UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADUADO EN INGENIERÍA DE SOFTWARE

Herramienta web para la extracción y procesamiento de información a partir de ficheros CSV

Parte 2

Web Tool for data extraction and processing from CSV files

Part 2

Realizado por
Javier Vázquez Moreno

Tutorizado por
Eduardo Guzmán De los Riscos

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, JUNIO DE 2019

Fecha defensa:

Fdo. El/la Secretario/a del Tribunal

Resumen

En este trabajo de fin de grado se describe una aplicación web que extrae datos y los procesa a partir de archivos CSV.

Normalmente, en la administración, muchas de las tareas están informatizadas. El hecho de que las tareas administrativas estén informatizadas además de por seguridad, para que no se pierdan datos, es por velocidad. Hasta ahora se encuentra con que el personal de administración no posee herramientas sencillas para combinar archivos de tipo csv o xlsx. Este TFG precisamente intenta dar solución a este problema a través de una aplicación web.

La aplicación funciona como una plataforma en la que el usuario puede subir archivos de tipo csv y xlsx con el fin de combinar datos de una forma automática. El resultado de estas operaciones podrá ser descargado en el dispositivo del usuario o guardado en la base de datos. Además la aplicación ofrece una interfaz web que proporciona al usuario diferentes herramientas a través de las cuales podrá visualizar los datos, podrá buscar conceptos e incluso ver representaciones gráficas asociados a los datos.

Si bien es cierto que esta aplicación esta enfocada para el sector administrativo, para facilitar las tareas y ahorrar tiempo, es una aplicación que puede ser muy útil para cualquier usuario que pretenda combinar datos de diferentes Excel o archivos de tipo csv.

Palabras clave: CSV, excel, procesamiento de datos, Spring, MongoDB.

Abstract

In this final project, a web application that allows data extraction and processing from CSV files is described.

Usually, in bureaucracy, many of the tasks are computerized. The main reason for administrative tasks to be computerized (besides security, so that data is not lost) is the speed. So far we find that the administration staff does not have an intuitive and simple tool to combine csv or xlsx files. Precisely, this project tries to provide a solution to this problem through a web application.

The developed application works as a platform in which the user can upload csv or xlsx files in order to combine data in an automatic way. The result of this operations may be downloaded to the user's device or stored in a database. In addition, the application offers a web interface that provides the user with different tools through which he/she can view data results, search concepts and even see graphical representations associated with the data.

Although the main usage the application is designed for is the administrative sector (in order to facilitate tasks and save time), its features can be generalized and be applied to any user willing to combine data from different format files.

Keywords: CSV, excel, data processing, Spring, MongoDB.

Índice

1

Introducción

1.1 Motivación

En el ámbito administrativo, el usuario se encuentra de manera frecuente ante archivos muy extensos con una gran cantidad de datos que se ven obligados a procesar de manera rudimentaria. En particular, si nos encontramos ante el caso en el que el usuario debe juntar datos de dos o más archivos en uno, éste deberá de hacerlo a mano copiando y pegando datos lo que puede inducir a error.

A raíz de este tipo de problemas surge la idea de desarrollar una aplicación que automatice todo el proceso. Con tan sólo con tres pasos, los cuales sirven para configurar el formato del archivo, podemos obtener el archivo resultado esperado, descargarlo, guardarlo en el servidor e incluso mostrar el análisis estadístico a partir de datos obtenidos del archivo.

1.2 Objetivos

El objetivo principal de este proyecto es desarrollar una aplicación web que permita combinar, mostrar y analizar archivos con una gran cantidad de datos. El fin que se persigue es que los usuarios, en especial aquellos del ámbito administrativo, puedan llevar a cabo estas funcionalidades de una forma fácil e

intuitiva. La aplicación web permitirá, además, gracias a la gestión de usuarios almacenar archivos en un servidor de forma privada.

El proyecto se desarrollará en grupo con un trabajo equilibrado por parte de los miembros del grupo. Haciendo uso de la metodología ágil Scrum dividiremos tareas, realizando algunas de ellas por los dos miembros a la vez y otras de formas más autónoma, en el punto 3.2 podemos ver las tareas con más detalle.

Con esto conseguimos proporcionar una herramienta muy potente a los usuarios a través de una página web, en la cual simplemente subiendo los archivos deseados pueden combinarlos por columnas, unirlos o obtener gráficas de los datos contenidos en los mismos.

1.3 Estructura de la memoria

La estructura de la memoria se ha organizado, principalmente, siguiendo la metodología usada para el desarrollo del proyecto, la cual ha tenido las etapas que encontramos en cualquier desarrollo software. Así el lector podrá observar la evolución que ha tenido el proyecto de manera iterativa.

1.3.1 Introducción

En este primer capítulo se exponen los motivos por los que surge la necesidad de desarrollar esta plataforma, los objetivos del proyecto, y un resumen del contenido de esta memoria.

1.3.2 Tecnologías utilizadas

En esta sección se tratará el motivo por el que hemos elegido utilizar esas tecnologías.

El lector podrá observar las tecnologías utilizadas dividas por categorías. Entre las explicadas encontramos frameworks, base de datos, herramientas de desarrollo web y librerías java utilizadas.

1.3.3 Metodología y planificación

En esta sección se trata la gestión del proyecto, desde la metodología elegida para el desarrollo del mismo hasta la planificación que hemos realizado para llevarlo a cabo.

1.3.4 Análisis de especificación.

Esta sección es de gran importancia debido a que en ella desarrollamos la primera fase del proyecto, la captura de requisitos.

Describimos con detalle los requisitos tanto funcionales como no funcionales que obtenemos a partir de reuniones celebradas con el tutor del proyecto.

1.3.5 Diseño

En esta sección encontramos la especificación de los casos de uso más significativos. Detallaremos los actores, escenarios y condiciones que se deben cumplir. Gracias a esto podremos realizar la definición del sistema y el diagrama de clases, los cuales serán esenciales para llevar a cabo el proyecto.

1.3.6 Implementación y Pruebas

Esta sección describirá la última fase del proceso. Trataremos con detalle la implementación back-end y front-end, se hará un resumen sobre los problemas más importantes que hemos encontrado y como los hemos solucionado.

1.3.7 Conclusiones y líneas futuras

En esta última sección de la memoria podremos encontrar un resumen de los conocimientos adquiridos gracias al desarrollo del proyecto, y haremos una comparación entre los objetivos perseguidos al inicio del proyecto y los resultados conseguidos.

2

Tecnologías y herramientas

A continuación se muestra una tabla en la cual que se describen brevemente las tecnologías y herramientas que hemos utilizado a lo largo del proyecto.

Plataforma de Desarrollo	
	**Java Enterprise Edition**, una extensión de la plataforma java con un conjunto de especificaciones para funciones empresariales como la computación distribuida y los servicios web. Esta plataforma nos permite utilizar una arquitectura basada en capas distribuida y ejecutada sobre un servidor de aplicaciones.

Software	
	Netbeans, es un entorno de desarrollo libre para Java. Este IDE permite desarrollar aplicaciones desde un conjunto de componentes modulares software. Contiene muchas funcionalidades, para distintos tipos de aplicaciones y para facilitar al máximo la programación, prueba y depuración de las aplicaciones que se desarrollan.
	**Maven** es una herramienta de software para la gestión y construcción de proyectos Java, cuyo objetivo principal es simplificar el proyecto. Se basa en el concepto de Modelo de Objetos de Proyecto (POM) el cual permite que las dependencias puedan ser declaradas para estandarizar la construcción. El archivo de configuración se llama pom.xml y en el contiene tanto información de nuestro proyecto, como una sección de dependencias y de plugins.
	**MagicDraw**, es una herramienta de modelado, muy utilizada a lo largo de nuestro grado, destinada a aumentar el balance en el desarrollo software reduciendo el costo en términos de tiempo y dinero. Es una herramienta utilizada para desarrollar los diagramas de uso, de clase o de secuencia haciendo uso de la notación UML.

Base de Datos	
	MongoDB es una base de datos distribuida NOSQL free to use de documentos que ofrece una gran escalabilidad y flexibilidad, y un modelo de consultas e indexación avanzado. Almacena datos en documentos JSON flexibles, permitiendo manejar los datos de una forma más sencilla que una base de datos relacional.

	Robo 3T es una interfaz gratuita para administrar una base de datos MongoDB, con esta aplicación podemos ver los datos que se encuentran en la base de datos y realizar operaciones sobre ellos.

FrameWorks	
	Spring es un framework para el desarrollo de aplicaciones y contenedor de inversión de control, de código abierto para la plataforma Java. Está diseñado para poner en funcionamiento el proyecto lo más rápido posible con una configuración inicial mínima. Este framework no impone ningún modelo de programación en particular, pero es considerado como una alternativa, o complemento al modelo Enterprise JavaBean.

Desarrollo Web	
	JavaScript es un lenguaje de programación interpretado usado principalmente en el lado del cliente, cuya utilidad principal es la creación de páginas web dinámicas, además de permitir mejoras en la interfaz de usuario.
	JQuery es una librería de JavaScript de código abierto, la cual permite simplificar la manera de interactuar con los documentos HTML, manejar eventos y agregar interacción con la técnica AJAX.
	D3.js, siglas que vienen de Data-Driven Documents, es una biblioteca de JavaScript que nos permite agregar infogramas dinámicos e interactivos en navegadores a partir de datos. Debido a la complejidad de esta librería podemos hace uso de c3,la cual nos facilita la generación de infogramas basados en d3.js
	Bootstrap, es una biblioteca multiplataforma o un conjunto de herramientas de código abierto.

	Bootstrap facilita la maquetación de sitios web, además de ofrecernos las herramientas para crear interfaces de usuario limpias en nuestro sitio web y que se adapte bien a toda clase de dispositivos y pantallas, ayudándonos a no tener que rediseñar el sitio.

Librerías Java	
	**OpenCSV**, es una librería para Java que permite trabajar con archivos CSV de forma muy sencilla e intuitiva.
	**Apache POI**, es una librería para Java que permite la lectura y el procesado de datos de archivos con extensión XML, XLSX...

3

Metodología y planificación

3.1. Metodología

Para el desarrollo de este proyecto se ha elegido seguir una metodología ágil. El desarrollo ágil de software conlleva un enfoque para la toma de decisiones en los proyectos software, que se refiere a métodos de ingeniería del software basados en el desarrollo iterativo e incremental. Se trata de un proceso de gestión muy beneficioso debido a que nos permite analizar y mejorar el proyecto durante el desarrollo del mismo. Una de las principales ventajas que tiene esta metodología es que permite el cambio, al ser un proceso evolutivo, sin tener que esperar al final del proyecto para corregir fallos.

Esta metodología se sustenta en tres pilares: la adaptación como ya hemos comentado, la inspección y la transparencia. Al ser un proyecto en grupo, nos hemos beneficiado mucho de la transparencia que nos da este tipo de metodología. La transparencia ha implicado una comunicación completa de todo lo que sucede en el proyecto, además de ser siempre conscientes de qué problema surgen y cuales solucionamos. En consecuencia, esto ha implicado una coordinación excelente con

mi compañero. Las tareas han sido llevadas a cabo de dos maneras diferentes, en primer lugar encontramos las implementadas en conjunto facilitando la toma de decisiones y siendo más eficaces, haciendo uso de la técnica de programación por pares (*pair programming*). Esta técnica de desarrollo de software ágil que consiste en dos programadores trabajando juntos en un mismo dispositivo. Uno de ellos haría el papel de driver, el que escribe el código, y el otro de el observador, que su papel es revisar el código, estos roles son cambiados periódicamente. Esto nos ha permitido mejorar el código y una mejor cohesión de equipo entre otras ventajas.

La comunicación con el tutor del proyecto también ha sido de vital importancia, debido a que él nos proponía nuevas funcionalidades para añadirle al proyecto a medida que íbamos avanzando con el mismo. Además nos daba una visión externa al proyecto la cual era de gran ayuda, y gracias a ella hemos podido mejorar muchos aspectos de la aplicación.

3.2 Planificación

A continuación se muestra una tabla con las principales tareas de las que se ha compuesto nuestro proyecto. Nos encontramos con una columna "horas" por cada miembro del grupo, que nos indica la dimensión de las tareas.

Fase	Ricardo	Javier
Análisis y captura de requisitos	**20**	**20**
Modelado	**3**	**3**
Diseño	**40**	**40**
Implementación back-end		
Conexión a la base de datos	1	4
Identificar formato ficheros	1	3
Convertir excel a csv	10	14
Elegir archivo principal	0	2
Elegir tipado de las cabeceras	8	8
Elegir cabeceras a mostrar	3	3
Relacionar datos entre ficheros	100	100
Mostrar resultado	7	18
Descargar fichero	1	1
Guardar fichero en base de datos	4	4
Mostrar historial de ficheros	4	0
Gestión de usuarios	19	11

Combinar datos	6	0
Buscador en resultado	6	0
Implementación front-end		
CSS y boostrap	8	34
Gráficas de datos	30	6
Realización de la memoria	25	25

4

Análisis de Especificación

4.1 Análisis del Proyecto

En primer lugar vamos a analizar la situación que teníamos cuando surgió la idea de realizar este proyecto.

Nos encontramos con que el equipo administrativo de cualquier centro, se encuentra a menudo con archivos de tipo .xlsx (Microsoft Excel) o .csv con cantidades enormes de datos, los cuales tienen que ser procesados, combinando columnas de unos con otros e incluso uniendo datos, trabajo que actualmente se realiza manualmente. A esto se le añade una dificultad, la diversidad de datos que encontramos en ellos y el formato que trae cada archivo, distintos entre ellos, debido a que no hay un estándar.

Debido a este escenario, surge la idea de automatizar todo el proceso por una herramienta simple e intuitiva que facilite esta labor.

El proceso descrito conlleva una gran dificultad, debido a la unión de los datos, pues nos pueden llegar datos numéricos, categóricos o simplemente de texto. Según el tipo de datos que nos lleguen deberíamos de poder realizar ciertas acciones.

Además se le añade a este proyecto un apartado de análisis de datos mediante gráficas. Gracias al proceso de unir datos, podríamos representar datos categóricos en el eje x frente a datos numéricos en el eje y. Es preciso aclarar que con datos categóricos nos referimos a datos que se pueden dividir en categorías enumeradas.

4.2. Requisitos

4.2.1. Requisitos Funcionales

En esta sección vamos a listar los requisitos funcionales. Estos requisitos definen la funcionalidad del sistema y definen los roles que tendrán los diferentes usuarios del sistema. Para una mejor compresión del lector los requisitos serán listados por categorías, con un identificador, nombre y una breve descripción.

Las categorías elegidas son sistema y aplicación web, en ellas se encuentran divididas la funcionalidad considerando si pertenece más a la parte del sistema o a la parte web.

Categoría	Id	Nombre	Descripción
Sistema	S1	Conectar a Base de Datos	La aplicación debe estar conectada con una base de datos
	S2	Convertir archivos xlsx a csv	El usuario debe poder subir archivos de tipo xlsx (Microsoft Excel) o CSV de manera indiferente
	S3	Elegir archivo principal	Posibilidad de elegir un archivo principal para realizar las operaciones
	S4	Elegir el tipado	El usuario podrá seleccionar el tipo de datos de cada cabecera
	S5	Relacionar datos	Posibilidad de elegir qué cabeceras de qué archivos se van a combinar y unir los datos por dichas cabeceras
	S6	Unir datos iguales	Posibilidad de unir datos iguales de una misma columna

	S7	Cálculos al unir	Si los datos a unir son de tipo numérico, dar posibilidad de unir mediante suma o promedio

Categoría	Id	Nombre	Descripción
Aplicación Web	AW1	Realizar Login	Los usuarios deberán identificarse en la aplicación
	AW2	Realizar Logout	Los usuarios podrán cerrar sesión de la aplicación
	AW3	Subir Archivos	Opción de subir uno o más archivos para procesar la información
	AW4	Ver resultado	Los usuarios podrán ver el resultado de sus operaciones en la web sin necesidad de descargar o guardar el archivo
	AW5	Buscador en resultado	El usuario podrá buscar una fila concreta del archivo poniendo el dato en concreto o parte de él
	AW6	Guardar Archivo Resultante en Base de Datos	Posibilidad de guardar el archivo resultante en la base de datos para poder ser consultado a posteriori
	AW7	Descargar Archivo Resultante	Posibilidad de descargar el archivo resultante en formato .csv
	AW8	Ver historial	Los usuarios podrán ver un historial de sus acciones en el sitio web
	AW9	Mostrar gráficas	Los usuarios podrán ver diferentes tipos de gráficas sobre los datos de los archivos
	AW10	Crear Usuarios	El administrador podrá crear nuevos usuarios.

4.2.2. Requisitos No Funcionales

A continuación pasamos a listar los requisitos no funcionales, estos requisitos no describen funciones, sino características de funcionamiento. Seguiremos un modelo de listado similar al apartado anterior.

Categoría	Descripción
Seguridad	Autenticación basada en la encriptación y desencriptación de contraseñas
Seguridad	Cuando el administrador cree un nuevo usuario no sabrá la contraseña de éste.
Seguridad	No se permitirá el acceso a usuarios ajenos al sistema
Seguridad	Solo se permitirá el acceso a los datos pertenecientes al usuario y no a los de ningún otro
Accesibilidad	El sistema podrá utilizarse mediante las versiones más actuales de cualquier navegador (Google Chrome, Firefox, Safari, Internet Explorer…)
Interfaz	El usuario podrá interactuar con las gráficas que se muestren con los datos
Usabilidad	La navegación por la aplicación debe ser fácil e intuitiva
Usabilidad	La interfaz de la aplicación dará continuamente información al usuario, de que acción está llevando a cabo
Simplicidad	El usuario realizará su acción en pocos casos
Hardware	El sistema se debe de ejecutar en cualquier máquina con Java 8 y una base de datos MongoDB

4.3 Actores

En este apartado describiremos qué actores o roles consideramos en nuestro sistema.

- Usuario Normal: Este actor representa cualquier persona que acceda al sistema.
- Administrador: Este usuario tendrá todos los permisos en el sistema, además tendrá la posibilidad de crear nuevos usuarios. Estos usuarios serán creados sin contraseña.

5

Diseño

5.1 Descripción de los Casos de Uso

A continuación se listan de forma detallada los casos de uso.

Titulo	Elegir Archivo Principal
ID	CU1
Descripción	El usuario tendrá la posibilidad de elegir cuál será el archivo principal entre todos los subidos a la plataforma
Pre-condición	Haber subido algún fichero al sistema
Post-condición	El fichero seleccionado será el principal a tratar durante el proceso de análisis
Prioridad	Media
Escenario Principal	
1. El usuario introduce uno o varios ficheros al sistema. 2. El sistema muestra todos los ficheros por pantalla para que el usuario elija el principal. 3. El usuario elige el fichero principal. 4. El sistema prosigue con el análisis del fichero.	
Escenario alternativo	
Requisito asociado	

S3

Titulo	Elegir el tipado
ID	CU2
Descripción	El usuario será capaz de elegir qué tipo de dato contiene cada columna de cada archivo que haya subido.
Pre-condición	Haber subido uno o varios archivos al sistema y haber elegido el archivo principal.
Post-condición	Quedarán registrados el tipo de dato que contenga cada columna de cada archivo.
Prioridad	Alta
Escenario Principal	
1.El usuario selecciona el archivo principal. 2.El sistema muestra al usuario todas las cabeceras de todos los archivos subidos, dando la opción de elegir qué tipo de dato contiene cada una mediante un seleccionable. 3. El usuario selecciona todos los tipos de datos que contienen las columnas. 4. El sistema guarda la información y muestra la siguiente pantalla.	
Escenario alternativo	
-	
Requisito asociado	
S4	

Titulo	Relacionar datos
ID	CU3
Descripción	El usuario podrá seleccionar qué cabeceras va a querer tratar de cada archivo, y además con qué cabeceras se va a combinar los datos.
Pre-condición	Haber subido uno o varios archivos al sistema, haber seleccionado el archivo principal y haber elegido el tipado de cada uno de ellos
Post-condición	Los datos a tratar quedarán completamente relacionados y guardados en el archivo principal.
Prioridad	Alta
Escenario Principal	
1. El usuario selecciona el tipado de los archivos 2. El sistema guarda la información y muestra una pantalla las cabeceras de todos los archivos, dando opción al usuario a relacionarlas entre ellas.	

3. El usuario realiza todas las relaciones que crea convenientes. 4. El sistema guarda la información y muestra el resultado final por pantalla.
Escenario alternativo
Requisito asociado
S5

Titulo	Unir datos iguales
ID	CU4
Descripción	El usuario tendrá la capacidad de unir los datos que estén duplicados de cualquier columna, calculando el promedio o la suma de sus datos numéricos asociados.
Pre-condición	Haber guardado un fichero en el sistema.
Post-condición	El fichero quedará modificado con los cambios realizados en el sistema.
Prioridad	Media
Escenario Principal	
1.El usuario accede a un archivo registrado en su historial de ficheros. 2.El sistema muestra los datos por pantalla y la opción de combinar los datos. 3.El usuario selecciona la opción de combinar los datos. 4.El sistema modifica los datos, eliminando los datos duplicados y juntándolos en una sola fila, asociando el promedio o la suma a esta fila de los datos numéricos que pertenecieran a cada dato único.	
Escenario alternativo	
Requisito asociado	
S6	

Titulo	Realizar Login
ID	CU5
Descripción	Permite al usuario validar su identidad y acceder al sistema.
Pre-condición	El usuario debe introducir usuario y contraseña.
Post-condición	El sistema permite acceso a la aplicación
Prioridad	Alta
Escenario Principal	
1. El usuario introduce usuario y contraseña en el login.	

2. El sistema verifica que el usuario se encuentra registrado. 3. El sistema verifica que la contraseña coincide con el usuario. 4. El sistema permite que el usuario acceda y le redirige a la pagina inicial.
Escenario alternativo
2.b El usuario no se encuentra registrado. 2.b1 El sistema lanza un error comunicándole que el usuario no existe. 2.b2 Regresa al paso 1. 3.b La contraseña es incorrecta. 3.b1 El sistema lanza un error comunicando que la contraseña no es correcta. 3.b2 Regresa al paso 1.
Requisito asociado
AW1

Titulo	Realizar Logout
ID	CU6
Descripción	El usuario podrá cerrar su sesión
Pre-condición	El usuario debe tener su sesión abierta
Post-condición	El sistema cierra su sesión con éxito.
Prioridad	Alta.
Escenario Principal	
1. El usuario selecciona la opción de cerrar sesión 2. El sistema cierra la sesión y redirige a la pagina de login.	
Escenario alternativo	
Requisito asociado	
AW2	

Titulo	Subir Archivos
ID	CU7
Descripción	El usuario podrá subir uno o más archivos al sistema para procesarlos.
Pre-condición	El usuario debe de tener la sesión iniciada.
Post-condición	El archivo será subido al sistema
Prioridad	Muy Alta
Escenario Principal	
1. El usuario hace clic en subir archivo.	

2. El sistema muestra un explorador de archivos. 3. El usuario selecciona uno o más archivos a procesar. 4. El sistema redirige al usuario para que seleccione el archivo principal.
Escenario alternativo
3.b El usuario no selecciona archivos. 3.b1 El sistema cierra el explorador de archivos.
Requisito asociado
AW3

Titulo	Ver resultado.
ID	CU8
Descripción	El usuario podrá ser capaz de ver el resultado de sus archivos procesados.
Pre-condición	El usuario debe haber subido uno o varios archivos, elegido archivo principal, y elegido cabeceras a tratar.
Post-condición	El archivo resultante es mostrado con éxito
Prioridad	Alta
Escenario Principal	
1. El usuario llega hasta la página de ver resultado. 2. Visualiza el resultado de sus operaciones.	
Escenario alternativo	
Requisito asociado	
AW4	

Titulo	Buscar en resultado
ID	CU9
Descripción	El usuario podrá realizar una búsqueda dentro del resultado obtenido
Pre-condición	Haber procesado uno o más archivos
Post-condición	Obtener el resultado de la búsqueda
Prioridad	Media
Escenario Principal	
1. El usuario hace clic en el input de búsqueda 2. El usuario selecciona si quiere una búsqueda que contenga la palabra o que sea igual a la palabra. 3. El sistema busca según los parámetros. 4. El sistema muestra el resultado dinámicamente.	

Escenario alternativo
4.b El sistema no encuentra ningún dato con esos parámetros. 4.b1 El resultado mostrado será la tabla vacía.
Requisito asociado
AW5

Titulo	Guardar archivo resultante
ID	CU10
Descripción	El usuario podrá guardar el archivo resultante en el sistema para su posterior consulta
Pre-condición	Haber subido y procesado uno o más archivos
Post-condición	El archivo resultante es guardado en la base de datos.
Prioridad	Alta
Escenario Principal	
1. El usuario visualiza el resultado obtenido. 2. El usuario introduce un nombre para el archivo. 3. El usuario hace clic en guardar. 4. El sistema guarda el archivo con el nombre introducido. 5. El sistema redirige al usuario a la página inicial.	
Escenario alternativo	
2.b El usuario no introduce un nombre para el archivo. 2.b1 El sistema lanza un aviso de que no puede guardar el archivo sin nombre.	
Requisito asociado	
AW6	

Titulo	Descargar archivo resultante
ID	CU11
Descripción	El usuario podrá descargar el archivo a su dispositivo.
Pre-condición	Haber subido y procesado uno o más archivos
Post-condición	El archivo resultante es descargado con éxito.
Prioridad	Alta
Escenario Principal	
1. El usuario visualiza el resultado obtenido. 2. El usuario introduce un nombre para el archivo. 3. El usuario hace clic en descargar. 4. El sistema descarga el archivo con el nombre introducido. 5. El sistema redirige al usuario a la página inicial.	

Escenario alternativo
2.b El usuario no introduce un nombre para el archivo. 2.b1 El sistema lanza un aviso de que no puede descargar el archivo sin nombre.
Requisito asociado
AW7

Titulo	Ver historial
ID	CU12
Descripción	El usuario podrá ver un historial con los archivos procesados por él mismo.
Pre-condición	Haber subido, procesado y guardado uno o más archivos.
Post-condición	Mostrar una lista con todos los archivos procesados.
Prioridad	Alta
Escenario Principal	
1. El usuario se encuentra en la página inicial. 2. El sistema muestra un listado con los archivos procesados. 3. El usuario puede clicar sobre uno de los archivos y podrá visualizarlo.	
Escenario alternativo	
Requisito asociado	
AW8	

Titulo	Mostrar gráficas
ID	CU13
Descripción	El usuario podrá ver gráficas con representaciones sobre sus datos.
Pre-condición	Haber procesado y guardado uno o más archivos. Tener al menos una cabecera del tipo “enum” y al menos otra del tipo “numeric”.
Post-condición	La gráfica muestra los datos.
Prioridad	Alta
Escenario Principal	
1. El usuario selecciona que tipo de gráfica quiere visualizar. 2. El usuario hace clic sobre el botón visualizar. 3. El sistema le muestra la gráfica interactiva con sus datos correspondientes.	
Escenario alternativo	

Requisito asociado
AW9

Titulo	Crear Usuario
ID	CU14
Descripción	Se podrá crear usuarios nuevos con acceso a la aplicación.
Pre-condición	Haber realizado CU5 y ser administrador
Post-condición	El usuario es dado de alta en el sistema
Prioridad	Alta
Escenario Principal	
1. El sistema comprueba si el usuario tiene permisos de administrador. 2. El usuario introduce el nombre de usuario del nuevo cliente de la aplicación. 3. El sistema le muestra un mensaje de éxito.	
Escenario alternativo	
1.b El usuario no tiene permisos de administrador 1.b1 El sistema no le muestra el dialogo para añadir usuarios.	
Requisito asociado	
AW10	

5.2 Modelo de clases

La aplicación ha sido desarrollada por medio de Spring, un framework de Java para el desarrollo de aplicaciones. Haciendo uso de este framework hemos implementado nuestra aplicación por medio del patrón de diseño Modelo-Vista-Controlador , también conocido como MVC. En la capa conocida como vista, encontramos los archivos thymeleaf. Estos archivos, están basados en una tecnología de motor de plantillas, es decir, nos permite definir una plantilla basada en html que permita la interacción de datos con el controlador del sistema. Esta parte se comunica con la llamada capa de negocio (Controladores), capa que contiene toda la funcionalidad de la aplicación, por medio de modelos (Plain Old Java Object).

Una vez aclarado el patrón de diseño seguido, y el framework utilizado, podemos entender el modelo de clases de nuestra aplicación. Como podemos observar en la ilustración 4.1, nuestra aplicación está compuesta básicamente por dos entidades principales. En primer lugar tendríamos la entidad Usuario, la cual

guarda atributos sobre los usuarios del sistema, cabe destacar que se guarda un atributo password el cual almacena la contraseña del usuario encriptada y un atributo de tipo boolean que nos indica si el usuario es administrador o no. Por otro lado, tenemos la entidad Excel, ésta guarda todos los atributos de los archivos que introducimos a nuestra aplicación, desde el nombre del archivo hasta los datos que contiene.

Debido a la complejidad que presentan algunos de los atributos de la entidad Excel, vamos a explicar en detalle su estructura. El atributo data, es un atributo que almacena los datos de los archivos subidos, este tiene una estructura típica de una matriz. Una de las estructuras más importantes no sólo de esta entidad, sino de la aplicación en sí misma es el atributo match. Este atributo se utiliza para guardar las combinaciones de datos elegidas por el usuario, se trata de una lista enlazada, la cual almacena mapas enlazados cuyas claves son de tipo string y cuyos valores son de tipo mapas enlazados, que a su vez, éstos tienen como claves tipo string y como valores listas enlazadas.

Esta estructura es utilizada sólo durante el proceso de matching, proceso el cual se da cuando se combinan datos de varias cabeceras. En ese caso almacenaríamos las combinaciones elegidas por el usuario sólo durante el proceso, eliminando todos los datos cuando almacenados cuando el proceso finaliza.

Finalmente, cabe destacar que ambas entidades están relacionadas por los archivos que introduce el usuario en la aplicación y por los archivos resultantes de sus operaciones (inputExcel y outputExcel en la ilustración 4.1).

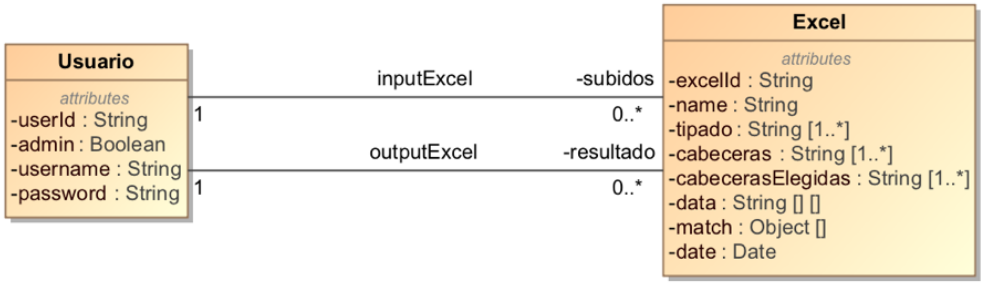


Ilustración 5.1 : Diagrama de clases de 1a aplicación

6

Implementación y pruebas

En esta sección vamos a tratar a fondo los problemas y dificultades ante las que nos hemos enfrentado durante la realización de este proyecto.

6.1 Implementación.

Una vez realizado el diseño, debemos modelar los comportamientos necesarios para esta aplicación. Con el fin de aclarar al lector las diferentes partes que han sido implementadas en el proyecto, dividiremos esta sección agrupando por funcionalidades.

6.1.1 Gestión de Usuarios.

La gestión de usuarios ha sido una funcionalidad de vital importancia que debíamos implementar en nuestra aplicación. Al ser una aplicación web, debíamos restringir la visualización de los archivos y las operaciones realizadas con ellos debido a la sensibilidad de los datos.

En cuanto a gestión de usuarios encontramos dos funcionalidades básicas:

- Iniciar sesión: Con el fin de acceder a la aplicación es necesario iniciar sesión, para ello debes tener un usuario y contraseña. Al introducir las credenciales en la aplicación, ésta buscará el usuario en la base de datos, y

si existe entonces comprobará que tiene contraseña asociada. Una vez realizado este proceso, el sistema se traerá la contraseña encriptada de la base de datos, la desencriptará y la comparará con la introducida por el usuario. Si éstas coincide entonces el sistema permitirá el acceso al usuario. Para entender mejor este procedimiento podemos fijarnos en la ilustración 6.2 de la línea 10 hasta la 20.

- Registrar usuario: Los usuarios sólo pueden ser dados de alta por un usuario de tipo administrador. Aquí nos encontramos con una dificultad añadida, pues el tutor nos recomendó que los administradores nunca supiesen la contraseña de los usuarios, debido a la seguridad y privacidad del usuario. Para resolver esta dificultad implementamos una solución la cual se basaba en que el usuario que daba de alta, tan sólo debía de introducir el nombre de usuario que quería crear. Cuando este usuario fuese a entrar al sistema, el sistema al comprobar que el usuario existe, pero que no tiene una contraseña asociada le reenviaría a un formulario en el cual el usuario introduciría su contraseña, de forma transparente al usuario administrador. Una vez introducidas las credenciales se encripta la contraseña y se le asocia al usuario, actualizando la base de datos como podemos ver en la ilustración 6.1. En la ilustración 6.3 podemos ver la interfaz asociada con este proceso.

```
@RequestMapping("/registrar")
public String registrar(Model model, @RequestParam("pass") String password, HttpSession sesion) {
    BasicTextEncryptor textEncryptor = new BasicTextEncryptor();
    String privateData = password;
    textEncryptor.setPasswordCharArray(password.toCharArray());
    String myEncryptedText = textEncryptor.encrypt(privateData);
    String name = (String) sesion.getAttribute("nameUser");
    Usuario u = dbUser.findByUsername(name);
    u.setPassword(myEncryptedText);
    dbUser.save(u);
    return "upload";
}
```

Ilustración 6. 1: Función para registrar un usuario

- Cerrar sesión: Desde cualquier punto de la aplicación, podemos cerrar sesión desde un botón que se encuentra en la parte superior derecha. El proceso de cerrar sesión es bastante sencillo, en el backend lo que hacemos es poner el atributo de sesión de usuario a null, por lo tanto obligamos a volver a iniciar sesión si quiere volver a acceder a alguna página de nuestra aplicación.

En la Ilustración 6.2 podemos visualizar el código que se usa para la gestión de usuarios, además vemos a modo de comentarios las diferentes comprobaciones que lleva a cabo.

```java
@RequestMapping("/login")
    public String login(Model model, @RequestParam("username") String usuario, @RequestParam("pass") String password, HttpSession sesion) {

        Usuario user = dbUser.findByUsername(usuario);
        model.addAttribute("errorPass", false);
        model.addAttribute("error", false);
        if (user != null) { //si existe el usuario
            if (user.getPassword() != null) { //si tiene contraseña
                try {
                    String pass = user.getPassword();
                    BasicTextEncryptor textEncryptor = new BasicTextEncryptor();
                    textEncryptor.setPasswordCharArray(password.toCharArray());
                    String plainText = textEncryptor.decrypt(pass);
                    if (plainText.equals(password)) { // login ok
                        sesion.setAttribute("usuario", user);
                        model.addAttribute("listaOutput", user.getOutputExcels());
                        model.addAttribute("admin", user.getAdmin());
                        model.addAttribute("mostrarBack", false);
                        List<Usuario> users = dbUser.findAll();
                        return "upload";
                    }
                } catch (EncryptionOperationNotPossibleException e) {
                    System.out.println("ERROR CONTRASEÑA");
                    model.addAttribute("errorPass", true);
                    return "index";
                }
            } else { // No existe la contraseña  y se registra
                sesion.setAttribute("nameUser", usuario);
                sesion.setAttribute("usuario", user);
                return "registrar";
            }
        } // no existe el usuario
        model.addAttribute("error", true);
        return "index";
    }
```

Ilustración 6. 2: Código de gestión de usuarios

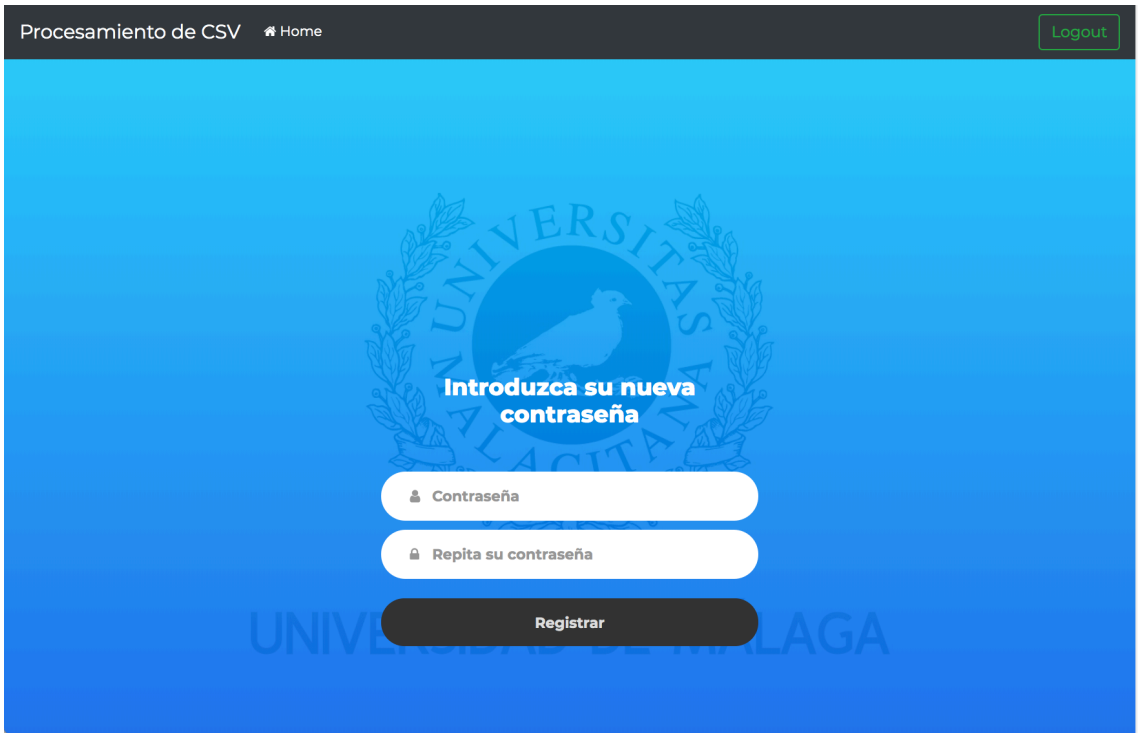


Ilustración 6. 3: Formulario para dar de alta a un nuevo usuario

6.1.2 Subir archivos

En este punto hablaremos sobre la funcionalidad principal de la aplicación: subir archivos y procesarlos.

El usuario al entrar en la aplicación lo primero que se encuentra es con la interfaz inicial mostrada en la ilustración 6.4. En ella podemos ver un logo de Excel, si pinchamos en el nos saldrá un explorador de archivos a través del cual podremos seleccionar uno o más archivos con el fin de procesarlos.

En esta parte encontramos varias dificultades, pues desde un primer momento no utilizábamos Spring. Fue entonces, tras varios intentos, cuando al consultarlo con el tutor nos decidimos a utilizar el framework Spring en nuestro proyecto, pues facilita todo este proceso. En el método recogemos los archivos, y por cada archivo comprobamos si es de tipo xlsx o csv. En el caso de que el archivo sea de tipo xlsx, lo pasamos a otro método que se encarga de pasarlo a tipo csv para así siempre trabajar con este tipo de archivos.

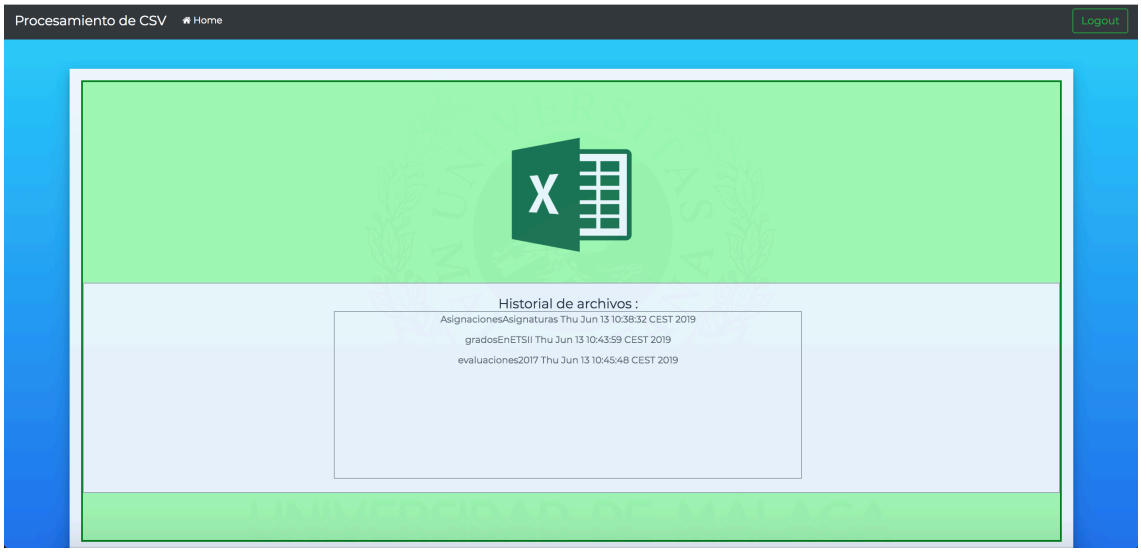
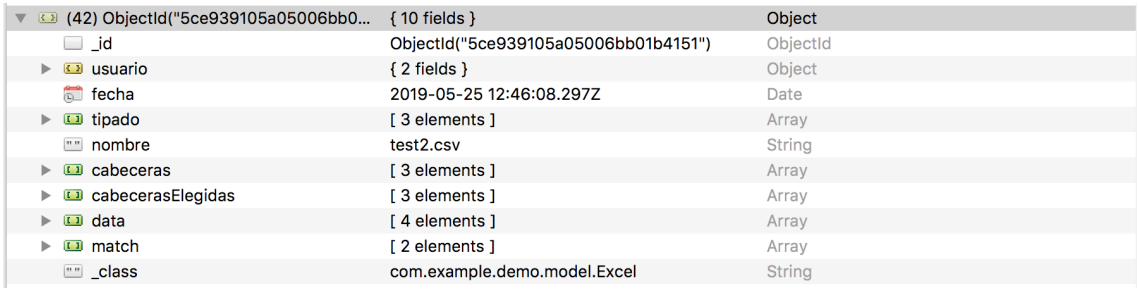


Ilustración 6. 4: Interfaz principal de la aplicación

El proceso de pasar un archivo de extensión .xlsx a CSV, llevado a cabo por el método crearCSV() que podemos encontrar en la clase ExcelController de la aplicación, es sencillo. En este método lo que hacemos en primer lugar es crear un archivo de tipo csv vacío, posteriormente pasamos a recorrer el archivo xlsx celda por celda, si la celda no es vacía comprobamos con un "switch case" qué tipo de valor contiene la celda y lo añadimos al nuevo csv creado. En el caso que sea vacía si se trata de una cabecera, introducimos el string "sin cabecera" para indicar al usuario las cabeceras que le faltan al archivo introducido.

Una vez realizada la conversión, el siguiente paso es guardar los datos del archivo en el sistema. En la base de datos, guardamos tanto los datos del archivo, como los datos del usuario que ha subido ese archivo. En la ilustración 6.5 podemos ver los campos que tiene en la base de datos MongoDB.



(42) ObjectId("5ce939105a05006bb0...	{ 10 fields }	Object
_id	ObjectId("5ce939105a05006bb01b4151")	ObjectId
usuario	{ 2 fields }	Object
fecha	2019-05-25 12:46:08.297Z	Date
tipado	[3 elements]	Array
nombre	test2.csv	String
cabeceras	[3 elements]	Array
cabecerasElegidas	[3 elements]	Array
data	[4 elements]	Array
match	[2 elements]	Array
_class	com.example.demo.model.Excel	String

Ilustración 6. 5: Documento guardado en MongoDB

6.1.3 Archivo Principal

Una vez realizado todo el proceso descrito en el punto 6.1.2 para subir el archivo, el usuario es redirigido a una nueva página en la cual debe especificar qué archivo será el principal para realizar todas las operaciones.

Para ello, lo que hacemos es introducir el indicado por el usuario como primer archivo en la lista enlazada y después introducir el resto. Esta lista enlazada será un atributo de sesión que nos servirá para realizar el resto de operaciones.

6.1.4 Seleccionar tipado por columnas

Mediante un seleccionable le damos la posibilidad al usuario de introducir que tipo de datos lleva cada columna. El seleccionable permite elegir entre Enum, String y Numeric. El tipo String es el tipo de dato que te da el sistema por defecto, pues realmente con este tipo de dato no podemos realizar operaciones de gráficas posteriormente. Si elegimos Enum indicamos que se trata de un dato categórico, mientras que si elegimos Numeric indicamos que se trata de un dato numérico.

Estos parámetros indicados por el usuario serán guardados en la base de datos mediante el atributo “tipado”. Este atributo nos servirá a la hora de mostrar las gráficas, como se explica en el punto 6.1.11.

6.1.5 Seleccionar cabeceras

Una vez elegido el tipado de las cabeceras, el siguiente paso es elegir sobre qué cabeceras quieres trabajar. Para ello, lo que hacemos es un checkbox en el

cual el usuario tiene que seleccionar para cada archivo las cabeceras que va a utilizar.

Este parámetro lo traemos de html como se muestra en la ilustración 6.6. Los datos que traemos son <nombre de la columna> @ a <id>. Este id lo conseguimos con thymeleaf, gracias a la función #ids.seq el cual asocia un id único a cada dato.

```
<li onchange="atLeastOne()"  th:each="col : ${archivo}">
    <div class="checkbox checkbox-primary">
        <label><span
                th:text="${col}"></span> <input   type="checkbox" th:name="idCabeceras"
                    th:value="${col}+'@'+ ${#ids.seq('a')}"/>
        </label>
    </div>
</li>
```

Ilustración 6. 6: Código html de recogida de cabeceras

Este parámetro lo procesamos asociándolo al archivo que pertenece y lo guardamos en la base de datos mediante el atributo “cabecerasElegidas”. Desde este método comprobamos el numero de archivos introducidos, si este es mayor de uno entonces el sistema redirige al usuario a la página match, sino, muestra el resultado.

6.1.6 Match

Este punto es la tarea de más envergadura de la aplicación y la que más esfuerzo ha supuesto. Esta función es la que se encarga de unir datos por cabeceras.

Nos encontramos ante muchos problemas con esta función pues debíamos de combinar datos de diferentes cabeceras de diferentes archivos, lo cual no es para nada trivial. A priori decidimos tener un mapa el cual las claves serían los nombres de las cabeceras, pero esta idea fue descartada pues podría darse el caso de tener dos archivos con el mismo nombre de cabeceras.

Finalmente, tras plantear diversas estructuras nos decantamos por una que contuviese tanto el nombre de la cabecera como el id del archivo con el que lo íbamos a relacionar. Esta estructura fue la siguiente: una lista enlazada la cual tuviese mapas enlazados con clave de tipo String y valores de tipo mapas enlazados. Estos valores de tipo mapas enlazados tendrían clave String y valores de tipo lista enlazada.

Esta estructura, en el primer mapa enlazado guarda como clave el nombre de la columna del archivo principal y como valor un segundo mapa enlazado, cuya clave sería el id del archivo con el que se combina y cuyo valor la columna de este

último. En la ilustración 6.7 podemos ver la estructura definida con valores reales en la cual se combinan las columnas “Asignatura” del archivo principal y la columna “Estudiantes” del archivo con el id mostrado.

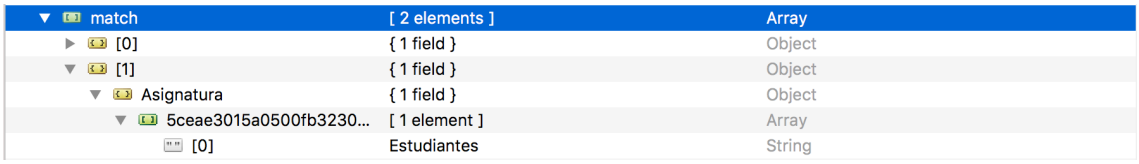


Ilustración 6. 7: Estructura match

Una vez diseñada la estructura, pasamos a la implementación, este método se encargaría haciendo uso de métodos auxiliares, que debido a la complejidad de la función nos vimos obligados a implementar, de almacenar los datos proporcionados por los archivos y por el usuario. Recorremos todos los archivos introducidos y procesamos por una parte las cabeceras elegidas para algún tipo de match y por otro lado las cabeceras no elegidas. Cuando procesamos cabeceras elegidas vamos eliminándolas de la base de datos del atributo “cabeceras elegidas”, así en el momento en el que ese atributo está vacío, significa que ya hemos realizado todas las correspondencias. Además tenemos un mapa como atributo de sesión en el que también llevamos esas correspondencias.

Cuando el usuario ha pulsado el botón finalizar de la interfaz de match (Ilustración 6.8) o no le quedan más cabeceras para poder combinar el sistema redirige al usuario a una página en la que se le muestra al usuario el resultado de su acción.

Ilustración 6. 8: Interfaz de match

Además cabe destacar que podemos combinar una columna del archivo principal con dos columnas de otro archivo, por ejemplo, si en el archivo principal

tenemos una columna Nombre y Apellido, y el segundo archivo encontramos que Nombre es una columna y apellido otra, nuestro sistema nos juntará la información de las dos columnas y te las unirá, un proceso que facilitará la unión de datos.

6.1.7. Mostrar Resultado

Una vez realizadas las operaciones el usuario será redirigido a una página en la que se le mostrará el resultado. El resultado será mostrado en una tabla como podemos observar en la ilustración 6.9.

Para mostrar la tabla con los datos, nos hemos ayudado de thymeleaf, pues lo que hacemos es pasarle una matriz con los datos y gracias a esta tecnología hemos sido capaces de mostrarlos.

6.1.8. Descargar y Guardar Resultado

Una vez procesado el archivo, en la página final (Ilustración 6.9) el sistema nos da la opción de descargar el resultado o de guardarlo en el sistema dándole un nombre.

La implementación de ambos métodos es muy similar, pues al descargar el resultado en formato csv el sistema también lo guarda en la base de datos. Para guardar siguiendo el formato csv hemos tenido que tomar unas cabeceras por defecto que llevan los archivos csv y escoger el nombre del archivo dado por el usuario.

6.1.9 Buscador en resultado

Una vez mostrado el resultado, tenemos la opción de filtrar dentro de la matriz de datos resultantes. Tenemos dos tipos de búsqueda las cuales serían buscar un dato exacto, eligiendo en el seleccionable la opción “equals” o buscar datos que contengan ciertos caracteres eligiendo la opción “contains”.

En este método encontramos una gran dificultad, pues está implementado en su totalidad en JavaScript, y la dificultad viene debido a nuestro escaso conocimiento sobre esta tecnología.

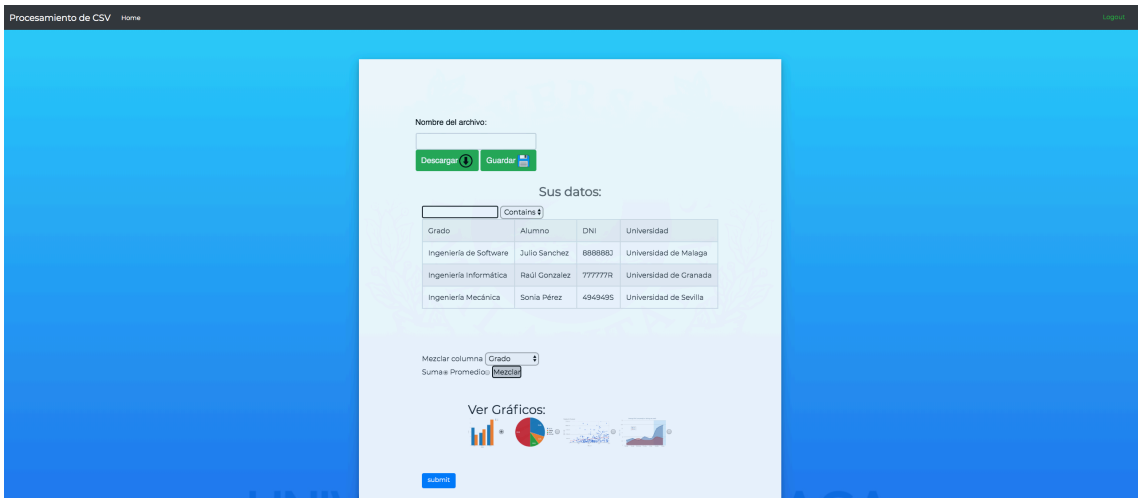


Ilustración 6. 9: Interfaz del resultado final de las operaciones

6.1.10 Mezclar Columnas.

Este método es una opción que tenemos en la página de visualizar el resultado final. En ocasiones nos encontramos que dentro de una misma columna se repiten datos y necesitamos que si dos datos de una misma columna son iguales se junten. Al juntarlos si cada uno tiene datos numéricos podemos sumarlos o hacer el promedio, sin embargo si los datos son de tipo texto entonces se juntarán dividiéndolos con el carácter "\". En la ilustración 6.10 nos vemos en esta situación y el usuario elige sumarlos por lo que nos lleva al resultado mostrado en la ilustración 6.11.

Para la realización de este método cogemos la matriz que tenemos como atributo de sesión, la columna indicada para juntar por el usuario y mediante un TreeMap almacenamos las coincidencias dentro de la columna. Y posteriormente vemos el resto de columnas su tipado y las unimos teniendo en cuenta lo citado en el párrafo anterior.

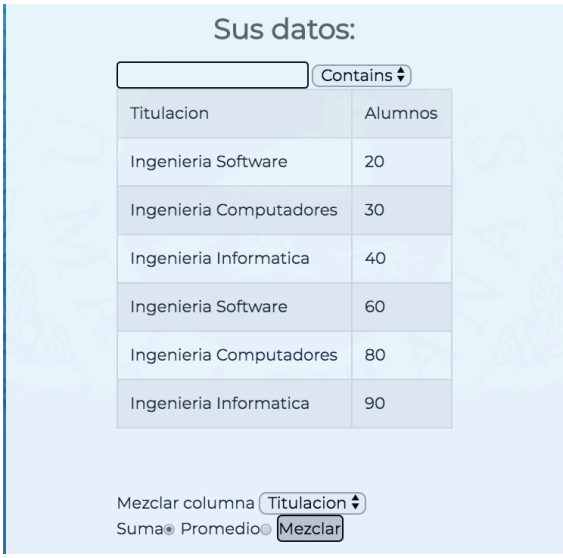


Ilustración 6. 10: Datos sin mezclar

Ilustración 6. 11: Datos mezclados con datos numéricos sumados

6.1.11 Mostrar gráficas

El usuario, en nuestra aplicación puede ver gráficas sobre los datos contenidos en los archivos introducidos. Le damos total libertad para que coja la representación que quiera de entre las cuatro posibles en la aplicación.

Para la representación de datos es necesario e importante los tipados elegidos por el usuario. Para poder ver las gráficas es necesario tener como mínimo una columna de tipo Enum y una columna de tipo Numeric. La columna de tipo numérico se representará en el eje y, mientras que la de tipo enum se representará en el eje x.

En este punto encontramos varias dificultades, en primer lugar no teníamos claro como representar los datos, pero gracias a c3.js pudimos hacerlo sin mucha dificultad. Por otro lado, encontramos la dificultad de conectar Spring con JavaScript con el fin de recoger los datos a mostrar. Esta tarea fue difícil debido a la poca documentación que encontramos. Finalmente se consiguió:

```
<script th:inline="javascript" defer>
    var matriz = /*[[${matrix}]]*/;
    var tipado = /*[[${tipado}]]*/;
    var header = /*[[${header}]]*/;
    var first1 = false;
```

Fue un poco confuso pues, parece que esta comentado, pero realmente es como se recoge desde JavaScript un dato de Spring. Una vez conseguidos los datos, y ayudados por c3.js conseguimos hacer el tratamiento de los mismos para conseguir las gráficas esperadas. En la ilustración 6.12 se muestra una gráfica realizada por nuestra aplicación en la que se muestra en el eje x grados y en el eje y el numero de alumnos de esos grados.

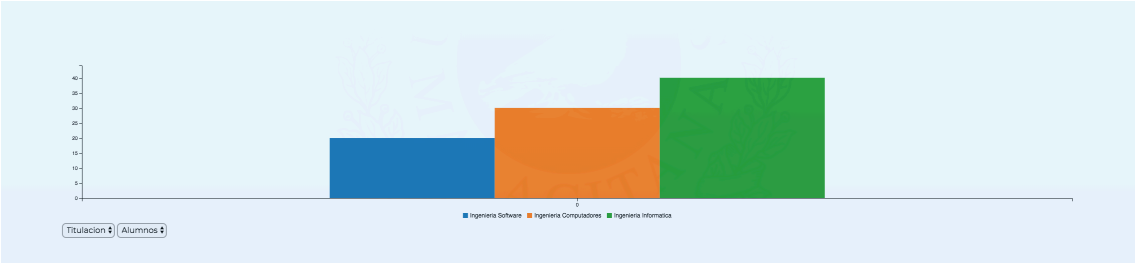


Ilustración 6. 12 Representación grafica de datos

6.1.12 Historial de Archivos

Uno de los requisitos pedidos era almacenar el historial de acciones por usuarios, que un usuario pudiese ver archivos que había subido anteriormente, volvérselos a descargar o ver representaciones, ordenados por fecha de creación.

Con este requisito teníamos una dificultad, debido a una restricción impuesta, solo podría ver los archivos creados por el mismo, por lo que los archivos serían exclusivos para cada usuario.

Para mostrar la lista de archivos, lo que hacemos es recoger de la base de datos los archivos salientes del usuario que ha iniciado sesión y los mostramos en la tabla historial como se muestra en la ilustración 6.4.

Esta lista está compuesta por enlaces, es decir, todos los archivos pueden ser pulsados y el sistema te redirige a la página para visualizar los datos. Esta página es idéntica a la mostrada en la ilustración 6.9. Para llevar a cabo la operación, cada link se trata de una petición get. A través de esta petición nos llega, por la URL, el id del archivo que quiere visualizar. Para evitar que cualquiera con ese enlace pueda acceder al contenido lo primero que hacemos es comprobar que el usuario que pide el contenido de ese dato es el mismo que el propietario del contenido, y si es así recogemos el dato de la base de datos y se lo mostramos al usuario.

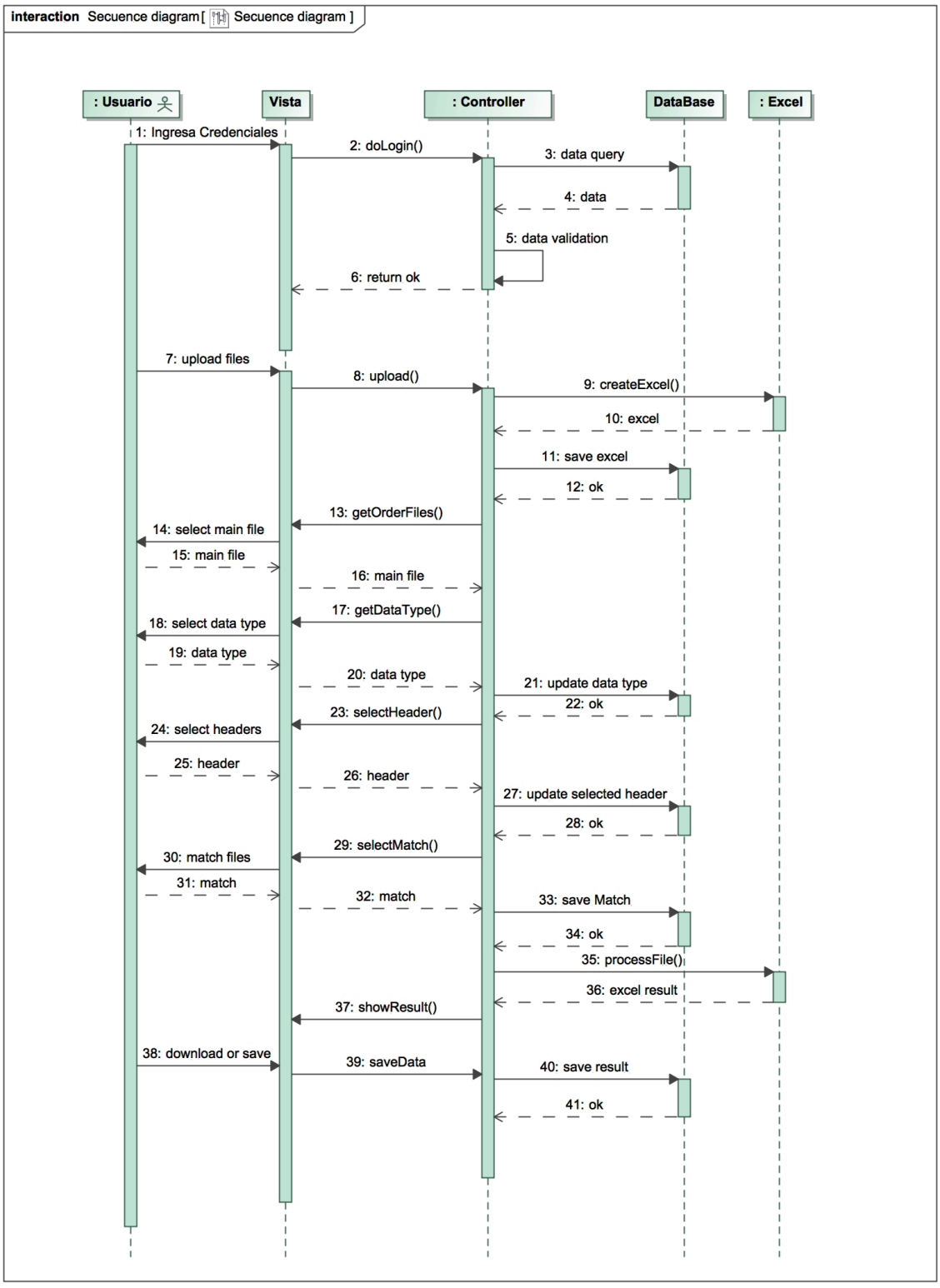


Ilustración 6. 13 Diagrama de secuencias de proceso típico en la aplicación.

6.2. Diagrama de Secuencias

Una vez explicadas todas las operaciones pasamos a mostrar el diagrama de secuencias (Ilustración 6.13) de una operación típica de nuestra aplicación.

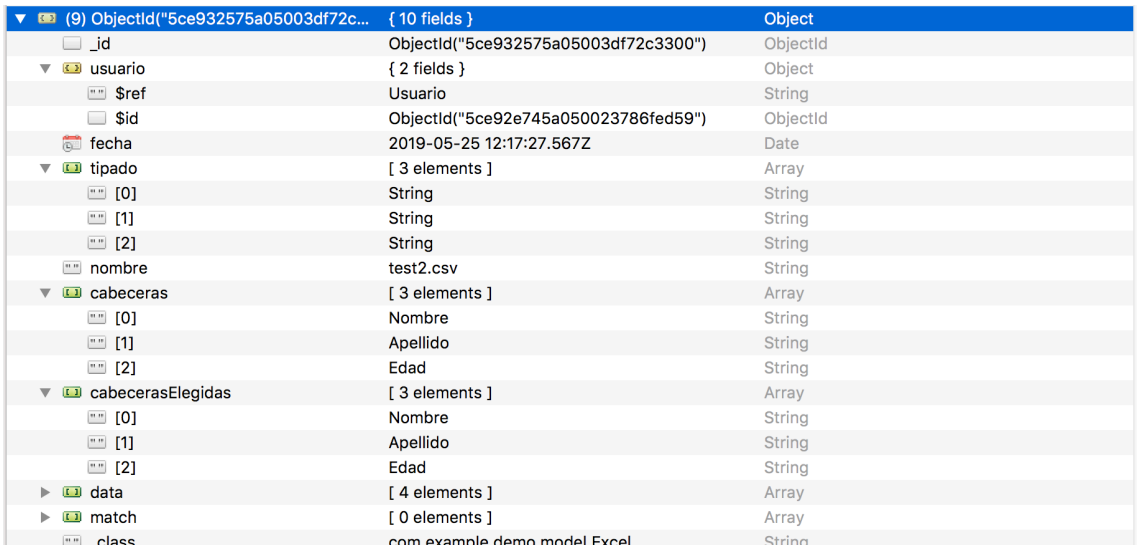
Desde el momento en el que el usuario ingresa sus datos en la herramienta hasta que obtiene el resultado final de sus operaciones.

6.3 Almacenamiento en MongoDB

Con el fin de realizar una mejor y más completa aplicación decidimos almacenar datos relacionados con los usuarios y las operaciones que realiza en la aplicación. Spring podía relacionarse con base de datos tanto relacional como no relacional por lo que estudiamos cual nos convendría mejor en nuestra aplicación.

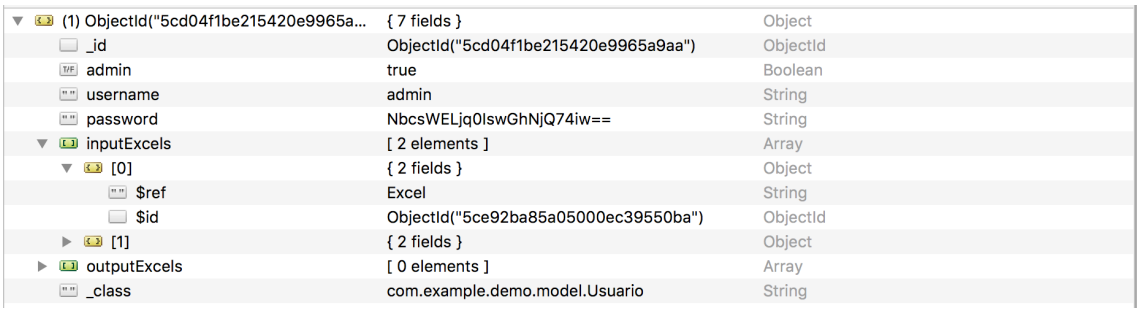
Nos decidimos por MongoDB por varios motivos: en primer lugar los datos almacenados por MongoDB son de fácil lectura para el usuario debido a que son almacenados en formato JSON, en segundo lugar vimos que MongoDB tenía un comportamiento excelente para trabajar con sistemas de gran volumen. Por otro lado, vimos algunas desventajas como que los documentos de MongoDB nos permiten almacenar toda la información que queramos, lo que nos podría ocasionar problemas en la consistencia de datos.

La integración con la Spring, la tecnología utilizada en nuestra aplicación se presenta bastante fácil y sencilla. Como vemos en la ilustración 5.1 nuestra aplicación se basa en dos clases principales: Excel y Usuario. Los objetos creados por estas clases son los que almacenamos en formato JSON. Para manejar los datos almacenados nos ayudamos de la herramienta Robo 3T que nos ofrece una interfaz grafica para visualizar y administrar la base de datos. Además nos permite visualizar los datos en formato JSON de una manera más simple. En la ilustración 6.14 vemos como se almacena un objeto Excel, mientras que en la ilustración 6.15 vemos como lo hace un objeto Usuario.



(9) ObjectId("5ce932575a05003df72c...	{ 10 fields }	Object
_id	ObjectId("5ce932575a05003df72c3300")	ObjectId
usuario	{ 2 fields }	Object
$ref	Usuario	String
$id	ObjectId("5ce92e745a050023786fed59")	ObjectId
fecha	2019-05-25 12:17:27.567Z	Date
tipado	[3 elements]	Array
[0]	String	String
[1]	String	String
[2]	String	String
nombre	test2.csv	String
cabeceras	[3 elements]	Array
[0]	Nombre	String
[1]	Apellido	String
[2]	Edad	String
cabecerasElegidas	[3 elements]	Array
[0]	Nombre	String
[1]	Apellido	String
[2]	Edad	String
data	[4 elements]	Array
match	[0 elements]	Array
class	com.example.demo.model.Excel	String

Ilustración 6. 14 Objeto Excel almacenado en MongoDB



(1) ObjectId("5cd04f1be215420e9965a...	{ 7 fields }	Object
_id	ObjectId("5cd04f1be215420e9965a9aa")	ObjectId
admin	true	Boolean
username	admin	String
password	NbcsWELjq0IswGhNjQ74iw==	String
inputExcels	[2 elements]	Array
[0]	{ 2 fields }	Object
$ref	Excel	String
$id	ObjectId("5ce92ba85a05000ec39550ba")	ObjectId
[1]	{ 2 fields }	Object
outputExcels	[0 elements]	Array
_class	com.example.demo.model.Usuario	String

Ilustración 6. 15 Objeto Usuario almacenado en MongoDB

Una vez explicado que es MongoDB y por qué decidimos hacer uso de esta tecnología en nuestra aplicación pasamos a explicar la integración con el resto de la aplicación.

En primer lugar, tuvimos que definir las clases java relacionadas con la base de datos con el fin de almacenarlas. Con el fin de que funcione correctamente debemos definir una serie de etiquetas las cuales podemos observar en la ilustración 6.16. Las más importantes son Document que define la clase para poder ser guardada en la base de datos. Id, que identifica el atributo id de la clase. Y finalmente DBRef, que se trata de una etiqueta que se coloca en los atributos que hacen referencia a otros objetos de la base de datos.

```
@Document(collection="Excel")
public class Excel {
    private ExcelRepository excelRepository;
    @Id
    private String id;
    @DBRef
    private Usuario usuario;

    private Date fecha;
    private String[] tipado;
    private String nombre;
    private String[] cabeceras;
    private LinkedList<String> cabecerasElegidas;
    private String[][] data;
    private LinkedList<LinkedHashMap<String, LinkedHashMap<String, LinkedList<String>>>> match;
```

Ilustración 6. 16 Clase Excel

También es necesario crear la clase que actúe como enlace entre el modelo de la aplicación y la base de datos (Ilustración 6.17). Esta clase extiende de MongoRepository.

```
@Repository
public interface ExcelRepository extends MongoRepository<Excel, String> {

    Excel findByNombre(String nombre);

}
```

Ilustración 6. 17 Clase ExcelRepository

Todas las operaciones vendrán en la clase Controller, en la que se encuentra todos los métodos de la aplicación. Para poder acceder a la base de datos tenemos que definir objetos de los citados repositorios, como vemos en la ilustración 6.18.

Las operaciones más importantes para tratar con la base de datos son:

- Save: se utiliza para guardar o actualizar campos en la base de datos.
- Find: Para obtener un objeto concreto de la base de datos.
- FindAll: Para obtener todos los objetos de la misma clase de la base de datos.

```
@Controller
@SessionAttributes("map")
public class ExcelController {

    @Autowired
    private ExcelRepository db;
    @Autowired
    private UsuarioRepository dbUser;
```

Ilustración 6. 18 Declaración de repositorios.

6.4 Pruebas

En este proyecto no nos hemos centrado en el diseño y desarrollo de pruebas estrictamente dichas. Las pruebas realizadas han sido depurando en nuestra aplicación, y probando distintos casos de usos y archivos alternativos para forzar el sistema y así poder solucionar los errores más importantes.

Como ingenieros sabemos de la vital importancia de las pruebas para el desarrollo software, pero siendo conscientes del tiempo que teníamos hemos decidido centrarnos más en la parte de desarrollo e implementarle más funcionalidades que en el diseño de pruebas para el sistema.

7

Conclusiones y líneas futuras

Esta sección está dedicada a las conclusiones a las que se ha llegado como consecuencia de la realización de este proyecto, y acabaremos comentando el futuro próximo que puede tener este proyecto.

7.1 Resultado final y Conclusiones

La aplicación final se presenta como una herramienta administrativa que facilitará diariamente las tareas a los trabajadores de este sector.

Los trabajadores podrán dedicar más tiempo a otras tareas gracias a la facilidad que les da nuestra aplicación. Bajo mi punto de vista, el problema planteado inicialmente ha sido solucionado y, aunque somos conscientes de que la aplicación podría tener más funcionalidades, tema que trataremos en el punto 7.2, hemos implementado aquellas más prioritarias.

En un primer lugar, el análisis del problema y el diseño de la solución ha sido bastante bueno, porque aunque nos ha llevado bastante tiempo nos aclaró sobre cómo llevar a cabo la implementación. El diseño ha estado condicionado al framework Spring, que si bien es cierto, antes de empezar el proyecto no teníamos conocimiento de su existencia, pero gracias a la documentación hemos podido llevar a cabo el desarrollo de esta aplicación.

Cabe destacar que la organización que tuvimos mi compañero y yo fue bastante buena, esta organización fue de vital importancia al ser un trabajo en grupo. Hemos estado en contacto a diario siendo conscientes del trabajo que se iba realizando y los problemas que iban surgiendo. Además, hemos tenido reuniones periódicas con el tutor en las cuales ha podido visualizar el progreso del proyecto.

La aplicación se presenta intuitiva al usuario, es verdad que la interfaz gráfica es demasiado simple, pero al ser simple, destacamos las funcionalidades principales de la aplicación. Al ver la aplicación se nota que la interfaz no es lo que más hemos trabajado, debido a la complejidad del problema propuesto.

A nivel académico, de este proyecto me llevo muchas cosas, en primer lugar el trabajo en equipo. En nuestro sector es muy importante saber trabajar en equipo y, aunque esto se intenta trabajar en la carrera, este proyecto ha sido de más envergadura por lo que ha sido necesario complementarme con mi compañero y realizar una organización más detallada. En segundo lugar, he puesto en práctica mucho de los conocimientos adquiridos a lo largo de la carrera, para darle solución a un problema real. Por último, en este proyecto se ha trabajado ser autosuficiente y autodidacta pues en ocasiones nos encontrábamos ante problemas de difícil solución a priori pero tras leer documentación, leer en foros e informarnos sobre las posibles soluciones hemos podido resolver estas dificultades.

Me ha motivado mucho hacer una aplicación inexistente en el mundo tecnológico, y me ha servido para ser consciente de qué podemos llegar a hacer, pues entre mi compañero y yo hemos realizado una aplicación, con una estructura de datos propia para conseguir solucionar el problema propuesto por el tutor.

7.2 Líneas futuras.

Esta aplicación se puede exprimir al máximo, pues nos encontramos ante una aplicación única que puede ayudar al personal del sector administrativo.

Aunque el estado en el que se encuentra la aplicación es bastante bueno, pues realiza las funciones principales para la que se pensó, se le pueden implementar diversas mejoras.

Actualmente, los archivos pertenecen sólo a un usuario y sólo este usuario puede visualizarlas. Sería una mejora que el usuario pudiese compartir con otros usuarios de la aplicación resultados obtenidos y gráficas.

Cuando obtenemos los resultados de las operaciones obtenidas actualmente no podemos modificar celdas dentro de la aplicación, en ocasiones cuando un usuario está revisando el resultado de sus operaciones identifica erratas en ese documento. Una buena mejoría sería darle la posibilidad al usuario de corregir estos datos a través del navegador, sin la necesidad de hacerlo individualmente y volver a realizar todo el proceso.

Hay ciertas opciones, como elegir el tipado de la cabecera que se elige en uno de los primeros pasos de la aplicación y no podemos modificarlo posteriormente. Sería recomendable deshacer pasos, o incluso poder modificar tipados en cualquier momento.

En definitiva, a la aplicación aunque cumple con los requisitos propuestos inicialmente, podemos implementarle mejorías para convertirla en una aplicación mucho más completa.

Referencias

- CSV – Wikipedia
  https://es.wikipedia.org/wiki/Valores_separados_por_comas

- Spring
  https://spring.io/

- Thymeleaf
  https://www.thymeleaf.org/

- Mongo DB
  https://www.mongodb.com/

- NO SQL – Wikipedia
  https://es.wikipedia.org/wiki/NoSQL

- APACHE MAVEN
  https://maven.apache.org/

- OpenCSV.
  http://opencsv.sourceforge.net/

- Apache POI
  https://poi.apache.org/

- Stack overflow
  https://stackoverflow.com/

- Bootstrap
  https://getbootstrap.com/

- JavaScript – Wikipedia
  https://es.wikipedia.org/wiki/JavaScript

- Netbeans
  https://netbeans.org/

- MagicDraw - Wikipedia
  https://es.wikipedia.org/wiki/MagicDraw_UML

- C3.js
  https://c3js.org/

- Basic Text Encryptor – Java
  http://www.jasypt.org/api/jasypt/1.8/org/jasypt/util/text/BasicTextEncryptor.html

- W3Schools
  https://www.w3schools.com

- Metodología Scrum – Wikipedia
  https://es.wikipedia.org/wiki/Scrum_(desarrollo_de_software)

Apéndice A

Manual de Usuario

1.Creación de Usuario.

Esta opción estará solo habilitada para los usuarios de tipo administrador. Cuando un usuario administrador quiere crear un usuario lo único que tiene que hacer es introducir el nombre de usuario y pulsar el botón crear usuario, como se muestra en la ilustración MU 1.

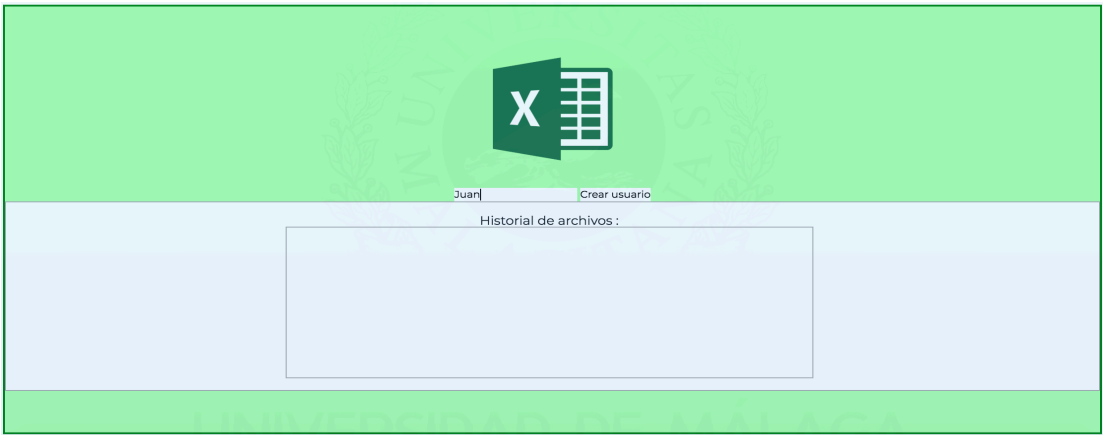


Ilustración MU 1: Creación de usuario

Si el nombre de usuario no existía con anterioridad, el sistema nos mostrará un mensaje de éxito como se muestra en la ilustración MU 2.

Una vez dado de alta el nombre de usuario del sistema, cuando éste vaya a entrar en la aplicación, el sistema detectará que no tiene contraseña asociada y entonces le pedirá que introduzca una nueva contraseña (Ilustración MU 3). Tras añadir la contraseña el usuario podrá navegar libremente por la aplicación

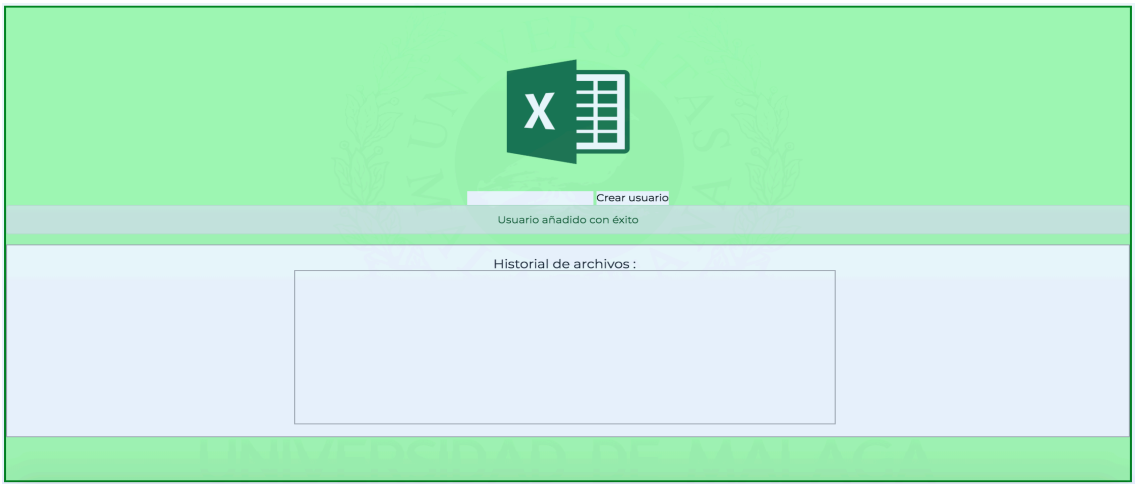


Ilustración MU 2: Mensaje de éxito al crear usuario.

2. Procesamiento de archivos

Para procesar archivos y hacer operaciones sobre ellos debemos realizar varios pasos, por eso hemos divido esta sección para aclarar cada paso.

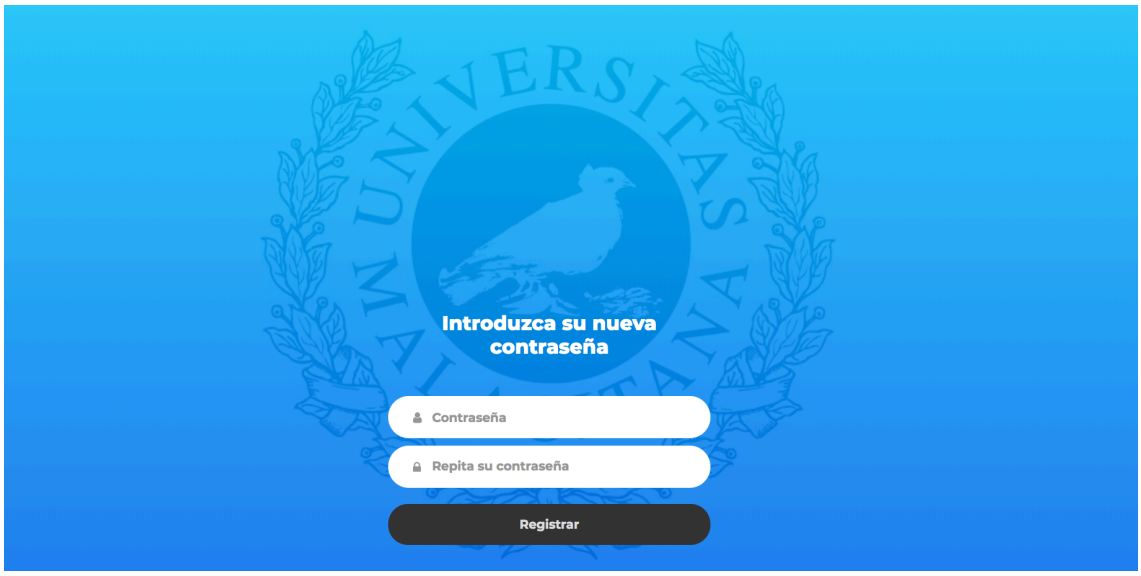


Ilustración MU 3: Añadir contraseña a nuevo usuario.

2.1 Subir archivo.

El primer paso que debe dar el usuario para realizar operaciones sobre uno o varios archivos es clicar sobre la imagen de Excel, en el centro de la interfaz principal como podemos ver en la ilustración MU 4.

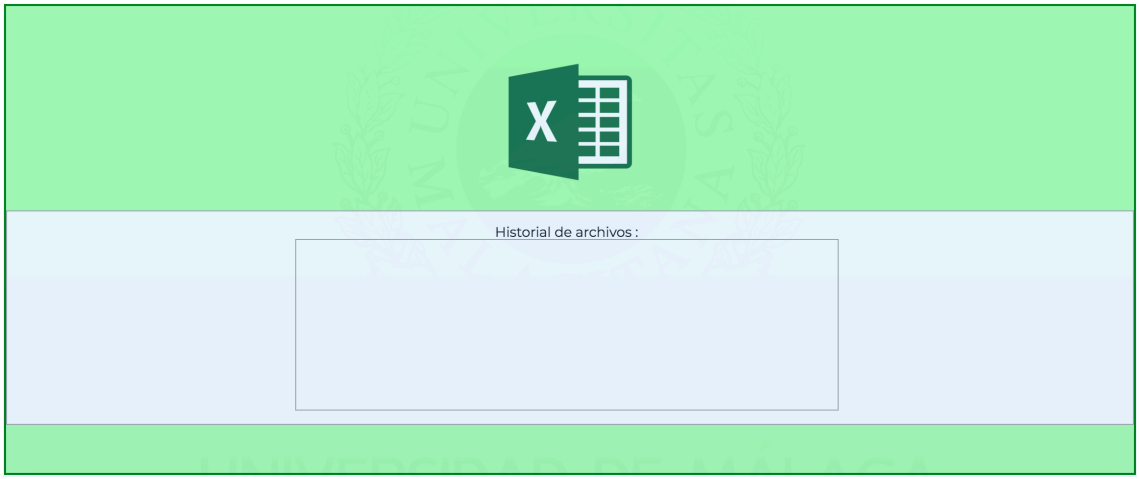


Ilustración MU 4: Interfaz principal de la aplicación

Al hacer clic sobre esa imagen, se le abrirá al usuario un explorador de archivos en el cual podrá seleccionar uno o varios archivos. Cuando estos sean seleccionados, el sistema mostrará el nombre de los archivos seleccionados y aparecerá un botón “Upload Files” para subir los archivos al sistema. (Ilustración MU 5)

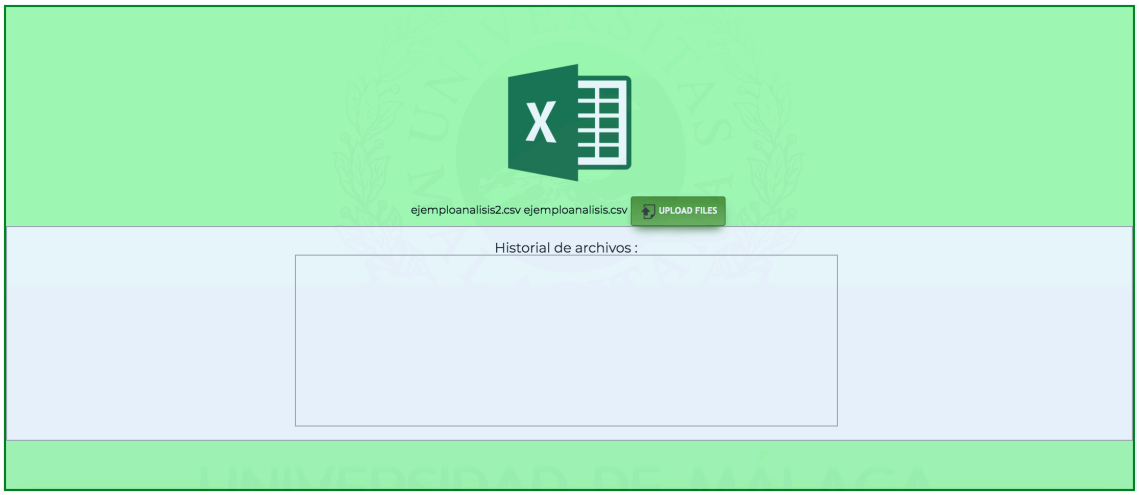


Ilustración MU 5: Subir archivos.

2.2 Seleccionar archivo principal

Una vez subidos los archivos al sistema, la aplicación nos pide que le indiquemos cual va a ser el archivo principal sobre el que se van a realizar las operaciones. (Ilustración MU 6).

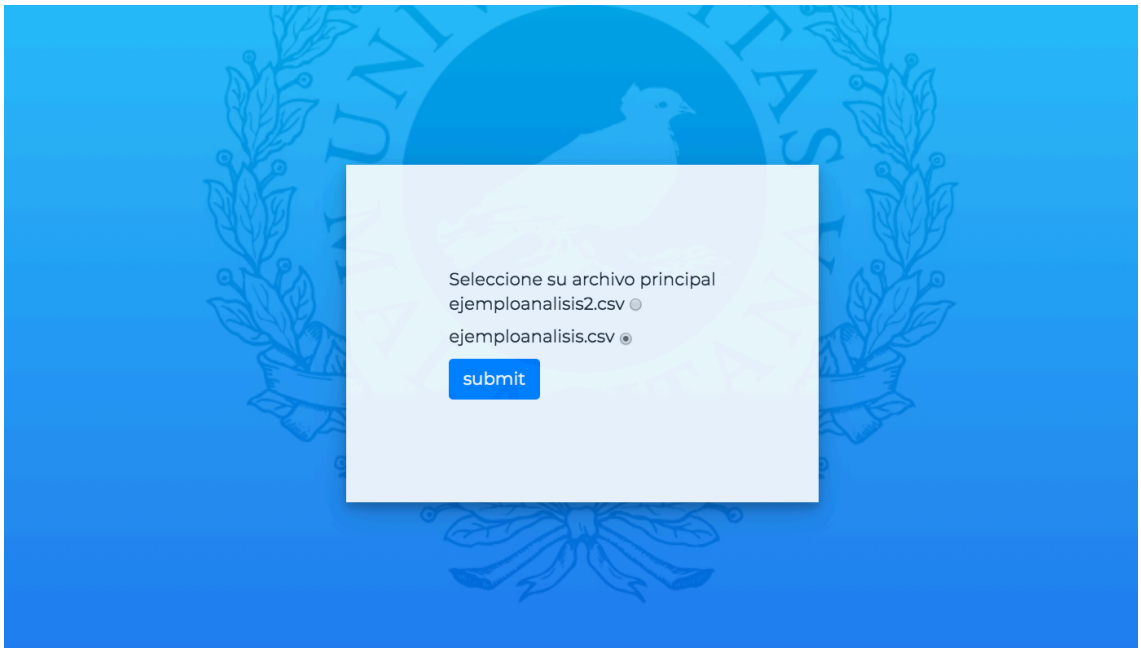


Ilustración MU 6: Seleccionar Archivo principal

2.3 Seleccionar Tipado

El siguiente paso será elegir el tipado de las cabeceras mediante un desplegable como se muestra en la ilustración MU 7. Por cada cabecera se nos da a elegir entre String, Enum, y Numeric. Es importante elegir bien pues en la representación gráfica se representará en el eje x los datos de tipo Enum y en el eje y los datos de tipo Numeric.

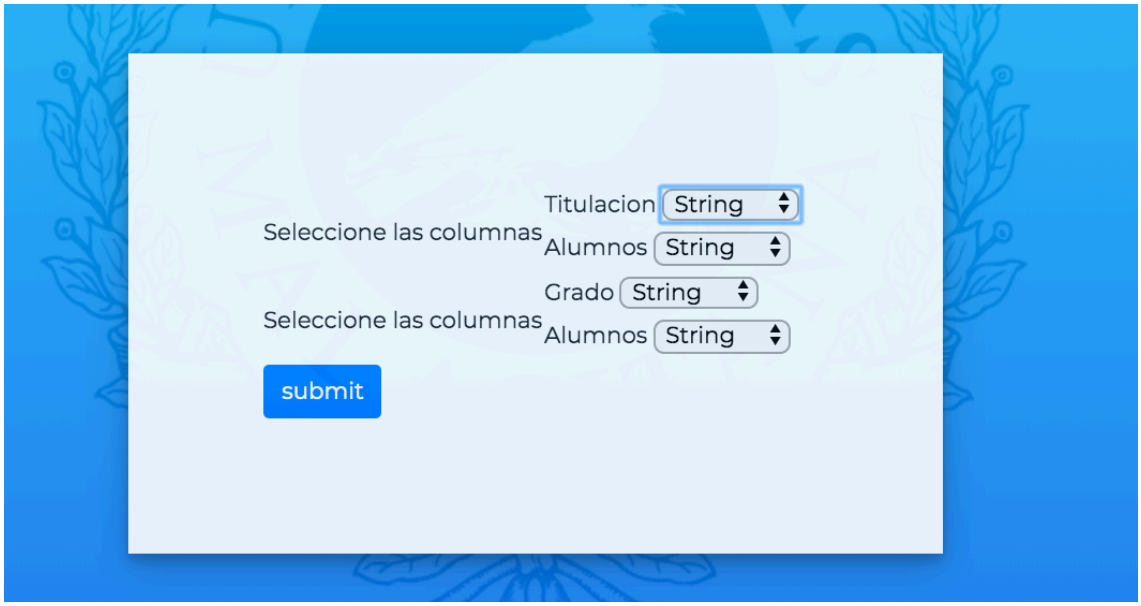


Ilustración MU 7: Seleccionar tipados

2.4 Elegir cabeceras

La siguiente opción que nos da la aplicación es la de elegir mediante checkbox las cabeceras que queremos conservar durante el proceso.

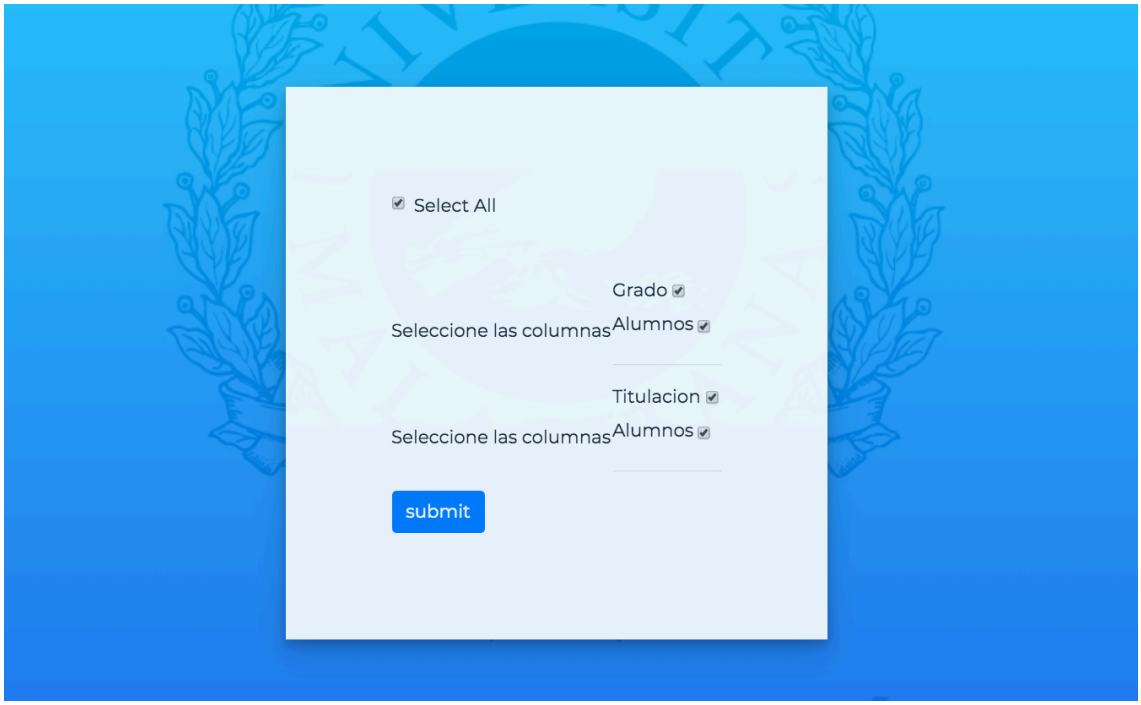


Ilustración MU 8: Seleccionar Cabeceras

2.3 Matches

Aquí viene la parte esencial, en este paso, mediante el desplegable podemos elegir la cabecera que queramos del archivo principal y combinarla con otra/s cabeceras de los otros archivos mediante el checkbox y pulsando en el botón submit. Cuando no tengamos mas cabeceras para unir el sistema nos dirigirá automáticamente a la página para ver el resultado, si consideramos que ya no queremos hacer más combinaciones también podemos pulsar el botón de finalizar y nos llevará a ver el resultado.

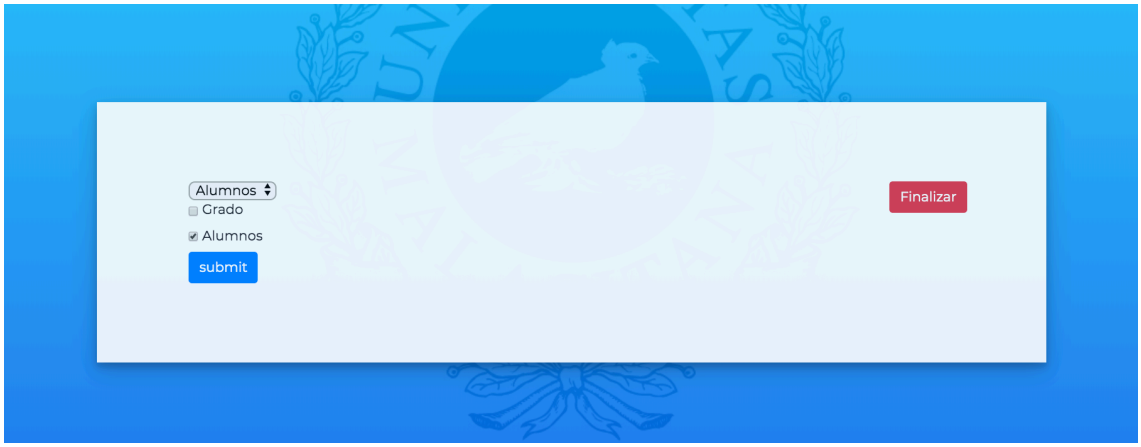


Ilustración MU 9: Combinar datos.

2.4 Ver resultado

Tras todas las operaciones podemos ver el resultado obtenido como se muestra en la ilustración MU10. En esta ventana se nos da varias opciones que serán descritas a continuación.

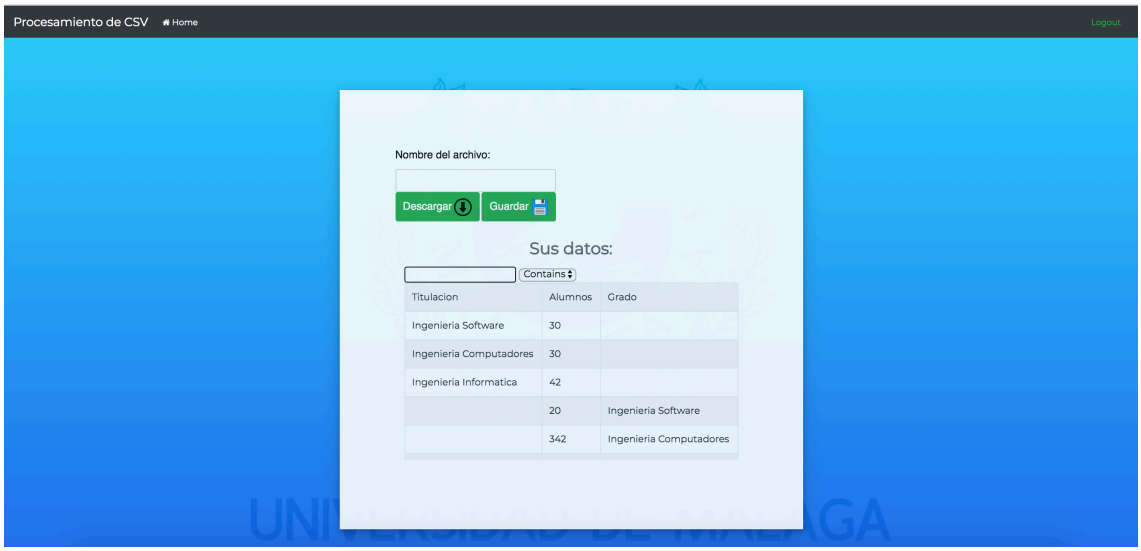


Ilustración MU 10:Ver resultado

2.4.1 Guardar archivo

Si queremos guardar el archivo en la base de datos del sistema, para poder visualizarlo en el futuro tan sólo tenemos que introducir el nombre del archivo y pulsar sobre el botón guardar.

2.4.2 Descargar archivo

Si queremos guardar el resultado obtenido en nuestro dispositivo con formato csv, la operación será similar a guardar archivo. Primero introduciremos el nombre del archivo y pulsaremos sobre el botón descargar.

2.4.3 Buscar en resultado

Mediante el filtro que tenemos encima de la tabla podemos realizar búsquedas en el resultado introduciendo que palabra queremos buscar el sistema nos irá mostrando los resultados dinámicamente. Tenemos dos opciones de búsqueda, “contains” que nos mostrará cualquier tipo de solución que contenga ese dato y la opción “equals” que solo nos mostrará los resultados que coincidan literalmente con la palabra introducida.

2.5 Mezclar Columnas

Si vemos que hay datos que se repiten de nuestro archivo resultante podemos juntar estos datos. Esta opción se dará sólo si se guarda el archivo.

Cuando queramos juntar los datos de una columna deberemos seleccionar que columna queremos y si estos datos están relacionados con valores numéricos, qué se debe hacer con estos, si sumarlos o hacer el promedio.

3.Visualizar gráficas.

Para poder visualizar el archivo antes debemos haberlo guardado en el sistema.

3.1 Ver historial de archivos.

Una vez guardado un archivo en el sistema aparecerá en la pantalla principal como vemos en la ilustración MU 11. Podremos pulsar encima de cualquier archivo y el sistema nos redirigirá a una pantalla en la que se muestre los datos de ese archivo.

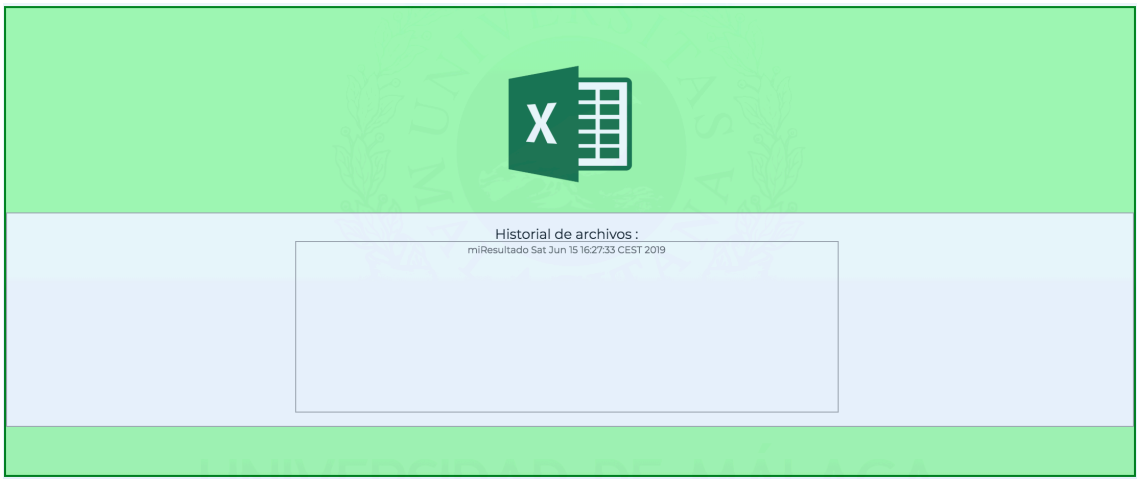


Ilustración MU 11: Historial de Archivos

3.2 Visualizar Gráficos.

Cuando entramos en un archivo, al estar ya guardado se nos muestra la opción mezclar columna (punto 2.5) y la opción de elegir que representación gráfica queremos visualizar.

Al elegir que representación queremos visualizar pulsamos sobre el botón submit y nos llevará a la representación de este.

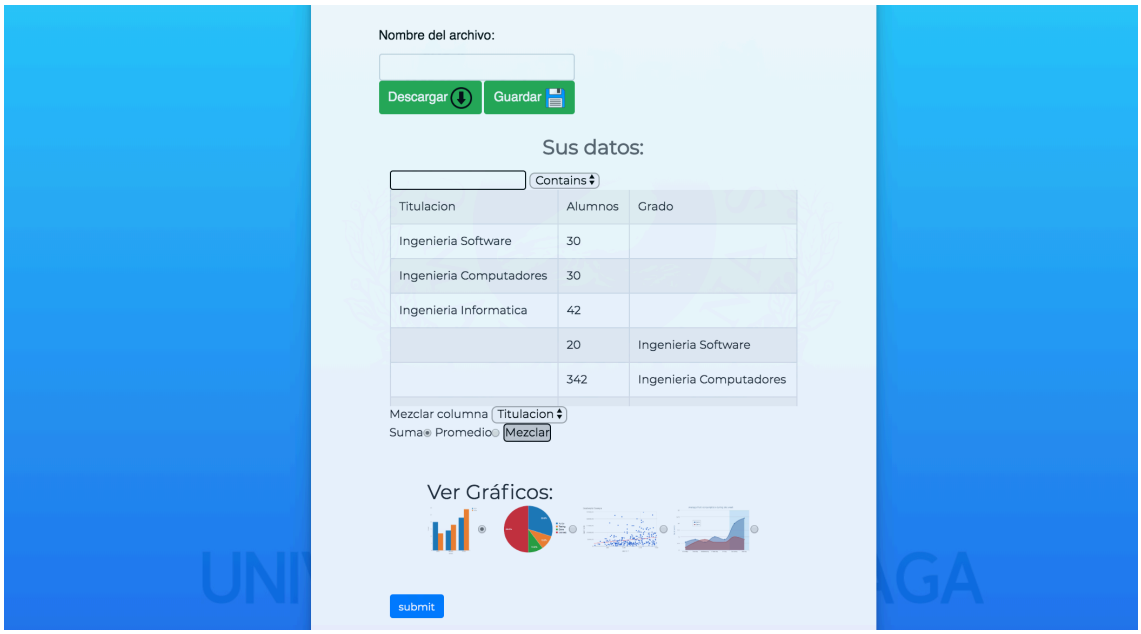


Ilustración MU 12: Archivo guardado

Será entonces cuando se nos mostrará una representación gráfica como vemos en la ilustración MU 13. En ella podremos poner el cursor sobre las columnas y nos mostrará los datos exactos como muestra la Ilustración MU 14.

Ilustración MU 13: Representación gráfica

También podremos pulsar sobre la leyenda del eje x y aparecerá o desaparecerá la columna seleccionada.

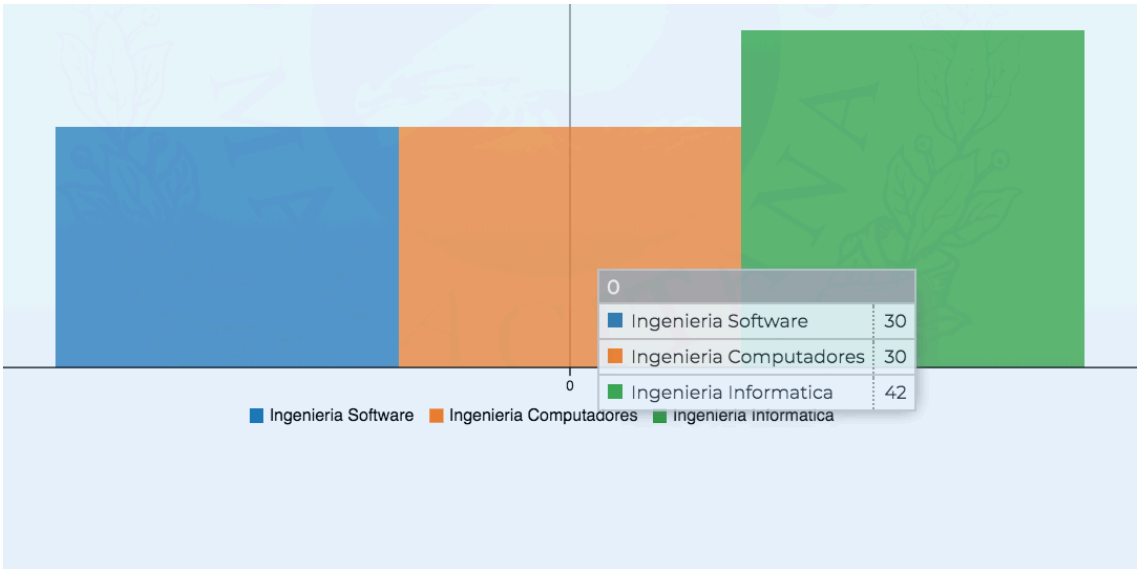


Ilustración MU 14: Datos exactos

Además mediante el seleccionable que podemos ver en la esquina inferior derecha de la ilustración 13 podemos cambiar los datos que se muestren en el eje x y en eje y.

Apéndice B

Manual de Instalación

El presente manual de instalación va destinado a usuarios tipo administrador que quieran instalar nuestra aplicación en un servidor.

1. Ejecutar la aplicación.

Para ejecutar nuestra aplicación es necesario instalar maven en nuestro equipo. En Mac OS podemos instalando mediante brew ejecutando el comando: "brew install maven". Una vez instalado, entramos a la carpeta de nuestro proyecto y dentro hacemos una instalación limpia del proyecto, para ello ejecutamos el comando: "mvn clean install". Al ejecutar este comando compilamos nuestra aplicación, y podemos ver que se crea un nuevo directorio llamado target. Este directorio es el que crea Spring por defecto como directorio de salida. El archivo que más nos interesa de esta carpeta es el llamado Procesamiento-CSV-1.0-SNAPSHOT.jar.

Para ejecutar este archivo .jar podemos hacerlo haciendo clic sobre el o si estamos en un entorno sin interfaz gráfica podemos hacer uso del comando: "java -jar target/Procesamiento-CSV-1.0-SNAPSHOT.jar" que nos sirve tanto para unix, como para Mac Os, como para Windows.

Una vez ejecutado el archivo de extensión .jar podremos acceder a la aplicación mediante la dirección http://localhost:8080.

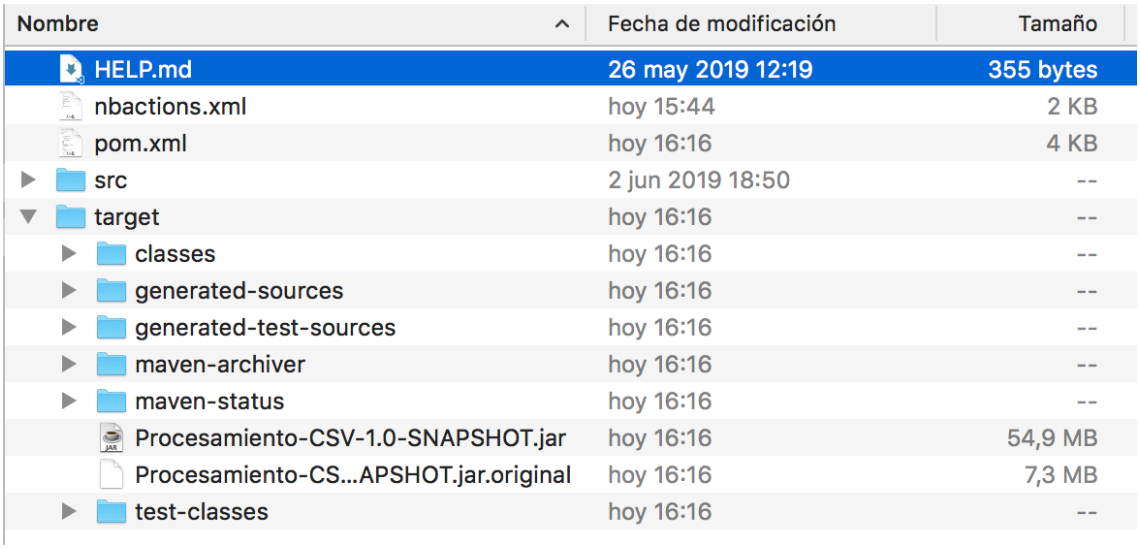


Ilustración MI 1: Directorio de la aplicación

2. Conectar con MongoDB

Nuestra aplicación necesita MongoDB como sistema de almacenamiento. Si no lo tenemos instalado, deberemos descargarlo desde https://www.mongodb.com.

La instalación es bastante sencilla basta con ejecutar el ejecutable que nos descargamos y seguir las instrucciones del instalador.

Tras la instalación nuestra aplicación se enlaza directamente con MongoDB. Es importante tener en cuenta que no sólo basta con instalar mongoDB, pues debemos iniciar un servidor mediante el comando “mongod” y crear una base de datos llamada “test”.

3. Creación de Usuario Administrador

Para poder empezar a usar nuestra aplicación es necesario crear un usuario administrador, sin embargo, el usuario “admin” lo creamos por defecto en nuestra aplicación.

Si nunca has ingresado en la aplicación antes con este usuario, el sistema te pedirá que introduzcas una nueva contraseña. Con esto finalizaría la instalación y la aplicación estaría lista para ser usada.