UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Desarrollo del backend de una aplicación de gestión del marcado CE de la Unión Europea

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: García Marín, Ginés

Tutor/a: Villanueva García, Alicia

Cotutor/a externo: VILLANUEVA GARCIA, JAVIER

CURSO ACADÉMICO: 2021/2022

Resumen

Para demostrar que algunos productos comercializados por la UE cumplan con sus requisitos exigidos, es necesario que tengan el marcado CE. A causa de la complejidad requerida para certificar con el marcado CE, los fabricantes de puertas necesitan herramientas que les faciliten la gestión del cumplimiento normativo, la generación de la documentación técnica y el control de cada unidad puesta en el mercado.

Ante esta problemática, se plantea el desarrollo de una aplicación que agilice el proceso. Este TFG se enmarca en unas prácticas en empresa durante las cuales se desarrolla la aplicación y se centra en el desarrollo del *backend* de la misma. El desarrollo del *frontend* correrá a cargo de otro miembro del equipo, también estudiante del GII, y será objeto de otro TFG.

En la aplicación web, los usuarios podrán gestionar el trámite de la obtención del certificado y los administradores podrán apoyar a los usuarios en dicho trámite además de poder manejar la información de los usuarios.

El proyecto será desarrollado con una metodología ágil a través de sprints y haciendo uso de las tecnologías: Spring como *framework* de *backend*, Hibernate para el mapeo a base de datos y MySQL como gestor de base de datos relacional.

Palabras clave: aplicación web, metodología ágil, Spring, MySQL, marcado CE.

Resum

Per demostrar que alguns productes comercialitzats per la UE compleixin els seus requisits exigits, cal que tinguin el marcatge CE. A causa de la complexitat requerida per certificar amb el marcatge CE, els fabricants de portes necessiten eines que els facilitin la gestió del compliment normatiu, la generació de la documentació tècnica i el control de cada unitat posada al mercat.

Davant d'aquesta problemàtica, es planteja el desenvolupament d'una aplicació que agilitzi el procés. Aquest TFG s'emmarca en unes pràctiques en empresa durant les quals es desenvolupa l'aplicació i se centra en el desenvolupament del backend. El desenvolupament del frontend anirà a càrrec d'un altre membre de l'equip, també estudiant del GII, i serà objecte d'un altre TFG.

A l'aplicació web, els usuaris podran gestionar el tràmit de l'obtenció del certificat i els administradors podran donar suport als usuaris en aquest tràmit a més de poder manejar la informació dels usuaris.

El projecte serà desenvolupat amb una metodologia àgil a través de sprints i fent ús de les tecnologies: Spring com a framework de backend, Hibernate per al mapeig a base de dades i MySQL com a gestor de base de dades relacional.

Palabras clave: aplicació web, metodologia àgil, Spring, MySQL, marcatge CE.

Abstract

To demonstrate that some products marketed by the EU meet its required requirements, it is necessary that they have the CE marking. Due to the complexity required to certify with the CE marking, door manufacturers need tools that facilitate the management of regulatory compliance, the generation of technical documentation and the control of each unit placed on the market.

Faced with this problem, the development of an application that speeds up the process is proposed. This bachelor thesis is part of an internship in a company during which the application has been developed. The TFG focuses on the development of its backend. The development of the frontend will be carried out by another member of the team, also a GII student, and will be the subject of another TFG.

In the web application, users will be able to manage the process of obtaining the certificate and administrators will be able to support users in said process as well as being able to manage user information.

The project will be developed with an agile methodology through sprints and using the following technologies: Spring as a backend framework, Hibernate for database mapping and MySQL as a relational database manager.

Palabras clave: web application, agile methodology, Spring, MySQL, CE marking.

Tabla de contenidos

1. Introducción

Nos situamos en el contexto del desarrollo de las prácticas de empresa, concretamente en una empresa consultora software, que se encarga de desarrollar software a medida de las necesidades de los clientes, ya sean aplicaciones web, Android o IOS.

Durante años el proceso de solicitud de los certificados de calidad para todo tipo de productos producidos por empresas de diferente índole ha sido bastante compleja y extendida en el tiempo, ya que exige mucho papeleo y constantes errores que alargan el procedimiento. Debido a esto, una empresa conformada por un grupo de ingenieros puso en marcha un proyecto para facilitar y agilizar toda esta gestión. Para cumplir su cometido decidieron desarrollar una aplicación web para llevar a cabo dicho trámite.

Llegados a este punto, la empresa decide contactar con la empresa en la que nos encontramos tanto yo como mi compañero Víctor recién incorporados con convenio de prácticas. Nuestro tutor de prácticas y CTO de la empresa en conjunto con la empresa cliente toman la decisión de arrancar este nuevo proyecto juntos.

1.1. Motivación

Este Trabajo de Fin de Grado (TFG) se enmarca en unas prácticas de empresa realizadas entre el 2 de febrero de 2022 y el 31 de agosto de 2022 en la empresa Dinacode S.L.

Tras participar en diversos proyectos basados en aplicaciones ya activas realizando tareas de mantenimiento. Junto con otro compañero, Víctor Ripoll Berenguer, nos hicimos cargo de un proyecto desde el comienzo de su desarrollo. Este nuevo reto nos permite poner en práctica todas las fases del desarrollo, desde el modelado hasta el despliegue y es en el que basaremos este TFG.

Otro aspecto importante de este proyecto es que no tiene partes implementadas por otras empresas y por lo tanto podemos aplicar enteramente las arquitecturas y diseños promovidos

por Dinacode, es decir, aplicar enteramente la Arquitectura Hexagonal [1] y Domain Driven Design [2]. Así podremos llegar a una mejor asimilación de los conceptos y poder aplicarlos mejor.

Aparte de esto también creemos que es una perfecta oportunidad para afianzar nuestros conocimientos en tecnologías muy populares hoy en día y que son usadas por empresas con bastante repercusión, a la par que las empresas que les dan soporte, como por ejemplo Google. Estas son Angular [3] para *frontend* (parte de la aplicación con la que interactúa el usuario), Spring [4] para *backend* (parte de la aplicación que actúa de servidor para proporcionar la información al usuario) e Hibernate [5] para mapeo con base de datos MySQL [6].

1.2. Objetivos

Actualmente, el proceso de solicitud de certificados de calidad es un trámite complicado por la excesiva cantidad de documentos a entregar y que se dilata mucho en el tiempo debido a los constantes rechazos de éstos y la longevidad de su aprobación.

Esto es lo que conlleva a plantear este proyecto, cuyo objetivo a *grosso modo*, es facilitar y agilizar este trámite mediante una aplicación web que permita a las empresas fabricantes de diversos productos llevar a cabo todo el proceso de solicitud de una manera sencilla y lo más intuitiva posible de forma completamente *online*.

La aplicación web estará dividida en dos contextos: el subsistema de usuario (empleado de una empresa productora de puertas que quiere obtener el certificado de calidad) y el subsistema del administrador (empleado de la empresa que gestiona la concesión de certificados de calidad). Por ello tenemos unos objetivos a cumplir relacionados con cada uno de los subsistemas.

Los objetivos del TFG en términos generales están centrados en desarrollar una base sólida del proyecto para poder gestionar la gran cantidad de datos que se manipulan en el *frontend* mediante formularios y se guardan en base de datos mediante una escalable y sólida arquitectura proporcionada por el *backend*.

Más concretamente, los objetivos del contexto del usuario son:

- El usuario podrá dar de alta las puertas, todos sus componentes, características y documentación asociada necesaria para la certificación.

Por lo que respecta al contexto del administrador:

- El administrador podrá gestionar los usuarios del sistema, es decir, poder llevar a cabo las operaciones CRUD (Create, Read, Update and Delete), acrónimo en inglés que hace alusión a las clásicas operaciones de crear, leer, actualizar y borrar. Estas operaciones serán aplicables sobre las cuentas de los usuarios y de las empresas a las qué están asociadas.
- El administrador podrá también gestionar algunos datos predefinidos de cara al usuario.

El proyecto abarca más objetivos futuros relativamente próximos como sería la solicitud por parte del cliente de los certificados de calidad y objetivos más lejanos como dar soporte a la solicitud de certificados de calidad de otros productos.

1.3. Metodología

La metodología escogida para el desarrollo del proyecto es de carácter ágil debido a que se basa en pequeñas entregas incrementales cada tres o cuatro semanas normalmente en las que va avanzando la cantidad de funciones implementadas y la calidad de éstas. Además, la carga de trabajo y la cantidad de tareas por sprint es variable, sujetas a las reuniones semanales entre las dos empresas. Además de mantener comunicación constante entre ambas empresas para el correcto y fluido desarrollo del proyecto.

1.4. Colaboraciones

Este trabajo, como he mencionado anteriormente, ha sido desarrollado en el contexto de las prácticas de empresa, en la cual tengo como compañero a Víctor Ripoll Berenguer y los dos

fuimos asignados a este proyecto para poder participar activamente en todo el proceso. Concretamente mi compañero se encarga de la parte *frontend* de la aplicación y yo de la parte *backend*. Asimismo, cada uno haremos la memoria de nuestra respectiva parte del trabajo.

2. Estado del arte

2.1. Introducción

El marcado CE, del francés “Conformité Européenne” y en español “de Conformidad Europea”, es una marca de la UE que se aplica a determinados productos destinados a ser comercializados en el Espacio Único Europeo (EEE) y en Turquía.

Ilustración 2.1 – Representación del marcado CE

Que un producto tenga el marcado CE supone que el fabricante asegura que dicho producto cumple con las leyes europeas sujetas al mismo, lo cual es necesario para su puesta en el mercado dentro del EEE (formado por los países miembros de la UE, Noruega, Liechtenstein y países de la AELC Islandia) además de Turquía, aunque el fabricante no sea perteneciente a cualquiera de estos países.

Legalmente, el marcado CE está ligado a la Directiva 93/68/CEE y no es necesario que todos los productos lo posean, solo lo deben llevar si pertenecen a categorías de productos sujetas a leyes que lo requieren. En adición, también es necesario que el fabricante de un producto marcado debe ser capaz de facilitar la documentación asociada a la evaluación de conformidad del producto, su expediente técnico y la declaración de conformidad CE firmada. [7]

2.2. Propuesta

Este proyecto, como he comentado anteriormente, lo estamos desarrollando en el contexto de nuestras prácticas de empresa, concretamente en la empresa Dinacode S.L. Esta empresa es una consultora software que se dedica a desarrollar software a medida de las necesidades de las empresas clientes eligiendo las tecnologías adecuadas para cumplir los objetivos del cliente. Aparte de esto también se dedica a asesorar a los equipos de desarrolladores de otras empresas.

Un factor bastante importante de Dinacode S.L. es que se caracteriza por usar *clean arquitectures* [8], una característica determinante para ser elegidos por empresas como la propietaria del proyecto que estamos desarrollando. Las *clean arquitectures*, entre otros beneficios que serán detallados en el apartado "Diseño y Arquitectura", nos ofrecen la posibilidad de crear un software escalable, lo que viene como anillo al dedo para el objetivo futuro del proyecto de dar soporte al marcado CE a otros productos aparte de las puertas.

El producto que estamos desarrollando aspira a ser una plataforma que proporcione una experiencia satisfactoria al usuario a través del proceso de dar de alta sus productos. Debido a la naturaleza del proceso es complicado que el usuario no se encuentre perdido al tener que dar de alta todas las entidades relacionadas con el producto en la web, a través de una alta cantidad de formularios y el complejo flujo de estos. Por lo tanto, nos enfocaremos en hacer este tedioso trámite más sencillo de recorrer para procurar una experiencia más amena al usuario en su cometido.

Hasta el momento no tenemos conocimiento de otros softwares que se ocupen de llevar a cabo esta tarea, lo que en principio podría ser una ventaja por falta de competencia, pero también partimos de cero, sin tener referencias anteriores para poder mejorarlas.

3. Especificación de requisitos

3.1. Introducción

A continuación, se detallará la especificación de requisitos del sistema basada en el formato IEEE 830-1998 [9].

3.1.1. Propósito

Este capítulo servirá para dar una visión global del comportamiento del sistema a través de la definición de los requisitos pactados entre la empresa cliente que solicita el proyecto y nosotros como empresa consultora con el fin de cumplir los objetivos descritos anteriormente.

3.1.2. Ámbito del sistema

El sistema que estamos desarrollando pretende proporcionar una web en la que los usuarios fabricantes de puertas puedan dar de alta en él los diferentes componentes necesarios para ensamblar las puertas y configurarlas para en un futuro poder solicitar su marcado CE dentro de la misma web para luego poder comercializarlas siguiendo la normativa europea.

Aparte de esto, el sistema busca dar soporte al otro tipo de usuario, el administrador de la web que tendrá control sobre las cuentas de los usuarios y las empresas productoras a las que pertenecen dentro del sistema. El administrador también podrá manejar opciones de componentes de puertas predeterminados.

Todas estas funciones se llevarán a cabo en una serie de interfaces similares en las que podrán por lo general: visualizar una tabla con las entidades disponibles, añadir una nueva entidad, modificarla y eliminarla.

3.2. Descripción general

A continuación, se describirá el contexto de los requisitos para su posterior explicación en la siguiente sección, Requisitos específicos.

3.2.1. Funciones del producto

En este apartado se expondrán las distintas funcionalidades que están a disposición por parte de los dos tipos usuarios del sistema: usuario y administrador.

El usuario administrador dispone de un panel de administración que le permite acceder a seis apartados principales: cuentas y contratos, puertas, mantenimiento, riesgo, contenido y configuración.

- **Cuentas y contratos**

Dentro de este apartado, el administrador podrá visualizar, crear, modificar y eliminar las cuentas de las compañías productoras de puertas, sus usuarios asociados y los contratos que las compañías mantienen con la empresa propietaria del sistema.

- **Puertas**

En esta sección podemos acceder a la gestión de todos los modelos relacionados con las puertas para poder visualizarlos en forma de tabla, añadir nuevos, modificar los existentes y eliminarlos. Estos modelos son: tipos de puerta, acabados, tipos de componentes, marcas, modelos de componentes, kits, configuración de puertas y laboratorios.

- **Mantenimiento**

Desde la sección de mantenimiento podemos llevar a cabo la gestión de las entidades relacionadas con el mantenimiento de las puertas. Podemos visualizar, editar, crear y eliminar operaciones de mantenimiento, periodos de mantenimiento y tipos de uso.

- **Riesgo**

En la sección de análisis de riesgo, como usuario administrador tendremos acceso a las operaciones CRUD de los modelos: análisis de riesgo, sus versiones, apartados, requisitos, soluciones y configuraciones. Aparte de esto podremos duplicar apartados, requisitos y soluciones para agilizar el proceso.

- **Contenido**

Dentro de esta sección podemos acceder a la gestión de noticias, ayudas y formaciones y normativas, esto es, crearlas y listarlas en forma de tabla para su visualización y así poder acceder a ellas para editarlas y eliminarlas.

- **Configuración**

Por último, en el apartado configuración podemos editar los datos de contacto y el resto de información asociada a la cuenta de administrador.

Por otro lado, está el usuario de las empresas fabricantes de productos. Este usuario básicamente pretende poder modificar los datos de su cuenta y tener el control sobre todas las entidades relacionadas con la producción de las puertas. Por ello, el usuario podrá acceder a una vista en la que puede editar los datos de su cuenta y a una por cada entidad que pueda manejar. También hay algunas de las vistas de manejo de entidades en las que se pueden manejar a su vez otras entidades anidadas, ya que las entidades principales de la vista las contienen en alguno de sus campos.

Así pues, estás entidades podrán ser visualizadas en forma de tabla, se podrán crear nuevas, editar las existentes y eliminarlas. Las entidades, agrupadas por vistas del usuario, son las siguientes:

- Puertas
- Clientes
- Modelos de puertas
- Certificados
    - Certificados de ensayo propios
    - Certificados de organismos certificados
- Componentes de puertas
- Usuarios
    - Usuarios
    - Puestos de trabajo
- Acabados
- Operaciones de mantenimiento
    - Operaciones
    - Puestos de configuraciones
- Laboratorios
- Equipos de trabajo
    - Equipos de trabajo
    - Tipos de equipos de trabajo

3.2.2. Restricciones

El sistema debe ser implementado con buenas prácticas de programación haciendo uso de los principios SOLID, arquitectura hexagonal y DDD. Estas arquitecturas y técnicas se deben complementar para conformar un sistema escalable acorde a los objetivos futuros del proyecto, que pone la mira en ampliar la gama de productos para los que proporcionara marcados CE.

3.3. Requisitos específicos

3.3.1. Interfaces externas

El sistema se caracteriza por funciones de registro de entidades y por tener un gran número de ellas, por lo que la mayoría de la aplicación está compuesta de tablas en las que se muestran las entidades y formularios para rellenar sus datos. Por lo que el requisito principal de las interfaces es qué el acceso a éstas esté bien clasificado por apartados de manejos de entidades a través de una barra de navegación y que el flujo de los formularios sea fácil de seguir por el usuario.

3.3.2. Funciones

Dicho esto, clasificaremos y describiremos los requisitos clasificados por los apartados de la barra de navegación del *frontend*, ya que es el orden en el que nos hemos basado para el desarrollo del *backend*.

- Sección “Cuentas y contratos”
    - Cuentas
        - Visualizar cuentas

Introducción	El administrador puede visualizar todas las cuentas registradas en el sistema en una tabla con buscador
Entrada	
Actores	Administrador
Proceso	El administrador pulsa en la subsección “Cuentas” de la sección “Cuentas y contratos” de la barra de navegación.
Salida	El administrador accede a la vista con todas las cuentas listadas en una tabla.

Tabla 3.1 – Visualizar cuentas

- Crear cuenta

Introducción	El administrador guarda la cuenta de una compañía cliente en el sistema
Entrada	Código de empresa, nombre, CIF, población, dirección, teléfono, fax, email, representante legal, cuenta bancaria, contrato, compañía, fabricante máster, acceso a la Api, prescriptor, estado
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Cuentas" de la sección "Cuentas y contratos" de la barra de navegación. Después pulsa en el botón "Añadir cuenta" de arriba de la tabla de compañías. Por último, rellena los campos del formulario y lo envía pulsando el botón "Crear".
Salida	La cuenta de la compañía se guarda en base de datos y el administrador es devuelto a la vista de la tabla de todas las compañías.

Tabla 3.2 – Crear cuenta

- Modificar cuenta

Introducción	El administrador actualiza los datos de la cuenta de una compañía cliente y los guarda en el sistema
Entrada	Código de empresa, nombre, CIF, población, dirección, teléfono, fax, email, representante legal, cuenta bancaria, contrato, compañía, fabricante máster, acceso a la Api, prescriptor, estado
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Cuentas" de la sección "Cuentas y contratos". A continuación, pulsa en el botón con el símbolo de editar de una compañía de la tabla. Así, accede al formulario de modificación relleno con los datos actuales de la compañía pudiendo editarlos y mandar los nuevos campos editados pulsando el botón "Modificar"
Salida	Los datos de la cuenta de la compañía se guardan en base de datos y el administrador es devuelto a la vista de la tabla de todas las compañías.

Tabla 3.3 – Modificar cuenta

- Eliminar cuenta

Introducción	El administrador elimina la cuenta de una compañía del sistema
Entrada	
Actores	Administrador
Proceso	El administrador pulsa en la subsección “Cuentas” de la sección “Cuentas y contratos”. A continuación, pulsa en el botón con el símbolo de editar de una compañía de la tabla. Por último, pulsa el botón “Eliminar” y vuelve a pulsar el botón “ELIMINAR” de la ventana emergente
Salida	La cuenta es eliminada de la base de datos y el administrador es devuelto a la vista de la tabla con todas las cuentas de las compañías

Tabla 3.3 – Eliminar cuenta

- Visualizar usuarios

Introducción	El administrador puede visualizar los usuarios pertenecientes a una compañía
Entrada	
Actores	Administrador
Proceso	El administrador pulsa en la subsección “Cuentas” de la sección “Cuentas y contratos” y pulsa en el botón con el símbolo de editar de una compañía de la tabla para poder visualizar una tabla con los usuarios pertenecientes a la compañía seleccionada
Salida	El administrador accede a la vista con todas las cuentas listadas en una tabla

Tabla 3.4 – Visualizar cuentas

- Crear usuario

Introducción	El administrador guarda un nuevo usuario perteneciente a una compañía en el sistema
Entrada	Nombre, código de usuario, email y contraseña
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Cuentas" de la sección "Cuentas y contratos" y pulsa en el botón con el símbolo de editar de una compañía de la tabla para poder visualizar una tabla con los usuarios pertenecientes a la compañía seleccionada. Seguidamente pulsa en el botón "Crear usuario" situado encima de la tabla, apareciendo así una ventana emergente con un formulario que el administrador rellena y envía pulsando el botón "Crear".
Salida	La cuenta del usuario de la compañía se guarda en base de datos y el administrador es devuelto a la vista de la tabla de todas las compañías.

Tabla 3.5 – Crear usuario

- Modificar usuario

Introducción	El administrador actualiza los datos de un usuario y los guarda en el sistema
Entrada	Nombre, código de usuario, email y contraseña
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Cuentas" de la sección "Cuentas y contratos" y pulsa en el botón con el símbolo de editar de una compañía de la tabla para poder visualizar una tabla con los usuarios pertenecientes a la compañía seleccionada. Seguidamente pulsa en el botón con el símbolo de editar de un usuario de la tabla, apareciendo así una ventana emergente con un formulario relleno con los datos actuales del usuario que el administrador puede modificar y enviar pulsando el botón "Modificar".
Salida	La cuenta del usuario de la compañía se guarda en base de datos actualizada y el administrador es devuelto a la vista de la tabla de todas las compañías.

Tabla 3.6 – Modificar usuario

- Tipos de contrato

    - Visualizar contratos

Introducción	El administrador puede visualizar todos los contratos registrados en el sistema en una tabla con buscador
Entrada	
Actores	Administrador
Proceso	El administrador pulsa en la subsección “Tipos de contrato” de la sección “Cuentas y contratos” de la barra de navegación.
Salida	El administrador accede a la vista con todos los contratos listados en una tabla.

Tabla 3.7 – Visualizar contratos

    - Crear contrato

Introducción	El administrador guarda un nuevo tipo de contrato en el sistema que después las compañías clientes podrán elegir
Entrada	Nombre, descripción, número máximo de puertas por mes, número máximo de puertas total, tarifa por mes, meses de cadencia y duración mínima en meses
Actores	Administrador
Proceso	El administrador pulsa en la subsección “Tipos de contrato” de la sección “Cuentas y contratos” de la barra de navegación. Después pulsa en el botón “Añadir contrato” de arriba de la tabla de compañías. Por último, rellena los campos del formulario y lo envía pulsando el botón “Crear”.
Salida	El tipo de contrato se guarda en base de datos y el administrador es devuelto a la vista de la tabla de todos los tipos de contrato.

Tabla 3.8 – Crear contrato

- Modificar contrato

Introducción	El administrador actualiza los datos de un contrato existente y los guarda en el sistema
Entrada	Nombre, descripción, número máximo de puertas por mes, número máximo de puertas total, tarifa por mes, meses de cadencia y duración mínima en meses
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Tipos de contrato" de la sección "Cuentas y contratos". A continuación, pulsa en el botón con el símbolo de editar de un contrato de la tabla. Así, accede al formulario de modificación relleno con los datos actuales del tipo de contrato pudiendo editarlos y mandar los nuevos campos actualizados pulsando el botón "Modificar"
Salida	Los datos del tipo de contrato se guardan en base de datos y el administrador es devuelto a la vista de la tabla de todos los tipos de contrato.

Tabla 3.9 – Modificar contrato

- Eliminar contrato

Introducción	El administrador elimina un tipo de contrato del sistema
Entrada	
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Tipos de contrato" de la sección "Cuentas y contratos". A continuación, pulsa en el botón con el símbolo de editar de un tipo de contrato de la tabla. Por último, pulsa el botón "Eliminar" y vuelve a pulsar el botón "ELIMINAR" de la ventana emergente
Salida	El tipo de contrato es eliminado de la base de datos y el administrador es devuelto a la vista de la tabla con todos los tipos de contrato

Tabla 3.10 – Eliminar contrato

- Sección “Puertas”
    - Tipos de puertas
        - Visualizar tipos de puerta

Introducción	El administrador puede visualizar todos los tipos de puerta registrados en el sistema en una tabla con buscador
Entrada	
Actores	Administrador
Proceso	El administrador pulsa en la subsección “Tipos de puerta” de la sección “Puertas” contenida en la barra de navegación.
Salida	El administrador accede a la vista con todos los tipos de puerta listados en una tabla.

Tabla 3.11 – Visualizar tipos de puerta

- Crear tipo de puerta

Introducción	El administrador guarda un nuevo tipo de puerta en el sistema
Entrada	Empresa, tipo de puerta, imagen del tipo de puerta, estado, imagen de los puntos de ensayo
Actores	Administrador
Proceso	El administrador pulsa en la subsección “Tipos de puerta” de la sección “Puertas” de la barra de navegación. Después pulsa en el botón “Añadir tipo de puerta” de arriba de la tabla de tipos de puerta. Por último, rellena los campos del formulario de la ventana emergente y lo envía pulsando el botón “Crear”.
Salida	El tipo de puerta se guarda en base de datos y la ventana emergente se cierra, devolviendo al administrador a la vista de la tabla de todos los tipos de puerta.

Tabla 3.12 – Crear tipo de puerta

- Modificar tipo de puerta

Introducción	El administrador actualiza los datos de un tipo de puerta existente y los guarda en el sistema
Entrada	Empresa, tipo de puerta, imagen del tipo de puerta, estado, imagen de los puntos de ensayo
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Tipos de puerta" de la sección "Puertas". A continuación, pulsa en el botón con el símbolo de editar de un tipo de puerta de la tabla. Así, accede al formulario de modificación en una ventana emergente relleno con los datos actuales del tipo de puerta pudiendo editarlos y mandar los nuevos campos actualizados pulsando el botón "Modificar"
Salida	Los datos del tipo de puerta se guardan en base de datos y la ventana emergente se cierra, devolviendo al administrador a la vista de la tabla de todos los tipos de puerta.

Tabla 3.13 – Modificar tipo de puerta

- Tipos de acabado

  - Visualizar tipos de acabado

Introducción	El administrador puede visualizar todos los tipos de acabado registrados en el sistema en una tabla con buscador
Entrada	
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Tipos de acabado" de la sección "Puertas" contenida en la barra de navegación.
Salida	El administrador accede a la vista con todos los tipos de acabado listados en una tabla.

Tabla 3.14 – Visualizar tipos de acabado

- Crear tipo de acabado

Introducción	El administrador guarda un nuevo tipo de acabado en el sistema
Entrada	Empresa, código, tipo de acabado, estado
Actores	Administrador
Proceso	El administrador pulsa en la subsección “Tipos de acabado” de la sección “Puertas” de la barra de navegación. Después pulsa en el botón “Añadir tipo de acabado” de arriba de la tabla de tipos de acabados. Por último, rellena los campos del formulario de la ventana emergente y lo envía pulsando el botón “Crear”.
Salida	El tipo de puerta se guarda en base de datos y la ventana emergente se cierra, devolviendo al administrador a la vista de la tabla de todos los tipos de puerta.

Tabla 3.15 – Crear tipo de acabado

- Modificar tipo de acabado

Introducción	El administrador actualiza los datos de un tipo de acabado existente y los guarda en el sistema
Entrada	Empresa, código, tipo de acabado, estado
Actores	Administrador
Proceso	El administrador pulsa en la subsección “Tipos de acabado” de la sección “Puertas”. A continuación, pulsa en el botón con el símbolo de editar de un tipo de acabado de la tabla. Así, accede al formulario de modificación en una ventana emergente relleno con los datos actuales del tipo de acabado pudiendo editarlos y mandar los nuevos campos actualizados pulsando el botón “Modificar”
Salida	Los datos del tipo de acabado se guardan en base de datos y la ventana emergente se cierra, devolviendo al administrador a la vista de la tabla de todos los tipos de acabado.

Tabla 3.16 – Modificar tipo de acabado

- Tipos de componente

  - Visualizar tipos de componente

Introducción	El administrador puede visualizar todos los tipos de componente registrados en el sistema en una tabla con buscador
Entrada	
Actores	Administrador
Proceso	El administrador pulsa en la subsección “Tipos de componente” de la sección “Puertas” contenida en la barra de navegación.
Salida	El administrador accede a la vista con todos los tipos de componente listados en una tabla.

Tabla 3.17 – Visualizar tipos de componente

  - Crear tipo de componente

Introducción	El administrador guarda un nuevo tipo de componente en el sistema
Entrada	Nombre, empresa, estado, general
Actores	Administrador
Proceso	El administrador pulsa en la subsección “Tipos de componente” de la sección “Puertas” de la barra de navegación. Después pulsa en el botón “Añadir tipo de componente” de arriba de la tabla de tipos de acabados. Por último, rellena los campos del formulario y lo envía pulsando el botón “Guardar y volver”.
Salida	El tipo de componente se guarda en base de datos y el administrador es devuelto a la vista de la tabla de todos los tipos de componente.

Tabla 3.18 – Crear tipo de componente

- Modificar tipo de componente

Introducción	El administrador actualiza los datos de un tipo de componente existente y los guarda en el sistema
Entrada	Nombre, empresa, estado y general
Actores	Administrador
Proceso	El administrador pulsa en la subsección “Tipos de componente” de la sección “Puertas”. A continuación, pulsa en el botón con el símbolo de editar de un tipo de componente de la tabla. Así, accede al formulario de modificación relleno con los datos actuales del tipo de acabado pudiendo editarlos y mandar los nuevos campos actualizados pulsando el botón “Guardar y volver”
Salida	Los datos del tipo de componente se guardan en base de datos y la ventana emergente se cierra, devolviendo al administrador a la vista de la tabla de todos los tipos de componente.

Tabla 3.19 – Modificar tipo de componente

- Marcas

  - Visualizar marcas

Introducción	El administrador puede visualizar todas las marcas registradas en el sistema en una tabla con buscador
Entrada	
Actores	Administrador
Proceso	El administrador pulsa en la subsección “Marcas” de la sección “Puertas” contenida en la barra de navegación.
Salida	El administrador accede a la vista con todas las marcas listadas en una tabla.

Tabla 3.20 – Visualizar marcas

- Crear marca

Introducción	El administrador guarda una nueva marca en el sistema
Entrada	Empresa, nombre de la marca, estado y general
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Marcas" de la sección "Puertas" de la barra de navegación. Después pulsa en el botón "Añadir marca" de arriba de la tabla de marcas. Por último, rellena los campos del formulario de la ventana emergente y lo envía pulsando el botón "Crear Marca".
Salida	La marca se guarda en base de datos y la ventana emergente se cierra, devolviendo al administrador a la vista de la tabla de todas las marcas.

Tabla 3.21 – Crear marca

- Modificar marca

Introducción	El administrador actualiza los datos de una marca existente y los guarda en el sistema
Entrada	Empresa, nombre de la marca, estado y general
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Marcas" de la sección "Puertas". A continuación, pulsa en el botón con el símbolo de editar de una marca de la tabla. Así, accede al formulario de modificación en una ventana emergente relleno con los datos actuales de la marca pudiendo editarlos y mandar los nuevos campos actualizados pulsando el botón "Modificar marca"
Salida	Los datos de la marca se guardan en base de datos y la ventana emergente se cierra, devolviendo al administrador a la vista de la tabla de todas las marcas.

Tabla 3.22 – Modificar marca

- Modelos de componente

  - Visualizar modelos de componente

Introducción	El administrador puede visualizar todos los modelos de componente registrados en el sistema en una tabla con buscador
Entrada	
Actores	Administrador
Proceso	El administrador pulsa en la subsección “Modelos de componente” de la sección “Puertas” contenida en la barra de navegación.
Salida	El administrador accede a la vista con todos los modelos de componente listados en una tabla.

Tabla 3.23 – Visualizar modelos de componente

  - Crear modelo de componente

Introducción	El administrador guarda un nuevo modelo de componente en el sistema
Entrada	Nombre, tipo de puerta, componente, marca, estado y general
Actores	Administrador
Proceso	El administrador pulsa en la subsección “Modelos de componente” de la sección “Puertas” de la barra de navegación. Después pulsa en el botón “Añadir modelo de componente” de arriba de la tabla de tipos de acabados. Por último, rellena los campos del formulario y lo envía pulsando el botón “Guardar y volver”.
Salida	El modelo de componente se guarda en base de datos y el administrador es devuelto a la vista de la tabla de todos los modelos de componente.

Tabla 3.24 – Crear modelo de componente

- Modificar modelo de componente

Introducción	El administrador actualiza los datos de un modelo de componente existente y los guarda en el sistema
Entrada	Nombre, tipo de puerta, componente, marca, estado y general
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Modelos de componente" de la sección "Puertas". A continuación, pulsa en el botón con el símbolo de editar de un modelo de componente de la tabla. Así, accede al formulario de modificación relleno con los datos actuales del modelo de acabado pudiendo editarlos y mandar los nuevos campos actualizados pulsando el botón "Guardar y volver"
Salida	Los datos del modelo de componente se guardan en base de datos y el administrador es devuelto a la vista de la tabla de todos los modelos de componente.

Tabla 3.25 – Modificar modelo de componente

- Kits

  - Visualizar kits

Introducción	El administrador puede visualizar todos los kits registrados en el sistema en una tabla con buscador y filtros por nombre y tipo de puerta
Entrada	
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Kits" de la sección "Puertas" contenida en la barra de navegación.
Salida	El administrador accede a la vista con todos los kits listados en una tabla.

Tabla 3.26 – Visualizar kits

- Crear kit

Introducción	El administrador guarda un nuevo kit en el sistema
Entrada	Nombre, tipo de puerta, empresa y estado
Actores	Administrador
Proceso	El administrador pulsa en la subsección “Kits” de la sección “Puertas” de la barra de navegación. Después pulsa en el botón “Añadir kit” de arriba de la tabla de tipos de acabados. Por último, rellena los campos del formulario y lo envía pulsando el botón “Guardar y volver”.
Salida	El kit se guarda en base de datos y el administrador es devuelto a la vista de la tabla de todos los modelos de componente.

Tabla 3.27 – Crear kit

- Modificar kit

Introducción	El administrador actualiza los datos de un kit existente y los guarda en el sistema
Entrada	Nombre, tipo de puerta, empresa y estado
Actores	Administrador
Proceso	El administrador pulsa en la subsección “Kits” de la sección “Puertas”. A continuación, pulsa en el botón con el símbolo de editar de un kit de la tabla. Así, accede al formulario de modificación relleno con los datos actuales del kit pudiendo editarlos y mandar los nuevos campos actualizados pulsando el botón “Guardar y volver”
Salida	Los datos del kit se guardan en base de datos y la ventana emergente se cierra, devolviendo al administrador a la vista de la tabla de todos los kits.

Tabla 3.28 – Modificar kit

- Configuración de puertas

  - Visualizar configuraciones de puertas

Introducción	El administrador puede visualizar todas las configuraciones de puertas registradas en el sistema en una tabla con buscador
Entrada	
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Configuración de puertas" de la sección "Puertas" contenida en la barra de navegación.
Salida	El administrador accede a la vista con todas las configuraciones de puertas listadas en una tabla.

Tabla 3.29 – Visualizar configuraciones de puertas

  - Crear configuración de puerta

Introducción	El administrador guarda una nueva configuración de puerta en el sistema
Entrada	Origen de puerta, grupo de puerta, tipo de puerta, tipo de uso, tipo de impulso, motorizada, puerta peatonal, frecuencia de uso, hojas móviles y hojas fijas
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Configuraciones de puertas" de la sección "Puertas" de la barra de navegación. Después pulsa en el botón "Añadir configuración de puerta" de arriba de la tabla de configuraciones de puertas. Por último, rellena los campos del formulario y lo envía pulsando el botón "Crear".
Salida	La configuración de puerta se guarda en base de datos y el administrador es devuelto a la vista de la tabla de todas las configuraciones de puertas.

Tabla 3.30 – Crear configuración de puerta

- Modificar configuración de puerta

Introducción	El administrador actualiza los datos de una configuración de puerta existente y los guarda en el sistema
Entrada	Origen de puerta, grupo de puerta, tipo de puerta, tipo de uso, tipo de impulso, motorizada, puerta peatonal, frecuencia de uso, hojas móviles y hojas fijas
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Configuraciones de puertas" de la sección "Puertas". A continuación, pulsa en el botón con el símbolo de editar de una configuración de puerta de la tabla. Así, accede al formulario de modificación relleno con los datos actuales de la configuración de puerta pudiendo editarlos y mandar los nuevos campos actualizados pulsando el botón "Modificar"
Salida	Los datos de la configuración de puerta se guardan en base de datos y el administrador es devuelto a la vista de la tabla de todas las configuraciones de puerta.

Tabla 3.31 – Modificar configuración de puerta

- Eliminar configuración de puerta

Introducción	El administrador elimina una configuración de puerta del sistema
Entrada	
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Configuraciones de puertas" de la sección "Puertas". A continuación, pulsa en el botón con el símbolo de editar de un tipo de contrato de la tabla. Por último, pulsa el botón "Eliminar" y vuelve a pulsar el botón "ELIMINAR" de la ventana emergente
Salida	La configuración de puerta es eliminada de la base de datos y el administrador es devuelto a la vista de la tabla con todas las configuraciones de puertas

Tabla 3.32 – Eliminar configuración de puerta

- Sección “Mantenimiento”
    - Operaciones de mantenimiento
        - Visualizar operaciones de mantenimiento

Introducción	El administrador puede visualizar todas las operaciones de mantenimiento registradas en el sistema en una tabla con buscador
Entrada	
Actores	Administrador
Proceso	El administrador pulsa en la subsección “Operaciones de mantenimiento” de la sección “Mantenimiento” contenida en la barra de navegación.
Salida	El administrador accede a la vista con todas las operaciones de mantenimiento listadas en una tabla.

Tabla 3.33 – Visualizar operaciones de mantenimiento

- Crear operación de mantenimiento

Introducción	El administrador guarda una nueva operación de mantenimiento en el sistema
Entrada	Empresa, operación y periodo
Actores	Administrador
Proceso	El administrador pulsa en la subsección “Operaciones de mantenimiento” de la sección “Mantenimiento” de la barra de navegación. Después pulsa en el botón “Añadir operación de mantenimiento” de arriba de la tabla de operaciones de mantenimiento. Por último, rellena los campos del formulario y lo envía pulsando el botón “Crear”.
Salida	La configuración de puerta se guarda en base de datos y el administrador es devuelto a la vista de la tabla de todas las operaciones de mantenimiento.

Tabla 3.34 – Crear operación de mantenimiento

- Modificar operación de mantenimiento

Introducción	El administrador actualiza los datos de una operación de mantenimiento existente y los guarda en el sistema
Entrada	Empresa, operación y periodo
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Operaciones de mantenimiento" de la sección "Puertas". A continuación, pulsa en el botón con el símbolo de editar de una operación de mantenimiento de la tabla. Así, accede al formulario de modificación relleno con los datos actuales de la operación de mantenimiento pudiendo editarlos y mandar los nuevos campos actualizados pulsando el botón "Modificar"
Salida	Los datos de la operación de mantenimiento se guardan en base de datos y el administrador es devuelto a la vista de la tabla de todas las operaciones de mantenimiento

Tabla 3.35 – Modificar operación de mantenimiento

- Periodos de mantenimiento

  - Visualizar periodos de mantenimiento

Introducción	El administrador puede visualizar todos los periodos de mantenimiento registrados en el sistema en una tabla con buscador
Entrada	
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Periodos de mantenimiento" de la sección "Mantenimiento" contenida en la barra de navegación.
Salida	El administrador accede a la vista con todos los periodos de mantenimiento listados en una tabla.

Tabla 3.36 – Visualizar periodos de mantenimiento

- Crear periodo de mantenimiento

Introducción	El administrador guarda un nuevo periodo de mantenimiento en el sistema
Entrada	Tipo de periodo, orden, periodo y modo
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Periodos de mantenimiento" de la sección "Mantenimiento" de la barra de navegación. Después pulsa en el botón "Añadir periodo de mantenimiento" de arriba de la tabla de periodos de mantenimiento. Por último, rellena los campos del formulario de la ventana emergente y lo envía pulsando el botón "Guardar".
Salida	El periodo de mantenimiento se guarda en base de datos y la ventana emergente se cierra, devolviendo al administrador a la vista de la tabla de todos los periodos de mantenimiento.

Tabla 3.37 – Crear periodo de mantenimiento

- Modificar periodo de mantenimiento

Introducción	El administrador actualiza los datos de un periodo de mantenimiento existente y los guarda en el sistema
Entrada	Tipo de periodo, orden, periodo y modo
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Periodos de mantenimiento" de la sección "Mantenimiento". A continuación, pulsa en el botón con el símbolo de editar de una marca de la tabla. Así, accede al formulario de modificación en una ventana emergente relleno con los datos actuales del periodo de mantenimiento pudiendo editarlos y mandar los nuevos campos actualizados pulsando el botón "Guardar"
Salida	Los datos del periodo de mantenimiento se guardan en base de datos y la ventana emergente se cierra, devolviendo al administrador a la vista de la tabla de todos los periodos de mantenimiento.

Tabla 3.38 – Modificar periodo de mantenimiento

- Tipos de uso

  - Visualizar tipos de uso

Introducción	El administrador puede visualizar todos los tipos de uso registrados en el sistema en una tabla con buscador
Entrada	
Actores	Administrador
Proceso	El administrador pulsa en la subsección “Tipos de uso” de la sección “Mantenimiento” contenida en la barra de navegación.
Salida	El administrador accede a la vista con todos los tipos de uso listados en una tabla.

Tabla 3.39 – Visualizar tipos de uso

  - Crear tipos de uso

Introducción	El administrador guarda un nuevo tipo de uso en el sistema
Entrada	Tipo de uso
Actores	Administrador
Proceso	El administrador pulsa en la subsección “Tipos de uso” de la sección “Mantenimiento” de la barra de navegación. Después pulsa en el botón “Añadir tipo de uso” de arriba de la tabla de periodos de mantenimiento. Por último, rellena los campos del formulario de la ventana emergente y lo envía pulsando el botón “Guardar”.
Salida	El tipo de uso se guarda en base de datos y la ventana emergente se cierra, devolviendo al administrador a la vista de la tabla de todos los tipos de uso.

Tabla 3.40 – Crear tipos de uso

- Modificar tipo de uso

Introducción	El administrador actualiza los datos de un periodo de mantenimiento existente y los guarda en el sistema
Entrada	Tipo de uso
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Tipos de uso" de la sección "Mantenimiento". A continuación, pulsa en el botón con el símbolo de editar de un tipo de uso de la tabla. Así, accede al formulario de modificación en una ventana emergente relleno con los datos actuales del tipo de uso pudiendo editarlos y mandar los nuevos campos actualizados pulsando el botón "Guardar"
Salida	Los datos del tipo de uso se guardan en base de datos y la ventana emergente se cierra, devolviendo al administrador a la vista de la tabla de todos los tipos de uso

Tabla 3.41 – Modificar tipo de uso

- Sección "Contenido"
  - Noticias
    - Visualizar noticias

Introducción	El administrador puede visualizar todas las noticias registradas en el sistema en una tabla con buscador
Entrada	
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Noticias" de la sección "Contenido" contenida en la barra de navegación.
Salida	El administrador accede a la vista con todas las noticias listadas en una tabla.

Tabla 3.42 – Visualizar noticias

- Crear noticia

Introducción	El administrador guarda una nueva noticia en el sistema
Entrada	Título de la noticia, descripción, enlace, fecha, orden y estado
Actores	Administrador
Proceso	El administrador pulsa en la subsección “Noticias” de la sección “Contenido” de la barra de navegación. Después pulsa en el botón “Añadir noticia” de arriba de la tabla de noticias. Por último, rellena los campos del formulario y lo envía pulsando el botón “Crear”.
Salida	La noticia se guarda en base de datos y el administrador es devuelto a la vista de la tabla de todas las noticias.

Tabla 3.43 – Crear noticia

- Modificar noticia

Introducción	El administrador actualiza los datos de una noticia existente y los guarda en el sistema
Entrada	Título de la noticia, descripción, enlace, fecha, orden y estado
Actores	Administrador
Proceso	El administrador pulsa en la subsección “Noticias” de la sección “Contenido”. A continuación, pulsa en el botón con el símbolo de editar de una noticia de la tabla. Así, accede al formulario de modificación relleno con los datos actuales de la noticia pudiendo editarlos y mandar los nuevos campos actualizados pulsando el botón “Modificar”
Salida	Los datos de la noticia se guardan en base de datos y el administrador es devuelto a la vista de la tabla de todas las noticias

Tabla 3.44 – Modificar noticia

- Noticias

  - Visualizar formaciones

Introducción	El administrador puede visualizar todas las formaciones registradas en el sistema en una tabla con buscador
Entrada	
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Formaciones" de la sección "Contenido" contenida en la barra de navegación.
Salida	El administrador accede a la vista con todas las formaciones listadas en una tabla.

Tabla 3.45 – Visualizar formaciones

  - Crear formación

Introducción	El administrador guarda una nueva formación en el sistema
Entrada	Título, tipo, agrupación, enlace, orden y estado
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Formaciones" de la sección "Contenido" de la barra de navegación. Después pulsa en el botón "Añadir formación" de arriba de la tabla de formaciones. Por último, rellena los campos del formulario y lo envía pulsando el botón "Crear".
Salida	La formación se guarda en base de datos y el administrador es devuelto a la vista de la tabla de todas las formaciones.

Tabla 3.46 – Crear formación

- Modificar formación

Introducción	El administrador actualiza los datos de una formación existente y los guarda en el sistema
Entrada	Título, tipo, agrupación, enlace, orden y estado
Actores	Administrador
Proceso	El administrador pulsa en la subsección “Formaciones” de la sección “Contenido”. A continuación, pulsa en el botón con el símbolo de editar de una formación de la tabla. Así, accede al formulario de modificación relleno con los datos actuales de la formación pudiendo editarlos y mandar los nuevos campos actualizados pulsando el botón “Modificar”
Salida	Los datos de la formación se guardan en base de datos y el administrador es devuelto a la vista de la tabla de todas las formaciones

Tabla 3.47 – Modificar formación

- Toma de datos

  - Visualizar tomas de datos

Introducción	El administrador puede visualizar todas las tomas de datos registradas en el sistema en una tabla con buscador
Entrada	
Actores	Administrador
Proceso	El administrador pulsa en la subsección “Toma de datos” de la sección “Contenido” contenida en la barra de navegación.
Salida	El administrador accede a la vista con todas las tomas de datos listadas en una tabla.

Tabla 3.48 – Visualizar tomas de datos

- Crear toma de datos

Introducción	El administrador guarda una nueva toma de datos en el sistema
Entrada	Título, tipo, documento, orden y estado
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Toma de datos" de la sección "Contenido" de la barra de navegación. Después pulsa en el botón "Añadir toma de datos" de arriba de la tabla de toma de datos. Por último, rellena los campos del formulario y lo envía pulsando el botón "Crear".
Salida	La toma de datos se guarda en base de datos y el administrador es devuelto a la vista de la tabla de todas las tomas de datos.

Tabla 3.49 – Crear toma de datos

- Modificar toma de datos

Introducción	El administrador actualiza los datos de una toma de datos existente y los guarda en el sistema
Entrada	Título, tipo, documento, orden y estado
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Toma de datos" de la sección "Contenido". A continuación, pulsa en el botón con el símbolo de editar de una toma de datos de la tabla. Así, accede al formulario de modificación relleno con los datos actuales de la toma de datos pudiendo editarlos y mandar los nuevos campos actualizados pulsando el botón "Modificar"
Salida	Los datos de la toma de datos se guardan en base de datos y el administrador es devuelto a la vista de la tabla de todas las tomas de datos

Tabla 3.50 – Modificar toma de datos

- Normativa

  - Visualizar normativas

Introducción	El administrador puede visualizar todas las normativas registradas en el sistema en una tabla con buscador
Entrada	
Actores	Administrador
Proceso	El administrador pulsa en la subsección “Normativa” de la sección “Contenido” contenida en la barra de navegación.
Salida	El administrador accede a la vista con todas las normativas listadas en una tabla.

Tabla 3.51 – Visualizar normativas

  - Crear normativa

Introducción	El administrador guarda una nueva normativa en el sistema
Entrada	Título, tipo, documento, orden y estado
Actores	Administrador
Proceso	El administrador pulsa en la subsección “Normativa” de la sección “Contenido” de la barra de navegación. Después pulsa en el botón “Añadir normativa” de arriba de la tabla de normativas. Por último, rellena los campos del formulario y lo envía pulsando el botón “Crear”.
Salida	La normativa se guarda en base de datos y el administrador es devuelto a la vista de la tabla de todas las normativas.

Tabla 3.52 – Crear normativa

- Modificar normativa

Introducción	El administrador actualiza los datos de una normativa existente y los guarda en el sistema
Entrada	Título, tipo, documento, orden y estado
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Normativa" de la sección "Contenido". A continuación, pulsa en el botón con el símbolo de editar de una normativa de la tabla. Así, accede al formulario de modificación relleno con los datos actuales de la normativa pudiendo editarlos y mandar los nuevos campos actualizados pulsando el botón "Modificar"
Salida	Los datos de la normativa se guardan en base de datos y el administrador es devuelto a la vista de la tabla de todas las normativas

Tabla 3.53 – Modificar normativa

- Sección "Riesgo"
  - Análisis de riesgo clásico
    - Apartados
      - Visualizar apartados

Introducción	El administrador puede visualizar todos los apartados de análisis de riesgos clásicos registrados en el sistema en una tabla con buscador
Entrada	
Actores	Administrador
Proceso	El administrador pulsa en el botón "Apartados" dentro de la subsección "Análisis de riesgo clásico" de la sección "Riesgo" contenida en la barra de navegación.
Salida	El administrador accede a la vista con todos los apartados listados en una tabla.

Tabla 3.54 – Visualizar apartados

- Crear apartado

Introducción	El administrador guarda un nuevo apartado de análisis de riesgo clásico en el sistema
Entrada	Nivel del apartado, orden, apartado y norma de la UNE
Actores	Administrador
Proceso	El administrador pulsa en el botón “Apartados” dentro de la subsección “Análisis de riesgo clásico” de la sección “Riesgo” contenida en la barra de navegación. Después pulsa en el botón “Añadir apartado” de arriba de la tabla de apartados. Por último, rellena los campos del formulario y lo envía pulsando el botón “Crear”.
Salida	El apartado se guarda en base de datos y el administrador es devuelto a la vista de la tabla de todos los apartados.

Tabla 3.55 – Crear apartado

- Modificar apartado

Introducción	El administrador actualiza los datos de un apartado de riesgo clásico existente y los guarda en el sistema
Entrada	Nivel del apartado, orden, apartado y norma de la UNE
Actores	Administrador
Proceso	El administrador pulsa en el botón “Apartados” dentro de la subsección “Análisis de riesgo clásico” de la sección “Riesgo”. A continuación, pulsa en el botón con el símbolo de editar de un apartado de la tabla. Así, accede al formulario de modificación relleno con los datos actuales del apartado pudiendo editarlos y mandar los nuevos campos actualizados pulsando el botón “Modificar”
Salida	Los datos del apartado se guardan en base de datos y el administrador es devuelto a la vista de la tabla de todos los apartados

Tabla 3.56 – Modificar apartado

- Preguntas

  - Visualizar preguntas

Introducción	El administrador puede visualizar todas las preguntas de análisis de riesgos clásicos registrados en el sistema en una tabla con buscador
Entrada	
Actores	Administrador
Proceso	El administrador pulsa en el botón “Preguntas” dentro de la subsección “Análisis de riesgo clásico” de la sección “Riesgo” contenida en la barra de navegación.
Salida	El administrador accede a la vista con todas las preguntas listadas en una tabla.

Tabla 3.57 – Visualizar preguntas

  - Crear pregunta

Introducción	El administrador guarda una nueva pregunta de análisis de riesgo clásico en el sistema
Entrada	Tipo de puerta, apartado, orden de la pregunta, grupo, cabecera y pregunta
Actores	Administrador
Proceso	El administrador pulsa en el botón “Preguntas” dentro de la subsección “Análisis de riesgo clásico” de la sección “Riesgo” contenida en la barra de navegación. Después pulsa en el botón “Añadir pregunta” de arriba de la tabla de preguntas. Por último, rellena los campos del formulario y lo envía pulsando el botón “Crear”.
Salida	La pregunta se guarda en base de datos y el administrador es devuelto a la vista de la tabla de todas las preguntas

Tabla 3.58 – Crear pregunta

- Modificar pregunta

Introducción	El administrador actualiza los datos de una pregunta de riesgo clásico existente y los guarda en el sistema
Entrada	Tipo de puerta, apartado, orden de la pregunta, grupo, cabecera y pregunta
Actores	Administrador
Proceso	El administrador pulsa en el botón "Preguntas" dentro de la subsección "Análisis de riesgo clásico" de la sección "Riesgo". A continuación, pulsa en el botón con el símbolo de editar de un apartado de la tabla. Así, accede al formulario de modificación relleno con los datos actuales de la pregunta pudiendo editarlos y mandar los nuevos campos actualizados pulsando el botón "Modificar"
Salida	Los datos de la pregunta se guardan en base de datos y el administrador es devuelto a la vista de la tabla de todas las preguntas

Tabla 3.59 – Modificar pregunta

- Sección "Configuración"
  - Ajustes del panel
    - Modificar datos del administrador

Introducción	El administrador actualiza los datos de su cuenta de administrador y los guarda en el sistema
Entrada	Nombre, CIF, población, dirección, código postal, teléfono, fax, email, representante legal, cuenta bancaria, *login* de la empresa y marcas
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Ajustes del panel" de la sección "Configuración". Así, accede al formulario de modificación relleno con los datos actuales del administrador pudiendo editarlos y mandar los nuevos campos actualizados pulsando el botón "Guardar"
Salida	Los nuevos datos del administrador se guardan en base de datos

Tabla 3.60 – Modificar datos del administrador

- Códigos postales

  - Visualizar códigos postales

Introducción	El administrador puede visualizar todos los códigos postales registrados en el sistema en una tabla con buscador
Entrada	
Actores	Administrador
Proceso	El administrador pulsa en la subsección “Códigos postales” de la sección “Configuración” contenida en la barra de navegación.
Salida	El administrador accede a la vista con todos los códigos postales listados en una tabla.

Tabla 3.61 – Visualizar códigos postales

  - Crear código postal

Introducción	El administrador guarda un nuevo código postal en el sistema
Entrada	Código postal, población, provincia y país
Actores	Administrador
Proceso	El administrador pulsa en la subsección “Códigos postales” de la sección “Configuración” de la barra de navegación. Después pulsa en el botón “Añadir código postal” de arriba de la tabla de códigos postales. Por último, rellena los campos del formulario y lo envía pulsando el botón “Guardar”.
Salida	El código postal se guarda en base de datos y el administrador es devuelto a la vista de la tabla de todos los códigos postales.

Tabla 3.62 – Crear código postal

- Modificar código postal

Introducción	El administrador actualiza los datos de un código postal existente y los guarda en el sistema
Entrada	Código postal, población, provincia y país
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Código postal" de la sección "Configuración". A continuación, pulsa en el botón con el símbolo de editar de un código postal de la tabla. Así, accede al formulario de modificación relleno con los datos actuales del código postal pudiendo editarlos y mandar los nuevos campos actualizados pulsando el botón "Modificar"
Salida	Los datos del código postal se guardan en base de datos y el administrador es devuelto a la vista de la tabla de todos los códigos postales

Tabla 3.63 – Modificar código postal

- Crear provincia

Introducción	El administrador guarda una nueva provincia en el sistema
Entrada	Código, provincia y país
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Códigos postales" de la sección "Configuración" de la barra de navegación. Después pulsa en el botón "Añadir provincia" del encabezado de la vista. Por último, rellena los campos del formulario y lo envía pulsando el botón "Añadir".
Salida	La provincia se guarda en base de datos y el administrador es devuelto a la vista de la tabla de todos los códigos postales.

Tabla 3.64 – Crear provinica

- Visualizar provincias

Introducción	El administrador puede visualizar todas las provincias registradas en el sistema en una tabla con buscador
Entrada	
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Códigos postales" de la sección "Configuración" contenida en la barra de navegación y pulsar en el botón "Añadir provincia" en la vista a la que es redirigido.
Salida	El administrador accede a la vista del formulario para crear una nueva provincia y debajo de este estará la tabla con todas las provincias registradas en el sistema

Tabla 3.65 – Visualizar provincias

- Modificar provincia

Introducción	El administrador actualiza los datos de una provincia existente y los guarda en el sistema
Entrada	Código, provincia y país
Actores	Administrador
Proceso	El administrador pulsa en la subsección "Códigos postales" de la sección "Configuración" contenida en la barra de navegación. Después debe pulsar en el botón "Añadir provincia" en la vista a la que es redirigido. Esto le llevará a la vista con la tabla de provincias y el formulario para crear una provincia. En esta vista el administrador pulsa en el botón de editar correspondiente a una provincia de la tabla. A continuación, los datos del formulario vacío se rellanarán con los de la provincia seleccionada, pudiendo editarlos y guardarlos en base de datos pulsando el botón "Editar"
Salida	Los datos de la provincia se guardan en base de datos y el administrador es devuelto a la vista de la tabla de todos los códigos postales

Tabla 3.63 – Modificar código postal

Cabe recalcar que en el *frontend* pueden diferir algunos casos de uso como los que sirven para duplicar algún modelo, esto se debe a que en el *backend* no supone un caso de uso nuevo, ya que realmente es un caso de uso de crear un nuevo modelo en el sistema adornado para facilitar la tarea al usuario.

4. Análisis

4.1. Introducción

Este capítulo contiene el análisis del sistema a desarrollar. En primer lugar, se describirán los actores que interactúan con el sistema, para luego poder relacionarlos con los casos de uso que pueden llevar a cabo y finalmente definir un diagrama de clases que nos proporciona la información de los modelos utilizados en el sistema y como se relacionan entre ellos.

Este análisis se llevará a cabo siguiendo el estándar UML [10], el lenguaje más extendido de modelado de sistemas en la actualidad, con el que podremos documentar los casos de uso y el diagrama de clases.

4.2. Actores

En nuestro sistema, como hemos podido apreciar en la especificación de requisitos, existen dos tipos de usuarios:

- Administrador:

El usuario administrador representa al empleado de la empresa poseedora del sistema y cuyas funciones son gestionar las cuentas de los clientes, así como el tipo de contrato adquirido; las puertas y entidades relacionadas con éstas; los mantenimientos de las puertas; sus análisis de riesgo; contenido y configurar la cuenta propia de administrador.

- Usuario:

Con usuario nos referimos al usuario cliente del sistema, que representa a una empresa productora de puertas. Este usuario contrata una tarifa con la empresa propietaria del sistema y en él puede dar de alta puertas, con todas sus entidades relacionadas, manejar los clientes de la empresa a la que representa y configurar su cuenta.

4.3. Casos de uso

Los actores descritos en el apartado anterior interactúan con el sistema mediante casos de uso. A continuación, mostramos los casos de uso que puede llevar a cabo el actor "Administrador" en el sistema. Separado en varios diagramas por su complejidad, pero representando uno solo.

Ilustración 4.1 - Diagrama de casos de uso 1

Ilustración 4.2 - Diagrama de casos de uso 2

Ilustración 4.3 - Diagrama de casos de uso 3

En estos tres diagramas podemos observar como todos ellos comparten el mismo actor, que se relaciona mediante las líneas simples con los óvalos azules que representan los casos de uso que puede llevar a cabo interactuando con el sistema a través de la web. Estos a su vez se relacionan entre sí con relaciones representadas por flechas discontinuas y el nombre de la relación por encima: “extends” e “includes”. En el caso de la relación “extends”, indicando que el caso de uso de la punta de la flecha de la relación puede desembocar en otro caso de uso si el actor así lo desea (desde el que sale la flecha en el diagrama), pero sin ser obligatoria su ejecución. Por otro lado, tenemos la relación “includes” que significa que el caso de uso desde el que sale la flecha no se puede ejecutar sin llevar a cabo el caso de uso al que apunta la flecha.

4.4. Diagrama de clases

Para acabar con el análisis, estableceremos las entidades del dominio del sistema mediante un diagrama de clases UML, con clases que corresponden a las entidades y donde se pueden apreciar las relaciones entre ellas representadas con asociaciones y sus correspondientes multiplicidades.

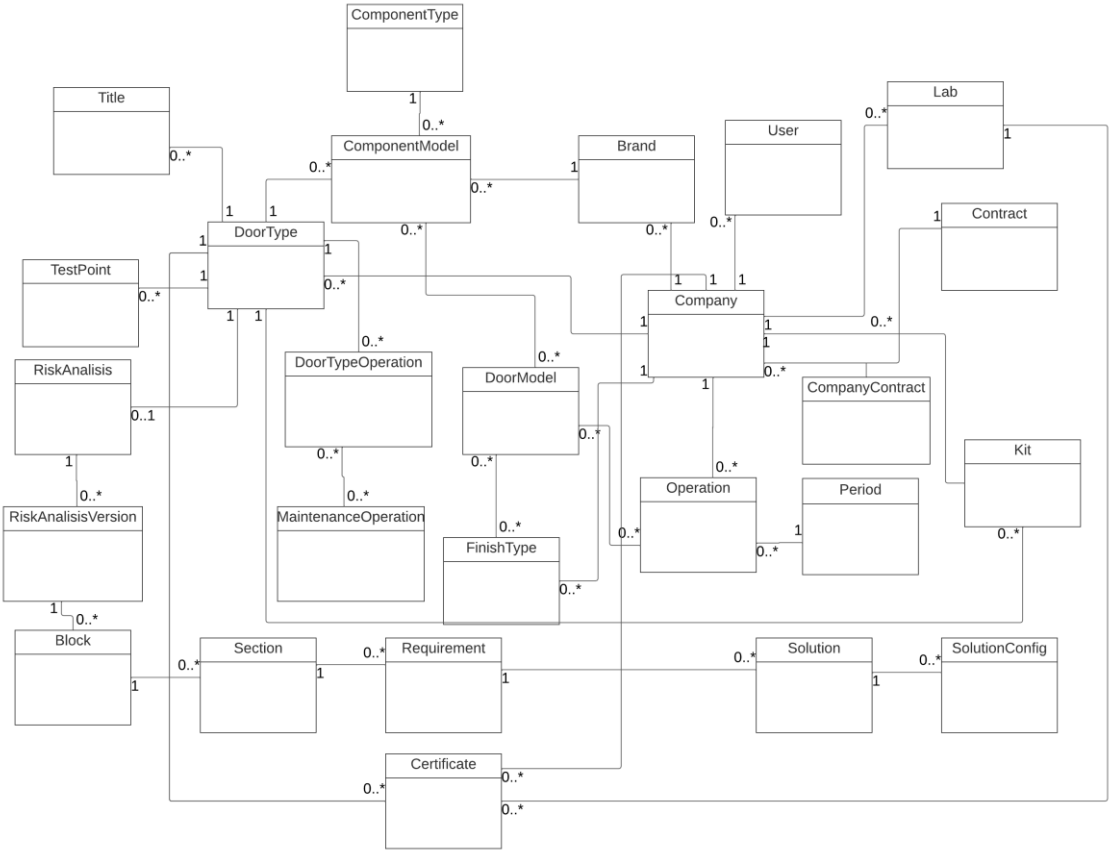

Ilustración 4.4 – Diagrama de clases 1

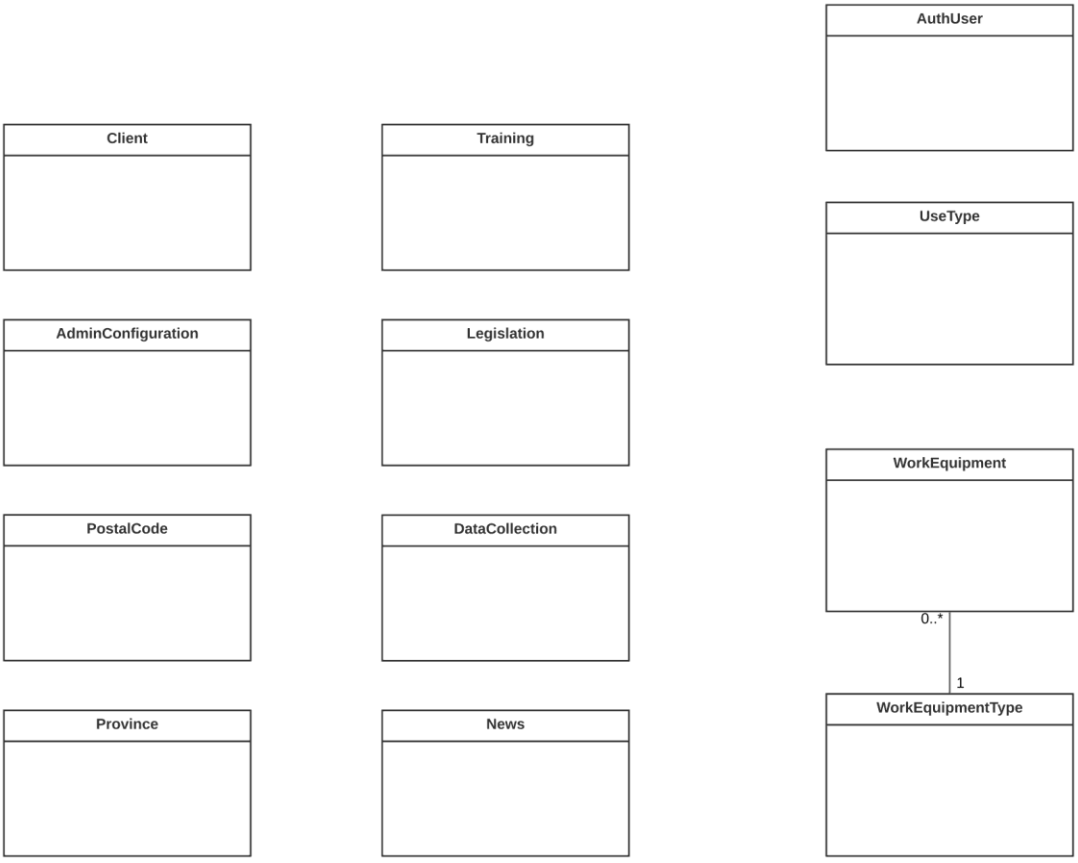

Ilustración 4.4 – Diagrama de clases 2

En estos dos diagramas se representan todos los modelos recogidos por el sistema, representados mediante cajas con su nombre correspondiente y las relaciones que mantienen entre ellos con sus multiplicidades, representadas en el diagrama mediante las líneas con los números de las multiplicidades en los finales de las líneas pegados a la caja del modelo. Las multiplicidades representan cuantas instancias de un objeto/modelo intervienen en la relación.

5. Diseño y arquitectura

5.1. Introducción

Antiguamente era un quebradero de cabeza programar un sistema complejo, ya que conforme va creciendo un proyecto es más difícil mantenerlo y ampliarlo debido a la progresiva complejidad del código [11]. Esta situación cambia si desde el inicio del proyecto se define una arquitectura software [12].

Una arquitectura software define los componentes en los que se divide un sistema, las interfaces mediante las qué se comunican y como utilizan estas interfaces para comunicarse. De la arquitectura de software que implantemos depende que se puedan cumplir los objetivos y requisitos tanto funcionales como no funcionales de un proyecto. Refiriéndonos a los requisitos funcionales como a las tareas que debe realizar el sistema, como por ejemplo que un usuario pueda registrarse en él y a los no funcionales a la calidad con la que se llevan a cabo los funcionales o características técnicas, como podría ser que la cuenta del usuario se registre en la base de datos en x milisegundos.

Actualmente, gracias a la comunidad de desarrolladores software que han creado diversas arquitecturas software, tenemos una amplia gama de estas para elegir una en la que basar nuestro proyecto. En nuestro proyecto concretamente hemos usado la Arquitectura Hexagonal.

5.2. Arquitectura Hexagonal

La Arquitectura Hexagonal, creada por Alistair Cockburn, propone la división de un sistema en capas con responsabilidades claramente diferenciadas, favoreciendo el desacoplamiento de éstas y facilitando su *testeo,* puesto que es más fácil localizar *bugs* pudiendo *testear* las capas por separado.

La división del sistema en capas, por otra parte, facilita la reutilización de fragmentos del sistema y la evolución de las partes de manera independiente, sin afectar al resto del sistema, lo cual dota al sistema de una alta adaptabilidad frente a posibles cambios por ejemplo a la hora de usar otros *frameworks* o tecnologías más actuales en el sistema evitando que éste se quedase inutilizado porque uno de sus componentes se quedase obsoleto.

Esta arquitectura también es conocida como arquitectura de puertos y adaptadores, el nombre se debe a la comunicación entre las diferentes capas. Un puerto es la interfaz que define cómo debe ser esa comunicación con la capa y el adaptador es la forma en la que se implementa esa comunicación en un contexto específico. Adicionalmente, este tipo de comunicación cumple con el último de los principios S.O.L.I.D. (con la letra D), “dependeny inversion principle” o principio de inversión de dependencias en español. Este principio afirma que se debe depender de abstracciones, no de implementaciones. En nuestro caso la capa depende de la abstracción de la otra capa mediante la interfaz.

El otro nombre que da a conocer la arquitectura hace referencia a la figura de un hexágono, pero podría ser representado con otra figura, ya que los lados de la figura simbolizan los puertos más exteriores del sistema, pero su existencia varía en base a la necesidad que tengamos de usar unos u otros puertos. Algún ejemplo muy común serían los controladores que permiten realizar peticiones HTTP al sistema o un Entity Manager que permite la persistencia de las entidades en una base de datos externa. [14] [15]

5.2.1. Capas

La Arquitectura Hexagonal se divide en tres capas: dominio, aplicación e infraestructura. Las tres capas mantienen dependencias entre sí de fuera hacia dentro, es decir, las capas interiores deben ser independientes de las exteriores. Así pues, la capa de dominio no debe usar objetos de la capa de aplicación ni de la capa de la infraestructura y la capa de aplicación no debe usar objetos de la capa de infraestructura.

La primera capa, dominio, es la encargada de definir las reglas de negocio y sus restricciones. En ella se albergan los modelos, representaciones de las entidades de la vida real involucradas en el sistema, así como los casos de uso en los que están involucrados, no obstante, no se implementan todavía, solo se declaran cuáles son y que datos requieren para implementarlos. Así, esta capa no conoce como se van a usar los objetos definidos en el resto del sistema.

En la siguiente capa, la de aplicación, es donde se implementan los casos de uso de la capa de dominio haciendo uso de los modelos anteriormente definidos. Estas implementaciones se desacoplan de las tecnologías que usemos para llevar a cabo ciertas acciones como mapeo de entidades a base de datos haciendo uso de interfaces como puerto abierto a diferentes adaptaciones.

Por último, tenemos la capa de infraestructura que tiene la tarea de implementar las adaptaciones de las interfaces definidas en las capas inferiores mediante las tecnologías que necesitemos, facilitando así posibles cambios de elección según convenga, de manera independiente a las capas inferiores. Algunas tecnologías que se usan en esta capa son *frameworks*, librerías externas o *SDKs*. [16]

5.3. Domain-Driven Design

Domain-driven design (DDD), en español: diseño guiado por el dominio, fue propuesta por Eric Evans. Esta filosofía o enfoque para el diseño de software propone una serie de prácticas para facilitar el desarrollo de aplicaciones con un modelo de dominio complejo, puesto que, como su propio nombre indica, propone dar especial importancia a la definición del dominio y que sirva como piedra angular del desarrollo del sistema. Así, el dominio del sistema se implementará basado en la comunicación de expertos de dominio y expertos técnicos desarrolladores de software mediante el *Ubiquitous Language* (Lenguaje ubicuo). [17]

Este concepto está claramente relacionado con la arquitectura que hemos decidido usar en este proyecto a través de su capa central, el dominio. Así pues, hacemos uso del enfoque DDD para afrontar la complejidad de modelado de nuestro proyecto, para obtener un dominio con un

código auto explicativo y bien estructurado haciendo uso de las prácticas promovidas por DDD que será detallado en el apartado de implementación.

6. Implementación

En este capítulo veremos cómo se ha implementado el código a través de las capas, ya introducidas en el anterior capítulo, de la arquitectura hexagonal y cómo se ha combinado con el enfoque DDD. Aparte de esto, también comentaremos las diferentes tecnologías utilizadas para la implementación de dicho código.

6.1. Dominio

La capa de dominio es la más interna y no debe saber nada de las capas exteriores. También es denominada capa de negocio porque en ella se define el modelado del sistema, los casos de uso que se llevan a cabo con ellos, y los eventos que generan.

6.1.1. Modelos

Para implementar los modelos nos hemos basado en el enfoque de modelado que usa DDD para la clasificación de los modelos. Los que hemos usado son:

- *Entity*: en español denominada entidad, es una representación de un objeto real que se identifica de forma inequívoca de las demás instancias. Un ejemplo de entidad podrían ser los diferentes empleados de una empresa.

- *Value Object*: a diferencia de las entidades, el *value object* no es único y si dos instancias de un mismo tipo de *value object* tienen las mismas características serán considerados el mismo. El ejemplo más típico son las fechas y las direcciones.

- *Aggregate*: llamado agregado en español, éste es una entidad promocionada, ya que agrupa a otras entidades y *value objects* y es el encargado de hacer que se cumplan las invariantes de todos ellos. Por ejemplo, un carrito de compra en una tienda online sería un agregado que podría contener entidades como los productos que se van añadiendo o

value objects como fecha de creación del carrito e invariantes como que el coste total del contenido siempre sea igual a la suma de todos los productos del carrito.

Sabiendo esto, clasificamos los modelos en estos tres tipos descritos y dentro de la carpeta "domain/model" creamos una carpeta por cada modelo, que contiene la clase java del modelo (con sus atributos, constructor y métodos) y su repositorio.

Dentro de las clases Java, usamos las anotaciones de Hibernate, tecnología que explicaremos con más detalle en el apartado de "Tecnologías utilizadas", para mapear los modelos y sus atributos con los de la base de datos.

Según DDD, un repositorio es una abstracción de las operaciones que se pueden realizar sobre un modelo en base de datos. Por esto ponemos la interfaz del repositorio de cada modelo en el dominio, de esta manera, la capa de negocio ignora cómo se implementen estas operaciones sobre base de datos y con qué tecnología se haga, consiguiendo desacoplamiento. La interfaz contendrá pues, los métodos que se podrán utilizar con respecto al modelo y los parámetros necesarios.

Por otra parte, los directorios de los agregados tienen peculiaridades, en ellos se encuentran su clase, su repositorio y las clases de las entidades que contiene el agregado. Estos últimos, siguiendo el enfoque DDD no tienen derecho a repositorio debido a que todas las operaciones que se hacen sobre ellos las debe llevar a cabo el agregado, ya que es el que los contiene y es el responsable de que se cumplan las invariantes atribuidas a estos.

6.1.2. Comandos

Los comandos en DDD se usan para recoger todos los datos necesarios para realizar un caso de uso. Nosotros hemos creado una carpeta "*command*" dentro de la carpeta "*domain*" en la que cada modelo tiene su carpeta con un comando por cada caso de uso que se lleve a cabo sobre el modelo correspondiente. Los comandos los hemos implementado con clases java que contienen

como atributos todos los parámetros que se necesitan para implementar el caso de uso, el constructor con los mismos parámetros que se asignan a los atributos y métodos *getters*, que son métodos que nos devuelven el valor de un atributo y hay uno por cada atributo de la clase.

6.1.3. Eventos

Los eventos, también usados en DDD, sirven para notificar al sistema de que algo ha ocurrido y así poder actuar en consecuencia mediante *listeners*, que son entidades que están pendientes de un evento concreto para llevar a cabo un método o fragmento de código. Por ejemplo, este patrón serviría para implementar una promoción de una tienda online en el día del cumpleaños del usuario, en el sistema hay un evento que se emite cada vez que realizas una compra y en el *listener* se comprueba si el día actual es el día del cumpleaños del usuario, así pues, se le aplicaría el descuento.

Los eventos los hemos implementado en la carpeta "*domain*/*events*", dentro de este directorio, tenemos una carpeta más por cada modelo del dominio y que cada una contiene una clase java por cada evento que puede generar el modelo correspondiente. Los eventos implementados corresponden a las acciones de crear un modelo y actualizar un modelo.

6.2. Aplicación

En esta capa es donde se ubican los *handlers*, estos son clases java en los que implementamos el código que ejecutan los casos de uso y se ubican en la carpeta "*application*/*command*", distribuidos en carpetas correspondientes a los diferentes modelos y dentro de ellas un *handler* por cada caso de uso del modelo en cuestión.

Las capas de la arquitectura hexagonal, como dijimos en el capítulo anterior, interactúan mediante dependencias de fuera hacia dentro. Así pues, la capa de aplicación usa diferentes clases de la capa del dominio.

```
private CompanyRepository repository;

 DINACODE
public CreateCompanyHandler(CompanyRepository repository) { this.repository = repository; }

 DINACODE
@Override
public void handle(CreateCompany command) {
    var company = new Company(command);

    repository.save(company);
}
```

Ilustración 6.1 – Fragmento de la clase "CreateCompanyHandler"

Para mostrarlo usaremos el ejemplo de la figura, el *handler* "CreateCompanyHandler", correspondiente al caso de uso de crear una compañía. El método "handle", recibe como parámetro un objeto ya definido en la capa de dominio y este es el comando correspondiente al mismo caso de uso del *handler* en cuestión, así ya podríamos usar los atributos necesarios para el caso de uso definidos en el comando, pero en este caso encajan con los parámetros del constructor de la compañía y le podemos pasar el objeto entero. Esta es la primera dependencia con la capa de dominio. En el ejemplo que nos incumbe, que es el de crear una compañía, necesitaremos usar la entidad "*Company*", definida también en el dominio, aquí tenemos otra dependencia más con la capa de dominio. El constructor de dicha entidad recibe como parámetro el comando "CreateCompany", por lo que nos bastará con pasarle el comando recibido como parámetro del método "handle". En último lugar, una vez hemos creado una instancia de la entidad y la hemos guardado en una variable, tenemos que guardarla en base de datos.

Para este último paso haremos uso del repositorio de la entidad que nos atañe, que se encuentra en la capa de dominio (otra dependencia con la primera capa). El problema que tenemos es que este repositorio es una interfaz que solo contiene los métodos definidos que podemos realizar sobre la entidad para interaccionar con base de datos. Este problema lo resolvemos gracias a la inyección de dependencias.

6.2.1. Inyección de dependencias

La inyección de dependencias es un patrón de diseño que pretende encargarse de la creación de instancias de clases que otras clases necesitan y suministrárselas. A estas instancias que necesitan otras clases las llamamos dependencias.

En nuestro caso queremos utilizar una instancia de la clase "CompanyRepository", es decir, el repositorio el repositorio correspondiente al modelo del caso de uso que estamos llevando a cabo. Para hacerse cargo de esto, el framework de Java que usamos, que es Spring, hace uso del concepto IoC (*Inversión of Control*), o Inversión de Control en español, para mantener en contexto a todas las instancias de la aplicación e inyectarlas a quien las necesite.

Las instancias que maneja Spring se llaman beans y este las almacena en un contenedor propio que le sirve para realizar su trabajo de IoC buscando las dependencias entre los objetos del proyecto e inyectárselas a partir de los beans del contenedor, como podemos ver en la siguiente imagen. [18]

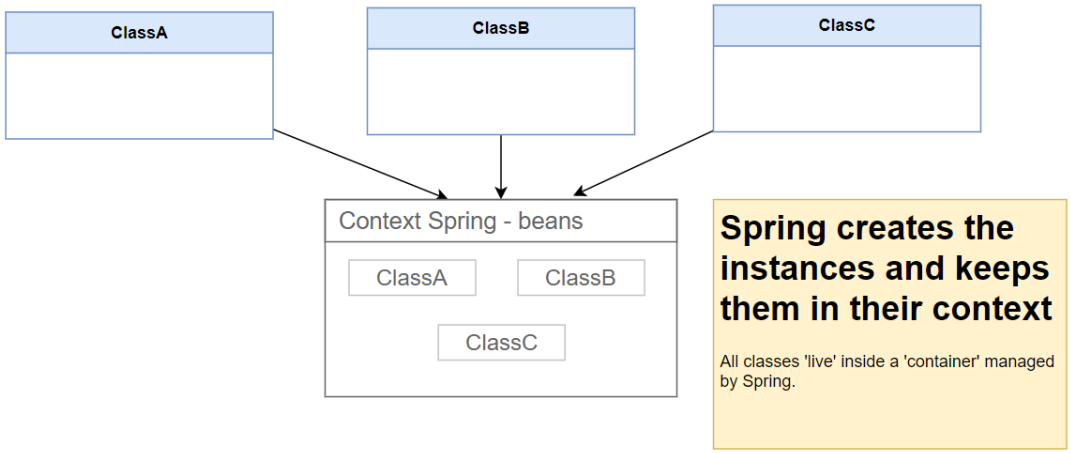

Ilustración 6.2.1 – Contexto de Spring

La forma más común y recomendada es inyectar la interfaz de la cual necesitamos la instancia, es decir de la que dependemos en el constructor [19]. El primer paso sería crear una variable para guardar la instancia del repositorio en la clase y después en el constructor solo tendríamos que poner como parámetro la interfaz del repositorio que necesitamos y asignarla a la variable

de la clase en el cuerpo del constructor. Esto es lo que hacemos en la clase "CreateCompanyHandler", como podemos ver en la siguiente ilustración.

```
private CompanyRepository repository;

 DINACODE
public CreateCompanyHandler(CompanyRepository repository) { this.repository = repository; }
```

Ilustración 6.2 – Inyección de dependencia en la clase "CreateCompanyHandler"

De esta manera ya tendríamos inyectada la dependencia en nuestra clase "CreateCompanyHandler" y aunque sea una interfaz con los métodos simplemente definidos, Spring se encarga de usar los métodos de la implementación de la interfaz, consiguiendo así desacoplamiento entre los *handlers* y la implementación de los repositorios que podríamos cambiar en cualquier momento solo modificando la clase de la implementación, así pues podríamos usar una nueva tecnología de comunicación con base de datos haciendo cambios en una sola clase.

6.3. Infraestructura

En la última capa es donde hemos ubicado los controladores y las implementaciones de los repositorios. Ya que los primeros actúan de puerto de la aplicación y los segundos son implementaciones con una tecnología concreta independiente del dominio y de la aplicación.

6.3.1. Controladores

Los controladores hacen de puerto de la aplicación y permite la comunicación con el *frontend*. Esta comunicación está basada en el protocolo HTTP, ya que este protocolo nos permite establecer una arquitectura de cliente-servidor entre el *frontend* (cliente) y el *backend* (servidor) [20].

Estos dos componentes del protocolo se pueden comunicar mediante diferentes métodos, pero nosotros usamos dos en concreto: *GET* y *POST* [21]. El método GET sirve para realizar una petición de datos, por ejemplo, en nuestros casos de uso serían los de visualizar las tablas de los modelos de datos o los de editar una entidad existente, ya que antes de modificar la entidad tenemos que pedir al *backend* la entidad actual con la que se rellena el formulario a modificar. Por otra parte, el método POST sirve para mandar una petición de cambio de datos en el servidor desde el cliente, teniéndole que mandar estos datos. En nuestro caso, el POST lo utilizamos en los casos de uso de crear, modificar o eliminar; mandándole la entidad sobre la que queremos ejecutar el caso de uso.

Cada modelo del dominio tiene su controlador dentro de la carpeta "*infraestructura/controller*". En estas clases usamos las facilidades que nos ofrece Spring para la creación de controladores, gracias a estas podemos asignar rutas y métodos HTTP a los métodos que definamos dentro de nuestra clase controlador. Así pues, las peticiones HTTP que se hacen desde el *frontend* usan las rutas que definamos en los controladores del *backend* para conseguir ejecutar la implementación de dichos métodos.

```
@RestController
@RequestMapping("/api/v1/company")
public class CompanyController {

    3 usages
    private final CommandBus bus;
    3 usages
    private final CompanyViewRepository repository;

    DINACODE
    public CompanyController(CommandBus bus, CompanyViewRepository repository) {
        this.bus = bus;
        this.repository = repository;
    }

    DINACODE
    @GetMapping(value = "get")
    public CompanyView companyGet(@RequestParam("id") String id) { return repository.viewById(id); }

    DINACODE
    @GetMapping(value = "all")
    public List<CompanyView> companyAll() { return repository.allViews(); }

    DINACODE
    @PostMapping(value = "create")
    public void create(@RequestBody CreateCompany command) { bus.handle(command); }

    DINACODE
    @PostMapping(value = "update")
    public void update(@RequestBody UpdateCompany command) { bus.handle(command); }

}
```

Ilustración 6.3 – Fragmento de la clase "CompanyController"

En esta ilustración tenemos un ejemplo de controlador, en concreto el de la entidad “Company”. En el podemos ver cómo hacemos uso de las herramientas que nos brinda Spring para la creación de un controlador. En primer lugar, indicamos a Spring que esta clase es un controlador mediante la anotación “@RestController” de la línea 12 y le indicamos la ruta base del controlador en la línea siguiente con la anotación “@RequestMapping”. Para asociar los métodos HTTP a los métodos de la clase, así como la ruta correspondiente a la llamada HTTP, usamos las anotaciones “@GetMapping” y “@PostMapping” (líneas 24, 29, 34 y 39). En último lugar, podemos extraer el cuerpo de la llamada POST mediante la anotación “@PostMapping” y los parámetros de las llamadas GET con “@RequestParam”.

Los métodos de controlador deben comunicarse con la capa de aplicación, siguiendo el modelo de dependencias de fuera hacia adentro, para poder ejecutar los casos de uso y esto se hace mediante el *command bus*. El *command bus* o bus de comandos en español es el que se encarga en DDD de la ejecución de los comandos. En los métodos POST del controlador son en los qué se usa el *command bus*. Para ello, en el método del controlador, tenemos que encapsular la información que nos llega por la petición HTTP en un comando del dominio y pasárselo como parámetro al método *handle* del bus inyectado en el controlador. Después de esto, el *command bus* se encarga de ejecutar el handler correspondiente al comando que le hemos pasado.

Por otro lado, los métodos del controlador que usan un método GET de petición, simplemente usan los métodos del repositorio del modelo para acceder a base de datos, buscar los datos correspondientes a la petición y devolverlos en forma de objeto JSON (JavaScript Object Notation), el formato común en comunicaciones HTTP.

6.3.2. Repositorios

En esta capa es donde están implementados los repositorios que habíamos definido en la capa de aplicación. Éstas se hayan aquí debido a que el método y la forma en que se hagan las operaciones con base de datos se debe ignorar en las capas inferiores, en las que solo está definido qué se debe poder hacer y esto está en las interfaces de los repositorios del dominio.

En esta aplicación vamos a comunicarnos me con una base de datos MySQL mediante la tecnología Hibernate, ambos serán más detallados en el apartado siguiente, "Tecnologías utilizadas", pero básicamente lo que nos permite Hibernate en la implementación de los repositorios es hacer consultas SQL en nuestra base de datos, es decir, seleccionar la información deseada de base de datos (dependiendo del método del repositorio que estemos implementando usaremos unos parámetros u otros) y devolverla.

6.4. Tecnologías utilizadas

6.4.1. Spring

Spring [22] es el *framework* del lenguaje Java, cuya primera versión fue lanzada en junio de 2003 por Rod Johnson. Desde nuestra empresa lo elegimos para el proyecto debido a que éste es un proyecto empresarial dentro del ámbito *corporate*, es decir, no es un proyecto de tipo *start up* y debido a esto necesitamos estabilidad y robustez en el proyecto. Estas características nos las proporciona un lenguaje fuertemente tipado como es Java, uno de los lenguajes más extendidos en el entorno de software corporativo. Al elegir Java también nos quedamos con su *framework,* Spring, éste está preparado para escalar, cosa que casa perfectamente con los objetivos del proyecto. Además de esto, Spring admite muy bien la arquitectura hexagonal debido principalmente a su sistema de inyección de dependencias, principio indispensable en dicha arquitectura por la naturaleza de puertos y adaptadores que promueve.

6.4.2. Hibernate

Esta tecnología es una herramienta que nos permite mapear los atributos de la base de datos con el modelado de aplicaciones Java, lo que es llamado una herramienta ORM (mapeo objeto-realcional) y esto es implementado en archivos XML o mediante anotaciones en las clases Java de los modelos. [23]

Como hemos visto ya en la implementación, en nuestro proyecto lo hemos usado mediante anotaciones en nuestros modelos del dominio para el mapeo con base de datos y para la realización de consultas a esta misma, en la implementación de los repositorios en la capa de infraestructura.

6.4.3. MySQL

MySQL [24] es un gestor de bases de datos muy extendido en la actualidad, la mayoría de las aplicaciones web confían en esta tecnología y, en parte, su popularidad se debe a que es *open source*, es decir, de código libre. Gracias a él, podemos definir un complejo sistema relacional como el que interesa en nuestro proyecto.

6.4.4. BitBucket

BitBucket [25] es el sistema de alojamiento de proyectos que utilizamos para crear el repositorio del proyecto y poder usar Git [26], un sistema de control de versiones muy extendido en la actualidad que nos facilita trabajar en paralelo en el desarrollo y poder acceder a versiones anteriores del código en caso de necesitarlo.

6.4.5. Docker

Docker [27] es una plataforma de software para el despliegue de aplicaciones, se basa en encapsular aplicaciones o partes de una aplicación en "contenedores". Estos contenedores están preparados con las herramientas necesarias para ejecutar la aplicación que deseemos, normalmente tienen un sistema operativo Linux con solo los elementos necesarios para poder correr la aplicación en cuestión rebajando así bastante su tamaño y haciéndolo mucho más liviano de lo que es un sistema operativo de uso corriente para un usuario cualquiera.

Nosotros actualmente solo usamos un contenedor para que contenga la base de datos del proyecto, ya que nuestro *backend* es un proyecto Spring y podemos compilarlo y ejecutarlo desde el mismo IDE. Este contenedor lo creamos a partir de una imagen sacada de Docker Hub [28] una plataforma en la que la gente puede subir imágenes ya creadas para crear contenedores de diferentes tipos, ya sean MySQL, aplicaciones Angular, Spring…

En un futuro, cuando el proyecto pase a producción, es decir, se despliegue la aplicación en un servidor para que cualquier persona pueda ejecutarlo desde su navegador accediendo a la web, por la parte del *backend*, tendremos el contenedor de MySQL y el del proyecto Spring.

7. Pruebas

En cuanto a las pruebas, en nuestra empresa utilizamos para todos los proyectos la tecnología SonarQube [29], que nos permite comprobar la calidad del código del proyecto. Se suelen hacer revisiones con esta herramienta cada cierto tiempo, pero no con mucha frecuencia debido a que las metodologías y arquitecturas de programación de la empresa suelen obtener buenos valores en los resultados. En este caso es la primera vez que se usa la herramienta.

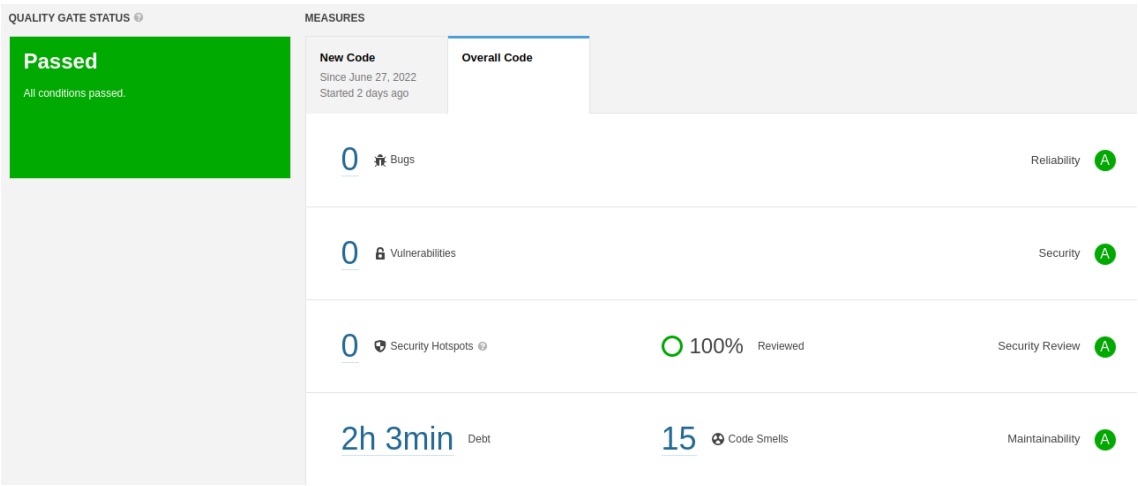

Ilustración 7.1 – Parámetros obtenidos con SonarQube

Estos son los principales parámetros que nos ofrece SonarQube y en los que nos fijamos en la empresa para darle el visto bueno a la calidad del código que escribimos. Buscando una puntuación de "A" en todos ellos.

El primer parámetro que mide esta tecnología es la fiabilidad del código a través de los errores encontrados, es decir, puntos de fallo potenciales o reales en el código, en nuestro caso no tenemos errores por lo que tenemos la mayor puntuación, que es una "A".

A continuación, tenemos el parámetro de seguridad, que la tecnología mide a través de las vulnerabilidades encontradas en el código, éstas son agujeros de seguridad en el código que

podrían ser usados para atacar a nuestro software. Como en el anterior parámetro tenemos una puntuación “A” debido a que no tenemos ninguna vulnerabilidad.

Relacionado con el anterior, está el parámetro de revisión de seguridad, que se mide en puntos de acceso de seguridad, este parámetro de seguridad se diferencia del anterior en que está sujeto a una revisión manual, ya que está sujeto a la interpretación del programador. Al igual que en los anteriores, tenemos la puntuación más alta en este punto.

Por último, SonarQube nos proporciona la mantenibilidad del código a través de la deuda técnica y los *code smells*, estos últimos indican malas prácticas de programación que pueden derivar en dificultades futuras en el proyecto. Este apartado no esta perfecto, ya que acumulamos un poco de deuda técnica y algunos *code smells* pero no es nada preocupante, ya que mantenemos una puntuación de “A”.

Así pues, pasamos los controles de calidad del código estipulados por la empresa con la máxima puntuación en todos los parámetros medidos, pero también se podría mejorar un poco más para aumentar la mantenibilidad futura del código.

8. Conclusiones

En cuanto a los objetivos iniciales del proyecto, éstos han sido cumplimentados con éxito:

- El usuario puede dar de alta y gestionar todas las entidades relacionadas con las puertas para poder solicitar su marcado CE.
- El administrador puede gestionar las entidades necesarias predefinidas de cara al usuario.
- El administrador puede gestionar las cuentas de las empresas y sus clientes.

Durante el transcurso del desarrollo del proyecto he podido afianzar conocimientos adquiridos en la carrera y adquirir otros nuevos. Respecto a los conocimientos afianzados, he podido aplicar conocimientos de asignaturas como DDS, ya que en la empresa en la que se ha desarrollado el proyecto, se promueven las buenas prácticas de programación y sobre todo se usan los principios SOLID, que ya aprendimos en dicha asignatura y he podido practicar y afianzar en el proyecto. Otra asignatura de la que he podido aprovechar conocimientos es BDA, en la que aprendimos a hacer consultas SQL y los fundamentos de las bases de datos. Estos conocimientos los he podido aplicar con la tecnología Hibernate, ya descrita en la memoria. Aparte de estas, en la asignatura TSR también aprendimos las bases de Docker para el despliegue de aplicaciones y con este proyecto he podido aplicarlo para un proyecto real.

Por otra parte, también he podido aprender cosas nuevas en el desarrollo de este proyecto y las principales son: Arquitectura hexagonal, DDD, Spring e Hibernate.

La arquitectura hexagonal y el enfoque DDD al principio fueron complejos de entender, ya que se alejan de la típica arquitectura MVC (Modelo Vista Controlador) que habíamos aplicado en los proyectos realizados durante el grado. Pero una vez te acostumbras a usarla te das cuenta de su gran utilidad. Spring e Hibernate al fin y al cabo son *frameworks* con mucha documentación y he podido informarme por mi cuenta y con la ayuda constante de mi tutor y mis compañeros para conseguir entenderlos y acostumbrarme a su uso.

Sin duda, el gran reto de este proyecto ha sido enfrentarme a estas nuevas arquitecturas y tecnologías que he tenido que ir aprendiendo poco a poco y con paciencia estudiando por mi cuenta, con la ayuda del tutor de prácticas, mis compañeros y cursos proporcionados por la empresa. Pero una vez finalizado el proyecto siento que entiendo mucho mejor cómo funcionan todas ellas y ahora puedo aplicarlas mucho más fácilmente. Personalmente, creo que ha mejorado mi adaptabilidad a nuevos retos y a afrontarlos más sabiamente, dividiendo los problemas e ir superando pequeñas metas.

8.1. Objetivos futuros

Como está escrito en los objetivos de la memoria, ahora que se da soporte a la documentación de puertas, el próximo objetivo sería dar soporte a otros tipos de productos para la obtención de su marcado CE con el manejo de todos los modelos que eso conlleva y para lo que estamos preparados ya que hemos desarrollado un software escalable, preparado para esto precisamente.

Bibliografía

[1] Davi Vieira, "*Designing Hexagonal Architecture with Java*", Pack Publishing, 2022

[2] Vladik Khononov, "Learning Domain-Driven Design: Aligning Software Architecture and Business Strategy", O'Reilly UK Ltd., 2021

[3] Documentación de Angular disponible en https://angular.io/docs [consulta: 15 mayo 2022]

[4] Documentación de Spring disponible en https://docs.spring.io/spring-framework/docs/current/reference/html/ [consulta: 15 mayo 2022]

[5] Documentación de Hibernate disponible en https://hibernate.org/orm/documentation/6.1/ [consulta: 15 mayo 2022]

[6] Documentación de MySQL disponible en https://dev.mysql.com/doc/ [consulta: 15 mayo 2022]

[7] Información acerca del marcado CE disponible en https://es.wikipedia.org/wiki/Marcado_CE [consulta: 22 mayo 2022]

[8] Robert C. Martin, "*Clean Architecture: A Craftsman's Guide to Software Structure and Design: A Craftsman's Guide to Software Structure and Design*", Addison-Wesley, 2017

[9] IEE Std 830-1998 disponible en https://wikis.fdi.ucm.es/ELP/Especificaci%C3%B3n_de_Requisitos_Software_seg%C3%BAn_el_est%C3%A1ndar_IEEE_830 [consulta: 26 mayo 2022]

[10] Página oficial de UML disponible en https://www.uml.org/ [consulta: 28 mayo 2022]

[11] Arquitectura de software disponible en https://es.wikipedia.org/wiki/Arquitectura_de_software [consulta: 29 mayo 2022]

[12] Arquitectura de software en el proceso de desarrollo disponible en https://www.voigtmann.de/es/desarrollo-de-software/arquitectura-de-software/ [consulta: 29 mayo 2022]

[13] Importancia de la arquitectura de software en las organizaciones disponible en https://www.icesi.edu.co/unicesi/todas-las-noticias/1949-importancia-de-la-arquitectura-de-software-en-las-organizaciones [consulta: 29 mayo 2022]

[14] Arquitectura hexagonal disponible en https://medium.com/@edusalguero/arquitectura-hexagonal-59834bb44b7f [consulta: 29 mayo 2022]

[15] Del código espagueti a la arquitectura hexagonal disponible en https://www.hiberus.com/crecemos-contigo/del-codigo-espagueti-a-la-arquitectura-hexagonal/ [consulta: 29 mayo 2022]

[16] Arquitectura hexagonal: introducción y estructura disponible en https://wata.es/arquitectura-hexagonal-introduccion-y-estructura/?lang=es [consulta: 29 mayo 2022]

[17] Diseño guiado por el dominio disponible en https://es.wikipedia.org/wiki/Dise%C3%B1o_guiado_por_el_dominio [consulta: 29 mayo 2022]

[18] Qué es la inyección de dependencias en Spring disponible en https://gustavopeiretti.com/spring-inyeccion-dependencias/#:~:text=La%20inyecci%C3%B3n%20de%20dependencias%20es,esta%20clase%20los%20pueda%20utilizar [consulta: 19 junio 2022]

[19] Inyección de dependencias usando Spring Framework disponible en https://www.ramoncarrasco.es/es/content/es/kb/124/inyeccion-de-dependencias-usando-spring-framework [consulta: 19 junio 2022]

[20] Generalidades del protocolo HTTP disponible en https://developer.mozilla.org/es/docs/Web/HTTP/Overview [consulta: 20 junio 2022]

[21] Métodos de petición HTTP disponible en https://developer.mozilla.org/es/docs/Web/HTTP/Methods [consulta: 20 junio 2022]

[22] Eugenia Pérez Martínez, "*Spring 5*", 2018

[23] Christian Bauer, "*Java Persistence with Hibernate*", Manning Publications, 2015

[24] Sr. Edgar D'Andrea, "MySQL 8: Lenguaje y diseño de bases de datos", Independently published, 2021

[25] Página oficial de BitBucket disponible en https://bitbucket.org/ [consulta: 21 junio 2022]

[26] Documentación de Git disponible en https://git-scm.com/doc [consulta: 21 junio 2022]

[27] Nigel Poulton, "*Docker Deep Dive*", Independently published, 2017

[28] Página oficial de Docker Hub disponible en https://hub.docker.com/ [consulta: 21 junio 2022]

[29] Documentación de SonarQube disponible en https://docs.sonarqube.org/latest/ [consulta: 21 junio 2022]

ANEXO

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. **Fin de la pobreza.**				X
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**		X		
ODS 4. **Educación de calidad.**				X
ODS 5. **Igualdad de género.**				X
ODS 6. **Agua limpia y saneamiento.**		X		
ODS 7. **Energía asequible y no contaminante.**				X
ODS 8. **Trabajo decente y crecimiento económico.**	X			
ODS 9. **Industria, innovación e infraestructuras.**				X
ODS 10. **Reducción de las desigualdades**.				X
ODS 11. **Ciudades y comunidades sostenibles**.	X			
ODS 12. **Producción y consumo responsables.**	X			
ODS 13. **Acción por el clima.**	X			
ODS 14. **Vida submarina.**		X		
ODS 15. **Vida de ecosistemas terrestres.**	X			
ODS 16. **Paz, justicia e instituciones sólidas.**				X
ODS 17. **Alianzas para lograr objetivos.**	X			

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

Primeramente, describiremos las relaciones más fuertes y directas del proyecto con los objetivos de desarrollo sostenible.

- **Trabajo decente y crecimiento económico,** este proyecto encargado por una empresa cliente de la nuestra, para empezar, está siendo desarrollado con constante comunicación entre ambas empresas generando una relación profesional plenamente satisfactoria, por lo que es un trabajo decente. Por otra parte, el proyecto está destinado a ahorrar costes a las empresas fabricantes de productos para facilitarles su comercio, así pues, favorece también a su crecimiento económico a la par que a la empresa proveedora y a nuestra empresa que les proporciona el software.

- **Ciudades y comunidades sostenibles,** el proyecto en desarrollo favorece a este punto debido a que contribuimos a la disminución de diversos recursos como papel más directamente y otros recursos como tinta e impresoras que son necesarias para llevar a cabo el papeleo que conlleva la solicitud de un marcado CE.

- **Producción y consumo responsables,** este punto está directamente ligado con el anterior como medio para conseguirlo, ya que disminuyendo el consumo de los recursos descritos en el anterior punto es como conseguimos contribuir a las ciudades y comunidades sostenibles.

- **Acción por el clima,** el proyecto está claramente vinculado a una buena acción por el clima gracias a que al no tener que imprimir papeles para las solicitudes de marcados CE, se evita la cadena de producción de impresoras y tinta, lo que conlleva producción de plásticos, a su vez generando emisiones de gas de efecto invernadero a la atmósfera, afectando negativamente al clima.

- **Vida de ecosistemas terrestres,** la propia contribución del proyecto al punto anterior implica una buena acción por los ecosistemas terrestres, pero adicionalmente, al reducir el consumo de papel, disminuimos simultáneamente la tala de árboles, recurso fundamental de la vida en el planeta Tierra.

- **Alianzas para lograr objetivos,** en esta misma memoria del proyecto hemos descrito una serie de objetivos a cumplir, que por una parte estamos llevando a cabo los desarrolladores del software y por otra la alianza de la empresa cliente que nos contrata para desarrollarlo y la nuestra. Como he dicho anteriormente las dos empresas están en constante comunicación supervisando el proceso para

la cumplimentación de los objetivos definidos. Lo que también es favorecido por el espíritu DDD.

Aparte de estas relaciones con los ODS nuestro proyecto también tiene relación con otros tres puntos, de forma indirecta, pero igualmente significativa.

- **Agua limpia y saneamiento,** la producción de plásticos por el uso de impresoras es un factor a tener en cuenta en la limpieza del agua de nuestros océanos, ya que como es sabido, se acumulan grandes cantidades de residuos plásticos en ellos, llegando a formar masas enormes y contaminando seriamente la salud de la vida marina. Debido a esto, si reducimos el uso de impresoras en el trámite de solicitud de marcados CE indirectamente, aportamos nuestro grano de arena en el saneamiento de los océanos.

- **Vida submarina,** si contribuimos a la limpieza del agua con el anterior punto, a su vez estamos apoyando la salud del ecosistema acuático, que se ve gravemente afectado por las mareas de plásticos.

- **Salud y bienestar,** como último punto y a modo de conclusión, por la alineación en mayor o menor medida de nuestro proyecto con esta sección de puntos de los ODS, se contribuye también a la salud y bienestar de seres vivos pertenecientes tanto al ecosistema acuático como al terrestre.