SEPROSO.
Modelo de Diseño.

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	22/12/08	Primera versión de modelo de diseño.	Rubén y Yuri
0.5	28/12/08	Versión 0.5 del diagrama de clases.	Francisco, Abel y Rubén
1.0	30/12/08	Versión completa del modelo de diseño estático.	Grupo III
1.1	05/01/08	Completado el modelo de diseño.	Rubén.

Indice

Indice de figuras

Capítulo 1

Introducción.

1.1 Propósito.

El objetivo de este documento es la especificación del modelo de diseño completo del sistema de SEguimiento de PROyectos SOftware (SEPROSO). En dicho documento se definen los diagramas complementarios a la realización de casos de uso, completando el modelo de diseño. Incluye el modelo de clases de diseño esquemático y los controladores previstos. Este documento será referencia para el uso posterior por los desarrolladores y en consecuencia por más documentos de planificación, completar diseño software y desarrollo de la aplicación.

Anexo a este documento se tiene

1.2 Ámbito.

El sistema a desarrollar se denominará SEPROSO. Es una herramienta para la gestión y seguimiento de proyectos software. La interación con la herramienta depende del rol a desempeñar por el usuario de la aplicación, distinguiéndose los siguientes roles:

- Administrador.
- Jefe de Proyecto.
- Desarrollador.
- Responsable de personal.

El sistema atenderá las peticiones de cada uno de ellos en función del trabajo que desempeñana
.

El modelo de análisis de uso provee una visión de la interación entre usuario y la aplicación dentro de la herramienta SEPROSO.

1.3 Definiciones.

Véase el Glosario.

1.4 Referencias.

1. Plantilla de elicitación de requisitos para Proceso Unificado UPEDU. www.upedu.org
2. Documento del estandar IEEE 830 sobre recomendaciones para la especifiación de requisitos software secciones 4, 5 y anexo A.3.
3. Visión general de la herramienta propuesta por el profesor de la asignatura.
4. Plan de Proyecto SoftwareGrupo III, Universidad de Valladolid.
5. Documento de especificación de recursos software, SRS, Grupo III, Universidad de Valladolid.
6. Modelo de Casos de UsoGrupo III, Universidad de Valladolid.
7. Diagramas de análisis realizados en la herramienta StarUMLGrupo III, Universidad de Valladolid.
8. Realización de Casos de Uso, Grupo III, Universidad de Valladolid.

1.5 Visión general.

El resto de este documento contiene los diagramas correspondientes al modelo de análisis.

Capítulo 2

Modelo de diseño.

2.1 Diagrama de Clases de Diseño.

2.1.1 Paquete ADMIN

AppControl

+calendar() Muestra el calendario general donde aparecen los dias de fiesta, trabajo y vacaciones.
+start_session(tipo, name) Inicia la sesión de tipo con el nombre de usuario indicado.
+end_session Termina la sesión de empleado actua.

TrabajadorControl

+index() Muestra la pantalla inicial de gestión de empleados. +create() Crea un empleado nuevo.
+delete() Borra un empleado. +edit() Actualiza los datos de un empleado existente.

LoginControl

+index() Muestra la pantalla de login.
+login() Hace login en el sistema.
+logout() Se desconecta del sistema.

ProyectoControl

+index() Muestra los proyectos actuales.
+create() Crea un nuevo proyecto.

InformeControl

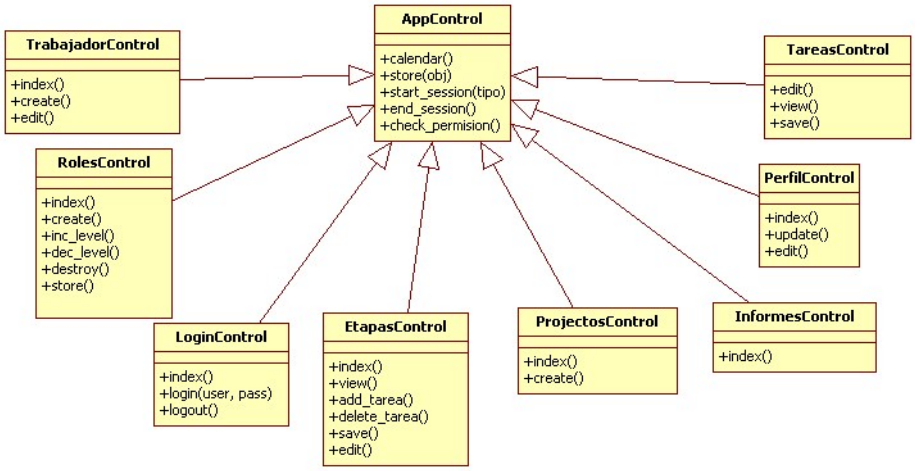

Figura 2.1: Paquede de administrador.

+index() Muestra una pantalla de acceso al sistema.

RolControl

+index() Muestra la lista de roles actuales.
+inc_level() Sube a un rol, un nivel.
+dec_level() Baja a un rol, un nivel.
+create() Crea un nuevo rol.
+destroy() Oculta un rol.

EtapasControl

+edit() Permite editar una etapa.
+view() Visualiza una etapa.
+save() Almacena los datos de una etapa.
+add_task() Añade una tarea a una etapa.
+delete_task() Borra una tarea de una etapa.

TareasControl

+edit() Altera los valores de una tarea.
+view() Visualiza los datos de una tarea.
+save() Almacena los valores de una tarea modificada.

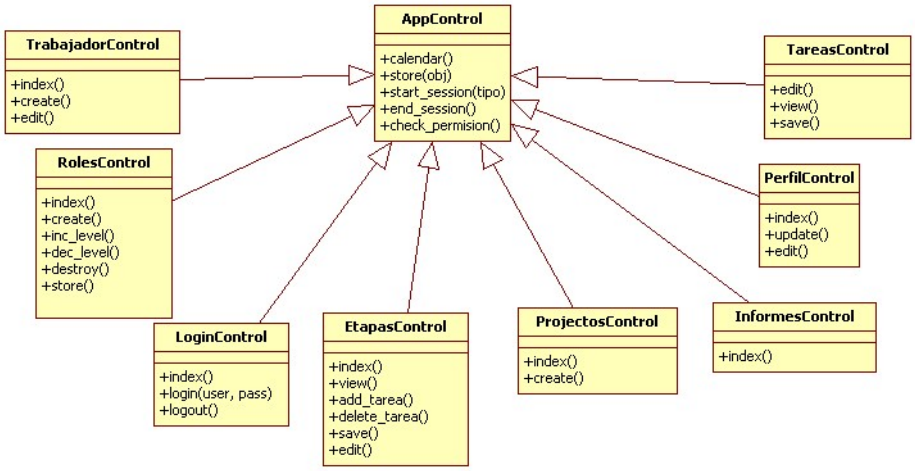

Figura 2.2: Paquede de trabajador.

2.2 Paquete Trabajador.

ActividadControl

+index() Muestra las tareas activas en las que se está asignado.
+create() Crea una nueva actividad.
+edit() Edita un informe de actividad.
+save() Almacena un informe de actividad editado
+destroy() Elimina un informe de actividad sin aceptar.

LoginControl

+index() Muestra la pantalla de login de usuario.
+login() Hace login del usuario.
+logout() Termina la sesión del usuario.

ManagerControl

+index() Muestra los proyectos a administrar por el jefe de proyectos.
+list() Muestra datos de un proyecto concreto.

PlanificarControl

+index() Muestra los proyectos para planificar.

+edit() Permite editar un proyecto.
+view() Permite ver los datos de un proyecto.
+save() Salva los datos de un proyecto modificado.
+start() Inicia un proyecto.

InformControl

+index() Muestra el índice con los informes disponibles.
+finalizados() Muestra los proyectos finalizados.

EtapasControl

+edit() Editar una etapa.
+view() Ver los datos concretos de una etapa.
+save() Salvar los datos de una etapa editada.
+add_task() Añadir una tarea a la etapa.
+delete_task() Borrar una tarea de la etapa.

TareasControl

+edit() Editar una tarea.
+view() Ver los datos concretos de una tarea.
+save() Guardar los cambios de una tarea editada.

VacacionesControl

+add() Añade un periodo vacacional.
+edit() Formulario de edición de un nuevo periodo vacacional.

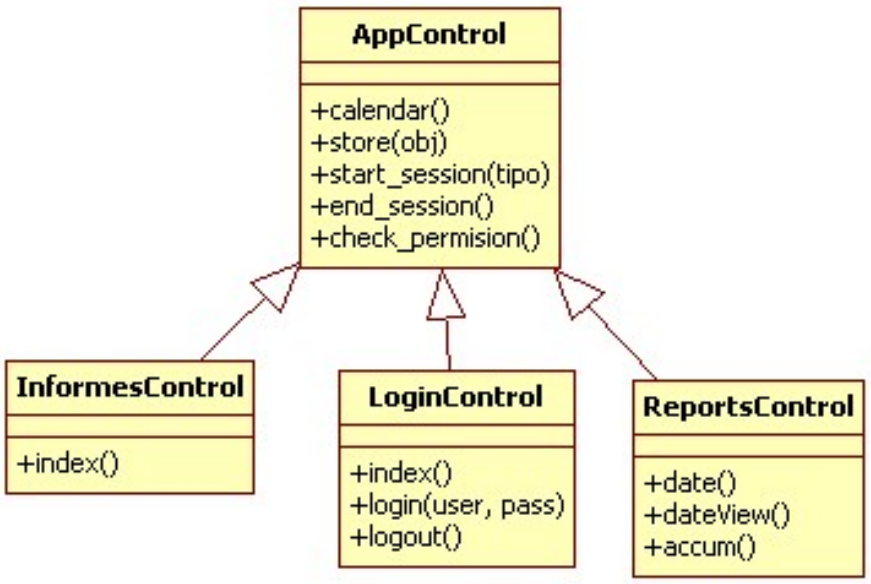

Figura 2.3: Paquede de Jefe de Proyecto.

2.3 Paquete Jefe

LoginControl

+index() Muestra la pantalla de login de usuario.
+login() Hace login del usuario.
+logout() Termina la sesión del usuario.

AppControl

+calendar() Muestra el calendario general donde aparecen los dias de fiesta, trabajo y vacaciones.
+start_session(tipo, name) Inicia la sesión de tipo con el nombre de usuario indicado.
+end_session Termina la sesión de empleado actua.

InformControl

+index() Muestra el índice con los informes disponibles.
+finalizados() Muestra los proyectos finalizados.

Capítulo 3

Vistas

Las vistas no se diseñaron, ya que por cada acción en el controlador se supone la posibilidad de tener una vista. Aquí se listan las vistas resultantes el modelo de diseño, pero existen bastantes más y se detallan posteeriormente en el modelo de implementación.

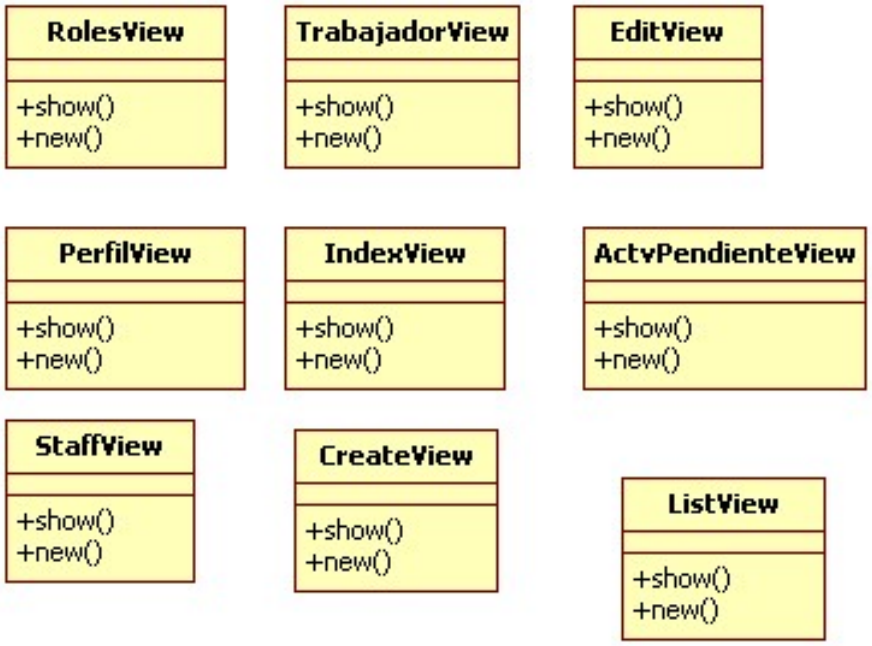

Figura 3.1: Vistas del modelo.

Capítulo 4

Modelo de clases

El modelo infiere los atributos directamente de la base de datos, que se han especificado en el esquema de la base de datos en el documento alterno. De todas formas es necesaria la determinación de una serie de métodos para recuperar la información de la base de datos y realizar diversas tareas.

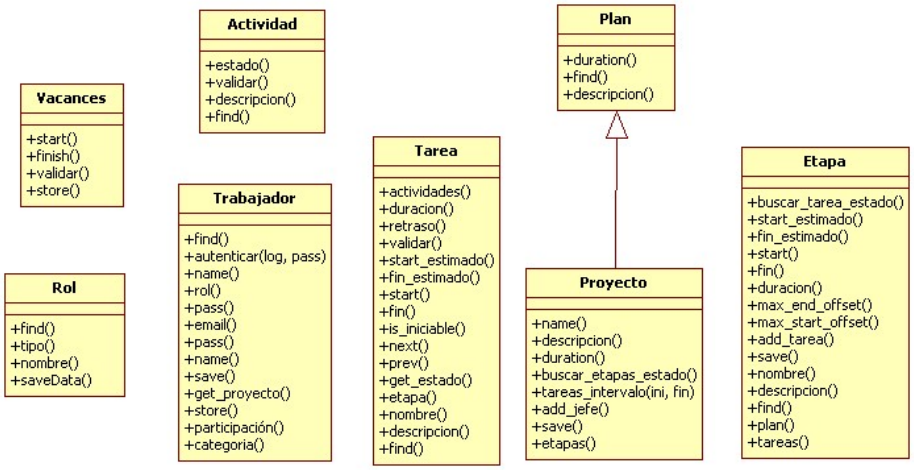

Figura 4.1: Modelo de clases.