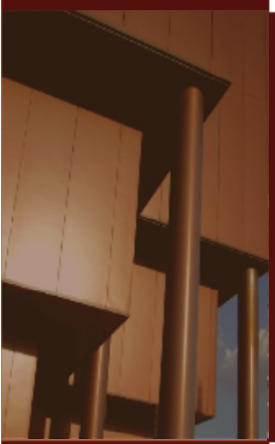
Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Desarrollo de una aplicación web para gestión y puntuación de trabajos.

Autor: Salvador Guzmán Osuna
Tutores: Antonio Jesús Sierra Collado
Juan Antonio Ternero Muñiz



Dep. Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2019

Trabajo Fin de Grado
Grado en Ingeniería de Telecomunicación

Desarrollo de una aplicación web para gestión y puntuación de trabajos.

Autor:
Salvador Guzmán Osuna

Tutores:
Antonio Jesús Sierra Collado
Juan Antonio Ternero Muñiz

Dep. Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2019

Trabajo Fin de Grado: Desarrollo de una aplicación web para gestión y puntuación de trabajos.

Autor: Salvador Guzmán Osuna

Tutores: Antonio Jesús Sierra Collado
Juan Antonio Ternero Muñiz

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2019

El Secretario del Tribunal

Febrero de 2019

Trabajo Fin de Grado (TFG) de Salvador Guzmán Osuna.

Grado En Ingeniería De Las Tecnologías De Telecomunicación.

Escuela Técnica Superior de Ingeniería (ETSI) -- Universidad de Sevilla (US).

Índice

Índice de Figuras

Resumen

En unas de las asignaturas impartidas por el Depto. de Ingeniería Telemática, se utiliza un procedimiento de evaluación consistente en la cumplimentación por parte del alumnado de formularios impresos que sirven para que los propios estudiantes realicen calificaciones del trabajo académico realizado.

Para esta tarea el profesorado dedica una parte significativa de tiempo de clase en aplicar el referido sistema de evaluación. La gestión y tramitación de esta documentación por parte del profesor, requiere de un largo proceso, que finaliza subiéndola a una plataforma web oficial.

En anteriores cursos, se ha llevado a cabo, también, el desarrollo "parcial" a modo de trabajo, de diferentes aplicaciones web basadas en los lenguajes con los que se ha llevado a cabo este proyecto.

De esta situación surge la idea del desarrollo de una aplicación web que permita al profesor poder gestionar las puntuaciones de los diferentes cursos que tiene una asignatura. Todo esto será accesible desde cualquier dispositivo que tenga acceso a internet, ya que será una página web.

El lenguaje utilizado para el desarrollo de la aplicación web será HTML y Java, haciendo uso de la tecnología de los Servlets para el control de la aplicación y de las páginas jsp para mostrar al cliente los datos. También se utilizará una base de datos SQL para el almacenamiento de la información y CSS para la maquetación.

Como herramientas utilizadas para la realización de la aplicación, ha sido utilizado el IDE Eclipse para el desarrollo de la aplicación, PostgreSQL para la gestión y manejo de la Base de Datos y los distintos navegadores web para realizar pruebas y depuraciones de código, entre otras.

Centrándonos en el contenido de esta memoria, además de lo que ha sido citado anteriormente, en primer lugar se realiza una introducción en la que se expondrá la justificación del proyecto así como la metodología. Después, se hace una explicación de los posibles objetivos planteados en este proyecto, y posteriormente se realiza un análisis teórico de las necesidades que debe cumplir la aplicación una vez se encuentre desarrollada. Una vez identificado el problema, se expone la solución propuesta.

En el siguiente apartado, se define de forma más técnica las acciones y decisiones a tomar sobre cómo se ha de desarrollar la solución del problema expuesto en el capítulo anterior. Acto seguido, se detalla cómo queda implementada la aplicación web distinguiendo entre los servicios ofrecidos al profesor así como al alumno.

Para finalizar, se encuentran un capítulo de conclusiones seguido de varios anexos que servirán de ayuda tanto en el despliegue de la aplicación web, instalación, manual de uso así como la incorporación del código utilizado.

Abstract

In one of the subjects taught by the Department of Telematic Engineering, an evaluation procedure is used consisting of the completion by students of printed forms that are used for students themselves to make grades of the academic work done.

For this task, the teaching staff dedicates a significant part of class time in applying the referred evaluation system. The management and processing of this documentation by the teacher requires a long process, which ends up uploading it to an official web platform.

In previous courses, the "partial" development of different web applications based on the languages with which this project has been carried out has also been carried out.

From this situation arises the idea of developing a web application that allows the teacher to manage the scores of the different courses that a subject has. All this will be accessible from any device that has access to the Internet, as it will be a web page.

The language used for the development of the web application will be HTML and Java, making use of Servlet technology to control the application and jsp pages to show the client the data. A SQL database will also be used for information storage and CSS for layout.

As tools used for the realization of the application, the IDE Eclipse has been used for the development of the application, PostgreSQL for the management of the Database and the different web browsers to make tests and debugging of code, among others.

Focusing on the content of this report, in addition to what has been mentioned above, first an introduction is made in which the justification of the project and the methodology will be explained. Afterwards, an explanation of the possible objectives raised in this project is made, and later a theoretical analysis of the needs that the application must fulfill once it is developed is carried out. Once the problem has been identified, the proposed solution is presented.

In the following section, the actions and decisions to be taken on how to develop the solution to the problem described in the previous chapter are defined in a more technical way. Next, it is detailed how the web application is implemented, distinguishing between the services offered to the teacher as well as to the student.

Finally, there is a chapter of conclusions followed by several annexes that will help both in the deployment of the web application, installation, user manual as well as the incorporation of the code used.

Translated with www.DeepL.com/Translator

1 Introducción

1.1. Introducción

Este apartado pretende proporcionar una visión general del presente proyecto. Se analizará el problema que se desea solventar, el motivo por el que se busca esta solución, qué objetivos se busca cumplir y cómo se conseguirán estos objetivos.

En un mundo en el que la digitalización y automatización se encuentra cada vez más presente, surge la necesidad de encontrar una solución que permita renovar el antiguo modelo de evaluación de formularios en papel. Para ese cometido, este proyecto planteará el desarrollo de una serie de herramientas necesarias para la evaluación y gestión de una aplicación web.

Para conseguir este objetivo, se desarrollará una aplicación web que, por un lado, presente al cliente la opción de puntuar a sus compañeros de otros grupos además de poder cambiar la contraseña; y por otro lado, administrar y gestionar la aplicación.

Se ha pensado en realizar una aplicación web que sea accesible por los alumnos y así aprovechar las ventajas de disponibilidad y flexibilidad que proporciona el acceso a la aplicación a través de Internet.

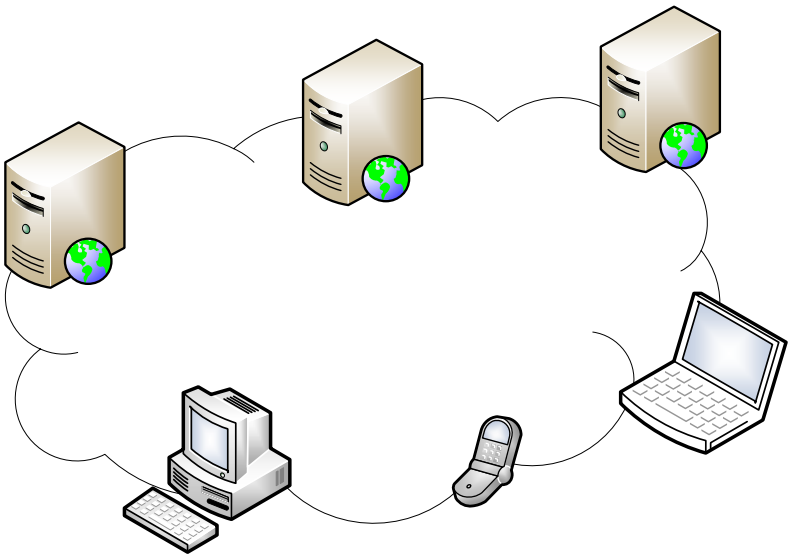


Figura 1. Esquema de arquitectura de aplicación web

1.2. Justificación del proyecto

En un mundo cada vez más conectado, es necesario una renovación del modelo de evaluación para ofrecer una mejor prestación tanto al profesor como al alumno y conseguir así una mejora en la calidad de servicio. De esta manera, se ofrece al alumno la comodidad de poder evaluar desde cualquier lugar.

De esta situación surge la idea del desarrollo de una aplicación web que permita al profesor poder gestionar las puntuaciones de los diferentes cursos que tiene una asignatura. Todo esto a través, de una página web, que será accesible desde cualquier dispositivo.

La decisión de optar por una aplicación web en vez de una aplicación de escritorio es la de aprovechar las ventajas de la web 2.0:

1. Universalidad de la aplicación: Solo se requiere un navegador web para acceder a ella, con independencia de la plataforma o sistema operativo desde el que se acceda
2. Independencia de ubicación: El acceso a la aplicación es través de internet, por lo que la ubicación del alumno y el profesor es independiente.
3. Gestión de los datos centralizada: Los datos de la aplicación se encuentran en una base de datos. De esta manera es más sencillo mantener la integridad de los datos de la aplicación.
4. Gestión dinámica de los recursos de la aplicación: Al tratarse de una aplicación hospedada en un servidor, se puede aumentar y disminuir la capacidad y los recursos requeridos por la aplicación actualizando el servidor según sea necesario.

1.3. Objetivos

El objetivo principal de este proyecto es digitalizar y renovar el modelo de evaluación utilizado actualmente en una asignatura impartida por el Depto. de Ingeniería Telemática, el cual está basado en la entrega en formato papel de unos formularios de evaluación en los cuales los alumnos deben calificar a sus compañeros siguiendo los criterios indicados en el formulario. Luego estos formularios el profesor tendrá que procesarlo individualmente.

Otro objetivo que tendrá que cubrir la aplicación sería el de incorporar las herramientas para la importación de los alumnos así como sus grupos para las diferentes asignaturas.

Para lograr este objetivo se desarrollará una aplicación que estará desplegada en un servidor web. Para el desarrollo de esta aplicación se utilizará lenguaje Java y estará diseñada utilizando el patrón de diseño de Modelo-Vista-Controlador.

Este servidor también tendrá acceso a una base de datos que, en este caso, utilizará como sistema gestor POSTGRESQL.

A parte de la aplicación del servidor, será necesario diseñar cómo representar la información a los alumnos. Para este cometido se utilizará el lenguaje HTML en su última versión (HTML5), junto con JavaScript y JSP para la dinamización de esta y CSS para el estilo y estética de estos datos.

Además de los objetivos principales ya indicados, en este proyecto también se plantearán unos objetivos secundarios que se intentarán cumplir en el desarrollo de la aplicación.

La aplicación deberá ser sencilla de utilizar. Para ello ha de presentar una interfaz sencilla que permita un uso intuitivo de la misma y que no provoque problemas para su uso tanto por los alumnos, como por los profesores.

1.4. Metodología

Para conseguir cumplir los objetivos planteados se han planteado una serie de hitos y tareas que permitirán ir completando paulatinamente el desarrollo de la aplicación y, de esta manera, cumplir el alcance planteado para este proyecto.

En primer lugar, se llevará a cabo el análisis y diseño de los servicios que ha de ofrecer la aplicación. Se trata de un análisis a alto nivel de las características y herramientas de las que dispondrá la aplicación, así de cuál será su modo de actuación:

- Diseño de los servicios a ofrecer por la aplicación.
    - Herramientas para gestionar la contraseña de los alumnos.
    - Herramientas para gestionar las asignaturas.
    - Herramientas para gestionar los grupos.
    - Herramientas para gestionar a los alumnos.
    - Herramientas para gestionar los permisos de calificación.
    - Herramientas para la visualización de las notas.

Una vez se han analizado los servicios y la funcionalidad que ha de presentar la aplicación, se ha de pasar al diseño de la base de datos que albergará toda la información. Este diseño englobará las acciones necesarias para que, al final de esta etapa, se disponga de una base de datos funcional para su uso en la aplicación.

- Diseño de una base de datos para la aplicación.
    - Análisis de la relación entre los datos (Diagrama Entidad-Relación).
    - Diseño de las tablas que compondrán la base de datos (Modelo Relacional).
    - Diseño de las consultas que será necesario realizar por la aplicación.
    - Implementación de la base de datos.

Una vez analizados los requisitos que ha de cumplir la aplicación y de disponer del soporte para los datos de la aplicación, se llega a la fase de desarrollo de la aplicación. Para ello se implementarán la lógica necesaria para prestar los servicios proyectados, así como se realizarán las pruebas necesarias para asegurar el correcto funcionamiento.

- Desarrollo de la aplicación.
    - Desarrollo del modelo de la aplicación web.
    - Desarrollo del control de la aplicación web.
    - Desarrollo de la interfaz de la aplicación (la vista).
    - Pruebas para asegurar funcionamiento correcto de la aplicación.

Una vez completado el desarrollo de la aplicación y asegurado su correcto funcionamiento, el último paso sería preparar el código y entorno para una correcta instalación y configuración de la aplicación.

- Instalación y configuración de la aplicación.
    - Empaquetado de la aplicación.
    - Instalación y configuración del servidor donde se alojaría la aplicación.
    - Instalación de la aplicación.

2 Objetivos

El objetivo principal de este proyecto es el desarrollo de una aplicación web que permita la autoevaluación del alumnado de los distintos grupos pertenecientes a las materias del Depto. de Ingeniería Telemática. Para ello se desarrollaría una aplicación que se alojará en un servidor web.

Conseguir el objetivo principal, requerirá los siguientes objetivos secundarios:

1. Análisis y diseño de las funciones que ha de ofrecer la API.

En primer lugar, se estudian y definen las características y herramientas de las que dispondrá la aplicación, así como de su modo de actuación.

2. Diseño de la Base de Datos.

Una vez se han analizado los servicios y la funcionalidad que ha de tener la aplicación, se realiza el diseño inicial de la base de datos. El lenguaje utilizado para la gestión de la Base de Datos será POSTGRESQL. Esta base de datos se irá conformado con la incorporación de nuevas tablas a lo largo de su desarrollo.

3. Desarrollo de la API.

Analizados los requisitos que ha de cumplir la aplicación y del soporte para los datos, se llega a la fase de desarrollo de la aplicación. Para ello se implementará la lógica necesaria para prestar los servicios proyectados, así como se realizarán las pruebas necesarias para asegurar su correcto funcionamiento. La aplicación estará fundamentada en lenguaje HTML en su última versión (HTML5) y maquetada en CSS. Las funciones y servicios web se presentan en lenguaje JAVA por parte del servidor, junto con su Framework JSP para darle más dinamismo a la visualización del usuario.

4. Implementación a los servidores.

Para completar el desarrollo de la aplicación y comprobar su correcto funcionamiento, se ha desarrollado el código en un entorno preparado y adaptado al servidor, con lo que las pruebas definitivas se realizarán una vez alojadas. Para ello debemos empaquetar el código, junto con su base de datos, para la instalación final de la aplicación en el servidor.

5. Posibilidad de ampliación de funciones.

La aplicación desarrollada en este proyecto se dirige para la autoevaluación del alumnado en las diferentes materias del Depto. de Ingeniería Telemática; aunque debido a sus características se pueden añadir nuevas funcionalidades que la perfeccionen. Un ejemplo sería la inclusión de criterios con las que el alumnado podría completar la calificación.

3 Análisis del Problema

Se realizará un análisis teórico de las necesidades que ha de satisfacer la aplicación una vez se encuentre desarrollada. Este análisis comenzará identificando el problema e indicando qué es lo que se pretende conseguir. Una vez identificado el problema, se expondrá la solución propuesta en este proyecto.

3.1. Definición del problema

Mediante la realización de este proyecto se pretende desarrollar una herramienta que permita la renovación tecnológica del método de evaluación para así ofrecer un mejor servicio a los profesores y alumnos. Otro punto a tener en cuenta es el facilitar el proceso de gestión de cursos. Este punto también deberá contemplarse también en el desarrollo del proyecto.

Para cumplir estos cometidos será necesario desarrollar una herramienta que permita a los alumnos a poder logarse a la aplicación mediante una contraseña individual. Se deberá ofrecer la forma de que se pueda cambiar la contraseña a voluntad del alumno.

Otro punto importante para ofrecer un mejor servicio, es el de ofrecer la posibilidad de añadir nuevas asignaturas a las que poder gestionar. En este caso el profesor podrá añadir nuevas asignaturas con nuevos grupos, de esta forma los alumnos podrán disfrutar de la aplicación para calificar de forma online.

A parte de las funcionalidades descritas para la mejora de la aplicación de cara al alumno, también se deberán desarrollar mecanismos que permitan una gestión más sencilla de los grupos. Se deberá disponer de una herramienta que permita gestionar los permisos para que el profesor permita a los alumnos puntuar cuando él decida.

Por último será necesario un componente que permita interpretar los datos de las calificaciones realizadas.

3.2. Solución propuesta

Para dar solución a todo lo requerido en el punto anterior, se ha decidido implementar una aplicación web que permita cubrir todas las funciones anteriormente descritas.

De esta manera se conseguirá realizar la renovación tecnológica del método de evaluación, permitiendo ofrecer un mejor servicio tanto al profesor como al alumno, al dar acceso a evaluar a los grupos desde cualquier dispositivo y desde cualquier lugar.

Por otro lado, se mejorará la eficiencia en la gestión de las asignaturas gracias al mantenimiento de los datos centralizados en la aplicación. Además se permitirá el acceso y modificación de los datos necesarios mediante una interfaz intuitiva.

El lenguaje utilizado para el desarrollo de la aplicación web será Java, haciendo uso de la tecnología de los Servlets para el control de la aplicación y de las páginas JSP para representar los datos. Estos archivos se encontrarán alojados en un servidor web que incorporará al contenedor de Servlets Apache Tomcat. Éste será el encargado de compilar y ejecutar el código que conformará la aplicación web.

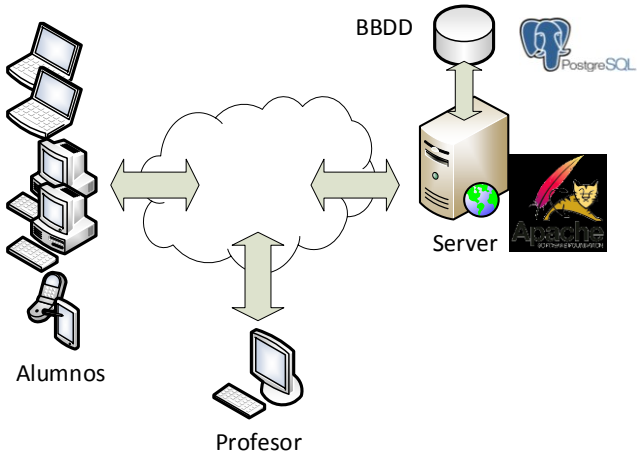


Figura 2. Esquema de funcionamiento de la API

Los datos de la aplicación serán almacenados en una base de datos gestionada con PostgreSQL y se accederá a través del servidor mediante las funciones necesarias, escritas en Java, que ejecutarán las sentencias necesarias para extraer o introducir la información requerida de la base de datos. En la Figura 2 se puede observar un esquema de cómo estaría construida la aplicación.

Al acceder a la aplicación, un alumno deberá logarse con una contraseña que siga un patrón de datos conocidos por él. El profesor especificará cual será el patrón que sigue dicha contraseña. Una vez logado, podrá acceder al servicio de puntuación y elegir una asignatura a la cual quiere calificar los distintos grupos que hay, siempre y cuando el profesor haya habilitado los permisos.

Para solventar los requisitos para la gestión de las asignaturas, se ha de desarrollar una serie de herramientas cuyo acceso estará restringido a los profesores.

4 Diseño de la solución

En este capítulo se detallará de forma más técnica las acciones y decisiones a tomar sobre cómo se ha de desarrollar la solución del problema expuesto en el capítulo anterior. El primer paso será analizar las tecnologías y herramientas con las que se puede llevar a cabo la aplicación, desde qué tecnologías se pueden utilizar para almacenar los datos hasta en qué lenguajes es podría desarrollar la aplicación, para después seleccionar las tecnologías que más convengan para confeccionar la aplicación.

Tras ello, se analizará la arquitectura de la aplicación desarrollada. Se describirá el diseño de la base de datos de la aplicación y también se describirá el cómo se relacionan entre sí las distintas capas de software.

4.1. Análisis de las tecnologías usadas

En este apartado se analizarán las posibles herramientas y tecnologías utilizables para desarrollar la solución al problema planteado. Se comenzará por un análisis de los distintos tipos de bases de datos, concluyendo con la selección del sistema gestor de bases de datos que se utilizará: PostgreSQL.

El siguiente paso a seguir es el de analizar los posibles lenguajes con los que implementar la aplicación web y, una vez expuestos los contendientes, se razonará el por qué se va a utilizar Java en su versión para aplicaciones web (Servlets y JSP).

El último paso, una vez determinada la tecnología que se va a utilizar, es el de seleccionar la infraestructura que dará soporte y que será encargada de prestar el servicio. Se analizarán los posibles servidores capaces de soportar esta tecnología y, para terminar, se elegirá cual servidor será el utilizado para albergar la aplicación.

4.1.1. Base de datos

Una base de datos es un conjunto de datos almacenados de forma estructurada y diseñada para hacer posible el acceso a las personas o aplicaciones autorizadas y satisfacer sus requisitos de información.

El uso de bases de datos en aplicaciones web es necesario para el correcto funcionamiento de la aplicación. La base de datos, será la encargada de guardar la información de los usuarios que deba persistir más allá de la sesión. También es necesaria para albergar toda la información necesaria para el correcto funcionamiento de la aplicación así como para nutrir a la aplicación de los contenidos que la dinamizarán.

En el caso de la aplicación que se desarrollará en este proyecto, la base de datos será la encargada de contener la información de los alumnos, cursos y grupos. También deberá mantener un registro de las calificaciones efectuadas por los alumnos tanto como los criterios como sus preferencias.

Para facilitar el acceso y mantenimiento de la base de datos y realizar consultas y realizar otras operaciones en la base de datos es necesario utilizar un Sistema Gestor de Base de Datos (SGBD).

Un Sistema Gestor de Base de Datos es una aplicación que permite realizar las acciones descritas anteriormente, garantizando la integridad de los datos, su fiabilidad y seguridad.

Al conjunto formado por la base de datos, el SGBD y las aplicaciones que dan uso de la base de datos se le denomina Sistema de Bases de Datos. En la siguiente figura se muestra el esquema que seguiría un sistema de bases de datos.

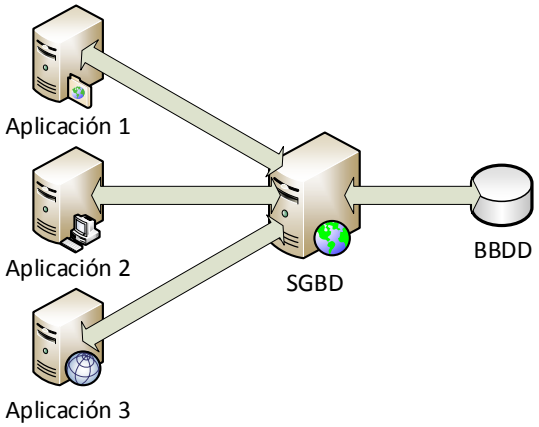


Figura 3. Esquema de funcionamiento de la BBDD

La base de datos relacional se caracteriza por almacenar los datos estructurados en forma de tablas referenciadas entre sí cuando exista relación entre ellas. Algunos ejemplos de SGBD que siguen este modelo son: MySQL, SQLite, PL/SQL, PostgreSQL, etc.

En cambio, las bases de datos no relacionales almacenan los datos de formas variadas: ya sea en forma de pares nombre-valor, en forma de objetos, etc. Este tipo de bases de datos permiten una mayor velocidad en las consultas a costa de carecer de una estructura o esquema tan definido como en el caso del modelo relacional.

En el caso de la aplicación que se va a desarrollar se ha decidido utilizar un modelo de base de datos relacional. Se trata de una tecnología muy madura y ampliamente utilizada, además de que la información a almacenar en la base de datos se amolda perfectamente al esquema lógico de este modelo.

De entre todos los sistemas gestores de base de datos existentes se ha optado por utilizar PostgreSQL, un potente gestor de bases de datos relacionales distribuido bajo licencia BSD. Se ha ganado una gran reputación por su estabilidad, potencia y robustez. Está disponible para los principales sistemas operativos y tiene interfaces para una gran cantidad de lenguajes de programación.

Algunas de las características más importantes de PostgreSQL:

- Es una base de datos 100% ACID.
- Disponible para la mayoría de sistemas operativos.
- Dispone de interfaz nativa para programar en los principales lenguajes (C/C++, Java, Perl, Python, etc.).
- Dispone de una documentación muy completa y una gran comunidad detrás.
- Es software libre distribuido bajo una licencia BSD.

4.1.2. Lenguaje de programación web

Existen diversas tecnologías capaces de implementar la aplicación web que se desea mostrar al cliente. En este apartado se realizará un análisis de los lenguajes más comunes para este tipo de aplicaciones y se seleccionará el lenguaje que se considere más oportuno para implementar la aplicación.

Se distinguen dos tipos de programación web: programación del lado del cliente donde el código se interpreta en el navegador del usuario que accede a la aplicación (JavaScript), y programación del lado del servidor donde el código es interpretado en el servidor y se envía el resultado al cliente para su visualización (JSP).

- **HTML**:

  HyperText Markup Language, es el lenguaje de programación utilizado para la elaboración de páginas web. Se trata de un lenguaje basado en etiquetas mediante las que se indica al navegador la estructura del contenido de la página web y qué función realiza cada bloque de contenido. HTML utiliza una serie predefinida de etiquetas que permiten identificar qué tipo de información está contenida en cada bloque. Los elementos que forman una página web siguen una estructura jerárquica en la que un elemento puede contener otros elementos anidados.

  La forma de distinguir el contenido de cada elemento es mediante el uso de etiquetas. Una etiqueta consiste en el nombre del elemento rodeado por corchetes angulares (<html>).

  La mayoría de los elementos se encuentran delimitado entre las etiquetas de inicio, que indica dónde empieza el elemento, y una etiqueta de cierre, que indica el final del elemento (</html>). También existen excepciones y algunos elementos son identificados con una sola etiqueta.

  Aunque cada elemento tiene una representación definida, su función no es estética, sino la de estructurar la información que se muestra. Para las labores de estilo e indicar cómo se ha de ver una página web se utilizan las hojas de estilo CSS (Cascading StyleSheet).

  HTML es necesario para la representación de las páginas web, aunque sufre de un gran inconveniente: sólo genera contenido estático y no es capaz de generar contenido dinámico. Para ello hay que complementar su uso con algún otro lenguaje de programación web que sí sea capaz de generar contenido dinámicamente.

- **JavaScript**

  JavaScript es un lenguaje de programación interpretado. Su principal uso es para la programación de scripts que serán interpretados y ejecutados por el navegador del cliente y que permitirán una interacción más dinámica entre el cliente y la aplicación web.

  Mediante el uso de estos scripts, se puede controlar el comportamiento del navegador, generar una interfaz de usuario más agradable y dinámica, y además permite realizar una comunicación asíncrona con el servidor, de tal manera que no sea necesario refrescar la ventana para obtener nueva información.

Existen librerías para JavaScript que simplifican en gran manera el trabajo de desarrollo de la aplicación, permitiendo simplificar la forma de interactuar con los elementos de la página web, consiguiendo, con menos código que con JavaScript puro, controlar el comportamiento de la aplicación, desarrollar animaciones en la interfaz, así como comunicarse de forma asíncrona con el servidor mediante AJAX.

- **Servlets**

  El Servlet es una clase especial del lenguaje de programación Java utilizada para ampliar las capacidades de un servidor y permitir el acceso a aplicaciones basadas en el modelo de petición/respuesta (request/response).

  Aunque un Servlet puede responder a cualquier tipo de petición, su principal uso es para aplicaciones web y, por ello, contiene clases de objeto específicas para este tipo de aplicaciones basadas en el protocolo HTTP.

  La forma de implementar un Servlet es la misma que la utilizada para el desarrollo de cualquier aplicación escrita en el lenguaje Java, con la salvedad de que es necesario utilizar una serie de métodos específicos encargados de la lógica a ejecutar en cada fase del ciclo de vida del Servlet. Estos métodos son los utilizados para inicializar el Servlet (el método init()), para indicar las acciones a realizar cuando lleguen las peticiones del protocolo HTTP (los métodos doGet(), doPost()...) o para destruir el Servlet (el método destroy()).

- **Java Server Pages**

  JavaServer Pages (JSP) es una tecnología para el desarrollo de páginas web dinámicas mediante el uso del lenguaje de programación Java. Es una abstracción a alto nivel de un Servlet con la que se pretende facilitar el desarrollo de páginas web dinámicas a usuarios más familiarizados con el diseño web que con el diseño de aplicaciones. Las páginas web desarrolladas mediante JSP son traducidas a lenguaje Java, la primera vez que se ejecutan, formando un Servlet equivalente el cual será compilado y ejecutado por la máquina Java cada vez que se utilice y persistirá en caché hasta que la página JSP sea modificada.

  La estructura de una página web escrita en JSP es similar a la de una página escrita en PHP: incrusta fragmentos de código entre el contenido escrito en HTML. Para ello utiliza unas etiquetas especiales (<% para el comienzo del bloque y %> para el final) que separan un tipo de código del otro.

  Aunque, siguiendo la mentalidad de abstracción, para facilitar el desarrollo de las páginas web dinámicas y simplificar el código, JSP permite el uso de etiquetas personalizadas que sustituyen el uso de código Java para la programación. También permite el uso de expresiones (EL) para acceder a las propiedades de los objetos Java guardados en la sesión y rellenar de contenido la aplicación.

  JSP se suele utilizar en combinación con los Servlet para realizar aplicaciones que sigan una arquitectura MVC, encargándose los Servlets de la lógica de que ha de seguir la aplicación y las páginas JSP de la presentación de la información al usuario.

  Las ventajas de utilizar esta tecnología respecto a otras alternativas son las mismas que las de utilizar los Servlet ya que, como pasa con los Servlets, sus principales ventajas radican en el lenguaje en el que están basados.

- **Conclusión**:

  Para la elaboración de este proyecto se ha decidido seleccionar las siguientes tecnologías de entre todas las descritas en los puntos anteriores:

  - HTML. Esta tecnología es imprescindible para el desarrollo del proyecto ya que es el encargado de la elaboración de las páginas web. Se utilizará junto a la utilización de CSS para controlar la apariencia de la aplicación.

  - JavaScript. Este lenguaje será utilizado para dinamizar las páginas web que serán utilizadas por los alumnos. Se utilizará para las animaciones de todas las interactuaciones del usuario con la aplicación, así como para la recolección de información y la comunicación asíncrona con el servidor.

  - Java y Servlet. Se ha elegido esta tecnología para la elaboración de la lógica de control de la aplicación. Así su función será la de, tras recibir la petición del cliente, seleccionar la acción a realizar. También se utilizará Java para la programación de la lógica necesaria para el tratamiento de los datos y la interactuación con la base de datos.

  - JavaServer Pages. Esta tecnología será utilizada para desarrollar la parte de presentación de la aplicación. Las páginas jsp se encargarán de recoger la información enviada por los Servlets y representarla al alumno.

  La principal razón para elegir los Servlet y las páginas JSP frente a las otras tecnologías descritas es el lenguaje en el que se desarrollan, Java, que permite realizar proyectos mejor estructurados y más fáciles de mantener.

  Además dispone de una gran cantidad de librerías que permiten realizar todas las funcionalidades necesarias para la aplicación. Otro motivo para haber elegido Java es la comodidad de haber trabajado anteriormente con este lenguaje.

4.1.3. Server de la API

Para que la API pueda prestar su servicio a los alumnos es necesario que se encuentre desplegada en un servidor web que sea capaz de interpretar el lenguaje en el que esté escrita, en este caso Java, tanto para el caso de los Servlets, como para las páginas escritas con la tecnología jsp. No todos los servidores web son capaces de interpretar estas tecnologías de forma básica. Por ello es necesario utilizar un tipo de servidor específico llamado contenedor de Servlets.

Los contenedores de Servlets son servidores web, se encargan de recibir las peticiones de los clientes y preparar los recursos solicitados para su envío, preparados para la utilización de la tecnología Java. Para llevar a cabo esta función es necesario que interpreten el código fuente en el que están programados los Servlet, han de gestionar el ciclo de vida de los Servlet (inicialización, servicio y destrucción), y deben enviar el recurso web generado de forma dinámica.

El servidor seleccionado para este caso es el servidor Apache Tomcat. Se trata de una implementación de código abierto de las tecnologías Java Servlet y JavaServer Pages desarrollado por la Apache Software Foundation.

Tomcat está conformado por varios componentes que son encargados de las distintas funcionalidades:

Catalina, el contenedor de Servlet encargado de la gestión y el ciclo de vida de éstos; Coyote, un servidor web que permite a Tomcat recibir y servir las peticiones de los clientes y Jasper, encargado de interpretar y compilar el código proveniente de las páginas escritas en jsp.

Tomcat está escrito en Java, por lo que se trata de un servidor multiplataforma capaz de funcionar en cualquier sistema operativo que disponga de una máquina virtual Java.

Este servidor es la opción elegida en numerosas aplicaciones web de gran tamaño pertenecientes a una gran variedad de industrias y organizaciones.

4.2. Arquitectura del software

En este apartado se describirá de forma detallada la estructura de la aplicación desarrollada en el proyecto. Se trata de la fase de diseño, en la que, partiendo de la solución propuesta en la fase de análisis y de las tecnologías seleccionadas, se dará forma a las distintas partes de la aplicación.

En primer lugar se detallarán los pasos seguidos hasta llegar a realizar la base de datos de la aplicación. A continuación se explicará cómo está diseñada la aplicación, partiendo de la estructura que va a seguir, MVC, y continuando de forma pormenorizada con el diseño de cada uno de los componentes de esta estructura.

4.2.1. Diseño de la Base de datos

La base de datos de la aplicación será la encargada de almacenar los datos pertinentes para el correcto funcionamiento de la aplicación. Deberá guardar la información de los usuarios, además de la información relevante de los cursos. También ha de tener guardada la información de los grupos así como las calificaciones de los alumnos.

El diseño de una base de datos continúa con una serie de pasos que se dirigirán hacia una solución específica que se ajuste perfectamente a la aplicación desarrollada, almacenando los datos de forma eficiente y permitiendo su consulta de forma sencilla.

En este apartado se describe los pasos de diseño seguidos para llegar a la solución deseada. Se comenzará analizando las restricciones que habrán de tener los datos almacenados. A continuación se realizará el modelado de los datos, comenzando por una recopilación de posibles datos en el que se muestre las necesidades a almacenar y, posteriormente, se traducirá al modelo relacional asociado, donde se planteará cómo se almacenarán los datos en la base de datos.

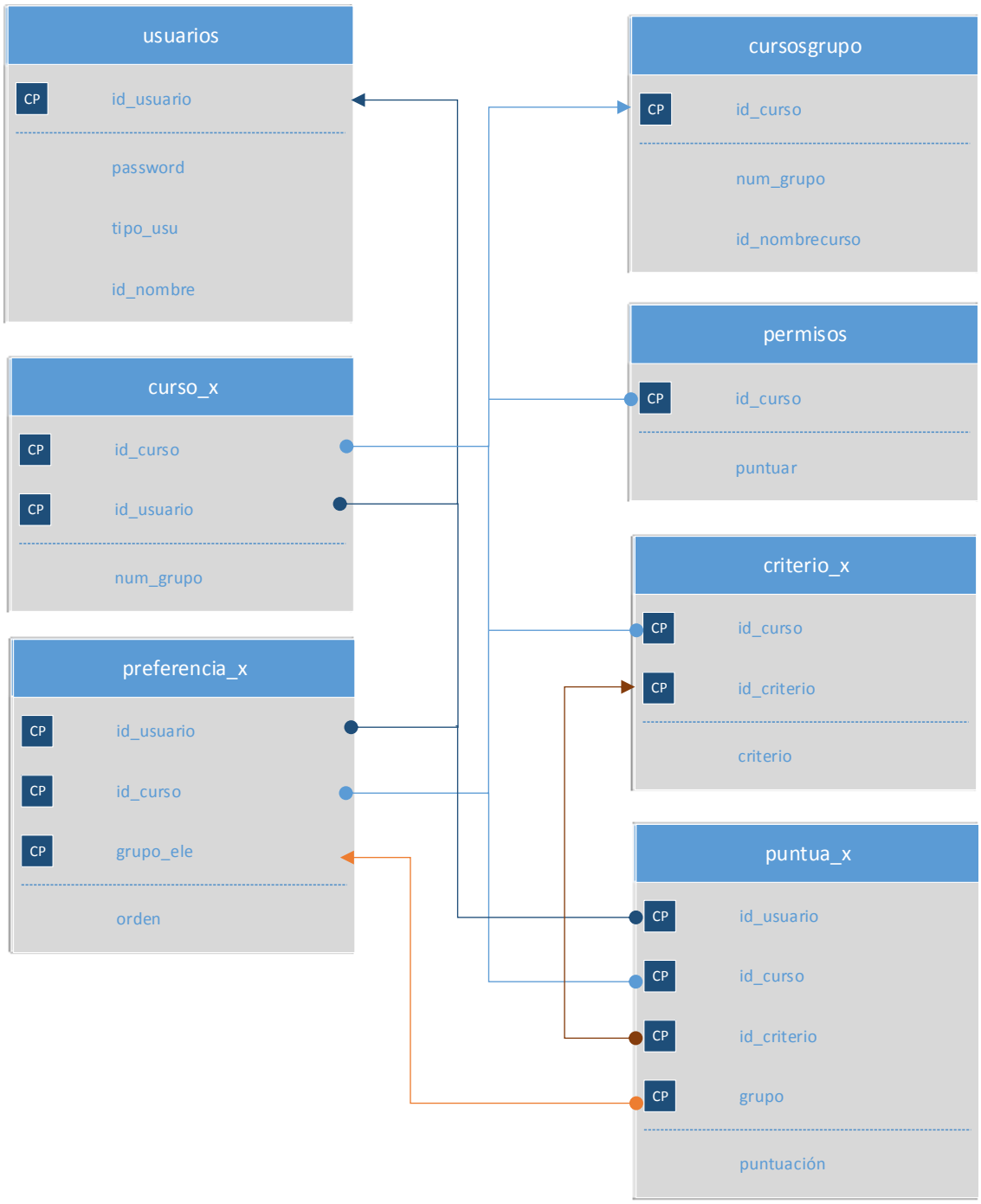
- **Datos necesarios para su provisión**

  El siguiente paso en el diseño de la base de datos es realizar el modelado de los datos. Se han de identificar los conjuntos de datos que se han de almacenar y la relación existente entre estos conjuntos.

- Datos necesarios para el alumno
    - Uvus
    - Password
    - Tipo de usuario
    - Nombre
- Datos necesarios para las asignaturas
    - Nombre de curso
    - Número de grupos
    - Identificador del curso
- Datos necesarios para los grupos
    - Uvus
    - Grupo
    - Identificación del curso
- Datos necesarios para puntuación
    - Uvus
    - Grupo puntuado
    - Puntuación
    - Identificación del curso
    - Criterio
- Datos necesarios para preferencia de grupos
    - Uvus
    - Grupo elegido
    - Orden
    - Identificación del curso
- Datos necesarios para diferenciar los distintos criterios de evaluación
    - Identificación de curso
    - Identificación del criterio
    - Criterio

- Datos necesarios para los permisos de asignaturas
  - Identificación del curso
  - Bandera de permiso

- **Traducción a Base de Datos**



4.2.2. Diseño de la API

Una vez se dispone de una base de datos en la que almacenar la información necesaria para la aplicación, el siguiente paso es el desarrollo de la aplicación que proporcionará los servicios necesarios para solventar el problema planteado en este proyecto.

Analizando los requisitos del problema, la solución propuesta se puede dividir en dos perfiles bien diferenciados en cuanto al uso que se dará a la aplicación: por una parte se encuentra el uso que ejercerán los administradores, en este caso el profesor y, por otra parte, el uso que harán de esta aplicación los alumnos.

Para el desarrollo de la aplicación se ha pensado seguir el patrón de arquitectura modelo-vista-controlador que permite modular la aplicación y separar los datos y la lógica implementada para el tratamiento de los datos de su representación en la aplicación. Esta arquitectura divide la aplicación en tres bloques, cada uno encargado de una función específica:

* Modelo: Parte encargada del tratamiento de los datos de la aplicación. Obtiene la información de la base de datos y la adapta a las necesidades de la aplicación.

* Controlador: Se encarga de recibir las peticiones del usuario (acciones que realice en la aplicación) y llama al modelo para obtener los datos requeridos. Una vez recibidos los datos del modelo, es el encargado de enviarlos a la vista para su representación.

* Vista: Presenta al alumno los datos recibidos del modelo. Se trata de la interfaz de usuario con la aplicación.

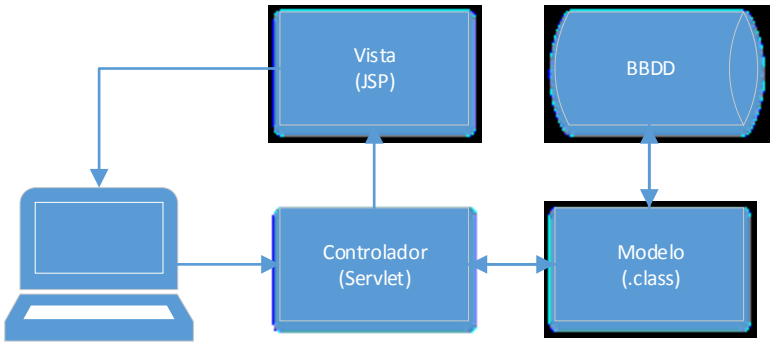


Figura 4. Diagrama Modelo-Vista-Controlador

5 Implementación de la API

En este capítulo se detallarán las distintas herramientas y servicios que ofrece la aplicación desarrollada para solventar el problema planteado. Los servicios se pueden clasificar entre los que serán utilizados por el profesor dos de la aplicación, ya sea para la administración de la API o para la gestión de cursos, y los servicios desarrollados para el uso de la aplicación por parte del profesor.

5.1. Servicios ofrecidos por el administrador (Profesor)

La aplicación desarrollada ofrece una serie de herramientas y servicios enfocados al profesor que facilitarán sus tareas y les permitirán realizarla de forma más eficiente. Los servicios ofrecidos a los administradores serán distintos de los que se ofrecerán a los alumnos que el uso que harán de la aplicación será diferente, más orientado a la administración de la aplicación.

El acceso a las herramientas de administración que se describen a continuación se realizará a través de las páginas web que conforman la vista de la aplicación. En este caso, la vista para administradores.

A continuación se describe la interfaz de la aplicación para los profesores. En un primer lugar se explicarán el acceso a la aplicación y la pantalla de inicio que se les mostrará. A continuación se describirán las herramientas que podrán utilizar.

5.1.1. Acceso e inicio de la API

El acceso a la aplicación específica para los profesores se realiza a través de una dirección web distinta a la de la página web accesible por el profesor. En concreto, se accede a esta aplicación introduciendo la siguiente URL: <dirección_de_la_página_web>/EvalProyecto/.

Lo primero que aparece al acceder a esta aplicación es un mensaje de bienvenida a modo de presentación de la API.

Figura 5. Inicio de sesión

Una vez pulsado el botón de **login**, aparecerá el formulario de inicio de sesión, el cual hay que rellenar con un usuario y contraseña válidos para poder acceder a los servicios de la aplicación. Si los datos introducidos no son válido se volverá a mostrar la página de bienvenida, a modo de error.

Figura 6. Formulario de Inicio de Sesión

5.1.2. Servicio ofrecidos al administrador (profesor)

Son los servicios y herramientas pensados para el uso del profesor, cuyas funciones serán la creación y administración de cursos, gestión de alumnos y grupos, creación y modificación de alumnos, administración de permisos, lista de alumnos matriculados y visualización de notas.

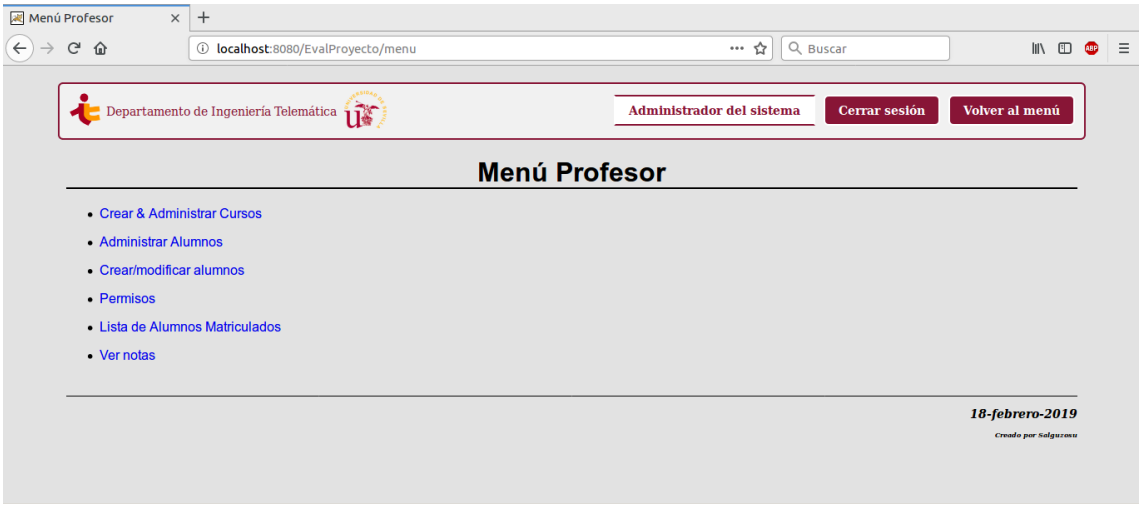


Figura 7. Página de Inicio de Menú del Administrador (Profesor)

- **Creación & Administración de Cursos**

En esta página permitirá la creación y modificación de cursos como función principal, además como funciones adicionales estará habilitado para los cursos recién creados que no tenga criterios de evaluación además de un listado con los cursos actuales.

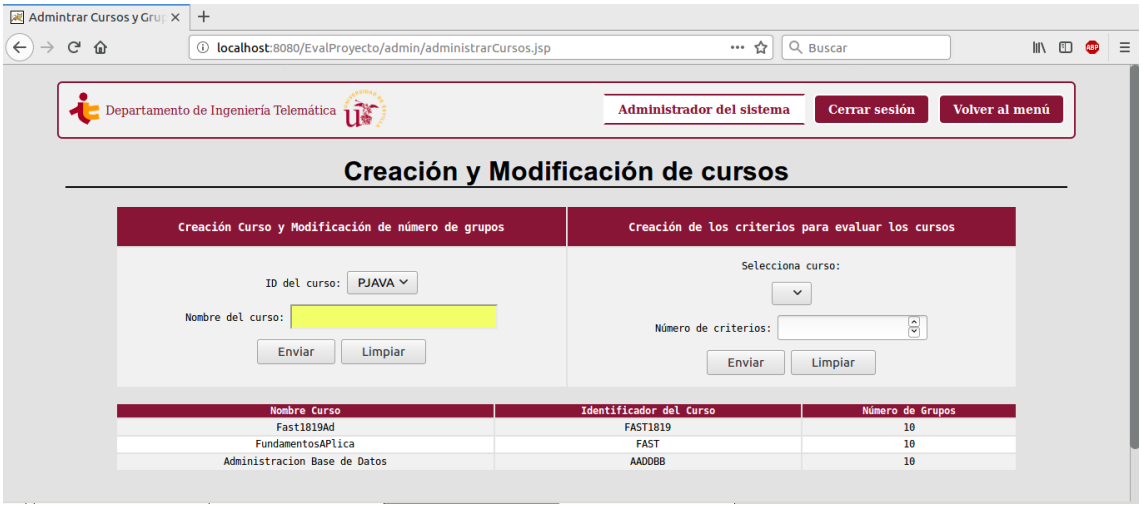


Figura 8. Página de Creación y Modificación de cursos

En la creación del curso aparecerá un formulario el cual tendrá como campos el id del curso, el cuál será el que identifica el curso, el nombre del curso y el número de grupos que tendrá el curso.

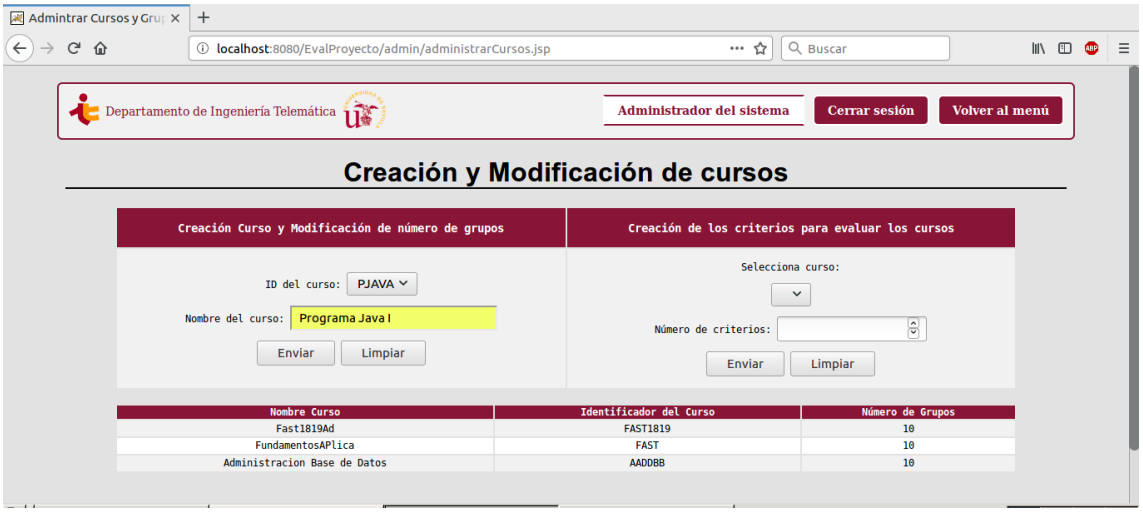


Figura 9. Formulario de Curso

Una vez relleno el formulario, aparecerá en el listado de cursos y estará habilitado para añadir los criterios de evaluación.

Figura 10. Página de Creación y Modificación una vez creado el curso

Una vez ya creado el curso, y elegido el número de criterios, se desplegará un nuevo formulario, el cual dependerá del número de criterios que se vayan a evaluar.

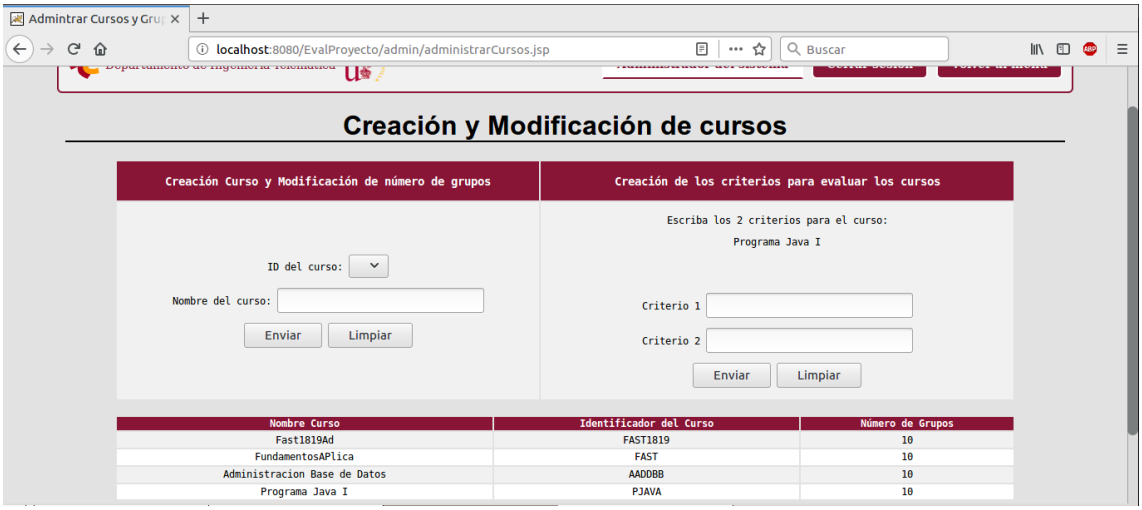


Figura 11. Formulario de creación de criterios

Una vez relleno y enviado el formulario, se volverá a la página inicial de la creación y modificación del curso.

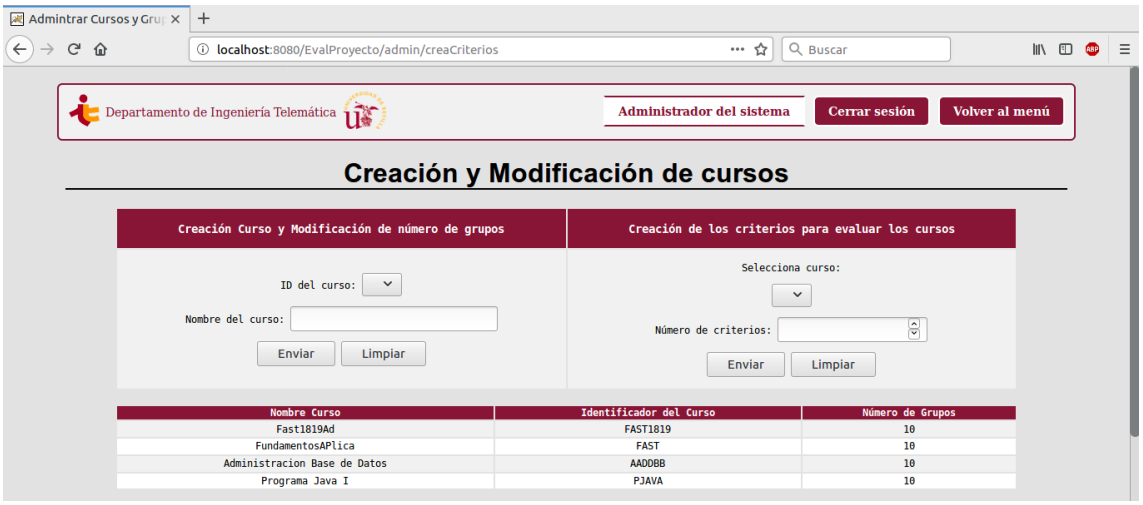


Figura 12. Página Inicial tras creación de curso y criterios

- **Administrar Alumnos**

En esta página permitirá la administración de grupos, para primero se tendrá que seleccionar el alumno a administrar de la lista desplegable donde estarán todos los alumnos que tengan acceso a la plataforma.

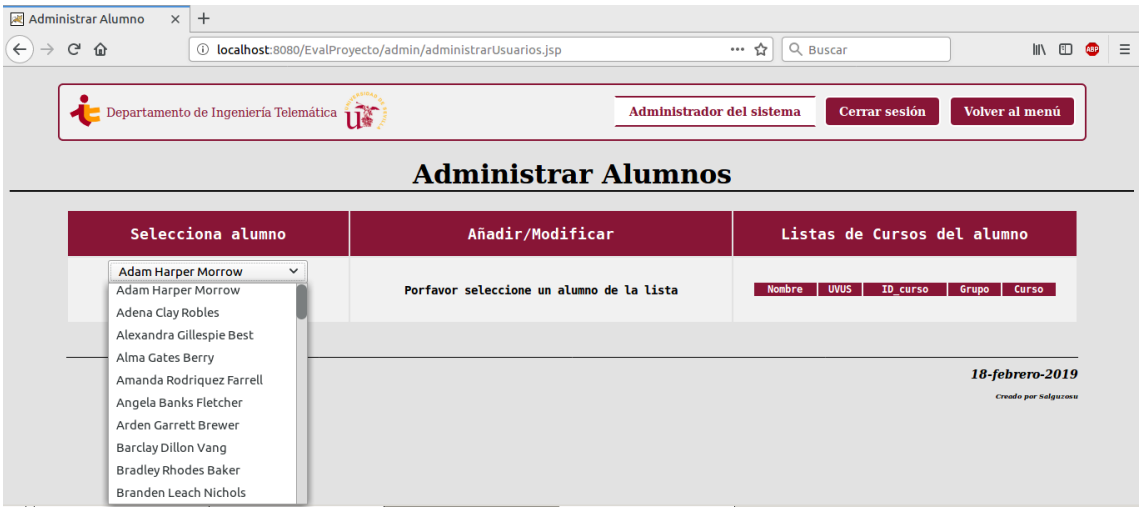


Figura 13. Administración de Alumnos

Una vez elegido el alumno, aparecerá un formulario con los campus UVUS, grupo y la asignatura, de esta manera se podrá añadir al alumno a un nuevo grupo de una asignatura, o si este ya está en un grupo poder cambiarlo. Para ello se dispone de una tercera columna en la que se recoge una tabla con las asignaturas y los grupos elegidos por el alumno.

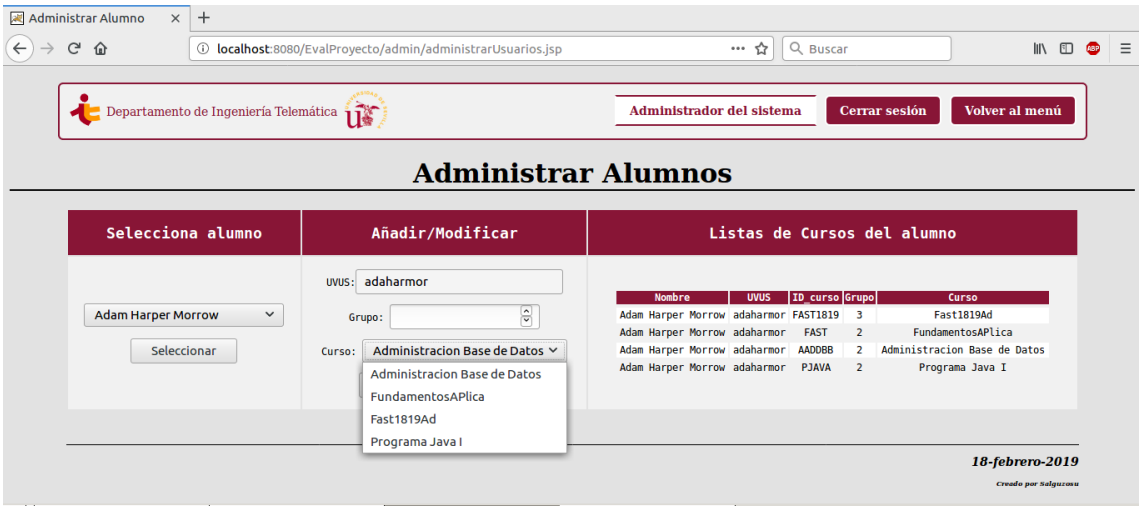


Figura 14. Cursos matriculados y grupos del alumno

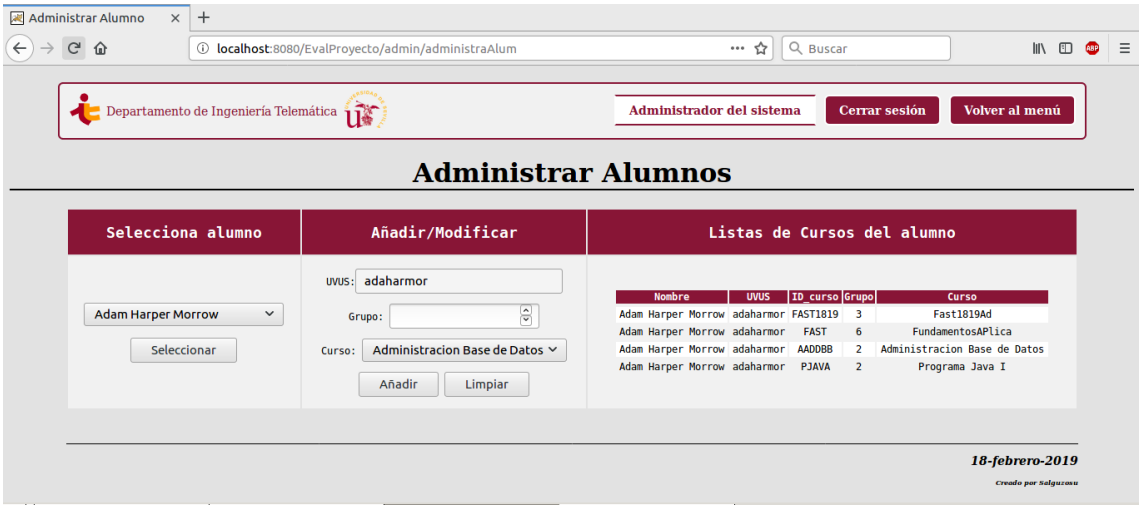


Figura 15. Añadido nuevo curso y grupo

- **<u>Creación/Modificación de Alumnos</u>**

En esta página permitirá tanto creación de alumnos como la modificación de algunos de sus campos como puede ser su contraseña o su nombre de usuario, menos el UVUS, que será fundamental para su identificación dentro del funcionamiento de la API.

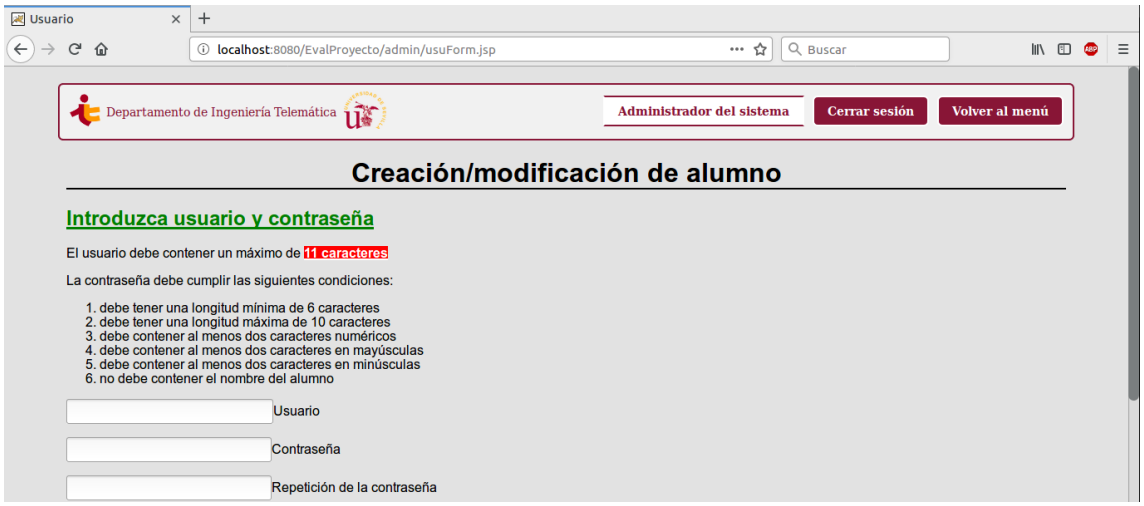


Figura 16. Página inicial de Creación/Modificación de alumno.

- **Ver Permisos**

En esta página se gestionará los permisos de puntuación de todos los cursos para dar o quitar permisos a los alumnos para que puedan empezar a puntuar.

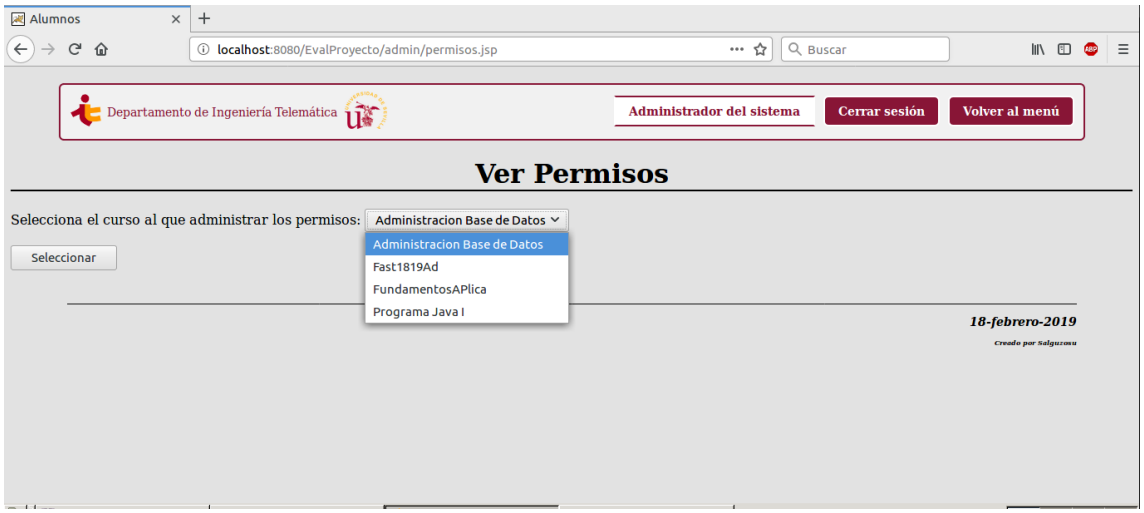
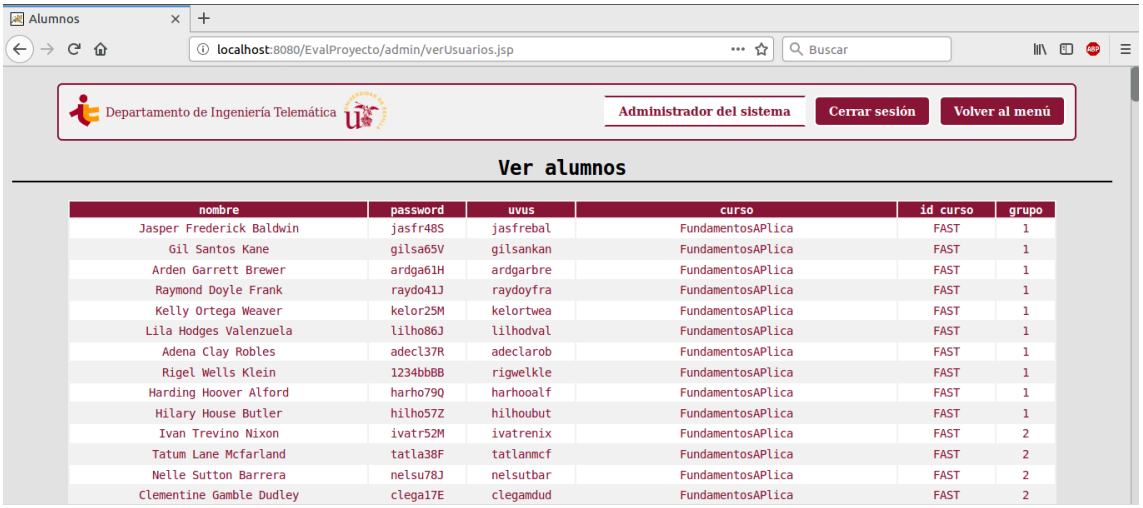


Figura 17. Elección de curso para administrar permisos

Figura 18. Gestión de Permisos

- **Ver Alumnos matriculados**

En esta página se listará todos los alumnos que estén matriculados a una asignatura y con un grupo seleccionado.



nombre	password	uvus	curso	id curso	grupo
Jasper Frederick Baldwin	jasfr48S	jasfrebal	FundamentosAPlica	FAST	1
Gil Santos Kane	gilsa65V	gilsankan	FundamentosAPlica	FAST	1
Arden Garrett Brewer	ardga61H	ardgarbre	FundamentosAPlica	FAST	1
Raymond Doyle Frank	raydo41J	raydoyfra	FundamentosAPlica	FAST	1
Kelly Ortega Weaver	kelor25M	kelortwea	FundamentosAPlica	FAST	1
Lila Hodges Valenzuela	lilho86J	lilhodval	FundamentosAPlica	FAST	1
Adena Clay Robles	adecl37R	adeclarob	FundamentosAPlica	FAST	1
Rigel Wells Klein	1234bbBB	rigwelkle	FundamentosAPlica	FAST	1
Harding Hoover Alford	harho79Q	harhooalf	FundamentosAPlica	FAST	1
Hilary House Butler	hilho57Z	hilhoubut	FundamentosAPlica	FAST	1
Ivan Trevino Nixon	ivatr52M	ivatrenix	FundamentosAPlica	FAST	2
Tatum Lane Mcfarland	tatla38F	tatlanmcf	FundamentosAPlica	FAST	2
Nelle Sutton Barrera	nelsu78J	nelsutbar	FundamentosAPlica	FAST	2
Clementine Gamble Dudley	clega17E	clegamdud	FundamentosAPlica	FAST	2

Figura 19. Lista de alumnos matriculados

- **Ver Notas**

En esta página se listará todas las calificaciones aportadas por los alumnos a los distintos grupos según los criterios de las asignaturas.

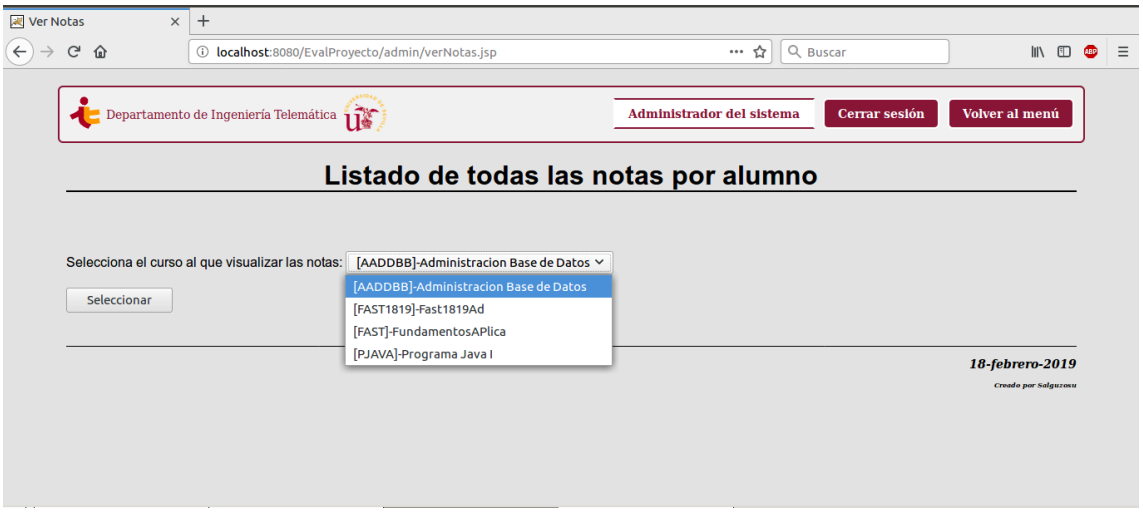


Figura 20. Listado de todas las notas por curso I.

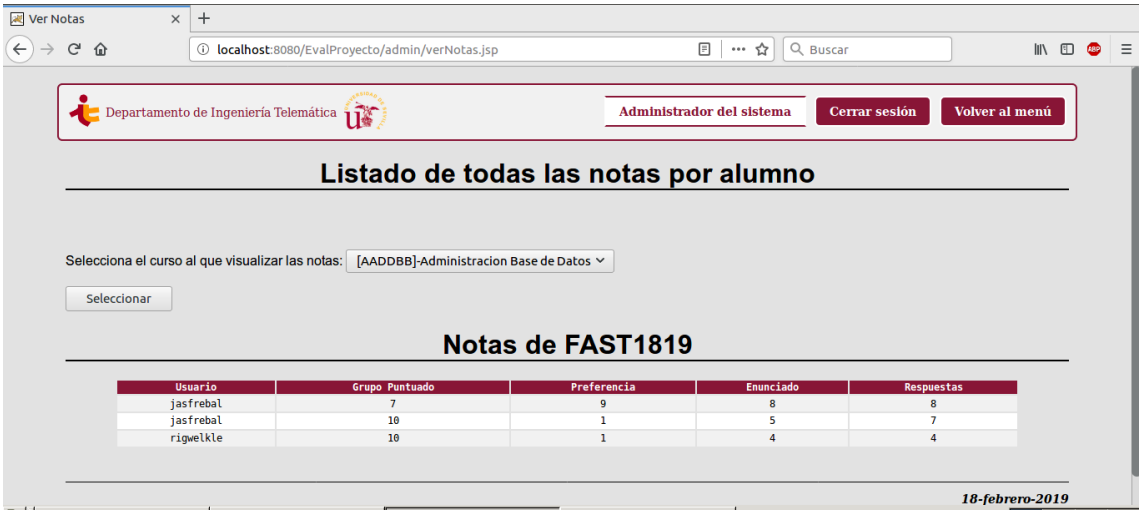


Figura 21. Listado de todas las notas por curso II.

5.2. Servicios ofrecidos por el alumno

El otro perfil que va a utilizar la aplicación es el de los alumnos. Para el uso de la aplicación por parte de éstos se ha diseñado una serie de herramientas y servicios.

El acceso a estos servicios que componen este perfil de la aplicación se realizará a través de la misma interfaz formada por las páginas web que conforman la vista de la aplicación.

Dentro de los servicios ofrecidos al alumno se diferenciará entre los servicios calificación de los grupos de las distintas asignaturas a las que está matriculada, y la opción de cambiar de contraseña.

5.2.1. Acceso e inicio de la API

El acceso a la aplicación específica para alumnos se realiza a través de una dirección web distinta a la de la página web accesible por los alumnos. En concreto, se accede a esta aplicación introduciendo la siguiente URL: <dirección_de_la_página_web>/EvalProyecto/.

Figura 22. Página de Inicio Alumnos I.

Figura 23. Página de Inicio Alumnos II

5.2.2. Servicio ofrecidos al alumno

Son los servicios y herramientas pensados para el uso del alumno, cuyas funciones serán calificación de los grupos y el cambio de contraseña.

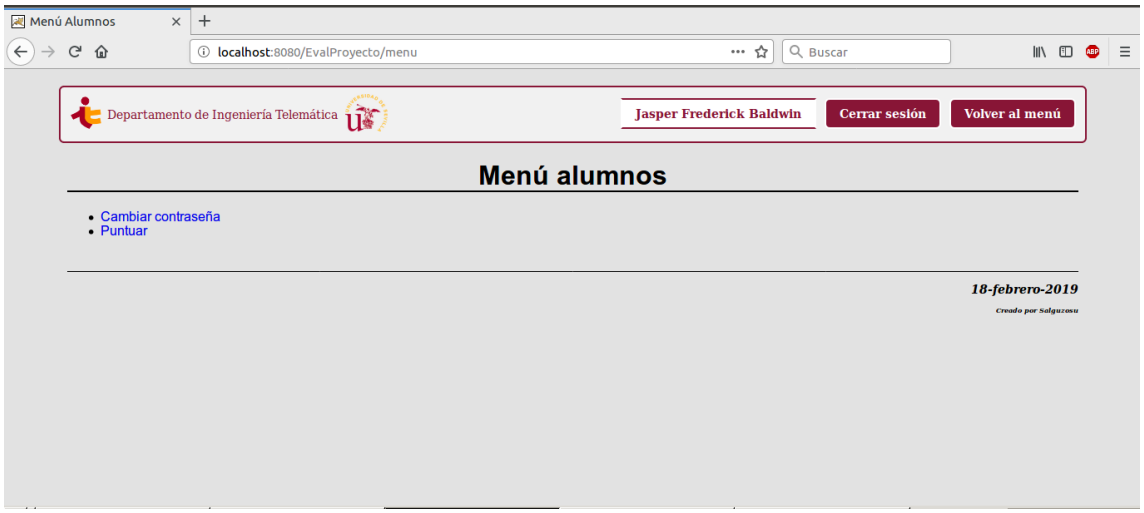


Figura 24. Menú de alumnos

- **Cambiar contraseña**

En esta página permitirá la modificación de la contraseña del usuario logado, siempre que se cumpla los criterios especificados.

Figura 25. Modificación de contraseña

- **Puntuar**

En esta página será la funcionalidad más importante de este proyecto, ya que es la calificación de los grupos y lo que usará los profesores. Lo primero que será necesario es seleccionar aquel curso que este el alumno matriculado y además que tenga permisos, sino no aparecerá como seleccionable.

Figura 26. Página de puntuación

Una vez seleccionado el curso, se habilitará una lista desplegable de todos los grupos menos el propio al que pertenezca el alumno

Figura 27. Elección de grupo

Elegido el grupo, se mostrará la forma de calificar a partir de la preferencia del grupo respecto al resto además de los criterios elegidos en la creación del curso.

Figura 28. Página de Calificación del alumno al grupo 9 de la asignatura

6 CONCLUSIONES

Durante la realización de un proyecto se producen cambios e imprevistos que provocan que la planificación original se vea alterada. Es recomendable realizar una evaluación de la situación tras la realización de dicho proyecto y así contemplar el estado final, evaluar la desviación ocurrida y sacar las conclusiones oportunas del estado final del proyecto.

En este apartado se intentará plasmar los imprevistos y puntos fuertes encontrados durante el proceso de desarrollo de la aplicación desde un nivel técnico, marcando las dificultades e inconvenientes encontradas, en el desarrollo de la aplicación y que provocaron el desvío respecto a la planificación inicial, como también desde un nivel personal, viendo qué impacto personal ha tenido la realización de este trabajo.

Por último, se propondrán líneas futuras y posibles mejoras realizables que aportarían un valor añadido a la aplicación y que no se realizaron, bien por encontrarse fuera del alcance de la aplicación y prolongarían el tiempo de desarrollo o bien por desconocimiento de la tecnología necesaria para llevarlas a término.

6.1. Conclusiones

En este capítulo se detallará de forma más técnica las acciones y decisiones a tomar sobre cómo se ha de desarrollar la solución del problema expuesto en el capítulo anterior. El primer paso será analizar las tecnologías y herramientas.

- Entre las mayores dificultades que he tenido durante la realización de este proyecto están las relacionadas con el entorno. Crear un entorno desde cero sin poder comprobar los posibles conflictos de compatibilidad o, en este caso, adaptar un entorno ya preparado o existente, con lo que conlleva.

- He conseguido diseñar un comportamiento correcto en cada una de las funcionalidades que se han pensado como necesarias para la aplicación. Se han desarrollado herramientas para la gestión de contraseñas, de los alumnos, de los cursos, de los grupos y de las notas.

- Realizar el diseño de una base de datos es una labor muy importante, ya que el funcionamiento correcto de la aplicación depende en gran medida de cómo serán almacenados sus datos y las relaciones entre ellos para realizar las consultas conjuntas. También se han de definir los límites y restricciones que han de tener estos datos. Y otras tareas como la traducción del diseño de la base de datos a lenguaje SQL y el diseño de las consultas a realizar a la base de datos.

- Una vez que ya tenemos definida la base de datos, es fundamental la interpretación correcta de los datos aportados por las plataformas, ya que muchos de ellos son repetitivos, y algunos deben de ser obviados, por no aportar información significativa para la aplicación.

- La aplicación desarrollada en este proyecto ha supuesto un gran reto personal debido principalmente a la complejidad, depuración y testeo de la misma, que han requerido mucho tiempo de aprendizaje de los distintos sistemas y lenguajes que la componen, además de un

correcto análisis de las pruebas funcionales, en la detección de posibles fallos de lógica o funcionamiento, una vez se encuentra la aplicación online.

En conclusión, ha sido una experiencia satisfactoria que me ha permitido gestionar un proyecto similar a cualquier proyecto profesional, con el que he aprendido a usar correctamente las tecnologías, HTML5, CSS, JavaScript y la preparación de datos mediante VBA, y permitido poner en práctica conocimientos adquiridos, como Java (con el uso de los Servlets y JavaServer Pages) o Bases de Datos.

6.2. Líneas futuras

Ninguna aplicación es perfecta y a todos los sistemas se les pueden introducir mejoras y la aplicación desarrollada en este proyecto no es una excepción.

Las posibles mejoras a introducir en la aplicación estarían dirigidas a aumentar su valor, ya sea añadiendo funcionalidades nuevas u optimizando las implementadas.

Una posible mejora podría ser un servicio de notificaciones mediante mensajería, con la que el alumnado estaría informado en cualquier momento. Para ello, sería necesario pedir la correspondiente autorización individual.

Otra mejora podría ser solicitar justificación del criterio de calificación por parte del alumnado, de esta forma el profesor tendría una visión más completa de las evaluaciones realizadas.

Y en tercer lugar, una última posible mejora sería el desarrollo de la lógica y funcionamiento de la API desarrollada en otros lenguajes más potentes, como Python o Android, pudiendo dar origen a una aplicación móvil más versátil que una web.

7 BIBLIOGRAFÍA

[1] Usuarios, «StackOverFlow,» [En línea]. Available: https://stackoverflow.com/.

[2] W3Schools, «W3Schools,» [En línea]. Available: https://www.w3schools.com/.

[3] PostgresSQL, «Documentacion Ubuntu,» [En línea]. Available: https://help.ubuntu.com/community/PostgreSQL.

[4] ORACLE, «oracle documentacion,» [En línea]. Available: https://www.oracle.com/technetwork/java/index-jsp-135475.html.

[5] Dit, «Trajano,» [En línea]. Available: http://trajano.us.es/.

[6] EV, «Plataforma Virtual,» [En línea]. Available: https://ev.us.es/webapps/portal/execute/tabs/tabAction?tab_tab_group_id=_29_1.

[7] Eclise, «Eclipse documentación,» [En línea]. Available: https://www.eclipse.org/documentation/.

8 . ANEXOS A: MANUAL DE INSTALACIÓN

Para poder utilizar correctamente la aplicación es necesario tener un entorno adecuada capaz de interpretar el lenguaje de programación en el que se ha desarrollado la aplicación. En este caso será necesario que se disponga de un servidor que tenga instalados un contenedor de Servlets y una base de datos que almacene toda la información.

Una opción sería alojar la aplicación en el ordenador del profesor en una máquina virtual preconfigurada a modo de localhost. De esta manera tan solo sería necesario realizar el despliegue de la aplicación, despreocupándonos del entorno.

En este anexo se explicará los pasos a seguir para la configuración inicial de la aplicación tanto en un equipo con sistema operativo Ubuntu.

8.1. Preparación de los datos

Lo primero que necesitaremos son los datos que vamos a tratar, para ello tendremos que descargarnos los nombres de los alumnos junto con sus respectivos grupos obtenidos por la plataforma EV. También necesitaremos el DNI de los alumnos para ello lo obtendremos de la plataforma Sevius.

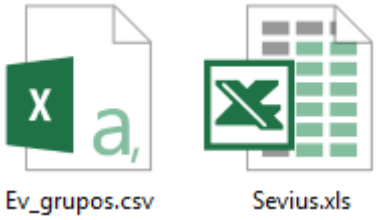


Figura 29. Ficheros de datos

Ev_grupos.csv será el archivo que contenga los datos de los grupos de los alumnos y tendrá el siguiente formato:

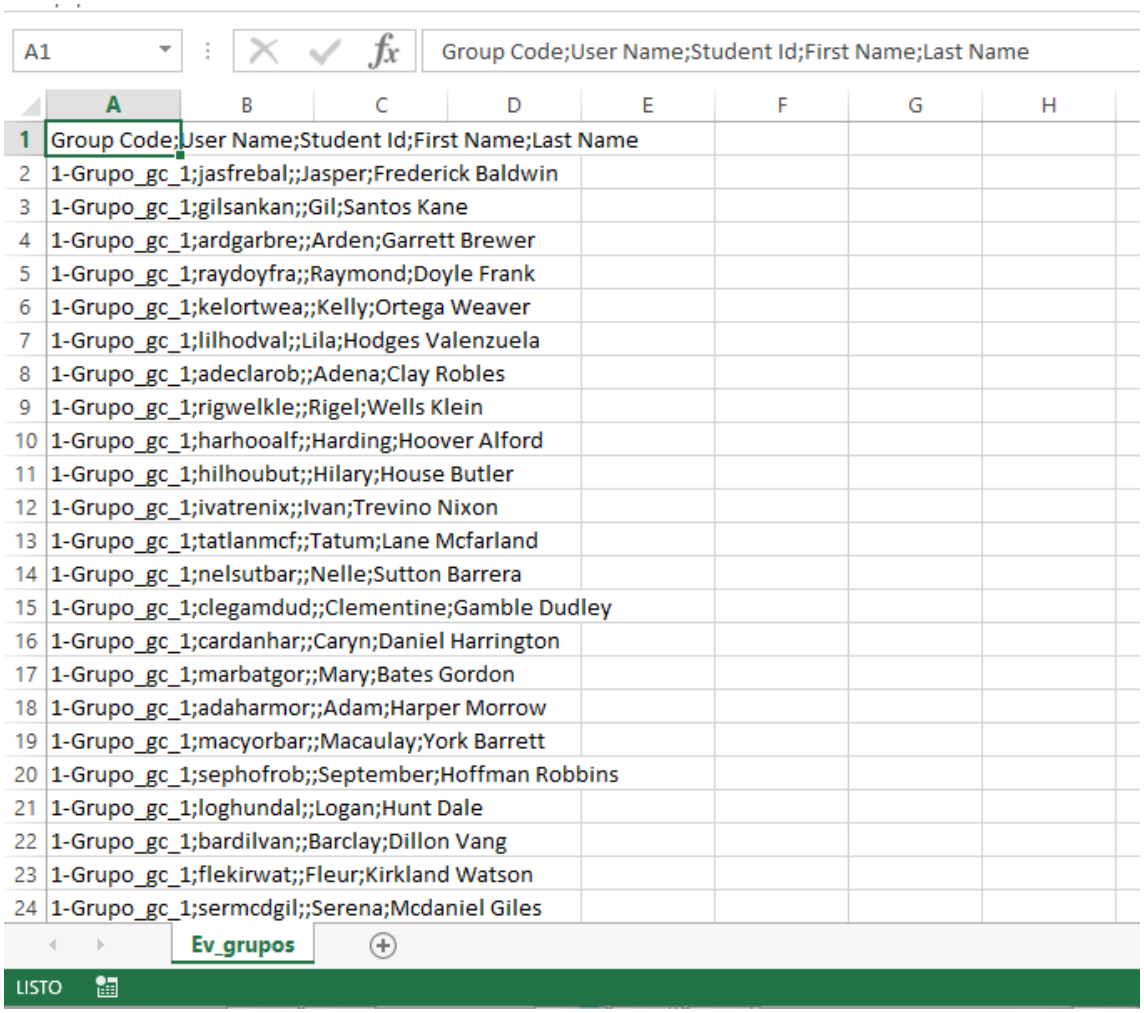
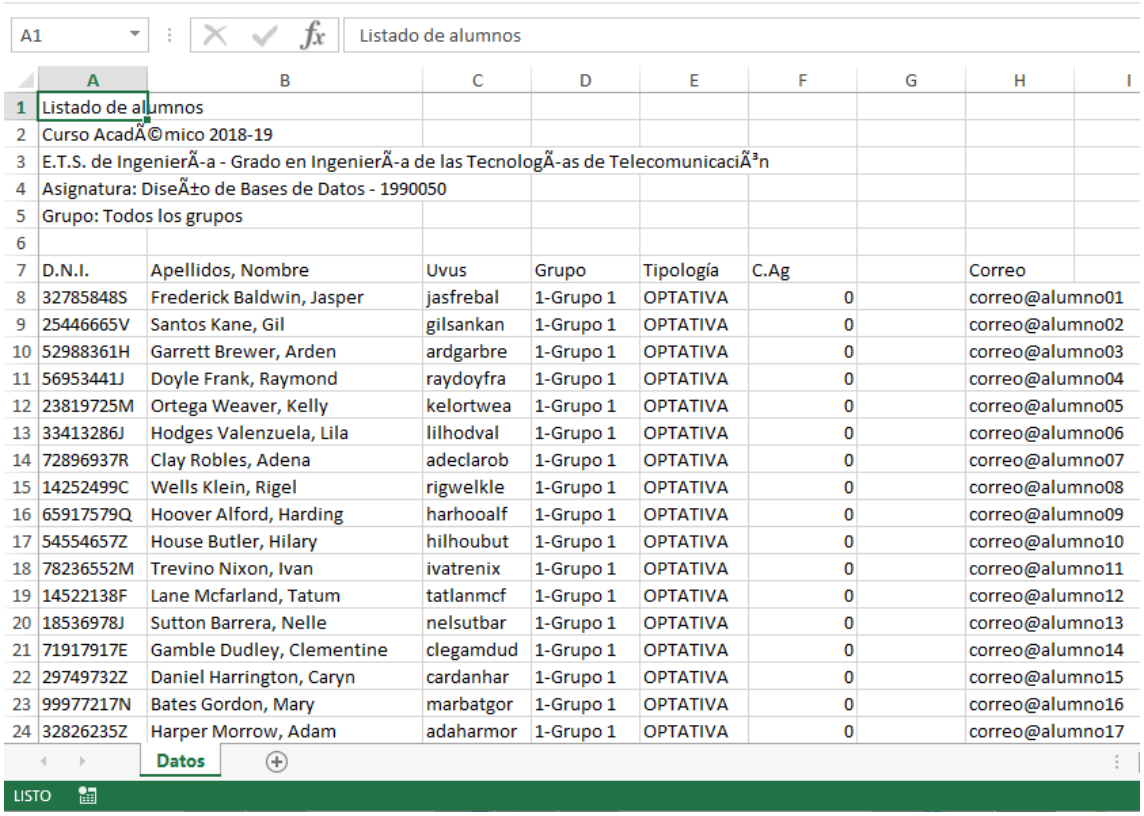


Figura 30. Formato de Ev_grupos.csv

Sevius.csv será el archivo que contenga los DNI de los grupos de los alumnos y tendrá el siguiente formato:



	A	B	C	D	E	F	G	H	I
1	Listado de alumnos								
2	Curso AcadÃ©mico 2018-19								
3	E.T.S. de IngenierÃ-a - Grado en IngenierÃ-a de las TecnologÃ-as de TelecomunicaciÃ³n								
4	Asignatura: DiseÃ±o de Bases de Datos - 1990050								
5	Grupo: Todos los grupos								
6									
7	D.N.I.	Apellidos, Nombre	Uvus	Grupo	Tipología	C.Ag		Correo	
8	32785848S	Frederick Baldwin, Jasper	jasfrebal	1-Grupo 1	OPTATIVA	0		correo@alumno01	
9	25446665V	Santos Kane, Gil	gilsankan	1-Grupo 1	OPTATIVA	0		correo@alumno02	
10	52988361H	Garrett Brewer, Arden	ardgarbre	1-Grupo 1	OPTATIVA	0		correo@alumno03	
11	56953441J	Doyle Frank, Raymond	raydoyfra	1-Grupo 1	OPTATIVA	0		correo@alumno04	
12	23819725M	Ortega Weaver, Kelly	kelortwea	1-Grupo 1	OPTATIVA	0		correo@alumno05	
13	33413286J	Hodges Valenzuela, Lila	lilhodval	1-Grupo 1	OPTATIVA	0		correo@alumno06	
14	72896937R	Clay Robles, Adena	adeclarob	1-Grupo 1	OPTATIVA	0		correo@alumno07	
15	14252499C	Wells Klein, Rigel	rigwelkle	1-Grupo 1	OPTATIVA	0		correo@alumno08	
16	65917579Q	Hoover Alford, Harding	harhooalf	1-Grupo 1	OPTATIVA	0		correo@alumno09	
17	54554657Z	House Butler, Hilary	hilhoubut	1-Grupo 1	OPTATIVA	0		correo@alumno10	
18	78236552M	Trevino Nixon, Ivan	ivatrenix	1-Grupo 1	OPTATIVA	0		correo@alumno11	
19	14522138F	Lane Mcfarland, Tatum	tatlanmcf	1-Grupo 1	OPTATIVA	0		correo@alumno12	
20	18536978J	Sutton Barrera, Nelle	nelsutbar	1-Grupo 1	OPTATIVA	0		correo@alumno13	
21	71917917E	Gamble Dudley, Clementine	clegamdud	1-Grupo 1	OPTATIVA	0		correo@alumno14	
22	29749732Z	Daniel Harrington, Caryn	cardanhar	1-Grupo 1	OPTATIVA	0		correo@alumno15	
23	99977217N	Bates Gordon, Mary	marbatgor	1-Grupo 1	OPTATIVA	0		correo@alumno16	
24	32826235Z	Harper Morrow, Adam	adaharmor	1-Grupo 1	OPTATIVA	0		correo@alumno17	

Figura 31. Formato de Sevius.xls

Una vez tengamos los ficheros, lo metemos en el mismo directorio que el Excel con la macro.

Figura 32. Archivos existentes en el directorio

Una vez tengamos todo preparado podemos abrir el Excel bajo el nombre de *prepara_csv.xls*. Aquí podemos ver información sobre el funcionamiento de la macro.

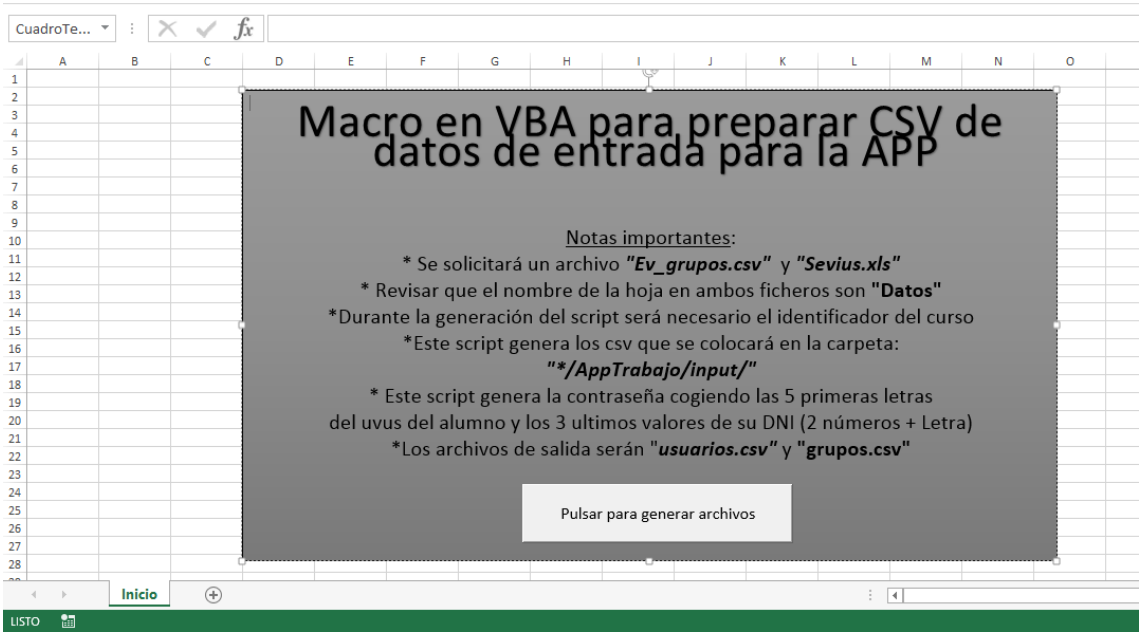


Figura 33. Página inicial de la macro

Si le damos al botón procederá la ejecución del código de la macro, durante esta ejecución te solicitará mediante una ventana emergente el identificador del curso (id_curso).

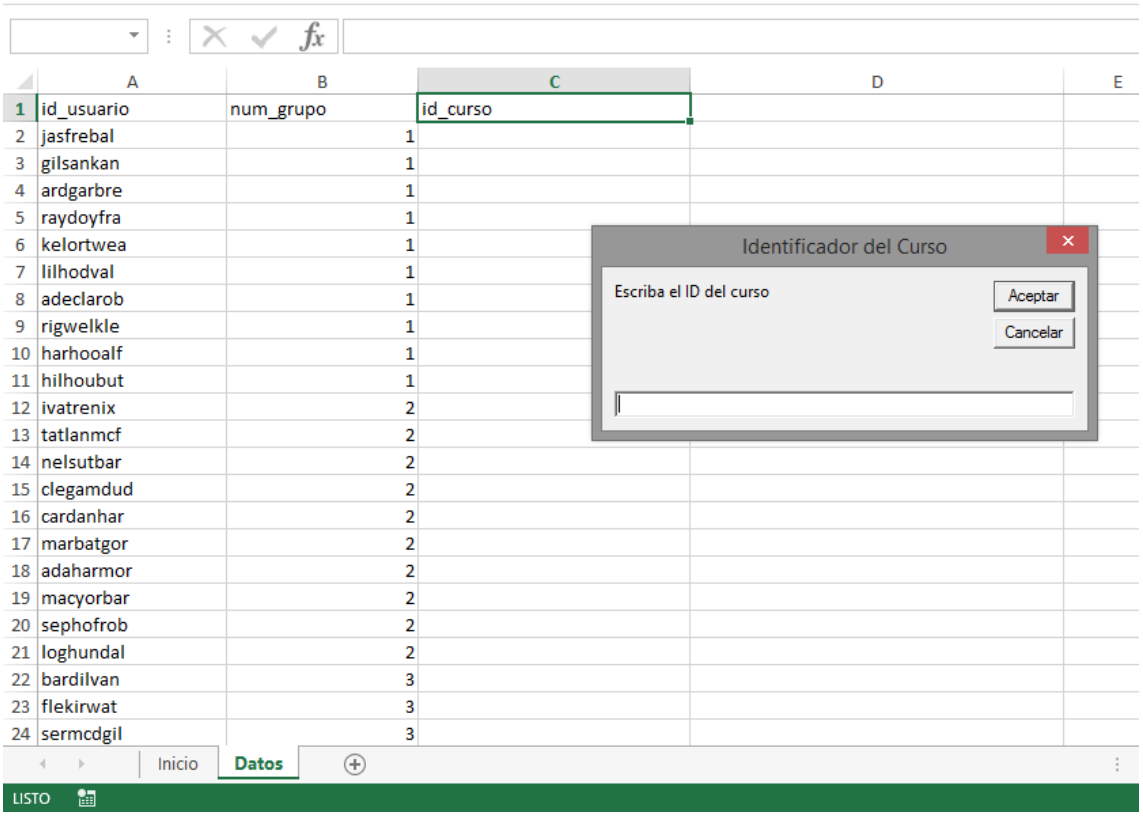


Figura 34. Solicitud de curso

Una vez acabe la ejecución, obtendremos 2 archivos que contendrá la información preparada para la importación.

Figura 35. Archivos generados

Este es el formato del archivo *usuarios.csv*:

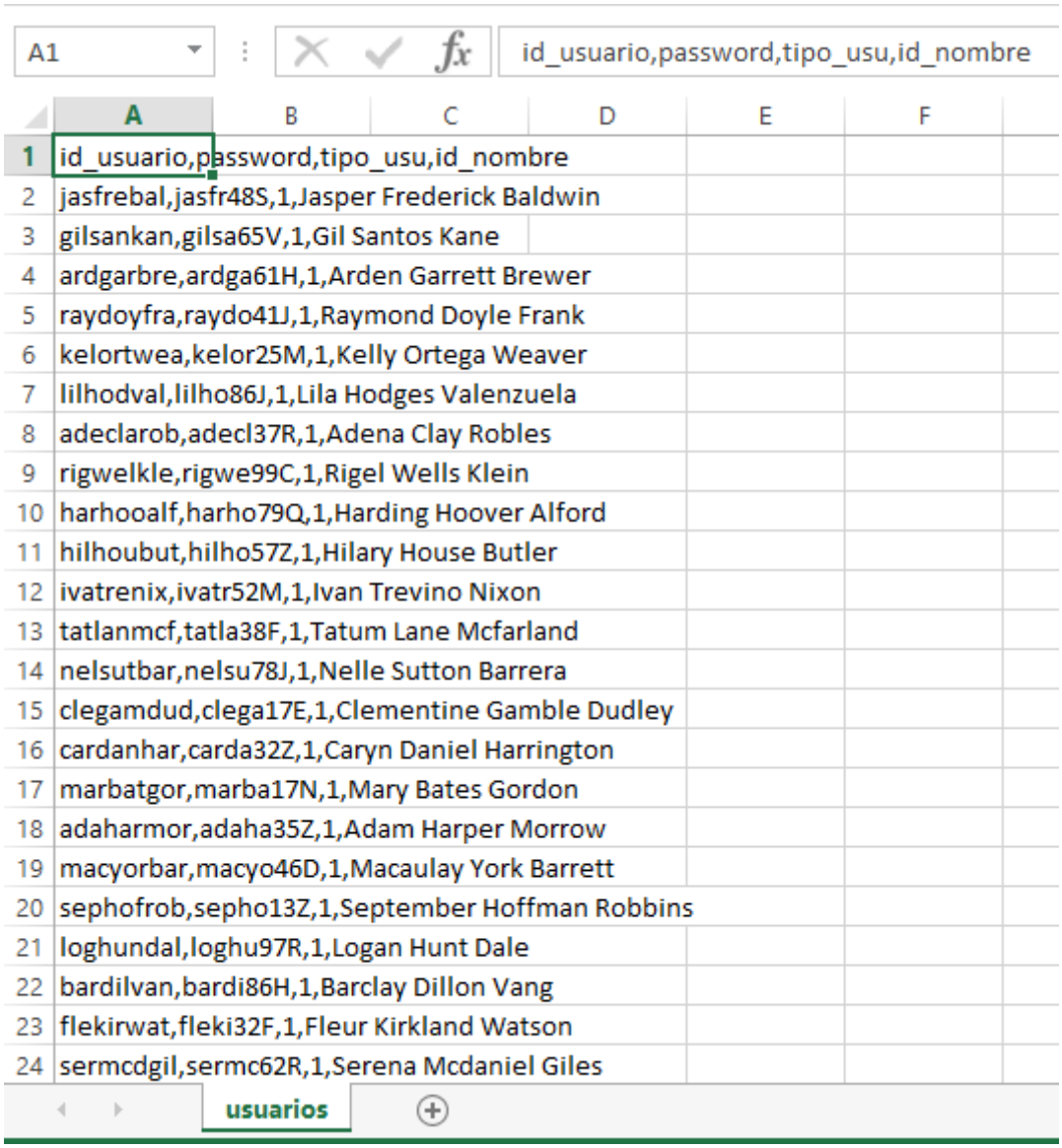


Figura 36. Formato usuarios.csv

Este es el formato del archivo *curso_x.csv*:

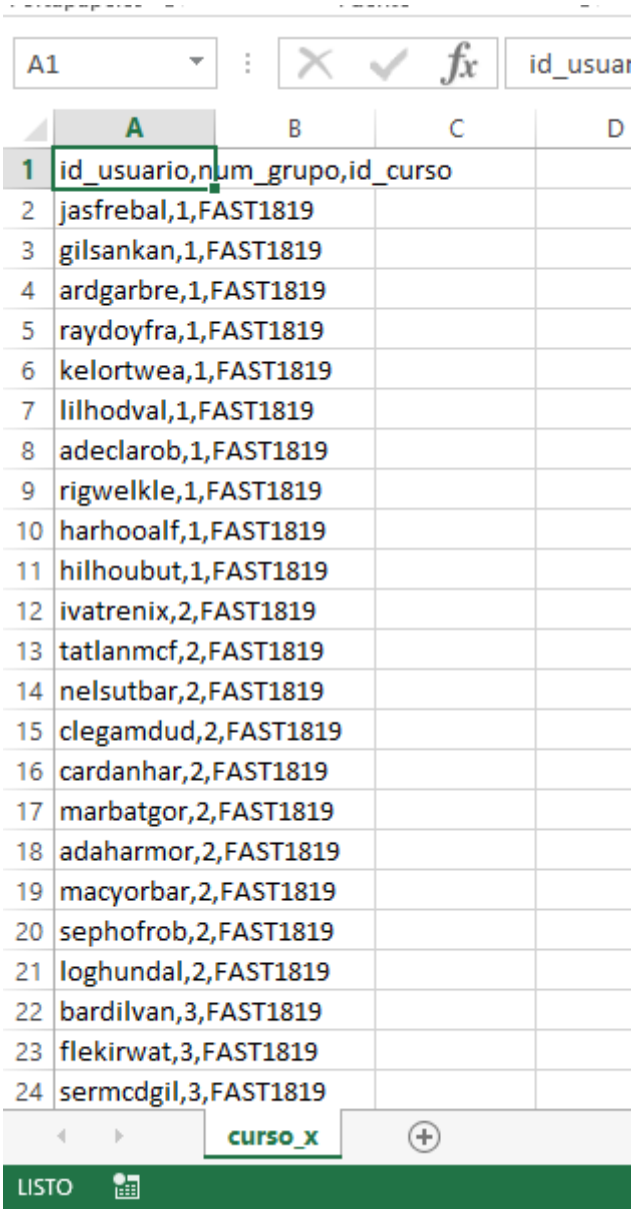


Figura 37. Formato curso_x.csv

Esto lo tendremos que realizar cada vez que queramos importar un nuevo curso, independientemente de que ya existan los alumnos ya que se actualizarán la información

8.2. Preparar entorno

Una vez tengamos listo los datos, pasamos a extraer la aplicación comprimido en un archivo .tar.gz.

Figura 38. Aplicación comprimida

Este archivo lo llevaremos a nuestro equipo con sistema Ubuntu y lo descomprimimos.

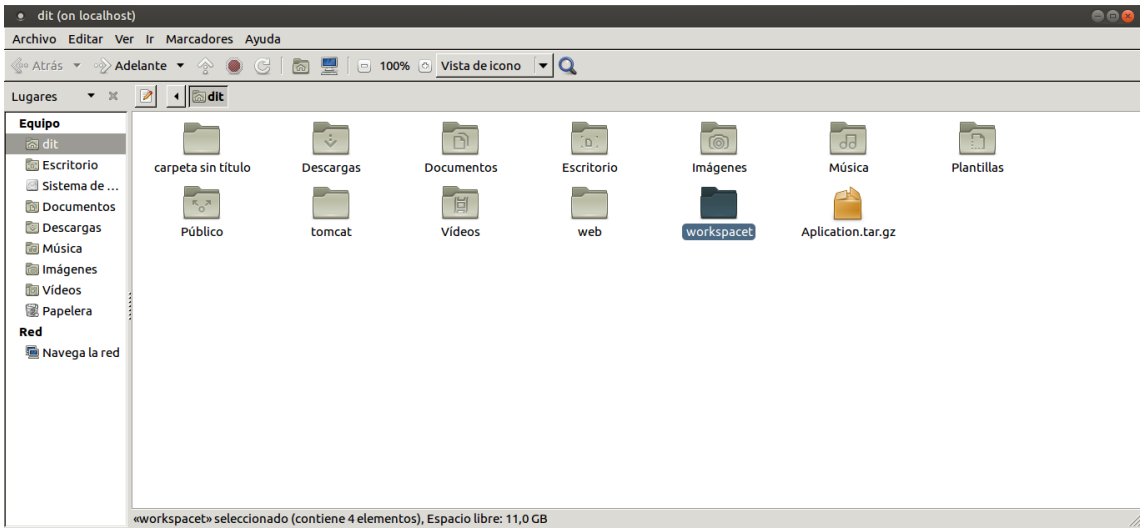


Figura 39. Directorio local

Una vez descomprimido accedemos dentro hasta localizar un shellscript nombrado *inicio.sh* y lo ejecutamos por consola de comandos.

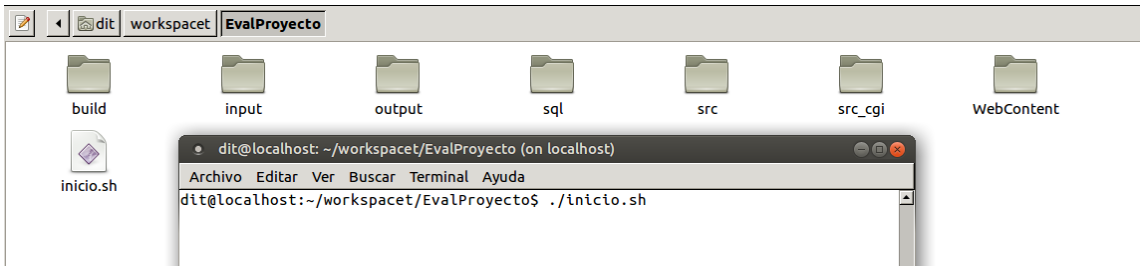


Figura 40. Ejecución 1 de inicio.sh

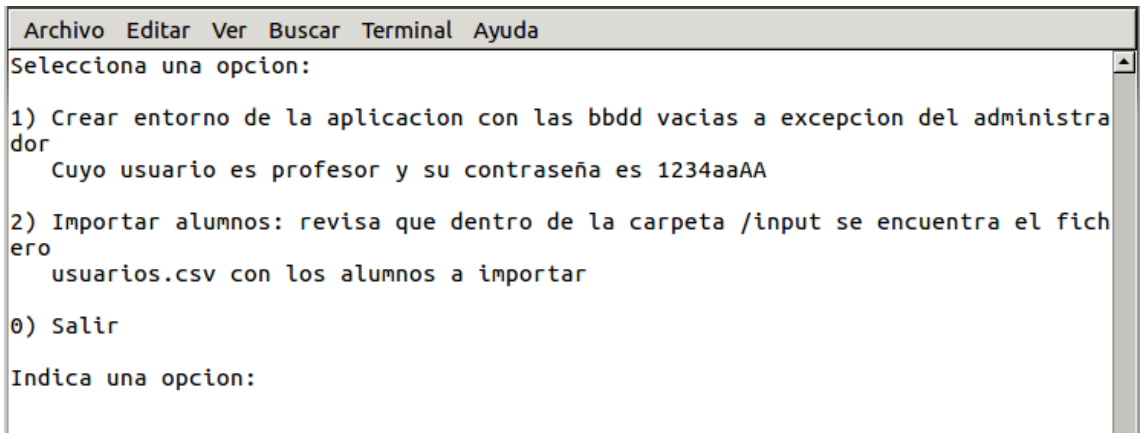


Figura 41. Ejecución 2 de inicio.sh

Como primer paso elegimos la opción 1 para dejar creadas las BBDD correspondientes,

```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
Has seleccionado la opcion 1
-------------------------------------

DROP TABLE
DROP TABLE
DROP TABLE
DROP TABLE
DROP TABLE
DROP TABLE
DROP TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
COPY 1
COPY 1
SET
 client_encoding
-----------------
 LATIN1
(1 row)


Entorno creado correctamente
-------------------------------------
Selecciona una opcion:

1) Crear entorno de la aplicacion con las bbdd vacias a excepcion del administra
dor
   Cuyo usuario es profesor y su contraseña es 1234aaAA
```

Figura 42. Ejecución Opción 1 de inicio.sh

Una vez ya configurado las BBDD (Esta opción solo deberemos ejecutarla la configuración inicial, ya que como se borra por defecto todas las tablas, borraría todo lo que tuviésemos) metemos nuestros archivos generados en la carpeta */input* para poder ejecutar la opción 2.

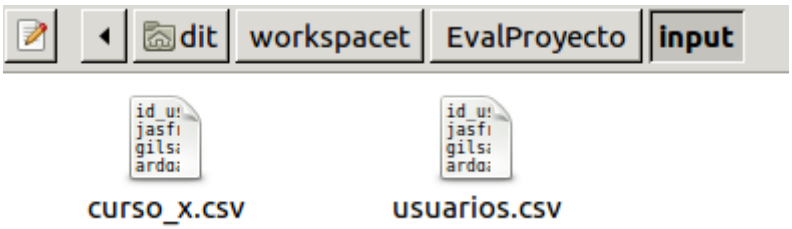


Figura 43. Archivos en /input

Una vez copiados los archivos, elegimos la opción 2.

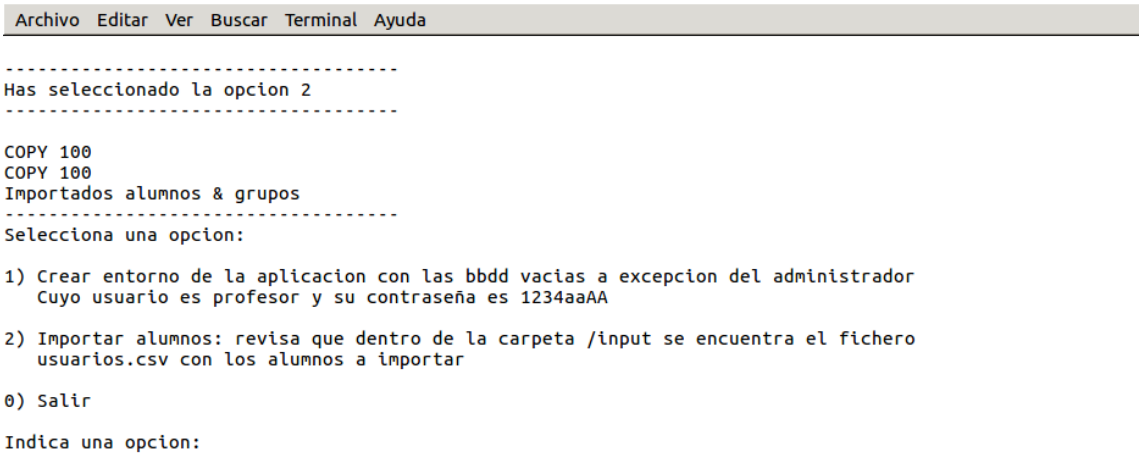


Figura 44. Ejecución Opción 2 de inicio.sh

8.3. Arrancar servidor

Finalizada la ejecución del script, abrimos el eclipse y arrancamos el servidor.

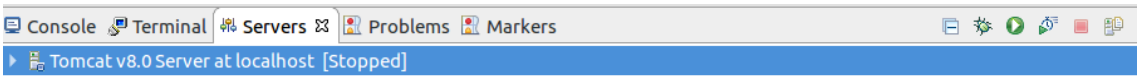


Figura 45. Servidor Parado

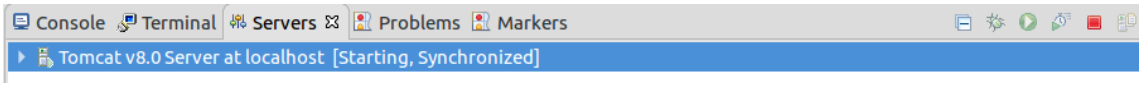


Figura 46. Servidor Iniciado correctamente

Y ya tendríamos nuestra aplicación en funcionamiento.

Figura 47. Aplicación en funcionamiento

9. ANEXOS B: MANUAL DE USO (PROFESOR)

Para el correcto uso de la aplicación se aportará este Anexo B, el cual detallará las distintas funcionalidades y servicios que ofrece.

9.1. Acceso

El acceso se hará mediante login de la web inicial, el administrador siempre tendrá el mismo usuario y contraseña inicial:

User: **Profesor**

Password: **1234aaAA**

Figura 48. Acceso a la Aplicación web

9.2. Creación y modificación de cursos

En esta página permite elegir el ID de curso y asignarle un nombre distintivo.

También será necesario una vez asignado el nombre al curso, asignarle unos criterios de evaluación con los que los alumnos puntuarán los cursos.

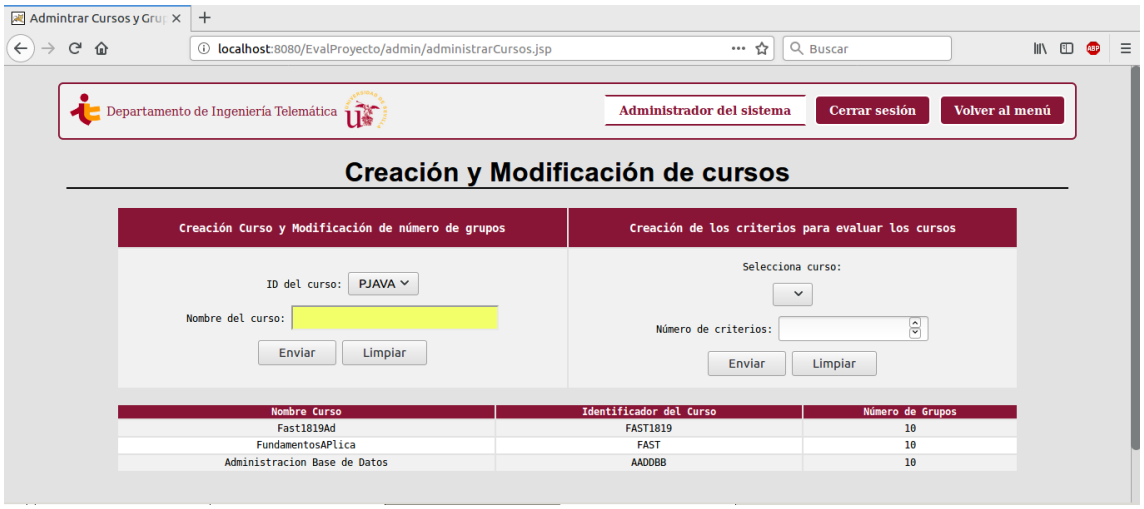


Figura 49. Creación y Modificación de Cursos I

Figura 50. Creación y Modificación de Cursos II

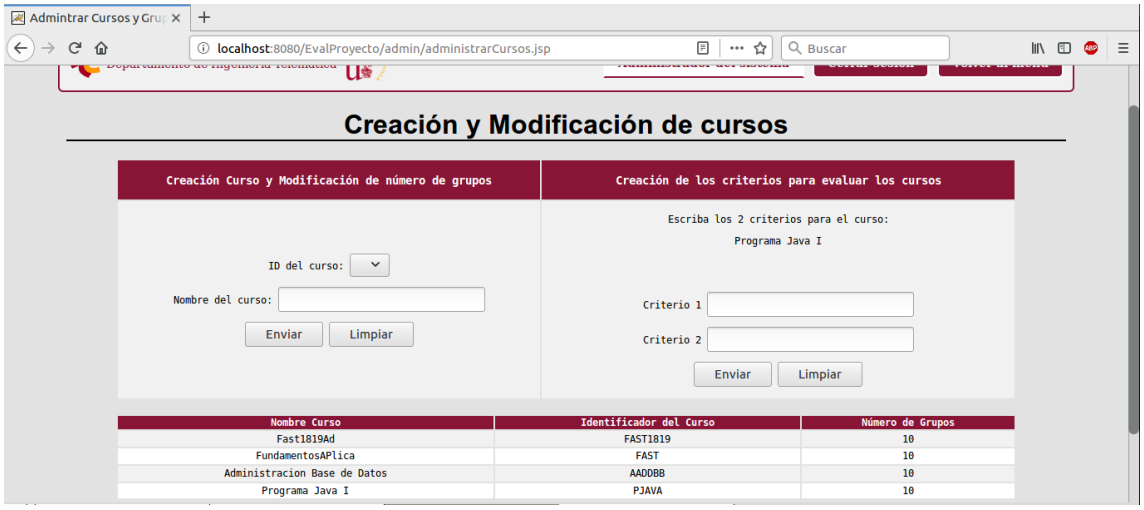


Figura 51. Creación y Modificación de Cursos III

9.3. Administrador de alumnos

En esta página permite administrar a los alumnos para poder cambiarlos a los distintos grupos de los distintos cursos existentes de la aplicación.

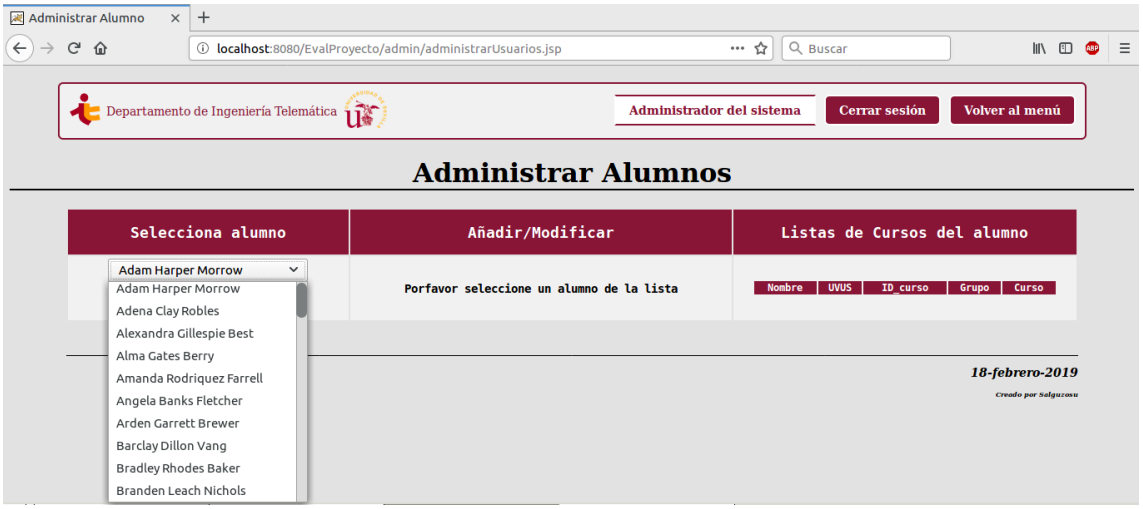


Figura 52. Administrar Alumnos I

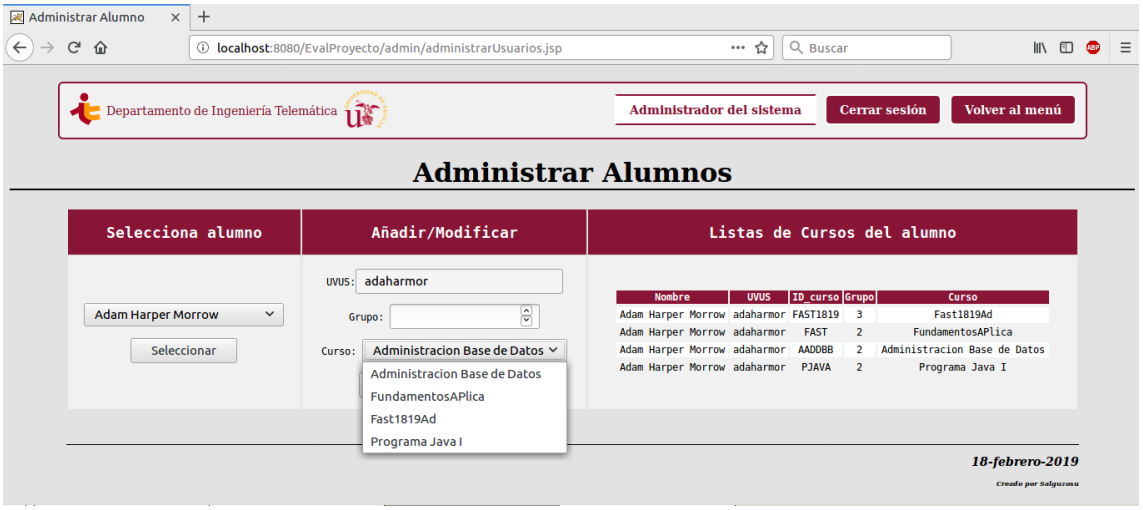


Figura 53. Administrar Alumnos II

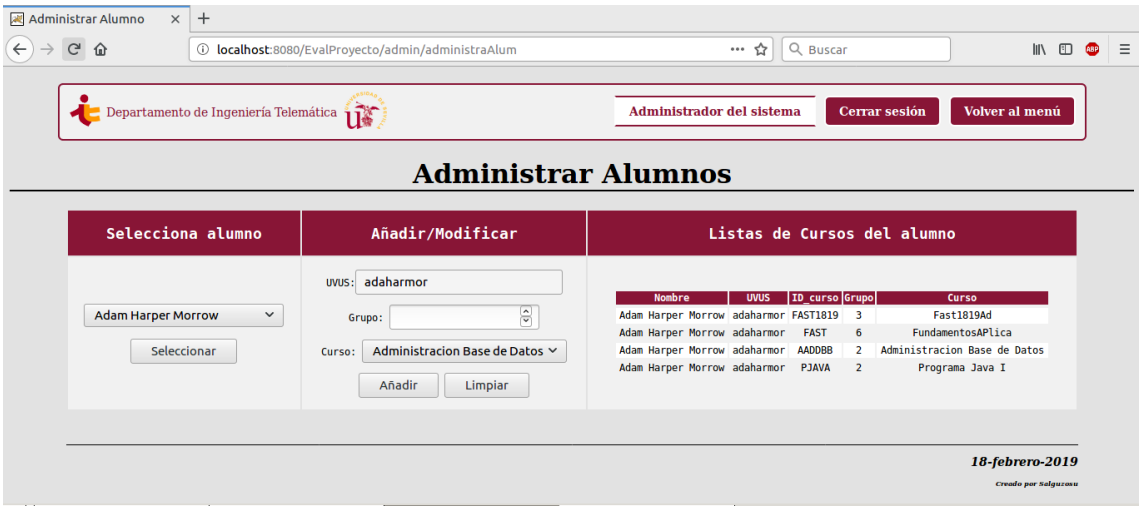


Figura 54. Administrar Alumnos III

9.4. Modificar contraseña de alumnos

En esta página permite administrar a los alumnos para poder cambiarlos a los distintos grupos de los distintos cursos existentes de la aplicación.

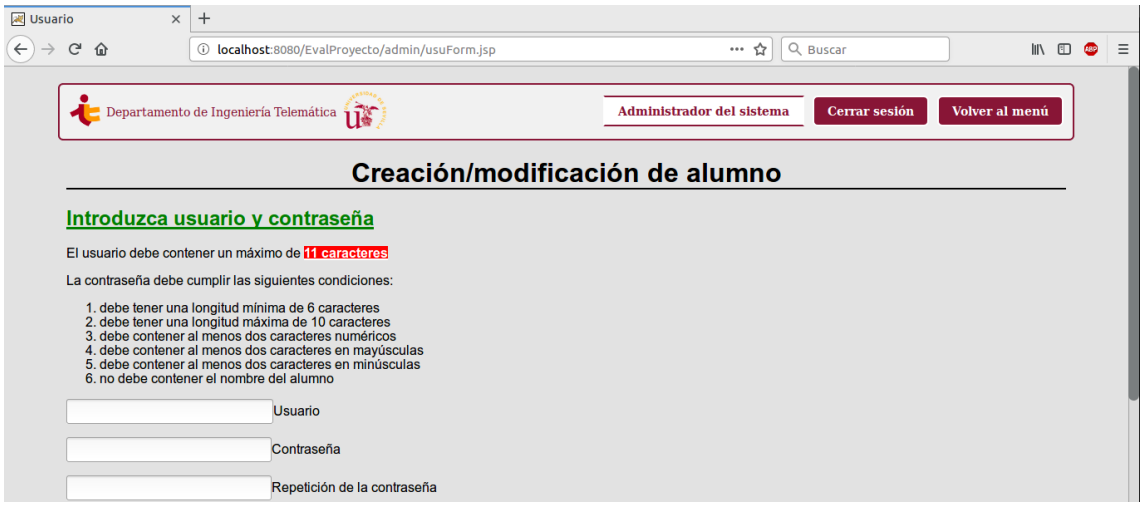


Figura 55. Modificación contraseña I

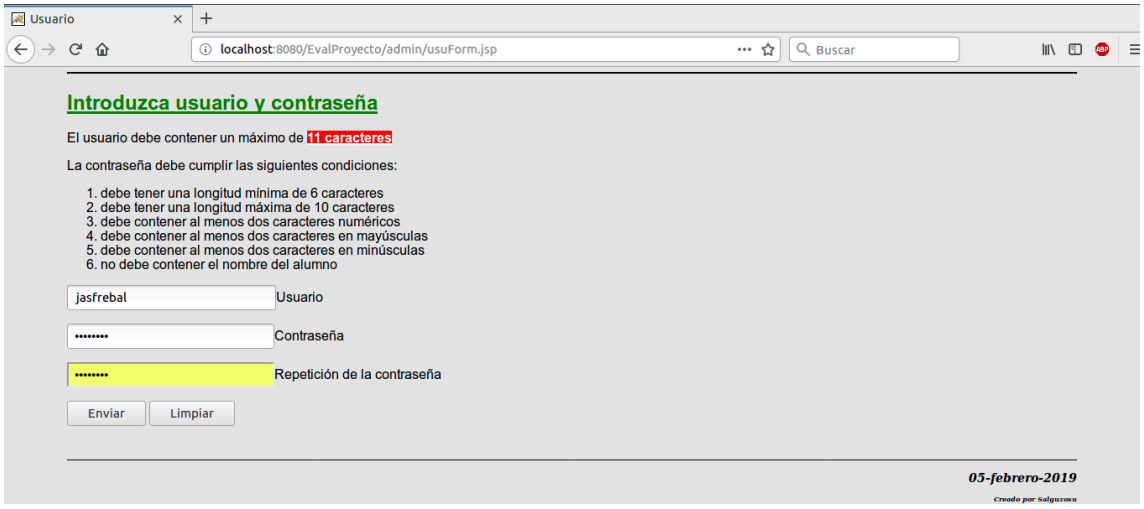
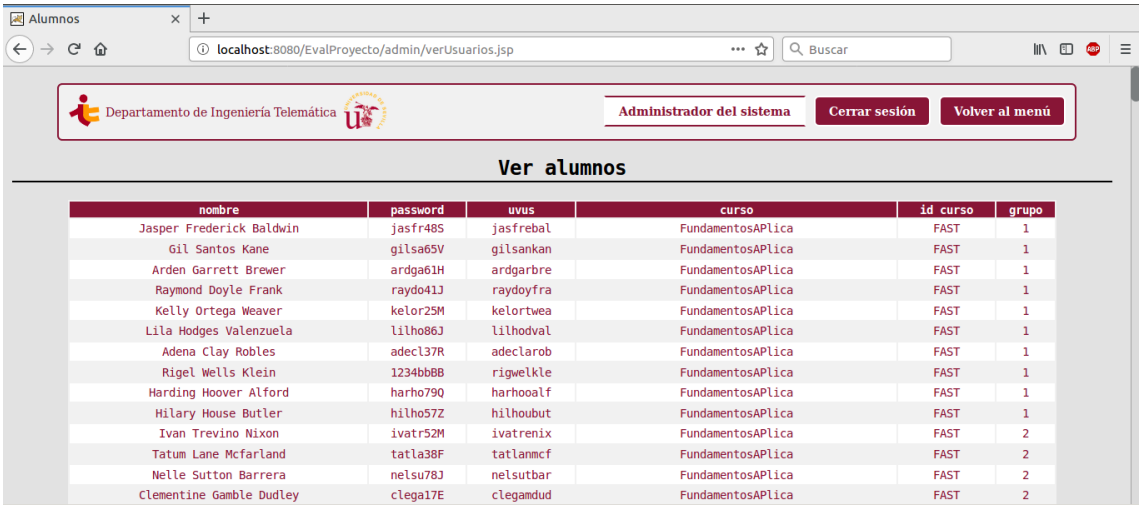


Figura 56. Modificación contraseña II

9.5. Ver todos los alumnos

En esta página permite visualizar toda la información relativa a los alumnos.



nombre	password	uvus	curso	id curso	grupo
Jasper Frederick Baldwin	jasfr48S	jasfrebal	FundamentosAPlica	FAST	1
Gil Santos Kane	gilsa65V	gilsankan	FundamentosAPlica	FAST	1
Arden Garrett Brewer	ardga61H	ardgarbre	FundamentosAPlica	FAST	1
Raymond Doyle Frank	raydo41J	raydoyfra	FundamentosAPlica	FAST	1
Kelly Ortega Weaver	kelor25M	kelortwea	FundamentosAPlica	FAST	1
Lila Hodges Valenzuela	lilho86J	lilhodval	FundamentosAPlica	FAST	1
Adena Clay Robles	adecl37R	adeclarob	FundamentosAPlica	FAST	1
Rigel Wells Klein	1234bbBB	rigwelkle	FundamentosAPlica	FAST	1
Harding Hoover Alford	harho79Q	harhooalf	FundamentosAPlica	FAST	1
Hilary House Butler	hilho57Z	hilhoubut	FundamentosAPlica	FAST	1
Ivan Trevino Nixon	ivatr52M	ivatrenix	FundamentosAPlica	FAST	2
Tatum Lane Mcfarland	tatla38F	tatlanmcf	FundamentosAPlica	FAST	2
Nelle Sutton Barrera	nelsu78J	nelsutbar	FundamentosAPlica	FAST	2
Clementine Gamble Dudley	clega17E	clegamdud	FundamentosAPlica	FAST	2

Figura 57. Ver alumnos

9.6. Ver notas de todos los alumnos por curso

En esta página permite visualizar toda las notas realizadas por los alumnos.

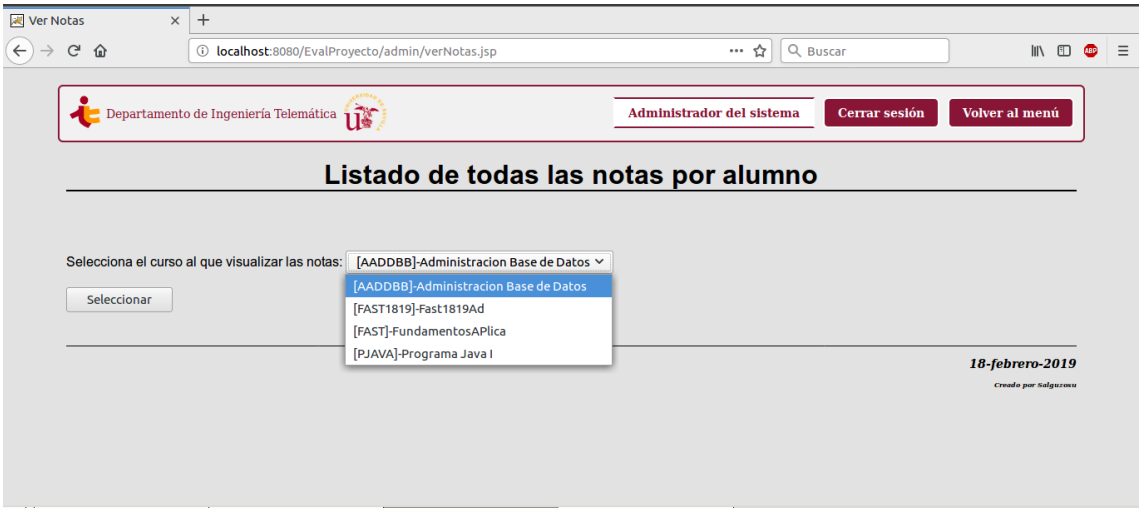


Figura 58. Ver Notas por Curso

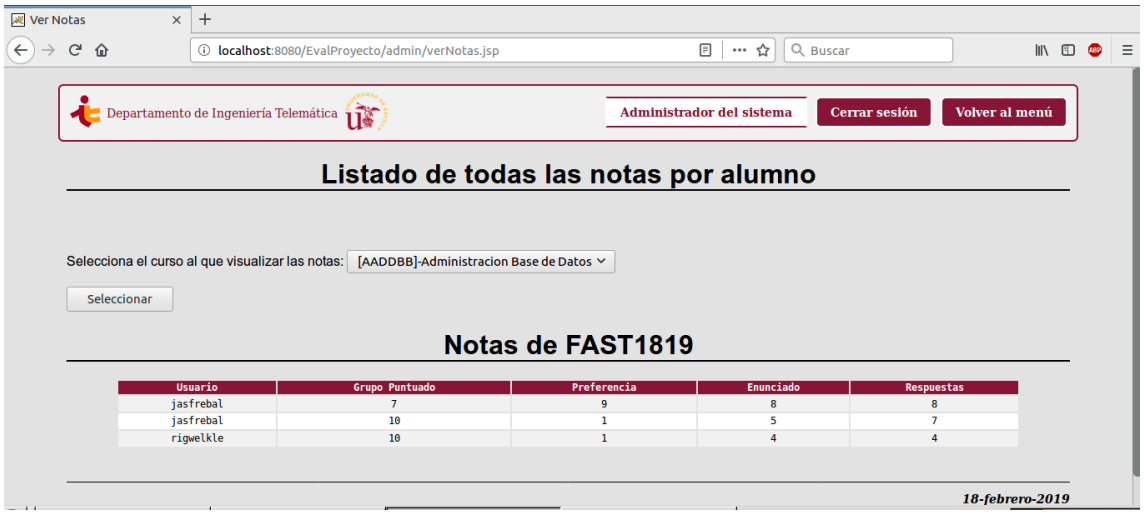


Figura 59. Ver Notas por Curso II

10. ANEXO C: MANUAL DE USO (ALUMNO)

Para el correcto uso de la aplicación se aportará este Anexo C, el cual detallará las distintas funcionalidades y servicios que ofrece.

10.1. Acceso

El acceso se hará mediante login de la web inicial, el administrador siempre tendrá el mismo usuario y contraseña inicial:

User: UVUS

Password: 5 primeras letras de tu UVUS en minúsculas + 2 últimos dígitos del DNI + Letra DNI en Mayúsculas

Figura 60. Acceso a la web

10.2. Cambiar contraseña

En esta página podrá el alumno cambiar la contraseña inicial por otra siempre que cumpla los requisitos de seguridad indicados:

Figura 61. Cambio de Contraseña I

Figura 62. Cambio de Contraseña II

Figura 63. Cambio de Contraseña III

Figura 64. Modificación de Contraseña IV

10.3. Puntuar

En esta página podrá el alumno puntuar, siempre que tenga permisos, a los distintos grupos de las distintas asignaturas según los criterios establecidos por el profesor.

Figura 65. Puntuar I

Figura 66. Puntuar II

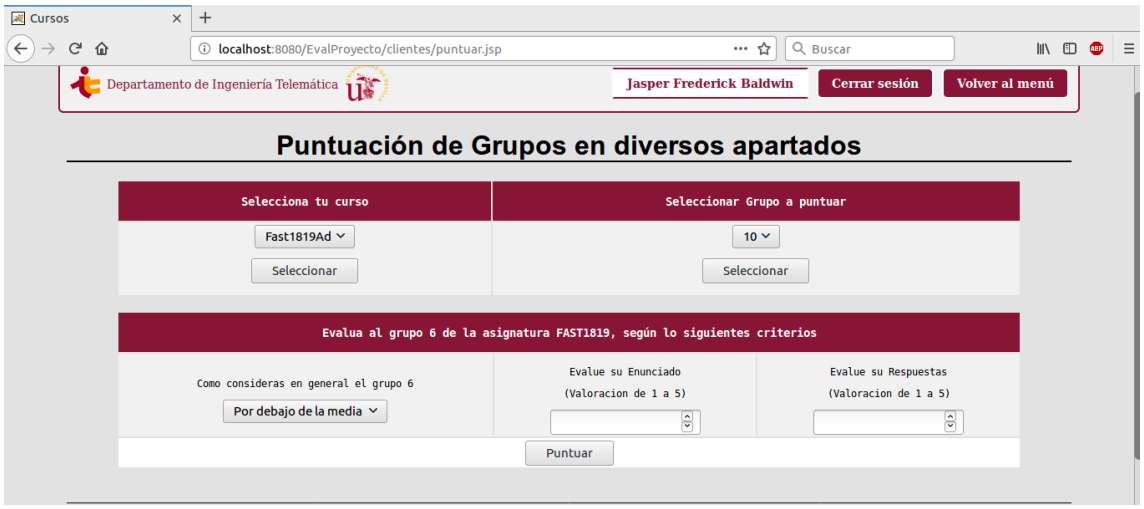


Figura 67. Puntuar III

11. ANEXO D: CÓDIGO

Es este apartado se aportará los distintos códigos usados:

11.1 /admin/menú.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
	pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
	<meta charset="UTF-8">
	<title>Menú Profesor</title>
<link href="css/estilos.css" rel="stylesheet" type="text/css" />
</head>
<body>
<jsp:include page="../cabecera.jsp" />
	<div id="contenedor">
		<h1>Menú Profesor</h1>
		<ul>

			<li><a href="admin/administrarCursos.jsp">Crear & Administrar
Cursos</a></li>
			<br>
			<li><a href="admin/administrarUsuarios.jsp">Administrar
Alumnos</a></li>
			<br>
			<li><a href="admin/usuForm.jsp">Crear/modificar alumnos</a></li>
			<br>
			<li><a href="admin/permisos.jsp">Permisos</a></li>
			<br>
			<li><a href="admin/verUsuarios.jsp">Lista de Alumnos
Matriculados</a></li>
			<br>
			<li><a href="admin/verNotas.jsp">Ver notas</a></li>

		</ul>
	</div>
<%@include file="../pie.jsp"%>

</body>
</html>
```

11.2 /admin/ administrarCursos.jsp

```
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.PreparedStatement"%>
<%@page import="java.sql.Connection"%>
<%@page import="javax.sql.DataSource"%>
<%@page import="javax.naming.InitialContext"%>
<%@page import="java.sql.SQLException"%>
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-
8"%>

<jsp:useBean id="ds" type="javax.sql.DataSource" scope="application" />
<!DOCTYPE html>
<html lang="es">
<head>
	<meta charset="UTF-8">
	<title>Admintrar Cursos y Grupos</title>
	<link rel="stylesheet" href="../css/estilos.css">
</head>
<body>
<jsp:include page="../cabecera.jsp" />
<div id="contenedor">
<h1>Creación y Modificación de cursos</h1>
<table>
<tr>
<th><h3>Creación Curso y Modificación de número de grupos</h3></th>
<th><h3>Creación de los criterios para evaluar los cursos</h3></th>
</tr>
<tr>
<td>
<form action="modificarCurso" id="formUsu" name="formUsu" method="Post" onsubmit=
"return validaFormulCurso()">

<p><label>ID del curso: <select name="id" id="id" required>
<%

try {
	Connection conn = ds.getConnection();
	String sql = "select distinct id_curso from curso_x where id_curso not in (select distinct
id_curso from cursosgrupo)";
	PreparedStatement st = conn.prepareStatement(sql);
	ResultSet rs = st.executeQuery();
	while (rs.next()){
		String id_curso = rs.getString(1);
	%>
	<option value="<%=id_curso%>"><%=id_curso%></option>
	<%
	}
	rs.close();
	st.close();
	conn.close();
	}catch (SQLException e){
		out.println("Excepción SQL Exception: "+ e.getMessage());
		e.printStackTrace();
	}
%></select ></label></p>
<p><label>Nombre del curso: <input type="text" id="curso" name="curso" maxlength=50
required/></label></p>
<p><input type="submit" name="enviar" value="Enviar" />
```

```
<input type="reset" name="borrar" value="Limpiar" /></p>
</form>
</td>
<td>

<%if(request.getParameter("cursoid")!=null) {

	int crit_max = Integer.parseInt(request.getParameter("criteri"));
	int i = 0;
	String cursoidnom="";
	try {
		Connection conn = ds.getConnection();

		String sql = "SELECT id_nombrecurso FROM cursosgrupo where id_curso = ?";
		PreparedStatement st = conn.prepareStatement(sql);
		st.setString(1, request.getParameter("cursoid"));
		ResultSet rs = st.executeQuery();
		while (rs.next()){
			cursoidnom = rs.getString(1);
		}
		rs.close();
		st.close();
		conn.close();
		}catch (SQLException e){
			out.println("Excepción SQL Exception: "+ e.getMessage());
			e.printStackTrace();
		}

%>
<form action="creaCriterios" method="Post" >
<p><label>Escriba los <%=crit_max%> criterios para el curso: </label></p>
<p><%=cursoidnom%></p>
<input type="text" id="cursoidn" name="cursoidn"
value=<%=request.getParameter("cursoid")%> readonly style="visibility:hidden"/>
<input type="number" id="criterioID" name="criterioID" value=<%=crit_max%> readonly
style="visibility:hidden"/>
<%while(i<crit_max){ %>
<p><label>Criterio <%=i+1 %> <input type="text" id="textocriterio_<%=i+1%>"
name="textocriterio_<%=i+1%>" maxlength=20 required/></label></p>


<% i++; } %>
		<p><label><input type="submit" name="enviar" value="Enviar" />
		<input type="reset" name="borrar" value="Limpiar" /></label></p>
</form>
<%} else { %>
<p><label>Selecciona curso:</label></p>



<form action="administrarCursos.jsp" method="Post">
<select name="cursoid">
<%
	try {
		Connection conn = ds.getConnection();

		String sql = "SELECT distinct a.id_nombrecurso,a.id_curso FROM cursosgrupo
a where a.id_curso not in (select id_curso from criterio_x) order by id_nombrecurso";
		PreparedStatement st = conn.prepareStatement(sql);
		ResultSet rs = st.executeQuery();
```

```
                while (rs.next()){
                        String id_curso = rs.getString(2);
                        String id_nombrecurso = rs.getString(1);

                %>
                <option value="<%=id_curso%>"><%=id_curso%></option>
                <%
                }
                        rs.close();
                        st.close();
                        conn.close();
                        }catch (SQLException e){
                                out.println("Excepción SQL Exception: "+ e.getMessage());
                                e.printStackTrace();
                        }
                %>
                </select>
                <p><label>Número de criterios: <input type="number" id="criteri" name="criteri"
min="1" max="10" required/></label></p>
                <p><label><input type="submit" name="enviar" value="Enviar" />
                <input type="reset" name="borrar" value="Limpiar" /></label></p>

</form>
<%} %>
</td>
</tr>
</table>




<%
        try {
                Connection conn = ds.getConnection();

                String sql = "SELECT * FROM cursosgrupo";
                PreparedStatement st = conn.prepareStatement(sql);
                ResultSet rs = st.executeQuery();
%>
<br>
                <table id="tablaCursosGrupos">

                <tr>
                <th>Nombre Curso</th>
                <th>Identificador del Curso</th>
                <th>Número de Grupos</th>
                </tr>
                <%
                while (rs.next()){
                        String curso = rs.getString(1);
                        Integer grupos = rs.getInt(2);
                        String id = rs.getString(3);

                %>
                <tr>
                        <td><%=curso%></td>
                        <td><%=id%></td>
```

```
                    <td><%=grupos%></td>
                </tr>
                <%
                }
                    rs.close();
                    st.close();
                    conn.close();
                    }catch (SQLException e){
                        out.println("Excepción SQL Exception: "+ e.getMessage());
                        e.printStackTrace();
                    }
                %>
                </table>
</div>
<jsp:include page="../pie.jsp" />

</body>
</html>
```

11.3 /admin/ administrarUsuarios.jsp

```
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.PreparedStatement"%>
<%@page import="java.sql.Connection"%>
<%@page import="javax.sql.DataSource"%>
<%@page import="javax.naming.InitialContext"%>
<%@page import="java.sql.SQLException"%>
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-
8"%>

<jsp:useBean id="ds" type="javax.sql.DataSource" scope="application" />
<!DOCTYPE html>
<html lang="es">
<head>
	<meta charset="UTF-8">
	<title>Administrar Alumno</title>
	<link rel="stylesheet" href="../css/estilos.css">
</head>
<body>
<jsp:include page="../cabecera.jsp" />
<h1>Administrar Alumnos</h1>
<div id="contenedor">
</div>
<table id="AdminAlum">
<tr>
<th><h2>Selecciona alumno</h2></th>
<th><h2>Añadir/Modificar</h2></th>
<th><h2>Listas de Cursos del alumno</h2></th>
</tr>
<tr>
<td id="ElecAlumno">
<form action="administrarUsuarios.jsp" method="Post">
<select name="alumno">
<%
	try {
		Connection conn = ds.getConnection();

		String sql = "SELECT * FROM usuarios where tipo_usu=1 order by id_nombre";
		PreparedStatement st = conn.prepareStatement(sql);
		ResultSet rs = st.executeQuery();
		while (rs.next()){
			String id_usuario = rs.getString(1);
			String password = rs.getString(2);
			Integer tipo_usu = rs.getInt(3);
			String id_nombre = rs.getString(4);

		%>
		<option value="<%=id_usuario%>"><%=id_nombre%></option>
		<%
		}
			rs.close();
			st.close();
			conn.close();
			}catch (SQLException e){
				out.println("Excepción SQL Exception: "+ e.getMessage());
				e.printStackTrace();
			}
		%>
		</select>
```

```
                    <br><br>
                    <input type="submit" name="enviar" value="Seleccionar">
                    </form>
                    </td>
                    <td id="ModificaAlum" >
                    <%
                    if(request.getParameter("alumno")!=null){
                    %>
                    <form action="administraAlum" id="formUsu" name="formUsu" method="Post"
onsubmit= "return validaFormulCurso()">

                    <p><label>UVUS:<input type="text" id="alumno" name="alumno"
value=<%=request.getParameter("alumno")%> readonly/></label></p>
                    <p><label>Grupo: <input type="number" id="grupo" name="grupo" value=""
min="0" max="20" required/></label></p>
                    <p><label>Curso:
                    <select form="formUsu" id="curso" name="curso" >
                    <%
          try {
                    Connection conn = ds.getConnection();

                    String sql = "SELECT id_curso,id_nombrecurso FROM cursosgrupo order by
id_curso";
                    PreparedStatement st = conn.prepareStatement(sql);
                    ResultSet rs = st.executeQuery();
                    while (rs.next()){
                              String curso = rs.getString(1);
                              String nombrecurso = rs.getString(2);

                    %>
                     <option value=<%=curso%>><%=nombrecurso%></option>
                     <%
                    }
                              rs.close();
                              st.close();
                              conn.close();
                              }catch (SQLException e){
                                        out.println("Excepción SQL Exception: "+ e.getMessage());
                                        e.printStackTrace();
                              }
                    %>
                    </select></label></p>
<p><input type="submit" name="enviar" value="Añadir" />

  <input type="reset" name="borrar" value="Limpiar" /></p>

</form>
<%} else {
          %>
<h3>Porfavor seleccione un alumno de la lista</h3>
<%} %>
</td>

<td>
<table>


          <tr><th>Nombre</th><th>UVUS</th><th>ID_curso</th><th>Grupo</th><th>Curso</th>
</tr>
                    <tr>
```

```
<%
	try {
		Connection conn = ds.getConnection();

		String sql = "SELECT b.id_nombre,a.id_usuario,c.id_nombrecurso,
a.num_grupo, c.id_curso FROM curso_x a, usuarios b, cursosgrupo c where a.id_usuario=?
and b.id_usuario=a.id_usuario and a.id_curso=c.id_curso";
		PreparedStatement st = conn.prepareStatement(sql);
		st.setString(1, request.getParameter("alumno"));
		ResultSet rs = st.executeQuery();
		while (rs.next()){
			String id_usuario = rs.getString(1);
			Integer grupo = rs.getInt(4);
			String id_nombrecurso = rs.getString(3);
			String id_nombre = rs.getString(2);
			String id_curso = rs.getString(5);

		%>
		<tr>
			<td><%=id_usuario%></td>
			<td><%=id_nombre%></td>
			<td><%=id_curso%></td>
			<td><%=grupo%></td>
			<td><%=id_nombrecurso%></td>
		</tr>
		<%
		}
			rs.close();
			st.close();
			conn.close();
			}catch (SQLException e){
				out.println("Excepción SQL Exception: "+ e.getMessage());
				e.printStackTrace();
			}
		%>
</tr>
		</table>
</td>
</tr>
</table>
<div id="contenedor">

</div>
<jsp:include page="../pie.jsp" />

</body>
</html>
```

11.4 /admin/ permisos.jsp

```
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.PreparedStatement"%>
<%@page import="java.sql.Connection"%>
<%@page import="javax.sql.DataSource"%>
<%@page import="javax.naming.InitialContext"%>
<%@page import="java.sql.SQLException"%>
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-
8"%>

	<jsp:useBean id="ds" type="javax.sql.DataSource" scope="application" />
<jsp:useBean id="usuario" type="fast1718.trabajo.Usuario" scope="session"/>
<!DOCTYPE html>
<html lang="es">
<head>
	<meta charset="UTF-8">
	<title>Alumnos</title>
	<link rel="stylesheet" href="../css/estilos.css">

</head>
<body>
<jsp:include page="../cabecera.jsp" />
<div class="contenedor">
		<h1>Ver Permisos</h1>


<form action="permisos.jsp" method="Post">
<p> Selecciona el curso al que administrar los permisos: <select name=curso>

<%
	try {
		Connection conn = ds.getConnection();
		String sql = "SELECT id_nombrecurso, id_curso FROM cursosgrupo order by
id_nombrecurso";
		PreparedStatement st = conn.prepareStatement(sql);
		ResultSet rs = st.executeQuery();
		while (rs.next()){
			String id_nombrecurso = rs.getString(1);
			String id_curso = rs.getString(2);

		%>
		<option value="<%=id_curso%>"><%=id_nombrecurso%></option>
		<%
		}
			rs.close();
			st.close();
			conn.close();
			}catch (SQLException e){
				out.println("Excepción SQL Exception: "+ e.getMessage());
				e.printStackTrace();
			}
		%>

</select>
</p>
 <p><input type="submit" name="enviar" value="Seleccionar" />
</form>
<%
if(request.getParameter("curso")!=null){
```

```
		Integer permite_grupo = 0;
		Integer permite_puntua = 0;
		try {
			Connection conn = ds.getConnection();
			String sql = "SELECT permite_grupo, permite_puntua FROM permisos where
id_curso = ?";
			PreparedStatement st = conn.prepareStatement(sql);
			st.setString(1, request.getParameter("curso"));
			ResultSet rs = st.executeQuery();
			while (rs.next()){
				permite_grupo = rs.getInt(1);
				permite_puntua = rs.getInt(2);
			}
			rs.close();
			st.close();
			conn.close();
			}catch (SQLException e){
				out.println("Excepción SQL Exception: "+ e.getMessage());
				e.printStackTrace();
			}
		%>

			<form action="modificaPermisos" method="Post">

			<table id="ElecPerm">
			<tr>

			<th><h2>Permisos para poder puntuar a los grupos del curso
<%=request.getParameter("curso")%></h2></th>
			</tr>
			<tr>
			<td>

		<h3><%if(permite_puntua==1){%>Habilitado<%}else{%>Deshabilitado<%}%></h3>
			</td>
			<tr>
			<td>
			<p>¿Deseas cambiar estos permisos?: <br>
				<input type="radio" name="permisopuntua" value="0"
checked>Deshabilitar permisos para puntuar grupos<br>
				<input type="radio" name="permisopuntua" value="1">Habilitar
permisos para puntuar grupos<br>
				<input type="text" name="idcurso"
value=<%=request.getParameter("curso")%> readonly style="display:none;">
			</p>
			</td>
			</tr>
			<tr>
			<td colspan="2">
				<p><input type="submit" name="enviar" value="Cambiar" />
				</p>
			</td>
			</tr>
			</table>
			</form>
			<% } %>
</div>
<%@include file="../pie.jsp"%>
</body>
</html>
```

11.5 /admin/ verNotas.jsp

```
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.PreparedStatement"%>
<%@page import="java.sql.Connection"%>
<%@page import="javax.sql.DataSource"%>
<%@page import="javax.naming.InitialContext"%>
<%@page import="java.sql.SQLException"%>
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-
8"%>

<jsp:useBean id="ds" type="javax.sql.DataSource" scope="application" />
<!DOCTYPE html>
<html lang="es">
<head>
	<meta charset="UTF-8">
	<title>Ver Notas</title>
	<link rel="stylesheet" href="../css/estilos.css">

</head>
<body>
<jsp:include page="../cabecera.jsp" />
<div id="contenedor">
<h1>Listado de todas las notas por alumno</h1>
<br>
<br>

<form action="verNotas.jsp" method="Post">
<p> Selecciona el curso al que visualizar las notas: <select name=curso>

<%
	try {
		Connection conn = ds.getConnection();
		String sql = "SELECT id_nombrecurso, id_curso FROM cursosgrupo order by
id_nombrecurso";
		PreparedStatement st = conn.prepareStatement(sql);
		ResultSet rs = st.executeQuery();
		while (rs.next()){
			String id_nombrecurso = rs.getString(1);
			String id_curso = rs.getString(2);

		%>
		<option value="<%=id_curso%>">[<%=id_curso%>]-
<%=id_nombrecurso%></option>
		<%
		}
			rs.close();
			st.close();
			conn.close();
			}catch (SQLException e){
				out.println("Excepción SQL Exception: "+ e.getMessage());
				e.printStackTrace();
			}
		%>

</select>
</p>
 <p><input type="submit" name="enviar" value="Seleccionar" />
```

```
</form>
<%
if(request.getParameter("curso")!=null){
	String cursoNote = request.getParameter("curso");
%>
<h1>Notas de <%=cursoNote%></h1>
	<table>
	<tr>
	<th>Usuario</th>
	<th>Grupo Puntuado</th>
	<th>Preferencia</th>
	<%
	try{
		Connection conn = ds.getConnection();
		String sql = "SELECT criterio FROM criterio_x where id_curso = ? order by
id_criterio";
		PreparedStatement st = conn.prepareStatement(sql);
		st.setString(1, request.getParameter("curso"));
		ResultSet rs = st.executeQuery();
		while (rs.next()){
			%>
			<th><%=rs.getString(1)%></th>
			<%
		}

	rs.close();
	st.close();
	conn.close();
	}catch (SQLException e){
		out.println("Excepción SQL Exception: "+ e.getMessage());
		e.printStackTrace();
	}
	%>
	</tr>
	<%
	String usernota = null;
	String gruponota = null;
	String cursonota = null;
	String puntuanota = null;
	try{
		Connection conn = ds.getConnection();
		String sql = "SELECT distinct a.id_usuario, a.grupo_puntuado, a.id_curso,
b.orden_grupo FROM puntua_x a, preferente_x b where a.id_usuario=b.id_usuario and
a.id_curso=b.id_curso and b.id_curso=? and a.grupo_puntuado=b.num_grupo order by
id_usuario";
		PreparedStatement st = conn.prepareStatement(sql);
		st.setString(1, request.getParameter("curso"));
		ResultSet rs = st.executeQuery();
		while (rs.next()){
			usernota = rs.getString(1);
			gruponota = rs.getString(2);
			cursonota = rs.getString(3);
			puntuanota = rs.getString(4);
			%>

	<tr><td><%=usernota%></td><td><%=gruponota%></td><td><%=puntuanota%></td>
			<%
			try{
				Connection conn2 = ds.getConnection();
```

```
                                        String sql2 = "SELECT puntuacion FROM puntua_x where
id_usuario=? and id_curso=? and grupo_puntuado=? order by id_criterio";
                                        PreparedStatement st2 = conn2.prepareStatement(sql2);
                                        st2.setString(1, usernota);
                                        st2.setInt(3, Integer.parseInt(gruponota));
                                        st2.setString(2, cursonota);
                                        ResultSet rs2 = st2.executeQuery();
                                        while (rs2.next()){
                                        %>
                                        <td><%=rs2.getString(1)%></td>
                                        <%
                                        }
                                }
                                catch (SQLException e){
                                        out.println("Excepción SQL Exception: "+ e.getMessage());
                                        e.printStackTrace();
                                }
                                %>
                                </tr>
                                <%
                        }
                        rs.close();
                        st.close();
                        conn.close();
                }
                catch (SQLException e){
                        out.println("Excepción SQL Exception: "+ e.getMessage());
                        e.printStackTrace();
                }
                %>
                </table>
<%
}
%>

</div>
<%@include file="../pie.jsp"%>

</body>
</html>
```

11.6 /admin/ verAlumnos.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
        pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html lang="es">
<head>
        <meta charset="UTF-8">
        <title>Alumnos</title>
        <link rel="stylesheet" href="../css/estilos.css">
        <script type="text/javascript" src="../clientes/js/funTablasURL.js"></script>
        <script type="text/javascript" src="../clientes/js/funCargaTablas.js"></script>

</head>
<body>
<jsp:include page="../cabecera.jsp" />

                <table id="tablaUser">
                <tr>
                <th id="alum"><h1>Ver alumnos</h1></th>
                </tr>
                <tr id="filaUser">
                <td id="listaUser">
                <div id="contenedor">
                <table class="interfaces" id="verUsuarios">
                </table>
                        </div>
                </td>
                <td id="listaMod">
                <div id="contenedor">
</div>
                </td>
                </tr>
                </table>

<%@include file="../pie.jsp"%>

</body>
</html>
```

11.7 /admin/ usuForm.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
         pageEncoding="UTF-8"%>


<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8"/>
<title>Usuario</title>

<link href="../css/estilos.css" rel="stylesheet" type="text/css" />

<script type="text/javascript" src="js/funUsu.js"></script>

<script src="../cgi-bin/jscgi"></script>

</head>
<body>

<jsp:include page="../cabecera.jsp" /> <!-- LA INCLUIMOS AL COMIENZO DEL BODY -->
<div id="contenedor">

<h1>Creación/modificación de alumno</h1>

<h2 class="entrada">Introduzca usuario y contraseña</h2>

<p>El usuario debe contener un máximo de <span class="warning entrada">11 caracteres
</span></p>

<p>La contraseña debe cumplir las siguientes condiciones: </p>

<ol>
  <li>debe tener una longitud mínima de 6 caracteres </li>
  <li>debe tener una longitud máxima de 10 caracteres </li>
  <li>debe contener al menos dos caracteres numéricos </li>
  <li>debe contener al menos dos caracteres en mayúsculas </li>
  <li>debe contener al menos dos caracteres en minúsculas </li>
  <li>no debe contener el nombre del alumno </li>
</ol>
<form action="modificarUsuario" id="formUsu" name="formUsu" method="Post" onsubmit=
"return validaFormulUsu()">
<p><label><input type="text" id="usu" name="usu" maxlength=11
required/>Usuario</label></p>
<p><label><input type="password" id="contra1" name="contra1" maxlength=10
required/>Contraseña</label></p>
<p><label><input type="password" id="contra2" name="contra2" maxlength=10 />Repetición de
la contraseña</label></p>
<p><input type="submit" name="enviar" value="Enviar" />
   <input type="reset" name="borrar" value="Limpiar" /></p>
</form>
</div>

<%@include file="../pie.jsp"%>
</body>
</html>
```

11.8 /clientes/ menu.jsp

```
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.PreparedStatement"%>
<%@page import="java.sql.Connection"%>
<%@page import="javax.sql.DataSource"%>
<%@page import="javax.naming.InitialContext"%>
<%@page import="java.sql.SQLException"%>
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-
8"%>

	<jsp:useBean id="ds" type="javax.sql.DataSource" scope="application" />
<jsp:useBean id="usuario" type="fast1718.trabajo.Usuario" scope="session"/>
<!DOCTYPE html>
<html>
<head>
	<meta charset="UTF-8">
	<title>Menú Alumnos</title>
 <link href="css/estilos.css" rel="stylesheet" type="text/css" />
</head>
<body>
<jsp:include page="../cabecera.jsp" />
	<div id="contenedor">
		<h1>Menú alumnos</h1>
		<ul>
			<li><a href="clientes/cambiacontra.jsp">Cambiar contraseña</a></li>
			<li><a href="clientes/puntuar.jsp"> Puntuar</a></li>

		</ul>
	</div>
<%@include file="../pie.jsp"%>
</body>
</html>
```

11.9 clientes/ puntuar.jsp

```
<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.PreparedStatement"%>
<%@page import="java.sql.Connection"%>
<%@page import="javax.sql.DataSource"%>
<%@page import="javax.naming.InitialContext"%>
<%@page import="java.sql.SQLException"%>
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-
8"%>

<jsp:useBean id="ds" type="javax.sql.DataSource" scope="application" />
<jsp:useBean id="usuario" type="fast1718.trabajo.Usuario" scope="session"/>
<!DOCTYPE html>
<html lang="es">
<head>
	<meta charset="UTF-8">
	<title>Cursos</title>
	<link rel="stylesheet" href="../css/estilos.css">
</head>
<body>
<jsp:include page="../cabecera.jsp" />
	<div id="contenedor">
	<h1>Puntuación de Grupos en diversos apartados</h1>
	<table>
	<tr>
	<th><h3>Selecciona tu curso</h3></th>
	<th><h3>Seleccionar Grupo a puntuar</h3></th>
	</tr>
	<tr>
	<td id="CursoPunt">
	<form action="puntuar.jsp" method="Post">
	<select name=curso>
	<%
	Integer max_curso = 0;
	Integer permiso = 0;
	try {
		Connection conn = ds.getConnection();
		String sql = "SELECT a.id_nombrecurso, a.id_curso,c.permite_puntua FROM
cursosgrupo a, curso_x b, permisos c where b.id_usuario=? and a.id_curso=b.id_curso and
c.id_curso=a.id_curso order by a.id_nombrecurso";
		PreparedStatement st = conn.prepareStatement(sql);
		st.setString(1, usuario.getUsu());
		ResultSet rs = st.executeQuery();
		while (rs.next()){
			String id_nombrecurso = rs.getString(1);
			String id_curso = rs.getString(2);
			permiso = rs.getInt(3);
			if(request.getParameter("curso")!=null &&
request.getParameter("curso").equals(id_curso)){%>
				<option value="<%=id_curso%>"
selected><%=id_nombrecurso%></option>
				<%
			}
			else{
				if(permiso!=0){%>
				<option
value="<%=id_curso%>"><%=id_nombrecurso%></option>
```

```
                <%      }
                                }
                }
                        rs.close();
                        st.close();
                        conn.close();
                        }catch (SQLException e){
                                out.println("Excepción SQL Exception: "+ e.getMessage());
                                e.printStackTrace();
                        }
                %>
        </select>
        <p><input type="submit" name="enviar" value="Seleccionar" />
        </form>
        </td>
        <td id="GrupoPunt">
        <%if (request.getParameter("curso")!=null){%>
        <form action="puntuar.jsp" method="Post">
        <label><input type="text" id="curso" name="curso"
value=<%=request.getParameter("curso")%> readonly style="display:none;" /></label>
        <select name=grupo>
        <%
        Integer num_grupo_max=0;
        Integer num_grup_ele=0;
                try {
                        Connection conn = ds.getConnection();
                        String sql = "SELECT a.num_grupo, b.num_grupo FROM cursosgrupo
a,curso_x b where a.id_curso=? and id_usuario=? and a.id_curso=b.id_curso";
                        PreparedStatement st = conn.prepareStatement(sql);
                        st.setString(1, request.getParameter("curso"));
                        st.setString(2, usuario.getUsu());
                        ResultSet rs = st.executeQuery();
                        while (rs.next()){
                                num_grupo_max=rs.getInt(1);
                                num_grup_ele=rs.getInt(2);
                        }
                        rs.close();
                        st.close();
                        conn.close();
                        }catch (SQLException e){
                                out.println("Excepción SQL Exception: "+ e.getMessage());
                                e.printStackTrace();
                        }
                 max_curso = num_grupo_max;
                while(num_grupo_max!=0){
                        if(num_grupo_max!=num_grup_ele){
                        %><option
value="<%=num_grupo_max%>"><%=num_grupo_max%></option><%
                        }
                        num_grupo_max--;
                }
                                %>
                                </select>
                         <p><input type="submit" name="enviar" value="Seleccionar" />
                </form>
                <%} else{ %><h3>Porfavor, selecciona antes curso.</h3><%} %>
                </td>

                </tr>
        </table>
```

```
<%if (request.getParameter("grupo")!=null){ %>
<br>
<form action="puntuaCurso" method='post'>

<table>
	<tr><th colspan="100%"><h3>Evalua al grupo <%=request.getParameter("grupo")%>
de la asignatura <%=request.getParameter("curso")%>, según lo siguientes
criterios</h3></th></tr>
<tr>

<%
String criter = "";
Integer id_criter = 0;
%>
<td>
<%
	Integer[] ordeni = new Integer[max_curso];
	Integer[] grupordeni = new Integer[max_curso];
	Integer a = 0;
	Integer on_point = 0;
	try {
		Connection conn = ds.getConnection();
		String sql = "SELECT orden_grupo, num_grupo FROM preferente_x where
id_curso=? and id_usuario=? order by orden_grupo";
		PreparedStatement st = conn.prepareStatement(sql);
		st.setString(1, request.getParameter("curso"));
		st.setString(2, usuario.getUsu());
		ResultSet rs = st.executeQuery();

		while (rs.next()){
			ordeni[a]=rs.getInt(1);
			grupordeni[a]=rs.getInt(2);
			a++;
		}
	rs.close();
	st.close();
	conn.close();
	}catch (SQLException e){
		out.println("Excepción SQL Exception: "+ e.getMessage());
		e.printStackTrace();
	}

	Integer indice = 0;
	while(indice<a){
		if(Integer.parseInt(request.getParameter("grupo")) == grupordeni[indice]){
			on_point=ordeni[indice];
			}
		indice++;
	}

if(on_point!=0){
	%>
<p>El grupo <%=request.getParameter("grupo")%> ya esta evaluado siendo el
<%=on_point%>º</p>
<%
} else{
	%>
```

```
<label><input type="text" id="cursoidn" name="cursoidn"
value=<%=request.getParameter("curso")%> readonly style="display:none;" /></label>
<label><input type="text" id="usuario" name="usuario" value=<%=usuario.getUsu()%> readonly
style="display:none;" /></label>
<label><input type="number" id="grupodn" name="grupodn"
value=<%=request.getParameter("grupo")%> readonly style="display:none;" /></label>
        <p>Como consideras en general el grupo <%=request.getParameter("grupo")%></p>
        <select name=preferencia>
        <option value="0">Por debajo de la media</option>
        <option value="1">En la media</option>
        <option value="2">Por encima de la media</option>
        </select>

<%
}
%>
        </td>
<%

try {
        Connection conn = ds.getConnection();
        String sql = "SELECT criterio , id_criterio FROM criterio_x where id_curso=?";
        PreparedStatement st = conn.prepareStatement(sql);
        st.setString(1, request.getParameter("curso"));
        ResultSet rs = st.executeQuery();
        while (rs.next()){
                criter = rs.getString(1);
                id_criter=rs.getInt(2);
%>
        <td>
        <p>Evalue su <%=criter%> </p>
        <p>(Valoracion de 1 a 5) </p>
        <input type="number" name="<%=id_criter%>" min=1 max=5 required>
        <br>
        </td>

<%              }

        rs.close();
        st.close();
        conn.close();
        }catch (SQLException e){
                out.println("Excepción SQL Exception: "+ e.getMessage());
                e.printStackTrace();
        }
%>
        </tr>
        <tr><td colspan="100%">
        <input type="submit" name="enviar" value="Puntuar" />
        </td>
        </tr>

</table>
</form>
        <% } %>
        </div>

<%@include file="../pie.jsp"%>
</body>
</html>
```

11.10 clientes/ cambiarcontra.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
         pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8"/>
<title>Usuario</title>
<link href="../css/estilos.css" rel="stylesheet" type="text/css" />
<script type="text/javascript" src="js/cambiaContra.js"></script>
</head>
<body>
<jsp:include page="../cabecera.jsp" />
<div id="contenedor">
<h1>Modificación de contraseña</h1>
<h2 class=${clase} id="mensaje">${mensaje}</h2>

<h2 class="entrada">Introduzca la contraseña actual y una nueva contraseña</h2>

<p>La contraseña debe cumplir las siguientes condiciones: </p>
<ol>
  <li>debe tener una longitud mínima de 6 caracteres </li>
  <li>debe tener una longitud máxima de 10 caracteres </li>
  <li>debe contener al menos dos caracteres numéricos </li>
  <li>debe contener al menos dos caracteres en mayúsculas </li>
  <li>debe contener al menos dos caracteres en minúsculas </li>
  <li>no debe contener el nombre de usuario </li>
</ol>
<form action="modificarPass" id="formUsu" name="formUsu" method="Post" onsubmit= "return
validaContra()">

<p><label><input type="password" id="contra" name="contra" maxlength=10
required/>Contraseña actual:</label></p>

<p><label><input type="password" id="contra1" name="contra1" maxlength=10
required/>Contraseña nueva:</label></p>

<p><label><input type="password" id="contra2" name="contra2" maxlength=10 />Repetición de
la contraseña nueva:</label></p>

<p><input type="submit" name="enviar" value="Enviar" />

   <input type="reset" name="borrar" value="Limpiar" />

   <input type="button" name="muestracontra" id="muestracontra" value="Mostrar"/>
   </p>

</form>
</div>

<%@include file="../pie.jsp"%>
</body>
</html>
```

11.11 /index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
	pageEncoding="UTF-8"%>
<%
	if (session.getAttribute("usuario") != null)
		response.sendRedirect("menu");
%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Plataforma de evaluación - TRAJANO</title>
<link rel="stylesheet" type="text/css" href="css/estilos.css" />
<script src="js/conmutaAcceso.js"></script>
</head>
<body>
	<noscript>
		<p>
			<span class="acceso"> Esta p&aacute;gina necesita JavaScript
				para funcionar </span>
		</p>
	</noscript>
	<div id="bienvenida">
		<h1>Bienvenido</h1>
		<h2>Plataforma de autoevaluación - Trajano</h2>
		<h3>Acceda con su uvus y contraseña</h3>
		<button id="botAcceso">Login</button>

	</div>
	<div id="formacceso">
		<form action="menu" method="post">
			<p>
				<label for="usu">Uvus:</label><br /> <input type="text"
					name="usu" size="11" id="usu" />
			</p>
			<p>
				<label for="contra">Contraseña:</label> <br /> <input
type="password"
					name="contra" size="10" id="contra" />
			</p>
			<p>
				<input class="boton" type="submit" name="entrar" id="entrar"
value="Entrar" />
				   <input class="boton" type="button" id="cancelar"
					value="Cancelar" />
			</p>
		</form>
	</div>
	<%@include file="pie.jsp"%>
</body>
</html>
```

11.12 /cabecera.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
         pageEncoding="UTF-8"%>


<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
</head>
<body>

<div id="cabecera">
<a href="http://www.us.es">
<img id="logCabecera2" src="http://ftp.us.es/ftp/pub/Logos/marca-dos-tintas_300.gif" alt="US"
height="45">
</a>
<a id="logCabecera"  href="http://trajano.us.es">
<img src="http://trajano.us.es/graficos/dit.gif" alt="DIT" height="45">
</a>
 <a id="dep">Departamento de Ingeniería Telemática</a>
<a id="volver" class="acceso" href="${pageContext.servletContext.contextPath}/menu">Volver
al menú</a>
<a class="acceso" id="cerrar" href="${pageContext.servletContext.contextPath}/cerrar">Cerrar
sesión</a>
 <span id="nombreusuario" class="acceso">${sessionScope.usuario.name}</span>
</div>

<div>


</div>
</body>
</html>
```

11.13 /pie.jsp

```
<%@page import="java.util.Date"%>
<%@page import="java.text.SimpleDateFormat"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Plataf. Evaluacion - Trajano</title>
</head>
<body>
<div id="pie">
<footer><h2>
<%Date date = new Date();
SimpleDateFormat dateFormat = new SimpleDateFormat("dd-MMMM-yyyy");
out.println(dateFormat.format(date));
%>
</h2>
<h3>Creado por ${applicationScope.autor}</h3>
</footer>
</div>
</body>
</html>
```

11.14 /AppListener.java

```
package fast1718.trabajo;

import java.sql.Connection;
//import java.sql.PreparedStatement;
//import java.sql.ResultSet;
import java.sql.SQLException;
//import java.util.HashMap;
//import java.util.Map;

import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.servlet.ServletContext;
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;
import javax.servlet.annotation.WebListener;
import javax.sql.DataSource;


/**
 * Application Lifecycle Listener implementation class AppListener
 *
 */
@WebListener
public class AppListener implements ServletContextListener {

    /**
     * Default constructor.
     */
    public AppListener() {
    }

        /**
     * @see ServletContextListener#contextDestroyed(ServletContextEvent)
     */
    public void contextDestroyed(ServletContextEvent arg0)  {
    }

        /**
     * @see ServletContextListener#contextInitialized(ServletContextEvent)
     */
    public void contextInitialized(ServletContextEvent event)  {
        String autor = "Salguzosu";
        ServletContext ctx1 = event.getServletContext();
       ctx1.setAttribute("autor", autor); // ("cadena que especifica el nombre del atributo", objeto
que almacena)
        System.out.println("Creado atributo autor: "+ autor); // Imprime por consola para
comprobar errores

        try {
                        // Usando DataSource ya definido en el servidor
                        InitialContext ctx = new InitialContext();
                        // /jdbc/dit is the name of the resource above
                        DataSource ds = (DataSource) ctx.lookup("java:comp/env/jdbc/dit");
                        Connection conn = ds.getConnection();

                        //String sql = "SELECT * FROM tipos_in";
                        //PreparedStatement st = conn.prepareStatement(sql);
```

```
                        //ResultSet rs = st.executeQuery();
                        //Map<Integer,String> tipos_in = new HashMap<Integer, String>();
//Mapa de tipos_in
                        //while (rs.next()) {
                        //          tipos_in.put(rs.getInt("id_ti"), rs.getString("des")); // le vamos
asignando los valores de tipos_in los valores de BBDD
                //          }
                //          ctx1.setAttribute("tipos_in", tipos_in); //edu ("cadena que especifica el
nombre del atributo", objeto que almacena)

                //          System.out.println("Creado atributo tipos_in: "+ tipos_in);
                        ctx1.setAttribute("ds", ds);
                        System.out.println("Creado atributo ds: "+ ds);
                //          rs.close();
                //          st.close();
                        conn.close();
                } catch (NamingException e) {
                        System.out.println("No está definida la base de datos en el servidor.
AppListener");
                } catch (SQLException e) {
                        e.printStackTrace();
                        System.out.println("Error de acceso a la base de datos. AppListener.");
                }

   }

}
```

11.15 /FiltroAdmin.java

```
package fast1718.trabajo;


import java.io.IOException;

//import javax.naming.NamingException;
//import javax.naming.InitialContext;
//import javax.naming.NamingException;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebFilter;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


/**
 * Servlet Filter implementation class FiltroMenu
 */
//@WebServlet("")

 @WebFilter("/admin/*") // filtro para la URL solo admin y con * todo lo que esté dentro de ello

public class FiltroAdmin implements Filter {


	private static final int ADMINISTRADOR = 0; //CONSTANTE CREADA POR MI



	/**
	 * Default constructor.
	 */
	public FiltroAdmin() {
	}

	/**
	 * @see Filter#destroy()
	 */
	public void destroy() {
	}

	/**
	 * @see Filter#doFilter(ServletRequest, ServletResponse, FilterChain)
	 */
	public void doFilter(ServletRequest request, ServletResponse response, FilterChain
chain)
				throws IOException, ServletException {
```

```
		HttpServletRequest req = (HttpServletRequest) request;
		//HttpServletResponse res = (HttpServletResponse) response; esta definida
abajo

		Usuario usuario = new Usuario();

		usuario = (Usuario) req.getSession().getAttribute("usuario"); // Cojemos el
objeto sesion usuario


			if ((usuario != null) && (usuario.getTipo_usu() == ADMINISTRADOR ))
			{

				// Continua la cadena de filtros
				chain.doFilter(req, response);
			}

			else

			{
				HttpServletResponse res = (HttpServletResponse) response;
				res.sendError(HttpServletResponse.SC_FORBIDDEN, "Acceso
prohibido");


			}
		}



	/**
	 * @see Filter#init(FilterConfig)
	 */
	public void init(FilterConfig fConfig) throws ServletException {
	}

}
```

11.16/FiltroClientes.java

```
package fast1718.trabajo;


import java.io.IOException;

//import javax.naming.NamingException;
//import javax.naming.InitialContext;
//import javax.naming.NamingException;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebFilter;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


/**
 * Servlet Filter implementation class FiltroMenu
 */
//@WebServlet("")

 @WebFilter("/clientes/*") // filtro para la URL solo admin y con * todo lo que esté dentro de ello

public class FiltroClientes implements Filter {



	//private static final int ADMINISTRADOR = 0; //CONSTANTE CREADA POR MI



	/**
	 * Default constructor.
	 */
	public FiltroClientes() {
	}

	/**
	 * @see Filter#destroy()
	 */
	public void destroy() {
	}

	/**
	 * @see Filter#doFilter(ServletRequest, ServletResponse, FilterChain)
	 */
	public void doFilter(ServletRequest request, ServletResponse response, FilterChain
chain)
				throws IOException, ServletException {
		HttpServletRequest req = (HttpServletRequest) request;
```

```
		//HttpServletResponse res = (HttpServletResponse) response; esta definida
abajo

		Usuario usuario = new Usuario(); // Hay que crear objeto usuario para luego
usarlo

		usuario = (Usuario) req.getSession().getAttribute("usuario");


			if (usuario != null)
			{

				// Continua la cadena de filtros
				chain.doFilter(req, response);
			}

			else

			{
				HttpServletResponse res = (HttpServletResponse) response;
				res.sendError(HttpServletResponse.SC_FORBIDDEN, "Acceso
prohibido");


			}
		}



	/**
	 * @see Filter#init(FilterConfig)
	 */
	public void init(FilterConfig fConfig) throws ServletException {
	}

}
```

11.17/FiltroMenu.java

```
package fast1718.trabajo;

import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

//import javax.naming.NamingException;
//import javax.naming.InitialContext;
//import javax.naming.NamingException;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.annotation.WebFilter;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.sql.DataSource;


/**
 * Servlet Filter implementation class FiltroMenu
 */

@WebFilter("/menu") // filtro para la URL

public class FiltroMenu implements Filter {



	private static final int ADMINISTRADOR = 0;
	/**
	 * Default constructor.
	 */
	public FiltroMenu() {
	}

	/**
	 * @see Filter#destroy()
	 */
	public void destroy() {
	}

	/**
	 * @see Filter#doFilter(ServletRequest, ServletResponse, FilterChain)
	 */
	public void doFilter(ServletRequest request, ServletResponse response, FilterChain
chain)
				throws IOException, ServletException {
```

```
		HttpServletRequest req = (HttpServletRequest) request;
		HttpServletResponse res = (HttpServletResponse) response;

		//Comprueba si es un login
		String usu = req.getParameter("usu"); // valores cojidos del index
		String contra = req.getParameter("contra");

		if (usu != null && contra != null)
		{
			Usuario usuario = new Usuario(); //Crea un objeto de tipo Usuario
			//Valida contraseña y usuario
			boolean verificado = false;
			try {
				DataSource ds = (DataSource)
req.getServletContext().getAttribute("ds");
				Connection conn = ds.getConnection();

				String sql = "SELECT tipo_usu,id_nombre FROM usuarios
WHERE id_usuario=? AND password=?";
				PreparedStatement st = conn.prepareStatement(sql);
				st.setString(1, usu);
				st.setString(2, contra);
				ResultSet rs = st.executeQuery();

				if (rs.next()) { // si esta en la BBDD
					verificado = true;

					usuario.setUsu(usu); // al objeto usuario le damos los
valores obtenidos
					usuario.setContra(contra);
					usuario.setTipo_usu(rs.getInt("tipo_usu")); //le ponemos
el valor del que tenga este usuario
					usuario.setName(rs.getString("id_nombre"));

				}
				rs.close();
				st.close();
				conn.close();
			} catch (SQLException e) {
				e.printStackTrace();
				System.out.println("Error de acceso a la base de datos.
FiltroMenu");
			}

			//Si es correcto, crea objetos en session
			if (verificado)
			{
				req.getSession().setAttribute("usuario", usuario); //Crea Atributo
sesión de nombre usuario
				// Creamos la cookie
				Cookie usuCookie = new Cookie( "usuario", usu); // valor del
usuario autentificado lo añade a respuesta
				res.addCookie(usuCookie);
			}

		}

		Usuario usuario = (Usuario) req.getSession().getAttribute("usuario");
//Buscamos atributo sesión de nombre usuario
		if (usuario != null) {
```

```
                                        if (usuario.getTipo_usu() == ADMINISTRADOR) // TENGO
QUE PONERLO QUE SE PASE POR PARAMETRO

                                        {


        request.getRequestDispatcher("/admin/menu.jsp").forward(request, response);
                                        }

                                        else

                                        {


        request.getRequestDispatcher("/clientes/menu.jsp").forward(request, response);
                                        }


                        } else {
                                request.getRequestDispatcher("/").forward(request, response); //Si no lo
encuentra envia a reinicio aplicacion
                        }

        }

        /**
         * @see Filter#init(FilterConfig)
         */
        public void init(FilterConfig fConfig) throws ServletException {
        }

}
```

11.18/ServletCreaCriterios.java

```
package fast1718.trabajo;

import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.Enumeration;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.sql.DataSource;

/**
 * Servlet implementation class ServletCreaCriterios
 */
@WebServlet(urlPatterns={"/admin/creaCriterios"})
public class ServletCreaCriterios extends HttpServlet {
	private static final long serialVersionUID = 1L;


    /**
     * @see HttpServlet#HttpServlet()
     */
    public ServletCreaCriterios() {
        super();
    }
	/**
	 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
	 */

	protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
			String cursoidn = request.getParameter("cursoidn");
			int id_criterio = Integer.parseInt(request.getParameter("criterioID"));
			int i=0;
			String[] criterios = new String[id_criterio];
			int x = 1;
		Enumeration<String> ed = request.getParameterNames();
		while(ed.hasMoreElements())
		{
			Object obj = ed.nextElement();
			String fieldName=(String)obj;
			String fieldValue=request.getParameter(fieldName);

		if (x>2 && x<=(id_criterio+2)) {
			criterios[i]=fieldValue;
			  i++;
		}
		x++;
			}
```

```
		boolean error = false;
		String mensaje = "";
		if (cursoidn!= null && !cursoidn.isEmpty())
		{
			String sql;
			int z=0;
			while(z<id_criterio) {
			try {
				DataSource ds = (DataSource)
request.getServletContext().getAttribute("ds");
				Connection conn = ds.getConnection();
				sql = "INSERT INTO criterio_x VALUES (?,?,?)";
				PreparedStatement st = conn.prepareStatement(sql);
				st.setString(1, cursoidn);
				st.setString(3, criterios[z]);
				st.setInt(2, z);
				int rows = st.executeUpdate();
				System.out.println("ROWS= "+rows);
			}
			catch (SQLException e) {
				error = true;
				mensaje = "Error con los campos del formulario";
			}
			z++;
			}
		}
		 else {
				error = true;
				mensaje = "Error en los datos.";
			}


		if (error) {
			request.setAttribute("mensaje", mensaje );

	request.setAttribute("volverURL","/EvalProyecto/admin/administrarCursos.jsp"); // Si
pinchamos se vuelve a la pagina anterior
			request.getRequestDispatcher("/clientes/error.jsp").forward(request,
response);
		} else {

	request.getRequestDispatcher("/admin/administrarCursos.jsp").forward(request,
response);
		}

	}

}
```

11.19 /ServletModificaContra.java

```
package fast1718.trabajo;
import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.sql.DataSource;

@WebServlet(urlPatterns={"/clientes/modificarPass"})
public class ServletModificarContra extends HttpServlet {
	private static final long serialVersionUID = 1L;
	private static final int LON_MIN = 6;
	private static final int LON_MAX = 10;
	private static final int MIN_MAYUSCULAS = 2;
	private static final int MIN_MINUSCULAS = 2;
	private static final int MIN_NUMERICOS = 2;
	private boolean compruebaContraUsu(String pass) {
		boolean res = false;
		int numericos = 0;
		int mayusculas = 0;
		int minusculas = 0;
		String mayus ="ABCDEFGHYJKLMNÑOPQRSTUVWXYZ";
		String minus ="abcdefghyjklmnñopqrstuvwxyz";
		String nume = "0123456789";

		if ( (pass.length() < LON_MIN) || (pass.length() > LON_MAX)) {

			res = false;

			return res;
		}
		  for (int i = 0; i < pass.length(); i++) {
				if( nume.indexOf(pass.charAt(i),0) !=-1) {
				numericos = numericos + 1;
				}
				if( mayus.indexOf(pass.charAt(i),0) !=-1) {
					mayusculas = mayusculas + 1;
				}
				if( minus.indexOf(pass.charAt(i),0) !=-1) {
					minusculas = minusculas + 1;
				}
		  }
			if ( numericos < MIN_NUMERICOS || mayusculas <
MIN_MAYUSCULAS || minusculas < MIN_MINUSCULAS ) {
				res = false;
			}
			else {
				res = true;
			}
		return res;
	}
```

```
   /**|
    * @see HttpServlet#HttpServlet()
    */
   public ServletModificarContra() {
      super();
   }
         /**
          * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
          */
         protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
                  String contra = request.getParameter("contra");
                  String contra1 = request.getParameter("contra1");
                  String contra2 = request.getParameter("contra2");
                  boolean error = false;
                  String mensaje = "";
                  String clase = "warning";
                  Usuario usuario = new Usuario();
                  usuario = (Usuario) request.getSession().getAttribute("usuario");
                  if (contra!= null && contra1!=null && contra2!=null
                                    && !contra.isEmpty() && !contra1.isEmpty() &&
!contra2.isEmpty()
                                    && contra1.equals(contra2)
                                    && compruebaContraUsu(contra1)
                                    && !contra1.equals(contra))
                  {
                           boolean contraCorrecta = false;
                           String sql;
                           try {
                                    DataSource ds = (DataSource)
request.getServletContext().getAttribute("ds");
                                    Connection conn = ds.getConnection();
                                             sql = "select * from usuarios where id_usuario =? and
password=?";
                                             PreparedStatement pst = conn.prepareStatement(sql);
                                             pst.setString(1, usuario.getUsu());
                                             pst.setString(2, contra);
                                             ResultSet rs = pst.executeQuery();
                                             if(rs.next())
                                                      contraCorrecta = true;
                                             rs.close();
                                             pst.close();
                                             if(contraCorrecta) {
                                             sql = "UPDATE usuarios set password =? where
id_usuario =?";
                                             PreparedStatement st = conn.prepareStatement(sql);
                                             st.setString(1, contra1);
                                             st.setString(2, usuario.getUsu());
                                             int rows = st.executeUpdate();
                                             if (rows != 1) {
                                                      error = true;
                                                      mensaje = "Error modificando la contraseña";
                                             }
                                             else {
                                                      mensaje="Modificación correcta";
                                                      clase="check";
                                             }
                                             st.close();
```

```
                                        }
                                                else {
                                                        error = true;
                                                        mensaje = "No coincide la contraseña";
                                                }
                                }
                                 catch (SQLException e) {
                                        error = true;
                                        mensaje = "Error con la BBDD";
                                }
                        } else {
                                error = true;
                                mensaje = "Error en los Datos.";
                        }
                        if (error) {
                                request.setAttribute("mensaje", mensaje );
                                request.setAttribute("clase", "warning");

                request.getRequestDispatcher("/clientes/cambiacontra.jsp").forward(request, response);
                        } else {
                                request.setAttribute("clase", "check");
                                request.setAttribute("mensaje", mensaje);

                request.getRequestDispatcher("/clientes/cambiacontra.jsp").forward(request, response);
                        }


                }

        }
```

11.20 /ServletModificaCurso.java

```
package fast1718.trabajo;
import java.io.IOException;
/**
 * Servlet implementation class ServletCrearUsuario
 */
@WebServlet(urlPatterns={"/admin/modificarCurso"})
public class ServletModificarCurso extends HttpServlet {
	private static final long serialVersionUID = 1L;
    /**
     * @see HttpServlet#HttpServlet()
     */
    public ServletModificarCurso() {
        super();
    }
	/**
	 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
	 */
	protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
		String curso = request.getParameter("curso");
		String id = request.getParameter("id").toUpperCase();
		String grupoINT = null;
		boolean error = false;
		String mensaje = "";
		String sql2;
		try {
			DataSource ds = (DataSource)
request.getServletContext().getAttribute("ds");
			Connection conn = ds.getConnection();
				sql2 = "select distinct num_grupo from curso_x where
id_curso=?";
				PreparedStatement st = conn.prepareStatement(sql2);
				st.setString(1, id);
				ResultSet rs = st.executeQuery();
				while (rs.next()){
					grupoINT = rs.getString(1);
				}
				rs.close();
				st.close();
				conn.close();
				st.close();
			}
		 catch (SQLException e) {
			error = true;
			mensaje = "Error con los campos del formulario";
		}
		int grupo = Integer.parseInt(grupoINT);
		if (id!= null && !id.isEmpty() && curso!= null && !curso.isEmpty())
		{
			String sql;
			try {
				DataSource ds = (DataSource)
request.getServletContext().getAttribute("ds");
				Connection conn = ds.getConnection();
					sql = "INSERT INTO cursosgrupo
(id_nombrecurso,num_grupo,id_curso) VALUES (?,?,?) ON CONFLICT (id_curso) DO UPDATE
SET id_nombrecurso=?,num_grupo=?";
```

```
                                        PreparedStatement st = conn.prepareStatement(sql);
                                        st.setInt(2, grupo);
                                        st.setString(1, curso);
                                        st.setString(3, id);
                                        st.setString(4, curso);
                                        st.setInt(5, grupo);
                                        int rows = st.executeUpdate();
                                        if (rows == 1) {
                                                //Modificación correcta
                                        } else {
                                                error = true;
                                                mensaje = "Error Creando/Modificando curso";
                                        }
                                        st.close();
                                }
                                catch (SQLException e) {
                                        error = true;
                                        mensaje = "Error con los campos del formulario";
                                }
                                try {
                                        DataSource ds = (DataSource)
request.getServletContext().getAttribute("ds");
                                        Connection conn = ds.getConnection();
                                                sql = "INSERT INTO permisos (permite_grupo,
permite_puntua,id_curso) VALUES (?,?,?)";
                                        PreparedStatement st = conn.prepareStatement(sql);
                                        st.setInt(1, 0);
                                        st.setString(3, id);
                                        st.setInt(2, 0);
                                        int rows = st.executeUpdate();
                                        if (rows == 1) {
                                                //Modificación correcta
                                        } else {
                                                error = true;
                                                mensaje = "Error Creando/Modificando curso";
                                        }
                                        st.close();
                                }
                                catch (SQLException e) {
                                        error = true;
                                        mensaje = "Error con los campos del formulario";
                                }
                        } else {
                                error = true;
                                mensaje = "Error en los datos del usuario.";
                        }
                        if (error) {
                                request.setAttribute("mensaje", mensaje );

                request.setAttribute("volverURL","/EvalProyecto/admin/administrarCursos.jsp"); // Si
pinchamos se vuelve a la pagina anterior
                                request.getRequestDispatcher("/clientes/error.jsp").forward(request,
response);
                        } else {

                request.getRequestDispatcher("/admin/administrarCursos.jsp").forward(request,
response);
                        }
                }
}
```

11.21 /ServletModificarPerm.java

```
package fast1718.trabajo;

import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.sql.DataSource;

@WebServlet(urlPatterns={"/admin/modificaPermisos"})
public class ServletModificarPerm extends HttpServlet {
	private static final long serialVersionUID = 1L;


    /**
     * @see HttpServlet#HttpServlet()
     */
    public ServletModificarPerm() {
        super();
    }
	/**
	 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
	 */
	protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
		String permite_puntua = request.getParameter("permisopuntua");
		String idcurso = request.getParameter("idcurso");

		boolean error = false;
		String mensaje = "";

		if (permite_puntua!= null && !permite_puntua.isEmpty())
		{
			String sql;
			try {
				DataSource ds = (DataSource)
request.getServletContext().getAttribute("ds");
				Connection conn = ds.getConnection();
					sql = "UPDATE permisos set permite_puntua=? where
id_curso=?";
					PreparedStatement st = conn.prepareStatement(sql);
					st.setInt(1, Integer.parseInt(permite_puntua));
					st.setString(2, idcurso);
					int rows = st.executeUpdate();
					if (rows == 1) {
						//Modificación correcta
					} else {
						error = true;
						mensaje = "Error Modificando los permisos";
					}
					st.close();
				}
```

```
                        catch (SQLException e) {
                                error = true;
                                mensaje = "Error con los campos del formulario";
                        }
                } else {
                        error = true;
                        mensaje = "Error en los datos del usuario.";
                }

                if (error) {
                        request.setAttribute("mensaje", mensaje );
                        request.setAttribute("volverURL","/EvalProyecto/admin/permisos.jsp"); //
Si pinchamos se vuelve a la pagina anterior
                        request.getRequestDispatcher("/clientes/error.jsp").forward(request,
response);
                } else {
                        request.getRequestDispatcher("/admin/permisos.jsp").forward(request,
response);


                }

        }

}
```

11.22/ServletModificarUsuario.java

```
package fast1718.trabajo;

import java.io.IOException;

/**
 * Servlet implementation class ServletCrearUsuario
 */
@WebServlet(urlPatterns={"/admin/modificarUsuario"})
public class ServletModificarUsuario extends HttpServlet {
	private static final long serialVersionUID = 1L;

	private static final int LON_MIN = 6;
	private static final int LON_MAX = 10;
	private static final int MIN_MAYUSCULAS = 2;
	private static final int MIN_MINUSCULAS = 2;
	private static final int MIN_NUMERICOS = 2;
	private static final int LON_MAX_USU = 11;

	/**
	 * Verifica que la contraseña y el usuario cumplen los requisitos
	 * @param pass
	 * @param usu
	 * @return Verdadero si cumple.
	 */
	private boolean compruebaContraUsu(String pass) {
		boolean res = false;
		int numericos = 0;
		int mayusculas = 0;
		int minusculas = 0;
		String mayus ="ABCDEFGHYJKLMNÑOPQRSTUVWXYZ";
		String minus ="abcdefghyjklmnñopqrstuvwxyz";
		String nume = "0123456789";

		if ( (pass.length() < LON_MIN) || (pass.length() > LON_MAX)) {

			res = false;

			return res;
		}
		  for (int i = 0; i < pass.length(); i++) {
				if( nume.indexOf(pass.charAt(i),0) !=-1) {
				numericos = numericos + 1;
				}
				if( mayus.indexOf(pass.charAt(i),0) !=-1) {
					mayusculas = mayusculas + 1;
				}
				if( minus.indexOf(pass.charAt(i),0) !=-1) {
					minusculas = minusculas + 1;
				}
		  }
			 if ( numericos < MIN_NUMERICOS || mayusculas <
MIN_MAYUSCULAS || minusculas < MIN_MINUSCULAS ) {
				res = false;
			}
			else {
				res = true;
			}
		return res;
```

```
        }

    /**
     * @see HttpServlet#HttpServlet()
     */
    public ServletModificarUsuario() {
        super();
    }
        /**
         * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
         */
        protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
                String contra = request.getParameter("contra");
                String contra1 = request.getParameter("contra1");
                String contra2 = request.getParameter("contra2");
                boolean error = false;
                String mensaje = "";
                Usuario usuario = new Usuario(); // Hay que crear objeto usuario para luego
usarlo
                usuario = (Usuario) request.getSession().getAttribute("usuario");
                if (contra!= null && contra1!=null && contra2!=null
                                && !contra.isEmpty() && !contra1.isEmpty() &&
!contra2.isEmpty()
                                && contra1.equals(contra2)
                                && compruebaContraUsu(contra1)
                                && !contra1.equals(contra))
                {
                        String sql;
                        try {
                                DataSource ds = (DataSource)
request.getServletContext().getAttribute("ds");
                                Connection conn = ds.getConnection();
                                        sql = "UPDATE usuarios set password =? where
id_usuario =?";
                                        PreparedStatement st = conn.prepareStatement(sql);

                                        st.setString(1, contra1);
                                        st.setString(2, usuario.getUsu());
                                        int rows = st.executeUpdate();
                                        if (rows == 1) {
                                                //Modificación correcta
                                        } else {
                                                error = true;
                                                mensaje = "Error modificando la contraseña";
                                        }
                                        st.close();
                                }
                         catch (SQLException e) {
                                error = true;
                                mensaje = "Error creando el usuario";
                        }
                } else {
                        error = true;
                        mensaje = "Error en los datos del usuario.";
                }

                if (error) {
                        request.setAttribute("mensaje", mensaje );
```

```
		request.setAttribute("volverURL","/EvalProyecto/admin/verUsuarios.jsp"); // Si
pinchamos se vuelve a la pagina anterior
                                request.getRequestDispatcher("/clientes/error.jsp").forward(request,
response);
                        } else {


		request.getRequestDispatcher("/admin/verUsuarios.jsp").forward(request, response);


                        }

		}

}
```

11.23/ServletPuntuar.java

```
package fast1718.trabajo;

import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.Enumeration;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.sql.DataSource;

/**
 * Servlet implementation class ServletPuntuar
 */
@WebServlet(urlPatterns={"/clientes/puntuaCurso"})
public class ServletPuntuar extends HttpServlet {
	private static final long serialVersionUID = 1L;


    /**
     * @see HttpServlet#HttpServlet()
     */
    public ServletPuntuar() {
        super();
    }
	/**
	 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
response)
	 */

	protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
		String cursoidn = request.getParameter("cursoidn");
		int grupodn = Integer.parseInt(request.getParameter("grupodn"));
		String user = request.getParameter("usuario");
		int i=0;
		int x = 1;
		int tam_max=0;
	Enumeration<String> ed = request.getParameterNames();
	while(ed.hasMoreElements())
	{
		Object obj = ed.nextElement();

			tam_max++;
		}
	System.out.println("TAM_MAX= "+tam_max);
	String[] criterios = new String[tam_max-5];
	Enumeration<String> nuevo = request.getParameterNames();
	while(nuevo.hasMoreElements())
	{
```

```
                Object obj2 = nuevo.nextElement();
                String fieldName=(String)obj2;
                String fieldValue=request.getParameter(fieldName);
                if(x>4 && x<=tam_max-1) {
                        criterios[i]=fieldValue;
                        i++;
                }
                x++;
        }
        boolean error = false;
                String mensaje = "";
                String sql;
                int z=0;
                while(z<i) {
                        try {
                                DataSource ds = (DataSource)
request.getServletContext().getAttribute("ds");
                                Connection conn = ds.getConnection();
                                sql = "INSERT INTO puntua_x VALUES (?,?,?,?,?)";
                                PreparedStatement st = conn.prepareStatement(sql);
                                st.setString(1, user);
                                st.setInt(2, grupodn);
                                st.setInt(3, Integer.parseInt(criterios[z]));
                                st.setString(4, cursoidn);
                                st.setInt(5, z);
                                int rows = st.executeUpdate();
                                System.out.println("ROWS= "+rows);
                        }
                        catch (SQLException e) {
                                error = true;
                                mensaje = "Error con los campos del formulario";
                        }
                        z++;
                        }
                try {
                        DataSource ds = (DataSource)
request.getServletContext().getAttribute("ds");
                        Connection conn = ds.getConnection();
                        sql = "INSERT INTO preferente_x VALUES (?,?,?,?)";
                        PreparedStatement st = conn.prepareStatement(sql);
                        st.setString(1, user);
                        st.setInt(2, grupodn);
                        st.setInt(3, Integer.parseInt(request.getParameter("preferencia")));
                        st.setString(4, cursoidn);
                        int rows = st.executeUpdate();
                        System.out.println("ROWS= "+rows);
                }
                catch (SQLException e) {
                        error = true;
                        mensaje = "Error con los campos del formulario";
                }
                if (error) {
                        request.setAttribute("mensaje", mensaje );
                        request.setAttribute("volverURL","/EvalProyecto/clientes/puntuar.jsp"); //
Si pinchamos se vuelve a la pagina anterior
                        request.getRequestDispatcher("/clientes/error.jsp").forward(request,
response);
                } else {
                        request.getRequestDispatcher("/clientes/puntuar.jsp").forward(request,
response);
```

```
                }
        }
}
```

11.24/ServletUsuarios.java

```
package fast1718.trabajo;

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
//import java.util.Map;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.sql.DataSource;

/**
 * Servlet implementation class ServletInterfaces
 */
@WebServlet(urlPatterns={"/admin/verUsuarios"})
public class ServletUsuarios extends HttpServlet {
        private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public ServletUsuarios() {
        super();
    }

        /**
         * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse
response)
         */
        protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
                response.setContentType("application/json");
                PrintWriter out = response.getWriter();
                try {
                //      Usuario usuario = (Usuario)
request.getSession().getAttribute("usuario");

                        DataSource ds = (DataSource)
request.getServletContext().getAttribute("ds");
                        Connection conn = ds.getConnection();


                        String sql = "";
```

```
                        sql = "Select a.id_nombre, a.password, a.id_usuario, c.id_nombrecurso,
b.id_curso, b.num_grupo from usuarios a, curso_x b, cursosgrupo c where
a.id_usuario=b.id_usuario and c.id_curso=b.id_curso";
                        PreparedStatement st = conn.prepareStatement(sql);
                        ResultSet rs = st.executeQuery();
                        out.println("[");
                        boolean hayDato = rs.next();
                        while (hayDato) {
                                out.print("  {");
                                String id_usuario = rs.getString(3);
                                String password = rs.getString(2);
                                String id_nombre = rs.getString(1);
                                String id_nombrecurso = rs.getString(4);
                                String id_curso = rs.getString(5);
                                int grupo = rs.getInt(6);
                                out.print("\"nombre\": \""+id_nombre+"\",");
                                out.print("\"password\": \""+password+"\",");
                                out.print("\"uvus\": \""+id_usuario+"\",");
                                out.print("\"curso\": \""+id_nombrecurso+"\",");
                                out.print("\"id curso\": \""+id_curso+"\",");
                                out.print("\"grupo\": \""+grupo+"\"");
                                out.print("}");
                                hayDato = rs.next();
                                if (hayDato)
                                        out.println(",");
                        }
                        out.println("\n]");

                        rs.close();
                        st.close();
                        conn.close();

                } catch (SQLException e) {
                        out.println("[]");
                        e.printStackTrace();
                        System.out.println("Error de acceso a la base de datos.");
                }
        }
}
```

11.25 /Usuarios.java

```
package fast1718.trabajo;


public class Usuario {

	private String usu;

	public String getUsu() {
		return usu;
	}

	public void setUsu(String usu) {
		this.usu = usu;
	}

	private String contra;

	public String getContra() {
		return contra;
	}

	public void setContra(String contra) {
		this.contra = contra;
	}

	private int tipo_usu;

	public int getTipo_usu() {
		return tipo_usu;
	}

	public void setTipo_usu(int tipo_usu) {
		this.tipo_usu = tipo_usu;
	}

	public static final int ADMINISTRADOR = 0;

	public static final int CLIENTE = 1;

	private String name;

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}
}
```