zend.com/manual/1.12/en/coding-standard.html

2 http://en.wikipedia.org/wiki/PHPDoc

3 http://www.phpdoc.org/

4 *GitHub (*https://github.com/) es un repositorio púbico muy popular para proyectos de software libre.

ClassExam, nota de derechos de autor.

- Interfaz de usuario: Layout, colores, navegación a través de las distintas secciones, páginas de participantes, paginas de preguntas, gráficas de notas.
- Seguridad y permisos de usuario : Disponibilidad del informe según el tipo de usuario en *EvaExam.*
- Estabilidad : Correcto funcionamiento del plugin frente a diversos grupos de datos y en distintos ambientes de trabajo.
- Rendimiento: Funcionamiento adecuado con cantidades de datos muy grandes, específicamente más de 200 participantes en un examen, o más de 200 preguntas.
- Idiomas: Interfaz en ingles o alemán de acuerdo al idioma del examinador.
- Compatibilidad con navegadores.

El tipo de pruebas que se realizaron en la etapa de validación puede considerarse como de *caja negra o black box test*, y su ejecución fue enteramente manual.

6.4.1 Plan de Pruebas

La especificación de las pruebas de validación consiste en guiones o planes de prueba que están formados por casos de prueba que evalúan algún aspecto del programa. El plan de pruebas para el *EHReport Plugin* se presentó por primera vez en la fase de análisis de requisitos, este se actualizó y extendió conforme se fue detallando el diseño del programa. Al final de la etapa de desarrollo se realizó la versión final que se utilizó para la validación del producto. Los casos de prueba para evaluar el *EHReport Plugin* se distribuyeron en los siguientes documentos:

- El plan de pruebas general: Cuenta con 44 casos de pruebas que verifican todos las característica del software especificadas anteriormente.
- El plan de compatibilidad con navegadores: Cuenta con 10 casos de prueba y evaluá las funciones e interfaz de usuario del *EHReport Plugin* en todos los navegadores con que tiene que ser compatible.

El resultado final de la ejecución de un plan de prueba está determinado por la severidad de los errores encontrados (véase *el punto 6.4.3 Errores y Bugtracking*) y puede ser de tres tipos:

- *passed o aprobado* : Si no se encontró ningún error.
- *passed with issues o aprobado con errores*: Si ninguno de los errores encontrados tuvo una severidad mayor a *minor.*
- *fail o no aprobado*: Si se encontró por lo menos un error con severidad *medium* o mayor.

6.4.2 Ambiente de Pruebas

Para cubrir todos los escenarios posibles el *EHReport Plugin* se evaluó en los siguientes ambientes de trabajo.

- EvaExam version 6.0 2000 , servidor web Apache 2.2, base de datos MySQL Server 5.1, sistema operativo Windows Server 2008 R2
- ClassExam version 6.0 200, servidor web Microsoft IIS, base de datos Microsoft MSSQL

Server 2010, sistema operativo Windows Server 2008 R2.

6.4.3 Errores y *Bugtracking*

Para llevar un registro de los errores o *bugs* encontrados durante las pruebas de validación se utilizó un sistema de seguimiento de errores o *bug tracking systemm*, que es una plataforma de software que facilita el registro, seguimiento y la clasificación de los errores encontrados. La herramienta de software que se utilizó para el registro de los errores encontrados fue la que proporciona el repositorio *GitHub*.

Dependiendo de la severidad del fallo encontrado los errores o *bugs* se clasificaron de la siguiente manera:

- *Tweak*: Error menor que no afecta el funcionamiento del programa, y que pude ser o no reparado.
- *Minor:* Error menor que afecta alguna funcionalidad del programa y que normalmente tiene que ser reparado.
- *Medium:* Error que afecta parcial o totalmente la funcionalidad de un programa y que tiene que ser reparado.
- *Major*: Error grave en la funcionalidad de un programa, y que puede ocasionar perdidas de datos.
- *Blocker:* Error grave que afecta todas las funcionalidades de un programa haciéndolo inutilizable.

6.4.4 Resultados de La Validación y Versión Final

El proceso iterativo que se describió en la punto 6.1 requirió de dos versiones beta y dos versiones RC hasta alcanzar una versión libre de errores. La versión *RC2* se convirtió en la versión final. Los resultados de las pruebas y errores encontrados durante las pruebas se resumen en la tabla siguiente.

Versión	Tweak	Minor	Medium	Major	Blocker	Totál	Resultado
Beta1	0	0	0	2	1	3	**Fail**
Beta2	0	1	3	2	0	6	**Fail**
RC1	1	2	0	0	0	3	**Passed with issues**
RC2	0	0	0	0	0	0	**Passed**

Tabla 6.2: Versiones de prueba y resultados de la validación

La tabla muestra el número de errores encontrados en cada versión, clasificados por su severidad. Se puede observar por la severidad de los errores encontrados: dos errores *major* y uno *blocker*, que la versión Beta1 aún era muy inestable. La versión Beta2 fue un poco más estable y permitió extender el alcance de las pruebas. En la versión *RC1* se encontraron solamente errores de severidad *minor* o *tweak* por lo que en este punto el software estaba casi listo para ser liberado. Finalmente la versión *RC2* pasó el proceso de validación libre de errores. Es notorio como la estabilidad y madurez del software fue aumentado con cada versión hasta llegar a la versión final.

6.4.5 Criterio de Validación

En el caso del *EHReport Plugin* se estableció como primer criterio de validación que el producto aprobara sin errores (*passed*) los dos planes de prueba existentes. Cuando se contó con una versión del software libre de errores se convocó a un *panel de control de liberación o release control board,i*ntegrado por todos los involucrados en el proyecto. Este panel evaluó los resultados de las pruebas de validación y la madurez del software y tomó la decisión de que el software estaba listo para su liberación.

6.5 Empaquetado *y distribución*

La versión final empaquetada para su distribución se conoce como *GoldMaster, y* se liberó al usuario final por medio del repositorio del proyecto creado en GitHub.

Como php y JavaScript son lenguajes interpretados o *script*, no existe diferencia entre el código fuente y el ejecutable de la aplicación, por lo que el paquete de instalación del EHReport Plguin sirve tanto al usuario final como a quien quiera utilizar el código para ejercer cualquiera de las libertades que garantiza la licencia libre del código.

El paquete del software es un archivo *zip* que contiene: todos los archivos que conforman el *EHReport Plugin*,un archivo tipo *readme.txt* con las instrucciones de instalación, la nota de derechos de autor y licencia del proyecto, y el texto de las licencias del software.

7 Resultados y Conclusiones

En este último capitulo de la memoria se presentan los resultados generales del proyecto y las conclusiones finales. El objetivo definido al comienzo del proyecto se cumplió satisfactoriamente, se siguieron todas las fases planificadas, la liberación del producto se dio en la fecha planeada y el número de horas que se estimaron para la realización del proyecto difirió del número de horas totales requeridas sólo marginalmente. El resultado es el *EvaExam HTML Report Plugin* ,un software funcional y estable, que ofrece una alternativa moderna para presentar los resultados de exámenes en *EvaExam.*

7.1 Recursos Consumidos

Como ocurre en casi todos los proyectos de desarrollo software existieron diferencias entre el número de horas originalmente calculadas para cada fase y las horas requeridas en la realidad, sin embargo estas diferencias se mantuvieron dentro del límite aceptable, se alcanzaron todos los hitos marcados, la mayoría de ellos en las fechas esperadas, y el producto se liberó en la fecha planeada.

El número de horas total para realizar el proyecto que se estimó al comienzo fue de 364 y el número de horas finalmente requerido fue de 360, por lo que la diferencia fue mínima. Si se observa la tabla 7.1, donde se muestra el tiempo requerido en cada etapa en comparación con el tiempo originalmente planificado, se puede ver que la mayoría de las etapas no tuvieron diferencias mayores al 10% respecto de las horas originalmente consideradas. Sin embargo las etapas de diseño y validación difirieron significativamente de la planificación original.

En el caso de la etapa de diseño la diferencia se debió a que se requirió invertir más horas en el aprendizaje de los fundamentos teóricos y prácticos de la arquitectura *mvc* y del *framework Zend*. Este proceso de familiarización, necesario para realizar la etapa de diseño, resultó más complejo de lo esperado por lo que terminó impactando sustancialmente en la duración total de esta fase.

Contrariamente al caso anterior, la etapa de validación requirió menos recursos de los esperados. Originalmente se estableció el tiempo necesario para la etapa de validación a partir del tiempo estimado para la etapa de desarrollo, utilizando un relación de 1.5 horas de validación por hora de desarrollo, para garantizar la calidad del producto. Sin embargo como se sabe el tiempo necesario para alcanzar una versión estable del producto varía dependiendo de la naturaleza del software. En este caso el proceso de validación concluyó cuando consiguió una versión del software que cumplía con todos los criterios de calidad establecidos, lo cual ocurrió antes de lo originalmente pensado.

En el balance final estos cambios no afectaron el total de los recursos necesarios ya que las horas extra de la etapa de diseño se compensaron con las horas que se ahorraron en la etapa de validación. Esto también significa que el número de horas adicionales invertido en las etapas previas produjo un versión de prueba más robusta y estable, lo que redundó en un proceso de validación más corto.

Número de horas por etapa		
Etapa	**Horas originalmente planeadas**	**Horas totales requeridas**
Definición general del proyecto	10	10
Análisis	24	30
Diseño	40	60
Desarrollo	110	120
Test	140	95
Documentación	40	45
Total de horas:	**364**	**360**

Tabla 7.1: Distribución de horas por fase del proyecto

7.2 Aspectos Técnicos y Legales

Desde el punto de vista técnico el proyecto permitió poner en práctica los conocimientos sobre desarrollo de aplicaciones web aprendidos durante el máster. Se utilizaron las tecnologías más comunes para el desarrollo de aplicaciones de Internet como:php, hmtl y javascript. Además se fortalecieron los conocimientos sobre arquitectura de software, y se aprendieron los principios del diseño basado en el paradigma *mvc* y el *framework zend*, lo que hoy en día se puede utilizar para crear sistemas web de gran escala.

El proyecto también sirvió para comprender mejor de forma práctica y directa las implicaciones legales de utilizar software libre. Dado que el proyecto cuenta con componentes desarrollados por varios autores, y distribuidos bajo distintas licencias de software libre, fue necesario realizar un análisis sobre la compatibilidad entre licencias para asegurar que no se violaba ninguna clausula de ellas.

7.3 Futuro del Proyecto

El proyecto ha sido una primera experiencia en el desarrollo de software libre para la empresa, y es incierto si existirán más desarrollos de este tipo. Una posibilidad discutida es la distribución como software libre de algunas de las aplicaciones que la empresa desarrolla para uso interno, como una contribución a la comunidad del software libre, lo que pudiera retribuir en la imagen de la empresa.

Al momento de escribir este documento aún no existe un definición clara de que uso le dará la empresa al producto generado. Existen dos posibilidades: que el *EHReport Plugin* se ofrezca a los clientes en forma *add-on*, para extender las funciones de programa. La segunda es integrar el código del proyecto en EvaExam, con lo que el *EHReport* se convertiría en una función estándar de la próxima versión de *EvaExam.*

En el primer caso el *plugin* llegaría de manera gratuita a los clientes, como una consecuencia indirecta de la licencia de software libre, aunque podrían obtenerse ganancias por el cobro de brindar asesoría técnica sobre el uso y configuración del *plugin*, que la empresa puede integrar fácilmente con el resto de los servicios incluidos en el contrato de soporte.

Por otro lado, si el código se integra con *EvaExam* , se tendría que licenciar bajo los términos de la licencia privativa de *EvaExam*. Legalmente no existe impedimento para esto pues como se mencionó la empresa es propietaria de los derechos patrimoniales y puede distribuir el código bajo las licencias que considere convenientes. Por lo que el EHReport Plugin se distribuiría en este caso bajo los términos de dos licencias, una la GPLv3 y otra privativa.

8 Bibliografía

Java 2 Interfaces Gráficas y Aplicaciones para Internet
Fco. Javier Ceballos
Alfaomega Grupo Editor S.A de C.V, México , 2008.

jQuery Cookbook
Cody Lindley
O´Really Media, 2009

php|architects´s Guide to PHP Design Patterns
Jason E. Sweat
php|architect nanobooks, 2005

Pro PHP Programming
Peter MacIntyre,Brian Danchilla, and Mladen Gogala
Apress, 2010

Programming Web Services with SOAP
James Snell, Doug Tidwell, Pavel Kuchenko
O´Really Media, 2001

Rapid Development
Steve McConnell
Microsoft Press 1996

The Bussines of Software
Michael A. Cusumano.
Simon and Schuster Inc. 2004

UML 2.0 in a Nutshell
Dan Pilone, Neil Pitman
O´Really, 2005

Understanding Open Source and Free Software Licensing
Andrew M. St. Laurent
O´Really Media, 2004

Various Licenses and Comments about Them *<en línea>*

Free Software Foundation ,2012

<http://www.gnu.org/licenses/license-list.en.html>

Zend Framework in Action
Rob Allen, Nick Lo, Steven Brown
Manning Publications, 2009

Lista de Anexos

A continuación se presenta la lista de los anexos que acompañan este documento:

Anexo I : Test Plan

Archivo : Anexo_I_TestPlan.pdf

Este documento consiste en el plan de pruebas que se utilizó para el proceso de validación del EvaExam HTLM Report.

Anexo II : EvaExam Soap Requests Samples

Archivo: Anexo_II_EvaExamSoapRequestsSamples.pdf

Este documento incluye ejemplos de las peticiones y respuestas a los métodos *soap* que se utilizaron en el código del EvaExam HTML Report.

Anexo III : EHReport Mapper and Model Class Diagram

Archivo: Anexo_III_MapperAndModelDiagram.pdf

Este documento contiene el diagrama de clase UML de los componetes que conforman el EvaExam HTML Report.

Anexo IV: Glosario de Software

Archivo: Anexo_IV_GlosarioDeSoftware.pdf

Contiene referencias sobre la mayoría de los programas mencionados en esta memoria.