

PRÁCTICA DE SISTEMAS INFORMATICOS III

Curso 2010 – 2011

APELLIDOS Y NOMBRE: Veamurguia Merck, Jorge; Gonzalez Muñoz, Carlos;

IDENTIFICADOR: jveamurgu1;cgonzalez509;

DNI: 45086199 N;03103200V

CENTRO ASOCIADO DE LA SESIÓN DE CONTROL: CALATAYUD

MAIL DE CONTACTO: jorge.veamurguia@gmail.com; gonzalez.karlos@gmail.com

TELÉFONO DE CONTACTO:645057755

Índice

1	Manual de Diseño	3
1.1	Arquitectura	3
1.1.1	Base de datos orientadas a objetos	4
1.2	Diseño	4
1.2.1	Diagrama de paquetes	4
1.1.	Paquete de persistencia y de base de datos	5
1.2.	Interfaz de usuario	6
1.2.2	Diagrama de clases	6
	Diagrama de clases de la persistencia	6
	Diagrama de clases de la manager	10
	Diagrama de clases del Modelo vista controlador	12
	Diagrama de casos de usos	15
	Diagrama de secuencias	16
1.2.3	Patrones empleados en la práctica	16
	Declaración de frameworks empleados	17
1.1.	Framework Sitemesh	17
1.2.	Framework Struts2	17
	Interceptors	18
	Acciones	18
	Empleo de etiquetas Struts2	19
	Validación	20
1.2.4	Implementación	20
1.2.5	Uso de la aplicación	21
2	Manual de usuario	21
2.1	Objetivo de la aplicación	21
2.2	Usabilidad	21
2.3	Funcionalidades comunes	22
2.3.1	Nuevo usuario	22
2.3.2	Edición de usuario	23
2.4	Funcionalidades del profesor	24
2.5	Funcionalidades del alumno	25
2.6	Funcionalidades del administrador	26
3	Manual de instalación	28

Índice de ilustraciones

Ilustración 1:	paquetes de la aplicación	4
Ilustración 2:	Diagrama de clase: Paquete de Persistencia:Alumno	7
Ilustración 3:	Diagrama de clase: Paquete de Persistencia:Centro Asociado	7
Ilustración 4:	Diagrama de clase: Paquete de Persistencia:Departamento	8
Ilustración 5:	Diagrama de clase: Paquete de Persistencia:Profesor	8
Ilustración 6:	Diagrama de clase: Paquete de Persistencia:Propuesta	9
Ilustración 7:	Diagrama de clase: Paquete de Persistencia:Solicitud	9
Ilustración 8:	Diagrama de clase: Paquete de Persistencia:Proyecto	10
Ilustración 9:	Diagrama de clase: Manager:ManagerAlumnosEx	11
Ilustración 10:	Diagrama de clase: Paquete de Manager:Manager	11

Ilustración 11: Diagrama de clase: Paquete de Web:Base Controller	12
Ilustración 12: Diagrama de clase: Paquete de Web:Download File	12
Ilustración 13: Diagrama de clase: Paquete de Web:PDFController	13
Ilustración 14: Diagrama de clase: Paquete de Web:PropuestaPFCController.....	13
Ilustración 15: Diagrama de clase: Paquete de Web:SessionInfo	14
Ilustración 16: Diagrama de clase: Paquete de Web:SolicitudController	14
Ilustración 17: UserActionsController	15
Ilustración 18: Casos de uso de la aplicacion	16
Ilustración 19: Diagrama de secuencia de una propuesta	16
Ilustración 20: Manual de usuario:Crear un nuevo usuario	17
Ilustración 21: Manual de usuario:Selección de usuario	18
Ilustración 22: Manual de usuario:Campos del profesor	18
Ilustración 23: Manual de usuario:Campos del alumno.....	19
Ilustración 24: Manual de usuario:Usuario logeado	19
Ilustración 25: Manual de usuario:Ver propuestas de proyectos fin de carrera	20
Ilustración 26: Manual de usuario: Seleccionar propuesta para hacer PFC.....	20
Ilustración 27: Manual de usuario: Crear propuesta de un PFC	21
Ilustración 28: Manual de usuario: Ver PFC asignados a los profesores	21
Ilustración 29: Manual de usuario: Cerrar un proyecto Fin de carrera	22

Introducción

Este documento está dividido en 2 partes:

- **El manual de diseño**, en el cual se explica las decisiones de arquitectura y diseño tomadas,
- **El manual de usuario**, o de uso de la aplicación donde se especifica los pasos para utilizar la aplicación.

1 Manual de Diseño

El manual de diseño presenta la arquitectura de la aplicación y explica las decisiones de diseño que se han tomado para el desarrollo de la misma.

1.1 Arquitectura

Para el desarrollo de la aplicación se ha usado un sistema gestor de base de datos MySQL con una Base de datos MySQL, tecnología java J2EE, como jsp, servlets. También se ha usado el framework de struts2 para implementar el patrón MVC entre otros.

La arquitectura se ha adoptado el patrón de arquitectura de 3 capas, soporte de base de datos, lógica de negocio e interfaz de usuario.

Para el soporte de la base de datos se ha utilizado el gestor de base de datos MySQL, y un patrón de arquitectura de base de datos orientada a objetos, creando nuestro propio framework e incluyendo el soporte de transacciones.

1.1.1 Base de datos orientadas a objetos

La base de datos de MySQL, es una base de datos relacional. Para realizar una mejor integración con la aplicación y no tener que construir las consultas SQL cada vez que se modifica una tabla, hemos realizado una aproximación a las base de datos orientadas a objetos lo que implica que no se realizan consultas de SQL directamente por cada tabla. Las consultas se realizan de manera dinámica por cada objeto. Se ha implementado usando la funcionalidad que da java a través de reflection.

Reflection

Reflection es la capacidad que tiene java en nuestro caso y otros muchos más lenguajes, de dar todas las características de una clase o de un objeto en tiempo de ejecución. En la aplicación el paquete *entidad*, se usa el nombre de la clase, los atributos de los objetos y sus valores. Con el nombre de la clase se accede a la tabla de la base de datos, los atributos de las clases indican los campos de esa tabla y los valores de cada uno de los objetos, indican el valor de ese registro.

1.2 Diseño

Se ha usado el modelo vista controlador. Para dar soporte a este modelo se ha empleado Struts como se verá más adelante. En el patrón M-V-C, el controlador es el encargado de a través de los eventos recibidos por las vistas seleccionar el vista adecuada y los datos necesarios para esas vista.

Para la conexión con la base de datos MySQL se ha usado las librerías mysql-connector-java-5.1.12.

1.2.1 Diagrama de paquetes

La aplicación esta dividida en una serie de paquetes, que son los siguientes

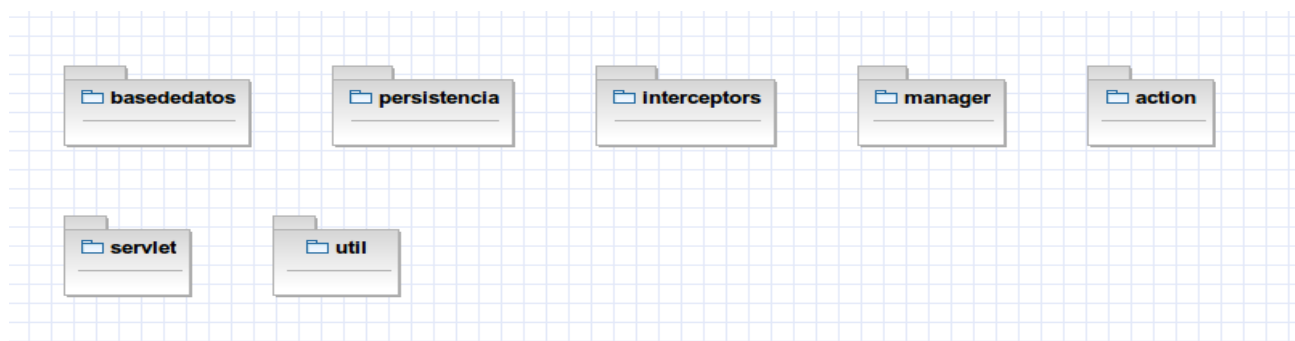


Ilustración 1: paquetes de la aplicación

El paquete base de datos se encarga de conectar con la base de datos y generar las consultas en tiempo de ejecución.

Práctica de Sistemas Informáticos III

El paquete persistencia son las clases que dan soporte al modelo de la base de datos

El paquete Intceptors aloja las clases que controlan el acceso autorizado a la aplicación mediante la captura de las peticiones Web para procesarlas.

El paquete manager se encarga de cargar los datos en los objetos del paquete de persistencia.

El paquete actions aloja las clases encargadas de procesar las peticiones. En un patrón MVC representan los Controladores

El paquete servlet son las clases que extienden la interfaz servlet

El paquete util son diversas funcionalidades útiles para la aplicación

Aparte, en la carpeta del sitio se ubican las páginas jsp que representan los componentes VISTA del patrón MVC.

1.1. Paquete de persistencia y de base de datos

Los paquetes de persistencia y de base de datos es.uned.si3.persistencia, es.uned.si3.basededatos

Persistencia

Los objetos entidad, son objetos muy simples, que solo contiene atributos. Estos atributos, y a través de reflection, se encarga de realizar la gestión de las tablas que son los encargados del soporte de cada uno de los datos de las tablas. En la aplicación se usa las clases del paquete *entidad* con ID que indica el Identificador del registro de la tabla, el cual es ID auto-numérico. Para crear las relaciones con las otras tabla de la base de datos, se usa la clave foránea. Para indicar a la clase que tiene una relación con otra tabla, se realiza con el “*ID_nombre de la clase*”, Este atributo será del tipo de dato igual a la clase entidad que se relacione. Es decir, en la clase entidad Alumno, hay un atributo como en todas las clases entidades que es ID de tipo entero y para relacionar los Alumno con los Mensajes y con los Proyectos las relaciones foraneas que son ID_Mensaje de tipo de datos Mensaje y ID_Mensaje que es de tipo de dato Mensaje. Son las únicas restricciones que se tienen a la hora de crear un clase en el paquete *entidad*. Las clase dentro del paquete entidad solo se relacionan uno a N. En nuestro caso solo se han encontrado relaciones de este tipo de datos. En el caso de haber encontrado algún tipo de entidad de tipo N a N, la solución seria dividir las relaciones en tres entidades.

Base de datos.

Dentro del paquete de la base de datos se han creado 2 clases para la gestión de la base de datos. Estas clases son GestorSql y CrearBaseDeDatos. GestorSql es el encargado de realizar la conexión a la base de datos, se realiza a través del patrón de diseño singleton. Esta conexión es única. Se guarda la instancia de la conexión de la base de datos y si se intenta crear una nueva conexión, se devuelve la conexión ya creada. Esto nos sirve para poder gestionar la conexión de la base de datos de forma centralizada. Además la base de datos, la clase GestorSql es la encargada de generar las consultas para cada tipo de objeto a través de reflection. Tiene un método por cada acción a realizar, como la inserción, la actualización, el borrado y la búsqueda. Para generar una consulta en tiempo de ejecución se para una objeto del paquete.

En la aplicación se ha realizado un control de acceso a datos como si fuera una base de datos

orientada objetos. No se ha implementado un patrón de diseño DAO, pero el funcionamiento ha estado muy cercano a este patrón. La implementación del acceso a la base de datos se ha realizado en 2 pasos. Primero se ha creado un acceso a la base de dato, implementando el patrón singleton creando un gestor de base de datos que se encarga de abrir el acceso a la base de datos y el cual implementa los métodos mas importantes de acceso a la base de datos que son create update delete. Estos métodos reciben cada uno de los objetos a serializar en la base de datos. Otro método importante es el find.

Metodo find

Como ejemplo de búsqueda se comenta como se usa el método find dentro de la gestión de la base de datos junto con reflection y los objetos entidad para dar una visión completa del trabajo realizado. Cuando se quiere mostrar unos datos dentro de la aplicación, primero se instancia un objeto del tipo de datos que necesitamos con la instrucción típica de java como:

```
Alumno cliente =new Alumno();
```

Solamente con esta instrucción se pasa al método find de la base de datos, devolverá un array de objetos del mismo tipo de datos, en nuestro caso de tipo cliente.

Si se quiere buscar o filtrar un objeto de cliente determinado para este fin, se usa los atributos de objeto, para nuestro ejemplo de la clase cliente sería

```
alumno.ID=1;
```

Al pasar el objeto con el atributo ID igual a 1, genera con reflection una consulta sql, que para nuestro caso es transparente y nos devuelve un array de los objetos clientes con un solo elemento si es que existe este elemento en la base de datos.

Si se quiere filtrar por algún campo el método find tiene un segundo parámetro además de el objeto a buscar, que se puede activa, con esta activación, la consulta sql que genera es en vez de que la consulta sea igual al campo se cambia por la instrucción de sql que indica que la consulta generada sea parecida con el operador LIKE útil para buscar por aproximaciones en campos de tipo cadena.

1.2. Interfaz de usuario

El interfaz de usuario se ha implementado a través de Java Server Pages..

1.2.2 Diagrama de clases

Diagrama de clases de la persistencia

A continuación se muestran un conjunto de esquemas donde se reflejan las El diagrama de clases de la persistencia está identificado en la siguiente imagen. En ella se refleja la estructura de datos que es almacenada en la base de datos.

Clase Alumno

La clase alumno, tiene instancias a centro asociado, a propuesta y a proyecto. A su vez Mensaje y solicitud tiene instancia a alumno

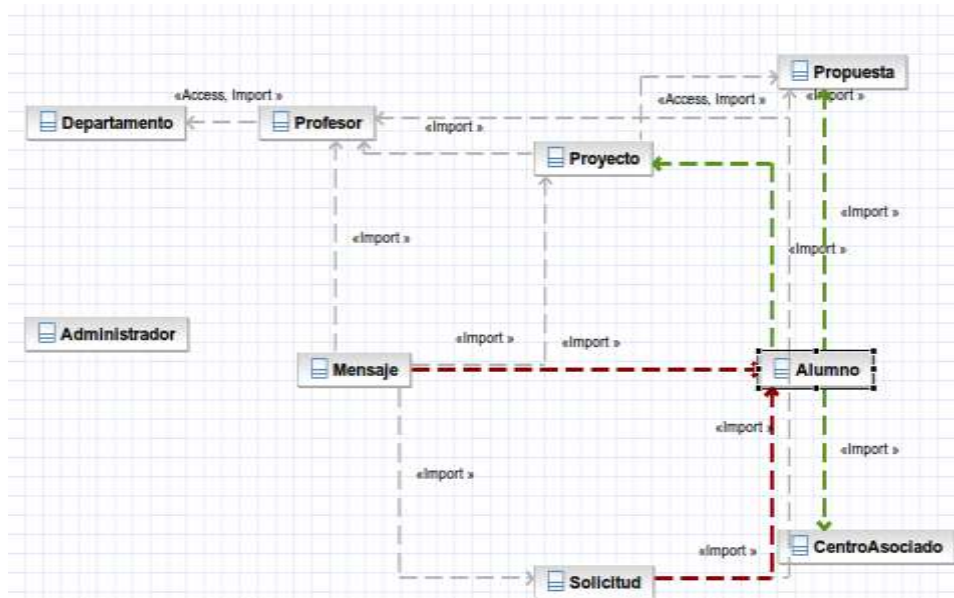


Ilustración 2: Diagrama de clase: Paquete de Persistencia:Alumno

Clase Centro asociado

La clase centro asociado, es instanciada solamente por alumnos

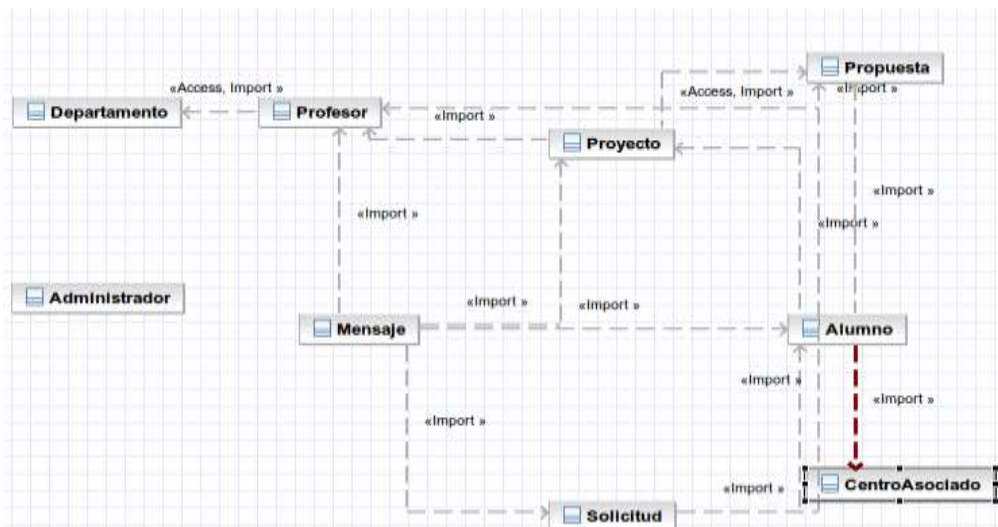


Ilustración 3: Diagrama de clase: Paquete de Persistencia:Centro Asociado

Clase Departamento

Solamente la clase profesor tiene una instancia a esta clase

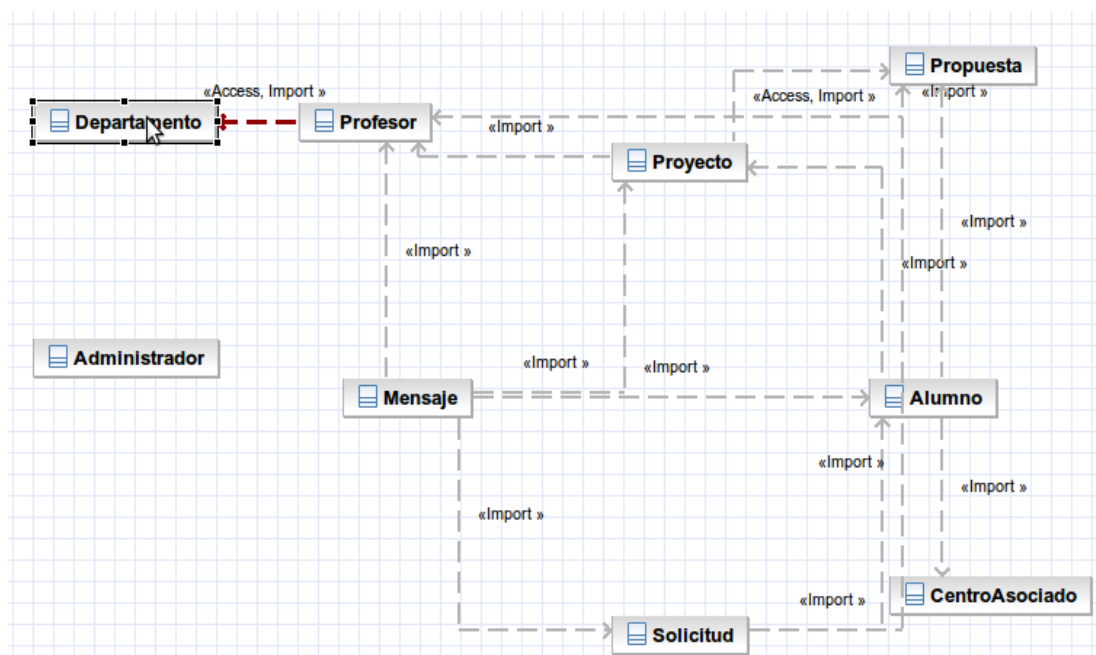


Ilustración 4: Diagrama de clase: Paquete de Persistencia:Departamento

Clase Profesor

La clase profesor es instanciada por propuesta proyecto mensaje y profesor solo instancia a departamento.

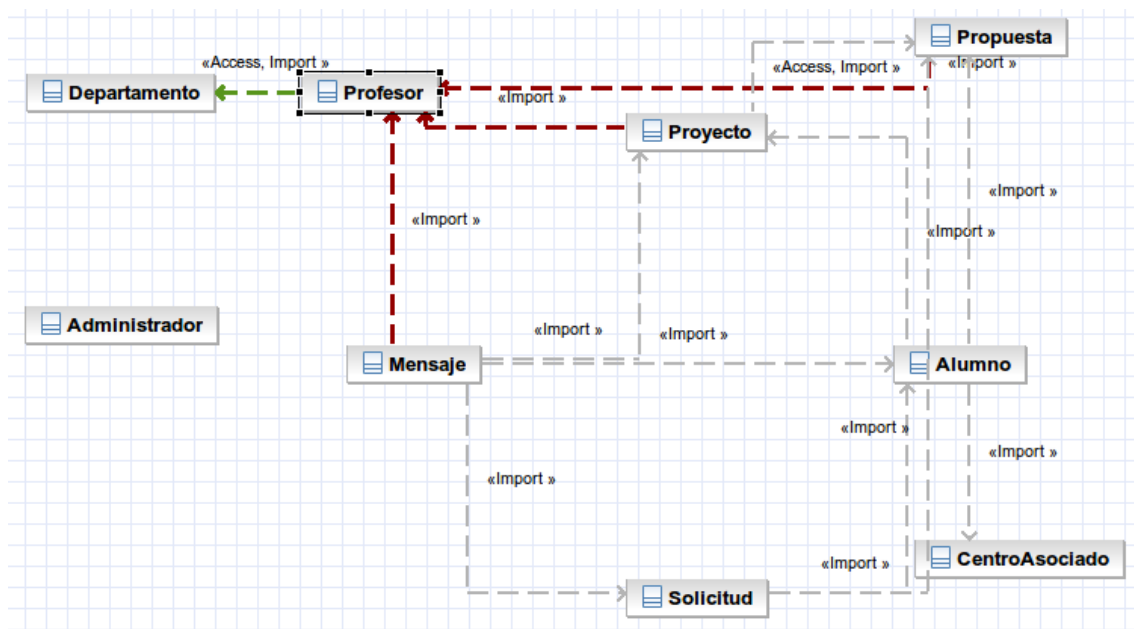


Ilustración 5: Diagrama de clase: Paquete de Persistencia:Profesor

Clase Propuesta

Es instanciada por alumno y solicitud e instancia solamente a profesor.

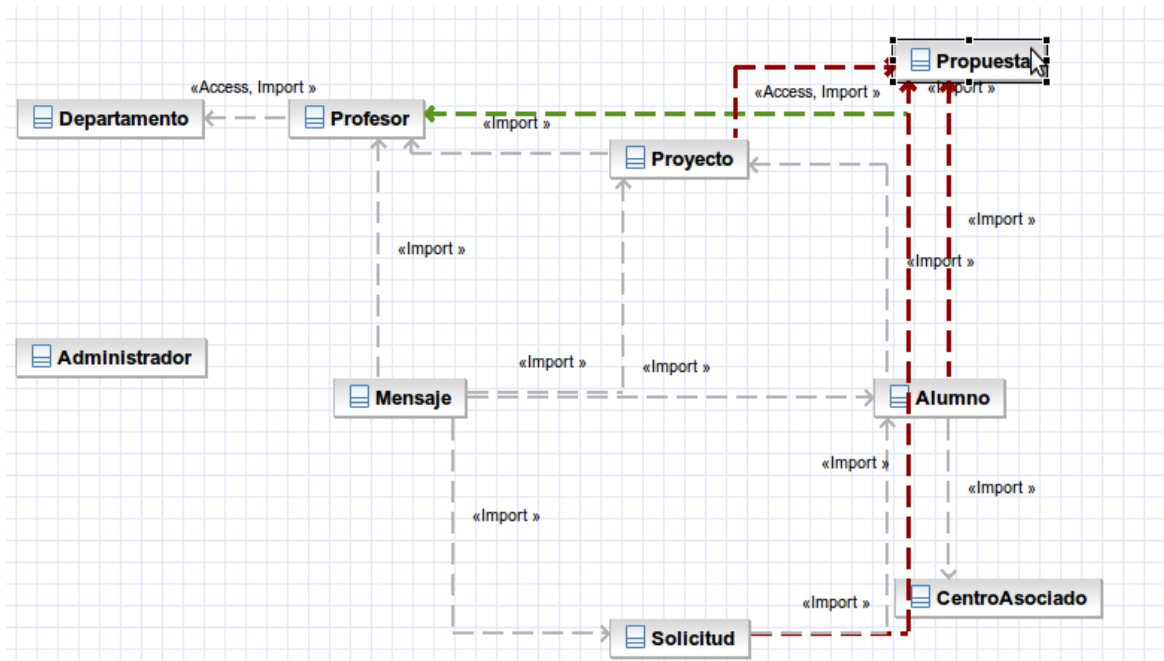


Ilustración 6: Diagrama de clase: Paquete de Persistencia:Propuesta

Clase Solicitud

Una solicitud está vinculada a un propuesta y a un alumno. Los mensajes tienen instancia de las solicitudes.

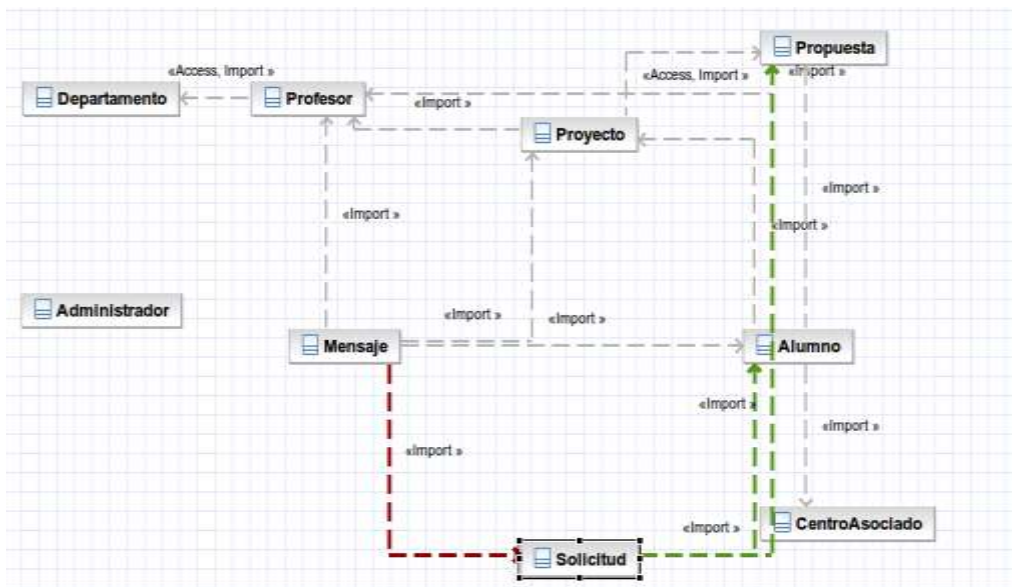


Ilustración 7: Diagrama de clase: Paquete de Persistencia:Solicitud

Clase Proyecto

Instancia a profesor y a propuesta y es instanciada por mensaje y alumno.

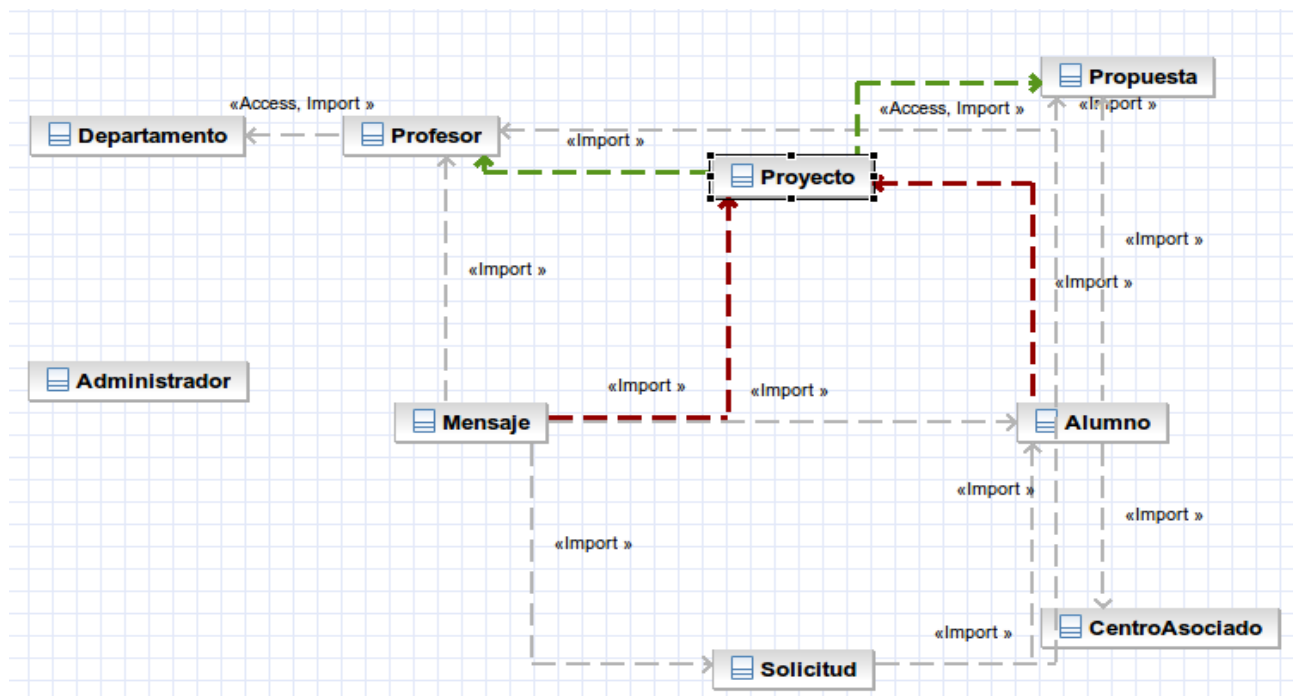


Ilustración 8: Diagrama de clase: Paquete de Persistencia:Proyecto

Diagrama de clases de la manager

Clases Manager

Cada una de estas clases se encargan de cargar los datos de la base de datos para crear instancias de los objetos. La clase ManagerAdministrator, Manager Profesor Manager Alumnos y ManagerMensajes, tiene el mismo funcionamiento, exceptuando que cargan los datos de distintas tablas.

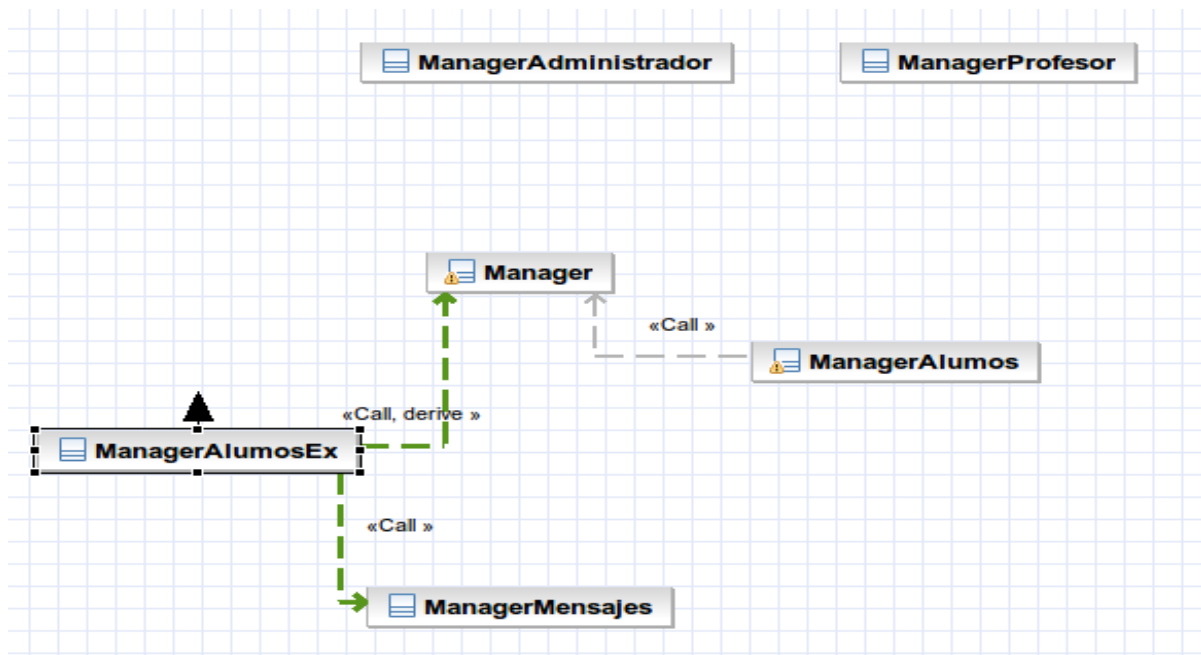


Ilustración 9: Diagrama de clase: `Manager:ManagerAlumnosEx`

Clase Manager

Es la clase que se encarga de coordinar cada una de las acciones del manager.

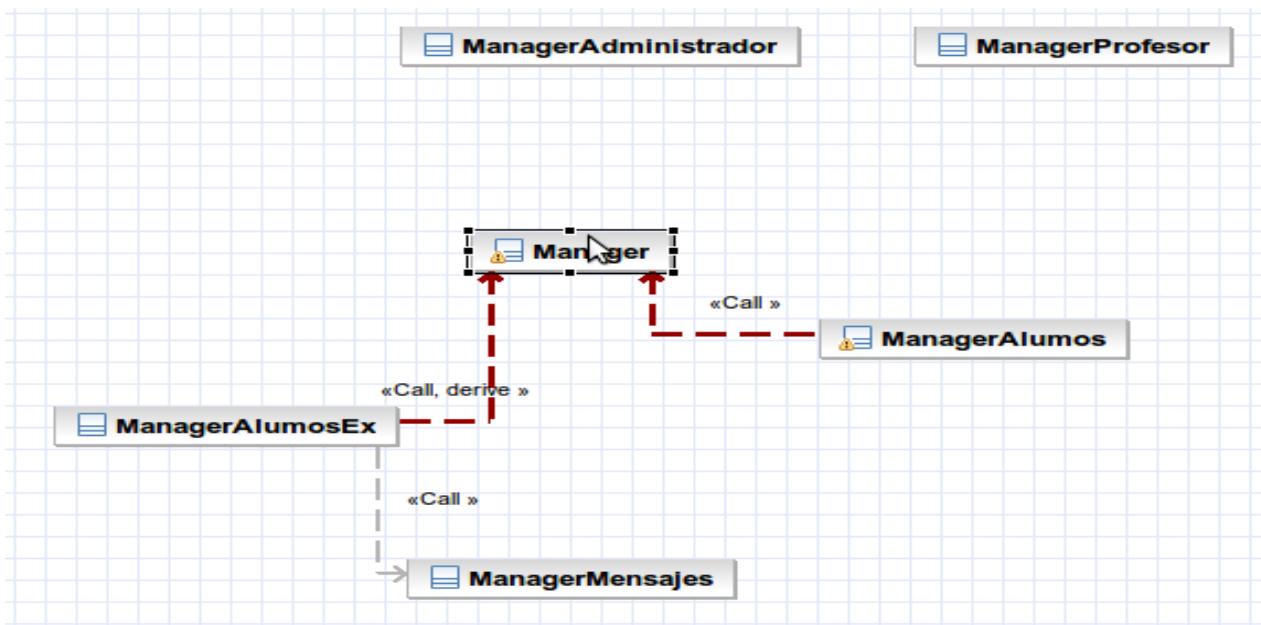


Ilustración 10: Diagrama de clase: Paquete de `Manager:Manager`

Diagrama de clases del Modelo vista controlador

Clase BaseController

Es la clase que implementa el controlador del patrón MVC. Esta clase es la clase padre de todas las clases controller. Usa la clase SessionInfo para recuperar los valores devueltos por los usuarios de la Web. Cada una de las clases controller hijas implementa una funcionalidad distinta según su uso que se especifica en los siguientes puntos.

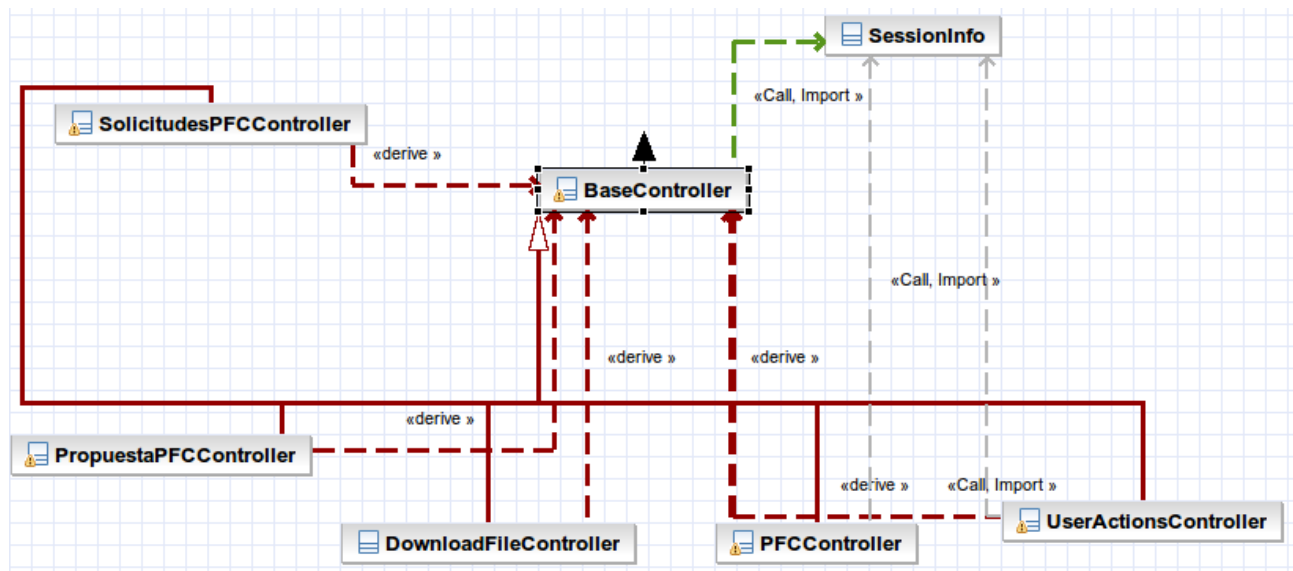


Ilustración 11: Diagrama de clase: Paquete de Web:Base Controller

Clase DownloadFileController

Implementa las acciones que debe realizar las descargas de la aplicación

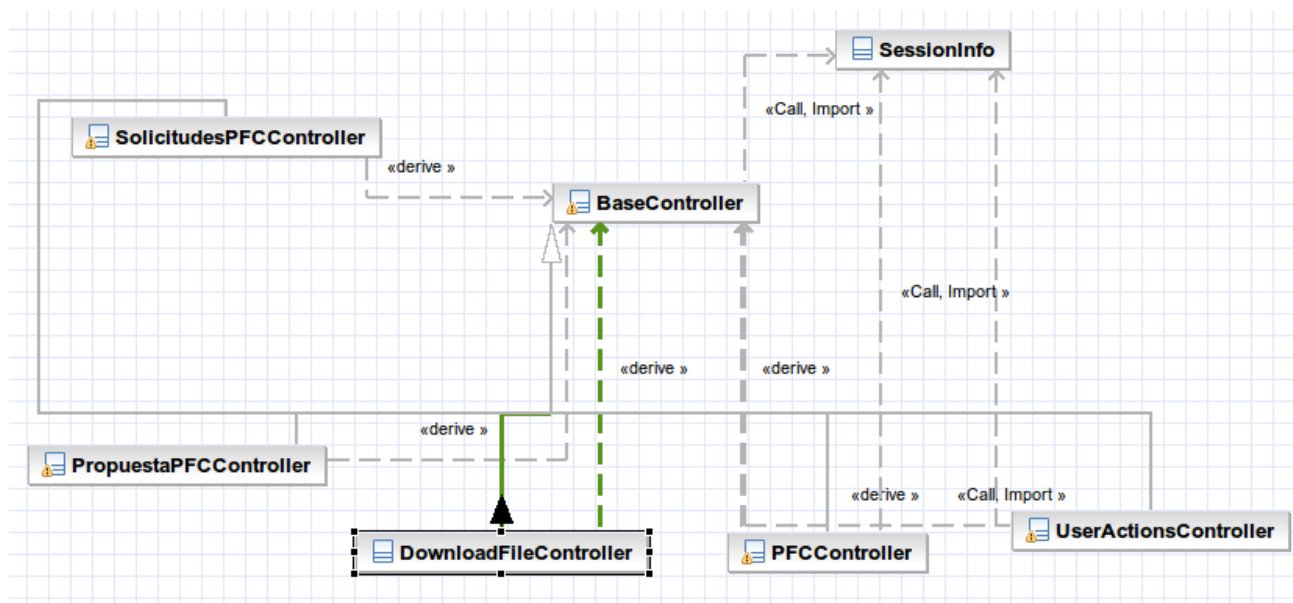


Ilustración 12: Diagrama de clase: Paquete de Web:Download File

Clase PFCController

Implementa las acciones que debe realizar para el manejo de las acciones del proyecto fin de carrera.

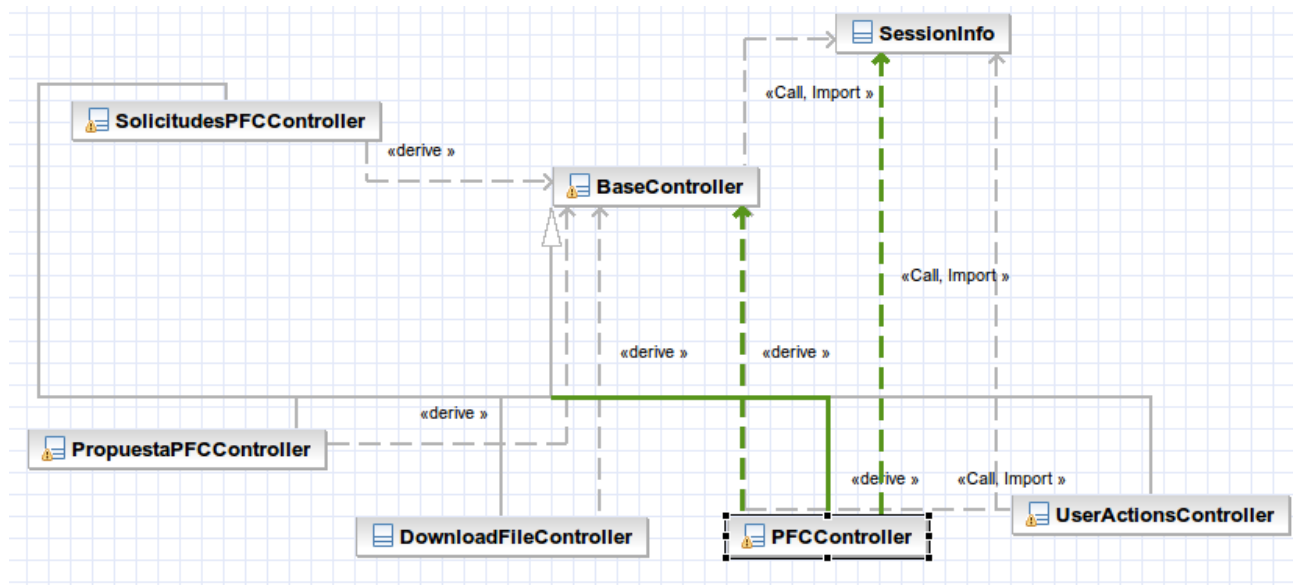


Ilustración 13: Diagrama de clase: Paquete de Web:PDFController

Clase PropuestasPFCController

Implementa las acciones que debe realizar para el manejo de las acciones de las propuestas del proyecto fin de carrera.

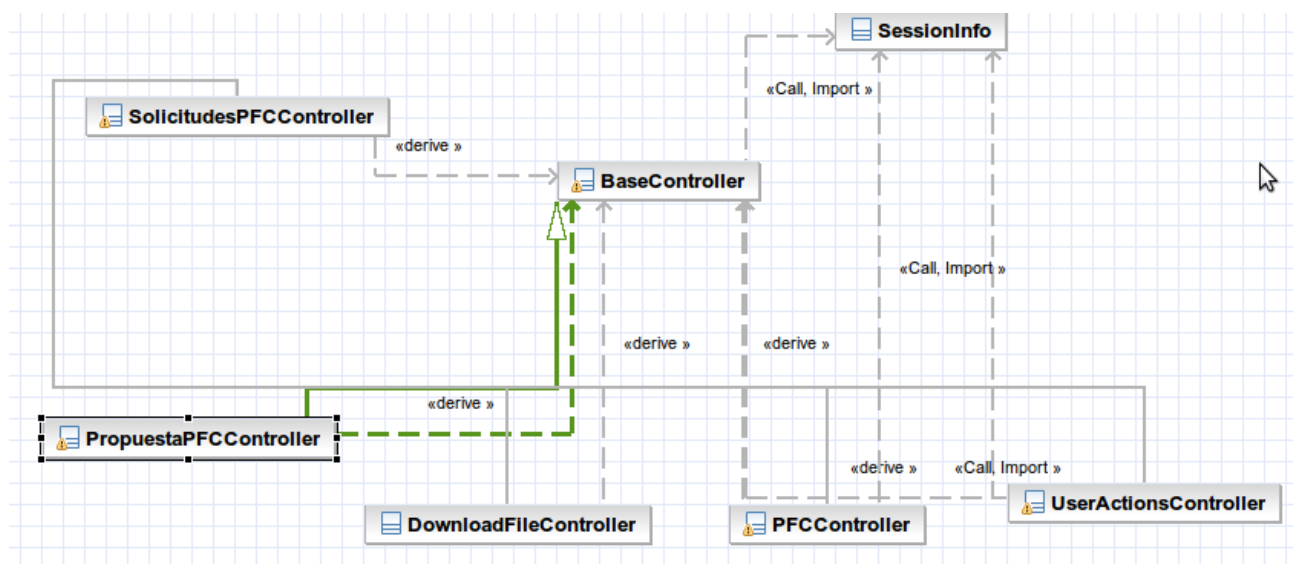


Ilustración 14: Diagrama de clase: Paquete de Web:PropuestaPFCController

Clase SessionInfo

Gestiona la información de los datos de la web necesarios para dar soporte a cada uno de los usuarios con sus datos en la sesión determinada. Es usada por todas las clases Controller ya sean a través de la clase padre BaseController, heredando el funcionamiento de la clase o por ellas mismas.

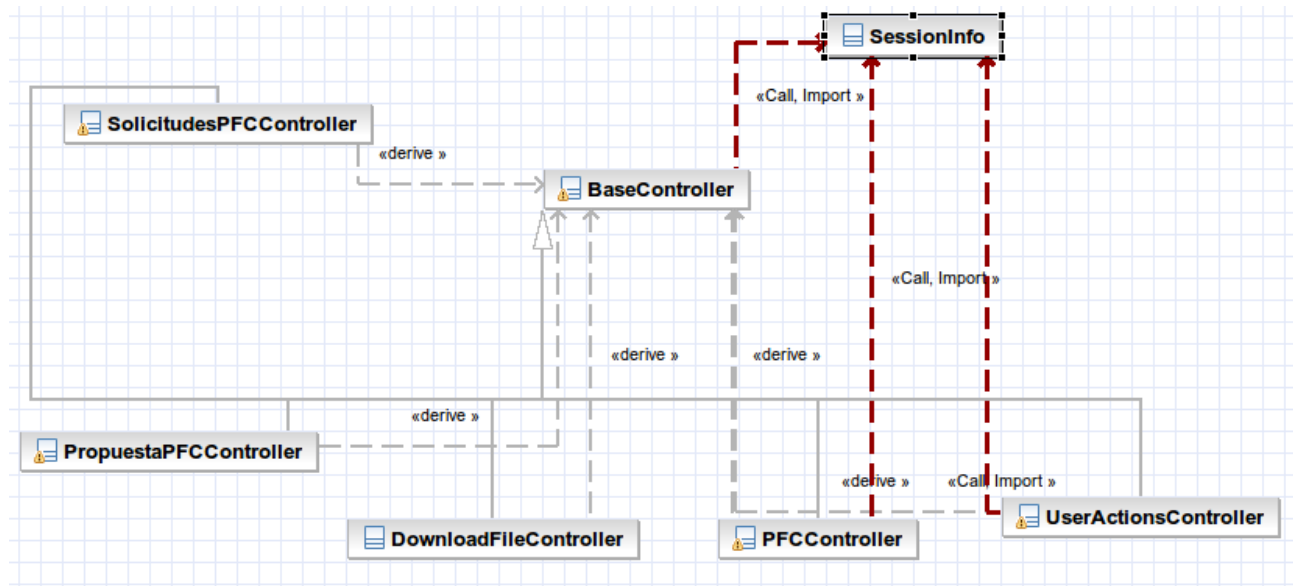


Ilustración 15: Diagrama de clase: Paquete de Web:SessionInfo

Clase SolicitudesPFController

Controlador de la acciones de las solicitudes del proyecto de fin de carrera.

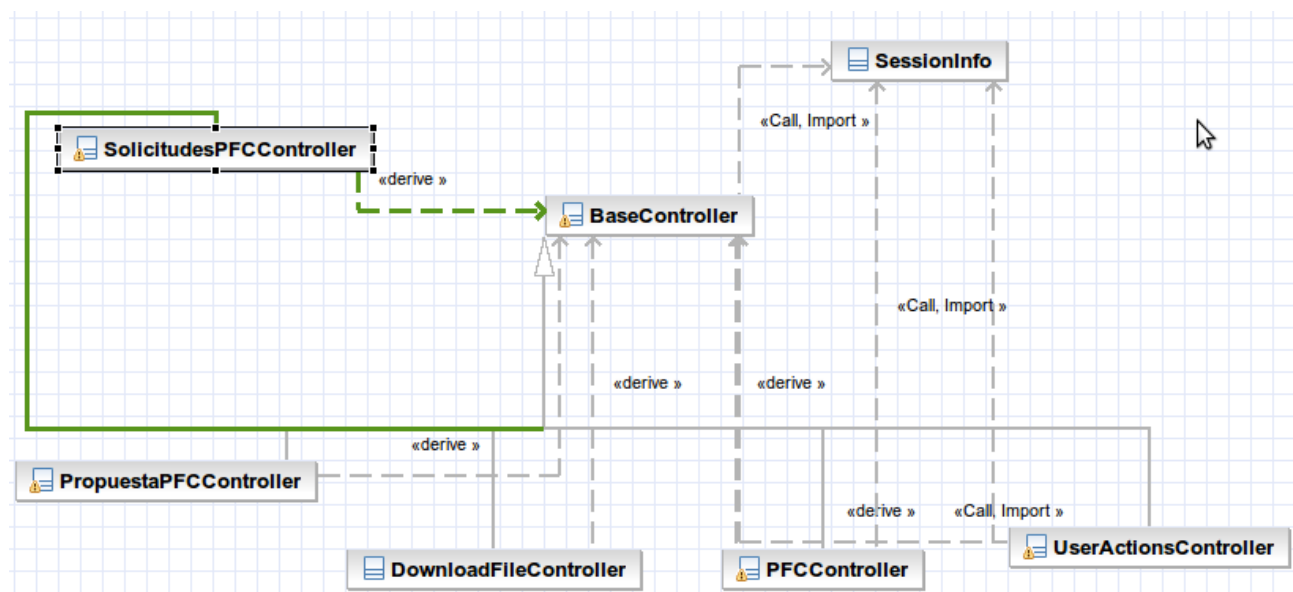


Ilustración 16: Diagrama de clase: Paquete de Web:SolicitudController

Clase UserActionController

Implementa dentro del patrón M-V-C las acciones del controlador para gestionar los datos.

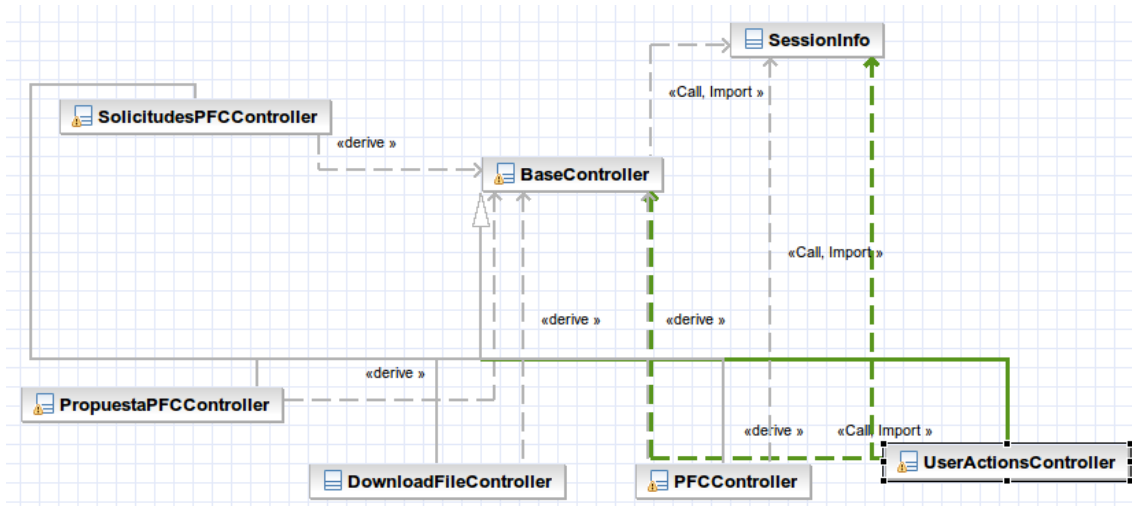


Ilustración 17: UserActionsController

Diagrama de casos de usos

En el sistema se identifican tres roles, el alumno, el profesor y el administrador

1. El alumno tiene las siguientes acciones
 - 1.1. Ver propuestas:
 - 1.2. Solicitar proyecto fin de carrera del listado de las propuestas
 - 1.3. Ver mensaje
2. El profesor:
 - 2.1. Crear una nueva propuesta
 - 2.2. Ver propuesta
 - 2.3. Cerrar proyecto fin de carrera
3. El administrador:
 - 3.1. Asignar propuesta
 - 3.2. Aceptar propuestas
 - 3.3. Ver mensajes

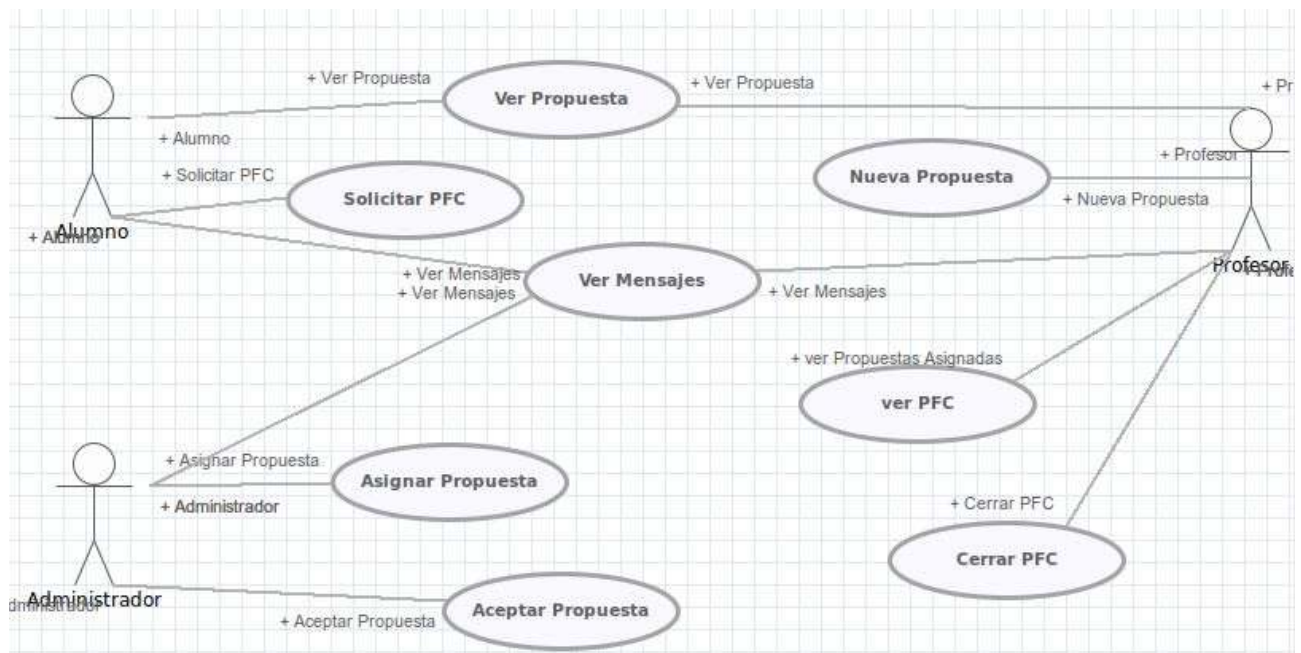


Ilustración 18: Casos de uso de la aplicación

Diagrama de secuencias

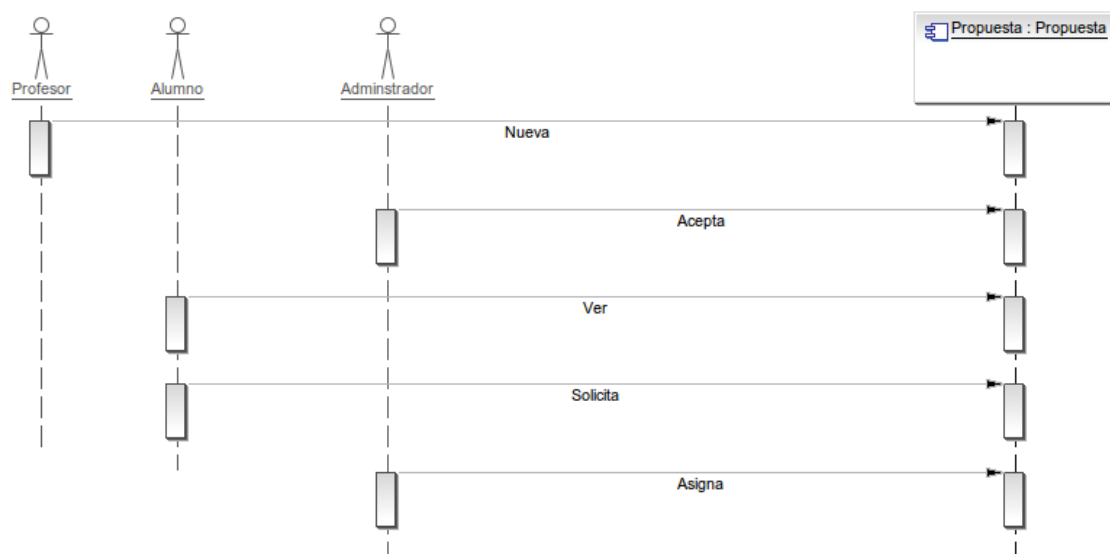


Ilustración 19: Diagrama de secuencia de una propuesta

1.2.3 Patrones empleados en la práctica

A continuación se hace una introducción de los distintos patrones empleados para mejorar y facilitar el desarrollo del portal web con java j2ee.

El objetivo fundamental de un patrón es el de ofrecer funcionalidades avanzadas a sus usuarios que permitan inhibirle de gran parte de los detalles que implican ciertas operaciones que de otra manera requerirían gran esfuerzo.

Cada petición a una página web de una aplicación web j2ee implica una serie de pasos que muchas

veces se escapan a nuestro conocimiento. Esta práctica ha supuesto un excelente caso de uso para profundizar en su comportamiento y poder aprovechar mejor las posibilidades que ofrece.

Declaración de frameworks empleados

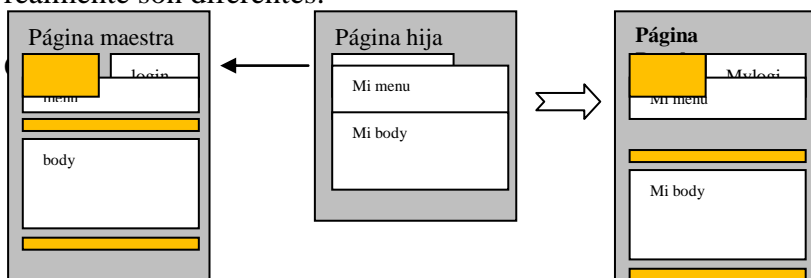
El fichero web.xml presente en la carpeta WEB-INF de todas las aplicaciones web j2ee contiene una declaración de las actividades que deberán llevarse a cabo sobre cada petición. Por ejemplo, si analizamos el fichero web.xml presente en la presente práctica, encontramos lo siguiente:

```
...<filter>
<filter-name>sitemesh</filter-name>
<filter-class>com.opensymphony.sitemesh.webapp.SiteMeshFilter</filter-
class>
</filter>
<filter-mapping>
<filter-name>sitemesh</filter-name>
<url-pattern>/*</url-pattern>
</filter-mapping>
...
```

Esta declaración xml está declarando un filtro que se va a aplicar a todas las peticiones que se hagan a esta página (<url-pattern>) y dicho filtro es ejecutado por la clase `com.opensymphony.sitemesh.webapp.SiteMeshFilter`.

1.1. Framework Sitemesh

La clase `com.opensymphony.sitemesh.webapp.SiteMeshFilter` es el punto de entrada a este framework de construcción de páginas web para compartir elementos comunes a todas ellas. En vocabulario de aplicaciones ASP.NET sitemesh permite la definición de páginas padre. En nuestra aplicación, se ha empleado dicha funcionalidad para dotar a todas las páginas del sitio web del mismo interfaz gráfico, posicionamiento de controles estándar lo cual mejora la usabilidad de la aplicación al reducir el número de pantallas diferentes durante el empleo de la aplicación. Por otro lado simplifica la creación de páginas ya que exceptuando la página “padre” empleada como patrón para construir el resto de páginas, el resto se reduce a declarar los elementos que realmente son diferentes.



1.2. Framework Struts2

El framework Struts2 es una herramienta de soporte para el desarrollo de aplicaciones Web bajo el patrón MVC bajo la plataforma Java EE (Java Enterprise Edition). De acuerdo con el patrón MVC (Modelo, Vista, Controlador), el procesamiento se separa en tres secciones diferenciadas llamadas el modelo, las vistas y el controlador de manera que se desacopla el desarrollo de las aplicaciones haciéndose más sencillo sustituir unas partes por otras sin afectar la funcionalidad de la aplicación. Entre las múltiples características que ofrece esta librería, en el presente proyecto se han empleado las siguientes:

Interceptors

Los interceptors se encargan de interceptar el flujo de ejecución de las peticiones y realizar distintas tareas. En el contexto del presente proyecto, se han desarrollado dos interceptores en el paquete `es.uned.si3.web.interceptors` para chequear que el usuario tenía la autorización para ejecutar cierto código.

Se declaran dentro del fichero de configuración de struts (`struts.xml`) :

```
...
    <interceptors>
        <interceptor name="authenticationInterceptor"
class="es.uned.si3.web.interceptors.AuthenticationInterceptor" />
        <interceptor name="rolesInterceptor"
class="es.uned.si3.web.interceptors.RolesInterceptor" />

        <interceptor-stack name="defaultSecurityStackWithAuthentication">
            <interceptor-ref name="defaultStack" />
            <interceptor-ref name="authenticationInterceptor">
...
                </interceptor-ref>

            <interceptor-ref name="rolesInterceptor">
...
                </interceptor-ref>
            </interceptor-stack>
        </interceptors>

<default-interceptor-ref name="defaultSecurityStackWithAuthentication" />
```

En esta declaración, se observa que definimos nuestros interceptors junto con la clase que los implementa, y después (`<interceptor-stack>`) creamos una pila de interceptores constituido por un conjunto por defecto junto con nuestros interceptores.

Esta herramienta es muy potente, porque podemos seleccionar aquellos interceptores que queremos que se ejecuten. Por ejemplo, entre aquellos que vienen en la pila por defecto, tenemos los que se encargan de validar el contenido de las páginas antes de pasarlo al servidor, de subir ficheros desde el cliente al servidor, etc.

Acciones

En el vocabulario Struts2, cuando un cliente solicita ejecutar algo a una aplicación web, le está pidiendo que realice una acción. Struts ofrece una sintaxis muy rica para describir el cómo debe llevarse a cabo dicha ejecución. Se define en el fichero de configuración de struts (`struts.xml`) y por ejemplo, encontramos:

```
<action name="PropuestaPFC.*"
class="es.uned.si3.web.action.PropuestaPFCController"
    method="{1}">
    <interceptor-ref name="defaultSecurityStackWithAuthentication"/>
    <interceptor-ref name="fileUpload">
        <param name="maximumSize">536870912</param> <!-- 512 MB -->
        <param name="allowedTypes">application/pdf</param>
    </interceptor-ref>
```

Práctica de Sistemas Informáticos III

```
<result name="input">/index.jsp</result>
<result name="crear">/private/editPropuestaPFC.jsp</result>
<result name="listaPropuestas">/private/propuestasPFC.jsp</result>
<result name="solicitar">/private/solicitarPFC.jsp</result>
<result name="start">/index.jsp</result>
<result name="error">/index.jsp</result>
</action>
```

Es en las páginas web (jsp) en las que se declaran las acciones que han de ejecutarse en el servidor. Por ejemplo, nos podemos encontrar con:

```
<s:form action="PropuestaPFC.solicitar" method="post"
  enctype="multipart/form-data">
  <s:file name="upload" label="Fichero de conocimientos"
    accept="application/pdf"></s:file>
  <s:submit align="left" value='Solicitar' />
```

Donde el resultado de pulsar el botón submit es el de invocar a la acción **PropuestaPFC.solicitar**. Esta acción es capturada por struts que encuentra en el fichero de configuración que dicha acción ha de ser procesada por la clase es.uned.si3.web.action.PropuestaPFCController y por el método "{1}". En este caso struts sigue la sintaxis que indica que el * se interpreta como cualquier cosa y se referencia como {1}, {2},... Así se invocará al método **solicitar de la clase PropuestaPFCController**. Esta clase lleva a cabo la tarea de Controlador y en este proyecto de Modelo.

Este método se encarga de dar valor a los atributos que servirán a la página jsp (que lleva a cabo la tarea de Vista) para dar contenido a la página e interpretar los valores devueltos por el cliente para ejecutar las acciones adecuadas en el servidor.

Empleo de etiquetas Struts2

- tanto de control, como de UI:
 - o **Tags de control:** Controlan el flujo de ejecución de las páginas en que son declaradas. Por ejemplo:

```
<s:if test="%{#userName != null}">
<table class="loginTable">
  <s:label value="Bienvenido %{#userName}" />
</table>
</s:if>
<s:else>
<span class="fuentepeql"> <s:url id="urlRegister"
  action="UserAction.newUser" /> <s:a href="%{urlRegister}"
  title="Nuevo usuario">Nuevo usuario</s:a>
</span>
</s:else>
```

- o **Tags de UI:** Pinta elementos visuales en la pantalla de forma sencilla e intuitiva. Por ejemplo:

```
<s:select name='centro_asociado' label='Centro asociado' headerKey=""
required='true' headerValue="--Selecciona--" list="listaCentrosAsociados"
/>
```

Esta simple etiqueta, genera en tiempo de ejecución el siguiente código html en base al contenido del atributo listaCentrosAsociados:

```
<select name="centro_asociado" id="UserAction_newUser_centro_asociado">
  <option value="">--Selecciona--</option>
  <option value="CALATAYUD">CALATAYUD</option>
  <option value="BARBASTRO">BARBASTRO</option>
  <option value="Almeria">Almeria</option>
  <option value="Almeria-EL EJIDO">Almeria-EL EJIDO</option>
  <option value="BAZA">BAZA</option>
</select>
```

Validación

Entre los interceptores explicados anteriormente, hay unos encargados de validar que el valor de los campos de los formularios es correcto. Estos interceptores, esperan un fichero con el nombre del Controlador, pero con la terminación NombreControlador-validation.xml.

```
<field name="dni">
  <field-validator type="regex">
    <param name="expression">
      [0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][a-zA-Z]
    </param>
    <message>Sintaxis de ejemplo del DNI:999999999V</message>
  </field-validator>
</field>
```

Aquí se muestra la sintaxis para ejecutar una validación sobre un DNI (8 dígitos terminados en una letra)

NOTA: Lamentablemente, el 1 de Junio de 2011, a mitad de la realización de la práctica, nos encontramos con la desagradable sorpresa de que ninguno de nuestros ficheros xml de validación funcionaban ya que la DTD que comprobaba la validez sintáctica de dichos ficheros había sido eliminada del site oficial. En su lugar, en vez de la TDT, había una nota explicando la discontinuación del proyecto.

Afortunadamente, en los foros –que no en la página oficial- encontramos un repositorio más o menos estable de DTDs (siendo el <http://struts.apache.org/dtds/xwork-validator-1.0.2.dtd> el empleado en esta práctica en vez del exoficial "http://www.opensymphony.com/xwork/xwork-validator-1.0.2.dtd").

1.2.4 Implementación

Para la implementación de la aplicación se ha usado *Eclipse Java EE IDE for Web Developers*. Version: *Helios Service Release 2*, también se ha usado plugins eclipse para la creación de los diagramas UML Omodo.

1.2.5 Uso de la aplicación

Se ha introducido dentro de la aplicación, unos usuarios de pruebas, así como datos para realizar las distintas funcionalidades de la Web.

Los usuarios de la Web, tienen la contraseña igual al nombre son:

Usuario	nombre	Contraseña
Administrador	admin	admin
Profesor	p1	p1
Alumno	a1	a1

2 Manual de usuario

2.1 Objetivo de la aplicación

Práctica SI3 Nuevo usuario

♦ Nuevo usuario

El presente portal ofrece tanto a alumnos, como profesores de la UNED una herramienta que permite automatizar el proceso oficial para la realización del Proyecto Fin de Carrera (PFC) de las titulaciones pertenecientes a la ETSI Informática.

A continuación deberá registrarse como alumno / profesor o Administrador. En caso de no haberse dado de alta, regístrese como nuevo usuario...

Nombre único de usuario*:

Contraseña*:

Práctica desarrollada por:
[Carlos González Muñoz](#)
[Jorge Vea Murquía](#)

Webdesign by
© [Carlos González Muñoz & Jorge Vea Murquía](#)
Students wishing to finish their studies!

La pantalla de bienvenida de la aplicación, explica de forma clara el objetivo del portal y la necesidad de registrarse para poder hacer uso de él.

2.2 Usabilidad

Una página web supone una experiencia nueva para un usuario. El hecho de reducir la curva de aprendizaje de las nuevas tecnologías es fundamental para el éxito y aceptación de cualquier iniciativa.

En este caso, se han seguido varios patrones de usabilidad para facilitar la vida a los usuarios.

- **Interface común en todas las páginas de la aplicación** gracias al uso del framework sitemesh.



- Empleo de hoja de estilos que permitiría modificar la presentación de la aplicación fácilmente
- Diseño de la página de manera que no dependa al 100% de la hoja de estilos y pueda ser usada en entornos que no la acepten:



Con hoja de estilo



Sin hoja de estilo

- Reducción en la medida de lo posible de la introducción de información por parte del usuario. Por ejemplo, para filtrar los años de los PFC, se le presenta un dropdown con solo aquellos años en los que haya PFCs.
- Presentación de los mensajes importantes en colores llamativos:

Nombre único de usuario*:	p3
Sintaxis de ejemplo del DNI:12345678V	
Dni*:	p3

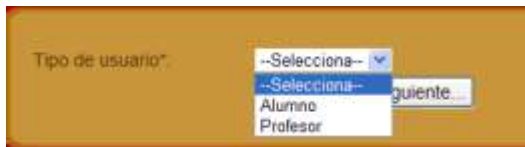
Este ejemplo indica al usuario que el DNI está mal introducido

2.3 Funcionalidades comunes

2.3.1 Nuevo usuario

Aparte del usuario administrador, puede haber usuarios profesores y alumnos. La misma plantilla se sigue en ambos casos, exceptuando que los profesores pertenecen a un departamento y los usuarios a un centro. Como se ve, todo es igual excepto ese campo desplegable. 1º se pide que tipo de usuario se vaya a crear:

Práctica de Sistemas Informáticos III



Tipo de usuario*

--Selecciona--

--Selecciona-- siguiente...

Alumno

Profesor

2º Se completan los demás datos del usuario:



Reservé los datos relativos al nuevo Alumno:

Nombre*

Apellidos*

Nombre único de usuario*

Dni*

E-Mail*

Centro asignado*

--Selecciona--

--Selecciona--

CALATAYUD

BARBASTRO

Almería

Almería-EL EJIDO

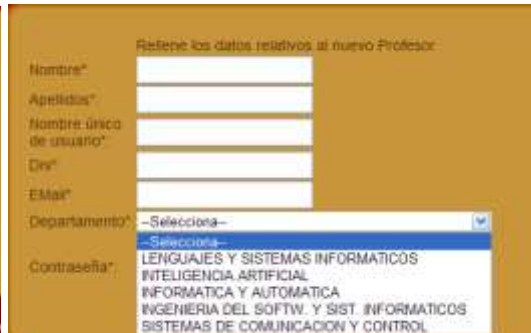
Almería- HUERCAL OVERA

BAZA

Contraseña*

Práctica desarrollada por

Nuevo Alumno



Reservé los datos relativos al nuevo Profesor:

Nombre*

Apellidos*

Nombre único de usuario*

Dni*

E-Mail*

Departamento*

--Selecciona--

--Selecciona--

LENQUAJES Y SISTEMAS INFORMATICOS

INTELIGENCIA ARTIFICIAL

INFORMATICA Y AUTOMATICA

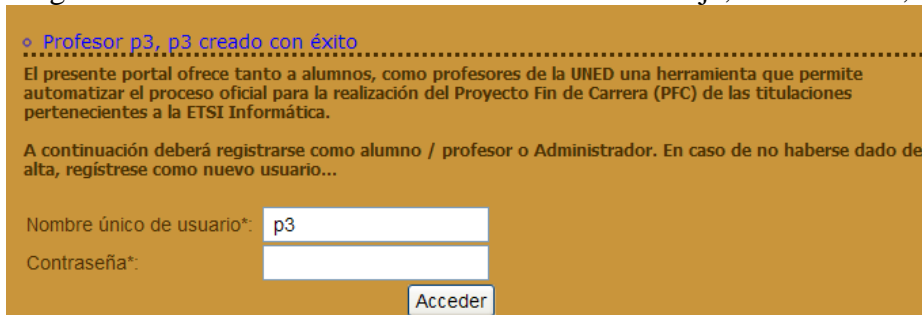
INGENIERIA DEL SOFTW. Y SIST. INFORMATICOS

SISTEMAS DE COMUNICACION Y CONTROL

Contraseña*

Nuevo Profesor

En general. Todas las acciones finalizan con un mensaje, bien de error, bien de confirmación:



◦ Profesor p3, p3 creado con éxito

El presente portal ofrece tanto a alumnos, como profesores de la UNED una herramienta que permite automatizar el proceso oficial para la realización del Proyecto Fin de Carrera (PFC) de las titulaciones pertenecientes a la ETSI Informática.

A continuación deberá registrarse como alumno / profesor o Administrador. En caso de no haberse dado de alta, regístrese como nuevo usuario...

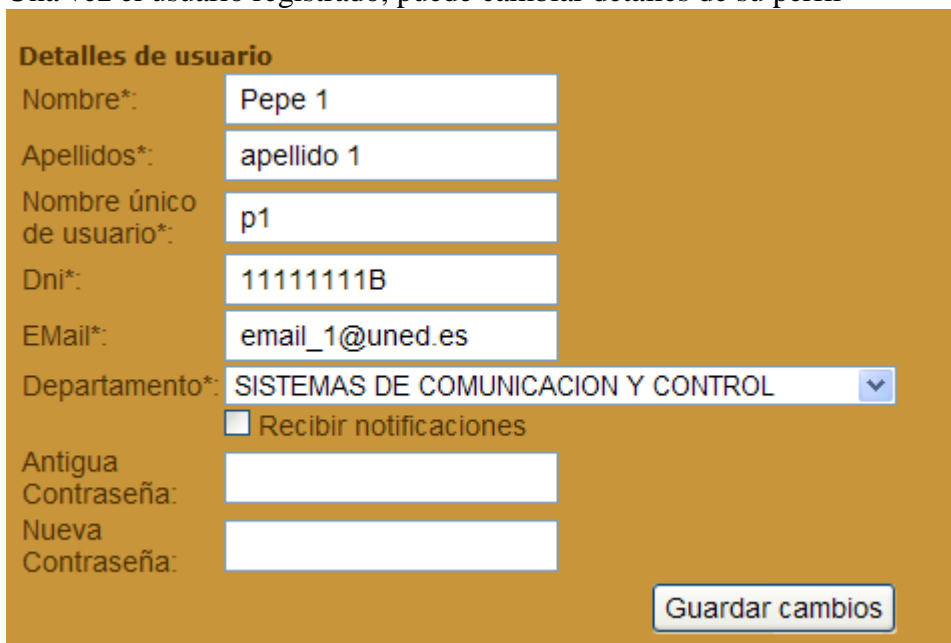
Nombre único de usuario*: p3

Contraseña*:

Acceder

2.3.2 Edición de usuario

Una vez el usuario registrado, puede cambiar detalles de su perfil



Detalles de usuario

Nombre*: Pepe 1

Apellidos*: apellido 1

Nombre único de usuario*: p1

Dni*: 11111111B

E-Mail*: email_1@uned.es

Departamento*: SISTEMAS DE COMUNICACION Y CONTROL

☐ Recibir notificaciones

Antigua Contraseña*:

Nueva Contraseña*:

Guardar cambios

2.4 Funcionalidades del profesor

Las opciones a su disposición son:

Ver propuestas de PFC: Lista las propuestas realizadas por el profesor



Esta opción permite filtrar por año

Crear propuesta de PFC: Crear una propuesta de proyecto.

El tipo de propuesta puede ser general o específica.

Ver PFC asignados: Desde aquí verá los PFC que se le han asignado, y podrá cerrarlos, asignando nota y adjuntando la memoria del mismo.

◦ PFC del profesor apellido 1, Pepe 1

- Seleccione año académico: Todos

- Filtrar por estado: Todos

PFC.titulo	Año	Alumno	Estado	Calificación	Acciones
1ª propuesta-especifica	2011	a3	desarrollo		Ver detalles/Cerrar

Pueden filtrarse por estado y por año. A la derecha, hay un link que muestra los detalles del proyecto y desde donde se puede cerrar.

Detalles de proyecto fin de carrera

Título	1ª propuesta-especifica
Profesor	p1
Alumno	a3
Tipo Proyecto	especifica
Conocimientos previos	1 conocimientos previos
Descripción	Entre las 1 mejores propuestas de proyecto
Objetivos	Aprender las 1 mejores técnicas de PM
Año	2011
Estado	desarrollo
Calificación	

Para cerrar el PFC ha de incluir:

- una memoria del mismo (* pendiente *)
- un archivo comprimido con el código del mismo (* pendiente *)
- la calificación final (* pendiente *)

Memoria: No file chosen

Esta pantalla explica lo que se necesita para cerrar el proyecto. Después de completar el proceso, el proyecto aparece cerrado y con dos links a su memoria, como el archivo comprimido con el código:

◊ PFC cerrado con éxito. Se ha enviado un mensaje al alumno!

Detalles de proyecto fin de carrera

Título	1ª propuesta-especifica
Profesor	p1
Alumno	a3
Tipo Proyecto	especifica
Conocimientos previos	1 conocimientos previos
Descripción	Entre las 1 mejores propuestas de proyecto
Objetivos	Aprender las 1 mejores técnicas de PM
Año	2011
Estado	cerrado
Calificación	10

Ver Memoria

Ver Código del proyecto

Ver Mensajes: Muestra los mensajes enviados al usuario.

Mensajes			
ID	Tipo	De	Información
8	PFCAsignado	Administrador	Asignación de PFC=1ª propuesta-especifica

2.5 Funcionalidades del alumno

Las opciones a su disposición son:

Ver propuestas de PFC: Ver listado de propuestas disponibles

Práctica de Sistemas Informáticos III

Ver propuestas por año académico: Todos ▾

Título	Año	Descripción	Tipo	Objetivos	Profesor	Conocimientos previos
2ª propuesta-general	2007	Entre las 2 mejores propuestas de proyecto	general	Aprender las 2 mejores técnicas de PM	apellido 2, Pepe 2	2 conocimientos previos
Pfc4	2011	descripción	especifica	muchos	p3, p3	Si
pfc5	2011	descripción 5	general	objetivos 5	p3, p3	no

Solicitar PFC: Permite enviar una solicitud de PFC sobre una propuesta. Pero solo una, ya que si tiene alguna pendiente o bien proyecto asignado, esta pantalla le presentará un mensaje de error:

El alumno a2 ya tiene una propuesta realizada (2ª propuesta-general).....

En cambio si el alumno no ha realizado propuesta, o esta ha sido rechazada se le presentan las propuestas abiertas y se le pide un fichero pdf de conocimientos.

Información: Cuando un alumno solicite la realización de una propuesta, deberá adjuntar un documento (en formato pdf) donde describa sus conocimientos y su adecuación al proyecto, para facilitar la asignación del mismo.

Seleccione la propuesta y adjunte el fichero de conocimientos:

Fichero de conocimientos: Choose File conocimientosAlumno.pdf

Solicitar

ID	Título	Año	Descripción	Tipo	Objetivos	Profesor	Conocimientos previos
1	1ª propuesta-especifica	2006	Entre las 1 mejores propuestas de proyecto	especifica	Aprender las 1 mejores técnicas de PM	apellido 1, Pepe 1	1 conocimientos previos
2	2ª propuesta-general	2007	Entre las 2 mejores propuestas de proyecto	general	Aprender las 2 mejores técnicas de PM	apellido 2, Pepe 2	2 conocimientos previos
4	Pfc4	2011	descripción	especifica	muchos	p3, p3	Si
6	pfc5	2011	descripción 5	general	objetivos 5	p3, p3	no

Una vez solicitado, deberá esperar que el administrador se lo confirme o rechace.

Ver Mensajes: Muestra los mensajes enviados al usuario.

Mensajes

ID	Tipo	De	Información
7	respuestaSolicitud	Administrador	Respuesta aceptada de solicitud de 1ª propuesta-especifica
9	disponibleNota	Profesor p1	Nota PFC=10

2.6 Funcionalidades del administrador

Las opciones a su disposición son:

Ver propuestas de PFC: Ver el listado de todas las propuestas realizadas, filtradas por año académico.

Ver propuestas por año académico: Todos ▾

Título	Año	Descripción	Tipo	Objetivos	Profesor	Conocimientos previos
1ª propuesta-especifica	2006	Entre las 1 mejores propuestas de proyecto	especifica	Aprender las 1 mejores técnicas de PM	apellido 1, Pepe 1	1 conocimientos previos
2ª propuesta-general	2007	Entre las 2 mejores propuestas de proyecto	general	Aprender las 2 mejores técnicas de PM	apellido 2, Pepe 2	2 conocimientos previos
Pfc4	2011	descripción	especifica	muchos	p3, p3	Si
pfc5	2011	descripción 5	general	objetivos 5	p3, p3	no

Ver solicitudes de PFC: Ver el listado de todas las solicitudes realizadas, filtradas por año académico.

Práctica de Sistemas Informáticos III

Ver Solicitudes de PFC

- Seleccione año académico: Todos

- Filtrar por estado: Todos

Revisar propuesta seleccionada

ID	Propuesta.titulo	tipo	Alumno Solicitante	Propuesta del profesor	Año	Estado	PDF conocimientos
1	1ª propuesta-específica	específica	apellido 1, Pepe 1	apellido 1, Pepe 1	2005	pendiente	PDF Conocimientos...
2	2ª propuesta-general	general	apellido 2, Pepe 2	apellido 2, Pepe 2	2005	pendiente	PDF Conocimientos...
3	1ª propuesta-específica	específica	apellido 3, Pepe 3	apellido 1, Pepe 1	2005	pendiente	PDF Conocimientos...
4	2ª propuesta-general	general	apellido 4, Pepe 4	apellido 2, Pepe 2	2005	pendiente	PDF Conocimientos...
5	Pfc4	específica	apellido 5, Pepe 5	p3, p3	2011	pendiente	PDF Conocimientos...

En la columna de la derecha, se puede observar un link al fichero pdf de conocimientos que subió el alumno al hacer la propuesta. Pulsar dicho link descarga el archivo en tu máquina.

Una vez seleccionada la solicitud que se quiere revisar, se pulsa el botón “revisar propuesta seleccionada”:

Ver Solicitudes de PFC

Seleccione al profesor a cargo del proyecto aceptado: p1

Aceptar propuesta seleccionada

Rechazar propuesta seleccionada

Seleccione profesor p1

ID	Propuesta.titulo	tipo	Alumno Solicitante	Propuesta del profesor	Año	Estado	PDF conocimientos
3	1ª propuesta-específica	específica	apellido 3, Pepe 3	apellido 1, Pepe 1	20053	pendiente	PDF Conocimientos...

Esta pantalla permite aceptar o rechazar la solicitud, pero si se acepta, hay que especificar el profesor que se hará cargo de llevar el proyecto. Este ejemplo, al tratarse de una propuesta “específica” solo puede tener al profesor que la creó como el encargado de la misma.

Asimismo, al aceptar esta propuesta, se rechazarán automáticamente el resto de solicitudes a la misma propuesta ya que esta es específica y por tanto solo asignable a un único alumno como se puede ver en la siguiente imagen:

ID	Propuesta.titulo	tipo	Alumno Solicitante	Propuesta del profesor	Año	Estado
1	1ª propuesta-específica	específica	apellido 1, Pepe 1	apellido 1, Pepe 1	20051	rechazada
2	2ª propuesta-general	general	apellido 2, Pepe 2	apellido 2, Pepe 2	20052	pendiente
3	1ª propuesta-específica	específica	apellido 3, Pepe 3	apellido 1, Pepe 1	20053	aceptada
4	2ª propuesta-general	general	apellido 4, Pepe 4	apellido 2, Pepe 2	20054	pendiente

Ver PFC: Este menú le permite ver el listado de todos los PFC en desarrollo, filtrados por año académico. Asimismo, esta opción le permite ver el listado de todos los PFC terminados, filtrados por año académico.

- Seleccione año académico: Todos

- Filtrar por estado: Todos

PFC.titulo	Año	Alumno	Estado	Calificación	Acciones
1ª propuesta-específica	2011	a3	cerrado	10	Ver detalles

Ver Mensajes: Muestra los mensajes enviados al usuario.

Mensajes			
ID	Tipo	De	Información
1	peticionSolicitud	Alumno a1	Solicitud de la propuesta=1ª propuesta-especifica
2	peticionSolicitud	Alumno a2	Solicitud de la propuesta=2ª propuesta-general
3	peticionSolicitud	Alumno a3	Solicitud de la propuesta=1ª propuesta-especifica
4	peticionSolicitud	Alumno a4	Solicitud de la propuesta=2ª propuesta-general
5	peticionSolicitud	Alumno a5	Solicitud de la propuesta=Pfc4

3 Manual de instalación

El archivo war generado con el contenido de la aplicación se sube a la aplicación de TomCat para el despliegue. La base de datos no es necesaria instalarla ya que se crea a través de código. Dentro de la pantalla de administrador se ha realizado dos operaciones, restaurar base de datos, que restaura la base de datos a su estado original y generar base de datos que lo que hace es generar el Script de ejecución de bases de datos.

4 Conclusiones

Esta práctica ha puesto de manifiesto la importancia de llevar los estudios aprendidos de un entorno puramente teórico a un entorno real mediante el empleo de herramientas profesionales pero a la vez al alcance de todos.

Hemos aprendido como planificar las tareas planteando una arquitectura y diseños en base a los casos de uso extrapolados del estudio, así como de los diagramas de secuencia

Dentro del entorno de desarrollo, muy interesante ha sido el emplear distintos frameworks para comprender cómo cada petición a una página web de una aplicación web j2ee implica una serie de pasos que muchas veces se escapan a nuestro conocimiento. Esta práctica ha supuesto un excelente caso de uso para profundizar en su comportamiento y poder aprovechar mejor las posibilidades que ofrecen.