



UNIVERSIDAD
DE MÁLAGA



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADUADA/O EN INGENIERÍA INFORMÁTICA

ASIGNACIÓN DE ESTUDIANTES A GRUPOS DE UN CENTRO UNIVERSITARIO

ASSIGNMENT OF STUDENTS TO GROUPS OF AN UNIVERSITY CENTER

Realizado por
Alejandro Arco Puerto
Tutorizado por
Enrique Soler Castillo
José Francisco Chicano García

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, JULIO DE 2022

Fecha defensa: Julio de 2022

Resumen

La ETSI Informática de la UMA, es uno de los pocos centros donde se le permite a los alumnos solicitar grupos distintos en diferentes cursos, al ser muy común el estar matriculado en asignaturas de diferentes cursos. Esta estrategia se adopta para evitar que les coincidan horarios a los estudiantes. El proceso de selección de los grupos por parte de los alumnos y la posterior actualización de los grupos seleccionados por parte de secretaria resulta un proceso muy tedioso y complicado.

Aquí es donde surge la idea del desarrollo de este proyecto. El proyecto consiste en una aplicación web visual e intuitiva, pensada para personas que no tienen conocimientos informáticos previos y en la que se podrán visualizar, editar, eliminar, crear, importar y exportar datos respectivos a la asignación y, además, ejecutar procesos para llevar a cabo la asignación y conseguir facilitar y automatizar este proceso tan complicado para la secretaria del centro.

La metodología utilizada en el desarrollo de la aplicación ha sido una metodología ágil con desarrollo iterativo e incremental, consiguiendo al final la aplicación completa. Las tecnologías utilizadas para implementar el proyecto han sido: la plataforma Java EE, el framework PrimeFaces, Oracle para la base de datos y el servidor de aplicaciones WildFly. Todo esto en conjunto forman una aplicación muy adecuada para lo que se buscaba.

Palabras clave: Aplicación Web, Aplicación Intuitiva , Secretaría, Automatización, Java EE, PrimeFaces, Oracle

Abstract

ETSI Informatica of UMA, is one of the few centers where students are allowed to request different groups in different courses, because it is very common to be enrolled in subjects of different courses. This strategy is adopted to avoid the collision of the students shedule. The process of selecting groups by students and the subsequent update of the selected groups by the secretary is a very tedious and difficult process.

This is where the idea of developing this project arises. The project consists in a visual and intuitive web application designed for people that have no deep computer science skills and where we can read, edit, delete, create, import and export data related with the assignment and also execute processes to carry out the assignment in order to achieve the main objective of facilitating and automating this complicated process for the secretary of the center.

The methodology used in the development of this application has been an agile methodology with incremental development, finally getting the complete web application. The technologies used to implement the project have been the Java EE platform, PrimeFaces, the Oracle technology for databases and the Wildfly application server. All this together form a suitable application for the purpose.

Keywords: Web Application, Intuitive Application, Secretary, Automation, Java EE, PrimeFaces, Oracle

Índice

Resumen	1
Abstract	1
Índice	1
Introducción	1
1.1. Motivación	1
1.2. Objetivos	2
1.3. Metodología	3
1.4 Estructura de la memoria	5
Tecnologías y herramientas a utilizar	7
2.1. Java EE.....	7
2.2. Wildfly.....	9
2.3. Primefaces	9
2.4. Maven	11
2.5. CSS	13
2.6. JavaScript	13
2.7. Oracle Database.....	14
2.8. Oracle SQL Developer.....	14
2.9. Visual Studio Code	15
2.10. GitHub	15
2.11. Docker	16
Análisis de requisitos	19
3.1. Requisitos funcionales.....	19
3.2. Requisitos no funcionales.....	24
3.3 Casos de uso	25
Diseño de la aplicación	39
4.1. Arquitectura de la aplicación	39
4.2. Proyecto Maven.....	41
4.3. Proyecto Base de Datos.....	44
Desarrollo	53
5.1. Primera iteración	53
5.2. Segunda iteración	64
5.3. Tercera iteración.....	69
5.4. Cuarta iteración	71
5.5. Mejoras	75
Pruebas	79
6.1. Iniciar Sesión.....	79
6.2. Recuperar la contraseña.....	80
6.3. Cambiar contraseña	81
6.4. Gestionar Entidad	82

6.4.1. Crear una entidad	82
6.4.2. Editar una entidad	84
6.4.3. Eliminar una entidad.....	85
6.4.5. Importar datos.....	87
6.5. Visualizar Datos	88
6.5.1. Visualizar lista de datos	88
6.5.2. Visualizar Entidad	88
6.5.3. Filtrar datos.....	89
6.6. Ejecutar procesos.....	90
6.7. Resultados obtenidos	92
Conclusiones.....	95
7.1. Conclusiones	95
7.2. Líneas futuras	96
Referencias.....	99
Guía de Instalación	103
A.1. Código necesario	103
A.2. Instalación docker y despliegue	105
Manual de inclusión de nuevos procesos.....	109
B.1. Backing bean	109
B.2. Página JSF.....	111
Manual de usuario	113
C.1. Iniciar Sesión	113
C.2. Recuperar contraseña	113
C.3. Cambiar contraseña.....	115
C.4. Gestionar Datos.....	115
C.5. Visualizar Datos	120
C.6. Ejecutar procesos	121
C.7. Cerrar Sesión	123
C.8. Rol ADMINISTRADOR.....	124

1

Introducción

1.1. Motivación

Es conocido que el trabajo que desempeñan las secretarías de los centros es fundamental para llevar a cabo el correcto funcionamiento académico y su función tiene aún más importancia en los primeros meses del curso, donde suelen surgir más complicaciones con los grupos asignados, las asignaturas escogidas por los alumnos, la coincidencia de horarios, etc.

En la ETSI Informática de la UMA se pretende dar una solución a todos estos problemas recurrentes durante el inicio del curso, dando la posibilidad al alumnado de escoger grupos distintos en diferentes cursos, al ser bastante común que un mismo estudiante pueda estar matriculado en asignaturas de diferentes cursos por lo que hay más posibilidades de que surjan problemas.

Este proceso se lleva a cabo mediante una encuesta, donde el estudiante puede solicitar el grupo que más le convenga y así evitar disconformidades en la asignación. A partir de esta encuesta se obtiene la información con la cual se produce la asignación y la posterior resolución de conflictos en los cambios de grupos. Este proceso es muy meticuloso y complicado al tener en cuenta numerosos requisitos, lo que produce verdaderos quebraderos de cabeza al personal de la secretaría del centro.

Es aquí donde surge la motivación para el desarrollo de este proyecto, con el que se pretende mitigar los contratiempos relacionados con la asignación de los alumnos a grupos, desarrollando una aplicación web intuitiva y fácil de usar dedicada a agilizar todo este proceso.

1.2. Objetivos

El objetivo principal de este trabajo de fin de grado, como se ha comentado con anterioridad, es agilizar todo el proceso de asignación de los alumnos a los grupos y cualquier función que tenga relación con esto. Para esto se ha pensado en desarrollar una aplicación web, en la que se podrán desempeñar tres funcionalidades principales:

- **Gestión de datos:** Esta gestión se realizará a través de tablas, donde se podrán crear datos nuevos, editar los datos, eliminarlos, importar datos nuevos en la aplicación y, además, exportar el contenido de estas tablas.
- **Visualizar datos de la asignación:** Esta parte de la aplicación está dedicada a la visualización de datos. Se realizará al igual que la parte de gestión, con tablas, para así poder visualizar todo lo deseado de un vistazo. Además, se disponen de filtros incorporados para poder obtener el resultado deseado de una forma más eficaz. También se da la posibilidad de exportar los datos de las tablas.
- **Ejecución de procesos:** En este módulo se realizará todo el proceso de asignación, pero de la forma más sencilla y automática posible. Aquí también se podrán obtener los resultados de la asignación y se podrán ver datos más específicos de la asignación y los procesos ejecutados.

Cabe destacar que en la aplicación existirán dos tipos de roles: el usuario y el administrador. La diferencia entre estos dos roles consiste en que el administrador, además de poder realizar las funcionalidades mencionadas anteriormente, tendrá la posibilidad de gestionar los usuarios, pudiendo eliminar, editar o crear nuevos usuarios,

por lo que se puede deducir que será el administrador el encargado de dar el acceso a la aplicación.

A parte de lo comentado, la aplicación estará pensada para personal que no tiene un conocimiento amplio en informática. Esto se logrará desarrollando una interfaz visual, intuitiva, minimalista y con una gran cantidad de ayuda integrada, para que el entendimiento de cada una de las partes que forman la aplicación sea una tarea sencilla.

1.3. Metodología

La metodología elegida para el proyecto ha sido el "Desarrollo Iterativo e Incremental" [12] . Una de las primeras referencias a este tipo de desarrollo aparece en el proyecto de investigación de la Nasa para el avión supersónico X-15, donde se menciona explícitamente que una de las contribuciones al éxito del X-15 fue el énfasis en este tipo de desarrollo, lo cual ya hace pensar que esta metodología es muy acertada para el desarrollo de productos. Para entender mejor esta metodología es conveniente explicar cada una de ellas por separado:

- Un desarrollo incremental, es aquel en el que, con cada entrega que se realiza se tiene acabada una pieza más del sistema, pero este no se supone acabado hasta que no es colocada la última pieza.

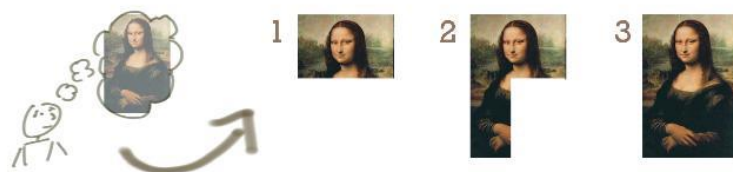


Figura 1.3.1: Desarrollo incremental.

Fuente[<https://itsadeliverything.com/revisiting-the-iterative-incremental-mona-lisa>]

- Un desarrollo iterativo, es aquel en el que con cada entrega que se realiza se espera la opinión del usuario y según su opinión se deciden los siguientes pasos en el desarrollo.



Figura 1.3.2: Desarrollo iterativo.

Fuente[<https://itsadeliverything.com/revisiting-the-iterative-incremental-mona-lisa>]

Tras ver cada una de ellas por separado, se entiende mucho mejor que el desarrollo iterativo e incremental se define como aquel con el que con cada entrega se añaden funcionalidades totalmente nuevas, pero también se producen mejoras en las funcionalidades que ya existían en el sistema. Esto supone dos ventajas principales y es que se puede aportar conocimiento al usuario sobre el desarrollo del proyecto y aparte se puede ir adaptando el sistema según lo que se vaya aprendiendo durante el desarrollo.



Figura 1.3.3: Desarrollo iterativo e incremental.

Fuente[<https://itsadeliverything.com/revisiting-the-iterative-incremental-mona-lisa>]

Según esta metodología el proceso de desarrollo ha estado formado por una serie de ciclos o iteraciones en las cuales se comenzó con un análisis de requisitos inicial, con el objetivo de tener claro desde un primer momento qué requisitos eran esenciales para la aplicación. Más tarde, se realizaba una selección de algunos de esos requisitos, se implementaban los requisitos elegidos y se realizaban las pruebas pertinentes para asegurar el buen funcionamiento de esos requisitos. Como última etapa de cada ciclo se mostraban esas nuevas funcionalidades a los tutores (los cuales hacían de clientes) y proponían por su parte mejoras y posibles cambios. Gracias a estos comentarios por parte de los tutores, se tenían distintos puntos de vistas de la aplicación y por ende, se

conseguía mejorar la aplicación en cantidad. Estos ciclos se fueron repitiendo de forma continuado durante el desarrollo hasta alcanzar un proyecto final lo más acertado posible.

1.4 Estructura de la memoria

En este punto se pretende explicar cómo va a ser la estructura de la memoria y que se abarcará en cada capítulo de esta:

- 1. Introducción.** Se realiza un primer acercamiento al lector. Hablando de temas como la motivación que ha llevado al desarrollo de este proyecto, los objetivos de este trabajo, la metodología de desarrollo empleada y la estructuración de la memoria.
- 2. Tecnologías a utilizar.** Se hará un repaso de cada una de las tecnologías utilizadas en este proyecto, explicando en cada caso en que consisten, para que se han utilizado en el proyecto y el porqué de su elección.
- 3. Análisis de requisitos.** Se realiza un estudio de los requisitos funcionales y no funcionales para que la aplicación cumpla todas las expectativas esperadas. Además, se generarán los distintos casos de uso describiendo detalladamente la iteración entre el sistema y el usuario.
- 4. Diseño y modelado de la aplicación.** Se describe el diseño utilizado en la aplicación tanto a nivel de almacenamiento, como a nivel de proyecto.
- 5. Desarrollo.** Se detalla cómo ha sido el proceso de desarrollo de la aplicación, entrando en detalle en cada una de las iteraciones del desarrollo.
- 6. Pruebas realizadas.** Se detallan cada una de las pruebas realizadas en el sistema con el objetivo de asegurar la funcionalidad, el rendimiento y la experiencia del usuario.

- 7. Conclusiones y líneas futuras.** Se concluye el proyecto haciendo un repaso de lo aprendido en él, de las dificultades encontradas en el mismo, las posibles actualizaciones y mejoras que se le puedan implementar.

Referencias. Se realiza una recopilación de cada una de las fuentes de información utilizadas para el desarrollo del proyecto.

Apéndice A. Guía de Instalación. Se detallan los pasos a seguir para instalar y utilizar la aplicación web en cualquier ordenador.

Apéndice B. Manual de inclusión de proceso. Se realizará una explicación detallada de cómo se podrían incluir nuevos procesos en la aplicación web para mejorar el futuro mantenimiento de la aplicación.

Apéndice C. Manual de Usuario. Se describen en una serie de pasos como utilizar cada una de las funcionalidades de la aplicación, con el objetivo de ofrecer una ayuda extra al usuario que decida utilizarla.

2

Tecnologías y herramientas a utilizar

En esta capítulo se hace un repaso de las diferentes tecnologías utilizadas en el desarrollo del proyecto, abordando tanto lenguajes, como herramientas. Todas ellas seleccionadas tras un estudio para conseguir la solución óptima para este proyecto.

2.1. Java EE



Figura 2.1.1: Logo JAVA EE

Java EE [1] es una plataforma formada por un conjunto de especificaciones, que se centra en el desarrollo de aplicaciones empresariales. Java EE sigue un modelo de diseño dividido en niveles: la capa de cliente, la capa web, la capa de negocio y la capa de sistemas de información.

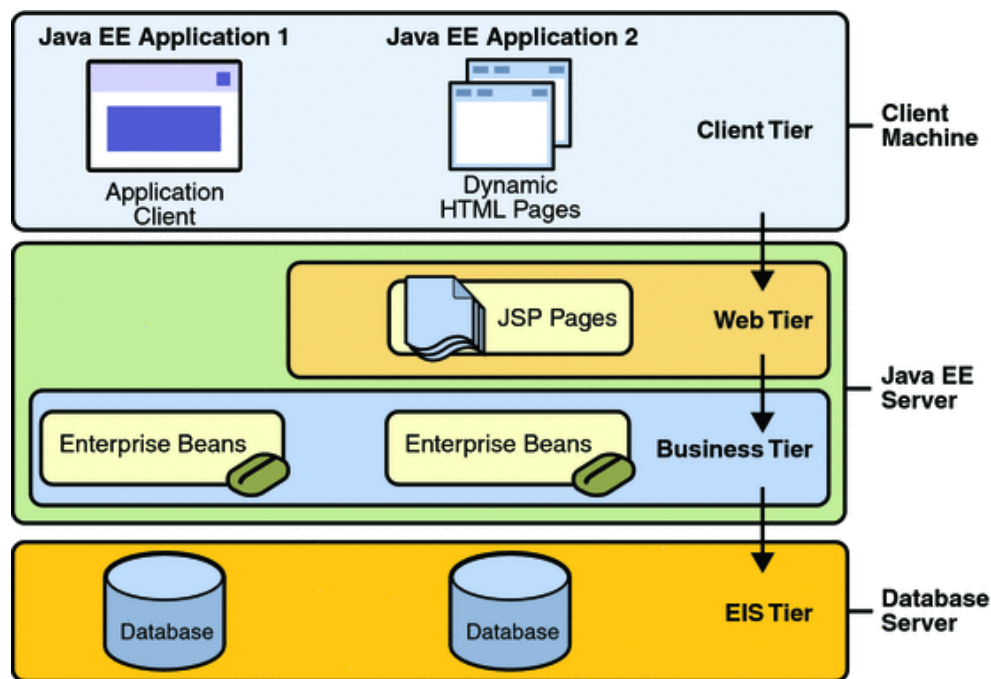


Figura 2.1.2: Diseño en capas Java EE

Esta plataforma se basa en el lenguaje Java [14], el cual es un lenguaje de propósito general que es capaz de ejecutarse en múltiples plataformas y de adaptarse a cualquier proyecto. Esto lo consigue gracias a su máquina virtual, que existe en cada sistema que sea capaz de ejecutar Java. El hecho de ser multiplataforma es una ventaja de gran peso para el desarrollo software, ya que no se tendrá que desarrollar un programa para cada plataforma o sistema operativo. Con Java se puede desarrollar cualquier tipo de proyecto, pero en este caso se ha aprovechado para el desarrollo de una aplicación web, aprovechando su fiabilidad y robustez. Cabe destacar que es uno de los lenguajes más utilizados en el mundo empresarial, si se analiza el TIOBE Programming Community Index, indicador de popularidad de lenguajes de programación, el cual está basado en más líneas de código escritas, nos encontramos a Java en la tercera posición:

May 2022	May 2021	Change	Programming Language		Ratings	Change
1	2	▲		Python	12.74%	+0.86%
2	1	▼		C	11.59%	-1.80%
3	3			Java	10.99%	-0.74%

Figura 2.1.3: TIOBE Index Java. Fuente[<https://www.tiobe.com/tiobe-index/>]

Esta plataforma ha sido escogido debido a que era conocida de la asignatura Sistemas de Información para Internet, donde se explicó en detalle y se facilitó su entendimiento. Además, al utilizar el lenguaje Java ha sido más fácil el desarrollo, ya que este lenguaje es bastante utilizado en el grado.

2.2. Wildfly



Figura 2.2.1: Logo de Wildfly

Wildfly [11] es un servidor de código abierto de aplicaciones Java EE. Este tipo de servidores resultan muy útiles para crear, implementar y hospedar aplicaciones Java. Su ventaja principal es que al estar basado en Java puede usarse en cualquier sistema operativo que tenga disponible la máquina virtual Java. Wildfly arranca bastante rápido y cuenta con la licencia de software libre, está basado en dos contenedores (contenedor ejb y contendor web) y cada uno de ellos en principio es independiente del otro.

La razón de la elección de este servidor de aplicaciones ha sido porque aparte de estar destinado a aplicaciones Java EE, se conocía de la asignatura Sistemas de Información para Internet, donde se utilizó para desplegar una aplicación que se desarrolló durante el curso.

2.3. Primefaces



Figura 2.3.1: Logo de Primefaces.

Cuando se comienza a utilizar la tecnología JSF para la vista de una aplicación web los componentes que incluye JSF son poco visuales y funcionales. Aquí es donde interviene

la biblioteca de componentes Primefaces dedicada a facilitar la creación de las vistas de las páginas web basadas en Java EE.

Primefaces [2] es una biblioteca ligera de código abierto, utilizada para el desarrollo de aplicaciones empresariales Java EE. Esta biblioteca nos proporciona un gran número de componentes y temas que mejoran en gran medida la visualización y versatilidad de las páginas Java Server Faces.

Esta no depende de otras bibliotecas de terceros y no necesita ningún tipo de configuración extra, se distribuye a través de un único .jar. En nuestro caso, al utilizar Maven con una simple dependencia en el POM se incluirá automáticamente en el proyecto:



```
<!-- https://mvnrepository.com/artifact/org.primefaces/primefaces
Para utilizar elementos primefaces-->
<dependency>
  <groupId>org.primefaces</groupId>
  <artifactId>primefaces</artifactId>
  <version>11.0.0</version>
</dependency>
```

Figura 2.3.2: Dependencia necesaria para utilizar Primefaces

Algunas de las características más interesantes que nos ofrece Primefaces son:

- Uso de JavaScript y CSS no intrusivo, es decir no aparecen líneas de código de JavaScript, ni de CSS dentro de los elementos.
- Proyecto open source, activo y estable.
- Compatible con otras librerías de componentes.
- Soporte nativo de Ajax
- Mas de 100 componentes.
- Compatible con todos los navegadores modernos.
- Incluye un kit para el desarrollo de aplicaciones en teléfonos móviles.

Esta biblioteca ha sido elegida por todas las facilidades que proporciona. Aparte tiene una gran comunidad en la cual se resuelven todo tipo de dudas, lo que facilita el desarrollo. Además, es de uso gratuito y le da a la vista un enfoque más profesional.

2.4. Maven



Figura 2.4.1: Logo de Maven.

Maven[1] es una herramienta software dedicada a la gestión y construcción de proyectos Java y estandariza la configuración de los proyectos en todo su ciclo de vida. Maven se sirve del Project Object Model (archivo POM que se define en cada proyecto Maven y en cada uno de los módulos que forman el proyecto) para describir el proyecto a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos. Algunas de sus características principales son:

- Es multiplataforma, nos permite utilizarlo tanto en sistemas operativos Linux como en Windows al ser una aplicación Java.
- Es compatible con muchos IDEs.
- Tiene un sistema de gestión de dependencias muy potente.
- Posibilidad de la incorporación de plugins, lo cual lo hace más adaptable.

Estas características se han podido ver reflejadas durante el desarrollo:

- Se tuvieron problemas con el entorno de desarrollo Eclipse, lo que nos llevó al cambio de Eclipse por Visual Studio Code, aquí nos pudimos beneficiar de las ventajas de Maven y su versatilidad, ya que no tuvimos impedimentos en el cambio.
- Se han tenido que incluir gran cantidad de bibliotecas, lo cual fue muy sencillo gracias a la existencia del pom.xml, donde a través de la página [MVNRepository](https://mvnrepository.com/) (repositorio remoto Maven) se puede obtener la dependencia necesaria a incluir en el proyecto, se copia el código en el pom y así ya se puede utilizar (al descargarse automáticamente un .jar en el proyecto):



Apache POI Common » 5.2.2

Apache POI - Java API To Access Microsoft Format Files

License	Apache 2.0
Categories	Excel Libraries
Organization	Apache Software Foundation
HomePage	https://poi.apache.org/
Date	(Mar 20, 2022)
Files	pom (3 KB) jar (2.8 MB) View All
Repositories	Central
Used By	2,042 artifacts

Maven
Gradle
Gradle (Short)
Gradle (Kotlin)
SBT
Ivy
Grape
Leiningen
Buildr

```

<!-- https://mvnrepository.com/artifact/org.apache.poi/poi -->
<dependency>
  <groupId>org.apache.poi</groupId>
  <artifactId>poi</artifactId>
  <version>5.2.2</version>
</dependency>
  
```

☒ Include comment with link to dependency

Figura 2.4.2: Dependencia en la página MVNRepository de Apache POI para poder manejar ficheros.
Fuente[<https://mvnrepository.com/>]

```

<dependency>
  <groupId>org.apache.poi</groupId>
  <artifactId>poi</artifactId>
  <version>5.2.2</version>
</dependency>
  
```

Figura 2.4.3: Ejemplo de dependencia incluida en el pom.xml¹

Maven también da la posibilidad de dividir el proyecto en módulos EAR, EJB y WAR, lo que después nos beneficiará a la hora de desplegar la aplicación en el servidor de aplicaciones. Todo ello gracias a la existencia de los plugins como se ha mencionado, que nos permite en este caso generar el archivo EAR para el servidor de aplicaciones.

```

<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-ear-plugin</artifactId>
      <configuration>
        <defaultLibBundleDir>lib</defaultLibBundleDir>
        <modules>
          <ejbModule>
            <groupId>es.uma.informatica.sii</groupId>
            <artifactId>Secretaria-EJB</artifactId>
          </ejbModule>
          <webModule>
            <groupId>es.uma.informatica.sii</groupId>
            <artifactId>Secretaria-WAR</artifactId>
          </webModule>
        </modules>
      </configuration>
    </plugin>
  </plugins>
</build>
  
```

Figura 2.4.4: Ejemplo del uso de plugins Maven

¹ Las figuras no referenciadas son fuente propia o ya se incluye la fuente en el texto.

Esta herramienta se eligió por las numerosas facilidades que da al desarrollo y al despliegue como ya se ha visto y, además, se utilizó asiduamente en la asignatura Sistemas de Información para Internet, por lo que era una herramienta conocida.

2.5. CSS



Figura 2.5.1: Logo de CSS.

CSS o Cascading Style Sheets [4] es un lenguaje que permite definir la presentación y el estilo de los diferentes elementos HTML dentro de la página web. Está diseñado para realizar una separación entre el contenido de la página y la forma de presentación de esta.

Este lenguaje no ha sido a penas necesario debido a Primefaces, que ya incorpora CSS en sus componentes, pero en ocasiones se han querido modificar aspectos como el tamaño de una imagen o el formato del texto, lo cual ha requerido su uso.

2.6. JavaScript



Figura 2.6.1: Logo de JavaScript.

JavaScript [5] es un lenguaje de programación ligero e interpretado que se utiliza como complemento de HTML y CSS para crear páginas webs. Este lenguaje permite dotar a la página de una mayor interactividad y dinamismo.

Este lenguaje se ha escogido porque proporciona a la aplicación algunas características interactivas que la hacen más visual y entendible. No se ha usado en cantidad, al igual que CSS, porque muchos de los componentes de Primefaces ya lo incluyen.

2.7. Oracle Database

Oracle Database [17] es una base de datos de tipo objeto relacional de un alto rendimiento que proporciona protección a los usuarios frente a accesos no autorizados y facilita la recuperación ante problemas mayores. Esta ha sido la base de datos seleccionada por su asidua utilización durante el grado y, además, todos los procedimientos y entidades proporcionados para realizar la asignación se proporcionaban en esta base de datos.

2.8. Oracle SQL Developer



Figura 2.8.1: Logo SQL Developer

Oracle SQL Developer [6] es el IDE de la base de datos de Oracle. Proporciona una interfaz gráfica que permite a los usuarios de la base de datos realizar consultas SQL, programar procedimientos y funciones en PL/SQL, exportar datos al formato deseado, y muchas más funciones. Se ha escogido debido a que simplifica las tareas de desarrollo de bases de datos y, además, se conocía en profundidad por su uso en todas las asignaturas relacionadas con bases de datos del grado.

2.9. Visual Studio Code

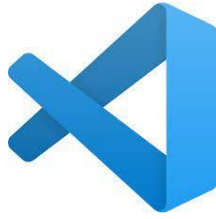


Figura 2.9.1: Logo Visual Studio Code

Visual Studio Code [7] es un editor de código desarrollado por Microsoft. Este es compatible con casi todos los sistemas operativos y es gratuito. Para el desarrollo de la aplicación este ha sido el editor seleccionado por sus innumerables ventajas como los atajos de teclado que facilitan y agilizan la programación. Además incluye control integrado de Git, resaltado de sintaxis y muchas más funcionalidades que hacen que sea un editor óptimo para el desarrollo.

2.10. GitHub



Figura 2.10.1: Logo de GitHub

GitHub [8] es una plataforma de desarrollo colaborativo utilizada para alojar proyectos principalmente informáticos y que utiliza el sistema de control de versiones Git. Esta plataforma se ha escogido porque asegura y agiliza el desarrollo de las aplicaciones, ya que se pueden generar nuevas versiones de la aplicación y se permite navegar entre estas versiones. También se pueden crear nuevas ramas de trabajo, lo cual es interesante para no fastidiar el código ya probado. Todo esto se puede realizar con unos simples comandos sencillos que mejoran el proceso de desarrollo.

2.11. Docker

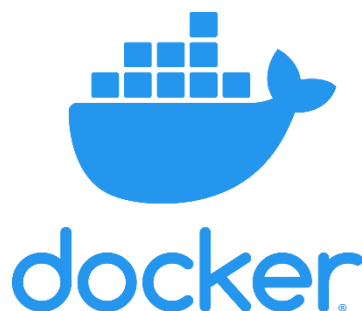


Figura 2.11.1: Logo de Docker.

Docker [9] es una plataforma de gestión de contenedores. Docker empaqueta el software en contenedores donde se incluye todo lo que necesita el software, como bibliotecas, herramientas del sistema, tiempo de ejecución, etc. Se podría asimilar a una máquina virtual donde en vez de virtualizar el hardware, los contenedores virtualizan el sistema operativo. Docker se puede ejecutar de manera fiable en cualquier lugar, lo cual facilita la portabilidad y la usabilidad. Además, proporciona unos comandos sencillos con los que se pueden crear, arrancar o parar los contenedores.

En nuestro caso se ha hecho uso de Docker Compose [10] que se trata de una herramienta para definir y correr aplicaciones con múltiples contenedores. Mediante comandos se pueden crear y arrancar todos los servicios de nuestra aplicación. Para utilizar Docker Compose hay que realizar un proceso de tres pasos:

1. La creación de un Dockerfile, que se puede asimilar a una receta para montar una imagen, donde se ha partido de una imagen ya existente "jboss/wildfly" y se han ido añadiendo instrucciones para adaptar la imagen a las necesidades de la aplicación:

```
1 FROM jboss/wildfly:18.0.1.Final
2 # Se añade un usuario para la administración
3 RUN /opt/jboss/wildfly/bin/add-user.sh admin Admin#70365 --silent
4 # Se exponen los puertos
5 EXPOSE 8080
6 EXPOSE 9990
7 EXPOSE 8787
8 #Directorio para la conexión con la bd
9 RUN mkdir -p /opt/jboss/wildfly/modules/system/layers/base/com/oracle/main
10 #Icono para la aplicación
11 RUN rm /opt/jboss/wildfly/welcome-content/favicon.ico
12 COPY favicon.ico /opt/jboss/wildfly/welcome-content
13 #ojdbc y module.xml para la conexión con la bd
14 COPY module.xml ojdbc8.jar /opt/jboss/wildfly/modules/system/layers/base/com/oracle/main/
15 #Se copian los archivos de configuración de wildfly
16 COPY standalone.xml standalone-full.xml /opt/jboss/wildfly/standalone/configuration/
17 #Se ejecuta el comando al inicio con los valores necesarios para arrancar wildfly
18 CMD ["/opt/jboss/wildfly/bin/standalone.sh", "-b", "0.0.0.0", "-bmanagement", "0.0.0.0", "-c", "standalone-full.xml", "--debug", ".*:8787"]
```

Figura 2.11.2: Dockerfile para Wildfly

2. Se definen los servicios que forman nuestra aplicación en el docker-compose.yml. En este caso solo tenemos uno, aunque como ya se ha dicho el Compose nos permite manejar varios servicios:



```
1 version: '3'
2 services:
3   contwildfly:
4     build: ./wildfly
5     ports:
6       - "8080:8080"
7       - "9990:9990"
8       - "8787:8787"
```

Figura 2.11.3: docker-compose.yml

3. Se ejecuta docker-compose up (con la opción --build para construirlo si es la primera vez) y ya se desplegaría el contenedor wildfly.

Docker se ha utilizado porque facilita el despliegue de la aplicación y aporta una gran portabilidad, además se había utilizado en la asignatura Sistemas de Información para Internet, donde también se trabajó con el Docker Compose.

Análisis de requisitos

Este capítulo se va a centrar en agrupar cada una de las necesidades que el sistema debe satisfacer para conseguir una aplicación lo más acertada posible. Para comenzar se hará un repaso de los requisitos tratando tanto los requisitos funcionales, como los requisitos no funcionales y por último, los casos de uso.

3.1. Requisitos funcionales

Este tipo de requisitos describen cualquier actividad que el sistema debe realizar, es decir el comportamiento que debe tener el sistema cuando se cumplen unas ciertas condiciones. Puede ser un cálculo, una interacción del usuario, una manipulación de datos o cualquier otra funcionalidad.

Cada uno de estos requisitos tendrá un código de la forma RF-XX (RF haciendo alusión a Requisito Funcional y XX es el número de dicho requisito), un título y una descripción que explique en que consiste el requisito.

CÓDIGO	TÍTULO	DESCRIPCIÓN
RF-01	Iniciar sesión	El usuario podrá iniciar sesión en la aplicación, a través de un usuario y contraseña.
RF-02	Cerrar sesión	El usuario podrá cerrar sesión en su cuenta.

RF-03	Gestionar titulaciones	El sistema podrá mostrar una lista de todas las titulaciones donde se tendrá la posibilidad de filtrar la información. Además, se podrán crear, modificar y eliminar titulaciones. Se podrá exportar información de las titulaciones a formato EXCEL y también se podrá importar información de las titulaciones.
RF-04	Gestionar asignaturas	El sistema podrá mostrar un listado de todas las asignaturas donde se tendrá la posibilidad de filtrar la información. Además, se podrá crear, modificar y eliminar asignaturas. Se podrá exportar información de las asignaturas a formato EXCEL y también se podrá importar información de las asignaturas.
RF-05	Gestionar grupos	El sistema podrá mostrar un listado de todos los grupos donde se tendrá la posibilidad de filtrar la información. Además, se podrá crear, modificar y eliminar grupos. Se podrá exportar información de los grupos a formato EXCEL y también se podrá importar información de las asignaturas.
RF-06	Gestionar aulas	El sistema podrá mostrar un listado de todas las aulas donde se tendrá la posibilidad de filtrar la información. Además, se podrá crear, modificar y eliminar aulas. Se podrá exportar información de las aulas a formato

		EXCEL y también se podrá importar información de las aulas.
RF-07	Gestionar grupos asignaturas	El sistema podrá mostrar un listado de todos los grupos de asignaturas donde se tendrá la posibilidad de filtrar la información. Además, se podrá crear, modificar y eliminar grupos asignatura. Se podrá exportar información de los grupos de asignaturas a formato EXCEL y también se podrá importar información de estos grupos asignatura.
RF-08	Visualizar matrículas	El sistema mostrará un listado de todas las matrículas, se podrá filtrar la información entre ellas y también se podrá exportar información de las matrículas.
RF-09	Visualizar alumnos	El sistema mostrará un listado de todos los alumnos, se podrá filtrar la información entre ellos y también se podrá exportar información de los alumnos.
RF-10	Visualizar encuestas de las asignaturas en inglés de los alumnos nuevo ingreso	El sistema mostrará un listado de todas las encuestas de selección de asignaturas en inglés de los alumnos de nuevo ingreso, se podrá filtrar la información entre ellas y también se podrá exportar información de las encuestas.

RF-11	Visualizar las encuestas de las asignaturas escogidas de los alumnos veteranos	El sistema mostrará un listado de todas las encuestas de selección de asignaturas de los alumnos veteranos, se podrá filtrar la información entre ellas y también se podrá exportar información de las encuestas.
RF-12	Visualizar la asignación de grupo de cada alumno	El sistema mostrará un listado de la asignación de cada alumno, se podrá filtrar la información y también se podrá exportar información de la asignación.
RF-13	Visualizar el número de alumnos asignados por asignatura, grupo y titulación	El sistema mostrará un listado del número de alumnos asignados, se podrá filtrar la información y también se podrá exportar información del número de alumnos asignados.
RF-14	Visualizar el número de alumnos y número de grupos por cada asignatura y titulación	El sistema mostrará un listado del número de alumnos asignados y el número de grupos, se podrá filtrar la información y también se podrá exportar información del número de alumnos asignados y el número de grupos.
RF-15	Ejecutar procesos que lleven a cabo la asignación	El sistema será capaz de llevar a cabo la asignación de cada alumno según unos criterios establecidos por la secretaría del centro y teniendo en cuenta los datos de todos los alumnos matriculados y las encuestas de selección tanto de los alumnos de nuevo ingreso, como los veteranos. Además se llevará un control de la

		ejecución de estos procesos sabiendo en todo momento la fecha y el usuario que ejecutó ese proceso.
RF-16	Importar datos necesarios para la ejecución de la asignación	El sistema será capaz de importar datos de los alumnos matriculados y de las encuestas tanto de alumnos veteranos como de nuevo ingreso.
RF-18	Visualizar datos relacionados con la ejecución de procesos	El sistema será capaz de mostrar un listado de todos los datos necesarios para la ejecución como los posibles errores que hayan surgido en la ejecución de los procesos, el reparto de las asignatura grupo, etc.
RF-19	Exportar resultados finales de la asignación	El sistema dará la posibilidad de poder exportar los resultados de la asignación a formato EXCEL.
RF-20	Gestionar Usuarios	Solo el administrador de la aplicación podrá acceder a una sección dedicada a gestionar los usuarios, donde podrá visualizar, crear, eliminar o editar los datos de los usuarios.
RF-21	Recuperar la contraseña	Cualquier usuario de la aplicación, tendrá la posibilidad de recuperar su contraseña. Esta se recuperará mediante el envío de un correo con su nueva contraseña.
RF-22	Cambiar la contraseña	Cualquier usuario de la aplicación podrá cambiar su contraseña, con la condición de que debe saber la contraseña actual.
RF-23	Notificar errores	El sistema incorporará un sistema de notificaciones que informarán al

		usuario de los posibles errores cometidos en el uso.
--	--	--

Tabla 3.1.1: Requisitos funcionales

3.2. Requisitos no funcionales

Este tipo de requisitos no se refieren a las funciones que el sistema debe realizar como el anterior punto, sino a las propiedades que el sistema debe incluir en su funcionamiento. Al igual que en los requisitos funcionales se seguirá una anotación con un código único RNF-XX, un título y una descripción del requisito.

CÓDIGO	TÍTULO	DESCRIPCIÓN
RNF-01	Secciones de ayuda	El sistema incluye ayudas en cada una de las partes de la interfaz que puedan crear dudas al usuario.
RNF-02	Robustez del sistema	La aplicación será capaz de comportarse de una forma razonable aún en circunstancias no anticipadas por los requerimientos.
RNF-03	Interfaz minimalista y visual	La interfaz será implementada de forma que cualquier usuario sin conocimientos técnicos sea capaz de entenderla.
RNF-04	Manual de usuario	La aplicación incorporará un manual de usuario escrito en español que explicará el funcionamiento y uso de la misma.
RNF-05	Manual de instalación	La aplicación incluirá un manual de instalación escrito en español en el que se explicará paso a paso como ponerla en funcionamiento.

RNF-06	Conexiones simultáneas	El sistema será capaz de recibir y soportar las conexiones simultáneas, comportándose de forma adecuada ante ellas.
RNF-07	Bajo tiempo de respuesta	El sistema responderá a cualquier tipo de petición en un tiempo menor a los 30 segundos.
RNF-08	Roles en la aplicación	El sistema incorporará dos tipos de roles el usuario y el administrador.
RNF-9	Credenciales cifradas	El sistema cifrará todo tipo de credenciales para evitar poner en riesgo el acceso a la aplicación de usuarios no autorizados.
RNF-10	Portabilidad	El sistema podrá utilizarse en cualquier ordenador que disponga de una conexión a internet.

Tabla 3.2.1: Requisitos no funcionales

3.3 Casos de uso

Los casos de uso

[13] se podrían definir como una descripción de toda interacción entre el usuario y el sistema. Estas interacciones son como peticiones al sistema y los casos de usos describen como el sistema responde a esas peticiones. Los casos de uso son una metodología que ayuda a entender y a organizar los requisitos de un sistema, por lo tanto son esenciales en el desarrollo de un proyecto. Los componentes clave para la definición de los casos de uso son:

- **Identificador:** Dedicado a identificar el caso de uso al igual que se hacía con los requisitos, será de la forma CU-XX, donde XX es el número del caso de uso.
- **Título o nombre:** Indica el resultado del objetivo que se quiere cumplir en el caso de uso.

- **Actores:** Cualquier cosas o persona que lleva a cabo alguna función en un escenario determinado.
- **Requisitos:** Requisitos involucrados en el caso de uso en cuestión.
- **Precondiciones:** Condiciones que deberá cumplirse antes del comienzo del caso de uso.
- **Postcondiciones:** Lo que ocurre después de haberse cumplido el caso de uso.
- **Escenario de éxito:** Flujo normal del caso de uso en caso de que todo concurra según lo previsto.
- **Escenario alternativo:** Posibles variaciones del flujo normal del caso de uso. Estas excepciones ocurren normalmente cuando las cosas no se han hecho bien en el sistema.

Identificador	CU-01
Título	Iniciar Sesión
Actores	Usuario y sistema
Requisitos	RF-01
Precondiciones	La aplicación debe estar iniciada y mostrando la pantalla inicial de iniciar sesión.
Postcondiciones	El usuario pasa a la página principal del sistema o el sistema no permite el acceso a la aplicación por la recepción del algún error.
Escenario de éxito	1. El usuario introduce el usuario que utiliza en el sistema. 2. El usuario introduce la contraseña que emplea en el sistema. 3. El usuario pulsa el botón de "Entrar". 4. El sistema comprueba que los datos introducidos son los

	correctos y lleva al usuario a la página principal del sistema.
Escenario alternativo	Credenciales incorrectas: 1. El usuario introduce datos que no se corresponden con ninguna cuenta registrada en el sistema. 2. El sistema notifica al usuario con el mensaje "Error. Credenciales inválidas". No se introducen datos en los campos: 1. El usuario pulsa el botón de "Entrar" sin introducir datos en los campos de usuario y contraseña. 2. El sistema notifica con un error de "Usuario necesario y contraseña necesario".

Tabla 3.3.1: CU-01 Iniciar Sesión

Identificador	CU-02
Título	Cerrar Sesión
Actores	Usuario y sistema
Requisitos	RF-02
Precondiciones	El usuario ha iniciado sesión en la aplicación y está dentro de alguna de las páginas que disponen del botón "Salir"
Postcondiciones	El usuario pasa a la página de iniciar sesión

Escenario de éxito	1. El usuario selecciona el botón de "Salir". 2. El sistema lleva al usuario a la página de iniciar sesión y cierra la sesión del usuario.
Escenario alternativo	Ningún escenario de interés

Tabla 3.3.2: CU-02 Cerrar Sesión

Identificador	CU-03
Título	Crear Entidad ²
Actores	Usuario y sistema
Requisitos	RF-03, RF-04, RF-05, RF-06, RF-07, RF-20
Precondiciones	El usuario ha iniciado sesión y se encuentra en alguna de las páginas disponibles para gestionar una entidad.
Postcondiciones	El usuario ha logrado crear una entidad o recibe un error por parte del sistema.
Escenario de éxito	1. El usuario pulsa el botón de "Nuevo". 2. El sistema carga una ventana dedicada a rellenar los datos de la nueva entidad. 3. El usuario introduce los datos de cada atributo de la entidad. 4. El usuario pulsa el botón de "Aceptar"

² Titulación, Asignatura, Grupo, Grupos Asignaturas o Aula

	5. El sistema comprueba que los datos son correctos, muestra un mensaje avisando de la creación. 6. El sistema carga de nuevo la lista de las entidades con la nueva entidad ya incluida.
Escenario alternativo	Entidad ya existente: 1. El usuario introduce datos en los campos de entidades que ya existen en la base de datos. 2. El sistema notifica al usuario con un mensaje de error diciendo "Datos erróneos entidad ya existente". No se introducen datos en los campos requeridos: 1. El usuario no introduce datos en los campos. 2. El sistema notifica que campos son obligatorios introducir marcándolos con una sombra roja y mantiene la ventana de creación hasta que se cierre, se pulse "Cancelar" o se decidan introducir datos.

Tabla 3.3.3: CU-03 Gestionar Entidad (Crear)

Identificador	CU-04
Título	Editar entidad
Actores	Usuario y sistema

Requisitos	RF-03, RF-04, RF-05, RF-06, RF-07, RF-20
Precondiciones	El usuario ha iniciado sesión y se encuentra en alguna de las páginas dedicadas a gestionar las entidades.
Postcondiciones	El usuario ha logrado editar la entidad con éxito y ya le aparecen los nuevos datos de la entidad o recibe un error por parte del sistema.
Escenario de éxito	1. El usuario pulsa en el botón en forma de lápiz dedicado editar una fila determinada. 2. El sistema carga una ventana con todos los datos actuales de la entidad y da la posibilidad al usuario de editarlos. 3. El usuario edita alguno de los datos y pulsa en "Aceptar". 4. El sistema comprueba que los datos de la entidad son correctos y muestra un mensaje de edición correcta. 5. El sistema carga de nuevo la lista con los datos de la entidad actualizados.

Escenarios alternativos	No se introducen los datos en los campos requeridos: 1. El usuario no introduce los datos en los campos requeridos y pulsa "Aceptar". 2. El sistema notifica los campos que son obligatorios introduciendo marcándolos con una sombra roja y mantiene la ventana de creación hasta que se cierre, se pulse "Cancelar" o se decida introducir datos.
--------------------------------	--

Tabla 3.3.4: CU-04 Gestionar Entidad (Editar)

Identificador	CU-05
Título	Eliminar entidad
Actores	Usuario y sistema
Requisitos	RF-03, RF-04, RF-05, RF-06, RF-07, RF-20
Precondiciones	El usuario ha iniciado sesión y se encuentra en alguna de las páginas dedicadas a gestionar las entidades.
Postcondiciones	El usuario ha logrado eliminar la entidad y ya le aparecen los nuevos datos sin la entidad borrada.
Escenario de éxito	1. El usuario selecciona la entidad a borrar y pulsa en el botón "Eliminar". 2. El sistema carga una ventana de confirmación. 3. El usuario pulsa el botón "Si".

	4. El sistema elimina la entidad y carga la lista de entidades sin la entidad que se acaba de borrar.
Escenario alternativo	Rechazo de la confirmación: 1. El usuario pulsa el botón "No" en la confirmación de eliminar la entidad. 2. El sistema cierra la ventana de confirmación.

Tabla 3.3.5: CU-05 Gestionar Entidad (Eliminar)

Identificador	CU-06
Título	Exportar datos
Actores	Usuario y sistema
Requisitos	RF-03, RF-04, RF-05, RF-06, RF-07, RF-08, RF-09, RF-10, RF-11, RF-12, RF-13, RF-14, RF-18, RF-19
Precondiciones	El usuario ha iniciado sesión y se encuentra en algunas de las páginas dedicadas visualizar datos o gestionar datos.
Postcondiciones	El usuario obtiene el fichero en formato EXCEL de los datos que deseaba exportar
Escenario de éxito	1. El usuario pulsa en el botón "Exportar" de la página. 2. El sistema recibe la señal de exportación y exporta la lista de datos en un archivo EXCEL.

Escenario alternativo	
------------------------------	--

Tabla 3.3.6: CU-06 Exportar datos

Identificador	CU-07
Título	Importar datos
Actores	Usuario y sistema
Requisitos	RF-03, RF-04, RF-05, RF-06, RF-07, RF-16
Precondiciones	El usuario ha iniciado sesión y se encuentra en alguna página del importar o en la sección de importar datos necesarios para la asignación.
Postcondiciones	El usuario consigue importar los datos en el sistema y ya le aparecen los datos en la lista de datos de la aplicación o recibe un error por parte del sistema.
Escenario de éxito	1. El usuario pulsa el botón "Escoger Fichero". 2. El sistema abre una ventana de selección con las distintas carpetas del dispositivo del usuario para que pueda escoger el archivo deseado. 3. El usuario selecciona el archivo y pulsa abrir en la ventana de selección. 4. El sistema carga el archivo en la aplicación y muestra el nombre. 5. El usuario pulsa en el botón "Subir". 6. El sistema importa el archivo en la aplicación y actualiza la lista con los datos nuevos.

Escenario alternativo	Botón "Subir" sin seleccionar el archivo: 1. El usuario pulsa el botón "Subir" sin seleccionar un archivo. 2. El sistema muestra un mensaje de error diciendo "Archivo no seleccionado". Datos repetidos: 1. El usuario importa un archivo el cual contiene datos repetidos. 2. El sistema detecta datos que ya existen en la lista de las entidades y muestra un mensaje de error diciendo "Datos repetidos".
------------------------------	--

Tabla 3.3.7: CU-07 Importar datos

Identificador	CU-08
Título	Visualizar datos
Actores	Usuario y sistema
Requisitos	RF-03, RF-04, RF-05, RF-06, RF-07, RF-08, RF-09, RF-10, RF-11, RF-12, RF-13, RF-14, RF-18, RF-19, RF-20
Precondiciones	El usuario ha iniciado sesión.
Postcondiciones	El usuario ha logrado visualizar los datos que deseaba.
Escenario de éxito	1. El usuario pulsa el botón de una de las páginas dedicadas a visualizar datos o a gestionar datos.

	2. El sistema muestra la lista de los datos de la entidad que el usuario ha pulsado en el paso anterior.
Escenario alternativo	

Tabla 3.3.8: CU-08 Visualizar datos

Identificador	CU-09
Título	Ejecutar procesos de la asignación
Actores	Usuario y sistema
Requisitos	RF-15
Precondiciones	El usuario ha iniciado sesión y se encuentra en la página dedicada a ejecutar procesos.
Postcondiciones	El usuario ha logrado ejecutar alguno de los procesos de la asignación.
Escenario de éxito	1. El usuario pulsa algún botón de "Ejecutar". 2. El sistema muestra una ventana de confirmación. 3. El usuario pulsa en el botón "Si" de la ventana de confirmación. 4. El sistema recibe la señal de ejecutar y ejecuta el proceso seleccionado. 5. El sistema muestra un mensaje de información diciendo "Ejecutando proceso..." y actualiza los valores del usuario y fecha de la ejecución.

Escenario alternativo	Rechazo de la confirmación: 1. El usuario pulsa en el botón "No" de la ventana de confirmación. 2. El sistema cierra la ventana de confirmación y no ejecuta ningún proceso.
------------------------------	--

Tabla 3.3.9: CU-09 Ejecutar procesos de la asignación

Identificador	CU-10
Título	Recuperar contraseña
Actores	Usuario y sistema
Requisitos	RF-21
Precondiciones	El usuario se encuentra en la página de iniciar sesión y no recuerda su contraseña.
Postcondiciones	El usuario ha recibido su nueva contraseña para acceder al sistema o recibe un error del sistema.
Escenario de éxito	1. El usuario pulsa en el botón de la página de iniciar sesión "¿Has olvidado tu contraseña?". 2. El sistema carga una ventana donde se introducirá el correo que estaba utilizando el usuario en la aplicación. 3. El usuario introduce su correo. 4. El sistema comprueba que el correo está registrado y manda un correo con una contraseña aleatoria.

	5. El usuario recibe la nueva contraseña en su correo.
Escenario alternativo	<u>Se introduce un correo no registrado</u> 1. El usuario introduce un correo que no está en la aplicación. 2. El sistema detecta que no está registrado ese correo en la base de datos y muestra un mensaje de error diciendo "El correo introducido no está registrado" y cierra la ventana dedicada a introducir el correo.

Tabla 3.3.10: CU-10 Recuperar contraseña

Identificador	CU-11
Título	Cambiar contraseña
Actores	Usuario y sistema
Requisitos	RF-22
Precondiciones	El usuario ha iniciado sesión y se encuentra en la página principal del sistema.
Postcondiciones	El usuario ha conseguido cambiar su contraseña o ha recibido un error.
Escenario de éxito	1. El usuario pulsa el botón donde aparece el nombre de su usuario en la página principal. 2. El sistema abre una ventana para introducir su contraseña actual y su nueva contraseña.

	3. El usuario escribe su contraseña actual y su contraseña nueva. 4. El sistema detecta que su contraseña actual coincide y su nueva contraseña es válida, le cambia la contraseña al usuario y muestra un mensaje de información diciendo "Contraseña cambiada con éxito".
Escenario alternativo	No introduce su contraseña actual correctamente: 1. El usuario introduce una contraseña que no coincide con su contraseña actual. 2. El sistema muestra un mensaje de error y no le permite al usuario cambiar su contraseña. Las nuevas contraseñas no coinciden: 1. El usuario introduce contraseñas nuevas que no coinciden. 2. El sistema detecta que las nuevas contraseñas no coinciden, borra los campos de Nueva Contraseña y Confirme su contraseña y los sombrea en rojo avisando del error.

Tabla 3.3.11: CU-11 Cambiar contraseña

4

Diseño de la aplicación

En esta sección de la memoria se va a explicar el concepto del diseño de la aplicación completa. En el diseño se va a comentar la estructura global del proyecto, entrando en detalle en la arquitectura que se ha utilizado para el desarrollo y como se comunican cada uno de los componentes de la arquitectura explicando tanto el proyecto de desarrollo, como el proyecto de la base de datos, los cuales se pueden considerar proyectos distintos, pero que requieren de su unión, para aprovechar al máximo cada uno de ellos.

4.1. Arquitectura de la aplicación

Como se mencionó en el capítulo de las tecnologías a utilizar para el proyecto se ha elegido la plataforma Java EE [15] para el desarrollo, la cual está basada en un modelo de aplicación dividido en tres capas, debido a que están distribuidos los componentes en tres diferentes localizaciones: la máquina del cliente, el servidor de aplicaciones Java EE y la base de datos.

Como se ve en la Figura 4.1.1 se hablan de tres capas, pero dentro de esas capas nos encontramos con otras cuatro capas, debido a que en el servidor de aplicaciones se encuentran tanto la capa de negocio, como la capa web. Es conveniente explicar en qué consiste cada una de esas cuatro capas y las tecnologías a utilizar en cada una de ellas:

- **Cliente:** En nuestro caso se trata del navegador al ser una aplicación web, pero puede ser un teléfono o una aplicación de escritorio. Es donde se presenta la información al usuario que va a utilizar la aplicación.

- **Capa web:** Es la parte que se comunica con el cliente y la capa de negocio. Su función principal es obtener los datos y mostrarlos al cliente en un formato adecuado. En esta capa de la aplicación se ha utilizado:
 - **Java Server Faces(JSF):** Tecnología dedicada al desarrollo de la interfaz de la aplicación web, a través de la inclusión de componentes de interfaz de usuario, como botones, paneles, tablas, etc.

 - **Backing Beans:** Son componentes asociados a los elementos de la interfaz de usuario. Los Backing Beans poseen métodos, que desarrollan funciones para los componentes de la interfaz de usuario. Algunas de estas funciones pueden ser validar la información de un componente, manejar eventos disparados por un componente en concreto o procesar la navegación hacia otra página.

- **Capa de negocio:** Es la capa en la que se implementa la lógica de la aplicación y en la cual se realiza el núcleo del procesamiento de datos. En esta capa se han utilizado:
 - **Enterprise Java Beans (EJB):** Se trata de componentes que implementan la lógica del negocio y los cuales deben ejecutarse en un contenedor de EJBs. Este contenedor de EJBs proporciona servicios de transacciones, concurrencia y escalabilidad. Los EJBs de nuestra aplicación son de tipo Session Beans y serán inyectados en los Backing Beans de los cuales se ha hablado en la capa de presentación.

 - **Java Persistence API:** Es una especificación a partir de la cual se puede recuperar información desde un sistema de almacenamiento y hacer que esta persista. Esta persistencia es de vital importancia en nuestra aplicación debido a la necesidad de acceder a la base de datos.

- **Sistema de información:** Esta sección está formada por los servicios que proporcionan los datos que son utilizados por la capa de negocio. En nuestro caso es la base de datos de Oracle.

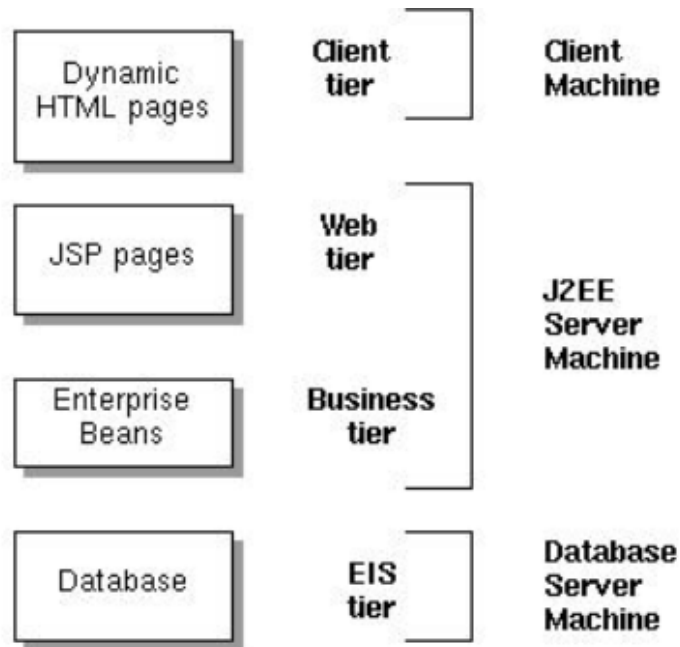


Figura 4.1.1: Arquitectura Java EE

Con el objetivo de realizar el proyecto de la forma más eficiente y organizada posible se ha dividido en dos subproyectos: uno es el proyecto maven donde se engloba la capa de negocio y la capa web y, por otro lado, el proyecto de la base de datos.

4.2. Proyecto Maven

Este proyecto se ha fragmentado a su vez en módulos donde en cada módulo se desarrolla cada una de las capas mencionadas anteriormente. Estos módulos se han creado gracias a la utilización de Maven y hacen un proyecto menos acoplado y más escalable. Los módulos de los que se hablan son los módulos EAR, EJB y WAR:

- **EAR:** Es el módulo principal. Está formado por el módulo WAR y el módulo EJB, que son empaquetados en un archivo comprimido .ear. Este archivo .ear después será subido al servidor de aplicaciones Wildfly.
- **EJB:** Es el módulo en el que se desarrolla la lógica de la aplicación, es decir la capa de negocio.

- **WAR:** Es el módulo dedicado al desarrollo de la interfaz de la aplicación, en otras palabras la capa web.

La estructura se puede apreciar mejor en la siguiente imagen, donde se ve claramente que el ejb y el war forman en conjunto el ear:



Figura 4.2.1: Estructura en módulos

Esta estructura de módulos se consigue en maven configurando adecuadamente el fichero pom.xml del proyecto padre, en el cual es necesario utilizar la etiqueta `<module>`:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>es.uma.informatica.sii</groupId>
  <artifactId>Secretaria</artifactId>
  <version>0.0.1-SNAPSHOT</version>

  <packaging>pom</packaging>
  <properties>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>

  <modules>
    <module>Secretaria-EJB</module>
    <module>Secretaria-WAR</module>
    <module>Secretaria-EAR</module>
  </modules>
</project>
```

Figura 4.2.2: pom.xml proyecto padre

La estructura definida en el *pom.xml* se ve en el IDE de la siguiente forma:

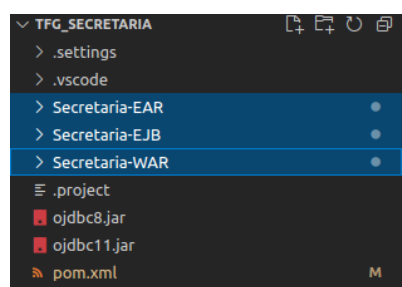


Figura 4.2.3: Estructura Proyecto Maven

Después de realizar en el pom.xml del proyecto padre los ajustes de los módulos, se procedería a ajustar cada uno de los submódulos.

En primer lugar se ajustaría el submódulo "Secretaria-EAR" esencial para poder desplegar la aplicación en el servidor de aplicaciones. Se va a utilizar el plugin *maven-ear-plugin*, el cual se encarga de generar el archivo Java EE (EAR). Además será necesario incluir como dependencias el resto de submódulos, es decir el Secretaria-EJB y el Secretaria-WAR, por lo que quedaría de la forma:

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>es.uma.informatica.sii</groupId>
    <artifactId>Secretaria</artifactId>
    <version>0.0.1-SNAPSHOT</version>
  </parent>
  <artifactId>Secretaria-EAR</artifactId>
  <packaging>ear</packaging>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-ear-plugin</artifactId>
        <configuration>
          <defaultLibBundleDir>lib</defaultLibBundleDir>
          <modules>
            <ejbModule>
              <groupId>es.uma.informatica.sii</groupId>
              <artifactId>Secretaria-EJB</artifactId>
            </ejbModule>
            <webModule>
              <groupId>es.uma.informatica.sii</groupId>
              <artifactId>Secretaria-WAR</artifactId>
            </webModule>
          </modules>
        </configuration>
      </plugin>
    </plugins>
  </build>
  <dependencies>
    <dependency>
      <groupId>es.uma.informatica.sii</groupId>
      <artifactId>Secretaria-EJB</artifactId>
      <version>0.0.1-SNAPSHOT</version>
      <type>ejb</type>
    </dependency>
    <dependency>
      <groupId>es.uma.informatica.sii</groupId>
      <artifactId>Secretaria-WAR</artifactId>
      <version>0.0.1-SNAPSHOT</version>
      <type>war</type>
    </dependency>
  </dependencies>
</project>
```

Figura 4.2.4: pom.xml Secretaria-EAR

El siguiente paso es continuar con el ajuste del submódulo "Secretaria-EJB", el cual va a contener la lógica de la aplicación. Este debe tener un fichero *persistence.xml* dentro de la carpeta "resources/META-INF", este fichero nos permite configurar la unidad de persistencia JPA. Esto se utiliza para manejar la persistencia de clases, el mapeo objeto/relacional y para utilizar *EntityManager* que nos va a permitir operar con el contexto de persistencia. El siguiente paso es conectar esa base de datos con la capa de presentación y para ello se usan los EJBs, los cuales realizarán las transacciones con la base de datos. Como se ve, hay dos partes en este submódulo EJB. Para organizarlo mejor se ha dividido en dos carpetas: *ejb*, en la cual se realizarán esas transacciones con la base de datos y *entidades*, donde se incluyen esas clases de mapeo con la base de datos:

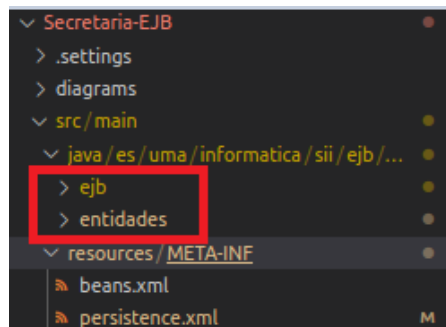


Figura 4.2.5: Organización módulo EJB

Para acabar con el ajuste de la estructura del Proyecto Maven se sigue con el submódulo "Secretaria-WAR" dedicado a la capa de presentación. En este módulo se utilizan los *backing beans*, que definen la funcionalidad de los componentes de la vista y las páginas *jsf*, que es donde se incluyen los componentes de la vista. Por tanto, se tienen dos partes de nuevo y para entenderlo mejor, se ha dividido en las carpetas: *backingbeans* y *webapp*.

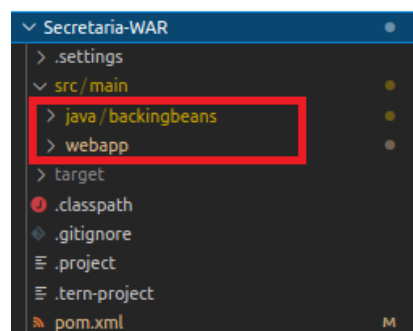


Figura 4.2.6: Organización módulo WAR

En este módulo, es esencial el fichero *web.xml*, el cual es un fichero descriptor del despliegue y contiene información y configuración de la aplicación web.

Con este último, se acabarían los distintos submódulos del proyecto Maven. En el capítulo 5 de desarrollo, se entrará más en detalle en cada una de las partes de los submódulos, explicando cómo se han ido desarrollando cada una de ellos.

4.3. Proyecto Base de Datos

El diseño de la base de datos de este proyecto está formado por una gran cantidad de tablas, vistas, procedimientos y funciones, que hacen más sencilla la implementación de

la parte de la aplicación web. Todos los componentes que forman esta base de datos en su mayoría se habían desarrollado con anterioridad al desarrollo de este proyecto y, por tanto, hay tablas que no se utilizan en la aplicación, ya que no son de importancia para la asignación de los alumnos a grupos.

La base de datos es objeto-relacional, por lo tanto almacena y proporciona acceso a datos que están relacionados entre sí. En esencia, este tipo de base de datos es un conjunto de tablas formadas por filas y columnas, y cada fila tiene un ID único o un conjunto de IDs llamados en un lenguaje más coloquial, claves primarias. Esto lo que permite, es evitar la duplicidad de datos y garantiza la integridad.

Por otra parte, gracias a la existencia de la clave primaria y la clave ajena se puede acceder de forma sencilla a la información entre tablas relacionadas, donde en una relación entre una tabla padre y una tabla hija, la tabla hija contendrá la clave ajena con el mismo valor que la clave primaria de la tabla padre y así se podría establecer una relación de una forma muy sencilla.

Teniendo en cuenta el tipo de base de datos hay que recalcar que los elementos utilizados de la base de datos para el desarrollo del proyecto han sido:

- **Tablas:** La tabla es la unidad básica de almacenamiento de la base de datos, tienen un nombre y un conjunto de columnas, estas últimas determinan los atributos de la tabla. Los datos se almacenan en la tabla en forma de filas y estas tablas pueden estar relacionadas con otras.
- **Vistas:** La vista es un tabla virtual que en realidad no existe físicamente. Están creadas mediante la unión de una o varias tablas y se utilizan para mostrar información que puede resultar de interés.
- **Procedimientos:** Los procedimientos son subprogramas desarrollados en PL/SQL que se pueden crear y guardar en la base de datos. Estos pueden ser llamadas

Ahora, se va a proceder a explicar cada uno de los elementos anteriores de forma más detallada.

[illegible]

Para entender mejor este diagrama es importante realizar una pequeña descripción de cada una de las tablas y para aclarar aún más, se incluirán algunos ejemplos:

- 46

- Tabla **EXPEDIENTE**: Guarda toda la información de los expedientes de los alumnos.
- Tabla **ALUMNO**: Almacena la información de los alumnos matriculados.
- Tabla **GRUPO**: Contiene la información de los diferentes grupos (Ej.: II1C - Ingeniería Informática 1º C).
- Tabla **GRUPOS_POR_ASIGNATURA**: Se utiliza como tabla para la relación Many-to-many entre la tabla Grupo y Asignatura. Representa los posibles grupos que puede tener una asignatura(Ej.: Si estas matriculado en la asignatura Cálculo lo normal es que se cojan grupos de primer curso como II1A,II1B,...).
- Tabla **TITULACIÓN**: Guarda toda la información de las diferentes titulaciones del centro ETSI Informática de la UMA (Ej.: Grado en Ingeniería informática, Grado en Ingeniería del Software).

Anteriormente se han mencionado todas las tablas con relaciones, pero también hay tablas en las cuales no era interesante incluir ningún tipo de relación y también se han utilizado:

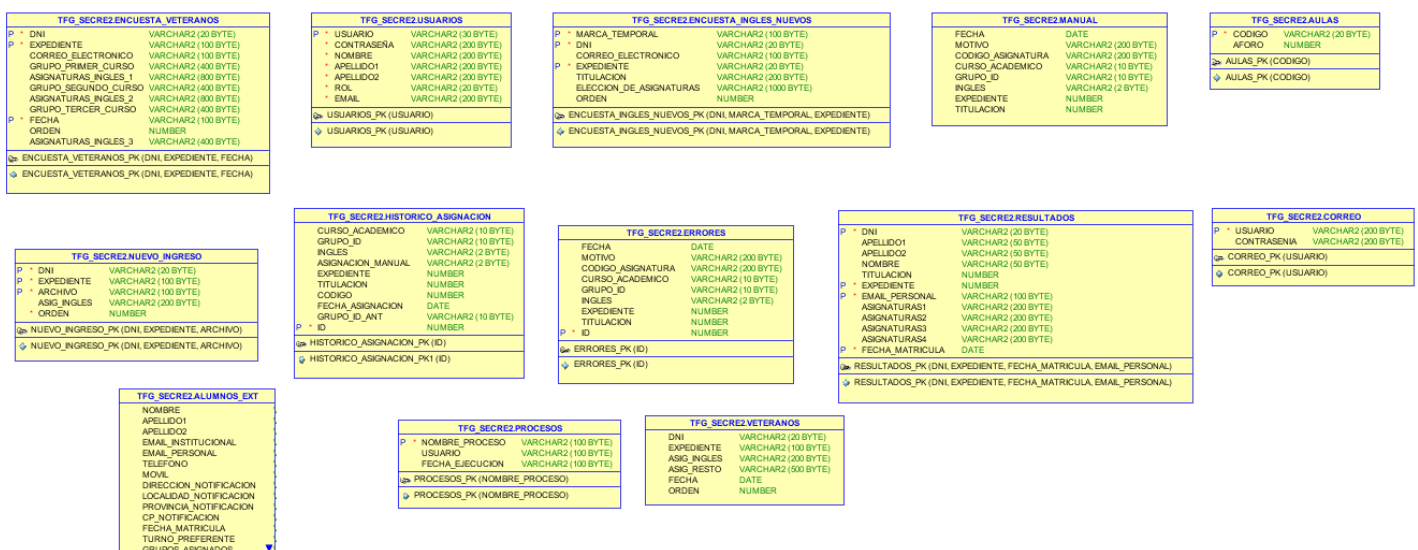


Figura 4.3.2: Segundo diagrama entidad relación

Vamos a realizar una pequeña descripción de cada una para que quede aún más claro:

- Tabla **ENCUESTA_VETERANOS**: Utilizada para almacenar la información de las asignaturas escogidas por los alumnos veteranos.
- Tabla **USUARIOS**: Tabla principal de la aplicación donde se almacenará la información de las personas que utilicen la aplicación y de donde se obtendrán los datos para poder acceder al sistema.
- Tabla **ENCUESTA_INGLES_NUEVOS**: Almacena toda la información de las asignaturas en inglés escogidas por los alumnos de nuevo ingreso.
- Tabla **MANUAL**: Tabla utilizada para guardar la información de asignaciones manuales en caso de no poder realizar la asignación mediante los procedimientos.
- Tabla **AULAS**: Contiene la información de las aulas del centro, según el módulo, la planta y el aula (Ej.: 303, 307).
- Tabla **NUEVO_INGRESO**: Contiene la información de los alumnos de nuevo ingreso y su selección de las asignaturas en inglés.
- Tabla **HISTORICO_ASIGNACIÓN**: Se utiliza para guardar un registro de la asignación realizada.
- Tabla **ERRORES**: Utilizada para almacenar posibles errores de asignación que puedan surgir durante la ejecución de procesos.
- Tabla **CORREO**: Almacena la información del correo que la aplicación utiliza para mandar un email en caso de se le olvide la contraseña a un usuario.
- Tabla **ALUMNOS_EXT**: Esta tabla almacena la información de los alumnos matriculados y es donde comienza toda la ejecución de los procesos, ya que de

esta tabla se saca toda la información para rellenar la mayoría de tablas de la base de datos.

- Tabla **PROCESOS**: Contiene la información de los procesos ejecutados, para llevar un control de la fecha y el usuario que ejecuta cada uno de los procesos.
- Tabla **VETERANOS**: Esta tabla contiene la información de todos los alumnos veteranos y su selección de asignaturas.
- Tabla **RESULTADOS**: En esta tabla se recogen todos los resultados de las asignaciones por expediente. Esta será la tabla con la información más importante y de la cual se obtendrán los resultados que después utilizará el personal de Secretaría.

Como se ha dicho, también se han utilizado vistas en el proyecto, ya que ayudan a mostrar información de diferentes tablas en una y por tanto visualizar diferentes puntos de vista interesantes de la información:

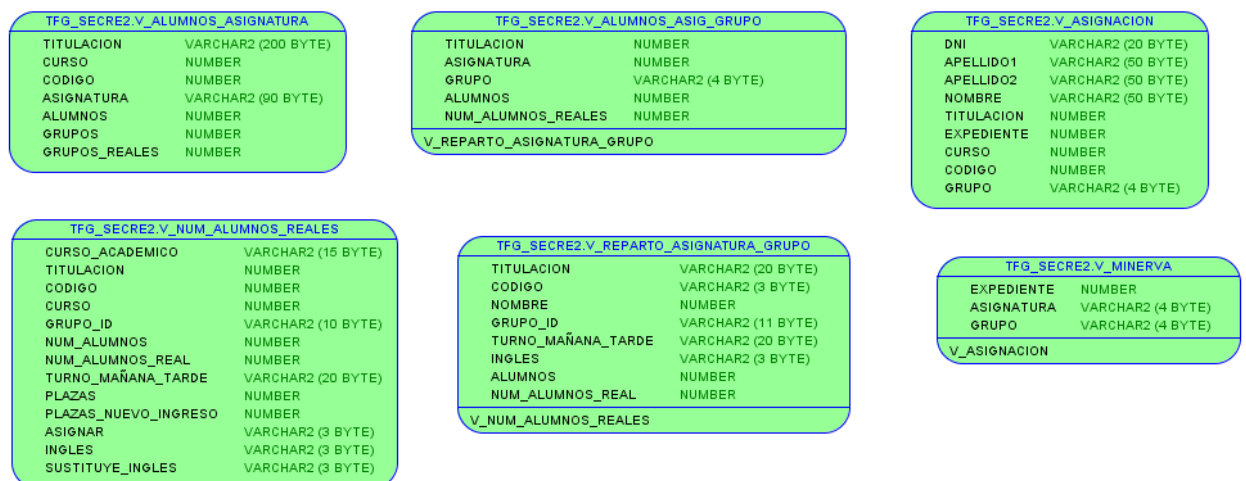


Figura 4.3.3: Diagrama vistas base de datos

Para poder entender mejor para que se utiliza cada una de ellas:

- Vista **V_MINERVA**: Utilizada para ver los resultados de la asignación por expediente, asignatura y grupo. Esta vista contiene los resultados de la

asignación, pero en un formato distinto a la tabla RESULTADOS, comentada con anterioridad.

- Vista **V_ASIGNACIÓN**: Se utiliza para mostrar la misma información que la anterior, pero además se incluye la titulación y datos del alumno.
- Vista **V_REPARTO_ASIGNATURA_GRUPO**: Se utiliza para observar los resultados de la asignación por asignatura y grupo, pudiendo observar los alumnos que quedan por asignar de una determinada asignatura.
- Vista **V_ALUMNOS_ASIGNATURA**: Vista utilizada para ver el número de alumnos y de grupos de una asignatura ordenado por titulación y curso.
- Vista **V_ALUMNOS_ASIG_GRUPO**: Vista dedicada a mostrar la información del número de alumnos por asignatura y por grupo.
- Vista **V_NUM_ALUMNOS_REALES**: Utilizada para poder visualizar toda información de cada uno de los grupos, incluyendo el número de alumnos asignados en cada grupo.

Para acabar con el diseño del proyecto de la base de datos es conveniente hablar de los procedimientos que se han utilizado para la asignación, los cuales son un punto clave de todo el proyecto. Hay que destacar *que hay muchos más procesos que realizan funciones relacionadas con asignación, pero los utilizados en la aplicación directamente son los siguientes:*

- **BORRAR_ASIGNACIÓN (Modo)**: Utilizado para borrar todos los datos de las tablas y poder realizar una asignación "limpia" de nuevo. Recibe el parámetro Modo, el cual puede ser 'TOTAL', en la que se borrarán todos los datos, incluidos los datos de los alumnos matriculados (tabla ALUMNOS_EXT), que es donde comienza toda la asignación. El modo también puede ser 'PARCIAL' en la que se borra todo, menos los datos de los alumnos matriculados y el historial de asignación. Este procedimiento resulta muy útil para poder reutilizar la

aplicación cada curso, ya que permite borrar todos los datos y poder actualizarlos después al curso nuevo.

- **TRAS_IMPORTAR:** Procedimiento utilizado para poder rellenar las tablas ALUMNOS, EXPEDIENTE, MATRICULA, ASIGNATURAS_MATRICULA, además asigna grupos que ya vengan asignados todo eso a partir de los datos de la tabla de ALUMNOS_EXT.
- **PR_ASIGNA_NUEVOS:** Se encarga de asignar los alumnos de nuevo ingreso, primeramente los que solicitan turno de tarde y en segundo lugar el resto de alumnos de nuevo ingreso, teniendo en cuenta un criterio como la fecha de matriculación u otro criterio establecido y esta asignación se guarda en la tabla HISTORICO_ASIGNACIÓN.
- **RELLENA_INGLES_NUEVO:** Actualiza la tabla NUEVO_INGRESO con las preferencias en inglés de los alumnos de nuevo ingreso obtenidas de la tabla ENCUESTA_INGLES_NUEVOS.
- **PR_ASIGNA_INGLES_NUEVO:** Este realiza la asignación de los grupos de inglés según los datos de la tabla de NUEVO_INGRESO, la cual queda actualizada en el paso anterior con las preferencias de inglés de los alumnos.
- **RELLENA_ENCUESTA_VETERANOS:** Actualiza la tabla VETERANOS según las preferencias de las asignaturas en inglés y el resto de las asignaturas, recorriendo la tabla de ENCUESTA_VETERANOS.
- **PR_ASIGNA_INGLES_VETERANOS:** Asigna los grupos de inglés a alumnos veteranos que han pedido asignaturas en inglés teniendo en cuenta sus preferencias sacadas de la tabla VETERANOS.
- **PR_ASIGNA_ENCUESTA_VETERANOS:** Asigna los grupos del resto de asignaturas que no son en inglés a los alumnos veteranos.

- **PR_ASIGNA_RESTO_VET:** Dedicado a asignar el resto de alumnos que no han rellenado la matrícula y no tienen un grupo asignado aún en alguna asignatura. Es importante por si en la asignación han quedado alumnos con alguna asignatura que no le han asignado a un grupo.
- **GENERAR_RESULTADO:** Procedimiento dedicado a rellenar la tabla de RESULTADOS tras cada asignación, con el objetivo de poder visualizar los posibles cambios en la asignación.

Los procedimientos explicados con anterioridad deben seguir un orden porque la asignación de grupos a alumnos no es un proceso que se pueda realizar inconscientemente. El procedimiento de *borrar_asignación* es muy importante al ser el primero y este permite realizar una asignación limpia desde cero al borrar los datos. Después se debe comenzar la asignación con los alumnos de nuevo ingreso, ya que los datos de estos, al realizar la selectividad en junio, se tienen antes que la de los veteranos; y por último se sigue con la asignación de los veteranos.

5

Desarrollo

En este capítulo se realizará una descripción detallada del proceso de desarrollo del proyecto con el objetivo principal de satisfacer cada uno de los requisitos especificados en el capítulo 3. Al haber utilizado la metodología iterativa e incremental, el desarrollo se ha dividido en diversas iteraciones, donde cada una se centra en una serie de requisitos y al final, cuando se han desarrollado todas las iteraciones, se consigue la aplicación completa. En cada una de las iteraciones, además de explicar el proceso de desarrollo de los requisitos elegidos, se especificarán algunas dificultades encontradas.

5.1. Primera iteración

En esta primera iteración, el objetivo era realizar un primer acercamiento a las tecnologías a utilizar en el desarrollo, comenzando a desarrollar los primeros requisitos de la aplicación y a estructurar la vista de la aplicación. Para comenzar con el desarrollo se pensó en implementar los requisitos relacionados con los usuarios, aparte de estructurar la vista de la aplicación:

- Iniciar Sesión, Cerrar Sesión, Recuperar la contraseña, Gestionar Usuarios, Cambiar contraseña.

Para comenzar con el desarrollo de las funcionalidades hay que empezar por la parte del módulo Secretaria-EJB, y después, la parte de la capa de Secretaria-WAR. Dentro del módulo Secretaria-EJB, ya se dijo que había dos partes más:

1) En la parte de JPA hay que mapear la clase Usuarios a la tabla Usuarios de la base de datos, para ello hay que seguir los siguientes pasos:

- Anotación `@Entity` para identificar la clase, además se debe tener en cuenta que todas las clases entidad de este tipo deben implementar la interfaz `Serializable`, que permite a Java serializar la clase una secuencia de bytes (que puede ser almacenada en disco o transmitida por la red, entre otras cosas).
- Más tarde, se sigue con los atributos de la entidad. Para indicar la clave primaria de la entidad se utiliza `@Id` y en cada uno de los atributos, para asegurarnos de que el nombre de la clase y la tabla coincidan se utiliza `@Column(name="...")` en esta etiqueta `@Column`. En esta etiqueta también se pueden utilizar propiedades como el `nullable=false`, para indicar que ese atributo no puede ser null y el `unique=true` para indicar que ese atributo es único.

```
Usuarios.java M X
Secretario-EJB > src > main > java > es > uma > informatica > sii > ejb > practica > entidades > Usuarios
1 package es.uma.informatica.sii.ejb.practica.entidades;
2
3 import java.io.Serializable;
4
5 import javax.persistence.Column;
6 import javax.persistence.Entity;
7 import javax.persistence.Id;
8
9 @Entity
10 public class Usuarios implements Serializable {
11
12     @Id
13     @Column(name = "USUARIO", nullable = false)
14     private String usuario;
15
16     @Column(name = "CONTRASEÑA", nullable = false)
17     private String contrasena;
18
19     @Column(name = "NOMBRE", nullable = false)
20     private String nombre;
21
22     @Column(name = "APELLIDO1", nullable = false)
23     private String apellido1;
24
25     @Column(name = "APELLIDO2", nullable = false)
26     private String apellido2;
27
28     @Column(name = "ROL", nullable = false)
29     private String rol;
30
31     @Column(name = "EMAIL", nullable = false, unique = true)
32     private String email;
33
34     public String getRol() {
35         return this.rol;
36     }
37
38     public void setRol(String rol) {
39         this.rol = rol;
40     }
41
42     public String getUsuario() {
43         return this.usuario;
44     }
45
46     public void setUsuario(String usuario) {
47         this.usuario = usuario;
48     }
49 }
```

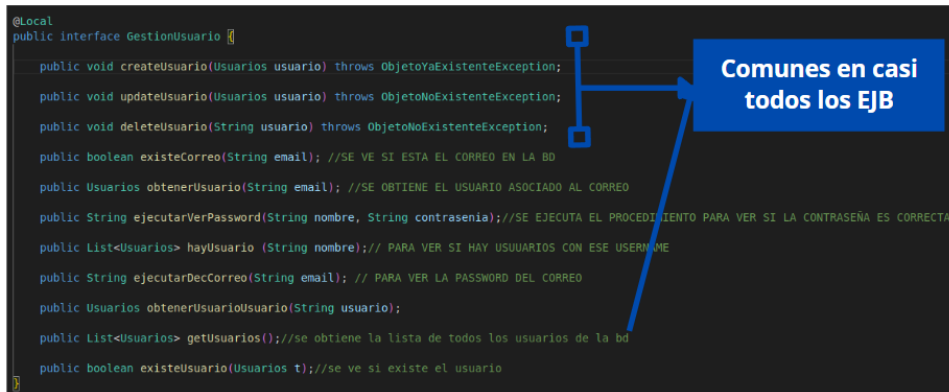
Figura 5.1.1: Clase Usuarios.java

- El último paso sería incluir los *getters* y *setters* de cada uno de los atributos, ya que se va a querer acceder a los atributos de la clase Usuarios.

Todo esto es lo necesario para realizar el mapeo adecuadamente, más tarde se explicarán posibles modificaciones que pueden surgir en una entidad (por ejemplo, cuando se tiene una id compuesta). Para que quede más claro lo anterior se puede observar la Figura 5.1.1.

2) La siguiente parte a implementar son los Enterprise Java Beans con lo que se harán las transacciones. El tipo de EJB utilizado en toda la aplicación ha sido *Session beans*, los cuales se inyectan en los backing beans e implementan la lógica. Este Session bean está formado por:

- Una **interfaz** en la que se declaran todos los métodos que se van a utilizar en el bean. En esta interfaz se utiliza la etiqueta *@Local*, ya que se está utilizando un servidor de aplicaciones, por lo que el cliente y el EJB están en la misma máquina virtual de Java.



```

@Local
public interface GestionUsuario {

    public void createUsuario(Usuarios usuario) throws ObjetoYaExistenteException;
    public void updateUsuario(Usuarios usuario) throws ObjetoNoExistenteException;
    public void deleteUsuario(String usuario) throws ObjetoNoExistenteException;
    public boolean existeCorreo(String email); //SE VE SI ESTA EL CORREO EN LA BD
    public Usuarios obtenerUsuario(String email); //SE OBTIENE EL USUARIO ASOCIADO AL CORREO
    public String ejecutarVerPassword(String nombre, String contrasenia); //SE EJECUTA EL PROCEDIMIENTO PARA VER SI LA CONTRASEÑA ES CORRECTA
    public List<Usuarios> hayUsuario (String nombre); // PARA VER SI HAY USUARIOS CON ESE USERNAME
    public String ejecutarDecCorreo(String email); // PARA VER LA PASSWORD DEL CORREO
    public Usuarios obtenerUsuarioUsuario(String usuario);
    public List<Usuarios> getUsuarios(); //se obtiene la lista de todos los usuarios de la bd
    public boolean existeUsuario(Usuarios t); //se ve si existe el usuario
}
  
```

Comunes en casi todos los EJB

Figura 5.1.2: Interfaz Session Bean GestionUsuario

En la etiqueta azul de la imagen anterior se pueden apreciar los métodos que casi siempre van a ser comunes en casi todos los EJBs, debido a que cuando se quiere gestionar una entidad siempre se realiza un crear, un actualizar, un eliminar o un obtener datos.

- Un **bean** con la etiqueta *@Stateless*. Esto implica que no guarda información entre llamadas, por lo tanto es sin estado y hace que sea más eficiente. Aquí es donde se implementan los métodos de la interfaz anterior. Una anotación importante es *@PersistenceContext* para inyectar el contexto de persistencia y poder utilizar *EntityManager* que controla el ciclo de vida de cada entidad del contexto de persistencia.



```

@Stateless
public class UsuarioEJB implements GestionUsuario {

    @PersistenceContext(name = "Secretaria")
    private EntityManager em;

    @Override
    public void createUsuario(Usuarios usuario) throws ObjetoYaExistenteException {
        Usuarios aux = em.find(entityClass: Usuarios.class, usuario.getUsuario());
        if (aux != null) {
            throw new ObjetoYaExistenteException(message: "Este usuario ya existe");
        }
        em.persist(usuario);
    }

    @Override
    public void updateUsuario(Usuarios usuario) throws ObjetoNoExistenteException {
        Usuarios aux = em.find(entityClass: Usuarios.class, usuario.getUsuario());
        if (aux == null) {
            throw new ObjetoNoExistenteException(message: "Usuario no encontrado");
        }
        // aux = codigo;
        em.merge(usuario);
    }

    @Override
    public void deleteUsuario(String usuario) throws ObjetoNoExistenteException {
        Usuarios aux = em.find(entityClass: Usuarios.class, usuario);
        if (aux == null) {
            throw new ObjetoNoExistenteException(message: "Usuario no encontrado");
        }
        em.remove(aux);
    }

    @Override
    public List<Usuarios> getUsuarios() {
        TypedQuery<Usuarios> query = em.createQuery(q1String: "Select t FROM Usuarios t", resultClass: Usuarios.class);
        return query.getResultList();
    }
}
  
```

Con EntityManager se puede realizar toda la gestión

Figura 5.1.3: UsuarioEJB

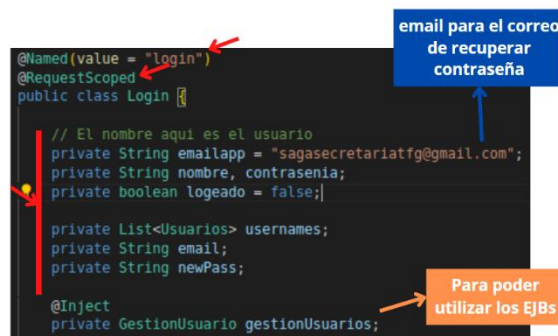
Estos cuatro métodos, como se ve en la Figura 5.1.4 hacen uso de EntityManager y con una simple instrucción realizan la transacción con la base de datos. En cada caso se utiliza una operación diferente:

- Para crear, se utiliza *persist* para persistir una nueva entidad en la base de datos.
- Para modificar, se utiliza *merge* para actualizar una entidad que ya está en la base de datos.
- Para eliminar, se utiliza *remove* la cual solo requiere la entidad.
- Para obtener los datos, una query con el uso de la operación *createQuery*.

Esto hace que la parte de gestionar una entidad e interactuar con la base de datos sea una tarea muy sencilla. Por tanto, con estos 4 métodos se habría completado la funcionalidad de Gestionar Usuarios.

Ahora se pasaría al módulo Secretaria-WAR, que se corresponde con la presentación y también incluye dos partes:

1) Por un lado los backing beans, que son la funcionalidad de la vista.



```
@Named(value = "login")
@RequestScoped
public class Login {

    // El nombre aquí es el usuario
    private String emailapp = "sagasecretariatfg@gmail.com";
    private String nombre, contrasenia;
    private boolean logeado = false;

    private List<Usuarios> usernames;
    private String email;
    private String newPass;

    @Inject
    private GestionUsuario gestionUsuarios;
}
```

Annotations and their purposes:

- `@Named(value = "login")`: Indicated by a red arrow and a blue box labeled "email para el correo de recuperar contraseña".
- `@RequestScoped`: Indicated by a red arrow.
- `@Inject`: Indicated by an orange arrow and an orange box labeled "Para poder utilizar los EJBs".

Figura 5.1.4: Backing bean Login.java

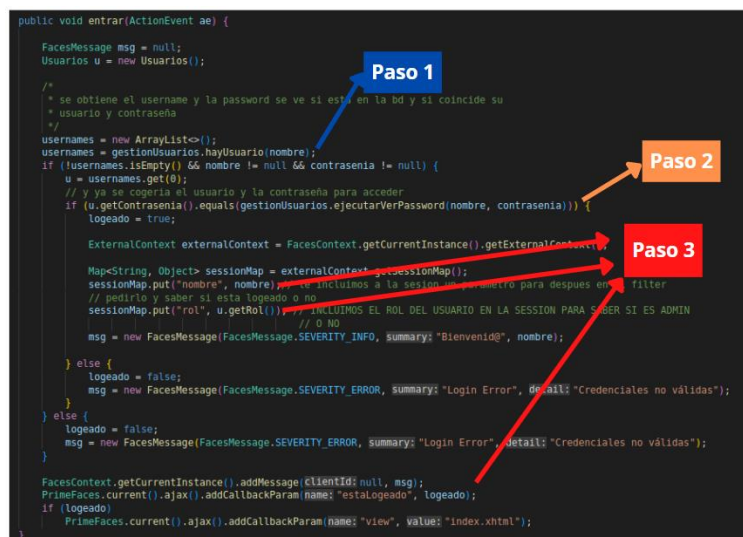
Aquí es necesario utilizar:

- Anotación *@Named* para poder usar ese backing bean en la vista, con un nombre concreto.
- Una anotación de ámbito, que indica el contexto en el que se crea del backing bean. En esta clase Login.java en concreto se usa *@RequestScoped* para que el bean exista durante una petición.

- Lo siguiente sería especificar los atributos que se van a utilizar en la vista. Como se van a incluir dos `inputText`, es decir dos cadenas (String), se incluyen los atributos nombre y contraseña y sus respectivos getters y setters.
- `@Inject` para inyectar EJBs.

El método del backing bean utilizado para iniciar sesión, va a utilizar sobre todo `EntityManager`:

1. Llamada al método del EJB *hayUsuario* para poder ver si existen usuarios con ese nombre.
2. Comprobar si la contraseña es correcta llamado al método del EJB *ejecutarVerPassword*, que llama a un proceso de la base de datos que cifra los credenciales y comprueba si coinciden con los de la base de datos, los cuales se almacenan cifrados.
3. Si coinciden incluye en nombre y el rol en la sesión, que después nos será útil y le pasa un valor true o false a una función en JavaScript que permite el acceso.



```

public void entrar(ActionEvent ae) {
    FacesMessage msg = null;
    Usuarios u = new Usuarios();

    /*
     * se obtiene el username y la password se ve si está en la bd y si coincide su
     * usuario y contraseña
     */
    usernames = new ArrayList<>();
    usernames = gestionUsuarios.hayUsuario(nombre);
    if (usernames.isEmpty() && nombre != null && contraseña != null) {
        u = usernames.get(0);
        // y ya se copia el usuario y la contraseña para acceder
        if (u.getContraseña().equals(gestionUsuarios.ejecutarVerPassword(nombre, contraseña))) {
            logeado = true;

            ExternalContext externalContext = FacesContext.getCurrentInstance().getExternalContext();
            Map<String, Object> sessionMap = externalContext.getSessionMap();
            sessionMap.put("nombre", nombre); // se incluimos a la sesión el nombre para después en el filter
            // pedirlo y saber si está logeado o no
            sessionMap.put("rol", u.getRol()); // INCLUIR EL ROL DEL USUARIO EN LA SESION PARA SABER SI ES ADMIN
            // O NO
            msg = new FacesMessage(FacesMessage.SEVERITY_INFO, summary: "Bienvenido", nombre);
        } else {
            logeado = false;
            msg = new FacesMessage(FacesMessage.SEVERITY_ERROR, summary: "Login Error", detail: "Credenciales no válidas");
        }
    } else {
        logeado = false;
        msg = new FacesMessage(FacesMessage.SEVERITY_ERROR, summary: "Login Error", detail: "Credenciales no válidas");
    }

    FacesContext.getCurrentInstance().addMessage(ELClientId, msg);
    PrimeFaces.current().ajax().addCallbackParam(name: "estaLogeado", logeado);
    if (logeado)
        PrimeFaces.current().ajax().addCallbackParam(name: "view", value: "index.xhtml");
}

```

El diagrama muestra tres pasos numerados con flechas que indican la ejecución del código:

- Paso 1:** Indica la línea `usernames = gestionUsuarios.hayUsuario(nombre);`.
- Paso 2:** Indica la línea `if (u.getContraseña().equals(gestionUsuarios.ejecutarVerPassword(nombre, contraseña)))`.
- Paso 3:** Indica la línea `sessionMap.put("rol", u.getRol());`.

Figura 5.1.5: método `entrar()`

2) El segundo paso en el módulo WAR es la página JSF donde se incluye cada componente de Primefaces. Algunos de los componentes utilizados en la vista `login.xhtml` son botones, paneles, `inputText` para introducir texto y poco más al ser una vista muy sencilla. Para llamar a métodos o atributos del backing bean se seguirá siempre la misma estructura `#{nombreBackinBean.atributoOMetodo}`. Esta se puede

observar en la Figura 5.1.6 donde en cada componente que utiliza algo del backing bean, se utiliza esta estructura determinada:

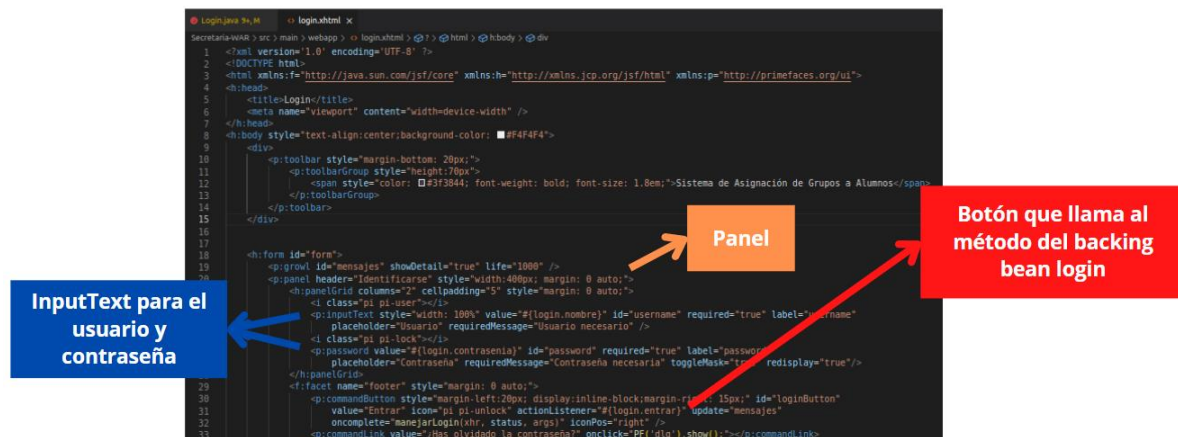


Figura 5.1.6: Página login.xhtml

Todo lo comentado anteriormente en conjunto formó la primera página de la aplicación:

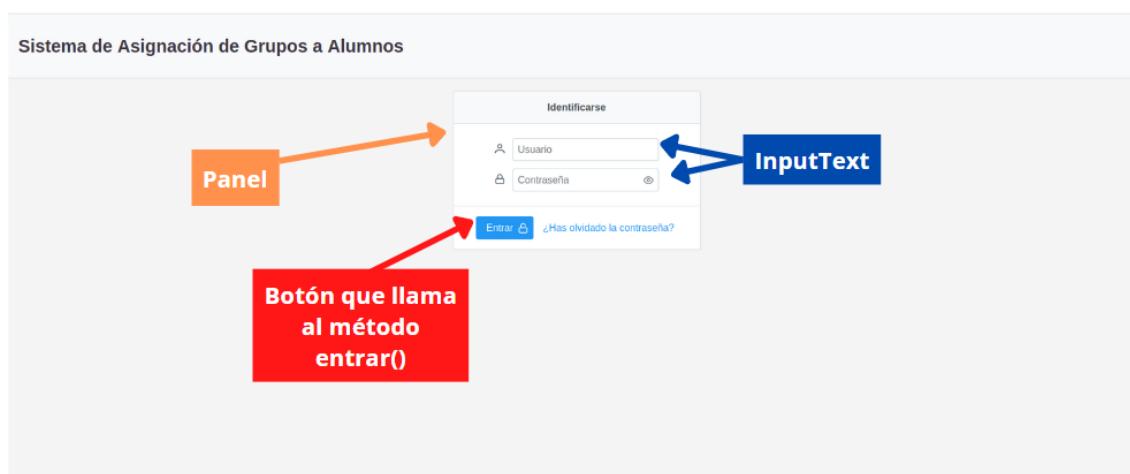


Figura 5.1.7: Vista de la página login.xhtml

Otra funcionalidad que comentar de esta vista es el *¿Has olvidado la contraseña?* que se va a encargar de cumplir el requisito *Recuperar Contraseña* y mandará un correo al usuario con su contraseña nueva. Para ello, en la parte de negocio se ha hecho uso de la *API JavaMail*, que facilita todo el proceso de recepción y envío de emails y de *Apache Commons Lang* para generar una nueva contraseña aleatoria.

```

public void enviarEmail() {
    if (gestionUsuarios.existeCorreo(email)) { // comprobamos que el correo esta antes en la bd si no no hacemos
        // nada
        Properties props = System.getProperties();
        props.put("mail.smtp.host", "smtp.gmail.com");

        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.port", "465");
        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.socketFactory.class", "javax.net.ssl.SSLSocketFactory");
        props.put("mail.smtp.socketFactory.port", "465");
        props.put("mail.smtp.socketFactory.fallback", "false");

        Session mailSession = Session.getDefaultInstance(props, new Authenticator(){});
        Message mailMessage = new MimeMessage(mailSession);

        try {
            mailMessage.setFrom(new InternetAddress(emailapp));
            mailMessage.setRecipient(Message.RecipientType.TO, new InternetAddress(email));

            newPass = RandomStringUtils.randomAlphanumeric(10); // se le manda una nueva contraseña aleatoria de 10
            // digitos

            mailMessage.setContent(
                "Hola se nueva contraseña es: " + newPass + "\nAcceda de nuevo al sistema con esa contraseña y cambiala.",
                "text/plain");
            mailMessage.setSubject("Nuevas credenciales SAGA");

            Usuarios correo = gestionUsuarios.obtenerUsuario(email); // VENOS EL EMAIL QUE HAN INTRODUCIDO

            Transport transport = mailSession.getTransport("smtp");
            transport.connect(host, "smtp.gmail.com", emailapp,
                gestionUsuarios.ejecutarDecCorreo(emailapp));
            transport.sendMessage(mailMessage, mailMessage.getAllRecipients());

            this.crearContraseña(correo); // SE LE CREA LA NUEVA CONTRASEÑA AL USUARIO
            FacesContext.getCurrentInstance().addMessage(null,
                new FacesMessage("Correo enviado a " + email + " con éxito"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

Figura 5.1.8: Método enviarEmail()

En la parte de la capa web hay un simple diálogo que se muestra al pulsar en "¿Has olvidado la contraseña?" y que contiene un `inputText` para introducir el correo. Este correo cabe destacar que debe cumplir una estructura determinada, para ello se ha hecho uso de una expresión regular y el componente `validateRegex` que se encarga de comprobarlo.

```

<h:form id="dlgForm">
    <p:growl id="mensajes1" showDetail="true" />
    <p:dialog header="¿Has olvidado la contraseña?" showEffect="fade" modal="true" widgetVar="dlg"
        responsive="true">
        <p:outputPanel id="manage-usuario-content" class="ui-fluid">
            <p:outputPanel>
                <div class="p-field">
                    <h:outputText value="Te enviaremos una nueva contraseña al correo que introduzcas" />
                    <br/>
                    <br/>
                    <p:inputText id="email" value="#{login.email}" required="true"
                        validatorMessage="Email invalido">
                        <f:validateRegex
                            pattern="[A-Za-z0-9-]+(\.[A-Za-z0-9-]+)*@[A-Za-z0-9-]+(\.[A-Za-z0-9-]+)*(\.[A-Za-z]{2,})$" />
                    </p:inputText>
                </div>
            </p:outputPanel>
            <p:outputPanel>
                <f:facet name="footer">
                    <p:commandButton value="Enviar" icon="pi pi-send" actionListener="#{login.enviarEmail}"
                        update="mensajes1">
                    </p:commandButton>
                    <p:commandButton value="Cancelar" icon="pi pi-times" onclick="PF('dlg').hide()"
                        class="ui-button-secondary">
                    </p:commandButton>
                </f:facet>
            </p:outputPanel>
        </p:dialog>
    </h:form>

```

Figura 5.1.9: Diálogo login.xhtml

Es conveniente explicar alguna de las propiedades utilizadas en los componentes de la vista, que se pueden apreciar en la Figura 5.1.6 y en la Figura 5.1.9, los cuales permiten aplicar un poco de interactividad y vistosidad a la página [16] :

- **Action:** Permite ejecutar métodos cuando se ejecuta el componente.

- **ActionListener:** Procesa un `actionListener` para manejar eventos cuando se ejecuta.
- **OnClick:** Realiza una llamada cuando el botón es pulsado.
- **Oncomplete:** Realiza una llamada cuando la petición Ajax es completada.
- **Icon:** Propiedad que permite incluir iconos en el componente, como por ejemplo el candado a la derecha del botón Entrar de la Figura 5.1.7.
- **Update:** Indica el componente a actualizar con Ajax.
- **Required:** Hace que la escritura en un `inputText` sea obligatoria.

Ajax es bastante recurrente en los componentes de Primefaces y como ya se dijo estos componentes utilizan internamente otras tecnologías y frameworks. Ajax es uno de ellos y permite actualizar y procesar ciertos componentes sin la interrupción de recargar la página web, es decir lo que hace es recargar y renovar la información asociada a un único componente.

Tras el desarrollo de la página inicial de Iniciar Sesión se pensó la estructura general que la vista debía seguir para ser al mismo tiempo minimalista, intuitiva y fácil de usar para cualquier persona. Para ello fue esencial el dividir toda la estructura de la aplicación en cuatro partes distintas:

- **Gestionar Entidades:** En esta sección de la vista se incluirían todas las páginas dedicadas a gestionar.
- **Visualizar Datos:** En esta parte se incluirían todas las páginas en las que se pudieran únicamente visualizar datos.
- **Ejecutar Procesos:** Esta es la última parte visible para cualquier usuario no administrador. En esta se añadiría todo lo relacionado con la ejecución de procesos.
- **Gestionar Usuarios:** Sección dedicada a realizar la gestión de los usuarios solo para los administradores.

Para esto se creó una página principal con un menú para llevar al usuario a cada una de las cuatro secciones anteriores y para que quedará claro que se hacía en cada una de ellas se incluiría una ayuda inicial explicando la aplicación.



Figura 5.1.10: Página principal

En la imagen anterior de la página principal de la aplicación también se puede observar el botón "Salir", el cual cumple el requisito *Cerrar Sesión*. Este simplemente llama al método *logout()* que invalida la sesión y lleva al usuario a la página de Iniciar Sesión. Otro detalle a destacar es que la sección Gestionar Usuarios se sabe que solo está disponible para los ADMIN, para eso se utiliza el atributo *rendered* de los componentes JSF, que nos permite establecer una condición lógica para determinar si el elemento se debe mostrar (Figura 5.1.11).

```
<p:menuItem value="Gestionar Usuarios" outcome="gestionarUsuarios" icon="icon-gu"
rendered="#{nom.rol eq 'ADMIN'}" id="gestUser" />
```

Figura 5.1.11: Uso del atributo rendered

Esto es posible gracias a que al iniciar sesión se incluye el rol en la sesión, entonces solo se tendría que comprobar si es ADMIN o USER. Para poder cumplir el requisito de Cambiar Contraseña, pulsando en el nombre de usuario se mostrará un diálogo, donde se tendrá que introducir la contraseña actual y la contraseña nueva, todo esto para que si un usuario indeseado accede a la cuenta tenga necesariamente que saber la contraseña actual.

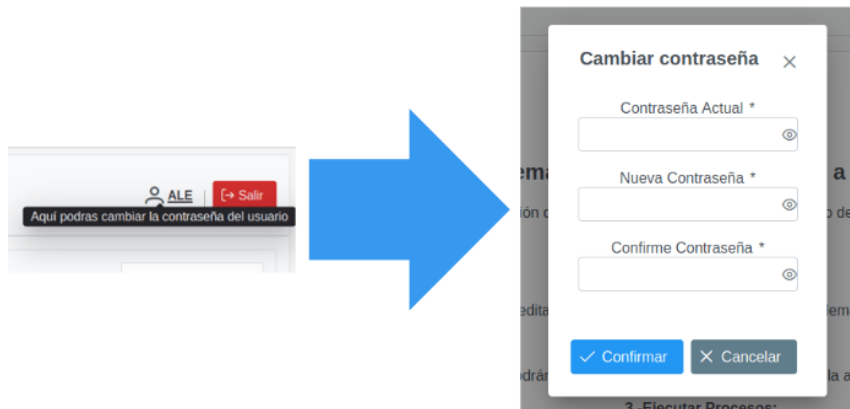


Figura 5.1.12: Cambiar contraseña

Esto se puede llevar a cabo haciendo uso del EntityManager y, concretamente, la operación *merge*, que ya se vio que se utilizaba para actualizar los datos de una entidad. En la Figura 5.1.12 se puede observar otro recurso, el texto que aparece al colocar el cursor encima de un elemento. Este se ha utilizado para incluir más ayudas en la propia aplicación, recibe el nombre de *tooltip* y su uso será recurrente en la vista.

El siguiente requisito a cumplir en esta iteración fue Gestionar Usuarios, para el cual se pensó que sería adecuado una tabla que mostrará todos los usuarios registrados y donde se podría eliminar, modificar y crear usuarios, pero con algunas características determinadas como no poder visualizar la contraseña de los usuarios, pero sí poder cambiarla en caso de haber sido olvidada por el usuario en cuestión. Para la tabla se utilizó el recurso de PrimeFaces *dataTable*, que permite mostrar el recurso deseado de uno en uno a partir de una lista (*java.util.List*) y después se le pueden incluir elementos como la paginación, la selección de las filas y mucho más.

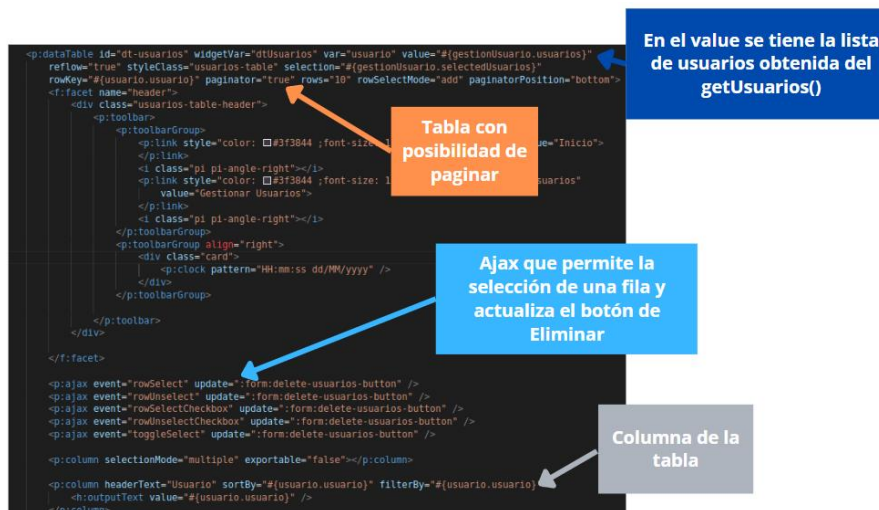


Figura 5.1.13: Primefaces datatable

Para obtener los usuarios se utilizó el *getUsuarios()* del EJB del cual se habló en la parte inicial de esta iteración y que devuelve en una lista todos los usuarios de la base de datos, gracias a que realiza una operación SELECT y que utiliza la anotación *@PostConstruct* en el método que llamada al *getUsuarios()* del EJB, esta anotación se utiliza para asegurar que está disponible el recurso de la lista de usuarios antes de empezar a llamar al resto de operaciones de gestión. Para el resto de operaciones de gestión se utiliza lo mencionado en la parte del EJB (*create, update, delete*). Esta página de Gestionar Usuarios quedó de la forma:

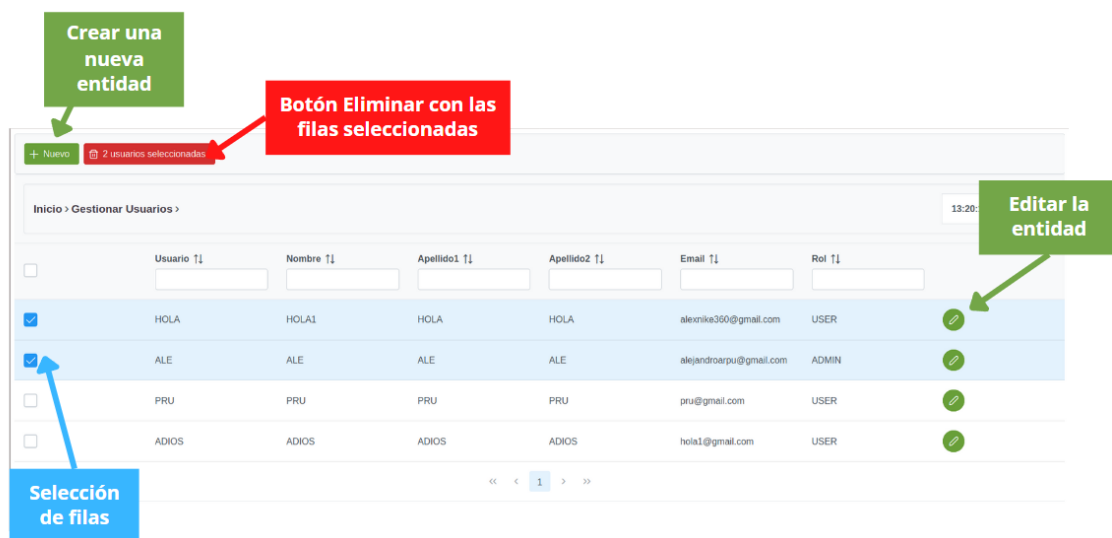


Figura 5.1.14: Página Gestionar Usuarios

El botón Eliminar es un poco peculiar solo se activa si se selecciona alguna fila gracias a Ajax, así permite eliminar varias filas de una vez y, además, permite que no se pulse el botón sin haber seleccionado fila alguna, ya que no tendría sentido. También se pueden apreciar los filtros encima de cada uno de los atributos, para poder filtrar por el atributo que se desee y obtener el resultado más acertado posible.

Para crear y editar se abrirá un diálogo, como se ha visto en anteriores funcionalidades con la simple diferencia de que en editar guarda los datos de la fila seleccionada y en crear no.

The image shows two side-by-side form panels. The left panel, titled 'Crear usuario', contains input fields for 'Usuario *', 'Contraseña *', 'Confirme Contraseña *', 'Nombre *', 'Apellido1 *', 'Apellido2 *', and 'Email *'. It also has radio buttons for 'Rol *' with options 'Usuario' and 'Administrador'. The right panel, titled 'Modificar datos', contains similar fields but with pre-filled values: 'Usuario *' is 'ADIOS', 'Nueva Contraseña' and 'Confirme Contraseña' are empty, 'Nombre *' is 'ADIOS', 'Apellido1 *' is 'ADIOS', 'Apellido2 *' is 'ADIOS', and 'Email *' is 'hola1@gmail.com'. Both panels have 'Aceptar' and 'Cancelar' buttons at the bottom.

Figura 5.1.15: Diferencias entre crear y modificar

La distinción entre ambas vistas se puede hacer muy fácilmente: en el botón de crear se llama a un método que utiliza el constructor sin argumentos (*new Usuario()*) y en el caso de editar se utiliza un elemento de tipo *Usuario* al cual se le ha llamado *selectedUsuario* y gracias al getter se puede traer el *Usuario* con todos sus atributos.

Esta primera iteración supuso más complicaciones que el resto, ya que fue el primer acercamiento con las tecnologías de la vista y requirió de mucha formación previa. El resultado de esta primera iteración fue satisfactorio y al final se consiguieron completar todos los objetivos planteados al inicio. Algunos de los problemas a destacar fueron la inclusión de elementos en la sesión (como el rol y el usuario) y el envío de correos que requirió de ajustes previos de la cuenta de *Gmail*, ya que las políticas de Google sufrieron un cambio respecto al envío de correo desde aplicaciones externas.

El resto de iteraciones van a estar enfocadas en desarrollar cada una de las secciones del menú principal.

5.2. Segunda iteración

En esta segunda iteración se ha buscado completar la sección del menú de la página principal *Gestionar Datos*, que está dedicada a cumplir todos los requisitos que estén

relacionados con gestionar una entidad, a excepción de gestionar usuarios que cuenta con su propia sección y ya se comentó en la primera iteración. Los requisitos a cumplir en esta segunda iteración se corresponden por tanto, con cualquier tipo de gestión:

- Gestionar Titulación, Gestionar Asignaturas, Gestionar Grupos, Gestionar Aulas y Gestionar Grupos Asignaturas.

Para poder cumplir cada uno de estos requisitos se creó una página nueva por cada uno. Para poder acceder a cada uno simplemente se pensó en una serie de botones minimalistas e intuitivos, donde si se coloca el cursor encima cambian de color gracias a la propiedad hover de css, se pueden observar mejor en la siguiente figura:

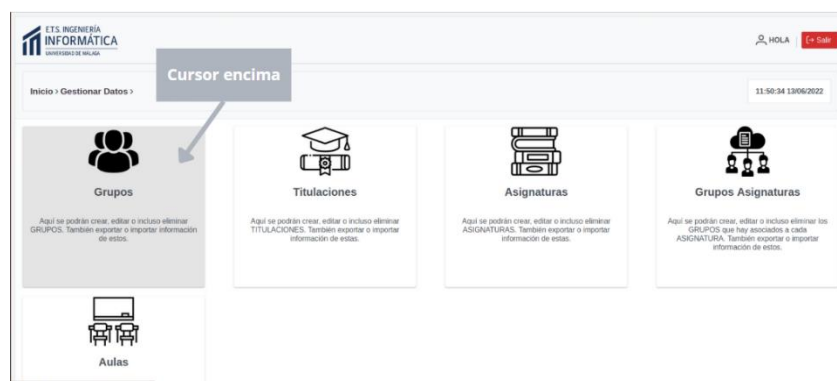


Figura 5.2.1: Página Gestionar Datos botones con hover

Y como se ve se incluye una pequeña ayuda explicativa en cada uno de los botones, para que quede claro que se puede hacer dentro de cada uno. Dentro de estos *Gestionar* se han utilizado tablas al igual que en el Gestionar Usuarios de la primera iteración por lo tanto, la lógica es prácticamente idéntica, ya que se sirven de los métodos del EJB create, delete, update y get. Aquí es necesario destacar que la anotación de ámbito de los backing bean de cada Gestionar será *@ViewScoped*, ya que es interesante que solo exista el bean mientras el usuario se mantenga en la vista de Gestionar, si no cargaría demasiado el rendimiento de la aplicación. Por lo tanto, la vista será prácticamente idéntica a la de Gestionar Usuarios, destacando dos diferencias principales la de exportar datos e importar datos:

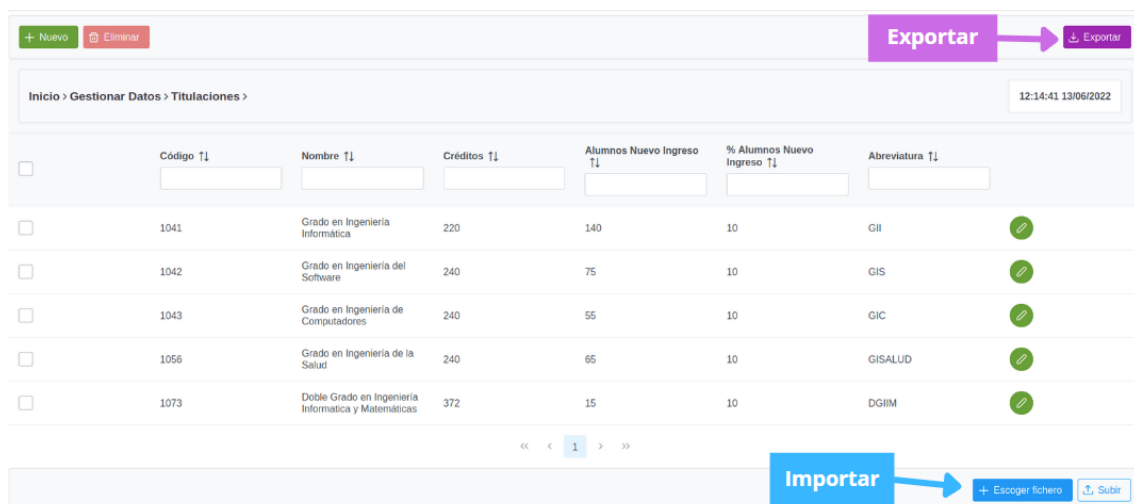


Figura 5.2.2: Página Titulaciones

La funcionalidad extra de Exportar fue muy sencilla de implementar gracias al componente `dataExporter`, que se trata de un botón que permite exportar una `dataTable` y la cual no necesita de la programación de una parte lógica, simplemente incluyendo el componente en la página *JSF* y la librería *apache poi* en el `pom.xml`:

```
<p:commandButton value="Exportar" icon="pi pi-download" styleClass="ui-button-help" ajax="false">
  <p:dataExporter type="xlsx" target="dt-titulaciones" fileName="titulaciones" />
</p:commandButton>
```

Figura 5.2.3: DataExporter Primefaces

Como se ve en la figura anterior es necesario incluir el tipo de fichero que se desea obtener en nuestro caso formato *xlsx* al querer un fichero Excel. Para la funcionalidad de importar fueron necesarios algunos pasos más, el primer paso es incluir el motor *FileUpload*, para ello se modifica el `web.xml` de la capa web:

```
<context-param>
  <param-name>primefaces.UPLOADER</param-name>
  <param-value>commons</param-value>
</context-param>
```

Figura 5.2.4: Motor FileUpload

Y más tarde los filtros del *FileUpload*, también dentro de `web.xml`:

```
<filter>
  <filter-name>PrimeFaces FileUpload Filter</filter-name>
  <filter-class>org.primefaces.webapp.filter.FileUploadFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>PrimeFaces FileUpload Filter</filter-name>
  <servlet-name>Faces Servlet</servlet-name>
</filter-mapping>
</filter>
```

Figura 5.2.5: Filtros FileUpload

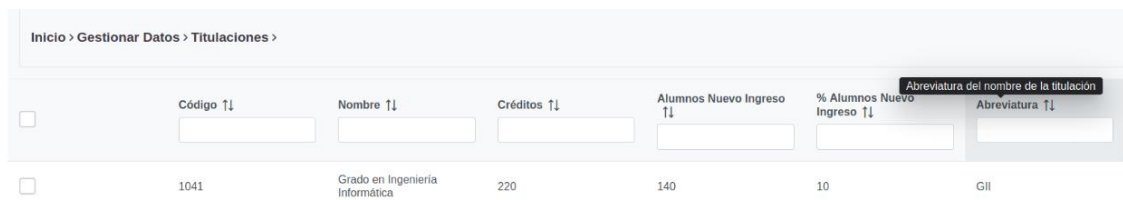
En la página JSF, van a ser necesarios dos componentes un `fileUpload`, que coge el fichero subido y un botón `commandButton` que llama al método para subir el fichero:

```
<p:fileUpload value="#{crudTitulacion.file}" mode="simple" skinSimple="true"
  allowTypes="/(\\.|\\/)(xlsx)$/" validateContentType="true" accept=".xlsx"
  label="Escoger fichero" />

<p:commandButton id="fichero" value="Subir" ajax="false" action="#{crudTitulacion.upload}"
  styleClass="p-mt-3 ui-button-outlined p-d-block" icon="pi pi-upload" />
```

Figura 5.2.6: Componentes JSF para la subida de un fichero

En el componente `fileUpload` se permite escoger el tipo de fichero que se quiere que se suba (*allowTypes* y *accept*), en este caso *xlsx* y así se evitarán posibles errores por ficheros que no son del formato adecuado. El método de lectura llamado por el botón en la propiedad *action* se sirve de *apache poi*, para poder recorrerse el fichero *xlsx* por filas y por celdas y cuando llega al final de una fila se crea una nueva entidad con el uso del método del EJB *create*. En este tipo de páginas de gestionar se han utilizado *tooltips* para los atributos que puedan generar duda y así poder explicar cada uno de forma detallada:



	Código ↑↓	Nombre ↑↓	Créditos ↑↓	Alumnos Nuevo Ingreso ↑↓	% Alumnos Nuevo Ingreso ↑↓	Abreviatura ↑↓
☐	1041	Grado en Ingeniería Informática	220	140	10	GII

Figura 5.2.7: Tooltip en página titulación

En la creación de entidades con relaciones hay que destacar la existencia de un `SelectOneMenu`, el cual incluirá cada uno de los elementos de la relación para evitar el incluir elementos que no existan (Ej: Si asignatura está relacionada con titulación, en la sección de creación de asignatura es conveniente que aparezcan cada una de las titulaciones disponibles, ni más ni menos). Eso se lleva a cabo de la misma forma que la lista de datos, con un *get* del EJB:

Figura 5.2.8: SelectOneMenu titulaciones

El resto de páginas de esta sección de Gestionar son exactamente idénticas, en cuanto a la vista se refiere. Las diferencias a recalcar se encuentran en la parte de JPA, donde hay entidades que utilizan la clave primaria compuesta y, además, incluyen relaciones, es el caso de la entidad Asignatura, por tanto conviene explicar ambas.

Para poder mapear una clave primaria compuesta, se necesita incluir la etiqueta `@IdClass` en la entidad y crear una clase con los atributos que sean clave primaria, esta clase deberá definir `hashCode`, `equals`, `getters`, `setters` y constructores. Por otro lado en la clase Asignatura se debe incluir `@Id` para cada uno de esos atributos que forman la clave primaria:

```

@Entity
@IdClass(Asignatura.AsignaturaId.class)
public class Asignatura implements Serializable {}

public static class AsignaturaId implements Serializable {

    private Integer codigo;
    private Integer titulacionAsignatura;

    @Override
    public String toString() {
        return "AsignaturaId [codigo=" + codigo + ", titulacionAsignatura=" + titulacionAsignatura + "]";
    }

    public AsignaturaId() {
        super();
    }

    public AsignaturaId(Integer codigo, Integer titulacionAsignatura) {
        this.codigo = codigo;
        this.titulacionAsignatura = titulacionAsignatura;
    }

    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((codigo == null) ? 0 : codigo.hashCode());
        result = prime * result + ((titulacionAsignatura == null) ? 0 : titulacionAsignatura.hashCode());
        return result;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
    }
}

@Column(name = "REFERENCIA", nullable = false)
private String referencia;
@Id
@Column(name = "CODIGO", nullable = false)
private Integer codigo;
@Column(name = "CURSO")
private Integer curso;
@Column(name = "CREDITOS", nullable = false)
private Float credits;
@Column(name = "DURACION", nullable = false)
private String duracion;
@Column(name = "NOMBRE", nullable = false)
private String nombre;
@Column(name = "OFERTADA", nullable = false)
private String ofertada;
@Column(name = "CUATRIMESTRE")
private Integer cuatrimestre;
@Column(name = "CARACTER")
private String caracter;
@Id
@ManyToOne(optional = false, fetch = FetchType.EAGER)
@JoinColumn(name = "TITULACION")
private Titulacion titulacionAsignatura;

```

Figura 5.2.9: Requisitos clave primaria compuesta

En lo que respecta a las relaciones, se pueden utilizar las anotaciones *@ManyToOne*, *@ManyToMany*, *@OneToMany* o *@OneToOne* (dependiendo del tipo). Una propiedad muy usada en esta anotación ha sido el *fetch*, que determina como se carga una relación:

- FetchType Eager: Carga la relación con el resto de atributos de la entidad.
- FetchType Lazy: Carga la relación si se accede al método *get* de la misma.

En el caso de la entidad *Asignatura* se ve que se utiliza la *ManyToOne* con *Titulación* y la entidad *Titulación* tendrá una lista de asignaturas y utilizará la anotación *@OneToMany*. El resto de entidades de la aplicación utilizan de la misma forma las anotaciones de las relaciones. Esta iteración se daría por finalizada en el desarrollo de la última entidad a gestionar que se puede observar en la Figura 5.2.1 y es la entidad *Aula*.

En esta última iteración, al haber implementado ya el Gestionar Usuarios no surgieron muchas complicaciones, la única a destacar fue con la funcionalidad de importar que en un principio no cogía el fichero y al final se resolvió utilizando un tipo especial de atributo dedicado a este tipo de ficheros que recibe el nombre de *UploadedFile*. También se tuvieron problemas cuando en los ficheros que se importaban había celdas sin valor, para solucionarlo se utilizó *CREATE_NULL_AS_BLANK* que devuelve una celda de tipo "blank" y evita excepciones *NullPointerException*:

```
for (int j = 0; j < numCeldas; j++) {  
    Cell cell = row.getCell(j, MissingCellPolicy.CREATE_NULL_AS_BLANK);
```

Figura 5.2.10: *CREATE_NULL_AS_BLANK*

5.3. Tercera iteración

La tercera iteración está dedicada a la parte de Visualizar Datos. En esta parte no hay muchas diferencias significativas, ya que se utilizarán tablas para cada una de las entidades a visualizar y la lógica utilizada será un *get* del EJB que utilizará una consulta a la base de datos. En esta iteración se han completado desde el requisito RF-08 al RF-14, es decir todos los requisitos relacionados con la visualización.

Para la página de la sección Visualizar Datos se ha realizado algo idéntico a la sección de Gestionar Datos con botones con ayudas y con el uso de hover en el css, ya que se vio que era lo más intuitivo y visual:

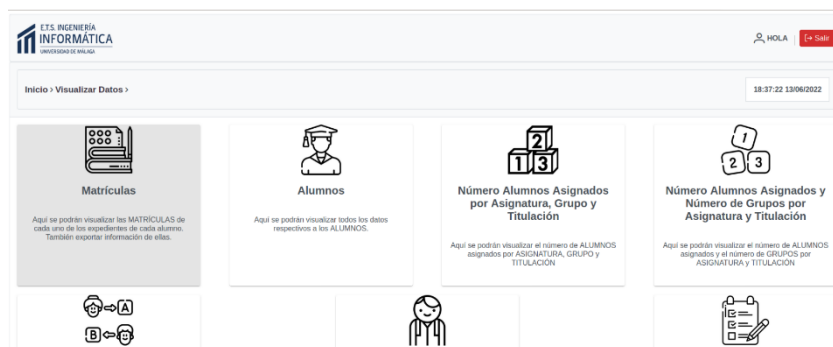


Figura 5.3.1: Página Visualizar Datos

Estas páginas de visualizar datos difieren de las de Gestionar en que no incluyen las posibilidades de crear, editar y eliminar entidades, solo se pueden visualizar y exportarlas. En estas páginas de visualizar se puede acceder a cada fila de la tabla para poder ver la información en detalle:

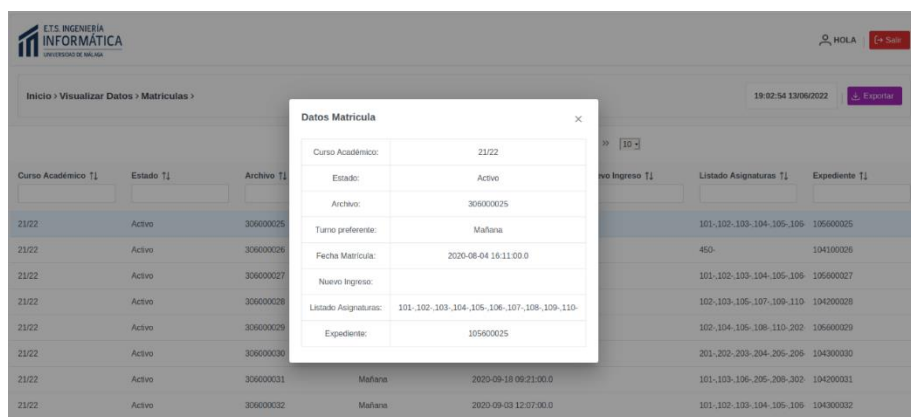


Figura 5.3.2: Selección de una fila

Esto se puede hacer gracias al uso de Ajax de nuevo, ya que tiene el evento de selección que cuando se selecciona una fila, se abra un diálogo con los datos de la fila en concreto:

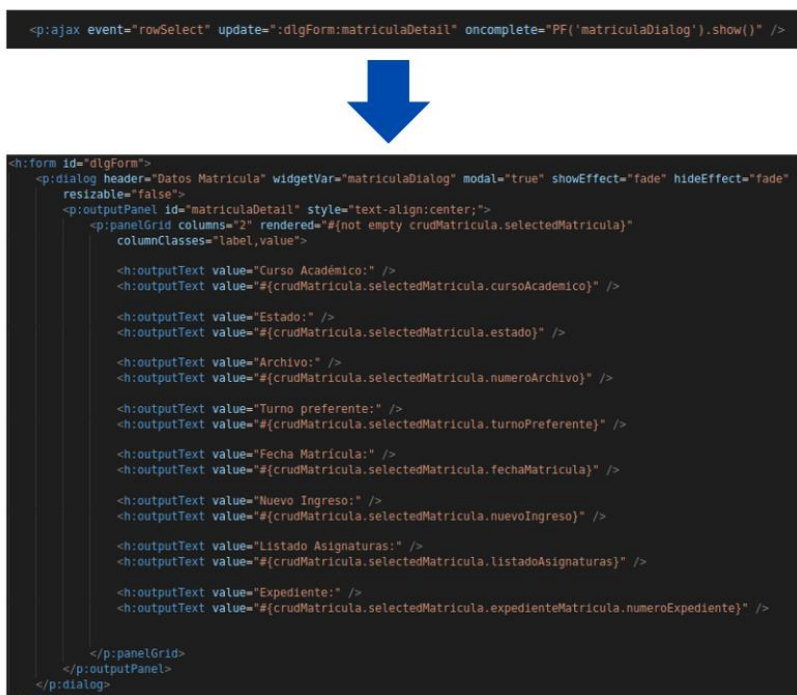


Figura 5.3.3: Ajax rowSelect al completarse abre el dialog

Esta iteración como se ve no supuso complicación alguna, al repetir lo mismo que en las anteriores iteraciones, con la diferencia de no poder gestionar la entidad.

5.4. Cuarta iteración

La cuarta iteración está destinada a completar la sección de Ejecutar Procesos y el resto de requisitos de la aplicación que son los relacionados con la ejecución de los procesos de la asignación, es decir del requisito RF-15 al RF-19. La asignación de grupos a alumnos como ya se dijo, debe seguir un orden concreto, por lo que se pensó en incluir una serie de pasos ordenados con números. Estos pasos están formados por procesos y por importaciones de datos necesarios para llevar a cabo la asignación.



Figura 5.4.1: Página Ejecutar Procesos

En esta sección de Ejecutar Procesos se incluyen funcionalidades ya vistas como pueden ser:

- La importación de los datos es idéntica a la que se encuentra en la sección de Gestionar Datos. Con la peculiaridad de que se puede exportar una plantilla con el formato necesario para la importación.
- La exportación de datos, que en este caso en vez de utilizar `dataExporter` de Primefaces, se ha implementado la lógica directamente con `apache poi`.
- La visualización de datos en la parte de Datos Ejecución, que es una sección idéntica a la parte de Visualizar Datos.

Para la ejecución de estos procesos se ha utilizado `EntityManager` y su operación de `createStoredProcedureQuery`, en la cual se especifica el nombre del proceso y lo busca en la base de datos, además permite incluir parámetros con `setParameter`, en caso de que sean necesarios:

```
StoredProcedureQuery query = em.createStoredProcedureQuery(procedureName: "BORRAR_ASSIGNACION")
    .registerStoredProcedureParameter(position: 1, type: String.class, ParameterMode.IN)
    .setParameter(position: 1, tipoBorrar);
query.execute();
```

Figura 5.4.2: CreateStoredProcedureQuery

La ejecución de cada uno de los procesos se lleva a cabo, pulsando en los botones de Ejecutar (se pueden observar en la Figura 5.4.2), por lo que es un proceso sencillo e intuitivo que no requiere de formación previa que era lo que se buscaba.

Otro punto importante en esta parte es que se ha implementado un control de la ejecución y la importación, teniendo en cuenta el *Usuario* y la *Fecha*. Todo esto, para evitar posibles errores durante el proceso de ejecución:



Carga INICIAL de Datos (Nuevo Ingreso)

Se comenzará con una carga inicial de todos los datos para poder realizar las operaciones convenientes, con los datos adecuados. Para ello es necesario seguir los siguientes pasos:

1) Borrar Asignación

Usuario: HOLA
Fecha: 2022-05-24 18:47:12

Modo
PARCIAL

Ejecutar

Figura 5.4.3: Detalles de la ejecución del proceso Borrar Asignación

Esto se ha llevado a cabo, a través de una consulta en el EJB que obtiene los detalles de cada proceso, simplemente sabiendo el nombre del proceso:

```
@Override
public Procesos selectProcesos(String nombre) {
    proceso = new ArrayList<>();
    TypedQuery<Procesos> query = em.createQuery(qlString: "Select p FROM Procesos p WHERE p.nombreProceso = :n",
        resultClass: Procesos.class);
    query.setParameter(name: "n", nombre);
    proceso = query.getResultList();

    return proceso.get(0);
}
```

Figura 5.4.4: Query EJB que extrae el proceso

Uno de los puntos importantes en esta sección es que se incluye un obtener "Resultados", tras cada asignación, con el objetivo de poder obtener la asignación y tener datos como respaldo en caso de que el alumno cambie de opinión en sus selecciones:



Figura 5.4.5: Resultados

También se incluyen ayudas en todos los paneles de la ejecución como se puede observar en la Figura 5.4.3, pero además se ha incluido una explicación de cada proceso y algunos detalles a tener en cuenta de cada uno, para que evitar dudas sobre la ejecución de estos procesos. Esto se ha realizado gracias al recurso ya utilizado `tooltip` de Primefaces, que permite ver la ayuda cuando se posa el cursor sobre el símbolo de pregunta amarillo. Este se puede observar en cada uno de los títulos de la ejecución de los procesos.



Figura 5.4.6: Ayudas en cada proceso

Esta iteración ha consistido en repetir muchas de las funcionalidades ya vistas en anteriores iteraciones, por lo que hay pocos problemas que comentar. Y con esta

iteración se habrían acabado todos los requisitos a cumplir, pero se pensó en incluir una iteración más en busca de posibles mejoras.

5.5. Mejoras

Este último paso está dedicado a mejorar algunos problemas de la aplicación, los cuales se observaron durante el uso de esta. El primer problema que se vio fue de seguridad y es que si un usuario introducía la URL de cualquier página de la aplicación podía acceder sin haber iniciado sesión previamente. Esto era un claro agujero de seguridad que había que solucionar, para ello se utilizó un *Filtro*.

En esta clase Filtro se debe utilizar la anotación `@WebFilter`, para poder definir ese filtro y, además, debe utilizar la interfaz *Filter*, que nos obliga al desarrollo del método `doFilter()`, donde se definirá como se van a filtrar las peticiones. En este caso, como al iniciar sesión se incluye el valor nombre en la sesión, lo que se ha hecho es verificar que ese valor no es nulo, en el caso de ser nulo se vuelve a la página de Iniciar Sesión. Un detalle que hay que tener en cuenta, es que se debe permitir el paso de Ajax, aunque no se observe el valor nombre en la petición y es que si no se permite esto, no cargarían los componentes en la página de Iniciar Sesión, ya que Primefaces utiliza Ajax para muchos de sus componentes. Para entender mejor lo explicado es conveniente observar el código:

```
@WebFilter("/Secretaria-WAR/**")
public class LoginFilter implements Filter {

    private static final String AJAX_REDIRECT_XML = "<?xml version='1.0' encoding='UTF-8'?'>"
        + "<partial-response><redirect url='%s'></redirect></partial-response>";

    @Override
    public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain)
        throws ServletException, IOException {
        HttpServletRequest request = (HttpServletRequest) req;
        HttpServletResponse response = (HttpServletResponse) res;
        HttpSession session = request.getSession(create: false);
        String loginURL = request.getContextPath() + "/login.xhtml";

        boolean loggedIn = (session != null) && (session.getAttribute(name: "nombre") != null); // aqui es donde vemos si el
                                                                                             // parametro nombre y la sesion
                                                                                             // tienen valor

        boolean loginRequest = request.getRequestURI().equals(loginURL);
        // boolean registroRequest = request.getRequestURI().equals(registroURL);

        boolean resourceRequest = request.getRequestURI()
            .startsWith(request.getContextPath() + ResourceHandler.RESOURCE_IDENTIFIER + "/");
        boolean ajaxRequest = "partial/ajax".equals(request.getHeader(name: "Faces-Request")); // para el ajax

        if (loggedIn || loginRequest || resourceRequest) {
            if (!resourceRequest) { // Prevent browser from caching restricted resources.
                response.setHeader(name: "Cache-Control", value: "no-cache, no-store, must-revalidate"); // HTTP 1.1.
                response.setHeader(name: "Pragma", value: "no-cache"); // HTTP 1.0.
                response.setDateHeader(name: "Expires", date: 0); // Proxies.
            }
            chain.doFilter(request, response); // Continúa con la sesion correctamente
        } else if (ajaxRequest) {
            response.setContentType(type: "text/xml");
            response.setCharacterEncoding(charset: "UTF-8");
            response.getWriter().printf(AJAX_REDIRECT_XML, loginURL);
        } else {
            response.sendRedirect(loginURL); // Volvemos al login si no se logea o son falso los parametros.
        }
    }
}
```

Figura 5.5.1: LoginFilter

En este filtro, además se debe incluir en el web.xml para que se tenga en cuenta en la aplicación y, por otro lado se debe incluir a que páginas se quiere aplicar, en este caso a todas las páginas porque no se puede acceder a una página sin iniciar sesión.

Otro problema de seguridad en el que se utilizó la misma solución, fue en el caso de Gestionar Usuarios, que se sabe que solo pueden acceder los usuarios con rol ADMIN. Para ello se pensó otra vez en el uso de un filtro, pero se tenía en cuenta el valor *rol* de la sesión en vez del nombre (que se incluye también al iniciar sesión) y se comprobaba si este valor rol es ADMIN o USER para permitir el paso del usuario a la página o no.

Una mejora en cuanto al despliegue fue la inclusión en el dockerfile de una instrucción que copiara el fichero .ear en la ruta correspondiente en el contenedor y cada vez que se ejecutará el comando "*docker-compose up*" la aplicación ya estuviera disponible, sin la necesidad de interactuar con la vista administrativa del servidor de aplicaciones Wildfly.



```
1 FROM jboss/wildfly:18.0.1.final
2 #RUN yum install -y xorg-x11-fonts-75dpi xorg-x11-fonts-Type1 openssl git-core fontconfig
3 RUN /opt/jboss/wildfly/bin/add-user.sh admin Admin#70365 --silent
4 EXPOSE 8080
5 EXPOSE 9990
6 EXPOSE 8787
7 RUN mkdir -p /opt/jboss/wildfly/modules/system/layers/base/com/oracle/main
8 RUN rm /opt/jboss/wildfly/welcome-content/favicon.ico
9 COPY favicon.ico /opt/jboss/wildfly/welcome-content
10 COPY module.xml ojdbc8.jar /opt/jboss/wildfly/modules/system/layers/base/com/oracle/main/
11 COPY standalone.xml standalone-full.xml /opt/jboss/wildfly/standalone/configuration/
12 COPY Secretaria-EAR-0.0.1-SNAPSHOT.ear /opt/jboss/wildfly/standalone/deployments/
13 CMD ["/opt/jboss/wildfly/bin/standalone.sh", "-b", "0.0.0.0", "-bmanagement", "0.0.0.0", "-c", "standalone-full.xml", "--debug", "--8787"]
14
15
```

Figura 5.5.1: Dockerfile mejorado

Una mejora en cuanto al entendimiento de la aplicación fue el incluir una ventana de carga, con el objetivo de que los posibles usuarios no se desconcertarán en caso que una página se demorara en su carga. Esta ventana de carga se ha utilizado sobre todo en las páginas en las que se cargan las tablas de los datos, es decir en Gestionar y en Visualizar, ya que en ocasiones son elevados los números de filas a cargar de la base de datos:

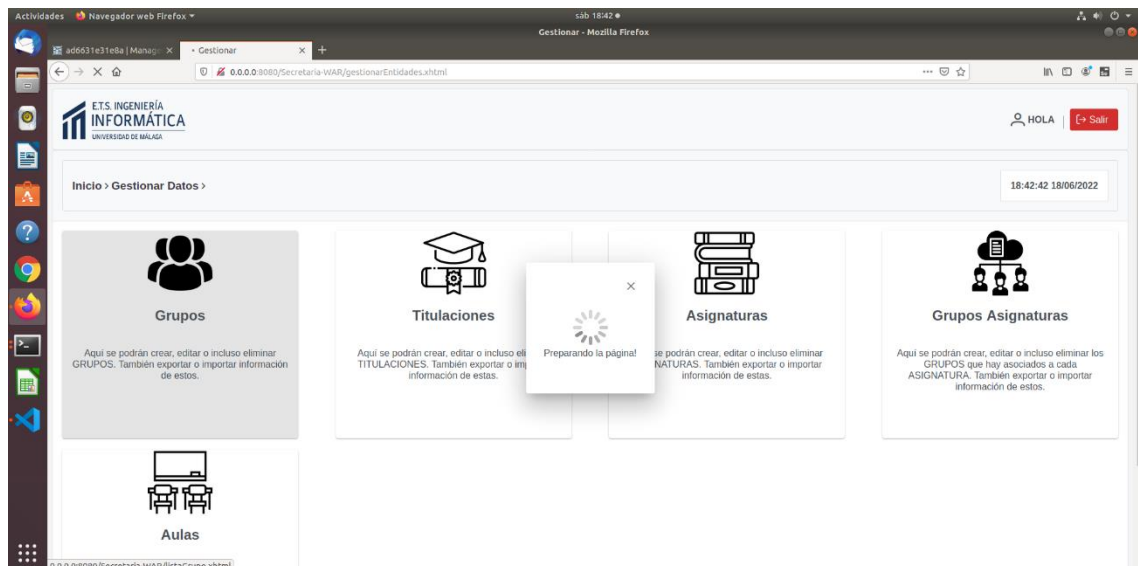


Figura 5.5.3: Ventana de carga

Otra de las mejoras incluidas en las páginas de Gestionar es la inclusión de asteriscos en las columnas con el objetivo de poder visualizar los atributos que son obligatorios rellenar y cuál es la clave de cada entidad, todo esto para evitar importaciones de datos que no cumplan el formato. Dos asteriscos indicarán el atributo que es clave y un asterisco indicará que es obligatorio el completar ese atributo:

	Codigo** T1	Nombre* T1	Créditos* T1	Alumnos Nuevo Ingreso T1	% Alumnos Nuevo Ingreso T1	Abreviatura T1	
☐	1041	Grado en Ingeniería Informática Antiguo	220	140	10	GII	
☐	1042	Grado en Ingeniería del Software	240	75	10	GIS	
☐	1043	Grado en Ingeniería de Computaciones	240	55	10	GIC	
☐	1056	Grado en Ingeniería de la Salud	240	65	10	GISALUD	
☐	1073	Doble Grado en Ingeniería Informática y Matemáticas	372	15	10	DGIM	
☐	10411	PRUEBA	200			PRU	

Figura 5.5.4: Asteriscos en las columnas

Con esta última parte de mejoras se daría por finalizado todo el desarrollo de la aplicación. El uso de la metodología iterativa e incremental ha permitido que el proceso

de desarrollo se lleve de una forma ordenada y sencilla, consiguiendo al final todas los requisitos necesarios para satisfacer al usuario final de la aplicación.

6

Pruebas

Tras el capítulo de desarrollo de la aplicación, es conveniente explicar cada una de las pruebas que verifican el correcto funcionamiento de los requisitos desarrollados en la parte anterior. Está etapa es de suma importancia, ya que se expone la calidad del producto a la parte interesada. Las pruebas se van a ir realizando de las funcionalidades más importantes del proyecto, explicando en cada caso en que consisten y mostrando capturas de pantalla para verificar la prueba en el caso de que procedan.

Cada una de las pruebas incorporará un identificador único PS-XX (aludiendo a Prueba de Sistema)

6.1. Iniciar Sesión

Identificador	PS-01
Objetivo	Acceder al Sistema de Asignación de Grupos a Alumnos.
Actores	Usuario registrado o usuario no registrado
Precondiciones	El usuario tiene la aplicación instalada en su ordenador y accede a la página de inicial de Login.
Postcondiciones	El usuario puede visualizar la página principal de la aplicación.

Escenario de éxito	1. El usuario se identifica introduciendo los credenciales correctos. 2. El usuario hace click en el botón de Entrar. 3. El usuario accede a la página principal de la aplicación.
Escenario alternativo	1. El usuario introduce los credenciales de forma incorrecta o no está registrado. 2. El sistema informa del error al usuario.

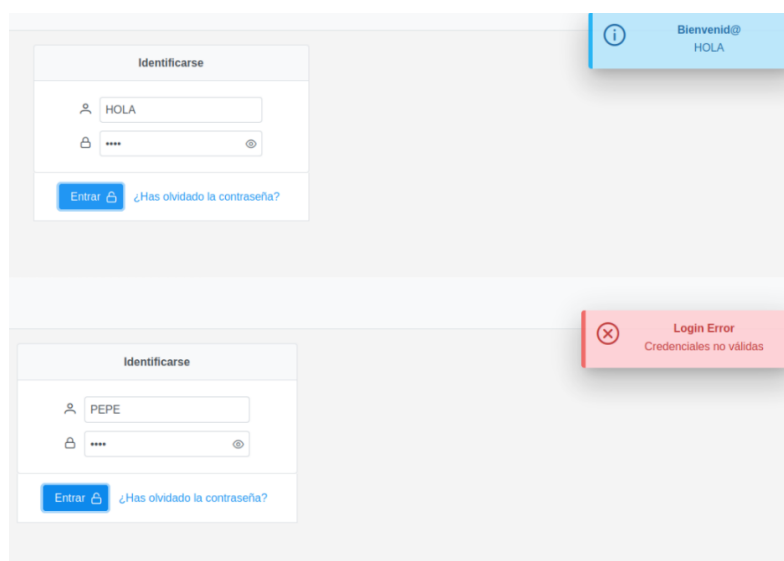


Figura 6.1.1: Escenario de éxito y alternativo respectivamente (Iniciar Sesión)

6.2. Recuperar la contraseña

Identificador	PS-02
Objetivo	Recuperar la contraseña olvidada por el usuario.
Actores	Usuario registrado
Precondiciones	El usuario está en la página inicial de Login.
Postcondiciones	El usuario consigue una contraseña para acceder al sistema.

Escenario de éxito	1. El usuario hace click sobre ¿Has olvidado la contraseña? 2. El sistema muestra la pantalla de recuperar contraseña. 3.1. El usuario introduce un correo registrado. 4.1 El sistema manda un correo al usuario con su nueva contraseña.
Escenario alternativo	3.2. El usuario ha introducido un correo que no está registrado. 4.2 El sistema muestra un mensaje de error.

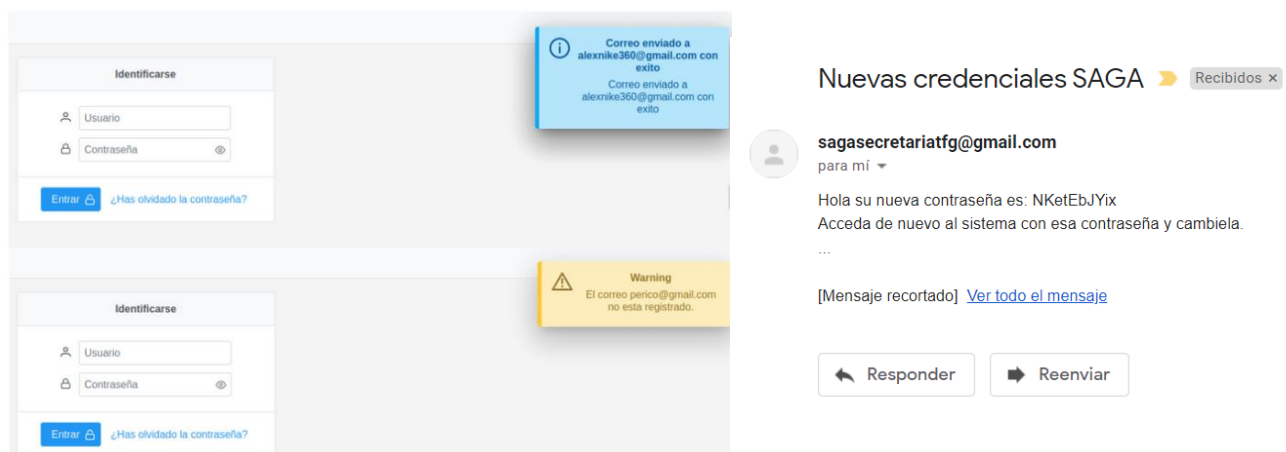


Figura 6.2.1: Resultados obtenidos

6.3. Cambiar contraseña

Identificador	PS-03
Objetivo	Cambiar la contraseña del usuario.
Actores	Usuario registrado
Precondiciones	El usuario se sitúa en la página principal de la aplicación.
Postcondiciones	Se ha conseguido cambiar la contraseña del usuario.

Escenario de éxito	1. El usuario hace click sobre su nombre de usuario. 2. El sistema muestra la pantalla de cambiar contraseña. 3.1. El usuario introduce los datos requeridos y hace click en Aceptar. 4.1. El sistema valida el cambio y notifica al usuario.
Escenario alternativo	3.2 El usuario introduce los datos incorrectamente. 4.2. El sistema muestra un mensaje de error.



Figura 6.2.1: Mensajes mostrados en caso de éxito o fracaso (Cambiar contraseña)

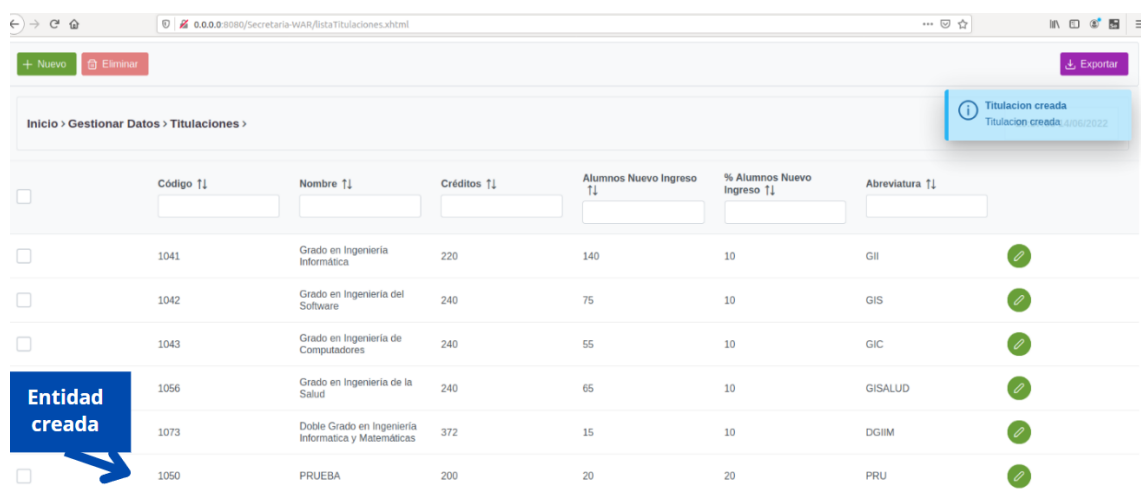
6.4. Gestionar Entidad

Esta prueba se dividirá en varias subpruebas, ya que la parte de gestionar engloba varias funcionalidades que son conveniente probar individualmente.

6.4.1. Crear una entidad

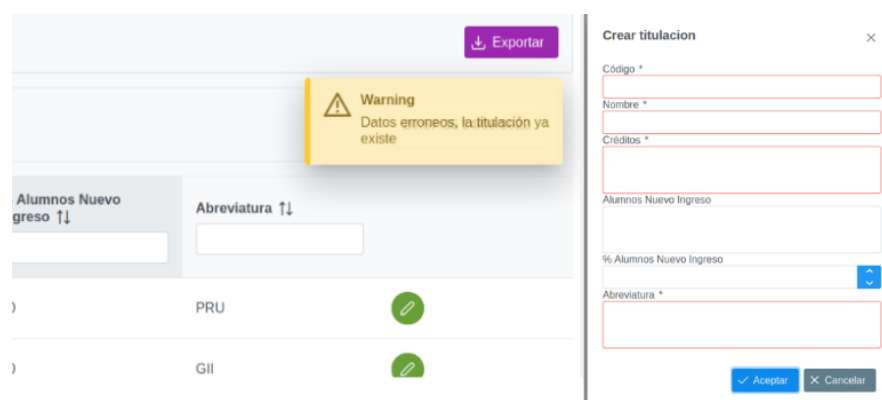
Identificador	PS-04.1
Objetivo	Creación de una entidad en concreto
Actores	Usuario registrado
Precondiciones	El usuario se encuentra en alguna de las páginas de la sección Gestionar Datos de la aplicación.
Postcondiciones	Se ha conseguido crear una entidad y ya es posible su visualización.

Escenario de éxito	1.1. El usuario hace click sobre el botón verde Nuevo e introduce los datos de la entidad. 2.1. El sistema crea la entidad y la muestra.
Escenario alternativo	1.2. El usuario introduce los datos de la entidad incorrectamente o no introduce datos. 2.2. El sistema muestra un error.



	Código T↓	Nombre T↓	Créditos T↓	Alumnos Nuevo Ingreso T↓	% Alumnos Nuevo Ingreso T↓	Abreviatura T↓	
☐							
☐	1041	Grado en Ingeniería Informática	220	140	10	GII	
☐	1042	Grado en Ingeniería del Software	240	75	10	GIS	
☐	1043	Grado en Ingeniería de Computadores	240	55	10	GIC	
☐	1056	Grado en Ingeniería de la Salud	240	65	10	GISALUD	
☐	1073	Doble Grado en Ingeniería Informática y Matemáticas	372	15	10	DGIIM	
☐	1050	PRUEBA	200	20	20	PRU	

Figura 6.4.1.1: Entidad creada



Warning
Datos erroneos, la titulacion ya existe

Crear titulacion

Código *

Nombre *

Créditos *

Alumnos Nuevo Ingreso

% Alumnos Nuevo Ingreso

Abreviatura *

Aceptar Cancelar

Figura 6.4.1.2: Escenario alternativo en caso de error en los datos

6.4.2. Editar una entidad

Identificador	PS-04.2
Objetivo	Editar una entidad en concreto
Actores	Usuario registrado
Precondiciones	El usuario se encuentra en alguna de las páginas de la sección Gestionar Datos de la aplicación.
Postcondiciones	Se ha conseguido editar una entidad y ya se pueden visualizar los cambios.
Escenario de éxito	1.1. El usuario hace click sobre el botón verde en forma de lápiz para editar e introduce los datos de la entidad. 2.1. El sistema edita la entidad con éxito y muestra los cambios en la vista
Escenario alternativo	1.2. El usuario introduce los datos de la entidad incorrectamente o no introduce datos. 2.2. El sistema muestra un error.

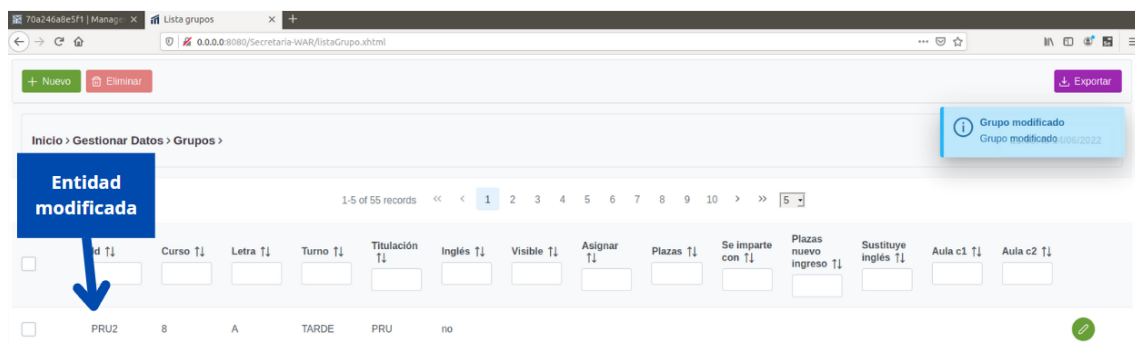


Figura 6.4.1.3: Entidad modificada con éxito

Figura 6.4.1.4: Escenario fracaso al no introducir los datos requeridos

6.4.3. Eliminar una entidad

Identificador	PS-04.3
Objetivo	Eliminar una entidad
Actores	Usuario registrado
Precondiciones	El usuario se sitúa en una de las páginas dedicadas a gestionar datos.
Postcondiciones	Se elimina la entidad deseada por el usuario.
Escenario de éxito	1. El usuario selecciona la entidad que desea eliminar y hace click en el botón Eliminar. 2. El sistema elimina la entidad y actualiza la lista de datos.
Escenario alternativo	En este caso, no hay posibilidad de un escenario alternativo.

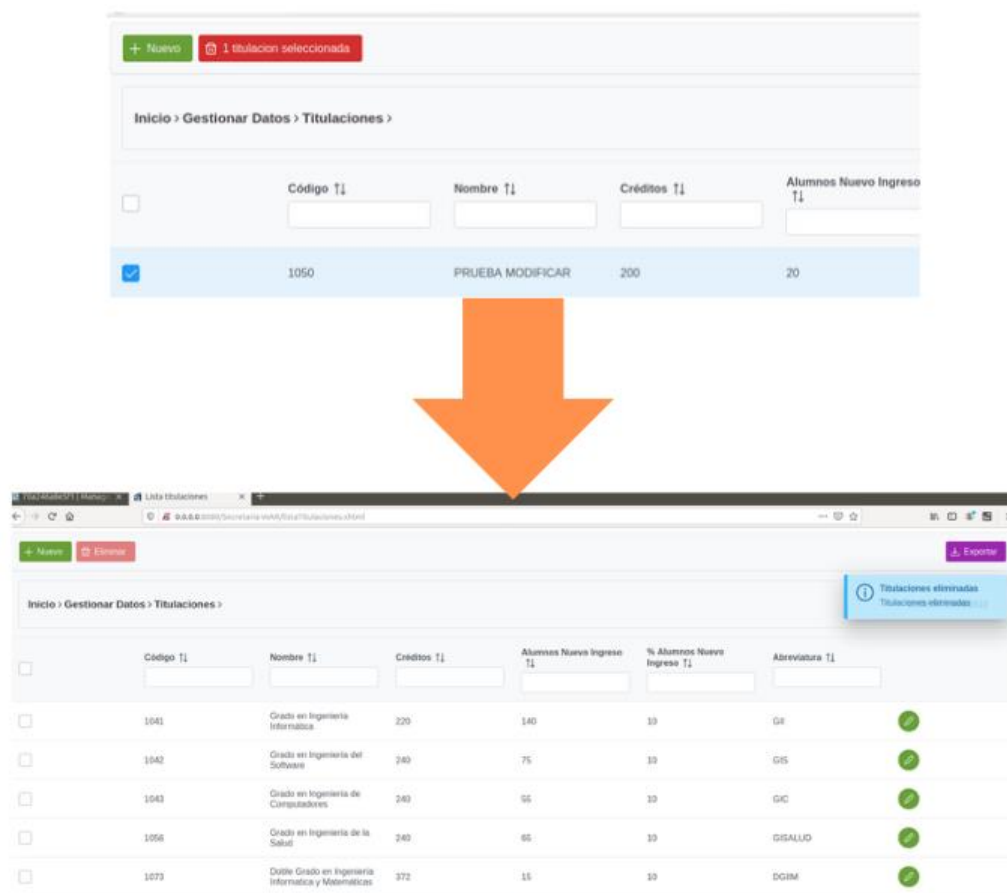


Figura 6.4.3.1: Escenario de éxito (Eliminar una entidad)

6.4.4. Exportar lista de datos

Identificador	PS-04.4
Objetivo	Exportar la lista de los datos en formato excel.
Actores	Usuario registrado
Precondiciones	El usuario se sitúa en una de las páginas dedicadas a gestionar datos.
Postcondiciones	Se obtiene un excel con la lista de datos deseada.
Escenario de éxito	1. El usuario hace click en el botón Exportar. 2. El sistema recibe la señal y exporta la lista a formato excel.
Escenario alternativo	En este caso, no hay posibilidad de un escenario alternativo.

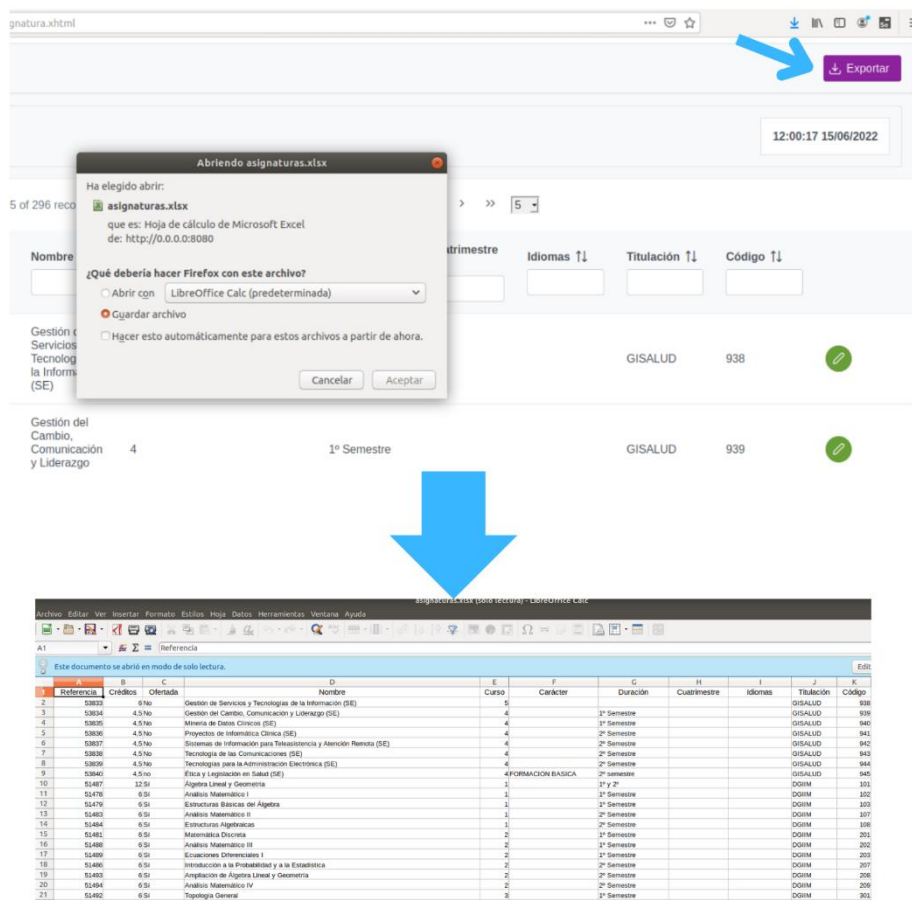


Figura 6.4.4.1: Escenario de éxito (Exportar lista de datos)

6.4.5. Importar datos

Identificador	PS-04.5
Objetivo	Importar datos a una lista de datos
Actores	Usuario registrado
Precondiciones	El usuario se sitúa en una de las páginas dedicadas a gestionar datos.
Postcondiciones	Se consiguen importar los datos y aparecen en la lista
Escenario de éxito	1. El usuario hace click en el botón de Escoger Fichero. 2. El sistema recibe la señal y abre una ventana para poder seleccionar un fichero concreto. 3.1. El usuario selecciona el fichero y hace click en el botón Subir. 4.1. El sistema importa los datos a la lista de datos y muestra todos los datos con los nuevos ya incluidos.
Escenario alternativo	3.2. El usuario selecciona un fichero con datos erróneos. 4.2. El sistema notifica del error y no importa los datos al sistema.

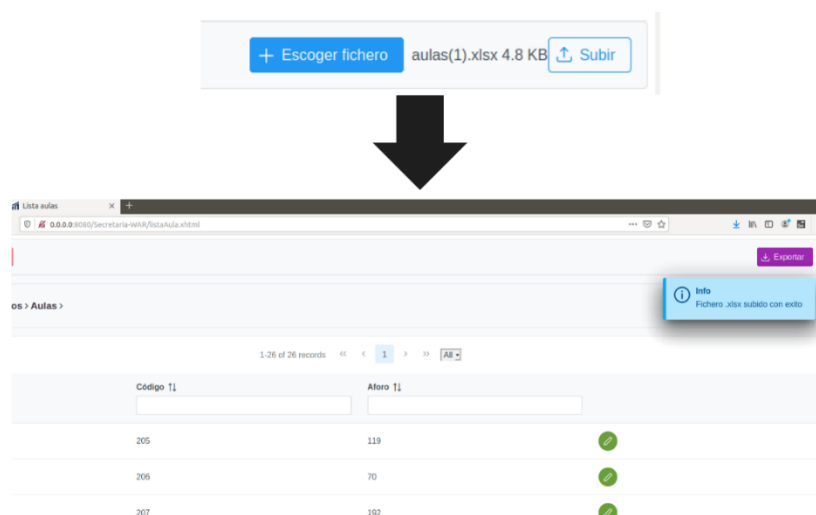


Figura 6.4.5: Escenario de éxito (Importar datos)

6.5. Visualizar Datos

Esta prueba al igual que la anterior, se va a dividir en varias subpruebas para poder abordar cada funcionalidad en detalle.

6.5.1. Visualizar lista de datos

Identificador	PS-05.1
Objetivo	Visualizar lista de datos
Actores	Usuario registrado
Precondiciones	El usuario se sitúa en la sección Visualizar Datos o Gestionar Datos
Postcondiciones	Se consigue visualizar la lista de datos de un tipo de entidad
Escenario de éxito	1. El usuario hace click en una de las páginas de la sección de Visualizar Datos o Gestionar Datos. 2. El sistema muestra la lista de datos según la selección del usuario.
Escenario alternativo	En este caso no hay escenario alternativo de interés.

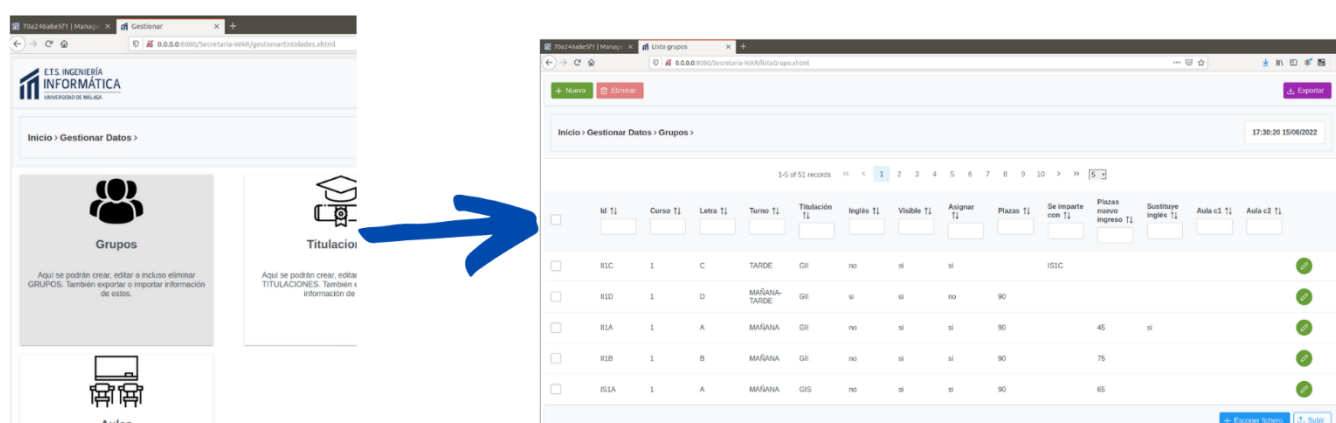


Figura 6.5.1.1: Escenario de éxito (Visualizar lista de datos)

6.5.2. Visualizar Entidad

Identificador	PS-05.2
---------------	---------

Objetivo	Visualizar una entidad en concreto con más detalle
Actores	Usuario registrado
Precondiciones	El usuario se sitúa en una de las páginas dedicadas a visualizar datos.
Postcondiciones	Se consigue visualizar los datos de una fila de una entidad en detalle.
Escenario de éxito	1. El usuario hace click sobre una de las filas de la lista de datos. 2. El sistema recibe la señal y abre una ventana con la entidad a visualizar.
Escenario alternativo	En este caso no hay escenario alternativo de interés.

The screenshot shows a web application interface for 'ETS. INGENIERÍA INFORMÁTICA' at the 'UNIVERSIDAD DE MÁLAGA'. The main content area displays a table with columns: 'Curso Académico', 'Estado', and 'Archivo'. A modal window titled 'Datos Matricula' is open, showing details for a specific record. A blue arrow points to a row in the table, with a text box stating 'Se hace click sobre la fila'.

Datos Matricula

Curso Académico:	21/22
Estado:	Activo
Archivo:	306000025
Turno preferente:	Mañana
Fecha Matricula:	2020-08-04 16:11:00.0
Nuevo Ingreso:	
Listado Asignaturas:	101-102-103-104-105-106-107-108-109-110-
Expediente:	105600025

Figura 6.5.1.1: Escenario de éxito (Visualizar una entidad)

6.5.3. Filtrar datos

Identificador	PS-05.3
Objetivo	Filtrar los datos de una lista por cada uno de los filtros disponibles.
Actores	Usuario registrado.

Precondiciones	El usuario se sitúa en una de las páginas dedicadas a visualizar datos o a gestionar datos.
Postcondiciones	Se consigue visualizar los datos que cumplen con los filtros establecidos.
Escenario de éxito	1. El usuario hace click sobre uno de los filtros disponibles en cada tabla e introduce el filtro deseado. 2. El sistema recibe la señal y filtra todos los datos.
Escenario alternativo	En este caso no hay escenario alternativo de interés.

The screenshot shows a web application interface for 'E.T.S. INGENIERÍA INFORMÁTICA'. The breadcrumb trail is 'Inicio > Ejecutar Procesos > Datos Ejecución > Nuevo Ingreso'. The current page displays a table with 5 columns: 'DNI', 'Expediente', 'Archivo', 'Asignaturas Inglés', and 'Orden'. The 'Expediente' column has a filter input box containing '1042'. Below the table, there is a list of 10 records, each with a DNI, Expediente, Archivo, Asignaturas Inglés, and Orden value.

DNI	Expediente	Archivo	Asignaturas Inglés	Orden
58440224V	104200112	306000112	101,104,109,110	61
69575980A	104200004	306000004	101,104,109,110	67
78541977C	104200022	306000022	101,104,109,110	70
69386416F	104200042	306000042	101,104,109,110	78
88266710G	104200332	306000332	101,104,109,110	84
94223758R	104200335	306000335	101,104,109,110	85
40859951G	104200342	306000342	101,104,109,110	86
11258287R	104200139	306000139	101,104,109,110	87

Figura 6.5.2.1: Escenario de éxito (Filtrar datos)

6.6. Ejecutar procesos

Identificador	PS-06
---------------	-------

Objetivo	Ejecutar cada uno de los procesos dedicados a la asignación.
Actores	Usuario registrado.
Precondiciones	El usuario se sitúa en la página dedica a ejecutar los procesos de la asignación.
Postcondiciones	Se consigue ejecutar el proceso de asignación.
Escenario de éxito	1. El usuario hace click sobre uno de los botones de Ejecutar. 2. El sistema recibe la señal y muestra una ventana de confirmación. 3. El usuario hace click en Si. 4. El sistema ejecuta el proceso, muestra un mensaje de ejecutando y actualiza los datos de ejecución.
Escenario alternativo	En este caso no hay escenario alternativo de interés.

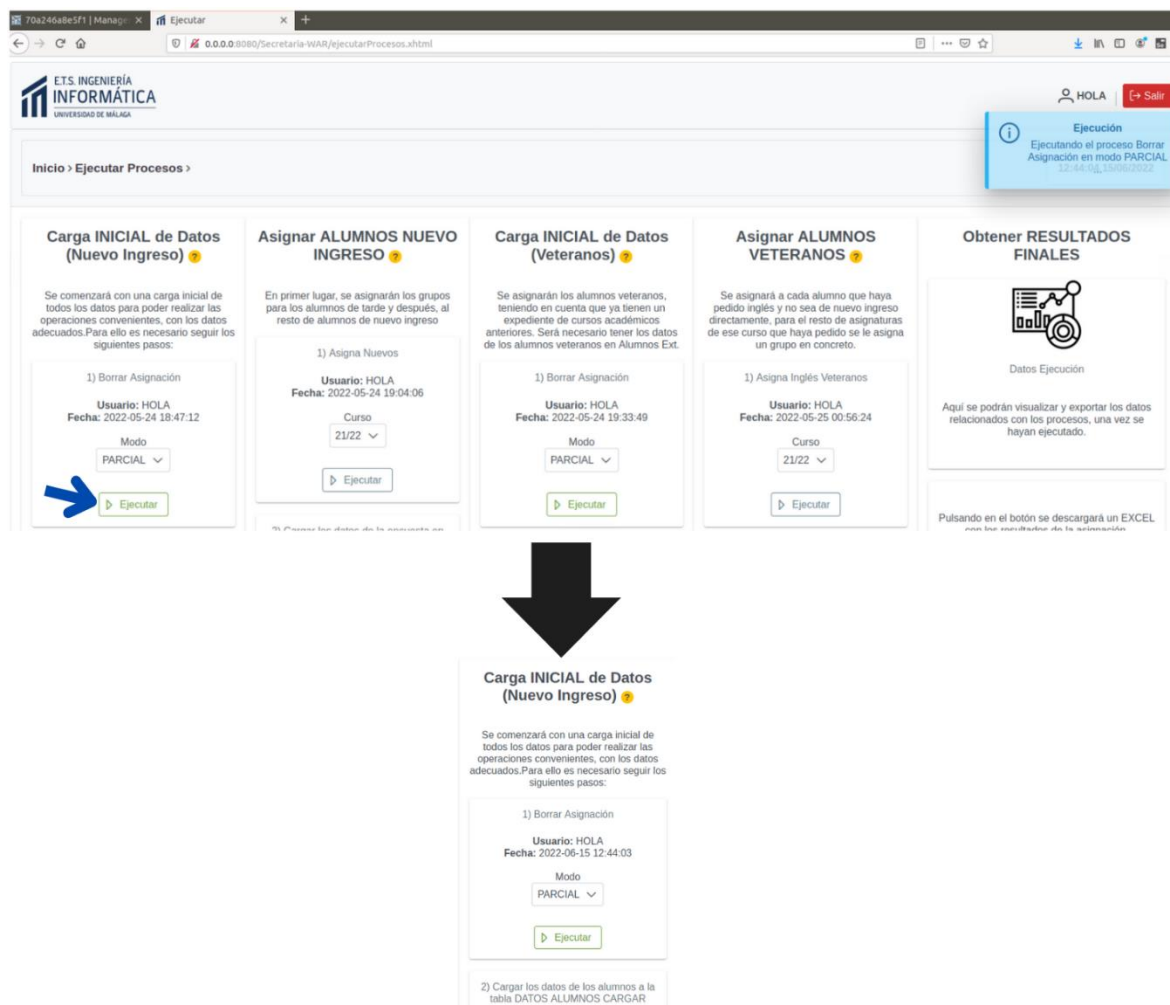


Figura 6.6.1: Escenario de éxito (Ejecución de un proceso)

6.7. Resultados obtenidos

La siguiente tabla recoge cada una de las pruebas realizadas que se han descrito anteriormente con objeto de resumir y esquematizar la etapa de pruebas al completo:

Caso de uso	Responsable	Entorno	Resultados obtenidos	Aprobado
Iniciar sesión	Alejandro Arco Puerto	Aplicación web (PC)	Redirección a la página principal	✓
Recuperar contraseña	Alejandro Arco Puerto	Aplicación web (PC)	Se recibe una nueva contraseña para poder acceder al sistema	✓
Cambiar contraseña	Alejandro Arco Puerto	Aplicación web (PC)	Se modifica la contraseña del usuario que lo solicita	✓
Crear entidad	Alejandro Arco Puerto	Aplicación web (PC)	Se crea un nuevo tipo de entidad en la base de datos	✓
Editar entidad	Alejandro Arco Puerto	Aplicación web (PC)	Se modifica la entidad elegida	✓
Eliminar entidad	Alejandro Arco Puerto	Aplicación web (PC)	Se elimina la entidad	✓
Exportar datos	Alejandro Arco Puerto	Aplicación web (PC)	Se obtiene en formato Excel la lista de datos	✓

Importar datos	Alejandro Arco Puerto	Aplicación web (PC)	Se importan los datos en el sistema y se guardan en la base de datos	✓
Visualizar lista de datos	Alejandro Arco Puerto	Aplicación web (PC)	Se muestra la lista de todos los datos de un tipo de entidad en concreto	✓
Visualizar entidad	Alejandro Arco Puerto	Aplicación web (PC)	Se muestra la fila de la entidad en detalle	✓
Filtrar datos	Alejandro Arco Puerto	Aplicación web (PC)	Se muestra la lista de datos tras aplicarle los filtros correspondientes	✓
Ejecutar procesos de asignación	Alejandro Arco Puerto	Aplicación web (PC)	Se ejecuta un proceso determinado de la asignación de procesos	✓

Conclusiones

En este último capítulo de la memoria se expondrán las conclusiones obtenidas tras el desarrollo de este proyecto. Estas conclusiones tendrán como principal objetivo el evaluar el desarrollo desde el punto de vista de la consecución de los requisitos y objetivos planteados, además se recapitulará lo aprendido y lo que ha supuesto el proyecto a nivel personal. También habrá un capítulo de líneas futuras donde se explicarán posibles mejoras a implementar en la aplicación.

7.1. Conclusiones

El objetivo principal a conseguir con este proyecto era la agilización en el proceso de asignación de grupos a alumnos. Para conseguir implementar este objetivo tan importante decidimos desarrollar una aplicación web adaptada para el propósito a conseguir. Y el objetivo de agilizar la asignación, se dividió en los diferentes requisitos del capítulo 3 donde todos ellos, correctamente implementados, formarían la aplicación completa.

Esta aplicación ha sido implementada basándose en la plataforma Java EE, la cual me ha permitido aprender e identificar cada una de las partes que forman el desarrollo de una aplicación web, desde la persistencia de datos con JPA hasta la capa web con JSF y Primefaces. Por otra parte, también he utilizado la base de datos de Oracle con la que he podido aprender como es el gestionar todos los elementos relacionados con la base de datos y como adaptarlos para poder conseguir una interacción entre base de datos y proyecto Java adecuada. No se puede olvidar lo aprendido con respecto a Maven que

ha facilitado en cantidad la construcción del proyecto para poder desplegarse en el servidor de aplicaciones Wildfly. En lo que respecta a Docker ha favorecido todo lo que es la instalación y despliegue de la aplicación y el resto de tecnologías secundarias como CSS o JavaScript han sido de gran ayuda para completar el proyecto. Todo esto en conjunto ha formado la aplicación web final y se puede decir que se ha conseguido el objetivo principal planteado al inicio de este proceso de desarrollo. Se ha formado una aplicación con un aspecto profesional y minimalista para ofrecer al usuario la mayor facilidad de uso y entendimiento posible.

Además, este proyecto me ha permitido alcanzar el objetivo personal de desarrollar no solo una aplicación, sino una aplicación que se puede utilizar en el mundo real y puede aportar soluciones reales. La consecución de este proyecto me ha permitido aprender y entender muchas tecnologías en detalle las cuales antes no conocía y ha hecho que me enfrente a nuevos retos a los cuales no me había enfrentado antes. En todo este proceso es importante destacar la labor de los tutores que con sus conocimientos y consejos siempre hacían que mejorara algo de la aplicación para que estuviera más cercana a lo real, además siempre me ayudaban y me alentaban a superarme y a superar los problemas encontrados.

En resumen, el desarrollo de este proyecto ha sido todo un éxito a nivel técnico y a nivel personal y me ha permitido entender que en esta profesión no se puede parar de aprender y es lo que lo hace bonito, el poder aprender siempre algo nuevo.

7.2. Líneas futuras

Entre las posibles líneas futuras para ampliar y mejorar los servicios ofrecidos por la aplicación web se podrían destacar:

- Mejora de los tiempos de respuesta: En ocasiones cuando se quieren visualizar una lista de datos que contiene en la base de datos más 10.000 filas se puede demorar la carga. Esto, se podría mejorar desarrollando una implementación de la lista de tipo lazy, lo que evitaría cargar esas filas al completo en la aplicación,

en este caso únicamente se cargarían el conjunto especificado y al filtrar los datos, se podrían obtener los deseados.

- Posibilidad de seleccionar diferentes lenguajes: En este proyecto únicamente se ha utilizado el idioma español, lo cual limita el uso para personas que puedan ser de otros países que no sean de habla hispana. Para esto se podría utilizar el idioma inglés, el cual es un idioma global y por ende, no se limitaría el uso de la aplicación.
- Detección automática de procesos de la base de datos: En la sección de Ejecutar procesos de la aplicación web se podría incluir algún tipo de funcionalidad que detectará la existencia de los diferentes procesos que se pueden encontrar en la base de datos, ya que puede ser que el proceso de asignación en un futuro requiera de la utilización de algún otro proceso, lo cual habría que incluirlo en el código de la aplicación y lo haría un proceso algo tedioso.
- Desarrollo de la aplicación en distintas plataformas: A pesar de que esta aplicación está destinada a la secretaría del centro ETSI Informática, puede darse el caso de que en alguna ocasión sea necesario un trámite rápido y con el teléfono móvil se podría llevar a cabo de una forma más ágil. También hay que destacar que a veces se prefiere una aplicación de escritorio, en vez de una aplicación web.
- Desarrollo de un banco de pruebas automatizadas: Se podrían realizar un conjunto de pruebas automatizadas para los procesos más repetitivos de la aplicación web, ya que se hará de forma más rápida y eficiente. Estas pruebas en conjunto con las manuales (ya realizadas), las cuales son más certeras en el caso de evaluar la usabilidad y la experiencia de usuario, pueden hacer que aumente la calidad del software en gran medida.

Referencias

[1] A. Goncalves, *Beginning Java EE 7*. Berkeley, CA: Apress, 2013. Accedido el 3 de junio de 2022. [En línea]. Disponible: <https://doi.org/10.1007/978-1-4302-4627-5>

[2] C. Yagüe. "Qué es Apache Maven". OpenWebinars.net. <https://openwebinars.net/blog/que-es-apache-maven/> (accedido el 2 de junio de 2022).

[3] E. Viñé Lerma. "Introducción a Primefaces - Adictos al trabajo". Adictos al trabajo. <https://www.adictosaltrabajo.com/2010/06/30/introduccion-primefaces/> (accedido el 30 de mayo de 2022).

[4] Colaboradores de los proyectos Wikimedia. "CSS - Wikipedia, la enciclopedia libre". Wikipedia, la enciclopedia libre. <https://es.wikipedia.org/wiki/CSS> (accedido el 30 de mayo de 2022).

[5] R. Ramos. "¿Qué es JavaScript y para qué sirve?" Agencia de Marketing Digital Sevilla - Rafa Ramos. <https://soyrafaramos.com/que-es-javascript-para-que-sirve/> (accedido el 30 de mayo de 2022).

[6] "¿Qué es SQL Developer? | Oracle España". Oracle | Cloud Applications and Cloud Platform. <https://www.oracle.com/es/tools/technologies/whatis-sql-developer.html> (accedido el 3 de junio de 2022).

[7] Colaboradores de los proyectos Wikimedia. "Visual Studio Code - Wikipedia, la enciclopedia libre". Wikipedia, la enciclopedia

libre. https://es.wikipedia.org/wiki/Visual_Studio_Code (accedido el 2 de junio de 2022).

[8] Colaboradores de los proyectos Wikimedia. "GitHub - Wikipedia, la enciclopedia libre". Wikipedia, la enciclopedia libre. <https://es.wikipedia.org/wiki/GitHub> (accedido el 3 de junio de 2022).

[9] "Contenedores de Docker | ¿Qué es Docker? | AWS". Amazon Web Services, Inc. <https://aws.amazon.com/es/docker/> (accedido el 3 de junio de 2022).

[10] "Overview of Docker Compose". Docker Documentation. <https://docs.docker.com/compose/> (accedido el 3 de junio de 2022).

[11] "WildFly". WildFly. <https://www.wildfly.org/> (accedido el 3 de junio de 2022).

[12] "Desarrollo iterativo e incremental". Se hace camino al andar... <https://blog.jmbeas.es/2019/09/22/desarrollo-iterativo-e-incremental/> (accedido el 4 de junio de 2022).

[13] E. Larson. "Uses Cases - What every project manager should know". Project Management Institute. <https://www.pmi.org/learning/library/use-cases-project-manager-know-8262> (accedido el 5 de junio de 2022).

[14] "Java". Home de DesarrolloWeb.com. <https://desarrolloweb.com/home/java> (accedido el 7 de junio de 2022).

[15] "Introduction to the Java EE Architecture". Pawlan Communications - Business Services. <http://pawlan.com/monica/articles/j2eeearch/> (accedido el 7 de junio de 2022).

[16] "commandButton (VDL Documentation Generator - Generated Documentation)".

PrimeFaces – Ultimate UI

Framework. <https://www.primefaces.org/docs/vdl/3.4/primefaces-p/commandButton.html> (accedido el 11 de junio de 2022).

[17] *Base de datos rentable y de alto rendimiento*. (s. f.). Oracle | Cloud Applications and Cloud Platform. <https://www.oracle.com/es/database/>

Apéndice A

Guía de Instalación

En el apéndice siguiente se describirán los pasos a seguir para realizar una instalación de la aplicación web y gracias a esta poder utilizar la aplicación en cualquier dispositivo. Para el ejemplo de la instalación de la aplicación se ha escogido un Windows 10 Pro de 64 bits, sin embargo el desarrollo y el despliegue de la aplicación se ha realizado en un Ubuntu 18.04.4, por lo que la instalación se puede llevar a cabo en cualquier computadora, siempre y cuando se sigan los pasos siguientes y se ajusten al sistema operativo correspondiente.

A.1. Código necesario

Para poder llevar a cabo la instalación, se necesita del `docker-compose.yml`, del `dockerfile`, del `.ear` de la aplicación y de los archivos necesarios para la configuración del servidor de aplicaciones. Para poder descargar este código se puede acceder al repositorio GitHub siguiente o descargarlo directamente de la entrega del TFG:

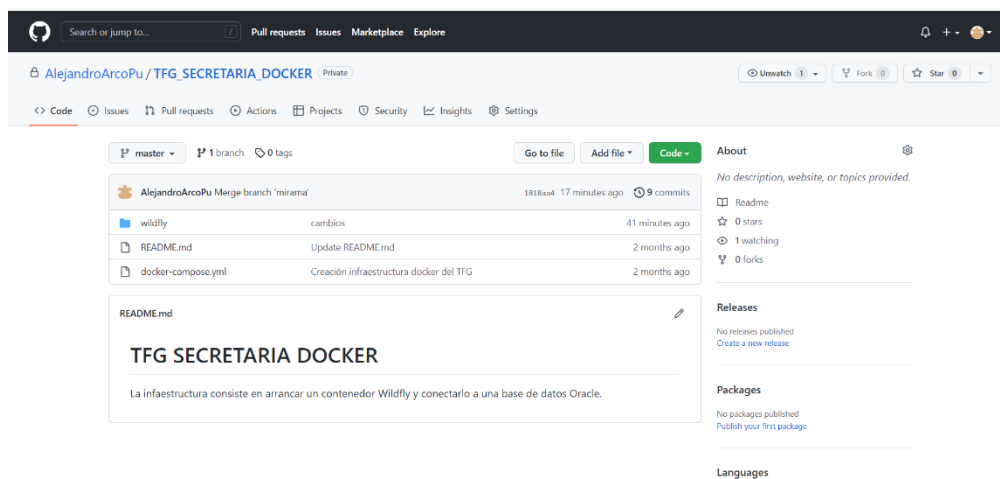


Figura A.1.1: Repositorio GitHub

El siguiente paso es pulsar en el botón verde de *Code* y se abrirá una ventana con varios apartados, debemos de hacer click sobre *Download ZIP* para obtener todo el código en un .zip. Al descargarlo se obtendrá algo parecido a esto:



Figura A.2.2: Archivo .zip

Este fichero se deberá extraer en un directorio a nuestra elección y a partir de él, se podrá realizar el despliegue, pero antes se debe seguir con la instalación de los herramientas necesarias.

A.2. Instalación docker y despliegue

Para poder arrancar y construir la aplicación vamos a necesitar Docker Desktop, el cual ya tiene instalado Docker Compose, que es lo que se ha utilizado para el despliegue de nuestra aplicación. Para ello nos dirigimos al enlace de [Docker Desktop](#) y hacemos click sobre Windows:

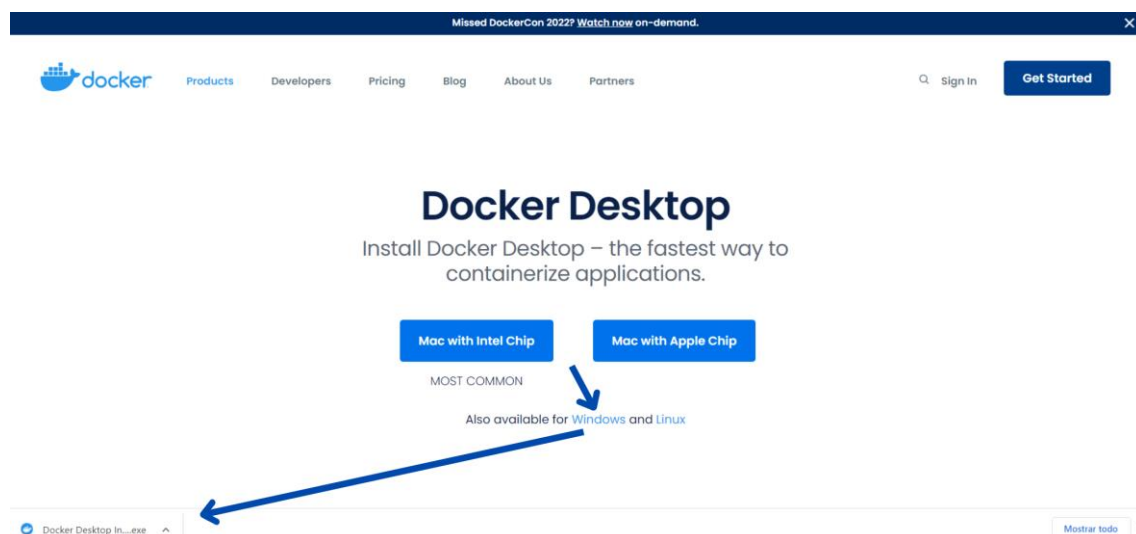


Figura A.2.1: Página de descarga Docker Desktop

Una vez se haya hecho click se nos descargará un .exe. Para seguir con la instalación es necesario ejecutar ese .exe, el cual nos abrirá una ventana de instalación, en la que como mínimo habrá que seleccionar el uso de WSL:

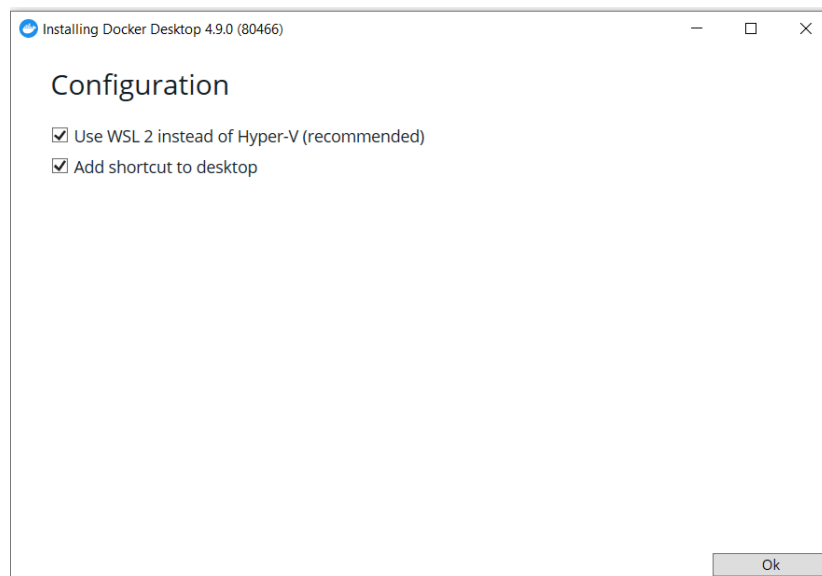


Figura A.2.2: Opciones configuración Docker Desktop

Tras pulsar en *Ok* comenzará la instalación de Docker y de todos los archivos de los que necesita Docker para poder funcionar. Una vez se haya completado la instalación anterior será necesario reiniciar el ordenador pulsando en el botón *Close and restart*:

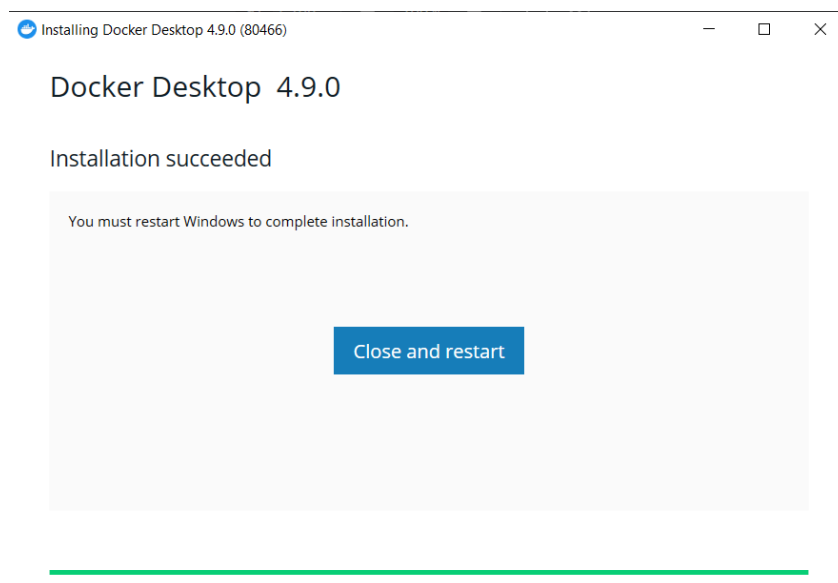


Figura A.2.3: Close and restart Docker Desktop

Después del reinicio habrá que aceptar los términos y condiciones de Docker y ya se nos abriría Docker Desktop:

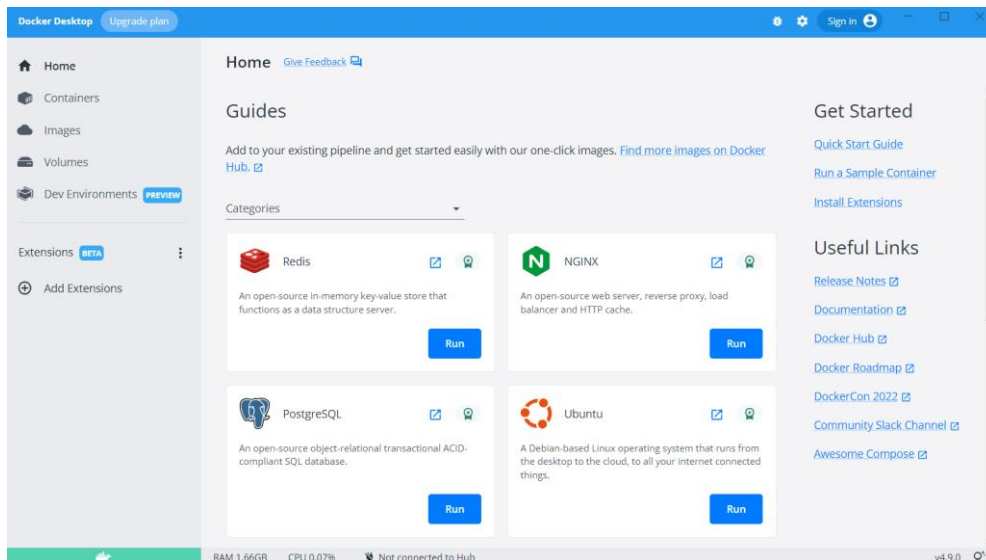


Figura A.2.4: Pantalla inicial Docker Desktop

Para poder desplazarnos hasta la carpeta donde se ha extraído el código del punto A.1, será necesario abrir un terminal de Windows e ir en concreto al directorio donde se encuentre el `docker-compose.yml`:

```
Windows PowerShell
PS C:\Users\aleja\OneDrive\Escritorio\TFG_SECRETARIA_DOCKER-master> ls

Directorio: C:\Users\aleja\OneDrive\Escritorio\TFG_SECRETARIA_DOCKER-master

Mode                LastWriteTime         Length Name
----                -
da---1           15/06/2022   21:19             4 wildfly
-a---1           15/06/2022   21:19            124 docker-compose.yml
-a---1           15/06/2022   21:19            127 README.md
```

Figura A.2.5: Ubicación `docker-compose.yml`

Después de situarnos en el directorio del `docker-compose.yml`, ejecutamos `docker-compose up --build`, que construirá y arrancará la aplicación al completo:

```
Windows PowerShell
PS C:\Users\aleja\OneDrive\Escritorio\TFG_SECRETARIA_DOCKER-master> docker-compose up --build
[+] Building 51.2s (13/13) FINISHED
=> [internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 841B                                              0.0s
=> [internal] load .dockerignore                                                 0.0s
=> => transferring context: 2B                                                  0.0s
=> [internal] load metadata for docker.io/jboss/wildfly:18.0.1.Final            2.1s
=> [internal] load build context                                                0.3s
=> => transferring context: 28.94MB                                             0.3s
=> [1/8] FROM docker.io/jboss/wildfly:18.0.1.Final@sha256:382f9c07af3a86c4e00f00af813cc012a2409731088a191b32c59 46.1s
=> => resolve docker.io/jboss/wildfly:18.0.1.Final@sha256:382f9c07af3a86c4e00f00af813cc012a2409731088a191b32c59d 0.0s
=> => sha256:b5ffff9dbcd0091fed19b21ccb063371e272e928a2ba9a327e8d7f81eef085 36.73MB / 36.73MB 19.1s
=> => sha256:36845d6d0218670411397da6424db66dc914199a19fb7d58c6dc9b370f0dda4 1.90kB / 1.90kB 0.2s
```

Figura A.2.6: Ejecución `docker-compos up --build`

Cuando se haya acabado de ejecutar el docker-compose, ya se tendrá disponible la aplicación web en la URL: <http://localhost:8080/Secretaria-WAR/login.xhtml>. En la siguiente imagen se confirma:

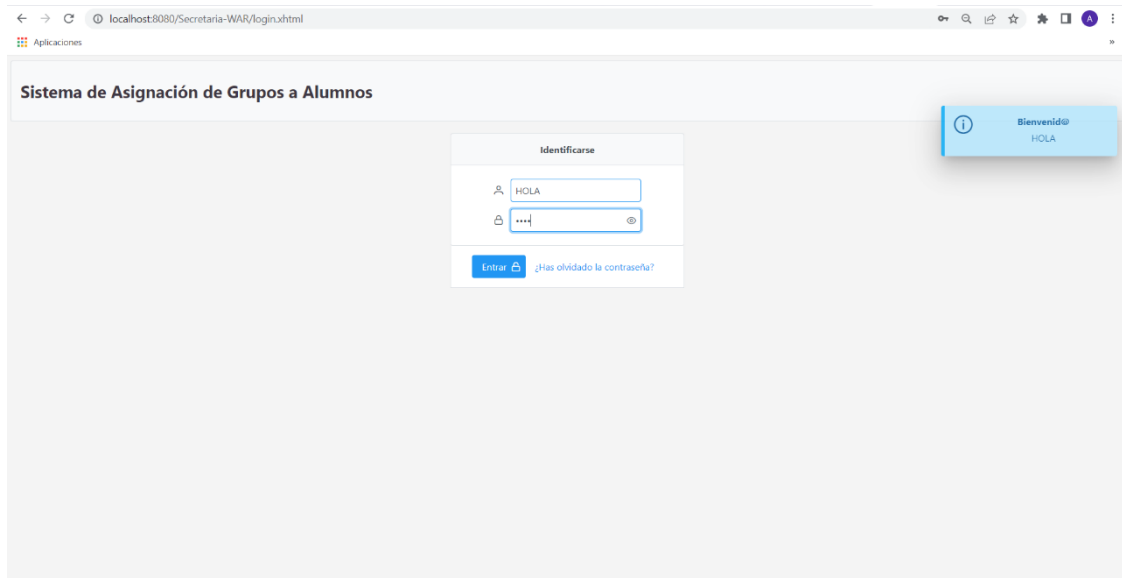


Figura A.2.7: Página inicial de la aplicación

Apéndice B

Manual de inclusión de nuevos procesos

Una de las posibilidades de la aplicación en un futuro próximo es que necesite de un nuevo proceso para llevar a cabo la asignación de grupos a alumnos, para que el mantenimiento en un futuro sea una tarea sencilla, se ha pensado en explicar cómo sería esta inclusión de un nuevo proceso en la aplicación.

B.1. Backing bean

El primer paso es incluir en la tabla PROCESOS de la base de datos, el nombre del proceso que se quiere incluir, ya que aquí se almacenará la información de la ejecución del proceso. El segundo paso para poder incluir un proceso nuevo en la aplicación, será crear un backing bean dentro de la carpeta del proyecto Secretaria-WAR/src/main/java/backingbeans/ejecutarprocesos:

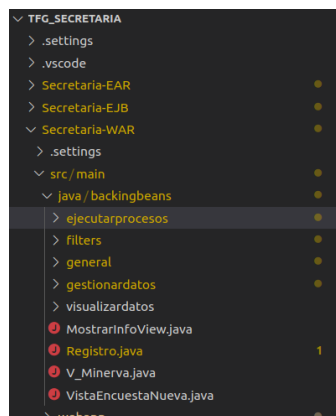


Figura B.1.1: Directorio ejecutarprocesos

Este backing bean será como los explicados en el capítulo de desarrollo, con una etiqueta *@Named*, para poder nombrarlos en la página JSF y con una anotación de ámbito que en este caso será *@RequestScoped*, ya que se busca que solo exista el bean durante una petición. Después de esto será necesario incluir en el backing bean los atributos que se van a utilizar en la página JSF, como se sabe cada uno de los procesos tiene un registro de la fecha de ejecución y el nombre del último usuario que lo ejecutó, por ello van a ser imprescindibles un atributo nombre y una atributo fecha del proceso. Para obtener la fecha y el nombre de la ejecución, nos servimos de EntityManager, que realizará una query sobre la tabla PROCESOS de la base de datos con un WHERE que coge el nombre del proceso, además para asegurar que después de la inyección de dependencias se cargan los resultados de la fecha y del usuario utilizamos el *@PostConstruct*:

The image shows a code snippet for a Java backing bean. It includes annotations like `@Named` and `@RequestScoped`, and attributes like `usuarioB` and `fechaB`. A `@PostConstruct` method is also present, which uses `EntityManager` to fetch data from the database. Three callouts highlight specific parts: 'Anotaciones' points to the `@Named` and `@RequestScoped` annotations; 'Atributos usuario y fecha' points to the `usuarioB` and `fechaB` attributes; and 'Se obtienen los valores de los atributos, indicando el nombre del proceso' points to the `selectProcesos` method calls in the `@PostConstruct` method.

```

16 import es.uma.informatica.sii.ejb.pract...
17
18 @Named(value = "borrarAsig")
19 @RequestScoped
20 public class BorrarAsignacion {
21
22     private String tipoBorrar;
23
24     private List<String> tiposBorrar;
25
26     private String usuarioB;
27     private String fechaB;
28
29
30     @Inject
31     private GestionProcesos gestionProcesos;
32
33     @PersistenceContext(name = "Secretaria")
34     private EntityManager em;
35
36     @PostConstruct
37     public void init() {
38         // necesario para incluir en el select de tipo borrar las cosas
39         tiposBorrar = new ArrayList<>();
40         tiposBorrar.add("PARCIAL");
41         tiposBorrar.add("TOTAL");
42
43         // Proceso BorrarAsignacion
44         usuarioB = gestionProcesos.selectProcesos(nombre: "BORRAR_ASSIGNACION").getUsuario();
45         fechaB = gestionProcesos.selectProcesos(nombre: "BORRAR_ASSIGNACION").getFechaEjecucion();
46     }

```

Figura B.1.2: Backing Bean proceso

El método que se va a encargar de llamar al proceso únicamente debe tener:

- La llamada al proceso nuevo, que se realizará a través de *createStoredProcedureQuery*.

- La actualización del proceso, para que cada vez que se ejecute se actualicen los valores del usuario y la fecha y así, poder llevar el control adecuadamente.

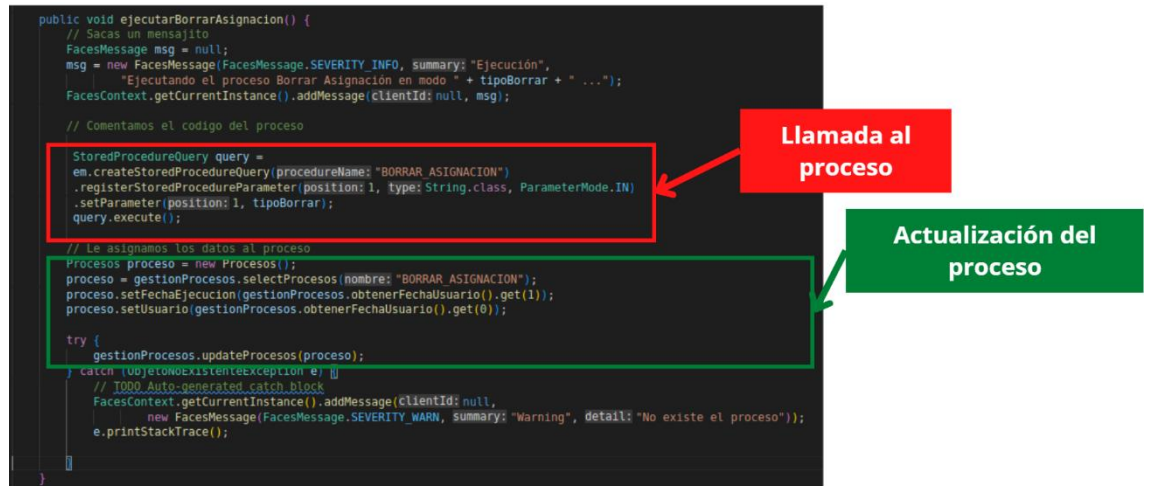


Figura B.1.3: Backing bean método ejecutar

Este método, después se enlazará a la página JSF.

B.2. Página JSF

En la página JSF, únicamente será necesario un incluir un botón que llame al método del nuevo proceso y los `outputText` para mostrar los datos del proceso (usuario y fecha):



Figura B.1.4: JSF proceso

Apéndice C

Manual de usuario

En este Apéndice se agruparan cada uno de los pasos y funcionalidades necesarias para realizar un correcto uso de la aplicación web, teniendo en cuenta también roles de la aplicación.

C.1. Iniciar Sesión

Cuando se entra por primera vez a la aplicación se puede observar un panel en el cual se deberán introducir los credenciales para poder acceder e interactuar con esta:

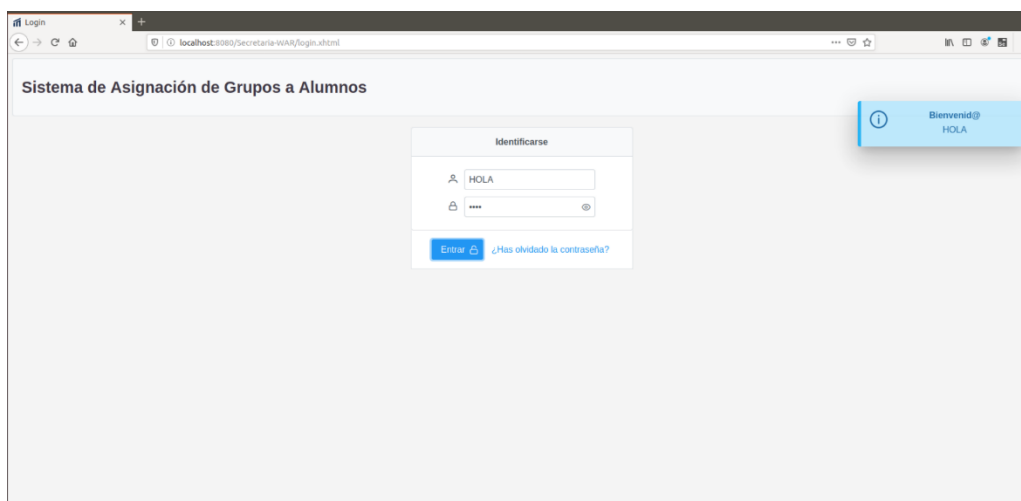


Figura C.1.1: Página de Iniciar Sesión

C.2. Recuperar contraseña

Si se olvida la contraseña se puede hacer click en *¿Has olvidado la contraseña?* , donde se enviará la nueva contraseña al correo introducido, siempre y cuando el correo este registrado en la aplicación:

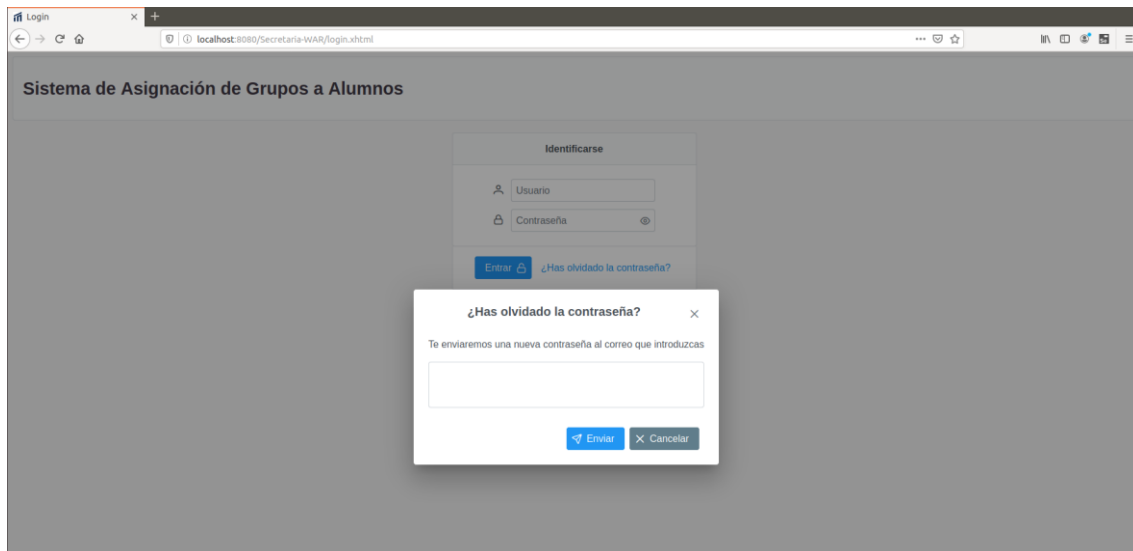


Figura C.2.1: Página ¿Has olvidado la contraseña?

Una vez introducidos los credenciales correctos se podrá acceder a la página principal de la aplicación. Está página consta de:

- Un menú lateral con las distintas secciones de la aplicación.
- Un panel central, donde se explica en que consiste cada sección de la aplicación.
- Un panel superior, con el nombre de usuario que permite cambiar la contraseña del usuario y con el botón de Salir que cerrará la sesión
- Un panel intermedio con la página en la que está situado el usuario para poder navegar por la aplicación rápidamente.

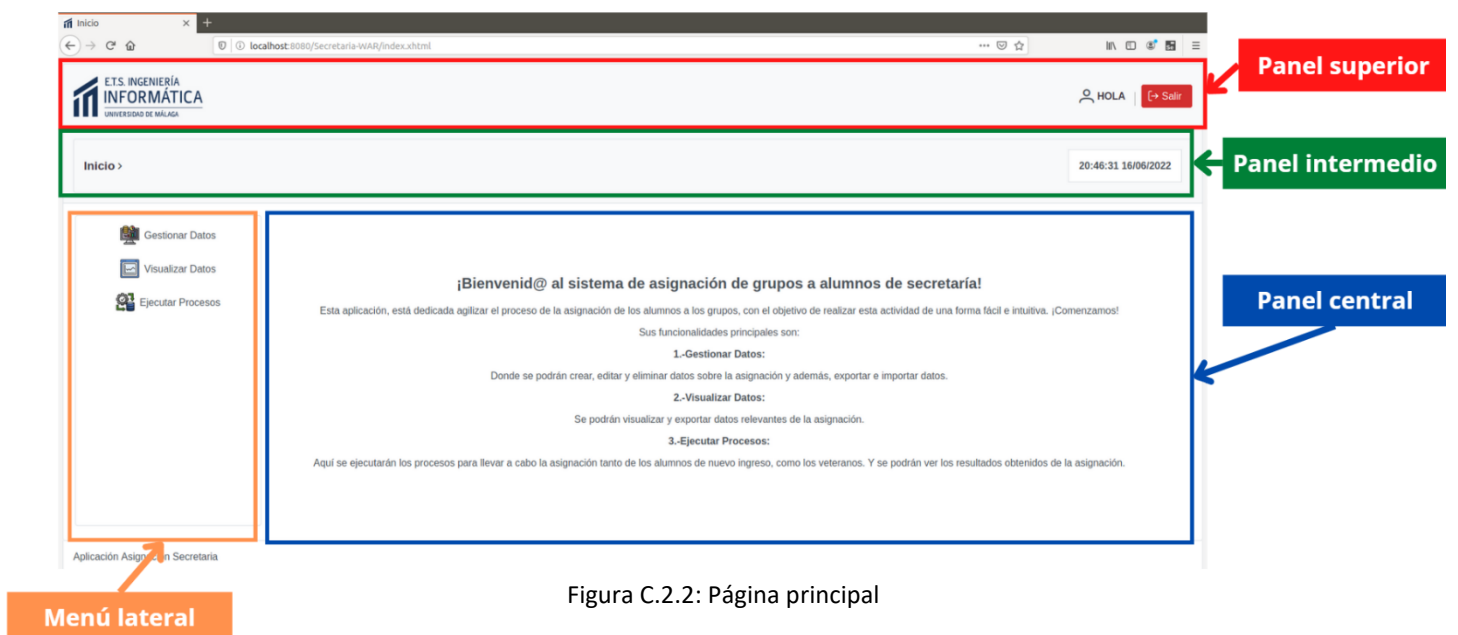


Figura C.2.2: Página principal

C.3. Cambiar contraseña

Si se desea cambiar la contraseña del usuario, haciendo click en el nombre del usuario de la página principal, se abrirá un pequeño panel, donde como requisito se tendrá que saber la contraseña actual:

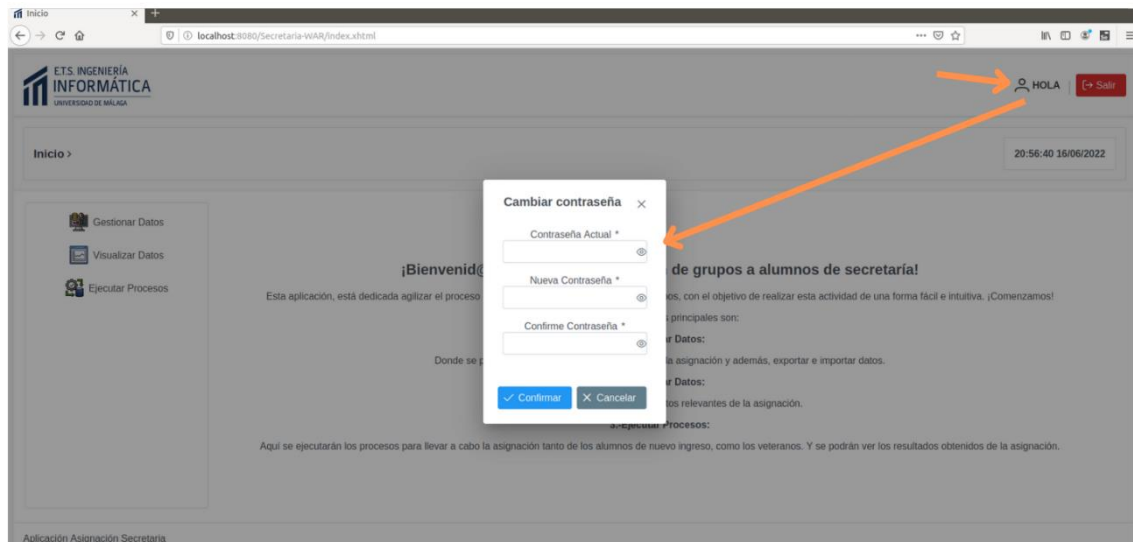


Figura C.3.1: Cambiar contraseña

C.4. Gestionar Datos

Si se hace click en el primer botón del menú Gestionar Datos, se abrirá una ventana con los posibles datos a gestionar:

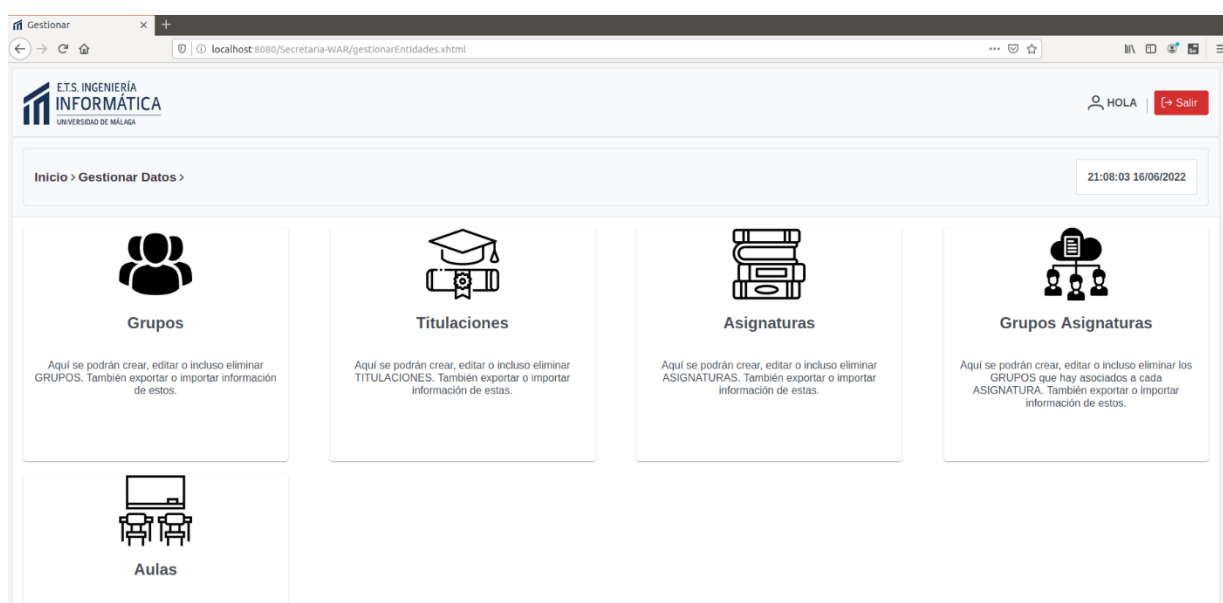


Figura C.4.1: Página Gestionar Datos

En cada uno de los posibles datos a gestionar se pueden realizar las mismas funciones, por tanto se va a explicar el primer tipo de datos, Grupos. Al hacer click en el botón Grupos, se abrirá una lista con todos los grupos y se podrán desempeñar las siguientes funciones:

- Crear datos
- Editar datos
- Eliminar datos
- Importar datos
- Exportar datos
- Filtrar datos

The screenshot shows the 'Lista grupos' web application interface. The browser address bar indicates the URL is 'localhost:8080/Secretaria/Word/ListaGrupos.html'. The page title is 'Lista grupos'. The main content area has a breadcrumb trail: 'Inicio > Gestionar Datos > Grupos >'. The date '21-11-01 16/06/2022' is displayed in the top right corner. Below the breadcrumb, there is a pagination bar showing '1-5 of 51 records' and a list of page numbers from 1 to 10. The main data table has columns: 'Id T1', 'Curso T1', 'Letra T1', 'Turno T1', 'Titulación T1', 'Inglés T1', 'Visible T1', 'Asignar T1', 'Plazas T1', 'Se imparte con T1', 'Plazas nuevo ingreso T1', 'Sustituye inglés T1', 'Aula c1 T1', and 'Aula c2 T1'. The table contains five rows of data. To the right of the table, there is a vertical list of green circular icons with checkmarks, labeled 'Lista de datos'. Above the table, there is a filter bar with various input fields, labeled 'Filtrar'. Below the table, there is a button labeled 'Importar datos' with a sub-label '+ Escoger fichero' and a 'Subir' button. In the top left corner, there are two buttons: '+ Nuevo' and 'Eliminar', labeled 'Crear y eliminar'. In the top right corner, there is a button labeled 'Exportar' with a sub-label '↓ Exportar'.

Figura C.4.2: Página Gestionar Grupos

Para poder crear datos se tendrá que hacer click en el botón *Nuevo* de la esquina superior izquierda y se abrirá un panel, donde se tendrán que introducir como mínimo los datos con asteriscos (*), ya que son los obligatorios para crear un dato nuevo. Tras rellenar los datos obligatorias si se hace click en Aceptar aparecerá el nuevo dato en la lista de datos:

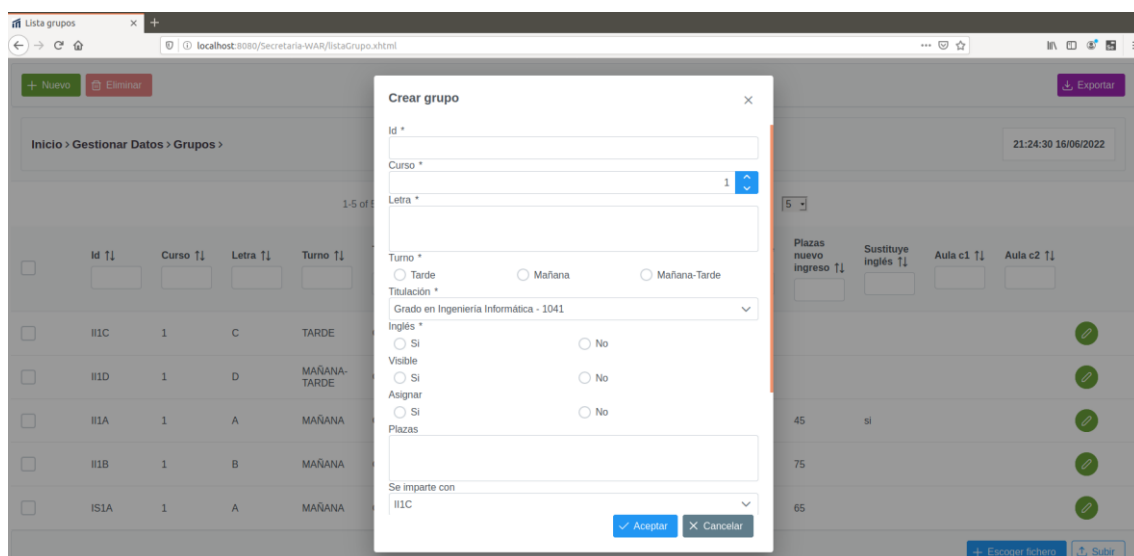


Figura C.4.3: Crear grupo

Para poder eliminar datos se requerirá la selección de las filas a eliminar y al instante se activará el botón Eliminar:

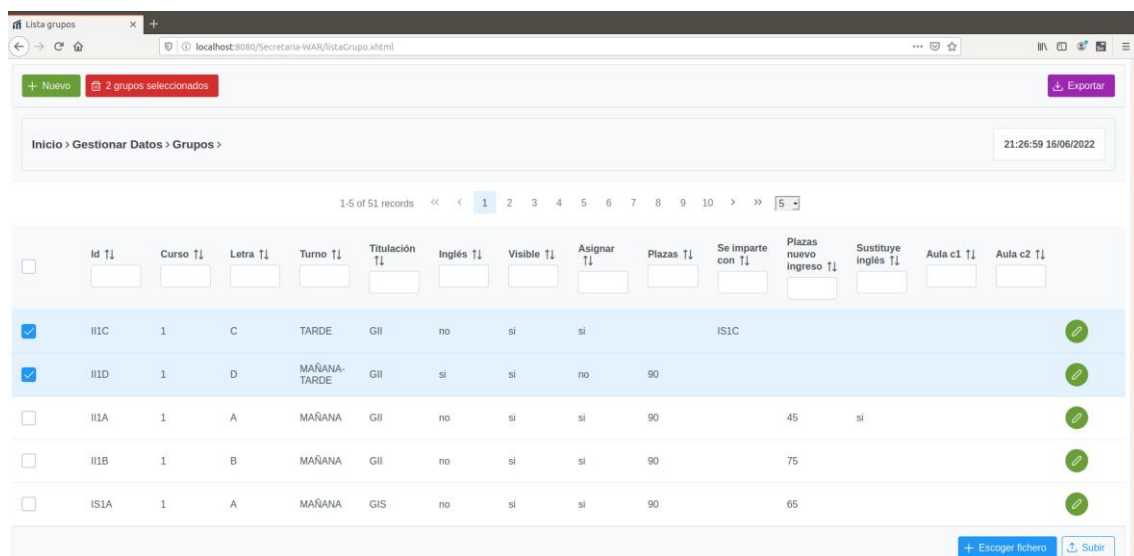


Figura C.4.5: Eliminar grupos

En cuanto a la edición de datos se podrá editar una fila haciendo click en el botón verde en forma de lápiz. Al pulsarlo se abrirá un panel similar al de crear, pero con los datos de la fila, tras pulsar en Aceptar cualquier modificación será aplicada y se podrá observar en la lista de datos:

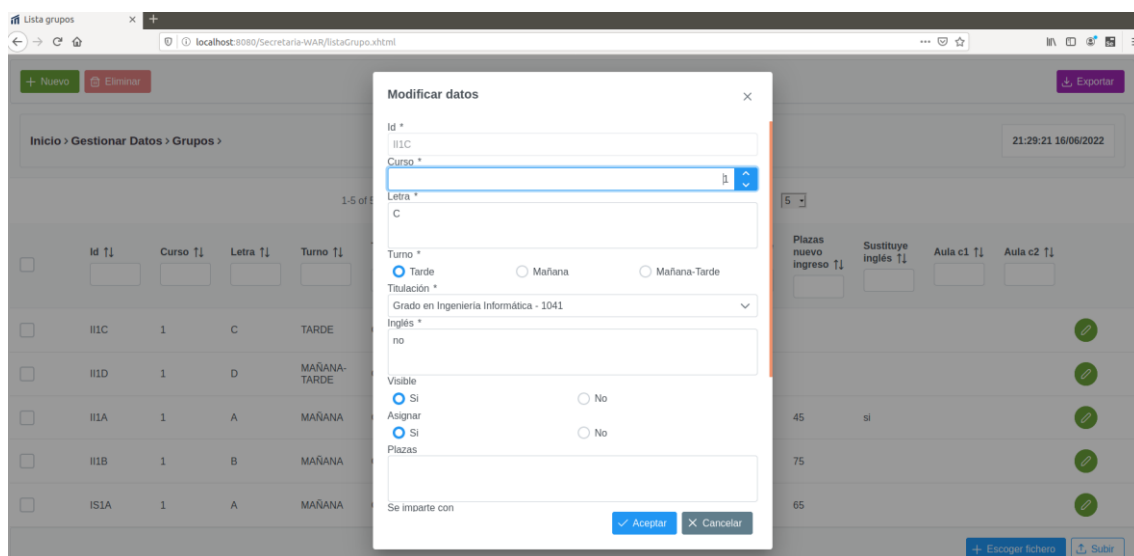


Figura C.4.6: Editar datos

Si se desea exportar la lista de datos se tendrá que pulsar en el botón de exportar y se obtendrá la lista de datos en formato .xlsx:

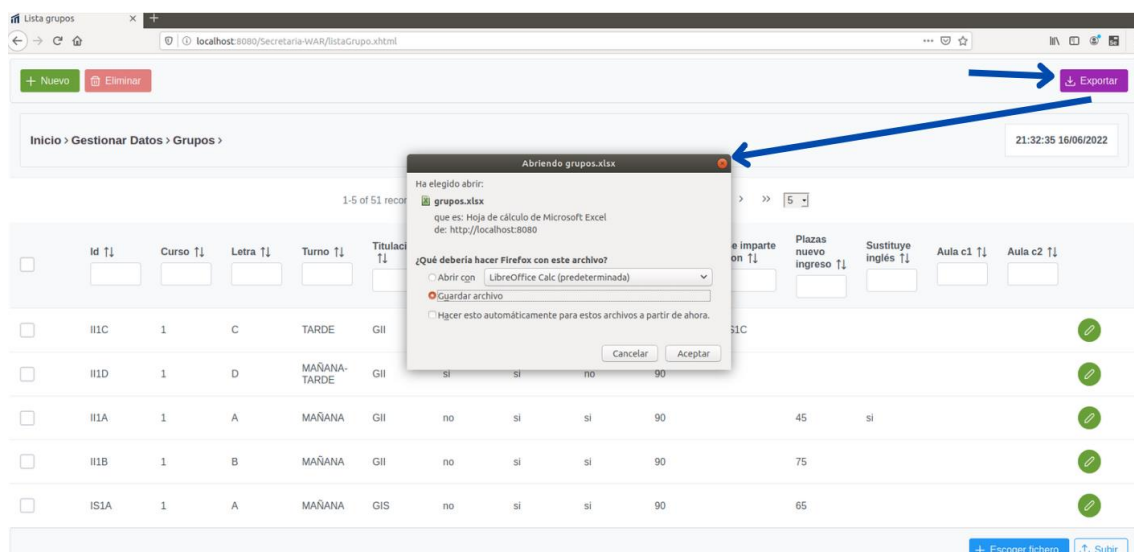


Figura C.4.7: Exportar datos

Para poder importar datos, se debe escoger primeramente el fichero pulsando en *Escoger Fichero* y más tarde pulsar en *Subir*, es importante que el fichero a subir disponga de los mismo atributos que la lista de datos:

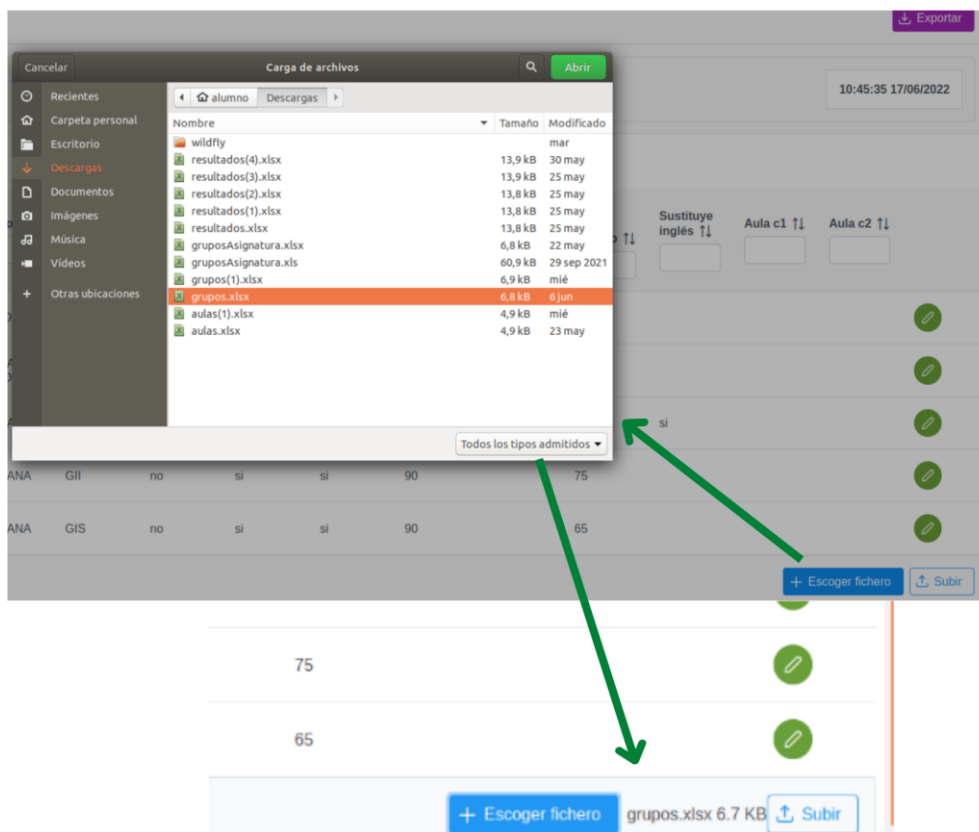


Figura C.4.8: Importar datos

Para acabar con las funcionalidades de Gestionar Datos se pueden filtrar los datos simplemente escribiendo en cada uno de los cuadros de texto de cada fila:

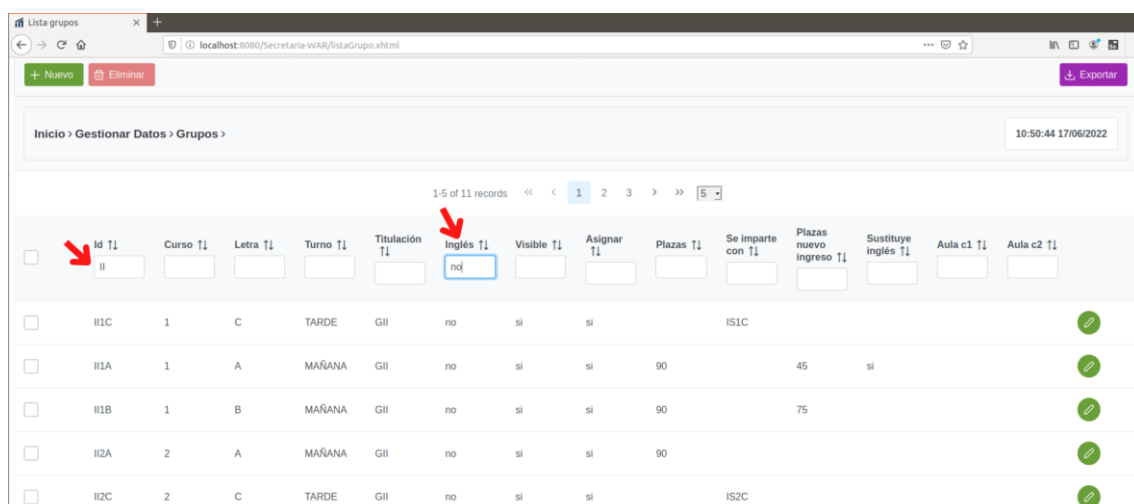


Figura C.4.9: Filtrar datos

C.5. Visualizar Datos

La siguiente sección del menú lateral de la página principal se corresponde con la de Visualizar Datos, que incluye prácticamente las mismas funcionalidades que la sección de Gestionar Datos. Al hacer click en Visualizar Datos se abrirá una nueva página con todos los datos a visualizar:

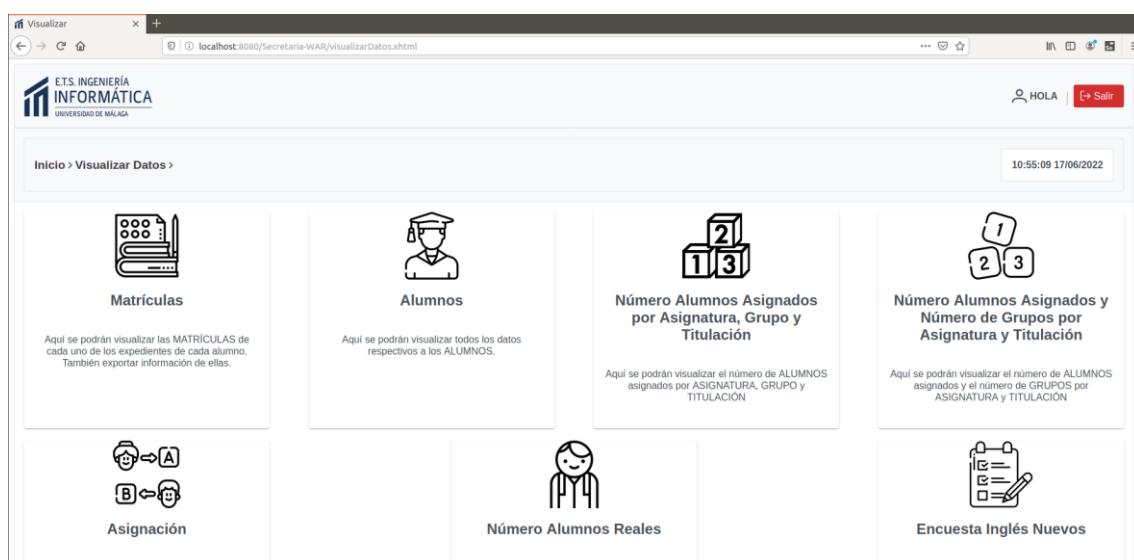


Figura C.5.1: Visualizar Datos

Al pulsar en cualquiera de los botones para visualizar datos se desplegará una tabla idéntica a la sección de Gestionar, con la única diferencia de que esta únicamente se podrá Exportar, Filtrar y Visualizar cada fila en detalle:

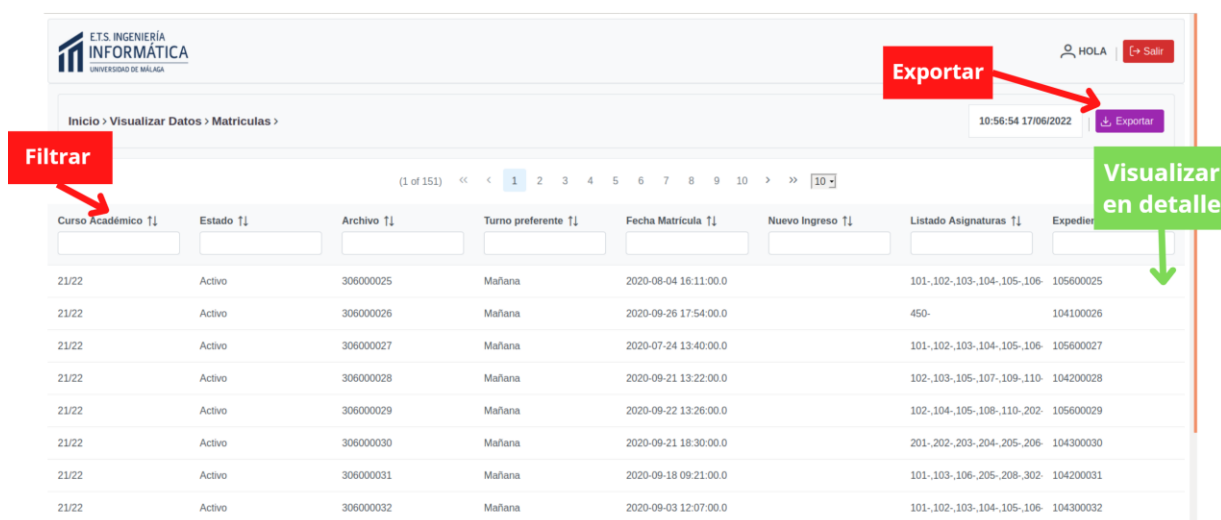


Figura C.5.2: Visualizar Matriculas

Si se hace click en una fila cualquiera los datos de esta se podrán ver en detalle:

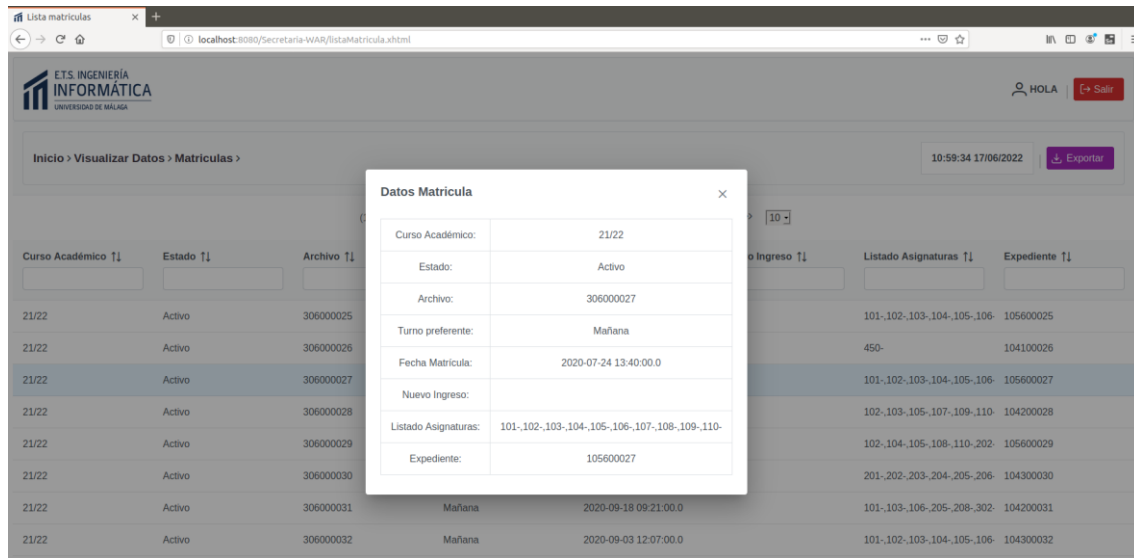


Figura C.5.3: Visualizar fila en detalle

C.6. Ejecutar procesos

La última sección se corresponde con la de Ejecutar Procesos, en la que se podrá llevar a cabo la asignación siguiendo cada uno de los pasos correspondientes, se podrán visualizar datos de interés de la ejecución y se podrán obtener los resultados de la asignación, además se incorporarán ayudas para saber la función de cada proceso:



Figura C.6.1: Página Ejecutar Procesos

Para ejecutar un proceso simplemente se hace click en el botón *Ejecutar* de cada uno y este comenzaría a ejecutarse:

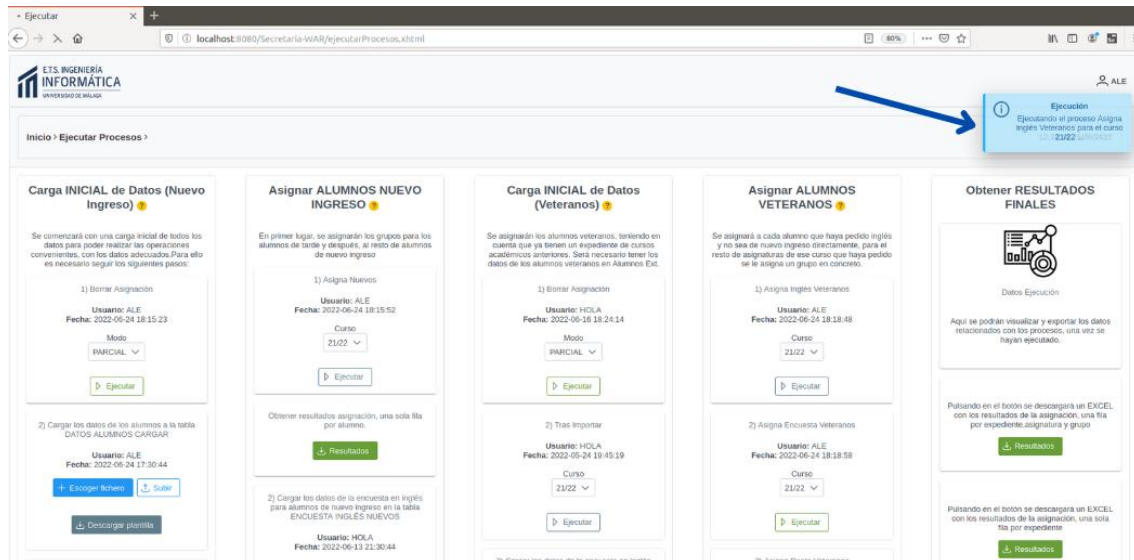


Figura C.6.2: Ejecutar un determinado proceso

Para importar datos necesarios para la asignación se procede de la misma forma que los importar de la sección de Gestionar. Si se desean visualizar datos de interés de la asignación simplemente se pulsa en Datos Ejecución y se abrirá una página con los posibles datos a visualizar:

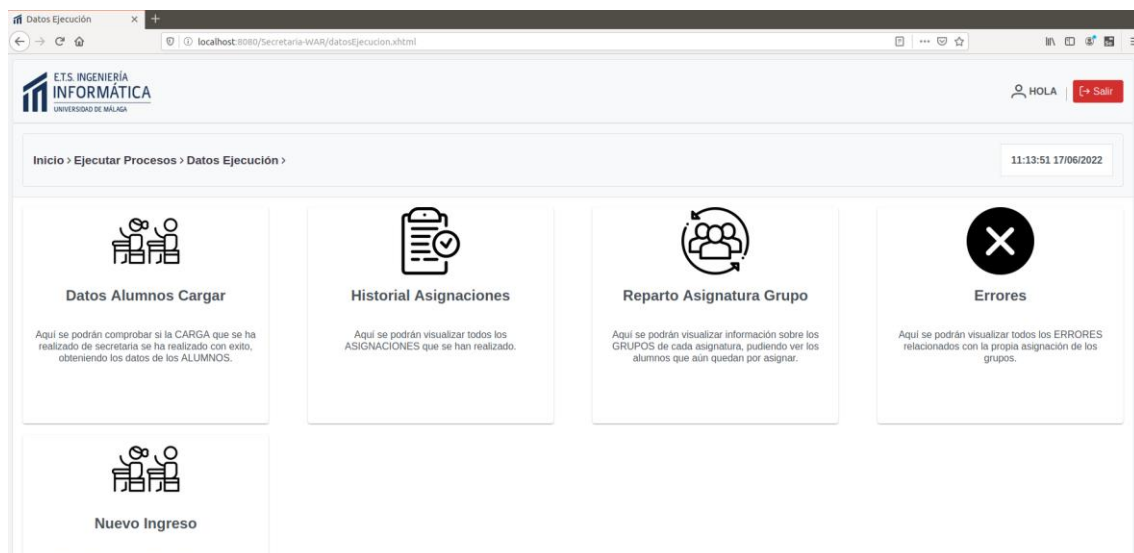


Figura C.6.3: Datos Ejecución

Lo último que se puede realizar en esta sección es obtener tanto los resultados de la asignación, como la descarga de las plantillas. Para proceder a alguna de las dos funcionalidades, simplemente se hace click en *Resultados* o en *Descargar Plantilla* y comenzará la descarga de un archivo .x/sx:

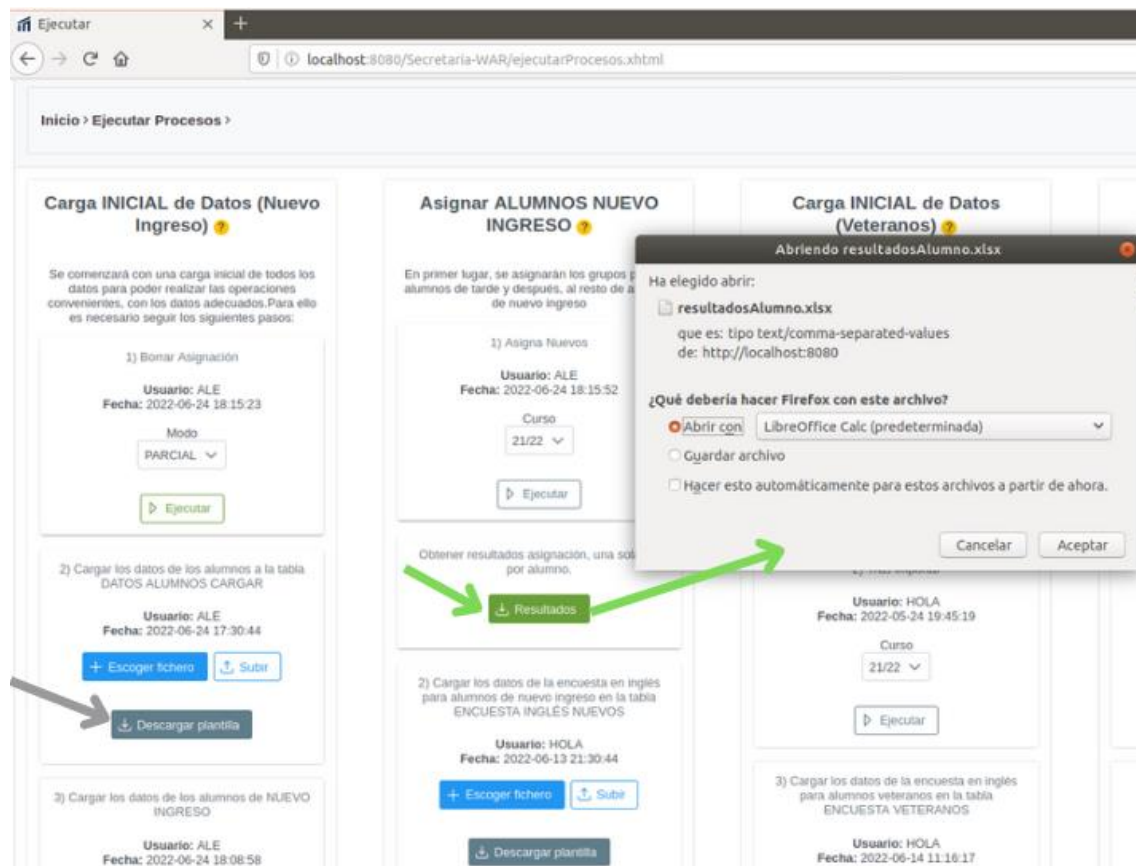


Figura C.6.4: Obtener resultados asignación

C.7. Cerrar Sesión

Si se desea salir de la aplicación y evitar posibles accesos indeseados, se tendrá que pulsar en alguno de los botones de *Salir* que se pueden observar en varias páginas de la aplicación:

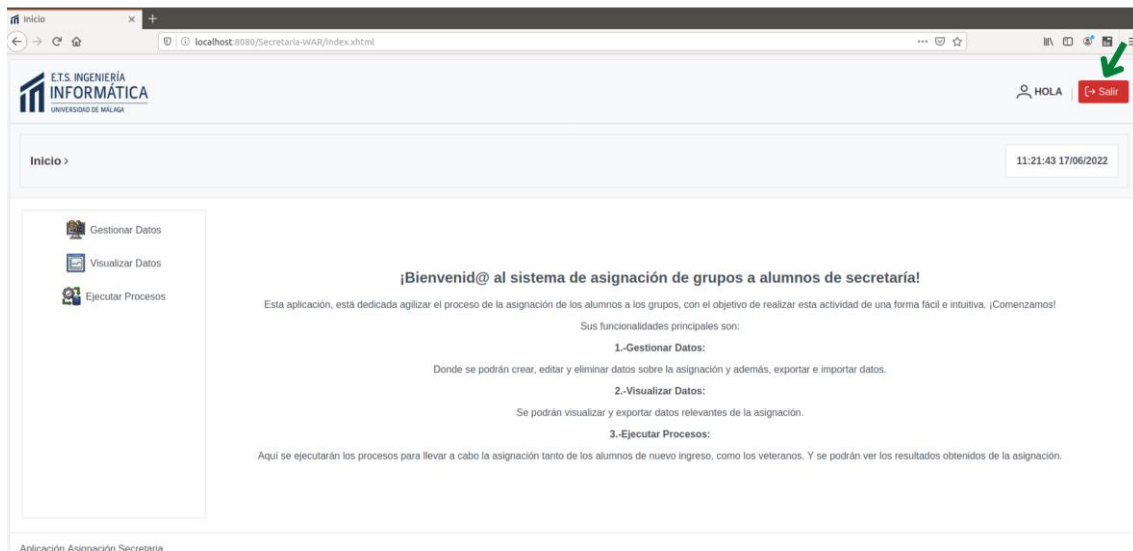


Figura C.7.1: Salir de la aplicación

C.8. Rol ADMINISTRADOR

El rol administrador difiere del rol usuario en que podrá realizar una gestión activa de todos los usuarios de la aplicación. El usuario administrador será el encargado de dar el acceso a la aplicación al resto de usuarios. Este tipo de usuario por lo tanto, dispondrá de una sección más en la página principal:



Figura C.8.1: Menú con Gestionar Usuarios

Dentro de esta sección se podrán apreciar todos los usuarios registrados en la aplicación y estos podrán ser eliminados o editados al igual que se explicó en el punto de Gestionar Datos, también se podrán crear nuevos usuarios:

☐	Usuario	Nombre	Apellido1	Apellido2	Email	Rol	
☐	HOLA	HOLA1	HOLA	HOLA	alexniike360@gmail.com	USER	
☐	ALE	ALE	ALE	ALE	alejandroarpu@gmail.com	ADMIN	
☐	PRU	PRU	PRU	PRU	pru@gmail.com	USER	
☐	ADIOS	ADIOS	ADIOS	ADIOS	hola1@gmail.com	USER	

Figura C.8.2: Gestionar Usuarios

