Universidad de Cantabria

Facultad de Ciencias

Diseño e implementación software para la mejora de la Plataforma de Tramitación W@nda

(Design and Implementation of Software for the Improvement of the W@nda Processing Platform)

Trabajo de Fin de Grado
para acceder al

Grado en Ingeniería Informática

Autor: Álvaro Peña Garcinuño
Directores: Diego García Saiz
Junio 2024

Resumen

El objetivo de este proyecto es la implementación de la nueva versión de la bandeja de entrada de la *Plataforma de Tramitación w@ndA*, perteneciente a la *Consejería de Hacienda de la Junta de Andalucía*. Una plataforma de tramitación tiene la función de permitir crear y gestionar de manera procedimentada expedientes administrativos de manera electrónica.

Debido a la antigüedad de las tecnologías utilizadas en la versión anterior, era imperiosa la necesidad de actualizar a tecnologías más modernas y eficientes. Estas tecnologías, a parte de mejorar el rendimiento del sistema, proporcionarán un diseño visual que mejorará la experiencia de usuario significativamente.

La implementación de la nueva versión conlleva desarrollo tanto de la parte de *Front-End* de la aplicación, usando tecnologías como Bootstrap, Java Script, Java Server Pages, etc; como de la parte de Back-End, implementada en Java junto con el framework Spring para la creación y gestión de microservicios.

Por último, se describirán las diferentes metodologías y aplicaciones usadas para la implementación de la metodología ágil *Scrum*, de tal manera que se pueda tener una entrega del proyecto más eficiente dividiendo el desarrollo en ciclos cortos.

Abstract

The objective of this project is the evolution and implementation of the new version of the inbox of the processing *Plataforma de Tramitación w@ndA*, belonging to the *Consejería de Hacienda de la Junta de Andalucía*. A processing platform has the function of allowing the creation and procedural management of administrative files electronically.

Due to the antiquity of the technologies used in the previous version, it was imperative to upgrade to more modern and efficient technologies. These technologies, apart from improving system performance, provide a visual design that will significantly improve the user experience.

The implementation of the new version involves the development of both the Front-End part of the application, using technologies such as Bootstrap, Java Script, Java Server Pages, etc; and the Back-End part, implemented in Java along with the Spring framework for the creation and management of microservices.

Finally, the different methodologies and applications used for the implementation of the agile Scrum methodology will be described, in order to have a more efficient project delivery by dividing the development in short cycles.

Índice

1. Introducción y objetivos

Una plataforma de tramitación debe permite crear y gestionar de manera procedimentada expedientes administrativos de manera electrónica. Su uso está muy extendido en las entidades públicas debido al gran número de trámites que se deben realizar en estas entidades.

El objetivo principal de este proyecto es la implementación y evolución tecnológica y funcional de la nueva versión de la bandeja de entrada de la *Plataforma de Tramitación w@ndA* (o PTw@nda). Concretamente, se desarrolla la versión 3.0 para un gestor de expedientes de contratación para el sector público. La plataforma tiene como objetivo principal dar soporte a la tramitación de expedientes administrativos de forma electrónica, ofreciendo las funcionalidades y servicios comunes a la tramitación de cualquier familia de procedimientos.

Este gestor de expediente estará adaptado a todo el marco legal que rodea el negocio de la contratación pública y a las necesidades propias de los tramitadores encargados de gestionar estos expedientes.

El sistema implementado está basado en las aplicaciones del universo w@ndA pertenecientes a la Consejería de Hacienda de la Junta de Andalucía añadiendo aquellas funcionalidades necesarias para adaptar un tramitador de expedientes genérico a las complejidades y exigencias del negocio de contratación pública. Uno de los mayores retos en este proyecto es la necesidad de una evolución tecnológica de unos sistemas con tecnologías anticuadas que permita, por un lado, abordar las exigencias del cliente tanto a nivel funcional como de usabilidad, así como el despliegue del sistema en un entorno *Cloud* privado de la entidad correspondiente sacando el máximo rendimiento al mismo.

Las funcionalidades a añadir pueden dividirse en dos conjuntos:

- **Funcionalidad básica**: aplicable a cualquier procedimiento
- **Funcionalidad de valor añadido**: aplicable a un gran número de ellos.

El gestor de expedientes es utilizado por un número elevado de departamentos y usuarios del cliente, debido a la funcionalidad y características aportadas a la implantación efectiva de la administración electrónica y de la tramitación de procedimientos de contratación en particular. Por tanto, la exigencia existente en el gestor es muy elevada, demandando continuamente los usuarios funcionalidades adicionales sobre el gestor, con el fin de enriquecer funcionalidad, mejorar la calidad, o mejorar la percepción de usuarios finales.

1.1. Necesidad y justificación del servicio

La implementación de la nueva versión se ha llevado a cabo debido a la necesidad de actualizar la aplicación a tecnologías más modernas y eficientes. La versión anterior de la aplicación utilizaba tecnologías obsoletas que no solo dificultaban el mantenimiento y la escalabilidad del sistema, sino que también presentaban limitaciones significativas en términos de rendimiento y usabilidad. Además, el diseño visual de la versión previa no otorgaba una experiencia de usuario adecuada, lo que afectaba negativamente la interacción de los usuarios con la aplicación.

La implementación de nuevas tecnologías no solo optimiza el rendimiento, sino que también permite la incorporación de un diseño visual más intuitivo y atractivo, mejorando así la experiencia del usuario. El proyecto busca proporcionar una base sólida para futuras mejoras y adaptaciones, asegurando que la aplicación permanezca eficiente en un entorno en constante evolución.

La evolución conlleva del uso de diferentes tecnologías de cara a implementar diferentes partes de la propia aplicación. La parte de Back-End de la aplicación está desarrollada en el lenguaje de programación Java junto con el *framework* Spring, el cual permite la gestión de microservicios, enfoque moderno del software, donde la distribución del código se hace en módulos independientes, lo que lo hace más manejable.

En la parte de Front-End, se realizará una mejora de la capa visual de la plataforma. Esta mejora incluye el uso de tecnologías como HTML5, CSS para la creación de hojas de estilo, Java Script, Bootstrap, etc; . Un ejemplo de ellas será la tecnología Java Server Pages (.jsp), Los ficheros JSP serán las páginas web que se mostrarán en el módulo, y sobre las que interactuará el usuario. Además para la creación de la bandeja de expedientes, módulo principal de la plataforma se usará la librería DataTables de Bootstrap.

Para la realización del software nuevo, habrá que seguir los estándares de codificación de la organización y verificar que este se adhiera a sus mejores prácticas.

1.2. Estructura del documento

A lo largo del documento, se entrará en detalle en los desarrollos comentados en el apartado anterior. Además, explicaremos los detalles de los siguientes puntos:

- En el capítulo 3, se incluirá una descripción del entorno de desarrollo utilizado incluyendo las aplicaciones usadas y para qué se usa cada aplicación.
- En la sección 6, se hablará sobre el desarrollo de la nueva versión de la bandeja, explicando que tecnologías se han usado y para qué tanto en el Front-End como en el Back-End de la aplicación.
- En el siguiente capítulo, se incluirá la explicación de las metodologías de trabajo empleadas para la realización del desarrollo, en este caso se ha seguido la metodología ágil *Scrum*. Comentaremos la definición de la misma así como el flujo de trabajo, reuniones, etc que hay que seguir para aplicar bien dicha metodología.
- Descripción de las aplicaciones utilizadas de cara a controlar el desarrollo de las distintas partes de la aplicación, gestión de tareas.
- Para finalizar, se incluirán los trabajos futuros que se realizarán en el proyecto.

2. Metodología de trabajo

Se ha aplicado una metodología de trabajo ágil, la cual consiste en un conjunto de técnicas aplicadas en ciclos de trabajo cortos, con el objetivo de que el proceso de entrega de un proyecto sea más eficiente. Estas metodologías se basan en el conocido como, *el Manifiesto Ágil*, Deemer [1]:

- Los individuos y las interacciones entre ellos son más importantes que las herramientas y los procesos empleados.
- Es más importante crear un producto software que funcione que escribir documentación exhaustiva.
- La colaboración con el cliente debe prevalecer sobre la negociación de contratos.
- La capacidad de respuesta ante un cambio es más importante que el seguimiento estricto de un plan.

Debido a las demandas continuas de nuevas funcionalidades por parte de los usuarios, una metodología ágil se acomoda mejor a las exigencias del proyecto.

Dentro de las metodologías ágiles, hay diferentes marcos de trabajo que permiten aplicar estos principios. En este proyecto se utiliza la metodología **Scrum**. En la figura 1, encontramos el ciclo de proceso. [1]

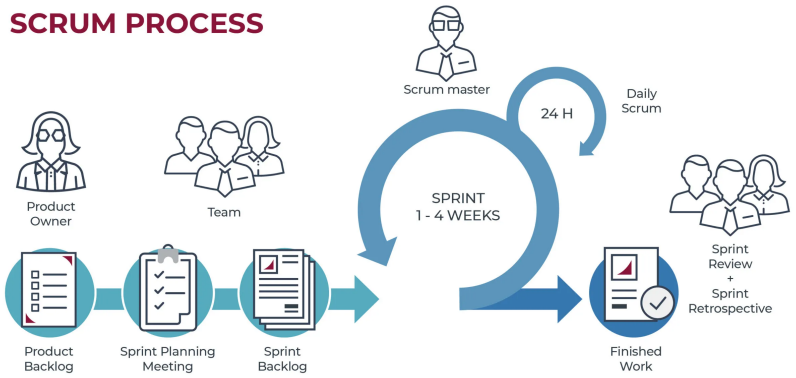

Figura 1: Ciclo de proceso Scrum

2.1. Metodología Scrum

El primer gran objetivo de Scrum, según Deemer [1], es desarrollar un producto que satisfaga las expectativas del cliente. Para ello, se crea un plan de requisitos del producto, priorizando según el valor para el cliente respecto a coste y riesgos.

[1] https://www.scl-consulting.com/wp-content/uploads/2024/03/MicrosoftTeams-image-2-300x150.jpg

Se establece un plazo de tiempo máximo, denominado *Sprint*, sobre el cual ir obteniendo *feedback* sobre el producto final a medida que se va desarrollando. Los sprints tienen siempre la misma duración (lo ideal es 3-4 semanas como máximo). El modelo de desarrollo reiterativo e incremental permite que al final de cada sprint el cliente revise las piezas que se han completado y solicite entregas (*Releases*) cuando lo desarrollado le aporta suficiente valor.

El segundo gran objetivo de Scrum es conseguir equipos de alto rendimiento motivados, que realizan planteamientos conjuntos para crear sinergias, aumentando la productividad y la creatividad, y que reflexionan regularmente para mejorar de manera continua el modo en el que se desarrolla el producto. Para esto, se utilizan técnicas y herramientas que fomentan la auto-organización y la motivación del equipo.

2.2. Roles

El conjunto de roles que participan en el desarrollo del proyecto puede dividirse en dos grupos:

- **Comprometidos:** Roles comprometidos con el proyecto y el proceso Scrum, que se encargan de realizar las tareas necesarias para que el producto obtenga el valor necesario.
  - ***Product Owner*:** Es el responsable de identificar las funcionalidades del producto y decidir cuáles de las mismas se van a desarrollar en el siguiente sprint, priorizando los elementos de más valor del negocio y menos coste. En caso de ser un producto comercial, se entiende como maximizar el retorno de inversión (ROI).
  - **Equipo de desarrollo:** Grupo de personas encargadas de construir el producto que va a utilizar el cliente. El equipo debe ser multi-funcional, teniendo todas las competencias y habilidades necesarias la entrega del producto.
  - ***Scrum master*:** Se encarga de que el grupo de trabajo aplique la metodología Scrum correctamente para ayudar a conseguir valor de trabajo. El *scrum master* no es el jefe del equipo, este guía al *Product Owner* y al equipo en el uso fructífero de Scrum.
- **Implicados:** Roles implicados en el proceso Scrum, que no son parte del proyecto, pero deben tenerse en cuenta, ya que su *feedback* es muy valioso para el resultado del producto:
  - **Cliente:** Es el encargado de contratar el desarrollo del producto, y es el máximo responsable en la organización cliente.
  - **Usuario final:** Grupo de personas que utilizarán el producto una vez terminado, demandando nuevas funcionalidades si es necesario.

2.3. Flujo de trabajo

En la figura 2, se muestra el flujo de trabajo seguido durante el ciclo de desarrollo. Como habíamos comentado anteriormente, se empieza por la planificación de los requisitos del producto. Se puede observar como la parte de planificación, ejecución y revisión/retrospectiva del sprint se puede repetir más de una vez. En los apartados sucesivos se describirán de manera más profunda.

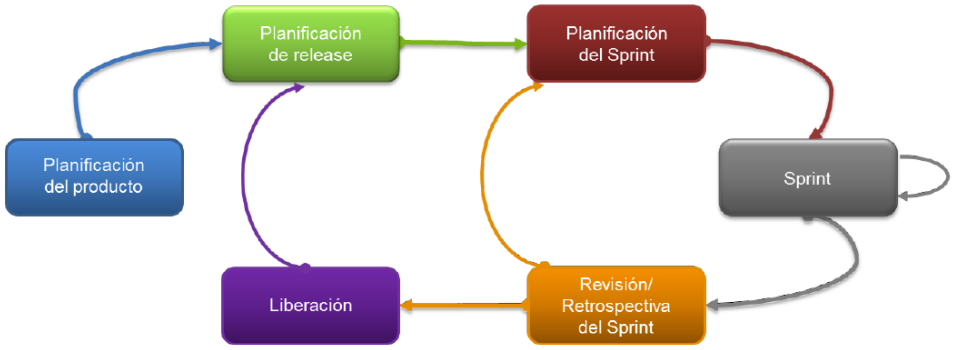

Figura 2: Flujo de trabajo

2.3.1. Planificación del producto

La primera fase del ciclo de desarrollo. Una vez se recibe la orden de trabajo con las funcionalidades a desarrollar, se realiza una estimación de cada una de ellas, de tal manera que el *Product Owner* puede priorizar las tareas a desarrollar por el equipo de desarrollo. Esto conforma la pila de tareas, denominada ***Product Backlog*** (trabajo pendiente del producto). Además, durante esta fase también se acordará la duración de los sprints y si esta puede ser flexible.

Periódicamente, se matendrán reuniones para analizar y priorizar el *product backlog*, esta tarea de revisión se denomina ***Product Backlog Refinement*** (refinamiento del trabajo pendiente del producto). Por último, se repasa el ***Impediment Backlog*** (lista de impedimentos) que el equipo de desarrollo actualiza, para delimitar el plan de acción en caso de no poder realizarse ciertas funcionalidades.

2.3.2. Planificación de release

La fase ***Release Planning*** (Planificación de liberación) tiene como objetivo seleccionar todas las funcionalidades que serán implementadas en la nueva versión a desarrollar del gestor de expedientes.

La reunión sobre la planificación del *release* tiene lugar al menos una semana antes del comienzo del sprint, en ella se analizan todas las funcionalidades que pueden tener cabida en la nueva versión a desarrollar y se seleccionan las que serán implementadas. En la reunión participan el equipo de desarrollo, el *product owner* (dueño del producto, encargado de gestionar el *product backlog*) y el responsable de negocio correspondiente.

Idealmente, en esta reunión, se creará un *product backlog* específico para la versión a implementar con las funcionalidades/tareas para la misma (***Product Backlog Release***), teniendo así dos *product backlog* distintos.

2.3.3. Planificación del Sprint

La tercera fase tiene como objetivo seleccionar el grupo de funcionalidades que se van a implementar a lo largo del siguiente sprint.

A partir del *product backlog release*, el equipo en conjunto define el alcance a realizar y las tareas que se van a realizar durante el ciclo, esto tiene lugar en la reunión de ***Sprint Planning 1*** (primera reunión de planificación del Sprint). El *product owner* debe facilitar esta tarea, indicando los objetivos y las condiciones de aceptación de cada una de las funcionalidades. Una vez seleccionadas las tareas, tanto el *product owner* como el equipo de desarrollo llevarán a cabo una estimación de cada una de las tareas a realizar en el sprint.

Durante la celebración del ***Sprint Planning 2*** (segunda reunión de planificación del Sprint) se realizará el análisis y diseño de cada característica a implementar en el sprint actual. En caso de requerirse un análisis o diseño más exhaustivo se incluirá una tarea para realizar la documentación global de análisis o diseño de la versión, dando lugar a los entregables ***Análisis Orientado a Objetos (AOO)*** y ***Diseño Orientado a Objetos (DOO)***.

2.3.4. Sprint

Durante la fase 'Sprint', el equipo de desarrollo implementa las funcionalidades definidas, obteniéndose al final del sprint una versión incremental y entregable del gestor de expedientes. A continuación, se muestran las actividades/tareas realizadas en esta fase:

1. ***Daily Scrum meeting*** (Reunión de Scrum diario): Se explicará en el apartado 2.4
2. ***Desarrollo***: El equipo de desarrollo lleva a cabo las tareas para realizar la implementación de las funcionalidades, siguiendo el análisis y diseño correspondiente.
3. ***Gestión de impedimentos***: Se añadirán los impedimentos detectados durante la implementación en el, ya mencionado anteriormente, ***Impediment Backlog***. De esta manera, se mejorará la gestión de cara a los siguientes sprints/avances.

2.3.5. Revisión/Retrospectiva del Sprint

La fase Revisión/Retrospectiva del Sprint tiene como objetivo obtener *feedback* de las nuevos desarrollos del producto, es decir, verificar si satisface las expectativas e identificar si hay que hacer ajustes. Se detalla a continuación las actividades/tareas realizadas en esta fase:

1. ***Preparación de la revisión del sprint***: Se actualiza el entorno de desarrollo con las nuevas funcionalidades implementadas a lo largo del sprint para poder mostrarlas durante la revisión.
2. ***Sprint Review***: Se consultarán los avances realizados durante el sprint. Se debe decidir si se aprueba o es necesario modificar el alcance de las funcionalidades. En caso de no ser aprobada se incluirá en el siguiente sprint para lograr la aprobación de esta.
3. ***Sprint retrospective*** (Retrospectiva sobre el sprint): Reflexionar sobre cómo fue el sprint, el progreso del proyecto, el proceso, la metodología de desarrollo y el coordinación entre los miembros del equipo, para que el siguiente sprint sea más productivo, con mayor calidad y motivación del equipo.

4. ***Revisión de calidad del código***: El objetivo de esta reunión es acordar entre todos los integrantes del equipo de desarrollo buenas prácticas que mejoren la calidad del código fuente del producto.

2.3.6. Liberación

La fase 'Liberación' tiene lugar una vez finalizada la implementación de todas las funcionalidades que conforman el ***Product Backlog Release***. Las tareas que se han de realizar para la liberación de una nueva versión serán gestionadas como un sprint cualquiera, aunque no se hayan detallado en la fase *'Sprint'*. Las tareas a realizar en la siguiente fase son:

1. **Despliegue de entorno pre-producción**.
2. **Realización de auditorías**: Se realizan las autorías sobre las siguientes temáticas: usabilidad, accesibilidad, seguridad y calidad del código fuente.
3. **Pruebas funcionales, integradas, de regresión y de rendimiento**: Con el fin de minimizar el impacto de la transición a las nuevas versiones, optimizando la calidad de los entregables se ejecutarán los ciclos completos de pruebas funcionales, integradas, de regresión y de rendimiento sobre la nueva versión construida.
4. **Pruebas de aceptación**: Este grupo de pruebas tiene la función de determinar si la nueva versión a liberar cumple con los requisitos establecidos por el cliente.
5. **Despliegue en entorno de producción**: Despliegue de la nueva versión del gestor de expedientes en el entorno de producción, previo acuerdo de cuando se llevará a cabo el mismo.

2.4. Reuniones

En esta sección, vamos a analizar cada una de las distintas reuniones que define el modelo Scrum para poder realizar el seguimiento y gestión del proyecto.

- ***Estimation Meeting*** **(Reunión de estimación de tareas)**: Se estiman y priorizan las funcionalidades a desarrollar para el producto.
- ***Sprint Plannification Meeting*** **(Planificación del Sprint)**: En esta reunión se toman como base las prioridades y necesidades de negocio del cliente, y se determina cuáles y cómo van a ser las funcionalidades que incorporará el producto tras el siguiente sprint. Tiene lugar en la fase de *'Sprint'*
- ***Daily Scrum Meeting*** **(Seguimiento diario de Scrum)**: Reunión diaria de 15 minutos en la que cada persona del equipo comenta su situación actual en el proyecto (las tareas en las que está trabajando, si se ha encontrado con algún impedimento, etc) e indica las ya terminadas, o los tiempos de trabajo que les quedan.
  En este caso particular, este tipo de reuniones se hacía cada 2 días de tal manera que en cada una de ellas haya algo destacable para comentar.
- ***Sprint Review*** **(Revisión del sprint)**: Como el propio nombre indica, tiene lugar en la fase *Revisión/Retrospectiva del Sprint*, al final del sprint. El equipo presenta al propietario del producto, clientes, usuarios, gestores, etc el incremento construido en el sprint.

- ***Sprint Restrospective* (Retrospectiva del sprint)**: Esta reunión, como la anterior, tiene lugar al finalizar un sprint. El objetivo de la misma es analizar la metodología y técnicas del trabajo del equipo, de tal manera que si se encuentra algún problema, se pueda solucionar.

2.5. Gestión de tareas

Con el objetivo de gestionar de mejor manera el proyecto, es recomendable la utilización de distintas de herramientas de seguimiento de tareas, permitiendo la interacción de todos los roles de trabajo implicados.

Para ello, se ha utilizado la herramienta de seguimiento **Jira** para la gestión de proyectos, centralizándose en la gestión de las funcionalidades, sprints, incidencias, etc

En la siguiente ilustración, figura 3, se puede observar un ejemplo de tarea en la aplicación. La tarea esta creada en el proyecto *PTWANDA 3.0*. Cada una, esta identificada con un código (representado con *PTW_######* en el *sketch*).

Desde esta pantalla, se puede modificar información importante de la tarea de manera sencilla.

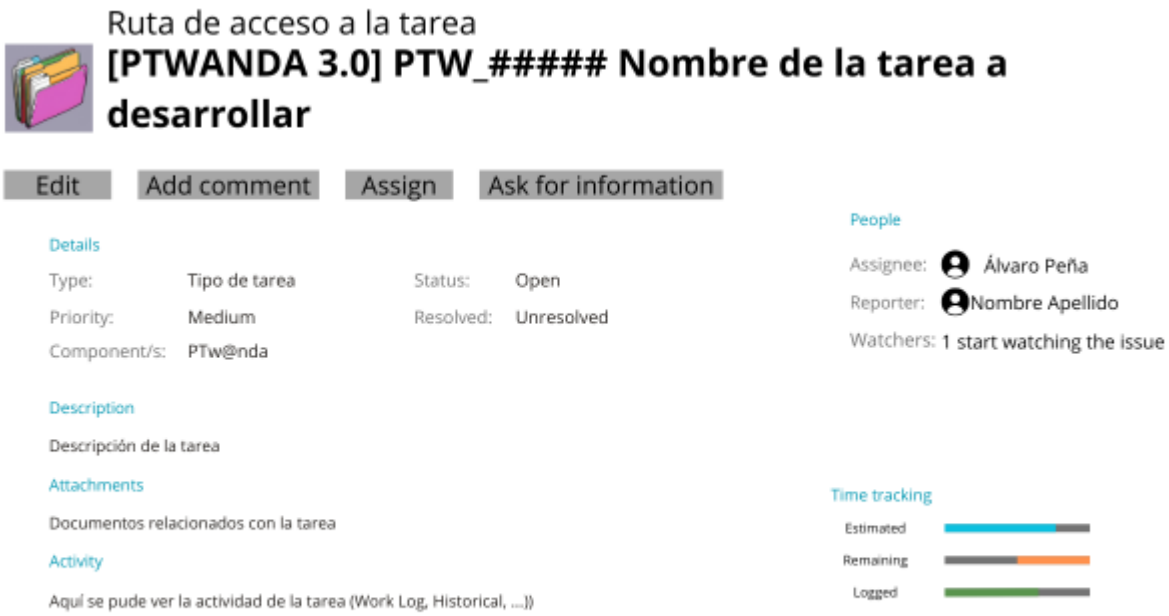

Figura 3: *Sketch* de una tarea de JIRA

2.6. Diagrama de Gantt

Un diagrama de Gantt es una herramienta de gestión de proyectos que ilustra el trabajo realizado durante un período de tiempo en relación con el tiempo previsto para el trabajo. En el diagrama, la duración de cada tarea se muestra con una línea horizontal. La longitud de dicha línea representa la cantidad de tiempo que ha hecho falta para llevar la tarea a cabo en base al eje horizontal de tiempo.

El diagrama de Gantt es ampliamente utilizado debido a las múltiples ventajas que conlleva su uso. Algunas de estas características son: visualización clara del proyecto, proporcionando una visión clara de las tareas y su secuencia; ofrecer un seguimiento del proyecto, ayudando a monitorizar el progreso del proyecto, facilitando la identificación de tareas completadas, en progreso y retrasadas; e identificación de dependencias, mostrando que tareas se han realizado y completado antes de iniciar otras.

En la siguiente figura, se puede apreciar un Diagrama de Gantt que indica las tareas que se han llevado a cabo para la realización de este trabajo de fin de grado y su duración.

Figura 4: Diagrama de Gantt sobre el proyecto.

3. Entorno de desarrollo

El entorno de desarrollo es un componente crucial en cualquier proyecto relacionado con la informática, ya que proporciona las herramientas necesarias para la creación, prueba y despliegue de software. En este capítulo, se describirán las herramientas y configuraciones empleadas para llevar a cabo el desarrollo del proyecto, asegurando un flujo de trabajo eficiente y organizado.

3.1. Eclipse

"Eclipse is an IDE for anything, nothing at all", Burnette [2]. Esta frase explica que Eclipse es un entorno de desarrollo integrado (IDE) que puede ser usado para implementar software en cualquier lenguaje.

Un entorno de desarrollo integrado (IDE) es una aplicación informática que proporciona servicios integrales para facilitar al desarrollador o programador el desarrollo de software.

Eclipse es un IDE de código abierto desarrollado originalmente por la compañía IBM y actualmente gestionada por la organización Eclipse Fundation, una entidad legal sin ánimo de lucro. Dispone de un editor de texto con un analizador sintáctico y auto-completado inteligente de código (IntelliSense). El IDE permite el uso de plugins o complementos informáticos, los cuales permiten extender las funciones de la aplicación.

En el entorno, tenemos en el centro de la pantalla el editor de texto. En la parte superior, encontramos la barra de tareas desde donde se puede configurar, ejecutar, etc En la parte izquierda, observamos los proyectos en el espacio de trabajo y el árbol de archivos en cada uno. En la parte inferior izquierda, encontramos funcionalidades como la consola, los fallos en el código, etc. [2]

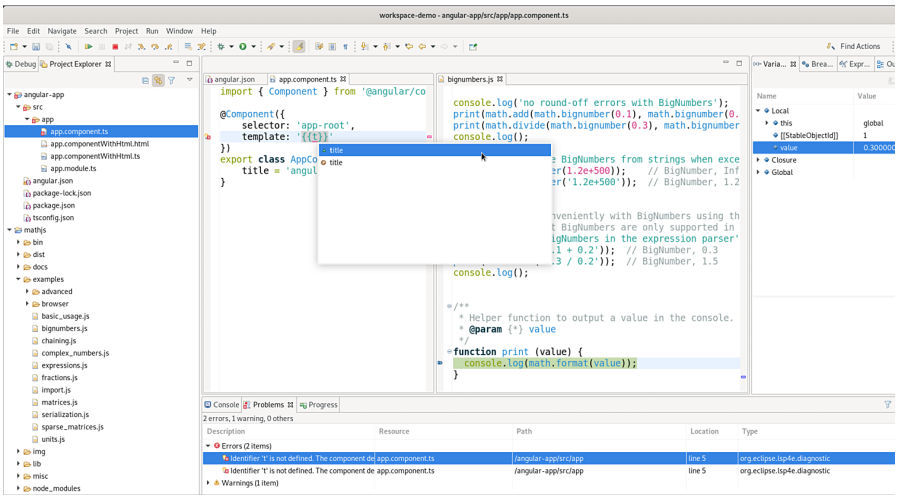

Figura 5: Interfaz de Eclipse IDE.

[2]El código mostrado en la figura 5 no están relacionadas con este proyecto. Es un ejemplo para mostrar la interfaz.

3.1.1. Plugins de Eclipse

Para facilitar el desarrollo en *Eclipse IDE*, se han añadido varios *plugins* para extender las funcionalidades del entorno. Los *plugins* más importantes son:

- **JAutodoc:** JAutodoc es un *plugin* para Eclipse que automatiza la generación de comentarios Javadoc en el código fuente de Java. Facilita la documentación del código, lo que es esencial para el mantenimiento y la comprensión a largo plazo.
- **SonarLint:** SonarLint es un *plugin* que actúa como una extensión de análisis estático de código en tiempo real. Ayuda a los desarrolladores a identificar y corregir problemas de calidad y seguridad en su código mientras lo escriben.
- **Spring-Boot:** El *plugin* de Spring Boot para Eclipse (generalmente parte del conjunto de herramientas de Spring, o Spring Tool Suite - STS) facilita el desarrollo de aplicaciones Spring Boot.
- **Enhanced Class Decompiler:** Permite descompilar archivos .class dentro de Eclipse, facilitando la inspección de código *bytecode* y la comprensión de bibliotecas externas sin código fuente disponible.
- **YAML Editor:** Proporciona soporte para la edición de archivos YAML en Eclipse. YAML es un formato de datos legible por humanos, comúnmente usado para configuración y serialización de datos.

3.2. Control de versiones

El control de versiones, también conocido como "control de código fuente", es la práctica de rastrear y gestionar los cambios en el código de software. Los sistemas de control de versiones son herramientas de software que ayudan a los equipos de desarrollo a gestionar los cambios a lo largo del tiempo.

Para el control de versiones del desarrollo se han usado dos aplicaciones compatibles entre sí. La aplicación principal que se encarga de realizar esta funcionalidad es *Git*.

3.2.1. Git

Git es un software de control de versiones diseñado por *Linus Torvalds*, pensando en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Algunas de las características principales de *Git* son:

- **Repositorio local:** Git es un sistema de control de versiones distribuido, lo que implica que cada clon local del proyecto actúa como un repositorio completo de control de versiones. Estos repositorios locales plenamente funcionales permiten trabajar sin conexión o de forma remota con facilidad. Los desarrolladores confirman su trabajo localmente y, a continuación, sincronizan la copia del repositorio con la del servidor.
- **Ramas (*Branches*):** Git permite la creación de ramas para desarrollar nuevas funcionalidades, corregir errores o experimentar, sin afectar la versión principal del código, el cual se suele

guardar en la rama principal. Una vez finaliza el desarrollo en el resto de ramas se combina con la principal.
Esto facilita el trabajo colaborativo y permite el desarrollo de diferentes características en paralelo.

- **Commits:** Cada cambio que se realiza en el código fuente se guarda como un commit, proporcionando un registro detallado de qué cambios se hicieron, cuándo y por quién. Esto es fundamental para el seguimiento y la reversión de cambios si es necesario.

3.2.2. Sourcetree

Como complemento al uso de la herramienta *Git*, se utiliza *Sourcetree*. Sourcetree es una interfaz gráfica de usuario (GUI) para Git que simplifica muchas de las tareas que los desarrolladores realizan a través de la línea de comandos.

En la figura 6, se puede observar la pantalla principal de Sourcetree. En la parte izquierda, aparece la lista de ramas del proyecto. En la parte central, el registro de commits que se han hecho en la rama seleccionada. Desde la barra superior se pueden realizar las operaciones básicas (pull, push, etc) y la configuración de la propia aplicación. Por último, en la parte inferior, aparecen los cambios que se han realizado en el commit seleccionado del registro. [3]

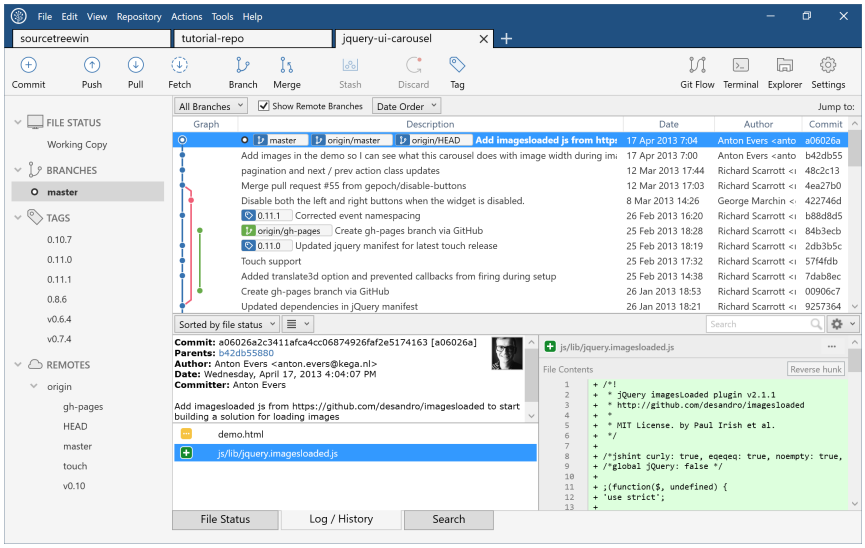

Figura 6: Pantalla principal de Sourcetree

Gracias a la interfaz gráfica que contiene la aplicación, se facilitan varios aspectos del control de versiones:

- Posibilidad de hacer las acciones básicas de *Git* (commit, pull, push, fetch, etc) en pocos clicks. Incluye la posibilidad de hacer commit solo de algunos de los ficheros modificados.

[3] La descripción de proyecto que aparece en la imagen 6 no está relacionada con este proyecto. Es un ejemplo para mostrar la interfaz.

- Visualización clara de las diferentes ramas del proyecto. Al hacer un click en una rama, aparece el registro de commits de la misma, proporcionando una visión clara de los cambios realizados y facilitando el seguimiento del desarrollo del proyecto.
- **Resolución de conflictos:** Sourcetree ofrece herramientas visuales para resolver conflictos de fusión, lo que simplifica este proceso y reduce la posibilidad de errores.

3.3. FortiClient VPN

En este apartado se va a hablar de una aplicación que, a pesar de tener una funcionalidad muy simple, es de vital importancia para el desarrollo del proyecto.

FortiClient VPN es una aplicación que permite a los usuarios conectarse de manera segura a una VPN o Virtual Private Network (Red Privada Virtual). Proporciona un cifrado robusto y autenticación para proteger los datos durante su transmisión.

Gracias a la aplicación, se puede acceder a la red privada donde se encuentran los servidores donde con recursos necesarios para el despliegue de la aplicación, como puede ser la base de datos, a pesar de la distancia. Esto es esencial para poder desarrollar el proyecto desde Santander cuando la red privada se encuentra en Sevilla, a cientos de kilómetros de distancia.

3.4. Oracle SQL Developer

Para la gestión de la base de datos, se utiliza la herramienta *Oracle SQL Developer*. A pesar de que el desarrollo de la base de datos no está directamente relacionado con el objeto de este trabajo de fin de grado, la base de datos es una parte importante, pues es el recurso donde se encuentran definidos los expedientes del gestor. Por este mismo motivo, la aplicación ha sido necesaria para las pruebas de la bandeja y comprobar que todo estuviese de manera correcta.

Oracle SQL Developer es una herramienta gráfica gratuita desarrollada por Oracle para simplificar el desarrollo y la gestión de bases de datos Oracle.

Esta aplicación cuenta con una interfaz de usuario intuitiva y gráfica que permite a los usuarios realizar tareas de administración de la base de datos sin la necesidad de utilizar la línea de comandos. La interfaz está diseñada para ser accesible incluso para aquellos con conocimientos limitados de SQL, facilitando la gestión eficiente de la base de datos.

Algunas de las tareas que se pueden realizar desde la interfaz de usuario incluyen la gestión de los objetos básicos de la base de datos, como la creación, modificación y eliminación de los elementos de la base de datos (tablas, índices, etc). Además, los usuarios pueden gestionar vistas, procedimientos almacenados y *triggers*, así como realizar consultas complejas de manera visual.

La aplicación también proporciona herramientas para importar y exportar datos, realizar copias de seguridad y restauraciones, y monitorizar el rendimiento de la base de datos. Estas funcionalidades permiten a los administradores de bases de datos mantener un control completo, optimizando tanto la seguridad como la eficiencia operativa.

En la figura 7 se puede apreciar los elementos principales de la interfaz. En el centro de la imagen se puede ver el elemento seleccionado, en este caso la tabla *BOOKS*. En el lado izquierdo, se puede apreciar las bases de datos con sus elementos mostrados en formato árbol. En el ejemplo, la base de datos *xe_local* está desplegada y se pueden observar las diferentes carpetas para los elementos (*Tables, Views, Indexes, etc*) con la carpeta *Tables* desplegada. En la parte superior, se encuentra la barra de herramientas con las opciones de gestión de la propia aplicación (*File, Edit, View*).[4]

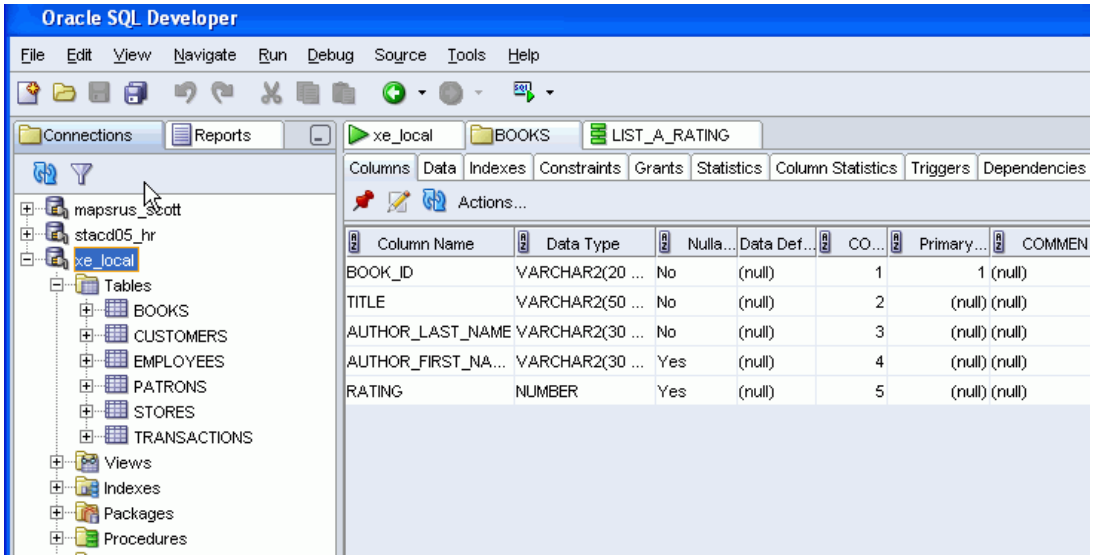

Figura 7: Pantalla principal de Oracle SQL Developer

3.5. WildFly

WildFly (formalmente WildFly Application Server), anteriormente conocido como JBoss AS, es un servidor de aplicaciones de código abierto escrito en Java. Es una plataforma robusta y flexible que se utiliza para desarrollar y ejecutar aplicaciones empresariales basadas en Java. JBoss fue uno de los primeros servidores de aplicaciones de código abierto en ganar popularidad en la comunidad de desarrolladores.

Wildfly es comúnmente usado en aplicaciones empresariales que requieren alta disponibilidad, escalabilidad y soporte para transacciones distribuidas. Debido a sus características, su soporte para Java y otras tecnologías modernas, WildFly es una opción viable para desarrollar y desplegar arquitecturas de microservicios. Además, proporciona un entorno robusto para las aplicaciones web basadas en Java, ofreciendo servicios web, JSP, etc

Más concretamente, se ha usado la versión WildFly 20, la cual incluye mejoras en la seguridad, entre otras.

[4]La base de datos que aparece en la imagen 7 no está relacionada con este proyecto. Es un ejemplo para mostrar la interfaz.

3.6. Maven

Apache Maven es un software para la gestión y comprensión de proyectos ampliamente utilizado en el desarrollo Software, especialmente en los proyectos Java. Desarrollada por Apache, Maven proporciona un marco de trabajo estándar para gestionar el ciclo de vida del proyecto, incluyendo la compilación, la prueba, la documentación y el despliegue.

El *Project Model Object* (POM), es la pieza fundamental de Maven. Es un fichero XML que contiene información sobre el proyecto y los detalles de la configuración de Maven. Algunas de las configuraciones que se pueden especificar en el POM son: la dependencia de proyectos, los *plugins* que se pueden ejecutar, los perfiles de desarrollo, etc. También se puede especificar información como la versión del proyecto, la descripción, etc. En la imagen 8 se muestra un ejemplo de POM. [5]

Figura 8: Ejemplo de declaración de dependencia en un POM

3.6.1. Maven 3.9.3

En el proyecto se ha utilizado la versión 3.9.3 de Maven, está versión incluye varias mejoras importantes que ayudan a mejorar el proyecto. Aparte de mejorar el rendimiento y estabilidad de la herramienta, esta versión incluye una mejora en la gestión de las dependencias, reduciendo los conflictos en proyectos complejos. Además incluye una actualización de varios *plugins* y dependencias, lo que proporciona acceso a nuevas funcionalidades de las mismas. Por último, esta versión mantiene la compatibilidad con versiones anteriores, permitiendo una transición en los proyectos ya existentes.

[5] La declaración que aparece en la imagen 8 no está relacionada con este proyecto. Es solo un ejemplo.

4. Requisitos y casos de uso

En esta sección vamos a ver los requisitos funcionales y no funcionales de la bandeja de expedientes, así como los casos de uso.

4.1. Requisitos funcionales

Los requisitos funcionales representan las funcionalidades o características principales que debe poseer un sistema de software para cumplir con el propósito previsto. En términos más simples, estos requisitos especifican lo que debe hacer el sistema. Los encontramos en la tabla 1.

Código	**Requisito**
RF1	La bandeja debe mostrar la búsqueda de resultados indicando el número de resultados obtenidos.
RF2	Para realizar la búsqueda, se debe poder filtrar uno o más campos, así como tener la posibilidad de realizar una búsqueda de texto.
RF3	Desde la bandeja se debe poder acceder al *Escritorio de Tramitación* del expediente seleccionado.
RF4	Desde la bandeja se debe acceder a la ventana para dar un expediente de alta.
RF5	El resultado de la búsqueda se debe poder exportar en tres formatos: CSV, XML y PDF.
RF6	Se debe poder configurar qué datos se muestran en la bandeja.

Cuadro 1: Requisitos funcionales del sistema

4.2. Requisitos no funcionales

En cuanto a los requisitos no funcionales, complementan los requisitos funcionales al especificar cómo un sistema de software debe realizar ciertas funciones. Definen las cualidades, características y limitaciones del sistema más que sus características específicas. Los encontramos en la tabla 2.

Tipo	**Requisito**
Seguridad	El sistema debe cumplir con estándares de seguridad y protocolos de cifrado específicos de la industria.
Seguridad	Asegurar que los datos sensibles están protegidos mediante cifrado y otras medidas de seguridad cumpliendo la ley de protección de datos.
Compatibilidad	El sistema debe ser compatible con los principales navegadores web.
Dispositivos	El sistema debe ser accesible desde diferentes dispositivos, incluyendo computadoras de escritorio, tabletas y teléfonos inteligentes.
Usabilidad	El sistema debe proporcionar mensajes de error que sean informativos y orientados a usuario final.
Disponibilidad	El sistema debe tener una disponibilidad del 99,99 % de las veces en que un usuario intente acceder.

Cuadro 2: Requisitos no funcionales del sistema

4.3. Casos de uso

En la siguiente figura, están descritos los casos de uso de la bandeja de expedientes. Como actor nos encontramos al usuario tramitador con las posibles secuencia de acciones que puede llevar a cabo una vez acceda al sistema.

Cabe destacar que en el caso llamado "Realizar búsqueda personalizada", una vez se realiza esa acción tendríamos otra vez las 4 opciones para realizar "Gestionar los expedientes", "Realizar otra búsqueda personalizada", "Tramitar un expediente" y "Exportar resultado"). Esto no ha sido incluido debido a la legibilidad de la letra en la imagen. También destacar que "Gestionar los expedientes" hace referencia a crearlos, eliminarlos, modificar sus datos, etc.

En la figura 9, encontramos los casos de uso.

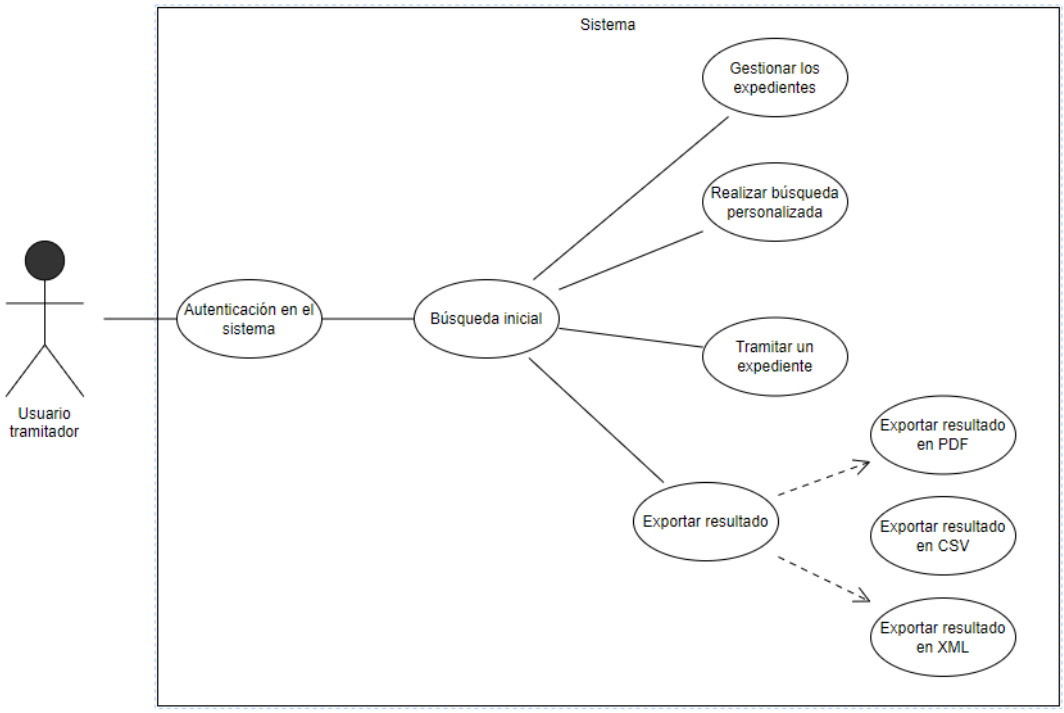

Figura 9: Casos de uso

5. Migración de motor de búsqueda

Para la nueva versión del gestor, una de las principales modificaciones que se ha hecho ha sido la migración del motor de búsqueda. En las versiones anteriores se usaba el motor de búsqueda *SolR*.

5.1. SolR

SolR (pronunciado "Solar"), es un motor de búsqueda de código abierto escrito en Java. Ha sido ampliamente utilizada debido a la potente capacidad de búsqueda y su flexibilidad para indexar y buscar datos. SolR está diseñado para impulsar potentes aplicaciones analíticas o realizar una recuperación de información (*Information retrieval*) que contienen datos no estructurados, semi-estructurados o una mezcla de no estructurados y estructurados.
Las características principales de SolR son:

- **Buscador de texto completo:** SolR funciona como un servidor independiente de búsqueda de texto completo. Utiliza la biblioteca de búsqueda escrita en Java *Apache Lucene* para la indexación y búsqueda de texto completo.
  Esto permite que el motor de búsqueda soporte búsquedas de texto con características avanzadas como resaltado y fragmentación.
- **Búsqueda por facetas:** Permite la búsqueda facetada para explorar grandes volúmenes de datos. Este tipo de búsqueda es una técnica de búsqueda paramétrica. Para agilizar la búsuqeda se basa en una *clasificación facetada*, lo que permite permite acceder a los datos de diferentes maneras.
- **Escalabilidad:** Se puede realizar un escalado horizontal añadiendo más números al *cluster*, lo que aumenta la capacidad de procesamiento y almacenamiento, y/o mediante balanceo de carga, repartiendo el peso de las solicitudes entre todos los nodos sin sobrecargar ninguno. Esto permite mejorar el rendimiento de las búsquedas.
- **APIs RESTful:** SolR proporciona una interfaz de programación de aplicaciones (API) RESTful que permite interactuar con el motor de búsqueda mediante solicitudes HTTP estándar.Los datos pueden ser enviados y recibidos en formatos comunes como JSON y XML, facilitando la interoperabilidad con otros sistemas.

A pesar de ser un motor de búsqueda muy popular, tienes varias limitaciones que han desbalanceado la balanza a favor del nuevo motor de búsqueda que se va a utilizar. Los defectos más importantes son:

- **Complejidad en la configuración y administración:** La configuración de SolR puede ser muy compleja a la hora de implementar grandes sistemas.
- **Escalabilidad limitada:** A pesar de ser escalable, es menos eficiente y más complicada en comparación con otros motores de búsqueda.
- **Soporte a la integración y complementos:** El ecosistema de complementos y herramientas es más limitado que el de otros motores. Sobre todo, en el ámbito de herramientas de visualización de datos.

- **Curva de aprendizaje:** La curva de aprendizaje puede ser muy empinada, sobre todo a para desarrolladores inexpertos en este ámbito.

En la figura 10, se observa la interfaz de *queries* de *SolR*. En el ejemplo, se usa el formato de texto JSON para realizar una *query* con el texto "genre:Fantasy" donde la llave es "genre" y el valor es "Fantasy". En el resultado se muestra la información de las películas que contienen el género (genre) Fantasía (Fantasy).[6]

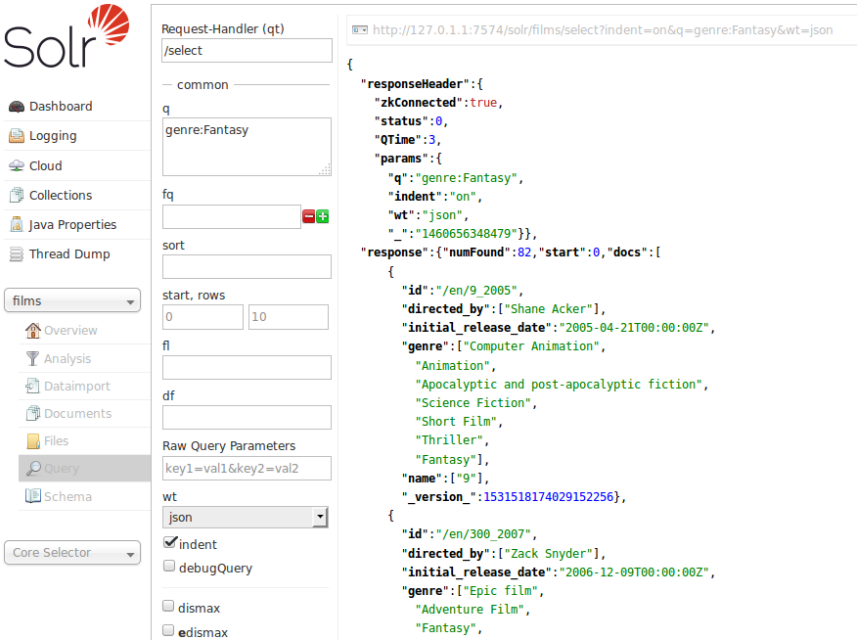

Figura 10: Interfaz de *queries* de SolR.

SolR fue el motor de búsqueda más popular entre la comunidad de desarrolladores durante muchos años. Este fue superado en 2016 por el motor de búsqueda que se va a implementar en la nueva versión de la bandeja de expedientes, *Elasticsearch* [16]

[6]Los datos que se muestran en la imagen no están relacionados con el proyecto de fin de grado.

5.2. Elasticsearch

Elasticsearch es un motor de búsqueda y análisis distribuido de código abierto basado en la biblioteca Apache Lucene, al igual que *SolR*. Inicialmente lanzado en 2010 por Elastic, Elasticsearch fue diseñado como solución Java distribuida para incorporar la funcionalidad de búsqueda de texto completo a documentos JSON sin esquema en múltiples tipos de base de datos. Se ha convertido en una opción popular debido a su facilidad de uso, escalabilidad y su integración nativa con herramientas de visualización como Kibana.

Elasticsearch proporciona prestaciones de búsqueda muy escalables, ya que es capaz de recopilar tipos de datos no estructurados de diversos orígenes y almacenarlos en formatos especializados para realizar búsquedas optimizadas basadas en el lenguaje.

Las principales ventajas que ofrece Elasticsearch:

- **Escabalidad horizontal:** Al igual que con SolR, se puede realizar un escalado horizontal. Elasticsearch facilita la agregación de más capacidad y fiabilidad a sus nodos y clústeres, además se puede realizar la escalabilidad sin necesidad de interrumpir el servicio. También incluye la funcionalidad CCR (*Cross-Cluster Replication*) que permite replicar información entre los clústeres. Esto le da a su organización la capacidad de utilizar todas las funciones de Elasticsearch, al tiempo que reduce la latencia para los usuarios y garantiza una alta disponibilidad de los servicios.
- **Compatible con múltiples lenguajes de programación:** Como solución de código abierto, Elasticsearch es extremadamente adaptable y accesible para los desarrolladores y admite varios lenguajes de codificación, incluidos Java, Python, .NET, PHP o Plus, entre otros.
- **Función de Autocompletar:** Otra gran característica de Elasticsearch es su funcionalidad de autocompletar. Su diseño intuitivo permite a los usuarios mejorar su búsqueda y encontrar resultados relevantes a medida que escriben sus consultas.
- **Soporte a la integración y complementos:** A diferencia de SolR, Elasticsearch tiene una gran compatibilidad con una amplia variedad de complementos e integraciones. Esto incluye extensiones de API, herramientas de alerta, complementos de seguridad, integraciones de recuperación de datos y más. Esta facilidad permite que el motor de búsqueda sea fácilmente adaptable a las necesidades del proyecto.

Debido a los puntos que se han comentado, se decidió migrar el motor de búsqueda de SolR a Elasticsearch. La velocidad y la flexibilidad de Elasticsearch lo convierten en el motor ideal para casos de uso con límites de tiempo. Además, para el uso de Elasticsearch, se ha incluido la herramienta *Kibana*.

5.3. Kibana

Kibana es una herramienta de visualización de datos que se integra de forma nativa con Elasticsearch. Permite a los usuarios explorar, visualizar y analizar los datos almacenados en Elasticsearch mediante una interfaz gráfica intuitiva.

"Desde exploración de datos hasta encontrar información y compartir resultados, Kibana te brinda la capacidad de comprender tus datos rápido, detectar tendencias y anomalías de un vistazo, y enrutar los hallazgos al equipo correcto al instante.", [6].

Kibana permite crear y compartir *dashboards* personalizados que se actualizan en tiempo real, ofreciendo una gran variedad de opciones de visualización de los datos. Ofrece el uso de filtros y consultas ad-hoc para facilitar la exploración de grandes volúmenes de datos. Otra funcionalidad importantes es la de poder configurar alertas basadas en condiciones específicas y monitorear el rendimiento de los sistemas en tiempo real.

A las funcionalidades comentadas previamente, se le añaden varias ventajas que ofrece el uso de la herramienta. La principal es la integración sencilla de la misma, pues funciona directamente con *Elasticsearch* eliminando las configuraciones adicionales. A esto se le suma la usabilidad, la interfaz gráfica es fácil de usar, permitiendo que tanto desarrolladores como analistas de datos puedan trabajar con ella sin necesidad de conocimientos técnicos avanzados.

5.4. Conclusión

La migración de SolR a Elasticsearch, junto con la integración de Kibana, ha proporcionado mejoras significativas en términos de rendimiento, escalabilidad, facilidad de uso y capacidades de análisis de datos. Esta transición ha permitido optimizar las búsquedas, ofreciendo una mejora en la experiencia de usuario. Para finalizar el capítulo, en las siguientes figuras se muestran una comparativa de los motores gráficos para un ejemplo de 1000 búsquedas. En el eje vertical se muestran los segundos, y en el eje horizontal el número de búsquedas hasta ese segundo. En cuánto a los colores en azul se muestra *Elasticsearch* y en rojo *SolR*.

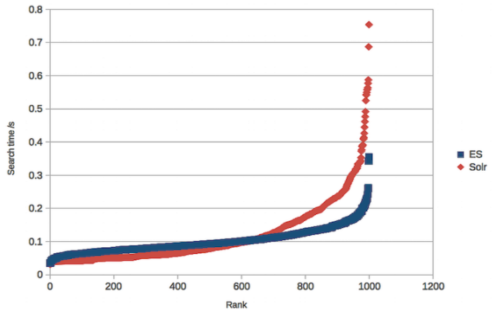

Figura 11: Primera comparación de los motores de búsqueda.

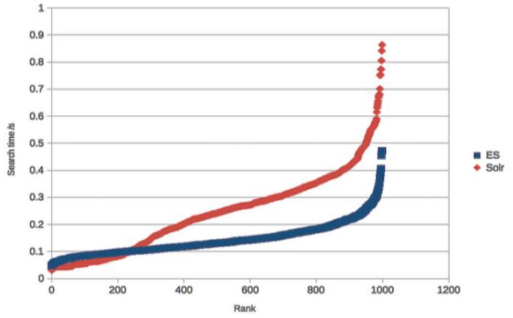

Figura 12: Segunda comparación de los motores de búsqueda.

6. Diseño y desarrollo

En este capítulo, se va a comentar sobre el desarrollo que se ha llevado a cabo para la creación de la bandeja de expedientes de la nueva versión, explicando en todo momento las tecnologías que se usan. Debido a que el código es propiedad de la empresa, no se incluirá ningún fragmento literal del código.

El tramitador de expedientes utiliza el patrón de arquitectura software MVC (Modelo-Vista-Controlador) para estructurar y organizar el código de manera eficiente. En esta arquitectura, el Modelo representa la lógica de negocio y el manejo de datos, asegurando que la información se gestione de forma coherente y segura. La Vista se encarga de la presentación y la interfaz de usuario, proporcionando una experiencia intuitiva y accesible. El Controlador actúa como intermediario entre el modelo y la vista, gestionando las interacciones del usuario y actualizando la vista conforme a los cambios en el modelo.

En la figura 13 se observa el proceso MVC. [7]

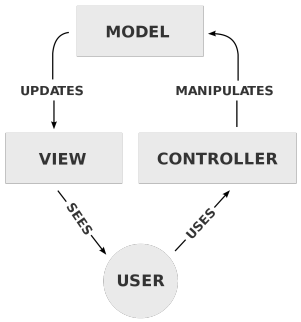

Figura 13: Proceso de arquitectura MVC

Para empezar a describir la implementación de la nueva versión de la bandeja de expedientes, es esencial empezar por la descripción del *Back-End* de la aplicación. El *Back-End* juega un papel crucial, ya que se encarga de gestionar toda la lógica del servidor, incluyendo la obtención, el procesamiento y la entrega de los datos necesarios para el funcionamiento de la aplicación. La función principal es la obtención de los datos del motor de búsqueda.

Para realizar esta tarea, el *Back-End* se encargará de gestionar las peticiones HTTP para conseguir los resultados de la búsqueda. Además de esto, implementará funcionalidades cruciales como la validación de datos, garantizando la integridad de los datos. Una vez que el *Back-End* ha obtenido y procesado los datos, este se encargará de enviarlos al *Front-End*. El *Front-End*, mediante distintas tecnologías que se comentarán más adelante, se encargará de presentar los datos de manera correcta para los usuarios, gestionando la interfaz de usuario para crear una experiencia de usuario adecuada.

[7] https://upload.wikimedia.org/wikipedia/commons/a/a0/MVC-Process.svg

6.1. *Back-End*

Una vez se ha elegido el motor de búsqueda que se va a utilizar y cómo funciona, nos centramos en la fase de obtener los datos del mismo. El *Back-End* de la aplicación está desarrollado en el lenguaje de programación Java.

A continuación, se encuentra el diagrama de clases UML. En este diagrama se encuentran únicamente las clases que han sido importantes a la hora de desarrollar la nueva bandeja de expedientes.

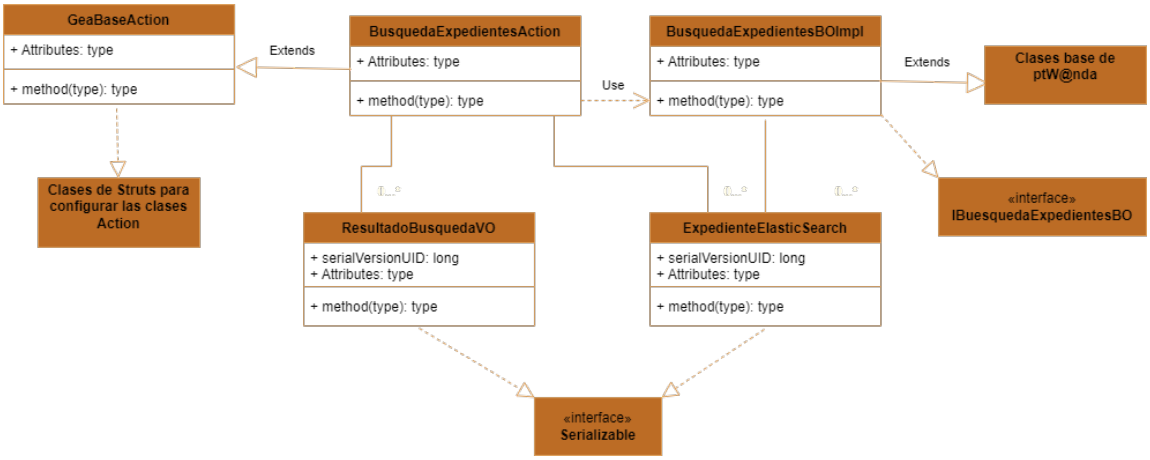

Figura 14: Diagrama de clase de *Back-End*

Una vez están definidos todos los atributos necesarios, se empiezan a desarrollar los distintos casos de uso. El primer método a desarrollar es la obtención de los KPIs y sus respectivos expedientes. Los KPI (Key Performance Indicator) también conocido como indicador clave de rendimiento, son métricas que definen el progreso de un proceso en cuanto al objetivo. En la bandeja de entrada, se muestran representados al principio de la ventana en las denominadas "tarjetas".

Como se aprecia en la figura 15, cada tarjeta indica el número de expedientes pendientes de tramitar, de subsanación o de firmar para el usuario que ha iniciado sesión. Además, la tarjeta de la derecha muestra los expedientes que faltan por tramitar de todos los usuarios, sin filtrar por el que ha iniciado sesión. Se puede hacer click en cada una de estas tarjetas para que se muestren los expedientes en dicho estado.

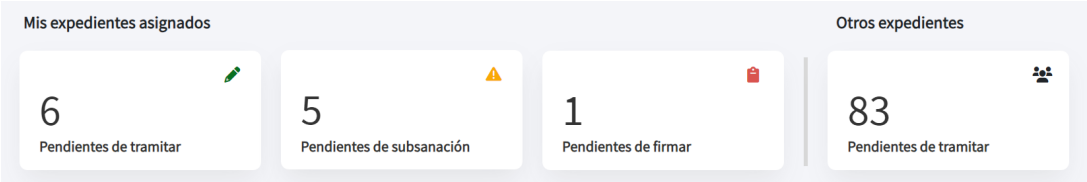

Figura 15: Tarjetas con los KPIs

Por defecto, cuando se entra a la bandeja de expedientes, mediante un método se obtiene cada número de los KPIs realizando una búsqueda para cada fase, a parte de obtener los números, se muestran todos los expedientes asignados al usuario, independientemente de la fase en la que se encuentren los mismos. En el caso de la figura 15, serían 12 resultados totales.

El siguiente método a comentar, gestiona el resultado del otro caso en el que el usuario pueda realizar una búsqueda. Este se refiere a que, mediante diferentes filtros y/o directamente un buscador genérico, en el que el usuario introduce una cadena de caracteres, el usuario realiza una búsqueda avanzada de expedientes.

Como se puede apreciar en la figura 16, la bandeja cuenta con la opción del buscador genérico y diferentes filtros sobre el procedimiento al que pertenece el expediente, la fecha de alta (dos diferentes para seleccionar desde/hasta), filtrar por la fase en la que se encuentra el expediente, por la razón de interés o por los usuarios interesados.

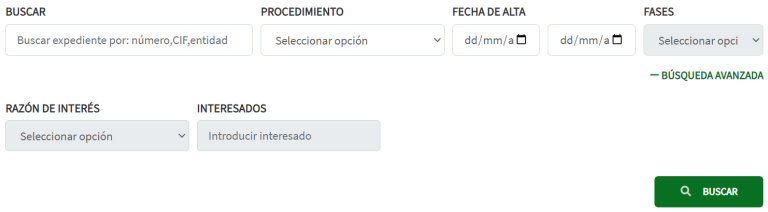

Figura 16: Buscador genérico y filtros de la bandeja de expedientes.

Cabe destacar que los dos métodos que se han comentado hasta ahora, los cuales se encuentran en la clase *Action*, no se encargan de realizar directamente la consulta al motor de búsqueda sino de preparar el entorno para realizar la búsqueda. Posteriormente, llaman a un método en la clase "BusquedaExpedientesBOImpl" que se encarga de realizar la búsqueda (se explicará a continuación). Las llamadas a este método tendrán atributos diferentes dependiendo del caso.

Además de los métodos ya mencionados anteriormente, se han tenido que añadir más métodos específicos en la misma clase para validar diferentes datos, como puede ser la fecha para comprobar si es necesario cambiarla de formato; y para la configuración del número máximo de resultados posibles.

El siguiente método a comentar, es el más importante de la parte de *Back-End*. Como hemos comentado previamente, se encuentra en la clase "BusquedaExpedientesBOImpl" y se encarga de conectar con el motor de búsqueda, crear la *query* y obtener los resultados de dicha búsqueda.

El primer paso del método es establecer la conexión con *Elasticsearch* mediante HTTP. Con los atributos necesarios, se obtienen la ubicación del servidor de *Elasticsearch* (Host, puerto y protocolo). Además se configura la autenticación en el servidor con el usuario y contraseña correspondiente. Todo esto se configura para la creación de una instancia de un cliente de alto nivel proporcionado por *Elasticsearch* que permite simplificar la interacción con la API. Una vez la conexión se establece, se empieza con la creación de la cadena de búsqueda (query). En este punto, nos encontramos tres posibles casos debido a los métodos mencionados previamente:

1. **Se ha llegado a la función debido a hacer click en una tarjeta:** En este caso se comprueba el estado (pendiente de tramitar, etc) a filtrar y se añade en la cadena de búsqueda lo necesario para recoger los expedientes en ese estado y que se obtengan solo los que el usuario tiene asignados.

2. **Se ha llegado a la función al realizar una búsqueda:** Este caso es más complejo, por lo que requiere el uso de métodos extra para incluir los filtros en la cadena de búsqueda. Estos métodos se encargar de ir comprobando cada uno de los valores de los filtros, y asignando a la cadena de búsqueda la información necesaria. Algunos de estos casos específicos más importantes requieren de métodos adicionales para procesar los datos. Por ejemplo, la búsqueda genérica tiene que comprobar si contiene caracteres especiales y procesar la cadena de búsqueda para que sea adecuada para el motor *Elasticsearch*, mismo caso que con el filtro de "Interesados". Ocurre lo mismo con la fecha, es necesario procesar la fecha en un método específico para que la cadena contenga la estructura correcta para la query en el motor de búsqueda. En este caso, además hay que comprobar que si se ha dado al botón de buscar sin rellenar ningún campo se vuelvan a mostrar todos los expedientes asignados al usuario, independientemente del estado.

3. **Se ha realizado una búsqueda después de filtrar mediante las tarjetas:** En este caso, se va a realizar los dos procedimientos comentados en los puntos anteriores. Empezando por el filtro debido a las tarjetas y a posteriori, los filtros relacionados con la búsqueda. Consiguiendo así en la cadena, lo necesario para realizar ambos filtros a la vez.

Una vez se ha creado la cadena de búsqueda necesaria, se gestiona la solicitud de búsqueda en *Elasticsearch*. Se crea una instancia que construye la fuente de datos de la búsqueda y se establece que la *query* va a ser la cadena de búsqueda creada anteriormente. Además de esto, se le configuran una serie de parámetros para indicar que todos los términos de la *query* deben coincidir, limitar el número máximo de resultados a un atributo ya programado o en su defecto 9.999 y que los expedientes del resultado estén ordenados en orden descendente por la fecha de alta de los mismos, pues se ha seleccionado este orden para ser el orden por defecto. Ya por fin, se envía la solicitud de búsqueda a *Elasticsearch* y se recoge la respuesta del servidor y el "status" de la misma, en caso de que el status sea igual a"200" (código del protocolo HTTP para indicar que la respuesta es correcta), se continúa con el método.

En el momento en el que se confirma el código 200, se obtienen todos los expedientes de la respuesta HTTP y se almacenan en una estructura de datos de tipo "ExpedientesElasticSearch", la cuál es gestionada por los métodos comentados previamente, pues se retornan a los mismos en la clase "Action". Una vez estos reciben la estructura de datos con los expedientes ya almacenados, se llama a un método común que se encargará de obtener los documentos de la búsqueda y gestionarlos de tal manera que se almacenen como objetos de tipo "ResultadoBusquedaVO", el cual sirve para abstraer el tipo de dato, de esta manera no se diferencia entre *ElasticSearch* o cuando se usaba *SolR*, con los atributos necesarios. Una vez se crea una lista de estos objetos, se almacenan los datos en la sesión HTTP para que el objeto este disponible para el resto de componentes de la aplicación. De esta manera, mediante un fichero JSP se podrá acceder a los resultados de la búsqueda almacenados en esa sesión.

Para facilitar el desarrollo de estos métodos, se han desarrollado métodos adicionales. Algunos de estos métodos adicionales permiten obtener todos los parámetros de la solicitud HTTP, obtener todos los filtros que se encuentran activos, borrar la estructura de datos donde se almacenan los resultados de la búsqueda y obtener la lista de los parámetros de la cadena de búsqueda asociando a estos los resultados obtenidos en la búsqueda.

Para terminar la explicación sobre el desarrollo en *Back-End*, se va a desarrollar otro caso de uso importante y con una implementación compleja. El siguiente caso de uso permitirá al usuario exportar los resultados de la búsqueda. En la siguiente figura, podemos ver el modal para hacer la exportación.

Figura 17: Modal para configurar la exportación

El usuario deberá elegir la información que se mostrará de cada expediente y en cual de los 3 formatos quiere realizar la exportación:

1. **PDF (*Portable Document Format*):** Ofrece una forma sencilla y segura de presentar e intercambiar documentos, con independencia del software, el hardware o el sistema operativo que utilice quien consulte el documento.
2. **CSV (*Comma-Separated Values*):** Tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas y las filas por saltos de línea.
3. **XML (*eXtensible Markup Language*):** Formato de datos basado en texto utilizado para almacenar datos en forma legible.

Para implementar este caso de uso, lo primero que se hace es comprobar que la estructura de datos con los resultados de la búsqueda no esté vacía. Una vez comprobado se inicializa un documento con el formato necesario (por ejemplo A4 para el formato PDF). A continuación, se prepara la tabla con las cabeceras y los datos específicos de los resultados de búsqueda. Cada expediente estará representado en una fila donde cada columna indica una propiedad del mismo (número, fecha de alta, etc). Esto se hace mediante un método a parte que se encarga de introducir las columnas seleccionadas en el modal. Por último se configura la trama HTTP para indicar que es un documento y se escribe el flujo de salida de la trama para que el usuario pueda descargarlo.

6.2. Spring Framework

Spring es un framework de aplicación de código abierto para la plataforma Java. Spring Framework proporciona un modelo completo de programación y configuración para aplicaciones modernas basadas en Java, en cualquier tipo de plataforma de implantación.

Un elemento clave de Spring es el soporte infraestructural a nivel de aplicación. El framework se centra en la base de las aplicaciones para que el equipo de desarrollo pueda centrarse en la lógica empresarial a nivel de aplicación, sin ataduras innecesarias a entornos de implantación específicos.

Mediante Spring framework facilitamos la inversión de control (IoC). La inversión de control es un principio de diseño de software en el que el flujo de ejecución de un programa se invierte respecto a los métodos de programación tradicionales. En lugar de que los objetos creen y gestionen sus propias dependencias, Spring toma el control y gestiona estas responsabilidades, facilitando la inversión de control (IoC).

En Spring, los objetos que forman la columna vertebral de su aplicación y que son gestionados por el contenedor IoC de Spring se denominan *beans*. Una interfaz denominada "ApplicationContext.xml" representa el contenedor IoC de Spring y es responsable de instanciar, configurar y ensamblar los *beans*. El contenedor obtiene las instrucciones sobre los componentes a instanciar, configurar y ensamblar mediante la lectura de metadatos de configuración. Para la definición de *beans*, se han utilizado los *Bean scopes*. Los *bean scopes* definen cuantas instancias se pueden crear por *bean* definido en cada contenedor IoC.

Nos vamos a enfocar en los *scopes* más importantes:

- **Singleton:** (Por defecto) Para una única definición de *bean* se permite crear una única instancia de objeto para cada contenedor IoC de Spring. Todas las peticiones de *beans* con un ID o IDs que coincidan con esa definición de *bean* resultan en esa instancia de *bean* específica
- **Prototype:** Para una única definición de *bean* se permite crear cualquier número de instancias de objetos. Se crea una nueva instancia de *bean* cada vez que se realiza una solicitud para ese *bean* específico.

Mediante estos métodos, el contenedor de spring gestiona el ciclo de vida de los objetos ,se encarga de crear, inicializar, y destruir los *beans* según sea necesario.

La gestión de dependencias en Spring se realiza mediante Inyección de Dependencias (DI). DI es un patrón de diseño por el que un objeto recibe sus dependencias del contenedor, a la hora de crear el *bean*, en lugar de crearlas él mismo. La inyección de dependencias en Spring se puede realizar de tres maneras distintas:

- **Inyección por Constructor:** Las dependencias se pasan al *bean* a través de su constructor. Utilizada en este proyecto.
- **Inyección por Setter:** Las dependencias se establecen mediante métodos *setter*.
- **Inyección por Campos:** Utilizando anotaciones para inyectar directamente en campos del objeto (más común en Spring Boot y configuraciones basadas en anotaciones).

Por último, Spring proporciona soporte para la programación de tareas. Esto permite ejecutar tareas de forma periódica o en momentos específicos. La configuración de tareas programadas en Spring incluye:

- **Definir Tareas (*Jobs*):** Especificar la lógica de las tareas que se deben ejecutar.
- **Definir *Triggers*:** Configurar cuándo y con qué frecuencia se deben ejecutar las tareas.
- **Configurar el *Scheduler*:** Configurar un planificador que gestione la ejecución de las tareas según los *triggers* definidos.

En resumen, Spring simplifica el desarrollo de aplicaciones empresariales mediante la gestión automática de la creación, configuración y destrucción de objetos, la inyección de dependencias y la programación de tareas, promoviendo un diseño más limpio, modular y mantenible.

En el proyecto, el contenedo se divide en cuatro grandes bloques. El primero de estos bloques, se encarga de definir los *beans* necesarios para la búsqueda de expedientes indexados. El segundo de los bloques, se centra en la defincón de *beans* sobre la lógica de negocio relacionada con la búsqueda de expedientes. La tercera sección, se encarga de la definición de *beans* para los "Actions" de la aplicación (se explicará en el siguiente apartado lo que es un Action). Por último, se define la configuración para la ejecución de tareas programadas utilizando Quartz.

Figura 18: Logo de Spring Framework

6.3. Struts

Struts es un *framework* de código abierto desarrollado en Java que se utiliza principalmente para crear aplicaciones web. Es una herramienta poderosa que proporciona una estructura robusta y coherente para desarrollar aplicaciones web escalables y mantenibles. Struts es un *framework* MVC (Modelo-Vista-Controlador) que separa claramente la lógica de presentación (vista), la lógica de negocio (controlador) y la gestión de datos (modelo). Esta separación facilita el desarrollo y mantenimiento de aplicaciones web al promover un diseño modular y una mayor reutilización de código. Fue desarrollado por Apache Software Foundation. Esta tecnología está compuesta por una serie de componentes base:

- **ActionServlet:** Es el componente principal de Struts. Su tarea es recoger las solicitudes desde el lado del cliente, determinar la acción que se debe de tomar y luego dirigir la solicitud a la clase *Action* correspondiente.
- **Action:** Es una clase que representa la lógica de negocio o el controlador específico que maneja la solicitud. Cada Action está diseñada para realizar una tarea específica y puede interactuar con la capa de modelo para procesar la lógica de negocio. En este proyecto, la clase Action está compuesta por algunos de los métodos que hemos comentado previamente en la descripción del *Back-End* (Exportaciones y los dos primeros métodos, que se encargan de la preparación del entorno para realizar la búsqueda y la gestión de los resultados, no de realizar la propia búsqueda).
- **ActionForm (o Form Bean):** Es una clase JavaBean que actúa como un contenedor para los datos del formulario. Se utiliza para transferir datos entre la vista (JSP) y el controlador (Action).
- **ActionMapping:** Es una clase que contiene la configuración de mapeo entre las solicitudes de usuario y las acciones correspondientes. Este mapeo se define en un archivo de configuración XML. Cuando se activa una acción del usuario, este fichero la asocia al método (indicando su clase) que se ejecutará en ese caso, además añade el JSP que se mostrará cuando el método se haya ejecutado correctamente.
- **JSP (JavaServer Pages)**: Es la tecnología utilizada para crear la capa de vista en una aplicación Struts. Los JSPs generan el contenido HTML dinámico que se envía al cliente web. Se explicará a continuación.
- ***Business Logic y Data Store:*** La lógica de negocio puede estar implementada en clases Java que interactúan con la capa de datos (base de datos u otro tipo de almacenamiento). Estas capas manejan la lógica y los datos de la aplicación.

Struts es un *framework* poderoso para el desarrollo de aplicaciones web en Java EE, ofreciendo una estructura organizada que facilita la implementación del patrón MVC. Aunque hay *frameworks* más modernos, sigue siendo relevante en muchos proyectos.

En la siguiente figura, se mostrará el flujo de Sruts. Es importante destacar que en nuestro caso, no se ha hecho uso de los *Form bean*.

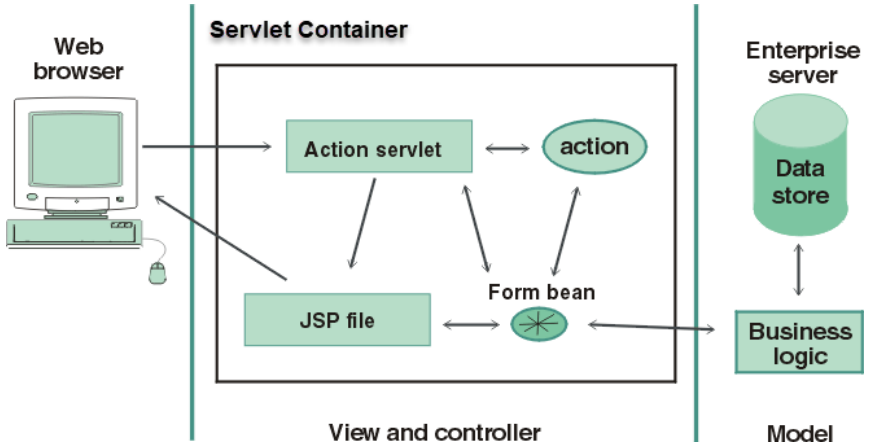

Figura 19: Flujo de Struts.

6.4. *Front-End*

El *Front-End* es la parte de una aplicación web con la que los usuarios interactúan directamente. Se compone de todo lo que los usuarios ven en sus pantallas cuando se accede a la página web: el diseño visual, las interfaces de usuario, y la experiencia general de navegación. Un buen diseño de *Front-End* supone que el usuario tenga una experiencia agradable a la hora de interactuar con la aplicación.

Además, un *Front-End* eficiente se debe de encargar de comunicarse de manera efectiva con el *Back-End* de la aplicación, asegurando que los datos se presenten de manera correcta. Debido a esto, el *Front-End* se encargará de recoger los resultados de la búsqueda del *Back-End* para mostrar la bandeja con los expedientes correctamente.

6.4.1. HTML

La primera tecnología que se va a comentar sobre el desarrollo del *Front-End* es HTML. HTML (Lenguaje de Marcas de Hipertexto, del inglés HyperText Markup Language) es el componente más básico de la Web. Define el significado y la estructura del contenido web. HTML utiliza "marcas" para etiquetar texto, imágenes y otro contenido para mostrarlo en un navegador Web.

Gracias a los ficheros HTML definidos en el proyecto, se puede definir la estructura de la bandeja de expedientes. Empezando de arriba a abajo, lo primero a definir son la estructura de las tarjetas de los KPI para el usuario.

Las tarjetas se pueden observar en la figura 15. Se empieza con la declaración de los textos "Mis expedientes asignados" y "Otros expedientes", estos textos no tienen ninguna funcionalidad adicional. Posteriormente, justo debajo de los textos, se crea la estructura para las 4 tarjetas. En cada tarjeta se introduce el texto ("Pendientes de tramitar", "Pendientes de subsanación", etc), así como la imagen de cada una de las tarjetas. También se crea el espacio para el número con el 0 por

defecto que se modificará en otra parte del código. Tanto a cada tarjeta como a sus elementos, se les asignan las clases necesarias para modificar su aspecto. Por último a cada tarjeta se le asigna la llamada al método correspondiente en caso de que se haga click sobre alguna de ellas. En todas las tarjetas se llama al mismo método pero modificando sus atributos dependiendo de la tarjeta.

Una vez terminada la estructura de las tarjetas, se comienza con el formulario de búsqueda de la figura 16. Empezando por la primera fila, se crea la estructura para el buscador genérico, el filtro de procedimiento, el filtro de fecha (que a su vez, está dividido para la fecha de inicio y la fecha de fin), y la selección de fases. Después se añade el texto de "BÚSQUEDA AVANZADA" que al ser pulsado, despliega el panel inferior con los filtros de razón de interés, el buscador de interesados y el botón de buscar. Cuando el panel está desplegado, se añade el "-" delante del texto búsqueda avanzada y cuando no está desplegado, se añade el "+". A todos los elementos, se les añaden las clases necesarias para modificar su apariencia. Se añade la funcionalidad de desplegar el panel y cambiar el símbolo al hacer click en el texto de búsqueda avanzada. Además, se le añade la llamada al método correspondiente a la hora de hacer el click al botón de buscar. En algunos de los elementos de los filtros, también se les añade la llamada a un método cuando se modifique el valor del campo.

Para finalizar con la pantalla principal se define el espacio donde va la bandeja de entrada donde se muestran los expedientes y su información. La creación de la bandeja se explicará más adelante, pues se realiza desde un fichero JSP usando la librería *DataTables*. En el fichero HTML solo se identifican los índices de las columnas de la tabla que muestran la información (Fecha, fase, etc).

Por último, en el fichero HTML se estructuran los elementos para los 2 posibles modales que se pueden mostrar. Un ejemplo de estos, es el modal que se muestra en la figura 17. Los botones que activan la aparición pertenecen a la propia bandeja, por lo que serán explicados más adelante. La estructura de estos 2 modales es análoga. En ambas implementaciones, nos encontramos con un título arriba que puede ser "Configurar exportación" o "Configurar tabla", junto a una "x" con la funcionalidad de cerrar el modal. A continuación, ambos contienen los 6 "checkbox" para configurar las columnas que aparecerán en la exportación o en la bandeja de expedientes. La diferencia se encuentra en la parte de abajo, como se puede ver en la figura 17, en ese modal encontramos 3 botones compuestos por una imagen para indicar en que formato queremos realizar la exportación. En cambio en el modal de configurar la tabla, solo encontraremos un botón de aceptar para finalizar la selección de columnas.

6.4.2. CSS

Para poder realizar una actualización visual adecuada de la bandeja de expedientes es imprescindible el uso de CSS para gestionar el estilo de la aplicación.

Hojas de Estilo en Cascada (del inglés Cascading Style Sheets) o CSS es el lenguaje de estilos utilizado para describir la presentación de documentos HTML o XML, en nuestro caso HTML. CSS describe cómo debe ser renderizado el elemento estructurado en la pantalla.

Cada regla CSS comienza con un selector o una lista de selectores que indican al navegador a qué elemento o elementos deben aplicarse dichas reglas. Para este proyecto, se ha decidido que la selección de elementos se gestione principalmente con clases (como hemos comentado en la expli-

cación de HTML). Esto se debe a la repetición de estructuras de varios elementos diferentes. Un ejemplo claro de esto es de la figura 15, donde las 4 tarjetas comparten muchos estilos como el tamaño del texto, la posición de los iconos, el radio de los bordes y más; también podemos apreciar este patrón en los dos modales que comparten el título, la "x" para cerrar el modal y los checkboxs.

Las hojas de estilo implementadas son de tipo "externo", esto quiere decir que el CSS está escrito en un archivo independiente con una extensión ".css" y que lo vinculas desde un elemento "link" en el fichero HTML.[8]

```css
.container > div {
  flex: 1 0 auto;
}

#leader::first-letter {
  float: left;
  font: italic bold 2em Operator, sans-serif;
}
```

Figura 20: Ejemplo de 2 reglas de CSS.

6.4.3. JavaServer Pages

JavaServer Pages (JSP) es una tecnología que ayuda a los desarrolladores de software a crear páginas web dinámicas basadas en HTML y XML, entre otros tipos de documentos. JSP es similar a PHP, pero usa el lenguaje de programación Java.

Una de las características principales de JSP es su capacidad para separar la presentación de la lógica de negocio, permitiendo a los desarrolladores insertar fragmentos de código Java directamente en las páginas HTML.

Lo primero que se realiza en los ficheros JSP, es la configuración de la página JSP para usar el lenguaje Java. También hay que definir que el tipo de contenido va a ser HTML y especificar que la codificación de caracteres va a ser *UTF-8*.

A continuación, se declara la librería de etiquetas de *Struts*. Struts, como hemos comentado previamente, es una herramienta de soporte para el desarrollo de aplicaciones Web que permite reducir el tiempo de creación. Proporciona un conjunto de etiquetas personalizadas que simplifican la integración de la lógica de negocio y la presentación en las páginas JSP. El uso de esta librería proporciona diferentes ventajas como:

- **Simplicidad y Claridad:** Las etiquetas abstractas simplifican el código JSP, haciendo que sea más legible y fácil de mantener.
- **Reutilización de Código:** Las etiquetas permiten la reutilización de componentes de presentación en diferentes partes de la aplicación.
- **Separación de la Lógica de Negocio y Presentación:** Facilita la separación de la lógica de negocio y la presentación, mejorando la modularidad del código.

[8] El código no está relacionado con el proyecto del trabajo de fin de grado.

Para terminar con la configuración se declaran el resto de librerías, otra librería que es importante en el desarrollo en la librería *JavaServer Pages Standard Tag Library* (JSTL). JSTL extiende las JSP proporcionando cuatro bibliotecas de etiquetas (*Tag Libraries*) con utilidades ampliamente utilizadas en el desarrollo de páginas web dinámicas.

Implementación en el código

Para la implementación de la bandeja de entrada se han desarrollado diferentes ficheros JSP, nos vamos a centrar en los más importantes.

El primer fichero del que se va a hablar se encarga de mostrar la tabla. Lo primero que realiza este fichero es la configuración de una interfaz para mostrar los resultados de la búsqueda con la capacidad de colapsar y expandir esta sección (La sección por defecto aparece colapsada y una vez cargan los resultados se expande con la tabla). Para realizar estas funciones se utilizan las etiquetas de *Struts*, *JSTL* y *DisplayTag* (otra librería de etiquetas) junto con componentes de Bootstrap para la presentación y el comportamiento dinámico. Una vez se realiza esto, se comprueba que no haya habido errores, en ese caso se gestionan los mensajes de error pertinentes. Se accede a la estructura de datos con la búsqueda de resultados que se almacenó en la sesión HTTP en uno de los métodos del *Back-End* que se han comentado previamente. Si la estructura de datos no está vacía, se muestra la tabla con los resultados de la búsqueda de expedientes utilizando la librería *DisplayTag*. A la tabla se le asignan las clases pertinentes para que esta sea manejada por la librería *DataTables*.

Mediante las etiquetas de la librería DisplayTag, se facilita en gran medida la estructuración de la tabla. Para cada columna de la misma, se le asigna un campo de los expedientes al título de la columna a mostrar, haciendo esta asignación de manera automática para todas las filas de la tabla. También se configurará la columna de fase para que cuando el ratón este encima de la celda, aparezca el botón "Tramitar". Este botón redirigirá al usuario al escritorio de tramitación del expediente pertinente. El escritorio de tramitación es otra parte de la aplicación ptW@nda cuyo desarrollo no se lleva a cabo en este Trabajo de Fin de Grado.

Para finalizar el fichero, se añade un *script* JavaScript que nos permite inicializar y configurar el *plugin DataTables* sobre la tabla de resultados de la búsqueda. Para la configuración se definen opciones como el orden de las filas, que la tabla se adapte al tamaño de la pantalla y más. Se explicará la librería *DataTables* más adelante.

Para el resto de ficheros JSP, no solo se integra Bootstrap, en algunos de los ficheros se integra *JQuery* para interactuar de manera más sencilla con el HTML. La combinación de todas estás tecnologías permiten la creación de una interfaz de usuario mejorada con funcionalidades avanzadas. El resto de ficheros JSP creados tienen funcionalidades como la creación de formularios interactivos o de búsqueda con etiquetas de texto y campos de entrada; y la creación de modales.

Para finalizar con este apartado, la gestión de errores que se ha comentado en el fichero principal de los desarrollados con esta tecnología, también se gestionan desde ficheros especializados de JSP, en los que dependiendo del resultado se muestran mensajes de error o informativos de éxito.

6.4.4. JavaScript

JavaScript (JS) es un lenguaje de programación interpretado utilizado en el lado del cliente (*Front-End*) para dotar de mejoras dinámicas e interactivas a las páginas web, mejorando la experiencia de usuario. JavaScript es un lenguaje de programación ligero, interpretado, o compilado justo-a-tiempo (*just-in-time*) con funciones de primera clase (Los lenguajes con funciones de primera clase significa que las funciones son tratados como cualquier otra variable, por ejemplo, una función puede ser pasada como argumento a cualquier otra función). Desde su creación, JavaScript ha evolucionado para convertirse en uno de los lenguajes más importantes y versátiles del desarrollo web moderno. Además permite gestionar la comunicación asíncrona con el servidor a través de tecnologías como AJAX, la cual comentaros a continuación.

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. En el contexto de nuestra aplicación web, JavaScript juega un papel crucial en la gestión del *Front-End*. Mediante JavaScript, podemos manipular el DOM (Document Object Model) para conseguir actualizar dinámicamente el contenido de la página sin necesidad de recargarla. Javascript es el único lenguaje de programación que entienden de forma nativa los navegadores.

Implementación en el código

Para los desarrollos en JavaScript, se han implementado las denominadas *Arrow functions* (funciones flecha). Las funciones flecha fueron introducidas en ES6 (ECMAScript 2015) y proporcionan una sintaxis más concisa para escribir las funciones. A parte de reducir el código necesario, permiten gestionar el contexto de "this" de manera más intuitiva que las funciones tradicionales. [9]

```
// arrow function expression
const add = (a, b) => {
    return a + b;
}
// very simple and concise syntax
const add = (a, b) => a + b;
// eliminates (), if it has single parameter
const square = a => a * a;
```

Figura 21: Comparación función tradicional y función flecha

En algunas de las funciones desarrolladas, se utiliza la técnica *Asynchronous JavaScript and XML* (AJAX). Esta técnica es utilizada en el desarrollo para permitir a la aplicación web enviar y/o recibir datos del servidor de manera asíncrona sin tener que recargar la página. Para manejar esta tecnología se usa librería de JavaScript *jQuery*, la cual simplifica el manejo de AJAX. Para ver en más detalle como funciona, se va a explicar un pequeño ejemplo en el siguiente apartado del documento.

[9] El código no está relacionado con el proyecto del trabajo de fin de grado.

6.4.5. jQuery

jQuery es una biblioteca biblioteca de JavaScript que permite simplificar la manera de interactuar con los documentos HTML (explicados previamente), manipular el árbol DOM, gestionar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. Al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que requieren menos código que con el lenguaje nativo. jQuery es un software libre y de código abierto.

Mediante esta biblioteca, se simplifica el desarrollo a la hora de manipular el DOM y gestionar los eventos gracias a una sintaxis concisa. jQuery también incluye métodos para crear efectos y animaciones con facilidad. Además, gracias a la comunidad cuenta con un gran número de *plugins* y extensiones para expandir sus funcionalidades. A parte de esto, se encarga de las diferencias entre los navegadores, lo que permite a los desarrolladores escribir código que funcione en todos los navegadores web.

Implementación en el código

Como se ha comentado, en el apartado anterior, se ha utilizado la técnica AJAX. Mediante jQuery, podemos implementar AJAX de una manera mucho más clara y sencilla. Esta técnica ha tenido un gran peso en el desarrollo de la aplicación. A continuación, se va a explicar un pequeño ejemplo de como funciona la combinación de jQuery y AJAX.

En el ejemplo de la figura 22, lo primero que se realiza es la configuración AJAX. Mediante "type:POST", se indica que la trama HTTP va a ser de tipo POST a la "url:/things". Para finalizar la configuración, en el campo "data:" se incluye los datos a enviar en la trama. A continuación, se indica el campo "success", en este campo se implementa lo que la función debe hacer si la solicitud se ha completado de manera exitosa. En caso contrario, la función realizará lo que se incluye en el campo "error". Adicionalmente, se puede añadir el campo "complete", la implementación de este campo se ejecuta después de los campos previos, independientemente si la solicitud ha sido exitosa o no. [10]

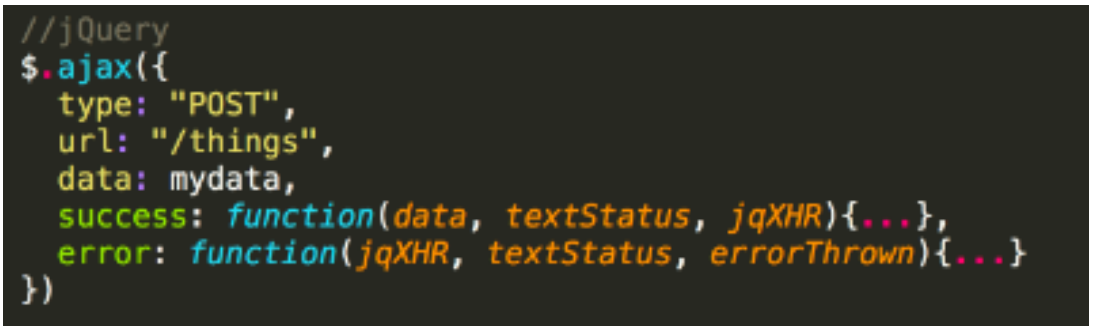

Figura 22: Ejemplo de AJAX usando JQuery

[10] El código de la imagen no está relacionado con el proyecto del trabajo de fin de grado.

6.4.6. Bootstrap

Bootstrap es un *framework Front-End* de código abierto que se utiliza para desarrollar sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como extensiones de JavaScript adicionales. A diferencia de muchos *frameworks* web, solo se ocupa del desarrollo *Front-End*.

Bootstrap es modular y consiste esencialmente en una serie de hojas de estilo LESS (Lenguaje de Hojas de Estilo) que implementan la variedad de componentes de la herramienta. Debido a esto, proporciona un conjunto de herramientas CSS, HTML y JavaScript que facilitan la creación de interfaces web modernas y uniformes. Aquí están algunas de las principales características:

- **Sistema de cuadrícula (*grid*):** Permite a los desarrolladores crear diseños de manera sencilla. Utiliza una cuadrícula de 12 columnas que se adapta a diferentes tamaños de pantalla.
- **Componentes predefinidos:** El *framework* incluye una amplia variedad de componentes reutilizables como botones, modales, barras de navegación y más. Estos componentes vienen con estilos predeterminados que se pueden personalizar para configurar el aspecto visual de la aplicación web.
- **JavaScript integrado:** Bootstrap incluye *plugins* de JavaScript que añaden funcionalidad avanzada a los componentes como modales, *tooltips*, *popovers* y más. Estos *plugins* se pueden activar mediante atributos de datos o utilizando la librería que hemos mencionado previamente, jQuery.

Gracias a Bootstrap podemos conseguir de manera sencilla una aplicación web responsiva (se adapta al tamaño de la pantalla) y con una apariencia consistente en toda la aplicación.

Implementación en el código

Uno de los elementos con más importancia de la aplicación que se ha desarrollado esencialmente con Bootstrap son los modales. Los modales permiten mostrar información adicional superponiendo sobre el resto de la pantalla. Con esta técnica podemos evitar el tener que navegar a una nueva página para mostrar esta información adicional.

Un ejemplo de los modales desarrollados en la aplicación son la ventana de exportar y configurar la tabla, el modal de exportar la tabla se encuentra en la figura 17. Estos modales constan de componentes clave: el botón que se encarga de activar/abrir el modal, el contenido del propio modal y las funciones JavaScript que se encargan de controlar su comportamiento.

Los modales de bootstrap tienen configuradas algunas propiedades por defecto que facilitan su uso y ofrecen funcionalidades interesantes para mejorar la experiencia de usuario. Una vez se abre el modal, al hacer click fuera de él se cierra automáticamente. Además, automáticamente se desactiva la función de *scroll* de la bandeja para poder realizar el *scroll* en el modal.

6.4.7. DataTables

DataTables es una potente librería de jQuery que se utiliza para mejorar la funcionalidad de las tablas HTML, proporcionando características avanzadas como paginación, búsqueda, ordenación y filtrado de datos. Esta librería es de las más importantes a la hora del desarrollo de la aplicación pues se usa para crear la bandeja de expedientes.

En la tabla, cada fila representa un expediente donde cada columna representa un dato sobre el expediente, obteniendo así una tabla en la que todas las celdas muestran un dato importante.

La librería ofrece varias características muy interesantes a la hora de su implementación en el desarrollo, algunas de las funcionalidades que ofrece son:

- **Paginación:** La librería incluye la función de la paginación con las opciones de página siguiente, anterior o haciendo click en el número de página.
- **Número de entradas por página:** Se permite al usuario seleccionar el número de expedientes que se muestran por página, actualizando la paginación al momento.
- **Búsqueda y filtrado:** Permite realizar un filtrado de los resultados de la búsqueda por medio de una búsqueda de texto y/o por posibles valores de las columnas.
- **Ordenación por columnas:** Permite ordenar las filas respecto al orden ascendente o descendente de los valores de una columna.
- **Múltiples fuentes de datos:** Permite el uso de varias fuentes (DOM, JavaScript, etc) a la hora de mostrar los datos.
- **Personalización:** DataTables es altamente configurable y permite a los desarrolladores personalizar la apariencia y el comportamiento de las tablas.

Con las funcionalidades comentadas previamente, se consigue una interfaz interactiva que ofrece una experiencia de usuario muy agradable. Además, esta librería permite la implementación del *framework* Bootstrap, integrar el *framework* es sencillo y permite que las tablas adopten el estilo y la funcionalidad de Bootstrap, haciendo que se vean consistentes con el resto de la aplicación. Basta con incluir los ficheros JS y CSS de DataTables Bootstrap necesarios para la inicialización de la librería.

Algunos ejemplos de personalización de la tabla que permite la librería y se han implementado en este proyecto pueden ser: Modificación de la paginación, se ha reducido el máximo número de páginas que aparece en el navegador de páginas de 7 a 5, además se ha modificado el estilo de los números. Este es solo un ejemplo de las posibilidades que ofrece la librería para personalizar la tabla de manera que se integre perfectamente con la aplicación web.

Por último, comentar que DataTables es un librería de código abierto que permite el uso de *plugins* para extender las funcionalidades que tiene la librería por defecto, convirtiéndola en una librería todavía más completa si cabe.

6.5. Resultado

Mediante el uso de todas las tecnologías que se han descrito hasta el momento, conseguimos el resultado final de la bandeja de expedientes de la Plataforma de Tramitación w@nda.

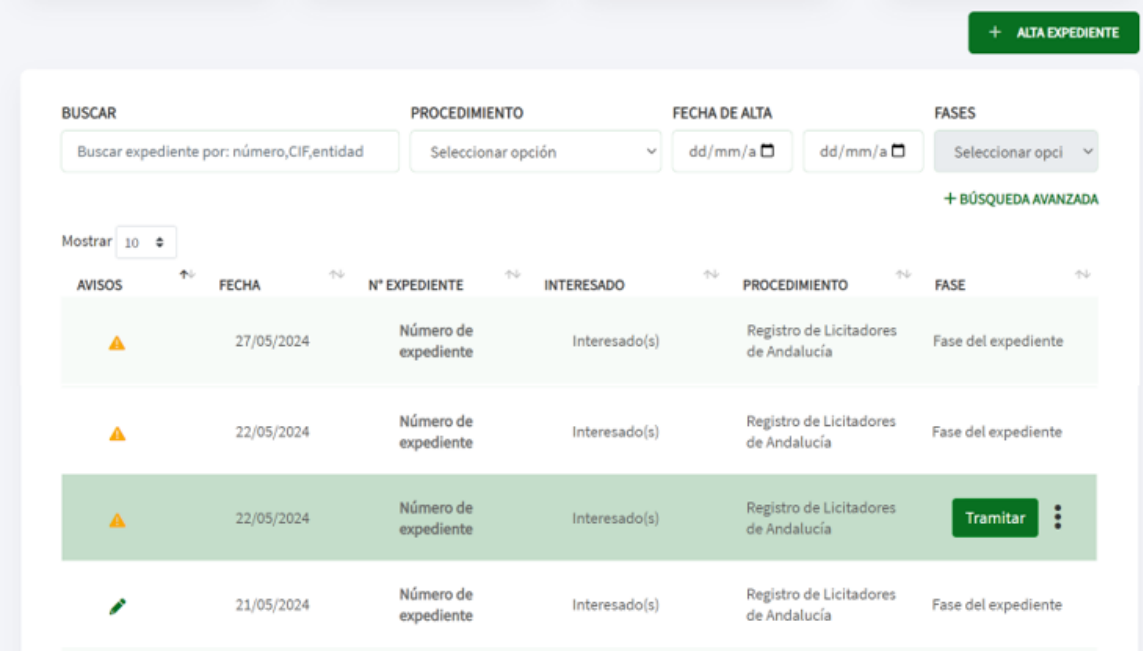

Figura 23: Primera parte del resultado de la bandeja.

Aunque no se aprecia en la figura, al principio de la bandeja nos encontramos con los KPIs que ya han sido mostrados en la figura 15, con las funcionalidades que hemos descrito previamente para la búsqueda de expedientes (por motivos de visualización de los datos no se muestran en la figura). A continuación, nos encontramos con un botón que nos redirige a la pantalla para crear expedientes (esta parte de la aplicación no corresponde con la desarrollada para este proyecto de fin de grado).

Lo siguiente que nos encontramos en con los filtros de la búsqueda (búsqueda de texto, selector de procedimiento, fechas y selector de fase) con el botón de búsqueda avanzada para desbloquear el resto de filtros. En la figura 16 encontramos el menú de búsqueda desplegado para ver todos los filtros.

Por último llegamos a la tabla que muestra los resultados de la búsqueda. En la primera columna encontramos un icono relacionado con los KPIs para indicar si el expediente está pendiente de tramitar, de subsanación o de firma. La siguiente columna muestra la fecha de creación del expediente, como se comentó previamente el resultado está ordenado de manera descendente según la fecha. En las siguientes columnas se muestran el número de expediente, los interesados y el procedimiento al que pertenece dicho expediente.

En la última columna, se muestra la fase en la que se muestra el expediente. Como se puede apreciar en la tercera fila de la tabla en la figura 23, cuando el cursor se encuentra encima de una fila, aparte de modificar el color de la fila para resaltar el expediente aparece un botón con el texto "Tramitar" que nos redirige al *Escritorio de Tramitación* de dicho expediente (esta parte de la aplicación no corresponde con la desarrollada para este proyecto de fin de grado). A parte del botón "Tramitar", hay un segundo botón compuesto por los 3 puntos que abre un menú desplegable para redirigir al usuario a ver más información sobre el expediente o asignar usuarios al expediente.

En la siguiente figura se va a mostrar la parte final de la bandeja de expedientes, en las figuras 23 y 24 se muestran 7 expedientes, pero cabe destacar que en la búsqueda se han mostrado 3 expedientes más que no se muestran para la correcta visualización de la información.

Figura 24: Segunda parte del resultado de la bandeja.

Para acabar con los expedientes, nos encontramos con los últimos expedientes que se muestran de los 10 resultados totales. Se puede apreciar como uno de estos está pendiente de firma debido a su icono en la primera columna.

Finalizando con la bandeja de entrada, a la izquierda de la pantalla nos encontramos con un texto que muestra "X resultados totales" (siendo X el número de resultados) o "No hay resultados", dependiendo del mismo.

En el centro, nos encontramos con la paginación, en este caso al haber solo 10 resultados totales y como se ve en la figura 23, estar configurado para mostrar 10 expedientes por página, solo tenemos 1 página. La paginación cuenta con el botón "Siguiente", "Anterior" y se puede hacer click en el número de página al que quieres dirigirte.

Por último, en la parte de la derecha nos encontramos con los dos botones que nos permiten desplegar los modales que hemos comentado previamente, el primero nos permite abrir el modal para exportar, mostrado en la figura 17 y el segundo nos permite mostrar el modal análogo para configurar que columnas se ven de la tabla.

7. Pruebas

En el desarrollo de cualquier aplicación, la pruebas son una etapa crucial para poder asegurar el correcto funcionamiento de la aplicación y que el software cumpla con los requisitos establecidos. Las *pruebas de software* son el proceso de evaluar y verificar que un producto o aplicación de software hace lo que se supone que debe hacer. Entre los beneficios de unas buenas pruebas se incluyen la prevención de errores y la mejora del rendimiento.

Las pruebas de software siguen un proceso común. Las tareas o pasos incluyen la definición del entorno de prueba, el desarrollo de casos de prueba, la elaboración de *scripts*, el análisis de los resultados de las pruebas y la presentación de informes de defectos.

Para conseguir unos buen resultado a la hora de realizar las pruebas, es esencial tener unas buenas prácticas a la hora de realizarlas. Las pruebas manuales o las pruebas *ad-hoc* pueden ser suficientes para compilaciones pequeñas. Sin embargo, en los sistemas más grandes, las herramientas se utilizan con frecuencia para automatizar las tareas. Las pruebas automatizadas ayudan a los equipos a implementar diferentes escenarios, probar elementos diferenciadores y obtener rápidamente comentarios sobre lo que funciona y lo que no.

Otro factor importante a la hora de realizar las pruebas es la continuidad. Una realización continua de pruebas permite validar el software en entornos de prueba realistas en una fase más temprana del proceso, lo que mejora el diseño y reduce los riesgos. El equipo de desarrollo prueban cada compilación a medida que está disponible, permitiendo acotar en gran medida los posibles errores durante la implementación.

Para la nueva versión de la bandeja de expedientes, se han llevado a cabo diferentes tipos de pruebas para garantizar la calidad de esta. A continuación, vamos a explicar los tipos de prueba más importantes que se han realizado:

- **Pruebas unitarias:** Validan que cada unidad de software funciona según lo esperado. Una unidad es el componente comprobable más pequeño de una aplicación.
- **Pruebas de integración:** Las pruebas de integración se realizan para verificar que los distintos módulos o componentes de la aplicación funcionan correctamente en conjunto. Una vez se realizó el desarrollo de este módulo, se comprobó la correcta integración con el resto de la aplicación.
- **Pruebas de aceptación:** son realizadas para validar que la aplicación cumple con los criterios y requisitos especificados por el cliente o usuario final. Estas pruebas se centran en la funcionalidad desde la perspectiva del usuario y aseguran que el sistema realice las tareas esperadas.

Como ejemplo de prueba unitaria que tuve que realizar durante el desarrollo. A la hora de comprobar que la bandeja funcionaba correctamente, se comprobó que se mostrase el número de resultados correctamente. Para comprobarlo, se utilizó la aplicación *SQL Developer* para introducir un gran número de expedientes de prueba. Una vez introducidos, se comprobó que cuando el número de resultados superaba las 4 cifras, el separador de miles se mostraba con "," en vez de con "."

Una vez se acotó donde estaba el problema, se modificó la configuración de la tabla para mostrar el separador de miles con "." en vez de con ",". Se comprobó que la solución había resuelto el problema correctamente. Cuando se comprobó que el número se mostraba correctamente, se eliminaron los expedientes de la base de datos.

Para asegurar la calidad y funcionalidad del tramitador de expedientes 3.0, se realizaron exhaustivas pruebas de aceptación al final del ciclo de desarrollo. Las pruebas conllevaron la verificación de todos los requisitos funcionales y no funcionales definidos al inicio del proyecto. Se probaron escenarios reales y casos de uso típicos para garantizar que el sistema cumpliera con las necesidades operativas. Gracias a la metodología ágil adoptada, se pudieron hacer ajustes y mejoras en tiempo real según los comentarios recibidos. Estas pruebas de aceptación confirman que la bandeja está lista para ser desplegada en producción.

8. Conclusión

A lo largo del desarrollo del proyecto se ha llevado a cabo una transformación integral de la bandeja de expedientes de una plataforma de tramitación, integrando esta en la versión 3.0 de ptw@nda. Este proceso ha supuesto varios cambios significativos que permiten mejorar tanto el aspecto de la aplicación como el rendimiento.

Para el diseño y desarrollo de *Back-end*, se ha migrado el motor de búsqueda de SolR a ElasticSearch, permitiendo una mejora en el rendimiento de las búsquedas. Esta migración conlleva el desarrollo en Java de los métodos para realizar las búsquedas en el motor. Además, se han implementado los métodos para exportar los resultados en los 3 formatos comentados previamente.

El uso del framework Spring proporcionó una estructura modular y escalable, mejorando la mantenibilidad y facilitando el desarrollo de nuevas funcionalidades. Struts se utilizó para gestionar la lógica de control, facilitando la comunicación entre el *Front-end* y el *Back-end* y asegurando una separación clara de responsabilidades.

En cuanto al desarrollo de *Front-end*, mediante HTML, CSS, JavaScript y el uso de librerías y frameworks como jQuery, Bootstrap y DataTables, se ha creado una interfaz de usuario atractiva y responsiva que mejora la experiencia de usuario en gran medida.

Para asegurar el correcto funcionamiento de los desarrollos de la aplicación que se han implementado en el código, se han realizado pruebas. Estas pruebas además, garantizan a que el software cumpla con los requisitos establecidos al comienzo del proyecto.

Todos estos desarrollos se han realizado siguiendo la metodología ágil, *Scrum*. Esto ha permitido al equipo de desarrollo acomodarse mejor a las exigencias del proyecto, debido a las continuas demandas de nuevas funcionalidades.

Gracias al uso de las nuevas tecnologías en este proyecto, se ha mejorado significativamente la funcionalidad y el rendimiento de la bandeja de expedientes del tramitador. Con la migración del motor de búsqueda se han optimizado las búsquedas y mediante los frameworks y/o librerías usadas en el *Front-end* se ha proporcionado una herramienta eficiente y responsiva.

Trabajar en este proyecto ha sido una experiencia muy enriquecedora que me ha permitido poner en práctica las habilidades obtenidas durante el grado. Este proyecto ha sido fundamental para mi crecimiento profesional, permitiéndome afrontar distintos desafíos a los que buscar solución. Esta experiencia me ha permitido prepararme para abordar proyectos futuros con confianza y competencia.

9. Trabajo futuro

Para finalizar con la memoria, en este último capítulo vamos a comentar los futuros pasos del proyecto que, en principio, se irán llevando a cabo en el futuro.

Por la tipología del proyecto, las líneas futuras vendrán marcadas principalmente por el cliente, aunque es posible por parte del equipo de desarrollo la proposición de nuevas mejoras para una posible fase tres del proyecto. Como propuestas de mejora, se listan las siguiente:

Se busca cambiar la infraestructura de la aplicación a Angular. Angular es un framework para aplicaciones web desarrollado en TypeScript, de código abierto, que se utiliza para crear y mantener aplicaciones web de una sola página. Su objetivo es aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles. Angular ofrece varias ventajas como: permitir desarrollar aplicaciones web en módulos independientes, lo que facilita la mantenibilidad y escalabilidad del código; optimiza el rendimiento de las aplicaciones mediante técnicas como la compilación AOT (*Ahead-Of-Time*) y la detección de cambios eficiente, lo que resulta en una experiencia de usuario más rápida y fluida; y el uso de componentes, que facilita la reutilización de código y la creación de interfaces de usuario consistentes, lo que reduce el tiempo de desarrollo y los errores.

Otro aspecto con el que se pretende mejorar en un futuro, es por medio de la aplicación de técnicas DevOps. DevOps es un conjunto de prácticas que agrupan el desarrollo de software y las operaciones de la tecnología de la información (TI). Su objetivo es hacer más rápido el ciclo de vida del desarrollo de software y proporcionar una entrega continua de alta calidad, cada cambio que pasa las pruebas automáticas puede ser desplegado en producción de manera rápida y segura. Esto reduce el tiempo de entrega de nuevas funcionalidades y correcciones de errores. Además de esto, se pretende conseguir una integración continua, permitiendo que los cambios en el código se integren y prueben automáticamente. Cada modificación se verifica mediante pruebas automatizadas, asegurando que los nuevos cambios no introduzcan errores y que el código siempre esté en un estado desplegable.

Además, muy a la larga, se buscará que la plataforma funcione de manera parecida a un "*marketplace*", en el cual el usuario tramitador pueda seleccionar que módulos necesita de la plataforma.

A pesar de los avances tecnológico comentados previamente, la base de datos continuará siendo Oracle. La decisión de mantener Oracle se debe a su robustez, fiabilidad y la integración existente con los sistemas actuales de la organización. Oracle ha demostrado ser una solución sólida para el manejo de datos críticos, asegurando la continuidad y estabilidad del sistema mientras se implementan mejoras en otras áreas.

Referencias

[1] Deemer, P., Benefield, G., Larman, C., & Vodde, B. (2009), *Información básica de SCRUM.*

[2] Burnette, E. (2005) *Eclipse IDE Pocket Guide: Using the Full-Featured IDE. .º'Reilly Media, Inc.".*

[3] **https://www.atlassian.com/es/git/tutorials/what-is-version-control**

[4] **https://maven.apache.org/**

[5] **https://solr.apache.org/guide/solr/latest/index.html**

[6] **https://www.elastic.co/docs**

[7] **https://www.ibm.com/es-es/topics/elasticsearch**

[8] AKCA, M. A., Aydoğan, T., & İlkuçar, M. (2016), *An analysis on the comparison of the performance and configuration features of big data tools Solr and Elasticsearch. International Journal of Intelligent Systems and Applications in Engineering, 4(Special Issue-1), 8-12.*

[9] **https://spring.io/projects/spring-framework**

[10] **https://struts.apache.org/**

[11] **https://developer.mozilla.org/es/docs/Web/HTML**

[12] **https://www.ibm.com/docs/es/dmrt/9.5?topic=files-javaserver-pages-jsp-technology**

[13] **https://jquery.com/**

[14] **https://getbootstrap.com/docs/4.6/getting-started/introduction/**

[15] **https://datatables.net/**

[16] **https://dattell.com/wp-content/uploads/2020/10/Screen-Shot-2022-12-20-at-10.26.44-AM-1024x519.png**