ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADO EN INGENIERÍA INFORMÁTICA

GESTOR DE EVALIACIONES DE SISTEMAS DE INFORMACIÓN

MANAGER OF INFORMATION SYSTEMS EVALUATIONS

Realizado por
Mª Araceli Burgueño Caballero
Tutorizado por
Carlos Rossi Jiménez
Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, SEPTIEMBRE, 2016

Fecha defensa:
El Secretario del Tribunal

E.T.S. DE INGENIERÍA INFORMÁTICA, UNIVERSIDAD DE MÁLAGA

Resumen

En este Trabajo de Fin de Grado se va a desarrollar un sistema gestor de evaluaciones de sistemas de información dirigido principalmente para el uso educativo en la asignatura de Introducción a los Sistemas de Información del Grado de Ingeniería Informática en la Universidad de Málaga.

Este sistema, basado en arquitectura web, se encarga principalmente de la gestión de modelos de calificación, una herramienta muy útil para la toma de decisiones para la adquisición de sistemas de información. Estos modelos se basan en un conjunto de criterios que se aplican sobre varias aplicaciones. El resultado de esa asociación son las valoraciones de las aplicaciones para dichos criterios. Además, permite la gestión de datos relacionados y necesarios para los modelos de calificación, como aplicaciones, valoraciones, etc. En este sistema los alumnos pueden elaborar, modificar y eliminar información relativa a los modelos d calificación, aplicaciones, categorías y criterios, junto con su información personal.

De manera adicional a la funcionalidad que pueden usar los alumnos, los profesores pueden gestionar las puntuaciones sobre los modelos de calificación realizados por los alumnos, así como manejar la información perteneciente a los usuarios registrados en el sistema. También les permite la consulta de las autorías sobre las acciones que se realizan en este.

Para llevar a cabo este proyecto se ha seguido una variación de la metodología SCRUM, realizando varios sprint para el transcurso del proyecto. El diseño se realiza con UML en la herramienta Magic Draw, mientras que para el desarrollo de la aplicación se ha usado la tecnología Java EE, JavaScript y Bootstrap en el IDE Netbeans.

Palabras clave: Modelo de calificación, Gestor de evaluaciones, Java EE, método adquisición de sistemas.

E.T.S. DE INGENIERÍA INFORMÁTICA, UNIVERSIDAD DE MÁLAGA

Abstract

In this Degree Project a manager of information system evaluations has been developed. It is mainly carried out for educational use in the course Introduction to Information Systems, which is part of the Computing Degree at the University of Málaga.

This system, based on a web architecture, principally performs the management of rating models, a very useful tool in the decision-making process for rating information systems acquisitions. These models are composed of a set of criteria that is associated with several applications. The result of these associations is the valuation of these applications given the set of criteria. Furthermore, it supports the management of related data to rating models, such as apps, valuations and so on. In this application, students are allowed to create, read, update and delete information associated to rating models, applications, categories, criteria and personal information.

In addition to the functionality the students can use, professors can manage information about rating models marks, as well as users registered in the system. Finally, the application allows professors to look up authorships of the actions that are made in this tool.

In order to develop this project, a SCRUM methodology variation has been utilized. Several sprints has been made in the course of this project. The design is developed using UML in the Magic Draw tool, whereas the app development is carried out using Java EE, JavaScript and Bootstrap technology in the Netbeans IDE.

Keywords: Rating model, manager of evaluations, Java EE, system acquisition method.

Índice

Capítulo 1

Introducción

A día de hoy, en cualquier empresa se necesitan sistemas de información para cubrir los requisitos de los subsistemas de los que hacen uso para su gestión (gestión comercial, almacén, RRHH, CRM, ERP...).

La mayoría de las empresas, especialmente las PYMES (Pequeñas Y Medianas Empresas), no disponen de recursos suficientes para desarrollar software a medida para su empresa, de modo que eligen la opción de adquirir productos software para cubrir estas necesidades de gestión.

Una de las tareas de la consultoría de sistemas de gestión es la elección del producto software más adecuado para una empresa, después de haber realizado un estudio sobre esta para conocer sus necesidades y qué requisitos debe cumplir el sistema a adquirir.

Para cada uno de los subsistemas antes citados hay multitud de productos comercializados, con diferentes características y bajo diversas modalidades (software propietario, libre, SaaS, aplicaciones móviles, etc.). Se deben evaluar todas estas aplicaciones de modo que de entre todas las opciones se coja la que más se ajuste a las exigencias de la compañía en cuestión.

Con el fin de que los alumnos de la asignatura de Introducción a los Sistemas de Información conozcan y se familiaricen con métodos de consultoría para el asesoramiento en sistemas de información, se les presentan los modelos de calificación como una herramienta de evaluación. Un modelo de calificación se basa en la idea de valorar una serie de aplicaciones respecto a unos criterios, aquellas características o funcionalidades que una empresa demanda sobre los

sistemas. A cada criterio se le asigna una ponderación (numérica), esto es, la importancia de ese criterio dentro del modelo, y para cada aplicación y criterio se asocia una valoración (grado en el que la aplicación cumple con el criterio a valorar, también numérica). La calificación se obtiene como la multiplicación de la ponderación del criterio por la valoración de la aplicación para ese criterio. La calificación total de una aplicación es la suma de todas las calificaciones de los criterios. Por lo general, la aplicación "ganadora" será aquella con la calificación total más alta, es decir, aquella que se ajusta mejor a las necesidades exigidas, aunque también pueden influir otros criterios cualitativos.

Un ejemplo de modelo de calificación sería el siguiente:

Criterio	Ponderación	Valoración App 1	Calificación App 1	...
Escalabilidad	7	6	42	...
Computabilidad	6	8	48	...
Total			90	...

Tabla 1.1: Ejemplo de modelo de calificación

El sistema que se va a desarrollar tiene como objetivo dar soporte a la toma de decisiones relativa a la adquisición de sistemas de información. Más concretamente, permitirá:

- gestionar modelos de calificación y toda la información relacionada a estos (dando la posibilidad de administrar criterios, ponderaciones y valoraciones, categorías de sistemas de información, etc.)
- gestionar usuarios (con los perfiles de alumno, grupo de alumnos, profesor y administrador)
- gestionar las evaluaciones de diferentes sistemas de información por parte de diferentes usuarios.

Actualmente, no se encuentran en el mercado aplicaciones similares a la que en este proyecto se desarrolla. La herramientas de consultoría más similares a la aquí desarrollada son EvaluandoERP y SoftDoit. EvaluandoERP se centra principalmente en la evaluación de sistemas ERP, con el incoveniente de que

no muestra el modelo de evaluación de las aplicaciones, sino un resultado final con las valoraciones. La diferencia de EvaluandoERP con SoftDoit radica en que SoftDoit permite la evaluación de más sistemas empresariales aparte del ERP y sus módulos más destacados.

Como metolodgía de trabajo, se ha seguido una metodología similar a SCRUM, basada en sprints de trabajo. En estos sprints se permite la variación de carga de trabajo de unos sprints a otros, además de poder modificar los requisitos y el trabajo a realizar en primer lugar y cambiar la duración de los sprints conforme al trabajo pendiente.

Para el desarrollo del sistema, basado en una arquitectura web, se ha utilizado como tecnología básica HTML5, específicamente XHTML(eXtensible HyperText Markup Language), junto con CSS3 (Cascading Style Sheets). Estas tecnologías junto al framework Bootstrap, el cual permite hacer la aplicación responsive además de proporcionar el CSS, conforman el front-end de la aplicación.

Para el back-end, se ha usado la tecnología Java EE 7, que proporciona el framework JSF (Java Server Faces) para la comunicación entre el cliente y el servidor, así como EJB (Enterprise Java Bean) y JPA (Java Persistence API) para la gestión de la base de datos. Cómo SGBD (Sistema Gestor de Base de Datos) se va a usar Derby Apache.

Esta memoria explica el desarrollo de todo el proyecto del sistema gestor de evaluaciones, el cual se ha dividido en las siguientes fases y las cuales se explican a continuación brevemente y con más detalle en los siguientes capítulos.

- Planificación: se comenta la planificación final establecida para este proyecto, ya que se han producido cambios respecto de la planificación inicial que se ideó.
- Análisis de requisitos: se estudian los requerimientos software mínimos para desarrollar la aplicación. Consta de tres secciones: requisitos funcionales, no funcionales y de información.
- Arquitectura y tecnologías: expone la arquitectura elegida para desarrollar la aplicación, así como una descripción de la tecnología a usar en cada parte de la arquitectura.

- Diseño: se expone parte del diseño elaborado para la aplicación a partir de los requisitos extraídos mostrados en esta memoria. El resto del diseño no se presenta por cuestión de espacio en esta memoria.
- Desarrollo: explica cómo se ha llevado a cabo la implementación del sistema al completo y cómo cumple con el análisis y diseño del sistema.
- Pruebas: en esta sección se comenta el procedimiento a seguir para comprobar que la aplicación ha cumplido satisfactoriamente con lo estipulado en los requisitos y que realmente se ha desarrollado lo que se esperaba.
- Conclusiones y líneas futuras: expone el resultado final del proyecto, además de otra información derivada de la elaboración de este proyecto. Asimismo, se habla sobre posibles ampliaciones que mejoren el proyecto.
- Referencias bibliográficas: documentación que se ha utilizado a lo largo de este proyecto.
- Anexos: información adicional que no ha lugar en el cuerpo de esta memoria, pero que por su contenido, resulta de interés para el lector.

Basándonos en esta estructura, se procede a continuación a desarrollar cada uno de los puntos que permitan así conocer los detalles de todas y cada una de las fases por las que ha pasado este proyecto.

Capítulo 2

Planificación

Para un correcto desarrollo del proyecto en todas sus fases, se ha establecido una planificación de acuerdo a una variación de la metodología SCRUM, puesto que SCRUM se basa en un grupo de trabajo, y este proyecto es realizado únicamente por una persona.

El desarrollo al completo de este proyecto consta de las siguientes fases:

- Una fase previa al comienzo de la ejecución de este proyecto para la formación completa sobre la tecnología a usar, previsto su fin para junio de 2016.
- La primera fase o sprint de este trabajo consiste en el análisis y diseño completo de toda la funcionalidad que se incluye en la aplicación web, finalizado en la segunda semana de julio.
- La segunda fase del proyecto se compone de 3 sprints para el desarrollo de la funcionalidad especificada en el análisis.
    - El primer sprint de desarrollo abarca la funcionalidad más básica de la aplicación: validación de usuarios, CRUD de usuarios, de aplicaciones, categorías y criterios, con una duración de 2 semanas.
    - El segundo sprint incluye el desarrollo del CRUD de ponderaciones, valoraciones, modelos de calificación y puntuaciones, consulta de autorías y fusión de modelos. Duración de 3 semanas.
    - El tercer sprint se basa en el desarrollo de CRUD de grupos de alumnos, registro de un usuario, CRUD de asignaturas, gestión de perfil y

eliminar categorías y aplicaciones para profesores y administradores. Duración de 1 semana.

- En el quinto sprint se realizan las pruebas del sistema, comprobando su correcto funcionamiento, con una fecha tope para finales de agosto.
- La sexta y última fase consiste en la redacción de esta memoria de Trabajo de Fin de Grado, con una fecha de finalización máxima para el día 7 de septiembre.

En la figura 2.1 se muestra el diagrama de Gantt asociado a la planificación final, donde se muestra cuando se han realizado las tareas junto con las dependencias entre ellas.

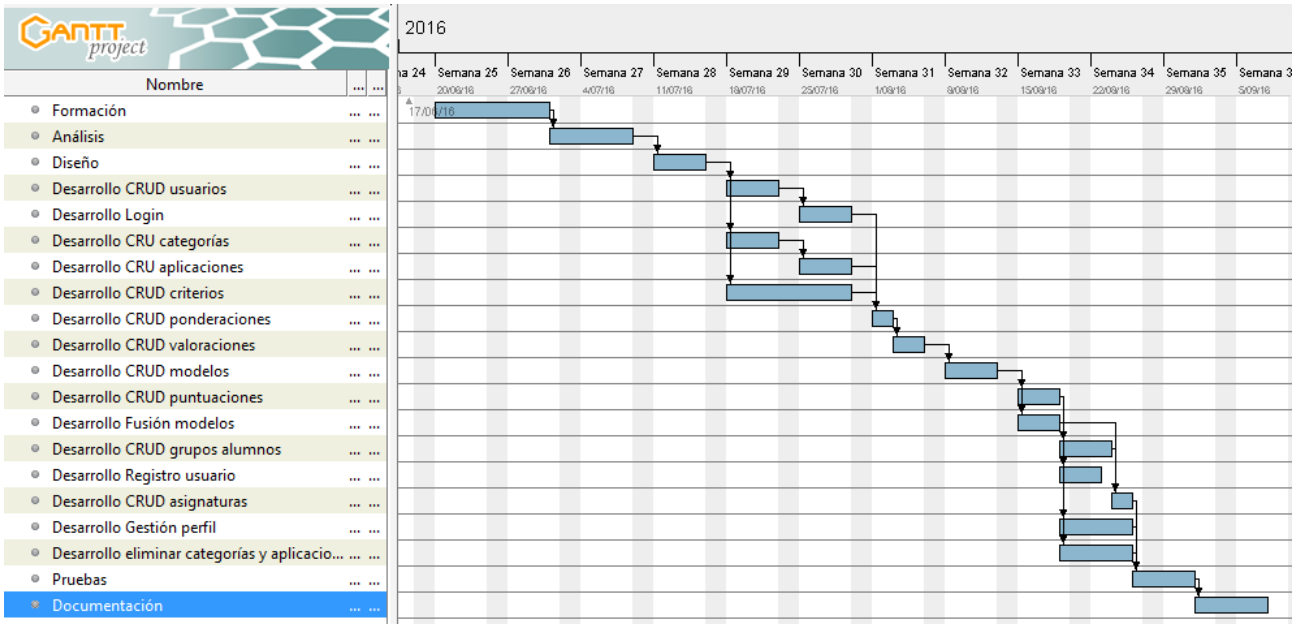


Figura 2.1: Gantt de la planificación.

La elección de esta metodología de trabajo de debe a la flexibilidad que esta permite para los cambios de los requisitos que puedan surgir y administrar de forma más libre la carga de trabajo entre los sprints.

Capítulo 3

Análisis de requisitos

En este capítulo se aborda el estudio de los diferentes requisitos que abarcarán en su completitud las necesidades a cubrir por el este proyecto. Además, se tiene en cuenta el tipo de usuario final de dicho sistema para conocer de mejor manera el uso que este va a tener, adaptarlo y hacerlo sencillo para estos usuarios.

Por lo tanto, atendiendo a las necesidades de este proyecto, los requisitos se pueden catalogar en tres grupos:

- Requisitos funcionales
- Requisitos no funcionales
- Requisitos de información

Así, una vez definidos el catálogo de usuarios y el catálogo de requisitos vemos descrito el alcance del sistema desarrollado en este proyecto y del cual se deducen los casos de uso que definen la funcionalidad de la aplicación.

3.1. Catálogo de usuarios

En este apartado se detallan los perfiles de usuarios que finalmente utilicen el sistema, pudiendo así conocer con más profundidad la funcionalidad que se espera de este y la cual debe tener.

De este modo, los distintos tipos de usuarios que pueden utilizar la aplicación son:

- Alumno: puede gestionar información relativa a los modelos de calificación, véase por ejemplo categorías, criterios y aplicaciones, además de gestionar modelos de calificación que le son propios. Una peculiaridad sobre este perfil puede ser el usuario grupo de alumnos, que tiene los mismos permisos sobre el sistema, con la diferencia de que el propietario no es una persona, sino un conjunto de alumnos denominados bajo este perfil.
- Profesor: además de toda la funcionalidad que el alumno cubre, el profesor es capaz de gestionar usuarios, puntuar modelos y fusionarlos, así como consultar las autorías de las acciones realizadas en en el sistema.
- Administrador: es el personal encargado de las gestiones más generales del sistema y de que todo funcione correctamente. Posee los permisos para utilizar la funcionalidad completa del sistema.

Una vez definidos los actores que interactúan con el sistema se procede a describir los requisitos que se necesitan para cumplir los objetivos esperados por cada uno de los perfiles.

3.2. Requisitos funcionales

En este punto se lista la funcionalidad a desarrollar en este Trabajo Fin de Grado, definiendo así el alcance del sistema y su frontera. Para una correcta formalización de los requisitos funcionales, son nombrados con las siglas RF (Requisito Funcional) seguidas de un número identificativo (correlativos para todos los requisitos), haciendo único dicho identificador.

De este modo, los requisitos funcionales son los que se muestran a continuación.

- **RF1: CRUD de categorías**. Creación, lectura y actualización de información perteneciente a las categorías de aplicaciones para los alumnos y, además, eliminar para los profesores y administradores. Se puede establecer una jerarquía entre estas.

- **RF2: CRUD de aplicaciones**. Creación, lectura y actualización de información perteneciente a las aplicaciones para los alumnos y, además, eliminar para los profesores y administradores.
- **RF3: CRUD de usuarios**. Creación, lectura, actualización y eliminación de información perteneciente a los usuarios (alumnos, profesores y administradores) registrados en el sistema. Solo los administradores pueden gestionar información sobre otros administradores. Los alumnos no pueden gestionar otra información que no sea la suya.
- **RF4: CRUD de criterios**. Creación, lectura, actualización y eliminación de información perteneciente a los criterios que se usarán para elaborar los modelos de calificación.
- **RF5: CRUD de modelos de calificación**. Creación, lectura, actualización y eliminación de información perteneciente a los modelos de calificación construídos sobre varias aplicaciones ya registradas a partir de criterios previamente definidos.
- **RF6: Consultar autorías**. Consulta del log con las autorías del sistema sobre los elementos modificables de este.
- **RF7: Gestión de claves**. Gestión de los usuarios y claves que tienen tanto alumnos como profesores y administradores para acceder al sistema.
- **RF8: Fusión de modelos**. Permitir de manera manual a un profesor fusionar dos o más modelos de calificación en uno solo, comparando criterios, ponderaciones, etc.
- **RF9: CRUD de grupos de alumnos**. Creación, lectura, actualización y eliminación de información relativa a los grupos de alumnos.
- **RF10: CRUD de asignaturas**. Creación, lectura, actualización y eliminación de información relativa a las asignaturas dentro del sistema.
- **RF11: Login**. Autenticación de los usuarios para poder ingresar en el sistema.
- **RF12: Gestión de perfiles**. Cada usuario podrá leer, actualizar y eliminar parte o toda de su información básica almacenada.

- **RF13: CRUD de ponderaciones**. Creación, lectura, actualización y eliminación de información perteneciente a las ponderaciones relacionadas con criterios utilizados en un modelo de calificación.
- **RF14: CRUD de valoraciones**. Creación, lectura, actualización y eliminación de información perteneciente a las valoraciones relacionadas con las ponderaciones de un modelo de calificación y una aplicación asociada a este modelo.
- **RF15: CRUD puntuaciones**. Un profesor podrá crear, leer, actualizar y eliminar puntuaciones sobre los modelos de calificación. Un alumno podrá consultar la puntuación realizada sobre sus modelos de calificación.
- **RF16: Registrar usuario**. Un usuario (alumno) podrá registrarse en el sistema para tener una cuenta.
- **RF17: Recordar contraseña**. Recordar la contraseña de un usuario de forma que el cuando se visite de nuevo la página el buscador e introduzca su DNI el navegador rellene la contraseña por el usuario.

Con la realización de estos requisitos se cubre por completo lo esperado del gestor de evaluaciones de este proyecto. El modelo de casos de uso diseñado para este sistema cubre el 100 % de estos requisitos.

3.3. Requisitos no funcionales

A continuación se especifican los atributos de calidad que pueden usarse para describir cómo operará el sistema a desarrollar para los requisitos expuestos en el apartado anterior. Del mismo modo que para los requisitos funcionales, la notación de los requisitos no funcionales se compone de las siglas RNF (Requisito No Funcional) seguido de un número identificativo, haciendo único el identificador. Así, los requisitos no funcionales para este proyecto son los que abajo se muestran:

- **RNF1: Facilidad de uso**. El sistema debe ser intuitivo para cualquier usuario del sistema, tanto alumnos como profesores y administradores.

- **RNF2: Tolerancia a fallos**. El sistema debe ser capaz de recuperarse ante determinados fallos.
- **RNF3: Sistema en tiempo real**. El sistema debe actualizarse de forma instantánea para que cualquier cambio quede reflejado en este y cualquier persona que acceda pueda observar dicho cambio en ese mismo momento.
- **RNF4: Soporte multiplataforma**. El sistema debe funcionar tanto en ordenadores como en dispositivos móviles a través del navegador. La aplicación web debe poseer un diseño responsive a fin de garantizar la adecuada visualización en múltiples dispositivos.
- **RNF5: Tiempo de respuesta del sistema**. El sistema deberá responder a una solicitud de la información en un tiempo máximo de veinte segundos.
- **RNF6: Gestión de acceso por parte del/de los administradores**. El/los administrador/es serán los encargados de crear las cuentas de los grupos de alumnos en el sistema y de las asignaturas.
- **RNF7: Almacenamiento de toda la información en una única base de datos**. Toda la información se almacena en una base de datos para que sea accesible en todo momento. Además, la BD deberá disponer de un pool de conexiones configurables en número para que la aplicación sea escalable en función de los recursos hardware y software disponibles.
- **RNF8: Seguridad en las contraseñas**. El sistema solo guardará los hash de las contraseñas para garantizar una mayor seguridad de los usuarios.
- **RNF9: Intuitividad para el uso**. Las interfaces de usuario deberán ser amigables e intuitivas para una navegación más cómoda.
- **RNF10: Acceso al sistema**. Los permisos de acceso al sistema podrán ser cambiados solamente por el administrador.
- **RNF11: Manejo de errores**. El sistema debe proporcionar mensajes de error que sean informativos y orientados a los usuarios finales.

Con el cumplimiento de este listado de requisitos podemos definir cómo se comportará el sistema, haciendo seguro, sencillo y eficaz.

3.4. Requisitos de información

De manera análoga a cómo se han descrito los requisitos funcionales y no funcionales, desarrollamos el listado de información necesaria para poder llevar a cabo los procesos dentro del sistema. La notación es la misma que en los puntos anteriores: se denotan con las siglas RI (Requisitos de Información) seguido de un número, siendo así un identificador único. Así pues, los requisitos de información son los siguientes:

- **RI1: Categoría**. Cuenta con información de ID (identificador), nombre y descripción sobre conjuntos de aplicaciones con características comunes.
- **RI2: Aplicación**. Asociada a información como el ID, nombre y descripción de una aplicación software.
- **RI3: Criterio**. Describe la información básica de un criterio: nombre y descripción.
- **RI4: Modelo de calificación**. Un modelo de calificación dispondrá de un nombre, descripción, visibilidad y valoración.
- **RI5: Ponderación**. Asociada a un modelo y un criterio se encuentra una ponderación, la cual indica la ponderación dentro del modelo de calificación.
- **RI6: Valoración**. Una valoración se asocia a una ponderación y a una aplicación, la cual se encuentra asociada a su vez al modelo de calificación de la ponderación. Establece la valoración que tendrá la aplicación para un determinado criterio dentro de un modelo.
- **RI7: Autoría**. Indica cualquier cambio realizado para un conjunto de objetos o entidades que se almacenan en el sistema. En esta se establece el usuario que realiza la modificación (creación, actualización o eliminación) junto con el objeto y la fecha.
- **RI8: Asignatura**. Establece las posibles asignaturas, nombre y curso, que se pueden dar lugar en el sistema.
- **RI9: Usuario**. Consta de toda la información básica de un usuario, ya sea DNI, nombre, apellidos, contraseña, etc.

- **RI10: Grupo de alumnos**. Similar al usuario, se compone información básica sobre el grupo, como el identificador, los alumnos que lo componen, contraseña, etc.
- **RI11: Puntuación**. Establece la puntuación dada por un profesor para un determinado modelo de calificación junto con un comentario opcional.

Con estos requisitos de información el sistema gestionará toda la información necesaria almacenable para el correcto desarrollo de la funcionalidad especificada con anterioridad.

3.5. Casos de uso

En este apartado se describe el modelado para implementar la funcionalidad del sistema, que se especifica a partir de los casos de uso de análisis, los cuales proporcionan una descripción de las actividades que se pueden realizar dentro del sistema para llevar a cabo un proceso. Se contemplan tanto los escenarios normales (escenarios donde el proceso de ejecuta con éxito) como los escenarios alternativos (escenarios de excepción donde por determinados motivos no se alcanza el objetivo esperado).

El diagrama de casos de uso UML para el gestor de evaluaciones puede ser el de la Figura 3.1.

Esta funcionalidad se puede dividir según el perfil que accede al sistema (actor): alumno (o grupo de alumnos, que tiene asociada la misma funcionalidad), profesor y administrador. Un profesor puede realizar toda la funcionalidad de un alumno además de otra que los alumnos no pueden realizar. De igual manera, un administrador puede realizar toda la funcionalidad de un profesor aparte de otra que puede realizar un administrador que un profesor no puede.

3.5.1. Casos de uso para un alumno

El alumno (o grupo de alumnos) es el perfil más básico del sistema, el que tiene la funcionalidad más restringida. El diagrama de la Figura 3.2 se muestra es una ampliación del diagrama de casos de uso del sistema focalizado en el alumno.

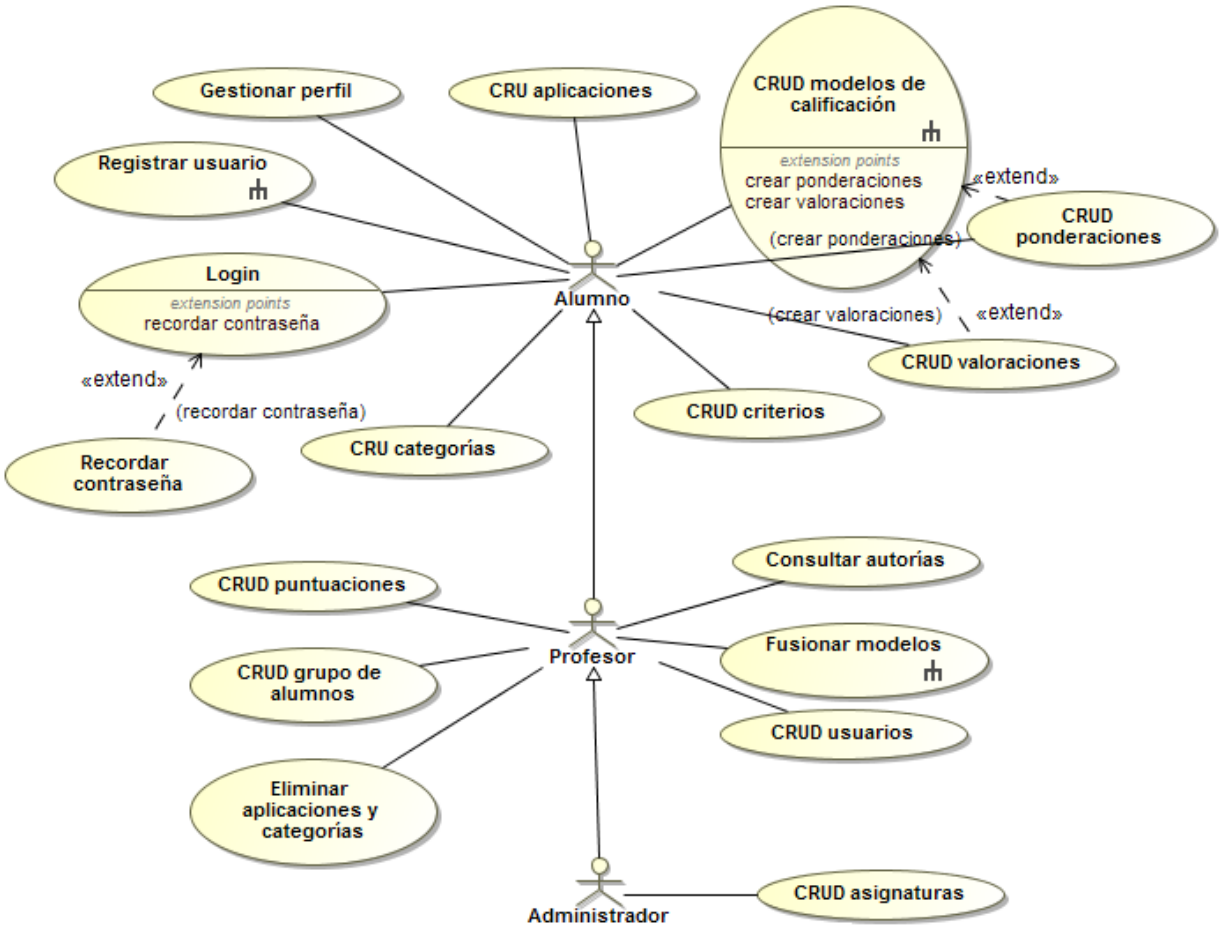


Figura 3.1: Diagrama de casos de uso del sistema.

Lo primero que debe realizar un usuario es identificarse en el sistema, introduciendo su DNI y contraseña. Nótese que si la validación del usuario no se realiza con éxito se mostrará un mensaje de error por los motivos por los que no se haya podido realizar la identificación (Login). Opcionalmente, el usuario podrá recordar su contraseña para próximas veces que acceda al sistema (Recordar contraseña).

Si un alumno no se ha registrado, desde la página de identificación puede dirigirse a la página de registro desde la que el alumno podrá darse de alta introduciendo sus datos básicos (Registrar usuario). Esta es la única funcionalidad que un grupo de alumnos no puede realizar que un alumno sí puede.

Una vez autenticado, el alumno elige la asignatura desde la que quiere traba-

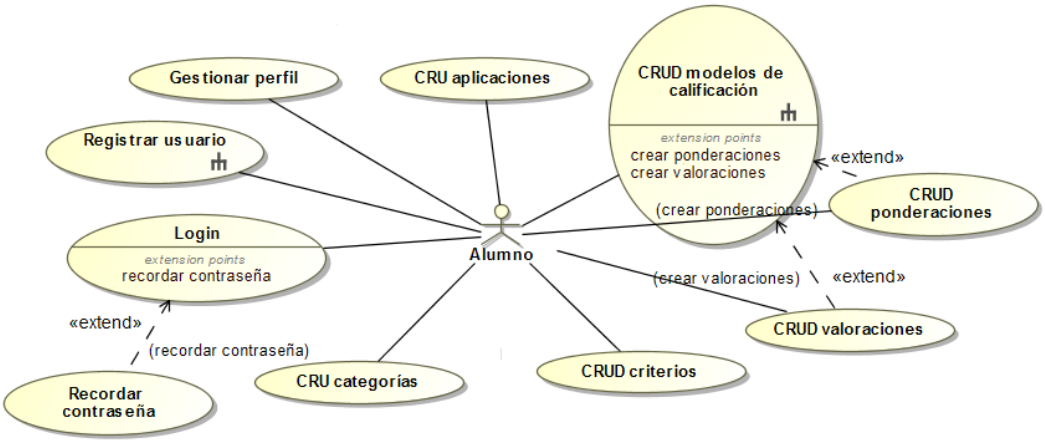


Figura 3.2: Casos de uso para un alumno.

jar, y cuando la haya elegido, se redirige para que vea sus modelos de calificación en esa asignatura.

Puede gestionar distintas entidades: puede manejar los datos del banco de categorías y aplicaciones, crear nuevas, acceder a los datos y modificar, pero no puede eliminar (CRU categorías y CRU aplicaciones). A modo ilustrativo, un ejemplo de descripción de escenarios del caso de uso CRU aplicaciones sería:

Escenario normal:

1. *Accede a la sección de aplicaciones.*
2. *Introduce los datos para crear una aplicación.*
3. *El usuario solicita crear la aplicación.*
4. *La aplicación es creada y muestra un mensaje de que ha sido creada.*

Escenario alternativo:

1. *Accede a la sección de aplicaciones.*
2. *Introduce los datos para crear una aplicación.*
3. *El usuario solicita crear la aplicación.*
4. *Se muestra un mensaje de error porque la aplicación ya existe.*

Del mismo modo que con las categorías y aplicaciones, el usuario puede gestionar todos los criterios que existen en el sistema, con la diferencia de que

sí puede eliminar los criterios.

Si existen categorías, aplicaciones y criterios, el usuario puede crear un modelo de calificación. Para crearlo es necesario primero introducir su información básica (nombre, descripción, aplicaciones, etc.). A continuación selecciona las ponderaciones para los criterios y después las valoraciones para los criterios por las aplicaciones selecciones. Para modificar se sigue el mismo procedimiento. Para ello es necesario incluir la funcionalidad de la gestión de ponderaciones y valoraciones en la gestión de los modelos de calificación.

Por último, el usuario también puede gestionar las valoraciones y ponderaciones sin necesidad de acceder directamente por el modelo, sino a través de interfaces distintas.

3.5.2. Casos de uso para un profesor

Además de toda la funcionalidad accesible por un alumno, un profesor es capaz de realizar las actividades que se muestran en el siguiente diagrama de casos de uso.

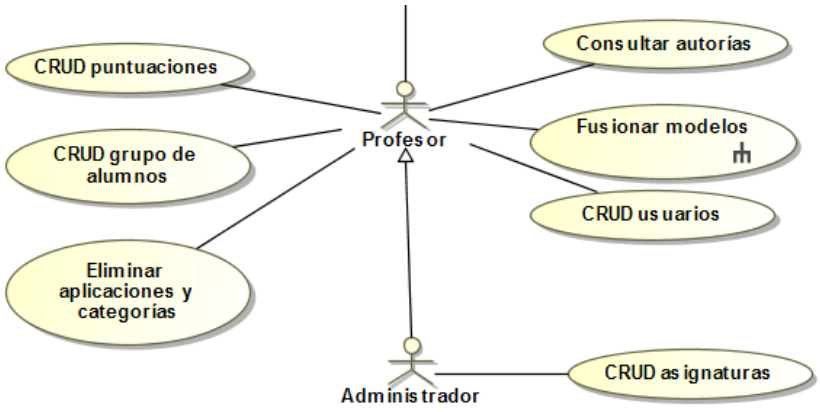


Figura 3.3: Casos de uso para profesor y administrador.

Un profesor puede eliminar categorías y aplicaciones (acción que los alumnos no pueden ejecutar). Además, puede consultar las autorías sobre los objetos que se modifican en el sistema, pudiendo visualizar quien ha realizado la acción, qué acción ha realizado, sobre qué objeto y cuándo.

Además de crear modelos como los alumnos, puede fusionar modelos, es decir, puede elegir múltiples modelos y a partir del conjunto de sus criterios, ponderaciones y valoraciones, elegir aquellos que considere más adecuados para hacer un modelo de calificación más completo que los ya existentes.

El profesor tiene la opción además de gestionar la información de los usuarios registrados en el sistema, con la salvedad de los administradores. Del mismo modo, puede gestionar la información relativa a los grupos de alumnos, permitiendo hacer las operaciones básica CRUD sobre el grupo (del inglés Create, Read, Update y Delete al español Crear, Leer, Actualizar y Eliminar).

En último lugar, un profesor puede puntuar un modelo de calificación, de forma que esa puntuación sea visible cuando se consulte el modelo de calificación. Un modelo tendrá una única puntuación, la cual se puede modificar y/o eliminar si se desea.

3.5.3. Casos de uso para un administrador

Como se observa en la figura del subapartado anterior, un administrador hereda todas las acciones que puede realizar un profesor, y además este puede modificar información sobre los administradores y sobre las asignaturas existente en el sistema.

Así pues, se concluye el análisis del gestor de evaluaciones con el establecimiento de todos los requisitos y casos de uso necesarios definidos en los subapartados anteriores que serán usados para el diseño del sistema y, subsiguientemente, para su desarrollo, comprobando su auténtica inclusión en las pruebas de sistemas.

Capítulo 4

Arquitectura y tecnología

A continuación, se procede a describir la arquitectura usada para ejecutar el desarrollo del sistema acorde a los requisitos especificados en el punto anterior. De manera adicional, se describe la tecnología utilizada para un desarrollo correcto y eficiente.

Para el desarrollo se ha elegido la tecnología Java EE 7 (Java Enterprise Edition), la cual ya establece la arquitectura básica del sistema. Esta se compone de tres capas: cliente, servidor Java EE y base de datos; las cuales se describen a continuación, y cuyas relaciones se pueden observar en la figura 4.1.

Por lo tanto, las capas de la arquitectura son:

- Cliente: en este nivel se ejecutan los procesos en las máquinas cliente que solicitan información. La información básica que requieren son las páginas web desde las que podrán acceder a la aplicación. El único requisito que se exige para esta capa es el acceso a través de un navegador web.

  Para la capa de presentación se ha hecho uso del lenguaje HTML5 (realmente implementado como XHTML) y CSS3. Además, se hace uso del framework Bootstrap, el cual permite que los componentes y elementos de HTML5 sean más atractivos visualmente y, adicionalmente, hacer que todas las interfaces de usuario sean responsive, es decir, puedan adaptarse a cualquier tamaño de pantalla.

- Servidor Java EE: esta capa se encarga de toda la lógica del sistema, la cual a su vez se compone de dos subcapas que se comunican entre ellas:

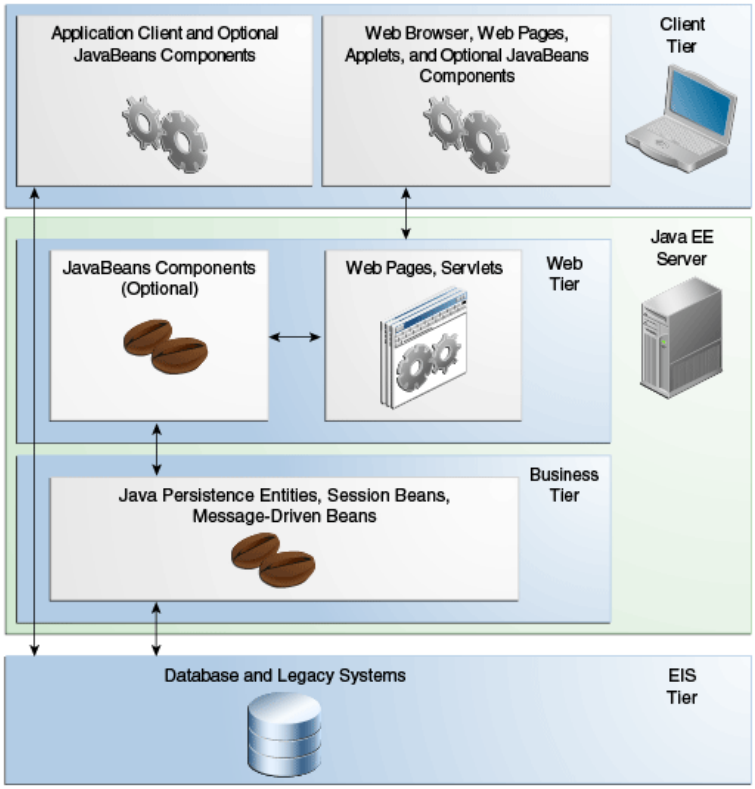


Figura 4.1: Arquitectura del sistema.

- Nivel web: en él se disponen las páginas web, esto es, las páginas XHTML modificadas para que haciendo uso del framework JSF se comuniquen con los controladores lógicos y estos los provean de la información necesaria para formar las páginas HTML.

  Asimismo, en la subcapa web se encuentran también los controladores que comunican las páginas web con la (sub)capa de negocio. Estos controladores se encargan de pedir datos a la capa inferior y proveerlos a la superior, control básico de entrada de datos (expresiones regulares, campos vacíos, etc.).

  Todos los controladores son implementados usando CDI y con un ámbito que bien puede ser de petición (Request Scoped) si el ámbito en el que va a existir es solo para una petición HTML, o bien de sesión (Session Scoped) si se requiere que el controlador exista durante más tiempo que una petición HTML, es decir, durante una sesión de navegación completa.

- Capa de negocio: en este nivel se pueden encontrar los bean de sesión, encargados de controlar el SGBD de la base de datos subyacente. La tecnología Java EE que permite implementar estos bean son los EJB.

  También se incluyen en esta capa las clases de entidad a usar en la base de datos, que ayudan a la gestión entre el servidor web y el SGBD.

El servidor Java EE es el encargado de comunicar la capa cliente con la de base de datos y aportar la lógica y coherencia al sistema.

- Base de datos: esta capa se encarga de toda la gestión de la base de datos de la que va a hacer uso la aplicación para almacenar los datos. La plataforma Java provee un framework para simplificar el manejo de los datos entre la aplicación y la base de datos, JPA. JPA solo permite la gestión de datos relacionales, por lo que la base de datos que se usa en este proyecto es una base de datos relacional.

  Para el acceso a los datos se hará uso de JPQL (Java Persistence Query Language), un lenguaje independiente de la plataforma que permite hacer consultas (de tipo SELECT, UPDATE y DELETE) sobre entidades almacenadas en la base de datos.

El despliegue del sistema se puede visualizar en el diagrama de red, donde se observa el hardware utilizado para el funcionamiento de la aplicación y la estructura de red que presenta este.

Cualquier usuario puede acceder al sistema a través del navegador de un ordenador. Este por HTTP pide la información al servidor de aplicaciones en el que se aloja la aplicación. El servidor de aplicaciones a su vez accede al SGBD en caso de ser necesaria la consulta, inserción o eliminación de datos en la base de datos.

En principio se han dividido los datos de la información de los usuarios del resto de datos almacenados. No obstante, no necesariamente tiene que ser el resultado final.

Por último, el administrador puede acceder directamente al servidor de aplicaciones por una intranet que conecta el ordenador desde el que accede el administrador y el servidor.

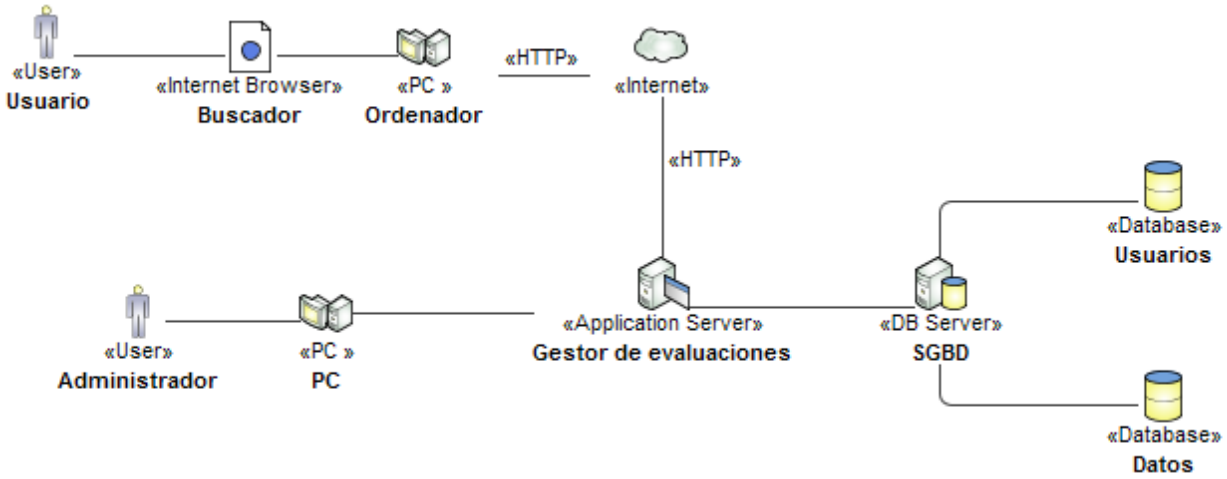


Figura 4.2: Despliegue del sistema.

El sistema gestor de evaluaciones se despliega sobre un servidor Glassfish 4.1 (versión más actual de este), servidor de aplicaciones creado especialmente para aplicaciones que corren haciendo uso de la tecnología Java EE. Sin embargo, no está de más añadir que también puede ejecutarse sobre un servidor Apache Tomcat.

Capítulo 5

Diseño

En este apartado se define el diseño del sistema gestor de evaluaciones que se debe cumplir para satisfacer los requisitos detallados en el apartado Análisis de requisitos de esta memoria.

De acuerdo a la arquitectura que tiene la aplicación podemos dividir el diseño de la aplicación en el diseño de las interfaces de usuario (páginas XHTML) y diseño de la base de datos.

5.1. Interfaces de usuario

A la hora de realizar las interfaces se ha optado por un diseño "minimalista", esto es, simple y conciso. Introduciendo únicamente los componentes que son esenciales para una correcta comprensión y uso adecuado de la interfaz.

Cada una de las vistas ha sido diseñada con un enfoque web, por lo que como layout principal se usa un contenedor que divide la vista en tres secciones (visibles en la figura 5.1):

- Barra de navegación: desde la barra de navegación se puede acceder a todas las secciones y/o funcionalidades accesibles por el perfil del usuario logueado. Esta cambia en función del perfil del usuario propietario de la sesión, puesto que no tendrá los mismos contenidos un alumno que un profesor o administrador.
- Contenido: es la sección principal de la interfaz en la que se expone la información que se precise de la aplicación. Es distinta para cada interfaz pudiendo dividirse a su vez en varias secciones internas.

- Pie de página: muestra información adicional que resulta de interés para el usuario, en este caso el nombre del sistema y la asignatura a la que pertenece.

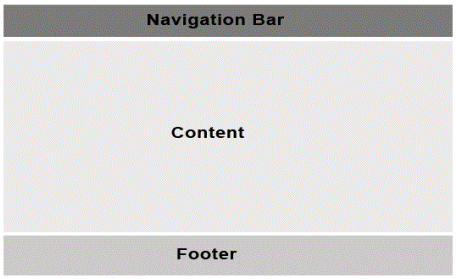


Figura 5.1: Layout principal.

Las vistas de validación del usuario y registro de un nuevo usuario no contarán con este layout, la diferencia existente es que en lugar de una barra de navegación habrá una cabecera con el nombre del sistema y la sección del contenido principal.

Para cada una de estas funcionalidades se ha realizado una o varias vistas personalizadas, adaptada a cada una de las necesidades de la misma, intentando cumplir todos los requerimientos de cada uno de los casos de uso que deben de cubrir (escenarios de éxito y escenarios alternativos).

Algunas de las maquetas en las que se han basado algunas de las interfaces se muestran en las siguientes figuras, donde se puede ver el maquetado de la gestión de los usuarios y de los dos primeros pasos de la creación de un modelo de calificación.

Desde las mismas interfaces se puede controlar si el contenido es renderizado o no dependiendo del perfil con el que se accede. A modo ilustrativo, si un usuario no validado intenta acceder a la gestión de usuarios, no se muestra contenido alguno (el contenido HTML no se renderiza) y además, se le redirecciona a la vista donde puede identificarse.

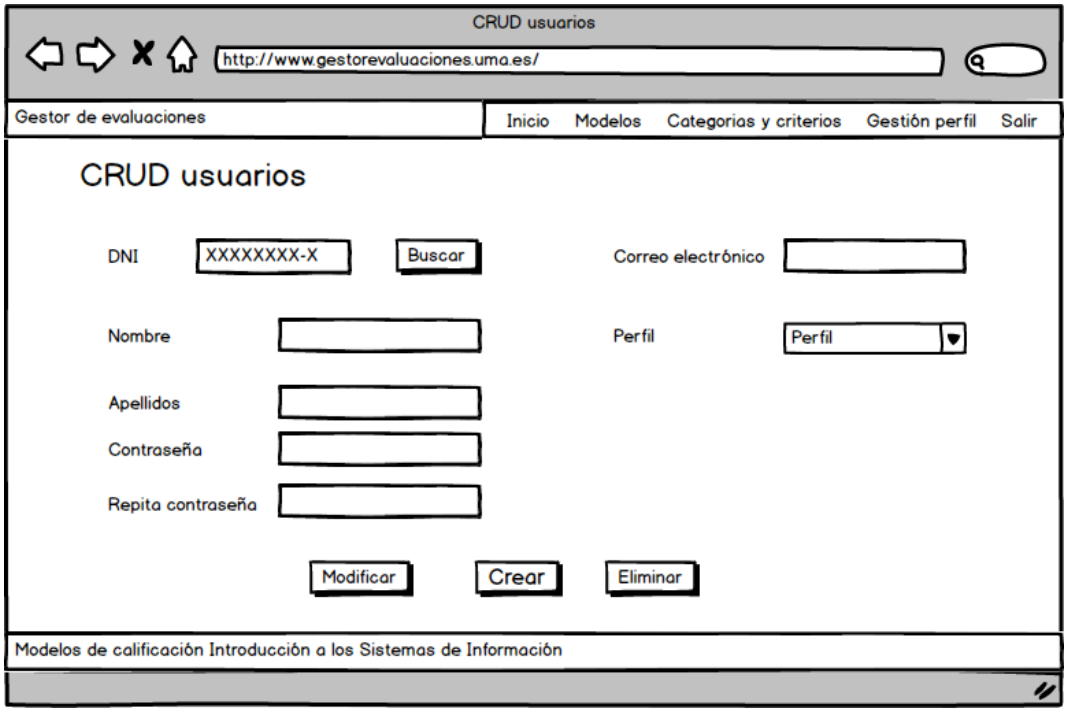


Figura 5.2: Gestión de usuarios.

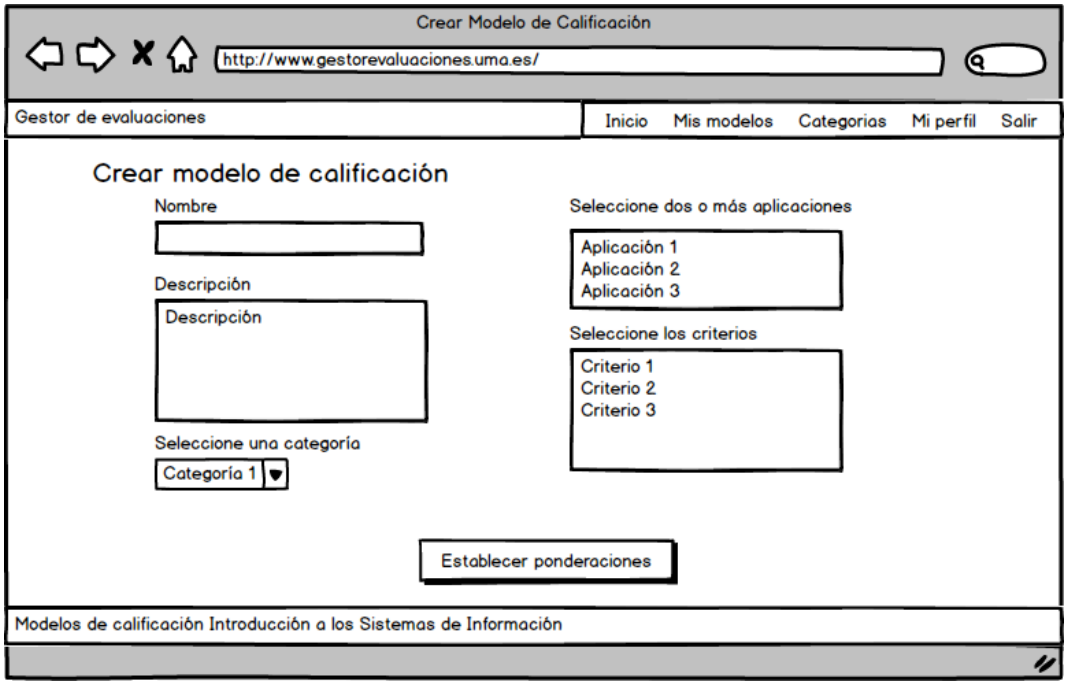


Figura 5.3: Datos para crear un modelo.

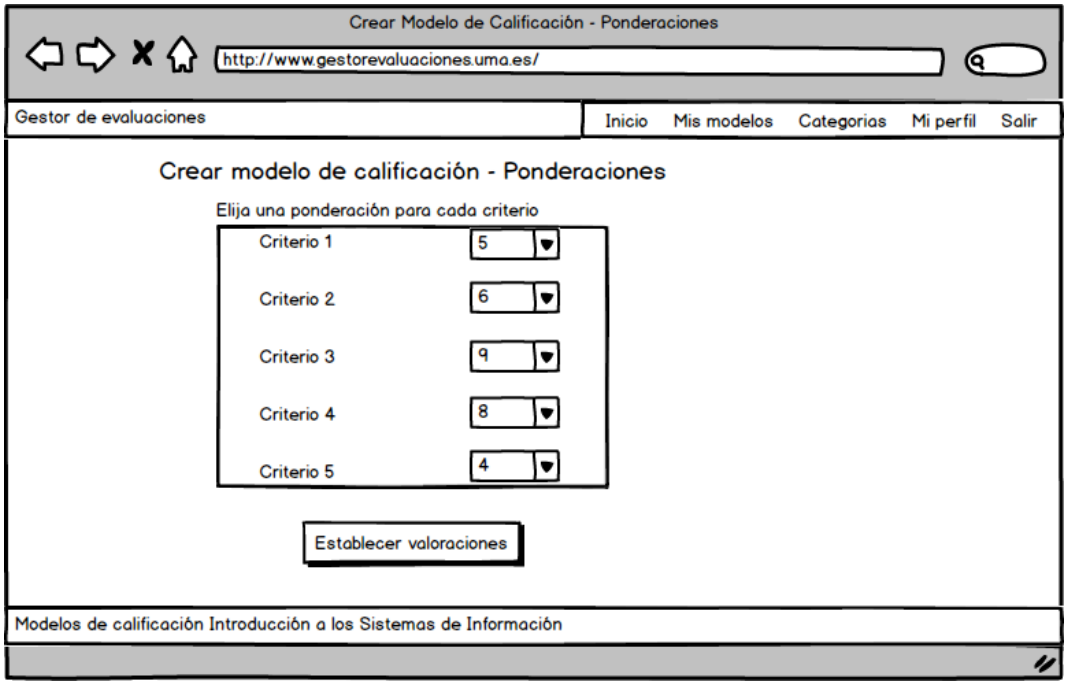


Figura 5.4: Ponderaciones de un modelo.

5.2. Modelo de estructura

Como se comentó en el apartado de Arquitectura y tecnología, la base de datos que se usa en este proyecto es una base de datos relacional, donde los datos se almacenan en tablas, correspondiéndose estas con entidades a nivel conceptual, y estas se relacionan unas con otras por medio de relaciones.

A partir de los requisitos de información podemos obtener fácilmente las clases que se gestionan en el sistema y las relaciones, que se infieren de los casos de uso de análisis. En la figura 5.5 se observa el diagrama de clases de entidad, el cual muestra la estructura del sistema: las clases, sus atributos, métodos y relaciones entre los objetos.

Una vez validado el modelo de clases de entidad, se derivan las entidades que componen la base de datos. A partir de diagrama de clases de entidad se elaboró el diagrama Entidad/Relación para mapear las clases a entidades en la base de datos. Por lo tanto, el diagrama E/R resultante que se desplegará en la

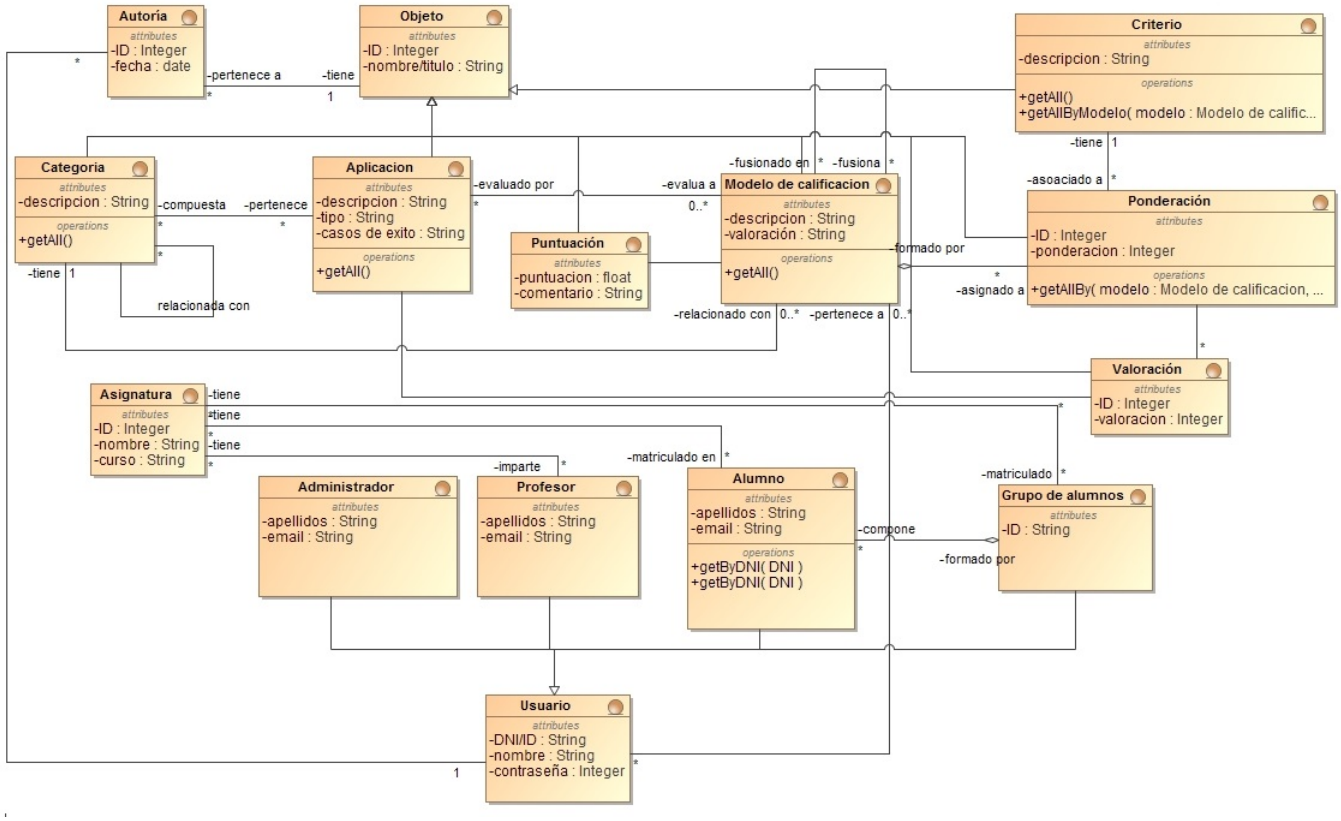


Figura 5.5: Clases de entidad.

base de datos se muestra en la figura 5.6.

La entidad Usuario cuenta con la clave primaria DNI, y contempla cuatro subentidades, cada una con unos atributos y relaciones específicos como se observan en la figura. Estas subentidades corresponden a cada uno de los distintos perfiles con los que se puede acceder al sistema.

La entidad Objeto engloba al conjunto de subentidades de las que debe existir una autoría. Tiene por clave primaria un identificador numérico simple por motivos de eficiencia. Las subentidades incluídas en Objeto son:

- Categoría: una categoría puede estar relacionada con muchas aplicaciones y a su vez con otras categorías estableciendo así una jerarquía entre categorías. También puede relacionarse con múltiples modelos de calificación.
- Criterio: un criterio puede estar relacionado con un modelo de calificación

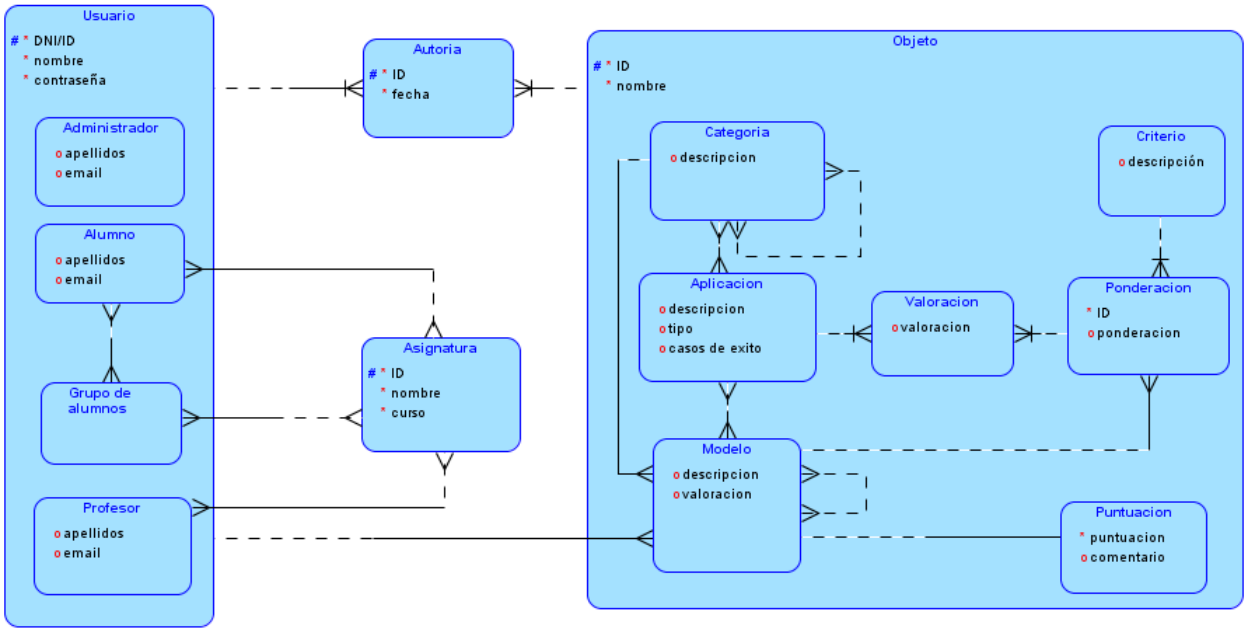


Figura 5.6: Diagrama Entidad/Relación.

a través de la ponderación que recibe ese criterio para el modelo.

- Ponderación: recoge la asociación entre el modelo de calificación y el criterio.
- Aplicación: una aplicación puede relacionarse con múltiples modelos de calificación y con valoraciones de modelos de calificación (descripción más adelante).
- Modelo de calificación: un modelo de calificación cuenta con una valoración de la aplicación ganadora del modelo. Cuando se crea un modelo es obligatorio que esté relacionado con un usuario (su propietario) con una categoría y con al menos dos aplicaciones.
- Valoración: una valoración establece la relevancia que tiene un determinado criterio dentro de un modelo de calificación (ponderación) para una aplicación dentro del modelo.
- Puntuación: un profesor puede calificar un modelo asignándole una nota numérica y adicionalmente añadiendo un comentario a modo de corrección o valoración.

Por último, la entidad Autoría relaciona el cambio en un objeto (ya sea su creación, modificación o eliminación) y lo asocia al usuario responsable de este cambio junto con la fecha en la que este se produce. La entidad Asignatura refleja la asignatura en la que los alumnos, grupos de alumnos y profesores pueden estar matriculados o impartiendo clase.

Con este diseño para la base de datos incluímos todos los requisitos de información, necesarios para poder desarrollar este proyecto de acuerdo a los objetivos establecidos.

5.3. Diseño de los objetos

En este subapartado se muestra el diseño realizado para la interacción entre los objetos del sistema y las distintas transiciones que un objeto puede tener, una vez se ha diseñado y validado el diagrama de clases del apartado anterior.

El flujo de acciones a desarrollar en un sistema es muy importante. Por lo tanto, se elaboraron también varios diagramas de secuencia donde se reflejan las acciones a realizar y la interacción entre objetos dentro de la herramienta. Solo se realizaron diagramas de secuencia para aquellos casos de uso considerados más importantes para el sistema, y solo para un escenario de éxito, ya que los demás resultarían similares. Todos los diagramas de secuencia que se han realizado son: crear modelo de calificación, registrar usuario y fusionar modelos de calificación, todos ellos elaborados para su caso de éxito.

Del mismo modo, se han realizado diagramas de máquinas de estado para identificar los estados y transiciones de varias vistas según recorre su ciclo de vida. Los diagramas de máquinas de estado realizados para la aplicación son: *ViewCRUDusuarios* para la vista que gestiona la información de los usuarios, *ViewFusionarModelos* y *ViewFusionarCriterios* para los dos primeras vistas a la hora de fusionar varios modelos y *VIewGestionarPerfil* para la vista que se encarga de modificar la información personal de un usuario.

En la figura 5.7 se muestra el diagrama de secuencia del caso de uso Crear modelo de calificación para el escenario de éxito.

El diagrama de máquinas de estado de la figura 5.8 representa los posibles estados y transiciones de la vista de gestión de usuarios, accesible para profesores y administradores.

El resto de diagramas que no se encuentran en la memoria pueden consultarse en la documentación entregada.

Ya descrito todo el diseño sobre la funcionalidad a implementar para la aplicación, se comprueba que, efectivamente, se satisfacen todos los requisitos funcionales especificados.

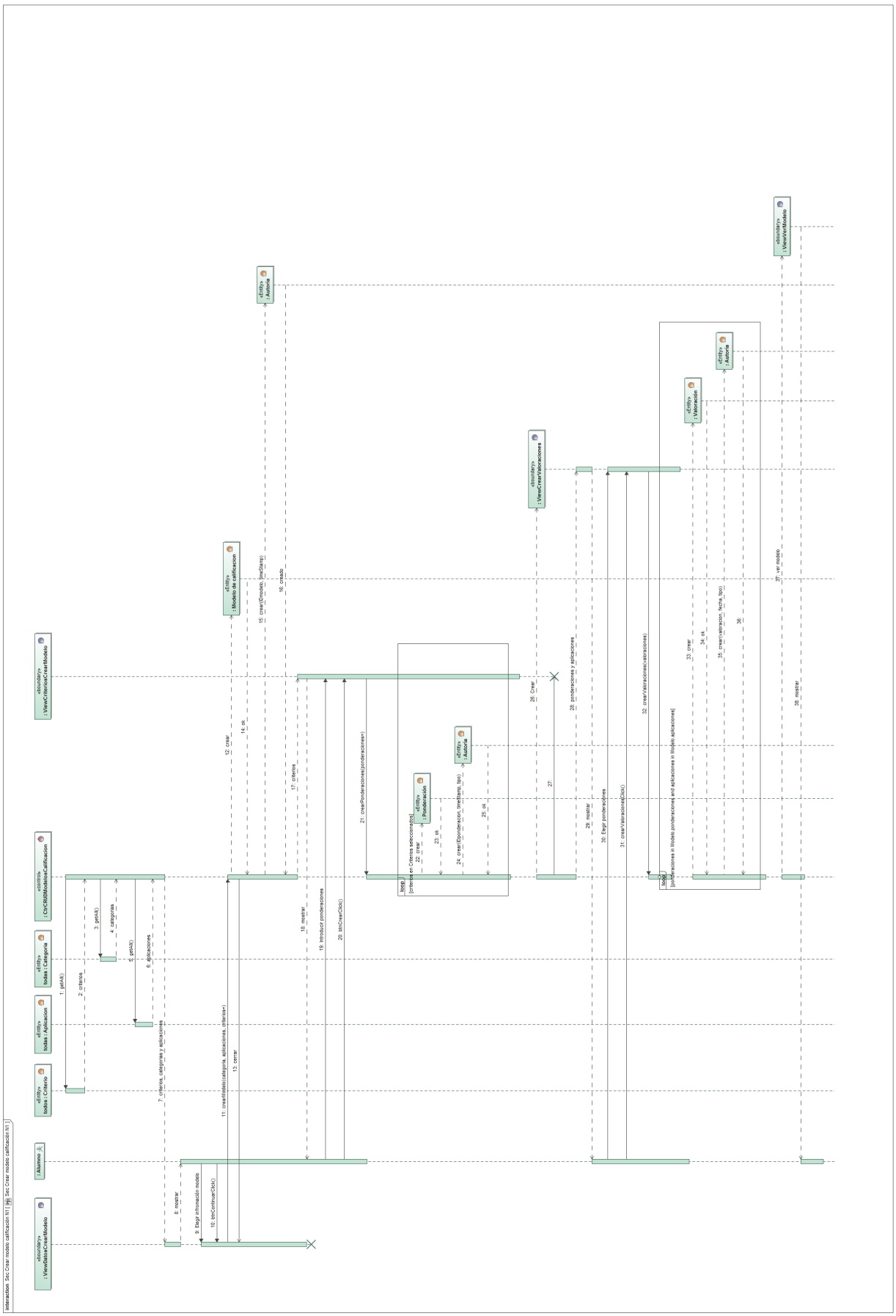


Figura 5.7: Diagrama de secuencia Crear modelo.

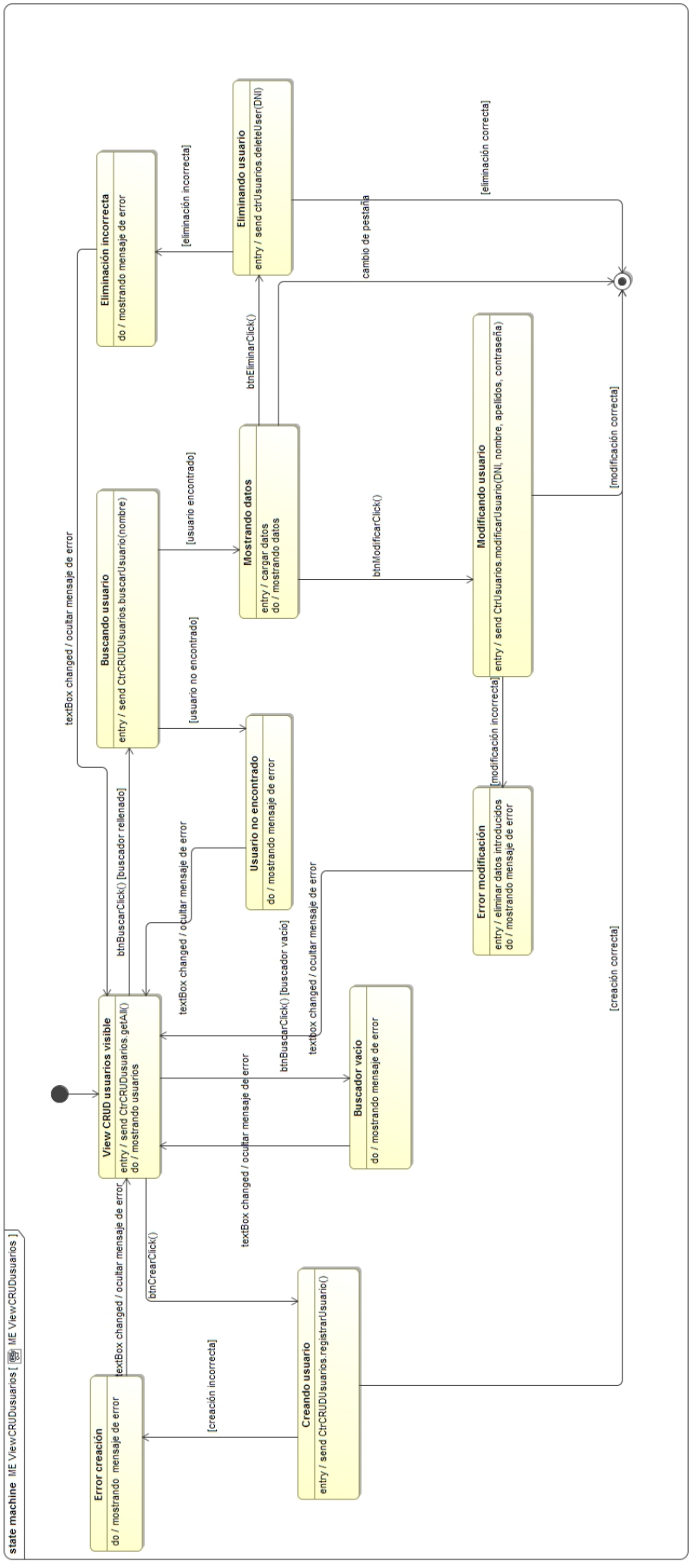


Figura 5.8: Diagrama de máquinas de estado.

Capítulo 6

Desarrollo

En este capítulo se describe cómo se lleva a cabo la implementación del sistema que ha sido definido en los puntos anteriores. Tal y como se ha explicado con anterioridad, el gestor de evaluaciones es desarrollado principalmente usando la tecnología Java EE 7 y ciertos frameworks adicionales que le aportan un valor añadido a la aplicación.

El sistema se desarrolla usando un servidor de aplicaciones Glassfish 4.1, que nos permite desplegar aplicaciones desarrolladas en Java EE. Por otra parte, el SGBD utilizado es un Apache Derby, elegido por estar basado en java, JDBC (Java DataBase Connectivity, en español, Conectividad a Base de Datos Java) y estar basado en estándares SQL. Por último, el proveedor de persistencia para la gestión de la BD subyacente es EclipseLink, puesto que para la base de datos elegida nos ofrece un alto rendimiento (principalmente en rapidez en las transacciones) comparado con otros proveedores de persistencia.

La implementación realizada se puede dividir siguiendo un patrón similar de desarrollo acorde a la arquitectura del sistema. Cabe destacar que en la siguiente descripción solo se muestran vistas o POJOs (Plain Old Java Object) representativos para la aplicación, siendo imposible mostrar todo el código.

6.1. Capa web

En este apartado se describe el desarrollo de la parte web de la aplicación, la cual incluye el aspecto visual (vistas) y los controladores de la aplicación que conectan las vistas con la capa de negocio.

6.1.1. Vistas JSF

A continuación se describe cómo se ha llevado a cabo la parte visual de la aplicación. Se ha utilizado JSF 2.2 (Java Server Faces en su versión 2, la más actual), un framework que permite el despliegue de páginas web en aplicaciones Java EE. Proporciona componentes para la interfaz de usuario y administrar su estado y beans administrados (los backing bean que se describen en el apartado siguiente).

Para una simplificación y reutilización de código, así como para la distribución del contenido en la página, se ha hecho uso de plantillas, un total de cuatro atendiendo al perfil del usuario que está en el sistema. Estas plantillas son:

- Plantilla de login: en ella se describe una cabecera y el contenido principal de la página. Los clientes de estas plantillas han sido las páginas de login (validación del usuario) y registro de un nuevo usuario.
- Plantilla de alumno: es la utilizada para aquella funcionalidad común a todos los usuarios (alumnos y grupos de alumnos, profesores y administradores). Esta se divide en cabecera, contenido y pie de página. La cabecera contiene la barra de navegación del usuario que se encuentre navegando por la aplicación. El pie de página contiene el nombre del sistema y su asignatura. Y finalmente, el contenido se asocia a la funcionalidad descrita en el apartado 5.3.1 de esta memoria.
- Plantilla de profesor: contiene la misma estructura que la plantilla de alumno, con la salvedad de que los alumnos (o grupos de alumnos) no pueden acceder a los contenidos de esta página. En caso de que se intentara acceder a ese contenido sin ser profesor o administrador, el contenido de la página no se renderiza (no se muestra) y es redireccionado a la página de autenticación.

- Plantilla de administrador: idem que la plantilla de profesor, con la diferencia de que solo los administradores pueden acceder a dicha funcionalidad establecida en el contenido.

Los clientes de las plantillas muestran la distinta funcionalidad del sistema. Por ejemplo, en el listado 6.1 se muestra el código de la vista ver_aplicacion.xhtml:

Listing 6.1: Fragmento de la vista ver_aplicacion.xhtml

```
<ui:composition template="./plantilla_alumno.xhtml">
    <ui:define name="title">Ver aplicación</ui:define>
    <ui:define name="content">
        <div class="container">
            <h2>Aplicación #{aplicacionCtr.nombre}</h2><br />
            <h:messages style="color:red"></h:messages>
            <h:form id="form_ver" class="form-horizontal">
                <h:inputHidden value="#{aplicacionCtr.id}" id="idAplicacion" />
                <div class="row">
                    <div class="col-md-5 col-lg-5 col-sm-6 col-xs-12">
                        <div class="form-group">
                            <div class="col-md-2 col-lg-2 col-sm-4 col-xs-12"><label>
                                ↪Nombre</label></div>
                            <div class="col-md-10 col-lg-10 col-sm-8 col-xs-12"><h:
                                ↪inputText class="form-control"  value="#{aplicacionCtr.
                                ↪nombre}"/></div>
                        </div>
                        <div class="form-group">
                            <h:outputLabel value="Descripción"/>
                            <h:inputTextarea rows="5"  class="form-control"  value="#{
                                ↪aplicacionCtr.descripcion}"/>
                        </div>
                      <!--Más vista-->
                </div>
            </h:form>
        </div>
    </ui:define>
</ui:composition>
```

La etiqueta composition de la librería facelets (ui) indica la plantilla desde la que se está editando el contenido y las etiquetas define indican los contenidos a ser introducidos en la página una vez el contenidos de las etiquetas Java EE sea renderizado a etiquetas XHTML.

Algunas de las etiquetas de Java EE se marcan en negro, correspondiendose con componentes Java EE que ayudan a la gestión de la aplicación y que posteriormente se renderizarán en otros XHTML. En algunos componentes, se hace uso de expresiones EL, aquellas de la forma #{something} dónde something re-

ferencia a elementos de los backing bean, ya sean atributos o métodos. Además, en las etiquetas div, dentro del atributo class se encuentran las especificaciones del CSS proporcionado por Bootstrap que cambia la apariencia de la vista.

Las etiquetas de la librería core (c) nos permiten, en este caso, elegir si un contenido va a ser renderizando de acuerdo a la condición que se le especifica en el atributo test.

Existen más librerías, que en el conjunto del proyecto han podido usarse, del mismo modo que otros componentes Java EE, funciones JavaScript, y demás.

6.1.2. Backing beans

Los backing bean forman junto con las interfaces la capa web de la aplicación. Un backing bean es un POJO anotado que se encarga de realizar la conexión entre la vista y el resto de la aplicación.

Un POJO es una clase Java simple, como su propio nombre indica. Estos no deben extender ni implementar nada en especial. Adicionalmente, los backing bean no deben ser abstract ni final, deben tener un constructor sin argumentos e implementar los getters y setters de los atributos (a excepción de otros beans inyectados en él).

Son los controladores del sistema, puesto que se encargan de proporcionar los datos a mostrar en la vista, capturan los datos de entrada de los usuarios, responden a los eventos de los usuarios y por último, pero no menos importante, determinan la navegación entre las distintas páginas.

Para el sistema, se ha implementado un bean por cada entidad a gestionar, de forma que un bean se encarga de toda la gestión de una entidad (a excepción del bean UsuarioCtr que gestiona los alumnos, profesores y administradores) y el bean LoginController que se encarga de los datos de autenticación.

A modo de ejemplo, en el siguiente listado se muestra el código perteneciente al bean AplicacionCtr, cuya función es la creación, lectura, modificación y eliminación de información relativa o relacionada con la entidad Aplicación de nuestro sistema.

Listing 6.2: Fragmento de la clase AplicacionCtr

```
@Named(value = "aplicacionCtr")
@RequestScoped
public class AplicacionCtr {

    @EJB AplicacionesEJBLocal negocio;
```

```
    @EJB CategoriasEJBLocal negocioCats;
    @Inject AutoriaCtr autorias;
    @Inject LoginController login;

    private String nombre, descripcion, casosExito, tipo;
    //Más atributos

    public AplicacionCtr() {
    }

    public String crearAplicacion(){
         //Implementación
    }

    public String verAplicacion(Long id){
         //Implementación
    }

    public String modificarAplicacion(){
        //Implementación
    }

    public String eliminarAplicacion(){
         //Implementación
    }

//Métodos auxiliares, getters y setters
```

Un backing bean debe tener siempre dos anotaciones, aquella que le indica la inyección del contexto y la que le indica el ámbito, el tiempo de vida del bean.La primera notación es *@Named*, una anotación de CDI usada para la inyección de contextos y dependencias, así como para poder resolver las expresiones EL dentro de la aplicación. Opcionalmente, se le puede asignar el nombre con el que van a ser llamadas desde estas expresiones.

La anotación de ámbito usadas en este proyecto son *@RequestScoped* (como la del listado anterior) o *@SessionScoped*. Por una parte, RequestScoped indica que el bean existirá durante una petición HTTP, tomará los datos de la vista actual, los procesará y cargará la página siguiente con los datos necesarios. Por otra parte, SessionScoped permite al bean existir durante una sesión de usuario, manteniendo los datos durante esta. Todos los backing bean han sido implementados con ámbito de petición, excepto LoginController, ModeloCtr y AsignaturaCtr que han sido implementadas con ámbito de sesión para permitir más fácilmente el paso de información entre distintas vistas.

La inyección de otros beans se realiza con las anotaciones *@EJB* para los EJB y *@Inject* para backing beans. La navegación entre páginas se realiza a

través de las llamadas a los métodos del backing bean (denominada navegación explícita), devolviendo estos el nombre de la siguiente vista a cargar.

Con esta descripción queda completo el desarrollo de la capa web de nuestro sistema, que se encuentra implementada en el paquete war del código fuente.

6.2. Capa de negocio

En este nivel se definen los EJB, cuya función consiste en conectar la capa web con la capa de base de datos. Los EJB que se emplean en este proyecto son session beans (POJO de Java), los que se inyectan en los backing beans e implementa la lógica de negocio.

Los EJB, al igual que los backing beans, han sido desarrollado orientado a cada entidad que se almacena en la BD. Cada bean hace de nexo entre la aplicación y la BD para una sola entidad, a excepción del EJB de los usuarios, que controla las entidades de Alumno, Profesor y Administrador. Por ejemplo, el EJB AplicacionesEJB gestiona exclusivamente el acceso a BD de la entidad Aplicación.

Antes de definir los EJB se definen las interfaces que estos van a implementar. Las interfaces son interfaces locales (anotadas con *@Local*), puesto que el cliente y el EJB se encuentran en la misma JVM (Java Virtual Machine, en inglés Máquina Virtual Java). Un ejemplo de interfaz local se muestra en el listado 6.3.

Listing 6.3: Fragmento de la interfaz AplicacionesEJBLocal

```
@Local
public interface AplicacionesEJBLocal {

    public Aplicacion findByID(Long id);
    public Aplicacion findByNombre(String nombre);
    public Objeto findObjetoByNombre(String nombre);
    public List<Aplicacion> findAll();
    public void crearAplicacion(Aplicacion app);
    public void modificarAplicacion(Aplicacion app);
    public void eliminarAplicacion(Aplicacion app);
}
```

Una vez se ha definido la interfaz, se crea el EJB que implementa dicha interfaz. A los EJB se le añade la notación *@Stateless*, que indica que el bean va

a ser normalmente usado para hacer operaciones independientes. Se le inyecta el contexto de persistencia con la anotación *@PersistenceContext* para poder realizar la comunicación con la BD para operar con sus operaciones básicas: creación, lectura, actualización y eliminación.

Un ejemplo de EJB es el siguiente, el cual implementa la interfaz mostrada más arriba.

Listing 6.4: Fragmento de la clase AplicacionesEJB

```
@Stateless
public class AplicacionesEJB implements AplicacionesEJBLocal {

    @PersistenceContext
    EntityManager em;

    public Aplicacion findByID(Long id){
        return em.find(Aplicacion.class, id);
    }

    public Aplicacion findByNombre(String nombre) {
        //Implementación
    }

    public List<Aplicacion> findAll() {
        TypedQuery<Aplicacion> query = em.createQuery("SELECT c FROM Aplicacion c",
            ↪Aplicacion.class);
        List<Aplicacion> apps = new ArrayList<>();
        try{
            apps = query.getResultList();
        }catch(NoResultException e){}
        return apps;
    }

    public void crearAplicacion(Aplicacion app) {
        em.persist(app);
    }

    public void modificarAplicacion(Aplicacion app) {
        em.merge(app);
        //Implementación para las categorías
    }

    public void eliminarAplicacion(Aplicacion app) {
        app = em.merge(app);
        em.remove(app);
    }
}
```

Se pueden realizar operaciones básicas con el contexto de persistencia, y además, operaciones más complejas sirviéndose del lenguaje JPQL.

6.3. Capa de base de datos

Este nivel es el más bajo de la aplicación, donde se ha realizado la implementación del modelo de la base de datos a partir del framework JPA, el cual nos permite simplificar el manejo de los datos de la BD a través de clases Java.

Las entidades son POJOs anotados que especifican cómo se va a mapear la clase Java en una entidad de la BD. En el listado 6.5 se muestra la clase Aplicación.

Listing 6.5: Fragmento de la clase AplicacionesEJB

```
@Entity
@Access(AccessType.FIELD)
@DiscriminatorValue("APLICACION")
public class Aplicacion extends Objeto implements Serializable {

    private static final long serialVersionUID = 1L;

    private String descripcion, tipo, casosExito;
    @ManyToMany(fetch = FetchType.LAZY, mappedBy="appsAsociadas") //Relacion con
        ↪categoria
    @JoinColumn(nullable = false)
    private List<Categoria> categoriasAsociadas;
    @ManyToMany(fetch = FetchType.LAZY, cascade = CascadeType.ALL) //Relacion con modelo
        ↪ de calificacion
    @JoinTable(name="APLICACION_MODELO", joinColumns=@JoinColumn(name="aplicacion_fk"),
            inverseJoinColumns = @JoinColumn(name="modelo_fk"))
    private List<ModeloCalificacion> modelosAsociados;
    @OneToMany(mappedBy = "aplicacion") //Relacion con valoracion
    private List<Valoracion> valoraciones;


    public Aplicacion(){
        categoriasAsociadas = new ArrayList<>();
        modelosAsociados = new ArrayList<>();
    }

    public Aplicacion(String nombre, String descripcion, String tipo, String casosExito)
        ↪{
        super(nombre);
        this.descripcion = descripcion;
        this.tipo = tipo;
        this.casosExito = casosExito;
        categoriasAsociadas = new ArrayList<>();
        modelosAsociados = new ArrayList<>();
    }
//Getters y setters
}
```

La notación *@Entity* denota que la clase se va a mapear a una entidad de

la BD, la anotación *@AccessType* indica que el acceso a los atributos se va a realizar por los campos, *@DiscriminatorValue* establece el valor de discriminación para Aplicación, un valor necesario ser establecido puesto que extiende de la clase Objeto, cuya estrategia de herencia requiere de establecer un valor discriminante. La clase Usuario también es una clase padre de las clases Alumno, Profesor, Administrador y Grupo de Alumnos.

Para mapear las relaciones entre distintas entidades se utilizan las anotaciones *@ManyToMany, @OneToMany, @ManyToOne* y *@OneToOne*, cada una para las relaciones muchos-a-muchos, uno-a-muchos, muchos-a-uno y uno-a-uno, respectivamente. En caso de ser bidireccionales, las relaciones pueden ser o bien manejadas por una entidad o por otra. Si la entidad no maneja la relación, se indica con el atributo mappedBy.

En el caso de las relaciones ManyToMany, se crea una tabla adicional donde se genera la relación, debiendo establecer así los campos de esta nueva tabla, junto con la clave foránea a la que referencia (anotación *@JoinTable*).

Con esta explicación queda completo el desarrollo de la capa de negocio y de base de datos de nuestro sistema, que se encuentra implementada en el paquete ejb del código fuente.

Con la descripción e implementación de las interfaces, los backing beans, los EJBs, las entidades JPA para toda la funcionalidad especificada en el apartado de requisitos se completa el desarrollo de este proyecto.

Capítulo 7

Pruebas

En este capítulo se detalla la fase de pruebas que le han sido realizadas al gestor de evaluaciones. Para comenzar, las pruebas se han llevado a cabo en el IDE Selenium (a través de un plugin para el navegador Mozilla Firefox), donde han sido realizadas pruebas de integración y de sistemas.

Las pruebas en Selenium se basan en torno a los asertos, un concepto elaborado para verificar si un resultado es el esperado y, en caso de que no lo fuera, detener la ejecución para un mayor control de los errores producidos.

Luego, se han elaborado baterías de pruebas sobre las operaciones básicas que se pueden realizar sobre la información gestionada, por ello se han ejecutado pruebas sobre la creación, lectura, modificación y eliminación de las entidades del sistema. Cabe destacar que las pruebas con Selenium no se han elaborado sobre el 100 % de los caminos no cíclicos sobre la navegabilidad de la aplicación, puesto que supondría un esfuerzo que sobrepasaría el tiempo establecido y recomendado en el que se encaja esta tarea dentro del proyecto. Por ello, han quedado fuera de las pruebas aquellas que resultan muy similares a otras, como por ejemplo la creación, modificación o eliminación de criterios, valoraciones o aplicaciones, operaciones muy similares a la creación, modificación o eliminación de categorías y ponderaciones.

Para las pruebas de creación, se han introducido los datos de la entidad, y posteriormente se ha comprobado que esa entidad esté almacenada en la base de datos con los datos modificados. Para la modificación se comprueba que los datos de la entidad a modificar han sido efectivamente modificados en

la BD. Finalmente, para la eliminación, se elimina la entidad en cuestión y a continuación se comprueba que la entidad deja de estar almacenada.

Se procede de la misma forma para los casos alternativos en los que se produciría una inconsistencia de datos y muestra mensajes de error no realizando así la operación básica prevista.

Además de las pruebas con el IDE Selenium, numerosas pruebas unitarias de caja blanca se fueron realizando conforme el desarrollo del sistema para comprobar su buen funcionamiento y si esa unidad funcional está ejecutando la funcionalidad de ella esperada. Una vez comprobado el correcto funcionamiento de las unidades mínimas funcionales, se fueron realizando pruebas de integración entre ellas, permitiendo así comprobar que la comunicación entre ellas estaba establecida y funcionando correctamente.

Una vez se han realizado las pruebas de caja blanca, se procede a ejecutar las pruebas de caja negra (pruebas de sistema), cuya finalidad es valorar si el sistema desarrollado cumple los requisitos funcionales (qué debe hacer) y requisitos no funcionales (cómo debe hacerlo) elaborados para dicho sistema, con la diferencia de que en este tipo de pruebas ya no se accede al código para comprobarlo, sino simplemente si cumple con lo requerido para esta aplicación.

Por consiguiente, con este conjunto de pruebas se comprueba que el sistema funciona correctamente, tanto en los casos de éxito como en los casos alternativos. Estas baterías de pruebas, junto con el buen funcionamiento observable en su utilización, nos permite comprobar que el sistema no posee problemas y/o fallos graves fácilmente detectables y posibles focos de ataques informáticos.

Conclusiones y líneas futuras

Con la realización de este proyecto se ha conseguido implementar un sistema gestor de evaluaciones de sistemas de información con una arquitectura web, accesible a través de un navegador. Esta herramienta se centra en la gestión de modelos de calificación de la que pueden hacer uso tanto alumnos y grupos de alumnos como profesores. De manera adicional, se puede gestionar también información relacionada con los modelos de calificación que dependiendo del perfil se tienen diferentes permisos sobre ellas.

Este desarrollo ha permitido el afianzamiento y puesta en práctica de conocimientos adquiridos a lo largo de todo el estudio universitario. Destaca comprobar que, verdaderamente, es necesario el uso de una metodología y una gestión adecuada al tipo de proyecto a realizar para poder ejecutar el proyecto de manera eficiente, ya que en otro caso el proyecto habría acabado muy probablemente fuera del tiempo establecido y con una cantidad de errores considerable.

Asimismo, el proyecto ha permitido la familiarización con tecnologías web desconocidas anteriormente, como puede ser el uso de Bootstrap como framework para el CSS o la inclusión de algunos detalles con AJAX (Asynchronous JavaScript And XML) para ciertas peticiones asíncronas para la aplicación. Junto con AJAX, el framework PrimeFaces incluye utilidades formidables para su uso en aplicaciones web. Finalmente, se decidió no incluirlo en este proyectos puesto que en una etapa avanzada del proyecto requeriría refactorización y tiempo, lo cual no compensaría el tiempo invertido con las utilidades que aportaría.

Por último, cabe destacar la mejora en la capacidad de reacción ante determinados problemas que han ido surgiendo a lo largo del desarrollo y el poder resolverlos autónomamente, que me han permitido una evolución como profesional.

Debido a la limitación de tiempo que se propone para el desarrollo de este

trabajo, cierta funcionalidad ha quedado fuera del alcance del sistema por añadir complejidad y no poder ejecutarla dentro de la exigencia temporal.

Una ampliación que pueden hacerse a este proyecto es la inclusión de filtros y un sistema de búsqueda para algunas entidades que hagan más sencilla su gestión, como puede ser para categorías, aplicaciones, criterios, etc. resultando más cómoda la búsqueda y lectura de los datos. Cabe decir que no llegó a ejecutarse puesto que había otra funcionalidad con una prioridad más alta.

Otro punto interesante sería permitir que, manualmente, un profesor pueda elegir si una información tiene visibilidad para alumnos (y grupos de alumnos) o no, de manera que las fusiones de modelos, por ejemplo, pudieran ser o no visibles.

Se podría tener en cuenta el acceso a información en "modo anónimo", esto es, podría verse información que estuviera catalogada como tal sin necesidad de autenticarse en la aplicación.

Por último, se consideraría la asociación de información que crean los alumnos con la asignatura en la que se encuentran actualmente navegando. Si no se elije ninguna asignatura, no debe darse acceso a la información.

Bibliografía

[1] https://docs.oracle.com/javaee/7/tutorial/.

[2] Antonio Goncalves. *Beginning Java EE 7*. Apress, 2013.

[3] Eric Jendrock y Ricardo Cervera-Navarro. *The Java EE 7 Tutorial*. Oracle, 2014.

[4] David Burns. *Selenium 2 Testing tools beginner's guide*. Packt Publising, 2012.

[5] Marijn Haverbeke. *Eloquent JavaScript: A Modern Introduction to Programing*. No Starch Press, 2011.

[6] Ryan Flores. *Getting started with Bootstrap 3.3*. Code Playground, 2015.

[7] http://getbootstrap.com/.

[8] https://netbeans.org/.

[9] http://es.slideshare.net/cptanalatriste/arquitectura-y-diseo-de-aplicaciones-java-ee.la.

[10] Steven M. Schafer. *HTML, XHTML and CSS: HTML, XHTML, and CSS bible*. Indianapolis, 2010.

Apéndice A

Manual de usuario

En esta sección se presenta la guía de la aplicación del gestor de evaluaciones de sistemas de información y toda la información concerniente a la gestión de los modelos de calificación de los usuarios de la herramienta.

Esta aplicación web está pensada para facilitar la familiariación de los alumnos con los modelos de calificación, el uso de esta herramienta para valorar sistemas de información y una plataforma para profesores para poder corregir y mejorar los modelos de calificación elaborados por los alumnos.

Esta herramienta ofrece una serie de funcionalidades que permiten la elaboración, modificación y corrección de los modelos de calificación a los alumnos. Para ello lo primero que deben hacer el autenticarse en el sistema (figura A.1).

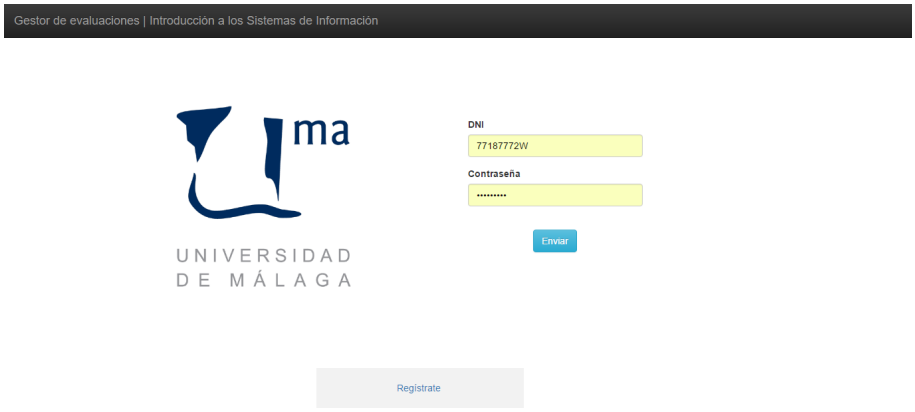


Figura A.1: Autenticación.

Una vez autenticado, el alumno puede elegir la asignatura en la que quiere trabajar y en la que se van a registrar los cambios de información.

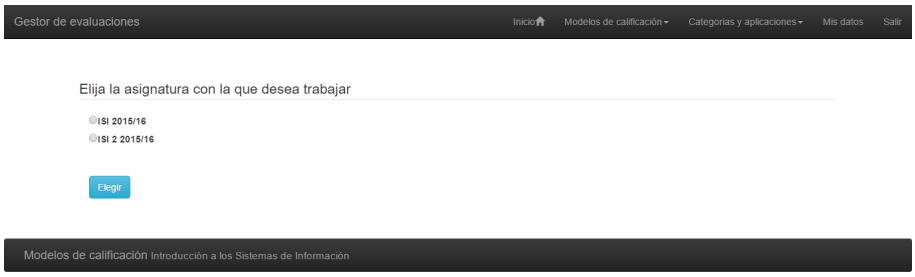


Figura A.2: Inicio.

Puede acceder y editar información sobre las categorías y aplicaciones que se encuentran en el sistema, así como crear nuevas aplicaciones y categorías, relacionarlas entre ellas dentro de la sección de Categorías y aplicaciones (figuras A.3, A.4 y A.5).

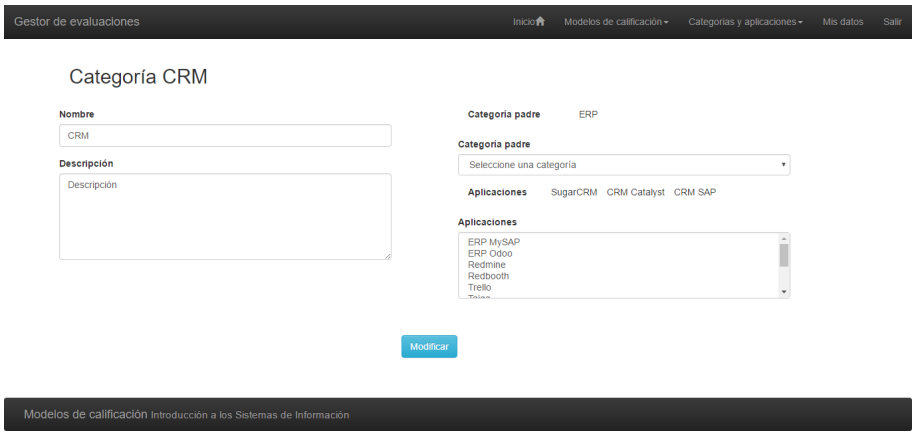


Figura A.3: Ejemplo de categoría.

Dentro de la sección de modelos de calificación, se puede acceder al banco de criterios disponibles para enlazarlos a los modelos, a las ponderaciones y valoraciones que cada usuario tenga en sus modelos. Para crear un modelo nuevo,

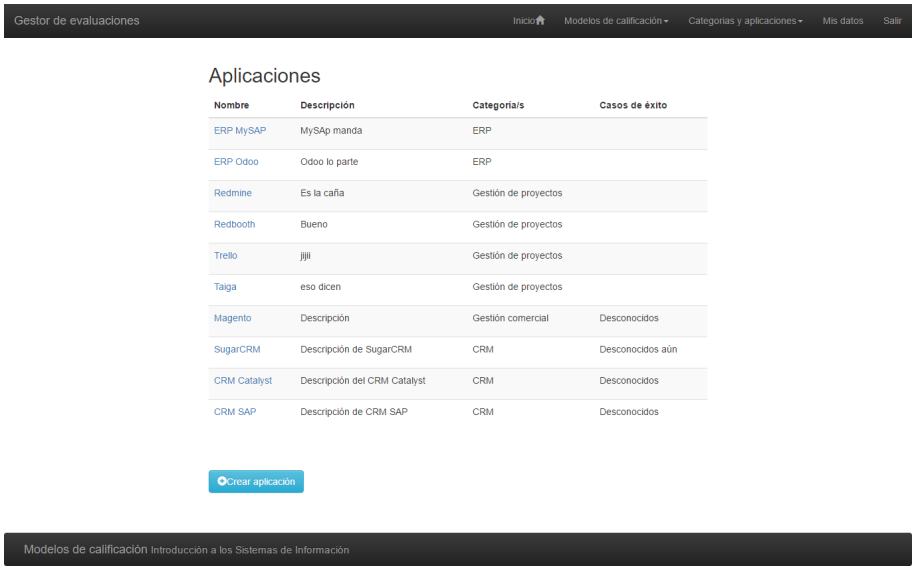


Figura A.4: Listado de aplicaciones.

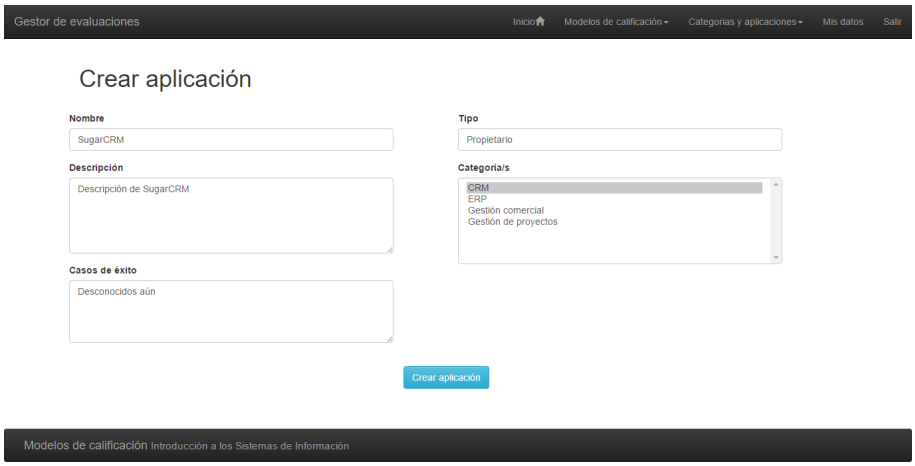


Figura A.5: Creación de una aplicación.

primero se deben introducir sus datos básicos en el sistema (nombre, descripción, categoría, etc.) como en la figura A.6. Después de eligen las ponderaciones para cada criterio seleccionado (figura A.7) y por último se eligen las valoraciones para cada criterio y aplicación del modelo (figura (A.8).

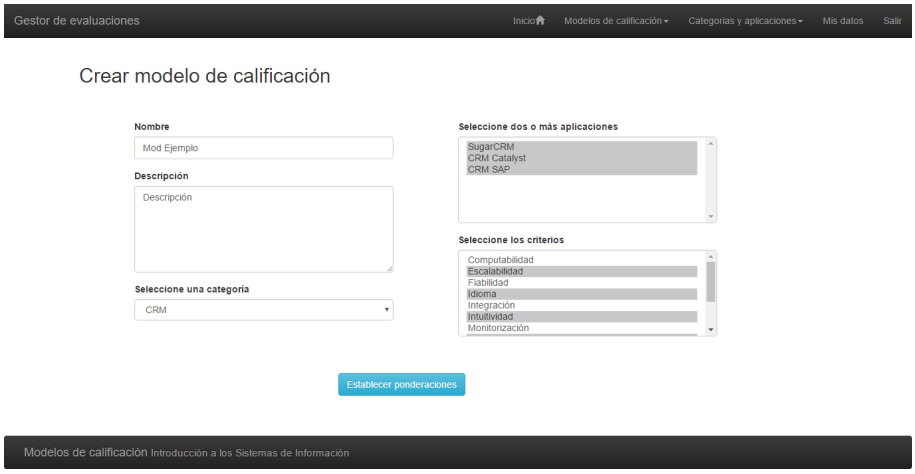


Figura A.6: Datos para crear un modelo.

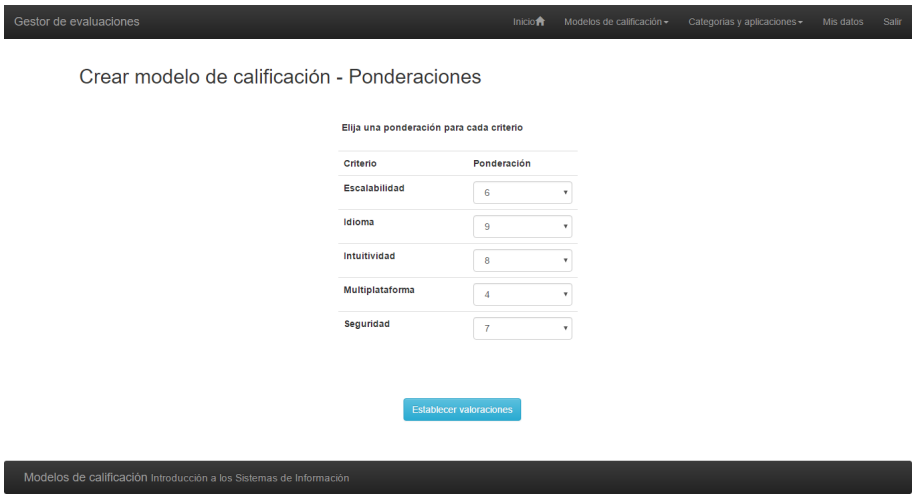


Figura A.7: Ponderaciones para crear un modelo.

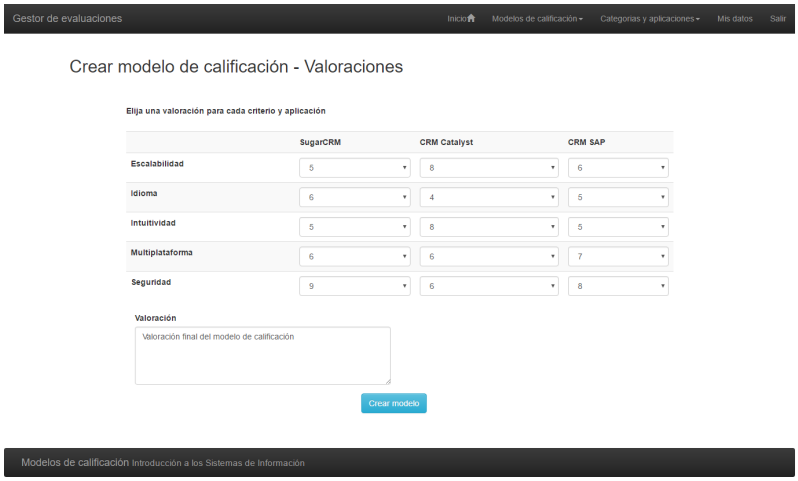


Figura A.8: Valoraciones para crear un modelo.

Los modelos de calificación elaborados por el usuario se pueden consultar en la misma sección, y a partir de este listado de modelos elaborados se puede acceder a la información propia de cada modelo de calificación (figura A.10).

Figura A.9: Listado de modelos.

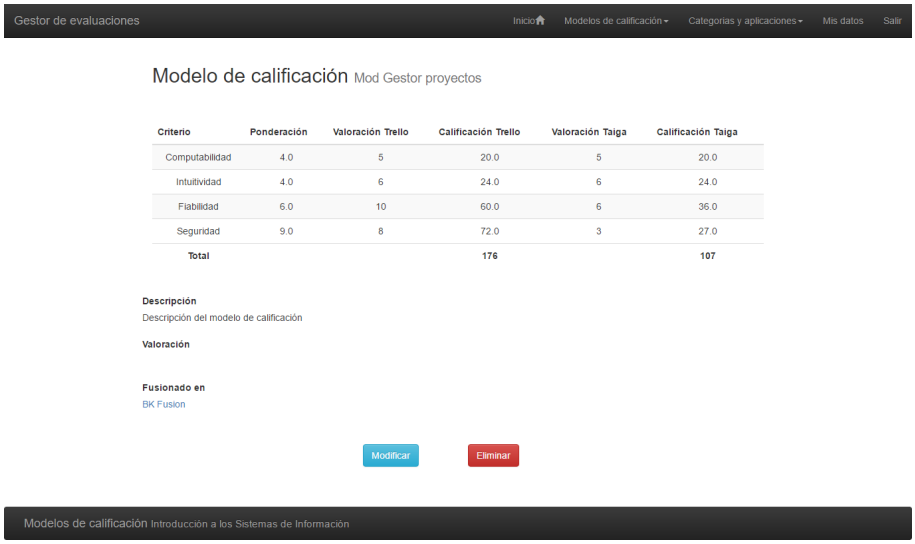


Figura A.10: Visualización de un modelo.

Por último, un alumno puede gestionar sus datos personales básicos, tal y como se muestra en la figura A.11.

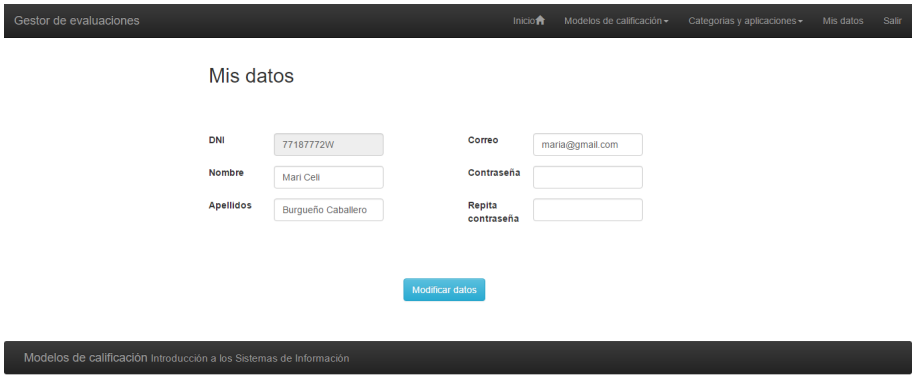
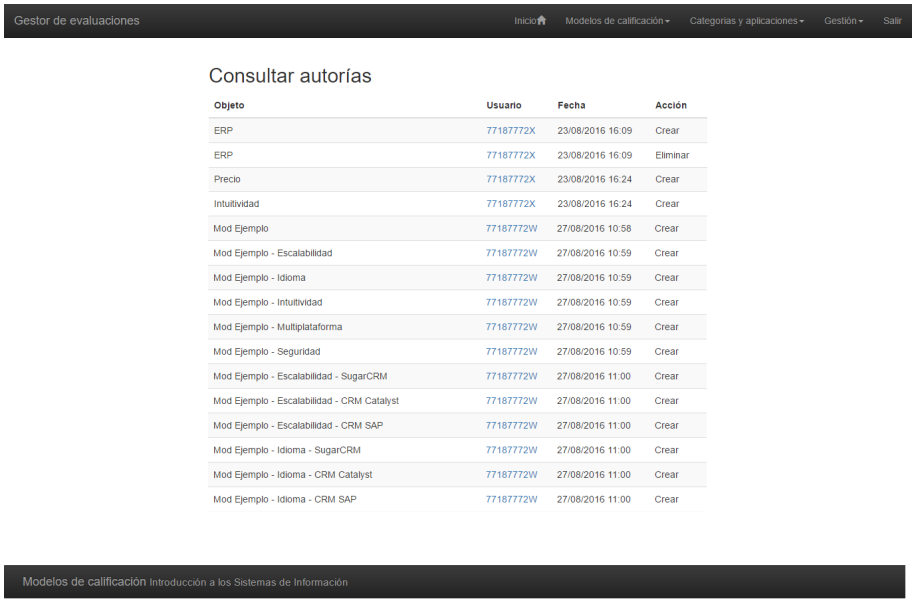
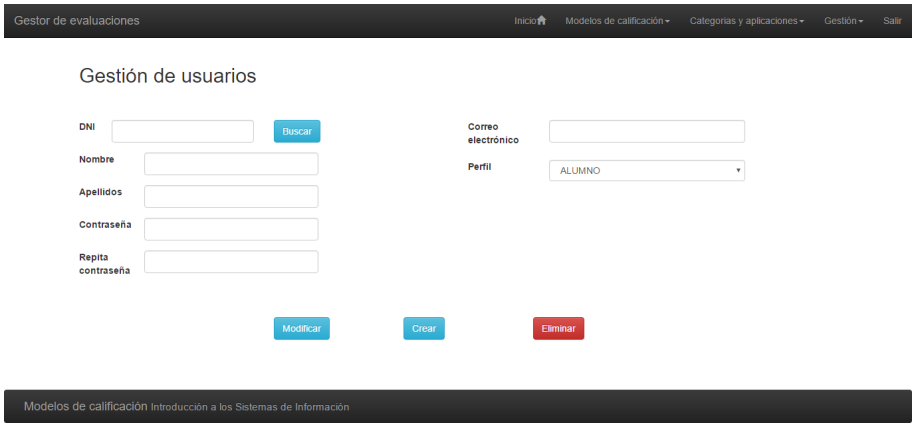


Figura A.11: Gestionar datos personales.

De manera adicional a la funcionalidad del alumno, el profesor también puede consultar las autorías del sistema, es decir, puede ver qué acción se ha realizado, cuando, y por quién.



También puede gestionar la información de los usuarios que se encuentran registrados en el sistema, tanto alumnos como profesores (pero no administradores, puesto que solo los administradores pueden manejar información sobre otros administradores).



Para la corrección de modelos, un profesor puede puntuar un modelo de forma que le asigna una puntuación numérica y de manera opcional puede dejarle un comentario para que el autor lo vea (figura A.12).

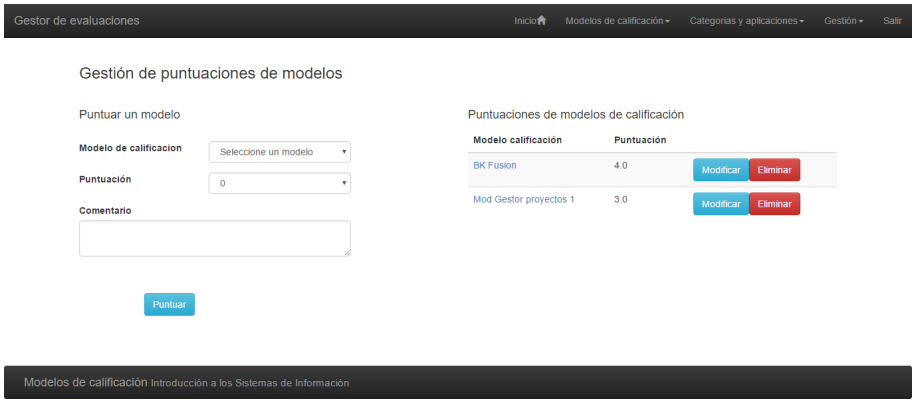


Figura A.12: Gestión de puntuaciones.

Para terminar, el administrador, además de toda la funcionalidad que recoge el profesor, gestiona las asignaturas del sistema. Puede crear, leer, modificar y eliminar asignaturas, y también matricular alumnos y grupos de alumnos a las asignaturas y asignar profesores que las impartan. En la figura A.13 se muestra la información perteneciente a la asignatura llamada *ISI*.

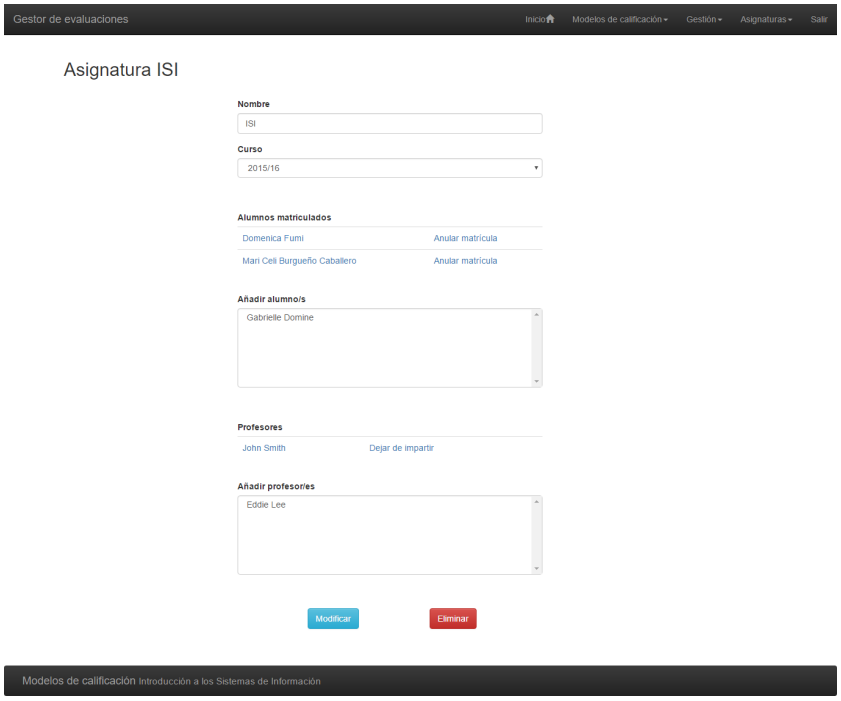


Figura A.13: Ver asignatura.