SEPROSO.
Arquitectura del sistema.

Francisco Javier Delgado del Hoyo
Yuri Torres de la Sierra
Rubén Martínez García
Abel Lozoya de Diego

Diciembre, 2008

Revisiones del documento

Historial de revisiones del documento

VERSIÓN	FECHA	DESCRIPCIÓN	AUTOR
0.1	19/11/08	Objetivos y visión general.	Francisco
1.0	01/11/08	Estandarización según el modelo de UPEDU.	Rubén
1.5	01/11/08	Diagrama de despliegue añadido.	Rubén y Francisco

Indice

Indice de figuras

Indice de tablas

Capítulo 1

Introducción.

1.1 Propósito.

En este documento se presenta una visión general de la arquitectura del sistema, utilizando diferentes vistas para destacar en cada una los aspectos más importantes del sistema. Se intenta capturar y mostrar las decisiones arquitectónicas significativas tomadas en el sistema, que influirán significativamente en el resto del diseño.

La especificación está destinada a los diseñadores y es un artefacto de análisis y diseño del modelo de proceso UPEDU muy importante, ya que el modelo está centrado en la arquitectura. Siempre que se desarrolle un nuevo subsistema, clase o interfaz debe hacerse respetando la arquitectura básica del sistema.

1.2 Ámbito.

La arquitectura presentada corresponde a la herramienta on-line SEPROSO, que se alojará en un servidor PHP y será utilizada por el cliente mediante un navegador Web. Permite el seguimiento y control de proyectos software tanto por parte del jefe de proyecto como el jefe de personal.

Es un documento esencial para el diseñador la hora de diseñar las pruebas más convenientes para el sistema, y surge del análisis de la arquitectura en la fase de elaboración.

1.3 Definiciones.

Véase el Glosario.

1.4 Referencias.

- Otra documentación de SEPROSO:
- Glosario, Grupo 3.
- Especificación de Casos de Uso, Grupo 3.
- Especificación Suplementaria, Grupo 3.
- Plan de Desarrollo Software, Grupo 3.
- Plan de Iteración, Grupo 3.
- UPEDU. http://www.upedu.org/upedu/

1.5 Visión general.

En las siguientes secciones veremos con más detalle algunos aspectos de la arquitectura que se representan cada uno en una vista diferente. Veremos un esquema global, una vista de casos de uso y una vista lógica, tras la cual se describirán más detalladamente las clases de diseño que forman esa vista. En todas ellas utilizaremos como lenguaje de representación UML, si no se especifica ningún otro.

Finalmente se realiza una descripción de los elementos principales que componen la interfaz de usuario y de los factores de calidad que han influenciado la decisión de escoger esta arquitectura. Acabaremos haciendo una descripción de la metodología utilizada para medir la calidad de este artefacto.

Capítulo 2

Representación arquitectónica.

Utilizaremos una arquitectura de tres capas: presentación, negocio y datos. Es la más tradicional y sencilla para una herramienta que sólo tiene un nivel: todos los componentes están en el servidor PHP. Cada capa sólo puede comunicarse con la inmediatamente superior o inferior.

Debido a esta simplicidad sólo incluiremos dos vistas: casos de uso (sólo contiene paquetes y casos de uso) y lógica (paquetes, interfaces y clases) que son estrictamente imprescindibles. Utilizaremos para la representación los diagramas de paquetes obtenidos de StarUML que a su vez utilizan el lenguaje de modelado UML.

Capítulo 3

Restricciones y objetivos arquitectónicos.

SEPROSO es una herramienta Web que se ejecuta en modo Cliente-Servidor, pero donde el cliente es un navegador Web común y el por tanto el proyecto consiste tan sólo en desarrollar la parte del lado del servidor como si fuera stand-alone. Puede ser accedida públicamente (desde cualquier ubicación) lo que hace necesaria la existencia de cuentas y de login para acceder a su funcionalidad.

La ejecución mediante un navegador Web hace que sea totalmente compatible con cualquier tipo de sistema operativo y hardware. Los datos se almacenarán fuera de ejecución en una base de datos remota MySQL, lo que obliga a utilizar una capa de acceso a datos, y la interfaz de usuario es una Web, lo que hace necesaria una capa de visualización o presentación debido a la complejidad de su diseño. La existencia de múltiples roles, donde cada uno tiene que tener una interfaz propia hace que esta capa contenga varias vistas diferentes. Para que sea compatible con cualquier navegador / servidor utilizaremos hojas de estilo CSS 3.0 estándar.

Como restricciones imponemos que:

- El lenguaje de programación será PHP 5.0, que incluye facilidades para acceder a MySQL.
- Emplearemos el framework PRADO PHP que contiene algunas clases PHP muy útiles y que utilizan la arquitectura de tres capas.
- La herramienta debe funcionar en el host jair.lab.fi.uva.es y las tres capas son la solución más segura.
- El equipo está formado por 4 implementadores y con tres capas es más sencillo separar el trabajo.

Capítulo 4

Vista de Casos de Uso.

En esta vista se incluyen los Casos de Uso / Escenarios del modelo que representan una funcionalidad básica y central del sistema. También se incluyen aquellos que tienen una cobertura sustancial de la arquitectura (influencian a muchos elementos arquitectónicos) o que ilustran un punto específico y delicado de la arquitectura.

Además la Vista de Casos de Uso es importante para seleccionar aquellos Casos de Uso en los cuales se debe centrar el desarrollo de una iteración concreta (recordemos que este modelo de proceso está guiado por esos Casos de Uso).

Véase el documento de Especificación de Casos de Uso para más detalles.

4.1 Realización del caso de uso.

Con la realización del siguiente Caso de Uso se intenta mostrar un ejemplo de cómo trabaja la herramienta gracias a la arquitectura planteada. Esto se consigue viendo cómo se utilizan y se relacionan los distintos elementos del diseño lógico que presentaremos en la siguiente sección.

Véase el documento de Realización de Casos de Uso para más información.

Capítulo 5

Vista lógica.

En esta sección se describen las partes del modelo de diseño del sistema que son significativas para la arquitectura, como son sus subsistemas internos y paquetes. Por cada paquete significativo se muestra su composición interna. Para las elementos internos más relevantes y complejos se realiza una descripción de responsabilidad, relaciones con otras, operaciones y atributos.

5.1 Visión general.

En la siguiente imagen se muestra la vista lógica del sistema descomponiendo el modelo de diseño en una colección de paquetes relacionados jerárquicamente. Agrupando los paquetes desde un punto de vista lógico (funcional) y según las dependencias que muestra la jerarquía, tenemos las capas de la arquitectura.

5.2 Diagrama de despliegue.

El diagrama anexo 5.1 muestra los nodos disponibles y los sistemas que los soportarán:

5.2.1 Nodos

Nodo: PC Cliente.
Sistemas soportados: Navegador web.
Dependencias: Nodo Servidor Aplicación: JAIR.
Descripción: Es un ordenador de sobremesa con cualquier sistema operativo capaz de ejecutar un navegador web mozilla o compatible, y conexión a una redcon acceso al nodo servidor.

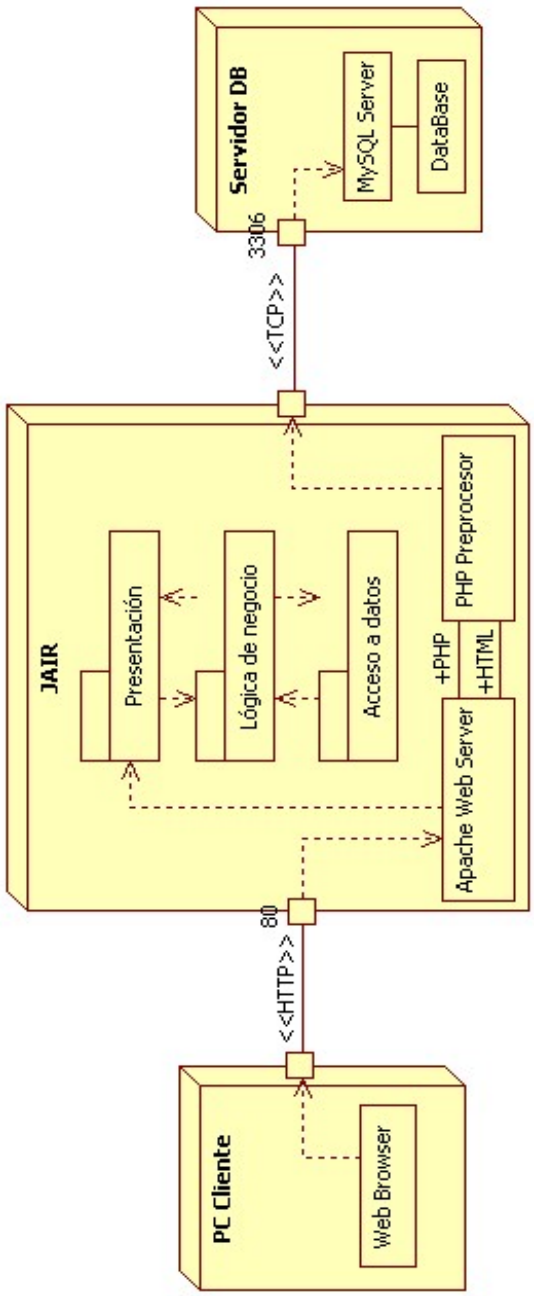

Figura 5.1: Diagrama de despliegue.

Nodo: Servidor aplicación.
Sistemas soportados: Aplicación y servidor web.
Dependencias: Base de datos MySQL.
Descripción: Un equipo informático servidor donde almacenar y ejecutar la aplicación. Deberá disponer de un servidor web.

Nodo: Servidor de Base de datos.
Sistemas soportados: Base de datos.
Dependencias: –
Descripción: Un equípo informático servidor que ejecuta un servicio de base de datos.

5.2.2 Componentes

Componente: Navegador WEB.
Dependencias: Componente de Servidor WEB.
Descripción: Software de pasarela entre la aplicación y el Navegador WEB. Debe soportar el protocolo HTTP.
Implementaciones: Apache 2.0.

Componente: Navegador WEB.
Dependencias: Componente de Servidor WEB.
Descripción: Software de visualización de páginas web a través de una red ip. Dicho software soportará las tecnologías HTML, XHTML, CSS y PHP.
Implementaciones: Firefox, Internet y Opera.

Componente: Aplicación.
Dependencias: Componente de Servidor WEB.
Descripción: Sistema software a implementar que cumple la totatildad de requisitos especificados por el cliente. Aplicación basada en PHP realizada con Prado PHP.

Componente: Base de datos.
Dependencias: –.
Descripción: Software de base de datos para el almacenamiento persistente de la información.
Implementaciones: Utilizaremos MySQL Community Server.

5.3 Paquetes de diseño significativos.

A continuación se especifica más detalladamente la composición de cada uno de los paquetes del modelo de diseño presentado en la sección anterior. Para cada uno se incluye una nueva subsección que contiene una breve descripción del paquete y un diagrama de contenidos que muestra sus clases y subpaquetes. Después se describe también las clases más significativas internas al paquete.

Capítulo 6

Descripción de la interfaz.

Las clases de gestión de la interfaz de usuario manejan formatos de visualización en pantalla, entradas válidas, salidas resultantes, etc. A continuación se describen las más importantes:

Capítulo 7

Tamaño y prestaciones.

La arquitectura utilizada soporta perfectamente los requisitos de tamaño y tiempo de respuesta en interacción con el usuario planteados inicialmente, cuando se ejecuta en modo cliente-servidor.

Ofrece buenas prestaciones en un servidor PHP de tamaño medio y ha sido diseñada para minimizar el espacio ocupado en el servidor. Soporta el almacenamiento persistente de toda la información que los posibles usuarios necesiten y es lo suficientemente ligera como para utilizarla con cualquier navegador

Capítulo 8

Calidad.

La arquitectura cumple todos los requisitos de calidad especificados por el cliente tal y como se recogen en los documentos SRS y SSS.

La arquitectura planteada aumenta la calidad del sistema en muchos factores fundamentales:

- **Extensibilidad**: la existencia de capas permite ampliar la funcionalidad de cualquiera o cambiarla sin demasiado esfuerzo ya que el acoplamiento entre capas es mínimo.
- **Fiabilidad**: la existencia de interfaces y la única comunicación posible con capas adyacentes reduce la posibilidad de errores.
- **Portabilidad**: el desarrollo en PHP permite ejecutarla en cualquier máquina con un servidor PHP que genere el código HTML tras procesar el código PHP, sin cambiar en absoluto. Además, el Gestor de Base de Datos puede cambiarse fácilmente porque la capa de datos mantendría su interfaz con la capa de negocio inalterada y sólo habría que cambiar esta última capa. Lo mismo ocurre con la interfaz de usuario que podría cambiarse a una interfaz para móviles por ejemplo.
- **Seguridad**: puesto que el acceso a los datos necesita pasar por la capa de negocio, se controla y valida al usuario antes de que éste puede llegar a sus datos, y no tiene ninguna posibilidad de acceder directamente desde la interfaz de usuario.