Universitat Oberta de Catalunya uoc.edu

DSMantenimiento. Gestión de incidencias

Miren Itziar Anso Otaegi
Grado de Ingeniería Informática
Java EE

Josep Maria Camps Riba
Santi Caballe Llobet

Junio 2021

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada 3.0 España de Creative Commons

FICHA DEL TRABAJO FINAL

Título del trabajo:	*DSMantenimiento – Gestión de incidencias*
Nombre del autor:	*Miren Itziar Anso Otaegi*
Nombre del consultor/a:	*Josep Maria Camps Riba*
Nombre del PRA:	*Santi Caballe Llobet*
Fecha de entrega:	06/2021
Titulación:	*Grado de Ingeniería Informática*
Área del Trabajo Final:	*Java EE*
Idioma del trabajo:	*Castellano*
Palabras clave	*Spring Boot, React, API REST*

Resumen del Trabajo:

En este trabajo final de grado se desarrolla una aplicación web para la gestión de incidencias en empresas de mantenimiento industrial. El objetivo principal del proyecto es la aplicación de los conocimientos adquiridos a lo largo del grado en distintas asignaturas, especialmente las cursadas en el itinerario de desarrollo de software.

El proyecto viene motivado por la necesidad de las pequeñas empresas de adaptarse a los cambios tecnológicos actuales, a pesar de que, en muchas ocasiones, no cuentan con grandes recursos para ello. En concreto, se plantea la necesidad de desarrollar un software a medida que ofrezca un servicio de gestión de incidencias a una empresa dedicada a ofrecer un servicio técnico de maquinaria de soldadura

El proyecto se aborda desde una perspectiva “Full stack”, es decir, desarrollando tanto la parte de lógica de negocio como la parte de presentación. Para el desarrollo de la parte backend se utiliza Spring Boot que se encarga del acceso a la base de datos PostgreSQL y de proporcionar datos JSON al frontend, desarrollado utilizando React JS.

La metodología de desarrollo empleada para este trabajo es el desarrollo en cascada que comprende las siguientes fases: Definición del proyecto, Requerimientos, Análisis y Diseño, y Desarrollo.

El resultado del trabajo es una aplicación funcional que cumple con las necesidades detectadas y que ha permitido abordar un proyecto de desarrollo de software en todas sus etapas.

Abstract:

In this final degree Project, a web application is developed for incident management in industrial maintenance companies. The main objective of the Project is to apply the knowledge acquired throughout the degree in different subjects, especially those in the software development itinerary.

The project is motivated by the need of small companies to adapt to current technological advances, despite the fact that, on many occasions, they do not have required resources to do so. Specifically, the need arises to develop custom software that offers an incident management service to a company dedicated to offering a technical service for welding machinery.

The project is approached from a “Full stack” perspective, that is, developing both the business logic part and the presentation part. For the development of the backend part, Spring Boot is used, which is responsible for accessing the PostgreSQL database and providing JSON data to the frontend, developed using React JS.

The development methodology used for this work is cascade development that comprises the following phases: Project Definition, Requirements, Analysis and Design, and Development.

The result of the work is a functional application that meets the needs detected and that has made it possible to tackle a software development project in all its stages.

Índice

1. Introducción ... 1
1.1 Contexto y justificación del Trabajo ... 1
1.2 Objetivos del Trabajo ... 1
1.3 Enfoque y método seguido ... 2
1.4 Planificación del Trabajo ... 5
1.4.1. Fase 1: Definición del proyecto ... 6
1.4.2. Fase 2: Requerimientos, análisis y diseño ... 6
1.4.3. Fase 3: Desarrollo ... 6
1.4.4. Fase 4: Memoria del proyecto ... 7
1.5 Breve sumario de productos obtenidos ... 7
1.6 Breve descripción de los otros capítulos de la memoria ... 7
1.6.1 Análisis funcional de la aplicación ... 7
1.6.2. Diseño técnico ... 7
2. Análisis funcional ... 8
2.1. Actores ... 8
2.2. Casos de uso ... 8
3. Diseño técnico ... 17
3.1. Diagrama de clases ... 17
3.2. Diseño del modelo de datos ... 18
3.2.1. Herencia entre User y Technician, Customer y Administrator ... 19
3.2.2. Relación n a m entre incidencias y máquinas ... 19
3.2.3. Almacenamiento de ficheros ... 19
3.3. Diseño de arquitectura ... 20
3.3.1. Punto de vista de la computación ... 20
3.3.1.1. Capa de presentación ... 21
3.3.1.2. Capa de Negocio ... 22
3.3.1.3. Capa de Integración ... 23
3.3.2. Refinamiento del punto de vista de la computación ... 24
3.3.2.1. Capa de presentación ... 24
3.3.2.2. Capa de negocio ... 27
3.3.2.3. Capa de integración ... 30
3.3.3. Selección de la tecnología ... 33
3.3.3.1. Capa de presentación ... 33
3.3.3.2. Capa de negocio ... 34
3.3.3.3. Capa de integración ... 36
3.3.4. Frontend ... 39
3.3.5. Securización de la aplicación ... 39
4. Desarrollo de la aplicación ... 41
4.1. Preparación del entorno de desarrollo ... 41
4.1.1. Preparación de la base de datos ... 41
4.1.2. Preparación del backend ... 41
4.1.3. Preparación del frontend ... 43
4.2. Construcción de la aplicación ... 44
4.2.1. Construcción de API REST ... 44
4.2.2. Autentificación en la API REST con JWT ... 46
4.2.3. Construcción del frontend ... 47
4.2.4. Herramienta de control de versiones ... 48

4.3. Producto final 49
4.3.3. Perfil Administrador 49
4.3.4. Perfil Técnico 50
4.3.5. Perfil Cliente 53
5. Pruebas 54
5.1. Pruebas de API REST con postman 54
5.2. Pruebas frontend 55
5.2.1. Componente Catálogo 55
5.2.2. Componente Incidencias 55
5.2.3. Componente Cliente 55
5.2.4. Componente Usuario 56
5.2.5. Componente Localización 56
6. Despliegue 57
7. Conclusiones 58
8. Glosario 60
9. Bibliografía 62
10. Anexos 64
10.1. Manual de instalación 64
10.1.1. Base de datos 64
10.1.2. Backend 65
10.1.3. Frontend 67

Lista de figuras

Ilustración 1: Frameworks más queridos por los programadores 5
Ilustración 2: Diagrama de Gantt 5
Ilustración 3: Planificación Fase 1 6
Ilustración 4: Planificación Fase 2 6
Ilustración 5: Planificación Fase 3 6
Ilustración 6: Planificación Fase 4 7
Ilustración 7: Diagrama de casos de uso 9
Ilustración 8: Diagrama de clases 17
Ilustración 9: Diagrama del modelo de datos 18
Ilustración 10: Punto de vista de computación 20
Ilustración 11: Punto de vista de computación. Presentación. 21
Ilustración 12: Punto de vista de computación. Negocio 22
Ilustración 13: Punto de vista de computación. Integración. 23
Ilustración 14: Capa de presentación, componente Catálogo 24
Ilustración 15: Capa de presentación, componente Cliente 25
Ilustración 16: Capa de presentación, componente Localización 25
Ilustración 17: Capa de presentación, componente Usuario 26
Ilustración 18: Capa de presentación, componente Incidencia 26
Ilustración 19: Capa de Presentación, Front Controller 27
Ilustración 20: Capa de negocio, componente Catálogo 27
Ilustración 21: Capa de negocio, componente Cliente 28
Ilustración 22: Capa de negocio, componente Incidencias 28
Ilustración 23: Capa de negocio, componente Usuario 29
Ilustración 24: Capa de negocio, componente Localización 29
Ilustración 25: Capa de integración, componente Catálogo 30
Ilustración 26: Capa de integración, componente Cliente 30
Ilustración 27: Capa de integración, componente Localización 31
Ilustración 28: Capa de integración, componente Usuario 31
Ilustración 29: Capa de integración, componente Incidencia 32
Ilustración 30: Capa de presentación, controladores 33
Ilustración 31: Refinamiento capa de negocio, componente Localización 34
Ilustración 32: Refinamiento capa de negocio, componente Catálogo 34
Ilustración 33: Refinamiento capa de negocio, componente Incidencia 35
Ilustración 34: Refinamiento capa de negocio, componente Cliente 35
Ilustración 35: Refinamiento capa de negocio, componente Usuario 36
Ilustración 36: Refinamiento capa de integración, componente Incidencia 36
Ilustración 37: Refinamiento componente integración, componente Usuario 37
Ilustración 38: Refinamiento componente integración, componente Cliente 37
Ilustración 39: Refinamiento componente integración, componente Catálogo 38
Ilustración 40: Refinamiento componente integración, componente Localización 38
Ilustración 41: Componentes principales de una aplicación React 39
Ilustración 42: diagrama de secuencia securización JWT 40
Ilustración 43: Ejecución de la base de datos Postgres mediante Docker 41
Ilustración 44: Generación de un proyecto mediante Spring Initializr 42
Ilustración 45: Eclipse IDE con el proyecto importado 43
Ilustración 46: proyecto frontend en el editor Atom 44

Ilustración 47: Documentación de la API REST 45
Ilustración 48: Información proporcionada sobre la petición para crear fabricante 45
Ilustración 49: Prueba de una petición para obtener un JWT 46
Ilustración 50: Ejemplo de petición no autorizada a la API REST 47
Ilustración 51: Estructura del proyecto React 48
Ilustración 52: Acceso para usuarios administradores 49
Ilustración 53: Gestión de usuarios 49
Ilustración 54: Gestión de fabricantes 49
Ilustración 55: Detalle de una empresa 50
Ilustración 56: Detalle de un fabricante 50
Ilustración 57: Listado de Mis incidencias 50
Ilustración 58: Listado de incidencias y opciones de filtro 51
Ilustración 59: Detalle de una incidencia 51
Ilustración 60: Opción de añadir máquinas a una incidencia 52
Ilustración 61: Acciones posibles en una incidencia Pendiente 52
Ilustración 62: Listado de incidencias de mi empresa 53
Ilustración 63: Detalle de una incidencia y opción de añadir comentarios 53
Ilustración 64: Petición en Postman para crear un nuevo fabricante 54
Ilustración 65: Resultado de la ejecución de tests en Postman 54
Ilustración 66: Acceso a la aplicación desde Heroku 57

1. Introducción

1.1 Contexto y justificación del Trabajo

En la época en la que nos encontramos, la transformación digital ha adquirido un papel muy importante en todo tipo de empresas. Como a todos los sectores de la economía, este cambio también afecta a las empresas industriales que aplican las tecnologías de la información y cada vez tienden a ser más innovadoras, productivas y eficientes.

Este cambio, a menudo es difícil de aplicar en empresas pequeñas que no cuentan con grandes recursos para implantar herramientas de gestión adecuadas o que, en ocasiones, son reticentes a realizar cambios en su forma de trabajar.

El problema con el que muchas empresas se encuentran es que existe una gran variedad de ofertas de software informático que muchas veces no llega a cubrir todas sus necesidades y que hace que tengan que emplear varias herramientas distintas. Por otro lado, también suele ocurrir que una determinada solución informática ofrezca más funcionalidades de las que requieren y que hace que su implantación no sea adecuada y que no sea rentable por no poder aprovecharla completamente.

Por estos motivos, en muchas ocasiones resulta más apropiado recurrir al desarrollo de software a medida. Las ventajas que aportan este tipo de proyectos son las siguientes:

- El software se diseña para unas necesidades específicas y se adapta a la forma de trabajar de la empresa.
- Suele tener una interfaz más sencilla e intuitiva de utilizar
- Ofrece mayor flexibilidad ya que pueden realizarse cambios en función de los requerimientos de la compañía.

Para este proyecto en concreto, se plantea la necesidad de desarrollar un software a medida que cubra las necesidades de una empresa dedicada a ofrecer un servicio técnico de maquinaria de soldadura.

1.2 Objetivos del Trabajo

El objetivo del trabajo es desarrollar una **aplicación web que sirva como gestor de incidencias a una empresa de mantenimiento de maquinaria de soldadura**. La herramienta permitirá a los empleados de la empresa gestionar las incidencias que se produzcan en la maquinaria de sus clientes, permitiendo en todo momento ver el estado en el que se encuentran, los técnicos a los que están asignadas o la solución proporcionada en cada caso.

Los objetivos que persigue la empresa con esta herramienta son los siguientes:

- Realizar un seguimiento de las incidencias abiertas
- Contar con una base de datos de conocimiento para futuras incidencias
- Mejorar el servicio al cliente

La herramienta podrá ser utilizada por usuarios administradores para realizar la configuración del sistema, técnicos que trabajan en la empresa de mantenimiento que podrán ver las reparaciones en las que están trabajando y clientes que podrán ver las incidencias asociadas a su empresa.

La aplicación permitirá hacer búsquedas por fabricantes y productos, de forma que puedan consultar incidencias producidas en el pasado y obtener así información que pueda ser útil a la hora de reparar nuevas averías que se vayan produciendo.

Las funcionalidades principales de la aplicación serán las siguientes:

Administración

- Alta, baja y modificación de clientes, fabricantes, productos
- Alta, baja y modificación de máquinas vinculadas a cada cliente
- Alta, baja y modificación de usuarios

Incidencias

- Alta de incidencias
- Consulta de incidencias asociadas a un técnico
- Consulta de incidencias asociadas a una empresa
- Consulta de incidencias entre fechas
- Marcar incidencias como resueltas
- Cerrar incidencias

Comentarios

- Añadir comentario a una incidencia
- Consultar comentarios vinculados a una incidencia

Documentos

- Añadir un documento a una incidencia
- Consultar documentos vinculados a una incidencia

1.3 Enfoque y método seguido

La metodología que se va a seguir a la hora de desarrollar el proyecto es la denominada **Waterfall** o **Cascada**, en la que las fases del ciclo de vida se realizan de manera lineal y el inicio de una fase no se produce hasta que finaliza la anterior. Las fases en las que se divide son las siguientes:

- Análisis
- Diseño

- Implementación
- Pruebas

Aunque es una metodología que en los últimos años se está sustituyendo por otros modelos iterativos cuenta con ventajas por las que se ha decidido aplicar en este proyecto:

- Permite estructurar el proyecto de forma clara en aquellos casos en los que se cuenta con fases claramente diferenciadas y con hitos definidos. En este caso, por la existencia de diferentes entregas a lo largo del curso resulta fácil definir dichos hitos.

- Se trata de un proyecto de corta duración por lo que el modelo en cascada resulta apropiado ya que no van a producirse grandes cambios respecto al análisis inicial.
- Es una metodología apropiada para proyectos con objetivo claro y solución conocida. Este proyecto cumple con estas características, se trata de un proyecto con poco riesgo y que se desarrolla en circunstancias previsibles.

La herramienta que se va a desarrollar estará dividida en dos partes:

- **Back-end**: consiste en una API REST conectada a una base de datos postgreSQL que se desarrollará utilizando Spring Boot

- **Front-end**: la capa de presentación se desarrollará mediante React js.

Tecnologías

- JDK 1.8
- Eclipse IDE for Java Developers 2020-12
- Spring Boot
- Maven
- PostgreSQL
- Spring Data JPA
- Font-end: React, Bootstrap

La decisión de desarrollar las partes front-end y back-end de forma separada se toma basándose en las siguientes ventajas con las que cuenta este modelo frente a otros en los que las interfaces también están en el lado del servidor:

- La lógica de negocio y los datos se encuentran centralizados y se exponen a través de una API que da servicio a múltiples interfaces de usuario. Esto implica que en el futuro podrían implementarse otras interfaces, por ejemplo, para su uso desde aplicaciones móviles.

- El intercambio de información entre cliente y servidor es ligero al tratarse de ficheros JSON

- Es un modelo más escalable, ya que permite aumentar más fácilmente los recursos en caso de que una de las partes así lo requiera.

- Los interfaces web cuentan con mayor rapidez y fluidez

En cuanto a la decisión de la tecnología a utilizar para el front-end, se plantea el uso de frameworks de JavaScript como Angular, React o Vue que son los más utilizados actualmente para este tipo de proyectos.

En la siguiente tabla podemos ver las principales características de cada uno de ellos:

	Vue	React	Angular
Rendimiento	DOM virtual Mejor rendimiento	DOM virtual Mejor rendimiento	DOM real Peor rendimiento
Tamaño	80 KB Adecuado para aplicaciones ligeras	100 KB Adecuado para aplicaciones ligeras	500 KB Adecuado para aplicaciones más pesadas
Popularidad	★★	★★★	★
Flexibilidad	La biblioteca oficial ofrece lo básico. Requiere librerías adicionales.	La biblioteca oficial ofrece lo básico. Requiere librerías adicionales.	Ofrece todo, no necesita agregar librerías
Curva de aprendizaje	Baja	Baja	Elevada

A continuación, podemos ver los resultados de una encuesta realizada por Stack Overflow, en la que se muestran los framework más apreciados por la comunidad de programadores. Podemos ver que en primer lugar aparecen React y Vue:

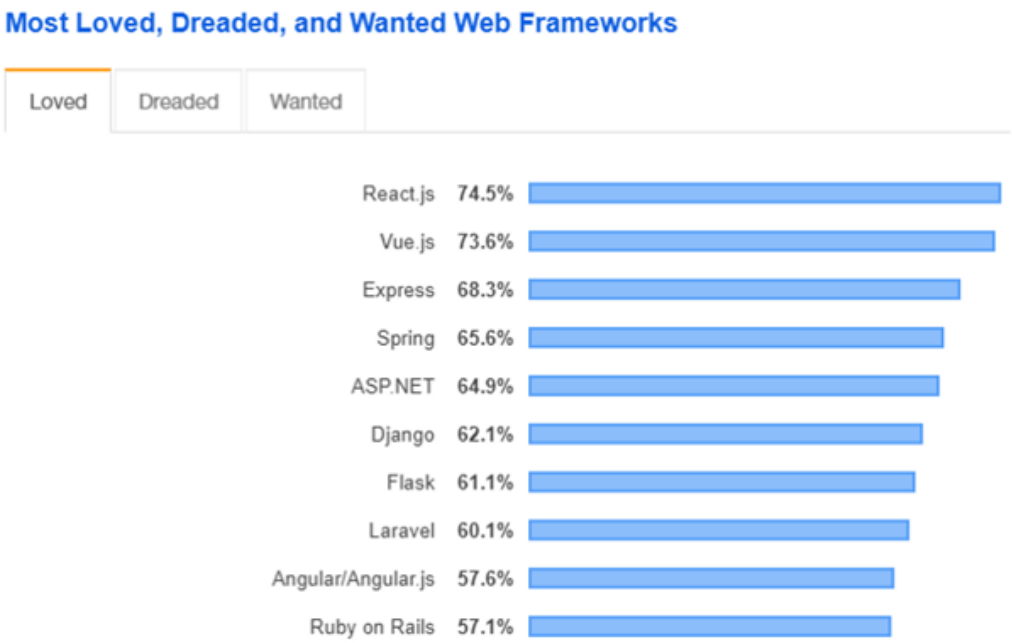
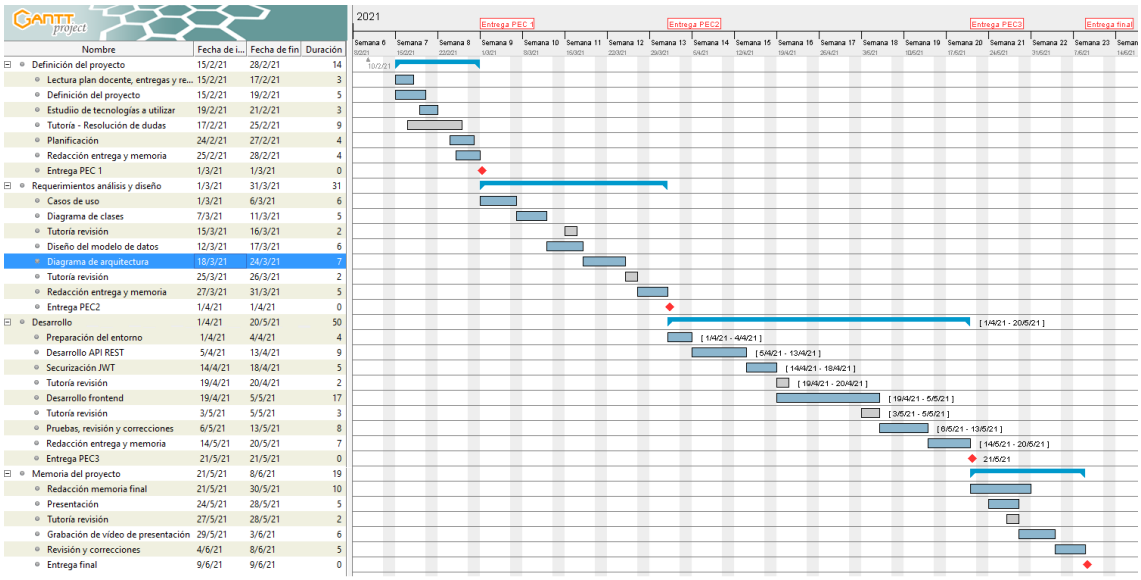


Ilustración 1: Frameworks más queridos por los programadores

La conclusión principal, viendo información sobre las 3 tecnologías es que todas suponen una buena opción a la hora de desarrollar una aplicación. Debido a que la herramienta a desarrollar es ligera y que React cuenta con mayor popularidad entre la comunidad de programadores se decide utilizarlo en este proyecto.

1.4 Planificación del Trabajo

El proyecto se divide en 4 fases definidas a partir del calendario de la asignatura TFG que cuenta con 4 entregas. A continuación, se muestra el diagrama de Gantt en el que se pueden ver las tareas en las que se desglosa cada fase y la duración en días de cada una de ellas:



Nombre	Fecha de i...	Fecha de fin	Duración
Definición del proyecto	15/2/21	28/2/21	14
Lectura plan docente, entregas y re...	15/2/21	17/2/21	3
Definición del proyecto	15/2/21	19/2/21	5
Estudiio de tecnologías a utilizar	19/2/21	21/2/21	3
Tutoría - Resolución de dudas	17/2/21	25/2/21	9
Planificación	24/2/21	27/2/21	4
Redacción entrega y memoria	25/2/21	28/2/21	4
Entrega PEC 1	1/3/21	1/3/21	0
Requerimientos análisis y diseño	1/3/21	31/3/21	31
Casos de uso	1/3/21	6/3/21	6
Diagrama de clases	7/3/21	11/3/21	5
Tutoría revisión	15/3/21	16/3/21	2
Diseño del modelo de datos	12/3/21	17/3/21	6
Diagrama de arquitectura	18/3/21	24/3/21	7
Tutoría revisión	25/3/21	26/3/21	2
Redacción entrega y memoria	27/3/21	31/3/21	5
Entrega PEC2	1/4/21	1/4/21	0
Desarrollo	1/4/21	20/5/21	50
Preparación del entorno	1/4/21	4/4/21	4
Desarrollo API REST	5/4/21	13/4/21	9
Securización JWT	14/4/21	18/4/21	5
Tutoría revisión	19/4/21	20/4/21	2
Desarrollo frontend	19/4/21	5/5/21	17
Tutoría revisión	3/5/21	5/5/21	3
Pruebas, revisión y correcciones	6/5/21	13/5/21	8
Redacción entrega y memoria	14/5/21	20/5/21	7
Entrega PEC3	21/5/21	21/5/21	0
Memoria del proyecto	21/5/21	8/6/21	19
Redacción memoria final	21/5/21	30/5/21	10
Presentación	24/5/21	28/5/21	5
Tutoría revisión	27/5/21	28/5/21	2
Grabación de vídeo de presentación	29/5/21	3/6/21	6
Revisión y correcciones	4/6/21	8/6/21	5
Entrega final	9/6/21	9/6/21	0

Ilustración 2: Diagrama de Gantt

1.4.1. Fase 1: Definición del proyecto

En esta primera fase se realiza la toma de contacto con el plan docente de la asignatura y cada una de las entregas que se debe realizar a lo largo del semestre. Además, se establece el primer contacto con el tutor y se acuerda el proyecto a desarrollar y un plan de trabajo.

Definición del proyecto	15/2/21	28/2/21	14
Lectura plan docente, entregas y re...	15/2/21	17/2/21	3
Definición del proyecto	15/2/21	19/2/21	5
Estudiio de tecnologías a utilizar	19/2/21	21/2/21	3
Tutoría - Resolución de dudas	17/2/21	25/2/21	9
Planificación	24/2/21	27/2/21	4
Redacción entrega y memoria	25/2/21	28/2/21	4
Entrega PEC 1	1/3/21	1/3/21	0

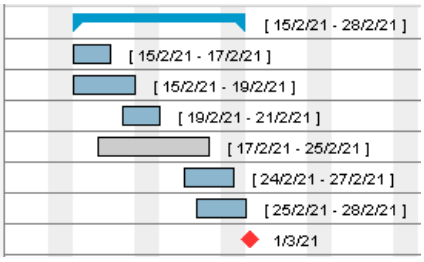


Ilustración 3: Planificación Fase 1

1.4.2. Fase 2: Requerimientos, análisis y diseño

En esta fase se realiza el análisis y diseño de la herramienta a implementar. Incluye la definición de casos de uso, diseño de la base de datos y la realización de diagramas de clases y arquitectura de la aplicación.

Requerimientos análisis y diseño	1/3/21	31/3/21	31
Casos de uso	1/3/21	6/3/21	6
Diagrama de clases	7/3/21	11/3/21	5
Tutoría revisión	15/3/21	16/3/21	2
Diseño del modelo de datos	12/3/21	17/3/21	6
Diagrama de arquitectura	18/3/21	24/3/21	7
Tutoría revisión	25/3/21	26/3/21	2
Redacción entrega y memoria	27/3/21	31/3/21	5
Entrega PEC2	1/4/21	1/4/21	0

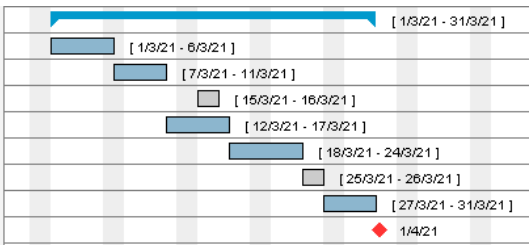


Ilustración 4: Planificación Fase 2

1.4.3. Fase 3: Desarrollo

En esta fase se lleva a cabo el desarrollo de la aplicación. Siguiendo la metodología en cascada, se divide el desarrollo en bloques que se van implementando de manera secuencial.

Desarrollo	1/4/21	20/5/21	50
Preparación del entorno	1/4/21	4/4/21	4
Desarrollo API REST	5/4/21	13/4/21	9
Securización JWT	14/4/21	18/4/21	5
Tutoría revisión	19/4/21	20/4/21	2
Desarrollo frontend	19/4/21	5/5/21	17
Tutoría revisión	3/5/21	5/5/21	3
Pruebas, revisión y correcciones	6/5/21	13/5/21	8
Redacción entrega y memoria	14/5/21	20/5/21	7
Entrega PEC3	21/5/21	21/5/21	0

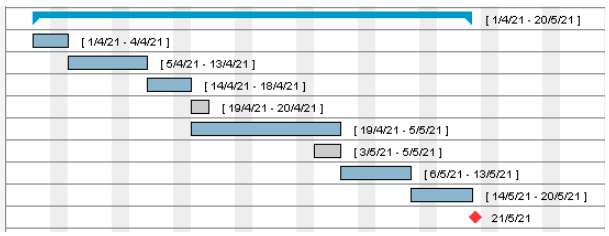


Ilustración 5: Planificación Fase 3

1.4.4. Fase 4: Memoria del proyecto

A lo largo de las 3 fases anteriores se habrá ido trabajando en las distintas partes de la memoria final. En esta última fase, se recopilará toda la documentación para completar la memoria final y se preparará la presentación para la entrega final.

Memoria del proyecto	21/5/21	8/6/21	19
Redacción memoria final	21/5/21	30/5/21	10
Presentación	24/5/21	28/5/21	5
Tutoría revisión	27/5/21	28/5/21	2
Grabación de vídeo de presentación	29/5/21	3/6/21	6
Revisión y correcciones	4/6/21	8/6/21	5
Entrega final	9/6/21	9/6/21	0

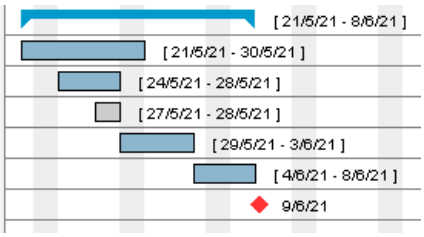


Ilustración 6: Planificación Fase 4

1.5 Breve sumario de productos obtenidos

Se desarrollan los siguientes productos:

- Fichero comprimido con el código fuente desarrollado
- Manual para la puesta en marcha de la herramienta
- Memoria final del trabajo
- Presentación en PowerPoint
- Vídeo con la presentación virtual del trabajo

1.6 Breve descripción de los otros capítulos de la memoria

En los siguientes capítulos de la memoria se sigue la siguiente estructura:

1.6.1 Análisis funcional de la aplicación

En este capítulo se describen los requisitos funcionales de la aplicación obtenidos a partir de reuniones realizadas y se definen los actores principales del sistema. A partir de esta información, se definen los casos de uso que se abordarán en el proyecto.

1.6.2. Diseño técnico

En este capítulo, se realiza el diseño del sistema a partir del análisis funcional realizado. Se divide en los siguientes apartados:

- Diagrama de clases: se definen las entidades, los atributos que contiene cada una y las relaciones entre ellas.

- Modelo de datos: diseño de las tablas que contiene la base de datos de la aplicación. Se definen las columnas de cada una de ellas y las claves primarias y foráneas.

- Diagramas de arquitectura: diseño de la arquitectura que se utilizará a la hora de desarrollar la aplicación.

2. Análisis funcional

Una vez definidos los objetivos del proyecto y sus funcionalidades principales, se realiza el análisis del sistema a desarrollar. En este apartado se definen los actores principales, el diagrama de casos de uso y sus fichas.

2.1. Actores

Los actores de la aplicación son los siguientes:

- **Administrador**: tiene acceso a la sección Configuración donde puede gestionar fabricantes, productos, modelos, empresas y máquinas asociadas a cada empresa. Además, puede gestionar los usuarios que tienen acceso a la herramienta.

- **Técnico**: tiene acceso a la sección de mantenimiento donde puede ver la información de todas las incidencias creadas en la plataforma. Podrá filtrar las incidencias para poder ver de forma sencilla aquellas que tenga asignadas. Un técnico puede abrir nuevas incidencias, editarlas y cambiar su estado.

- **Cliente**: tiene acceso a la sección de mantenimiento donde puede ver las incidencias asociadas a maquinaria de su empresa. Además, puede añadir comentarios a las incidencias.

2.2. Casos de uso

Con el objetivo de documentar los requisitos del sistema, se han utilizado los casos de uso que permiten definir cuál debe ser el comportamiento del sistema en cada caso. A continuación, se puede ver un diagrama con los casos de uso de la aplicación:

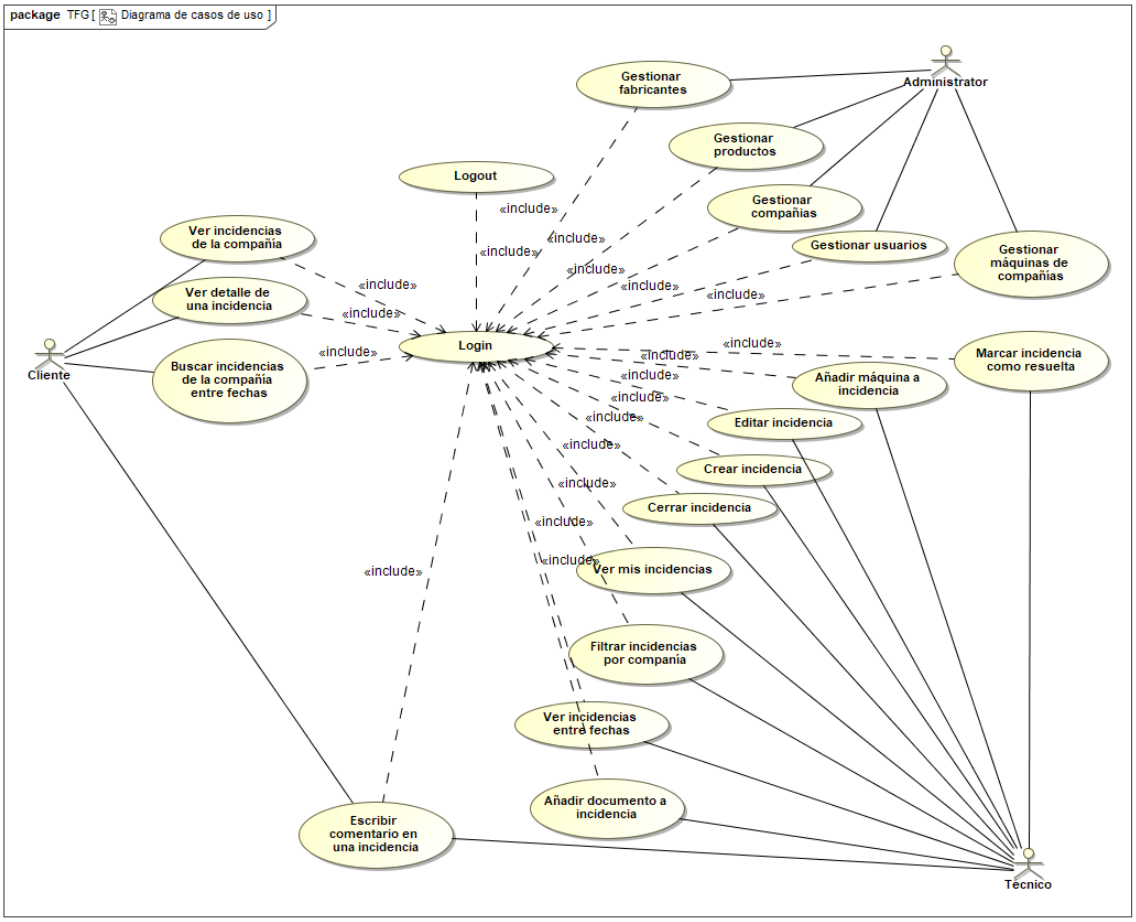


Ilustración 7: Diagrama de casos de uso

A modo de resumen, se incluye un listado con todos los casos de uso:

UC1 – Login
UC2 – Logout
UC3 - Crear incidencia
UC4 - Editar incidencia
UC5 – Añadir máquina a una incidencia
UC6 - Buscar incidencias de la compañía entre fechas
UC7 - Ver detalle de una incidencia
UC8 - Ver incidencias de la compañía
UC9 - Marcar incidencia como resuelta
UC10 - Cerrar incidencia
UC11 - Ver mis incidencias
UC12 - Filtrar incidencias entre fechas
UC13 - Filtrar incidencias por compañía
UC14 - Escribir un comentario en una incidencia
UC15 - Subir un documento a una incidencia
UC16 - Listar fabricantes
UC17 - Crear un fabricante
UC18 - Editar un fabricante
UC19 - Eliminar un fabricante
UC20 – Vincular máquinas a una empresa

A continuación, se incluyen las fichas de los casos de uso:

Caso de uso	UC1 - Login en el sistema
Actores	Usuario
Descripción	El usuario inicia sesión para acceder a su cuenta
Condición previa	El usuario debe estar registrado en la plataforma
Escenario principal de éxito	1. El usuario accede a la web 2. El sistema muestra la pantalla de login 3. El usuario introduce su usuario y password y hace click en “Acceder” 4. El sistema valida los datos introducidos y muestra la pantalla de inicio
Escenarios alternativos	4.a. Los datos introducidos no son correctos 4.a.1. Muestra un mensaje de error y vuelve al punto 2

Caso de uso	UC2 - Logout del sistema
Actores	Usuario
Descripción	El usuario cierra sesión en la plataforma
Condición previa	El usuario debe estar autenticado en la plataforma
Escenario principal de éxito	1. El usuario selecciona la opción “Cerrar sesión” 2. El sistema muestra un mensaje indicando que se ha cerrado la sesión correctamente
Escenarios alternativos	

Caso de uso	UC3 - Crear incidencia
Actores	Técnico
Descripción	El usuario crea una nueva incidencia en el sistema
Condición previa	El usuario debe estar autenticado en la plataforma
Escenario principal de éxito	1. El usuario pulsa el botón “Crear incidencia” 2. El sistema muestra un formulario de creación de incidencia 3. El usuario indica título, descripción y empresa de la incidencia 4. El sistema almacena la incidencia y muestra la pantalla de detalle
Escenarios alternativos	3.a. El usuario deja vacío la empresa, título y/o la descripción 3.a.1. El sistema muestra un mensaje de error y vuelve al punto 2

Caso de uso	UC4 - Editar incidencia
Actores	Técnico
Descripción	El usuario modifica datos de una incidencia
Condición previa	El usuario debe estar autenticado en la plataforma La incidencia debe tener estado “Pendiente”
Escenario principal de éxito	1. El usuario accede al detalle de una incidencia y pulsa el botón “Editar” 2. El sistema muestra el formulario de edición de la incidencia 3. El usuario realiza modificaciones en el título, descripción de la incidencia 4. El sistema guarda los cambios y vuelve al detalle de la incidencia
Escenarios alternativos	3.a. El usuario deja vacío la empresa, título y/o la descripción 3.a.1. El sistema muestra un mensaje de error y vuelve al punto 2

Caso de uso	UC5 – Añadir máquina a una incidencia
Actores	Técnico
Descripción	El usuario añade las máquinas afectadas por una incidencia
Condición previa	El usuario debe estar autenticado en la plataforma La incidencia debe tener estado “Pendiente”
Escenario principal de éxito	1. El usuario accede al detalle de una incidencia y pulsa el botón “Añadir máquina” 2. El sistema muestra el listado de máquinas de la empresa de la incidencia 3. El usuario selecciona las máquinas y pulsa “Añadir” 4. El sistema guarda los cambios y vuelve al detalle de la incidencia
Escenarios alternativos	3.a. El usuario no selecciona ninguna máquina 3.a.1. El sistema muestra un mensaje de error y vuelve al punto 2

Caso de uso	UC6 - Buscar incidencias de la compañía entre fechas
Actores	Cliente
Descripción	El usuario obtiene las incidencias de su empresa entre dos fechas
Condición previa	El usuario debe estar autenticado en la plataforma
Escenario principal de éxito	1. El usuario accede al apartado Incidencias 2. El sistema muestra un formulario de búsqueda con una fecha de inicio y fecha fin 3. El usuario indica las fechas y hace click en la opción Buscar 4. El sistema muestra el listado con las incidencias de la compañía del usuario con fecha en el rango indicado
Escenarios alternativos	3.a. El usuario un rango de fechas no válido 3.a.1. El sistema muestra un mensaje de error y vuelve al punto 2

Caso de uso	UC7 - Ver detalle de una incidencia
Actores	Cliente
Descripción	El usuario accede al detalle de una incidencia
Condición previa	El usuario debe estar autenticado en la plataforma
Escenario principal de éxito	1. El usuario accede al apartado de incidencias de su empresa 2. Selecciona la opción de visualizar una incidencia concreta
Escenarios alternativos	

Caso de uso	UC8 - Ver incidencias de la compañía
Actores	Cliente
Descripción	El usuario obtiene las incidencias asignadas a su compañía
Condición previa	El usuario debe estar autenticado en la plataforma
Escenario principal de éxito	1. El usuario accede al apartado de incidencias 2. El sistema muestra una tabla con las incidencias existentes para la compañía del usuario

Caso de uso	UC9 - Marcar incidencia como resuelta
Actores	Técnico
Descripción	El usuario marca como resuelta una incidencia La incidencia debe estar en estado Pendiente
Condición previa	El usuario debe estar autenticado en la plataforma
Escenario principal de éxito	1. El usuario accede al detalle de una incidencia 2. El usuario hace click en la opción Resolver incidencia 3. El sistema muestra un formulario para indicar solución y el importe facturado 4. El usuario rellena el formulario y guarda los datos 5. El sistema almacena los datos introducidos por el usuario y cambia el estado de la incidencia a Resuelta
Escenarios alternativos	4.a. El usuario no rellena alguno de los datos del formulario 4.a.1. El sistema vuelve al punto 3

Caso de uso	UC10 - Cerrar incidencia
Actores	Técnico
Descripción	El usuario cierra una incidencia del sistema La incidencia debe estar en estado Pendiente
Condición previa	El usuario debe estar autenticado en la plataforma
Escenario principal de éxito	1. El usuario accede al detalle de una incidencia 2. El usuario hace click en la opción Cerrar incidencia 3. El sistema solicita confirmación al usuario 4. El usuario confirma la acción 5. El sistema cambia el estado de la incidencia a Cerrada
Escenarios alternativos	4.a. El usuario cancela la acción 4.a.1. Finaliza el caso de uso

Caso de uso	UC11 - Ver mis incidencias
Actores	Técnico
Descripción	El usuario obtiene el listado de las incidencias que tiene asignadas
Condición previa	El usuario debe estar autenticado en la plataforma
Escenario principal de éxito	1. El usuario accede a la aplicación 2. El sistema muestra un listado con las incidencias asignadas al usuario
Escenarios alternativos	

Caso de uso	UC12 - Filtrar incidencias entre fechas
Actores	Técnico
Descripción	El usuario obtiene las incidencias creadas entre dos fechas
Condición previa	El usuario debe estar autenticado en la plataforma
Escenario principal de éxito	1. El usuario accede al apartado Incidencias 2. El sistema muestra un formulario de búsqueda con una fecha de inicio y fecha fin 3. El usuario indica las fechas y hace click en la opción Buscar 4. El sistema muestra el listado con las incidencias con fecha en el rango indicado
Escenarios alternativos	3.a. El usuario un rango de fechas no válido 3.a.1. El sistema muestra un mensaje de error y vuelve al punto 2

Caso de uso	UC13 - Filtrar incidencias por compañía
Actores	Técnico
Descripción	El usuario obtiene las incidencias de una compañía concreta
Condición previa	El usuario debe estar autenticado en la plataforma
Escenario principal de éxito	1. El usuario accede al apartado Incidencias 2. El sistema muestra un formulario de búsqueda con un desplegable para elegir compañía 3. El usuario indica una compañía y hace click en la opción Buscar 4. El sistema muestra el listado con las incidencias de la compañía elegida
Escenarios alternativos	

Caso de uso	UC14 - Escribir un comentario en una incidencia
Actores	Técnico o cliente
Descripción	El usuario añade un comentario a una incidencia
Condición previa	El usuario debe estar autenticado en la plataforma
Escenario principal de éxito	1. El usuario accede al detalle de una incidencia 2. El sistema muestra un formulario para introducir comentario 3. Indica el texto del comentario que desea publicar 4. El sistema almacena el comentario y la fecha en la que se ha realizado y este pasa a ser visible desde la incidencia
Escenarios alternativos	3.a. El usuario deja vacío el texto 3.a.1. El sistema vuelve al punto 2

Caso de uso	UC15 - Subir un documento a una incidencia
Actores	Técnico
Descripción	El usuario añade un documento a una incidencia existente
Condición previa	El usuario debe estar autenticado en la plataforma
Escenario principal de éxito	1. El usuario accede al detalle de una incidencia 2. Hace click en la opción de subir documento 3. El sistema muestra un formulario para seleccionar fichero 4. El usuario selecciona el documento a subir 5. El sistema almacena el documento y este pasa a ser visible desde la incidencia
Escenarios alternativos	4.a. Se produce un error a la hora de subir el documento 4.a.1. El sistema vuelve al punto 3 4.b. El usuario no selecciona ningún documento 4.b.1. El sistema vuelve al punto 3

Mantenimientos apartado administración: estos casos de uso se repiten para fabricantes, productos, usuarios y empresas.

Caso de uso	UC16 - Listar fabricantes
Actores	Administrador
Descripción	El usuario accede al listado de fabricantes existentes en el sistema
Condición previa	El usuario debe estar autenticado en la plataforma
Escenario principal de éxito	1. El usuario accede al apartado de fabricantes 2. El sistema muestra una tabla con los fabricantes existentes
Escenarios alternativos	

Caso de uso	UC17 - Crear un fabricante
Actores	Administrador
Descripción	El usuario crea un nuevo fabricante en el sistema
Condición previa	El usuario debe estar autenticado en la plataforma
Escenario principal de éxito	1. El usuario accede al apartado de fabricantes 2. El usuario hace click en la opción de crear fabricante 3. El sistema muestra el formulario de creación de un fabricante 4. El usuario indica el nombre del fabricante y hace click en crear 5. El sistema almacena el fabricante
Escenarios alternativos	4.a. El usuario deja vacío el nombre 4.a.1. El sistema vuelve al punto 3

Caso de uso	UC18 - Editar un fabricante
Actores	Administrador
Descripción	El usuario modifica un fabricante del sistema
Condición previa	El usuario debe estar autenticado en la plataforma
Escenario principal de éxito	3. El usuario accede al apartado de fabricantes 4. El usuario hace click en la opción de editar un fabricante concreto 5. El sistema muestra el formulario de edición de un fabricante 6. El usuario modifica el nombre del fabricante y hace click en guardar 7. El sistema almacena los cambios del fabricante
Escenarios alternativos	4.a. El usuario deja vacío el nombre 4.a.1. El sistema vuelve al punto 3

Caso de uso	UC19 - Eliminar un fabricante
Actores	Administrador
Descripción	El usuario elimina un fabricante del sistema
Condición previa	El usuario debe estar autenticado en la plataforma
Escenario principal de éxito	1. El usuario accede al apartado de fabricantes 2. El usuario hace click en la opción de eliminar un fabricante concreto 3. El sistema solicita confirmación al usuario para eliminar el fabricante 4. El usuario confirma la eliminación 5. El sistema borra los datos del fabricante
Escenarios alternativos	4.a. El usuario cancela la acción 4.a.1. Finaliza el caso de uso

Relación entre Máquinas y empresas:

Caso de uso	UC20 – Vincular máquinas a una empresa
Actores	Administrador
Descripción	El usuario añade una máquina a una empresa cliente
Condición previa	El usuario debe estar autenticado en la plataforma
Escenario principal de éxito	1. El usuario accede al detalle de una empresa 2. El usuario hace click en la opción de añadir una máquina 3. El sistema solicita fabricante y producto, además del número de serie de la máquina 4. El usuario confirma la creación 5. El sistema almacena los datos de la máquina
Escenarios alternativos	4.a. El usuario no indica producto y/o número de serie 4.a.1. El sistema muestra un mensaje de error y vuelve al punto 3

3. Diseño técnico

Una vez realizado el análisis y definido cuál debe ser el comportamiento de la herramienta, se realiza el diseño del sistema. En este apartado se incluye el diagrama de clases

3.1. Diagrama de clases

El diagrama de clases que se ha diseñado es el siguiente:

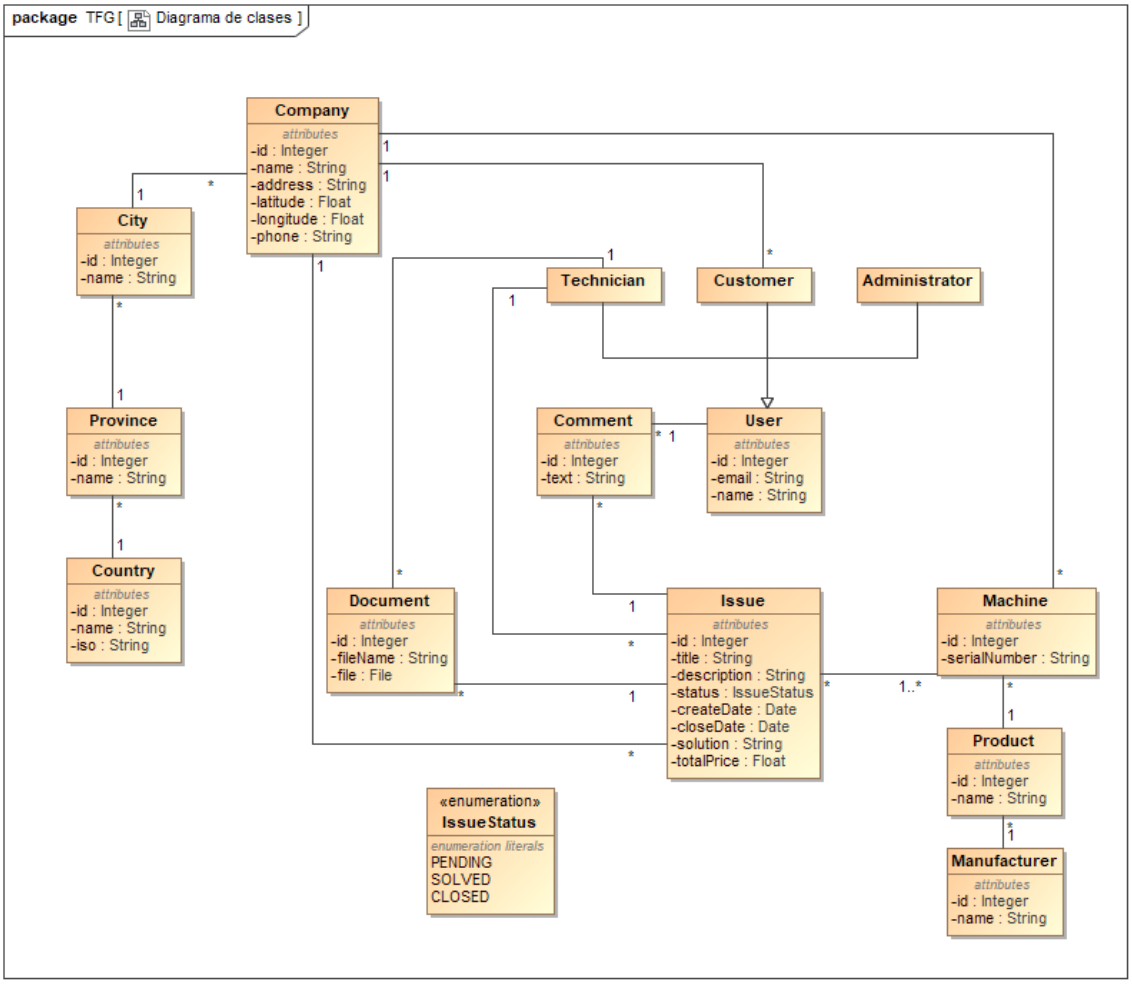


Ilustración 8: Diagrama de clases

En el diagrama se incluyen las siguientes clases:

User: modela los usuarios que tendrá la aplicación, de esta clase heredan Technician, Customer y Administrator que corresponden a los 3 tipos de usuarios que pueden existir.

Manufacturer: modela a los fabricantes de máquinas que se darán de alta en la aplicación.

Product: cada fabricante tiene distintos productos que se modelan con esta clase.

Company: cada una de las compañías cliente de la empresa de mantenimiento. Los usuarios de tipo Cliente tendrán asociada una empresa.

Machine: mediante esta clase se modelan las máquinas que tiene una empresa cliente en concreto. Cada una de estas máquinas tendrá un número de serie y tendrá asociado un determinado producto.

Issue: es la clase que modela una incidencia que afectará a una o varias máquinas de una empresa. Cada incidencia estará vinculada a un técnico y tendrá 3 estados posibles:

- Pendiente: para incidencias abiertas aún sin solución.
- Resuelta: cuando las máquinas hayan sido reparadas.
- Cerrada: se utilizará para los casos en los que no se pueda o se decida no realizar ninguna reparación.

Se debe tener en cuenta como condición que una incidencia solo podrá tener máquinas de la empresa a la que esté vinculada.

3.2. Diseño del modelo de datos

El siguiente diagrama muestra las distintas tablas que contendrá la base de datos:

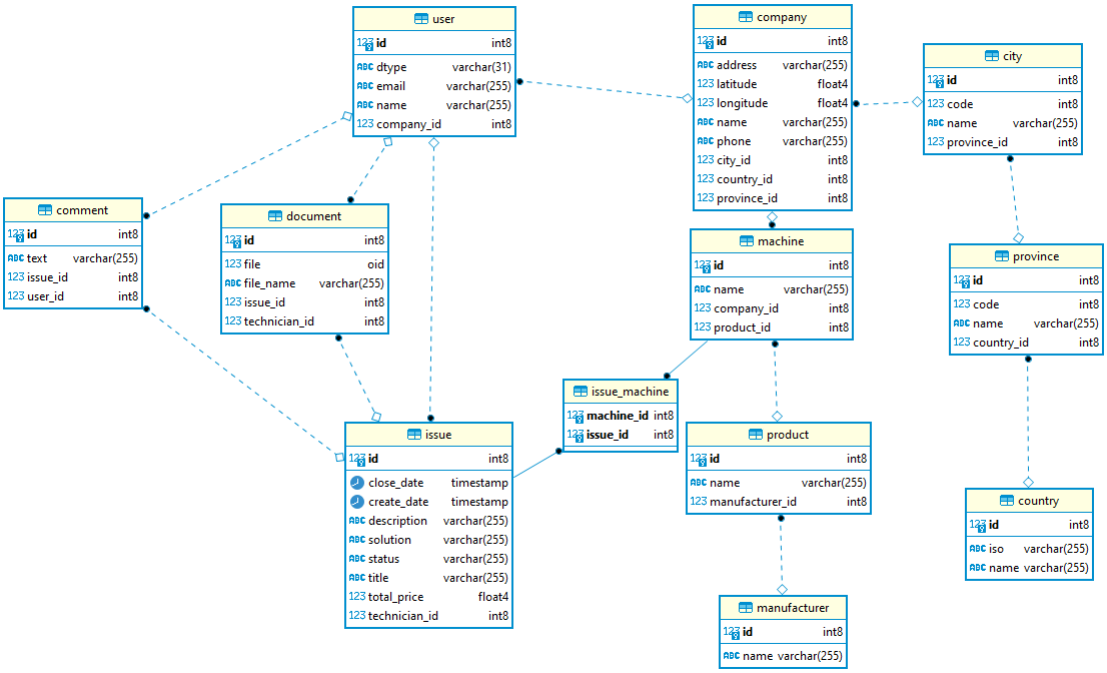


Ilustración 9: Diagrama del modelo de datos

Las decisiones más importantes tomadas en este punto son las siguientes:

3.2.1. Herencia entre User y Technician, Customer y Administrator

Entre las distintas soluciones posibles se decide utilizar la herencia de tabla única, se define una tabla user que contendrá todos los usuarios sea cual sea su subtipo. Existirá una columna que contendrá el tipo y permitirá discriminar el subtipo al que pertenece. Los motivos que han llevado a esta decisión son los siguientes:

- Se trata del diseño más sencillo ya que utiliza una única tabla. Se evita el uso de uniones de tablas o de joins.
- No existen gran cantidad de atributos específicos de los subtipos, el único es la compañía para los usuarios clientes por lo que no existirán gran cantidad de valores NULL para los casos en los que no aplica.
- No se preve la creación de nuevos tipos de usuarios, lo que provocaría tener que añadir nuevas columnas para nuevos atributos.

3.2.2. Relación n a m entre incidencias y máquinas

Para la relación n a m se crea una tabla intermedia issue_machine que tiene como clave primaria los identificadores de máquina e incidencia.

3.2.3. Almacenamiento de ficheros

La entidad Document contiene un fichero que debe almacenarse. Para ello, en la tabla document se ha definido una columna de tipo oid, utilizado para persistir objetos en bases de datos.

3.3. Diseño de arquitectura

El proyecto se realiza aplicando el patrón de arquitectura en tres capas: Presentación, Negocio e Integración.

3.3.1. Punto de vista de la computación

En el siguiente diagrama se representa el punto de vista de la computación. Se han dividido 5 componentes distintos: Incidencias, Usuarios, Catalogo, Localización y Clientes.

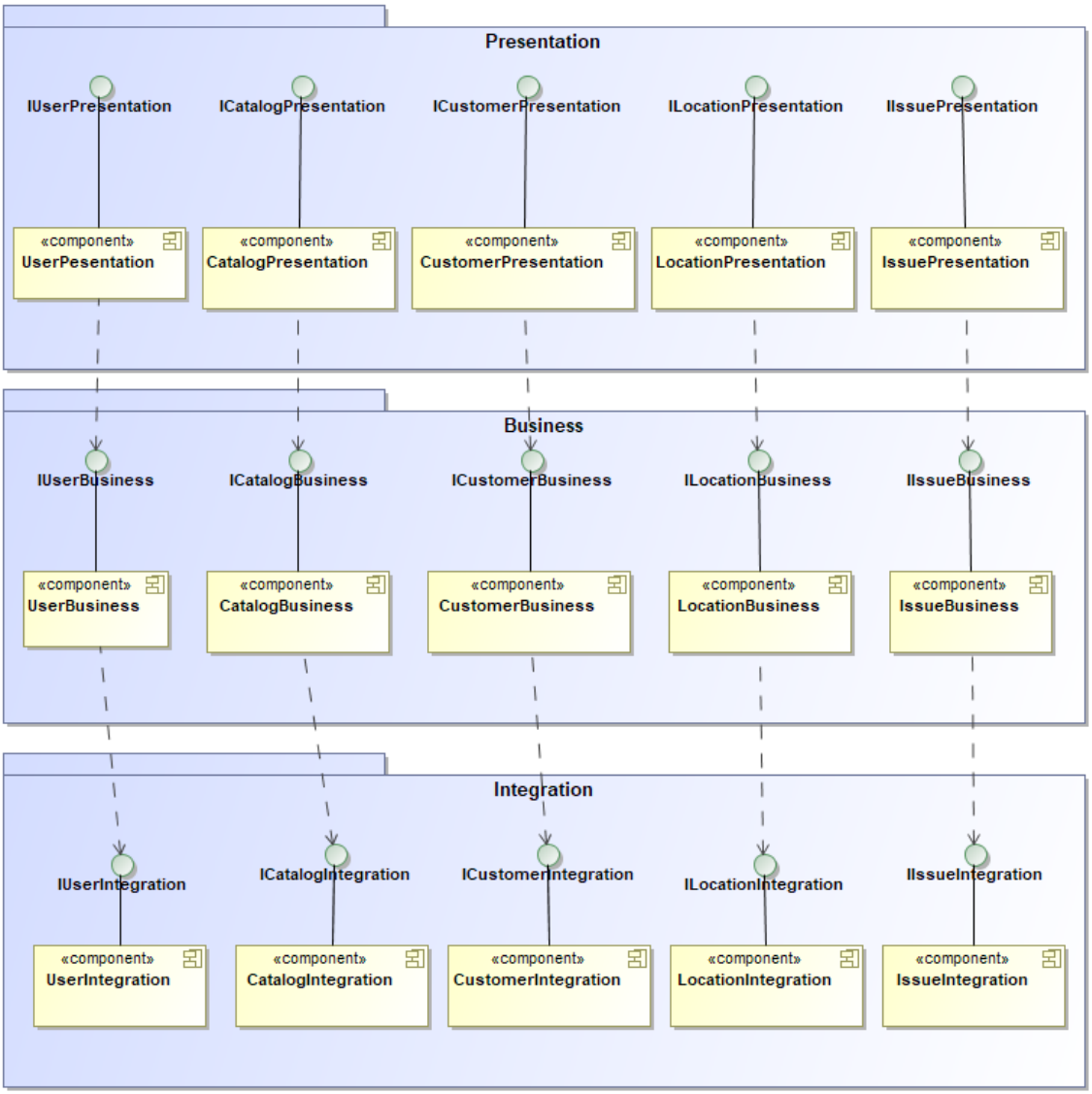


Ilustración 10: Punto de vista de computación

3.3.1.1. Capa de presentación

A continuación se muestran las operaciones de las interfaces de presentación:

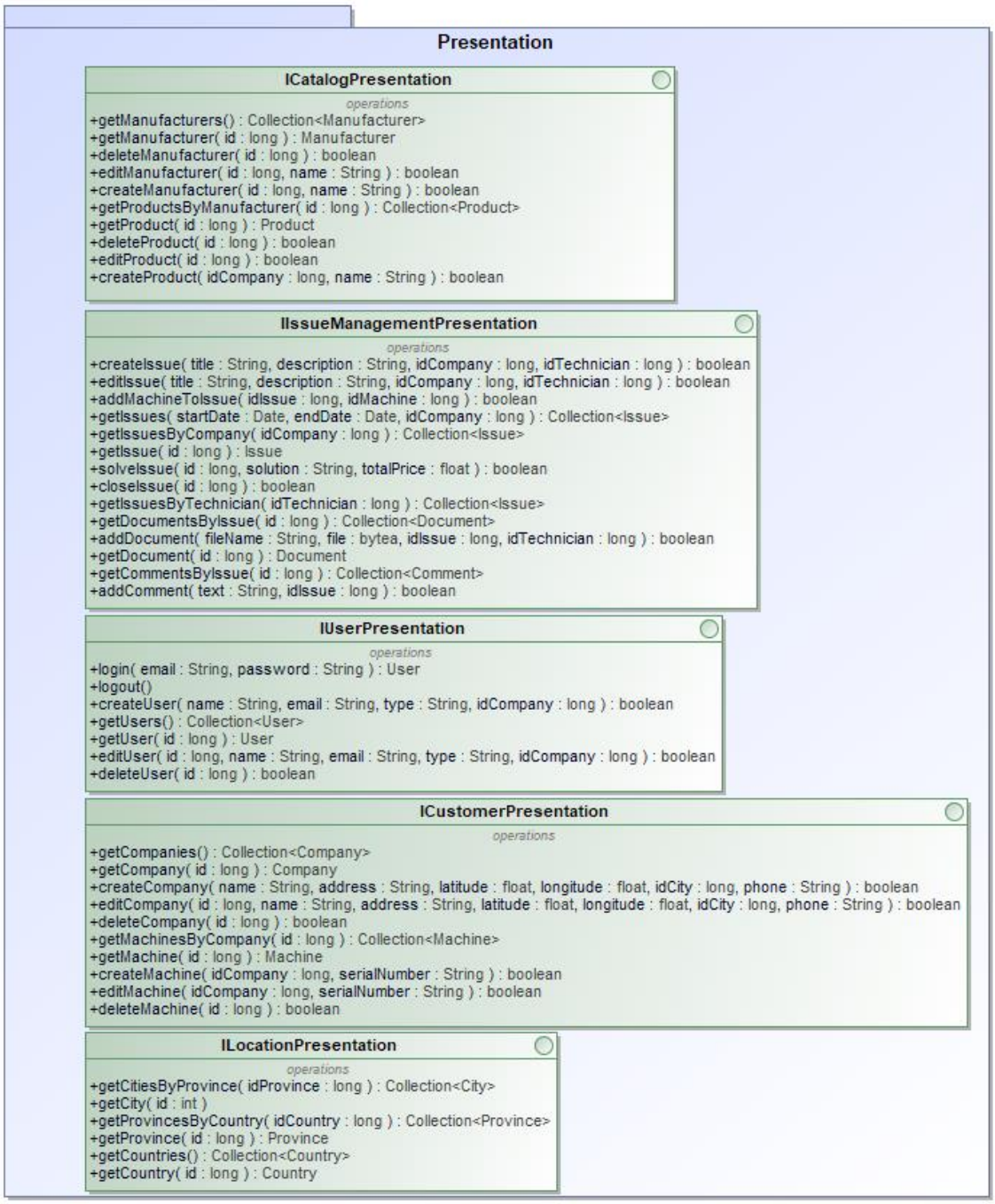


Ilustración 11: Punto de vista de computación. Presentación.

3.3.1.2. Capa de Negocio

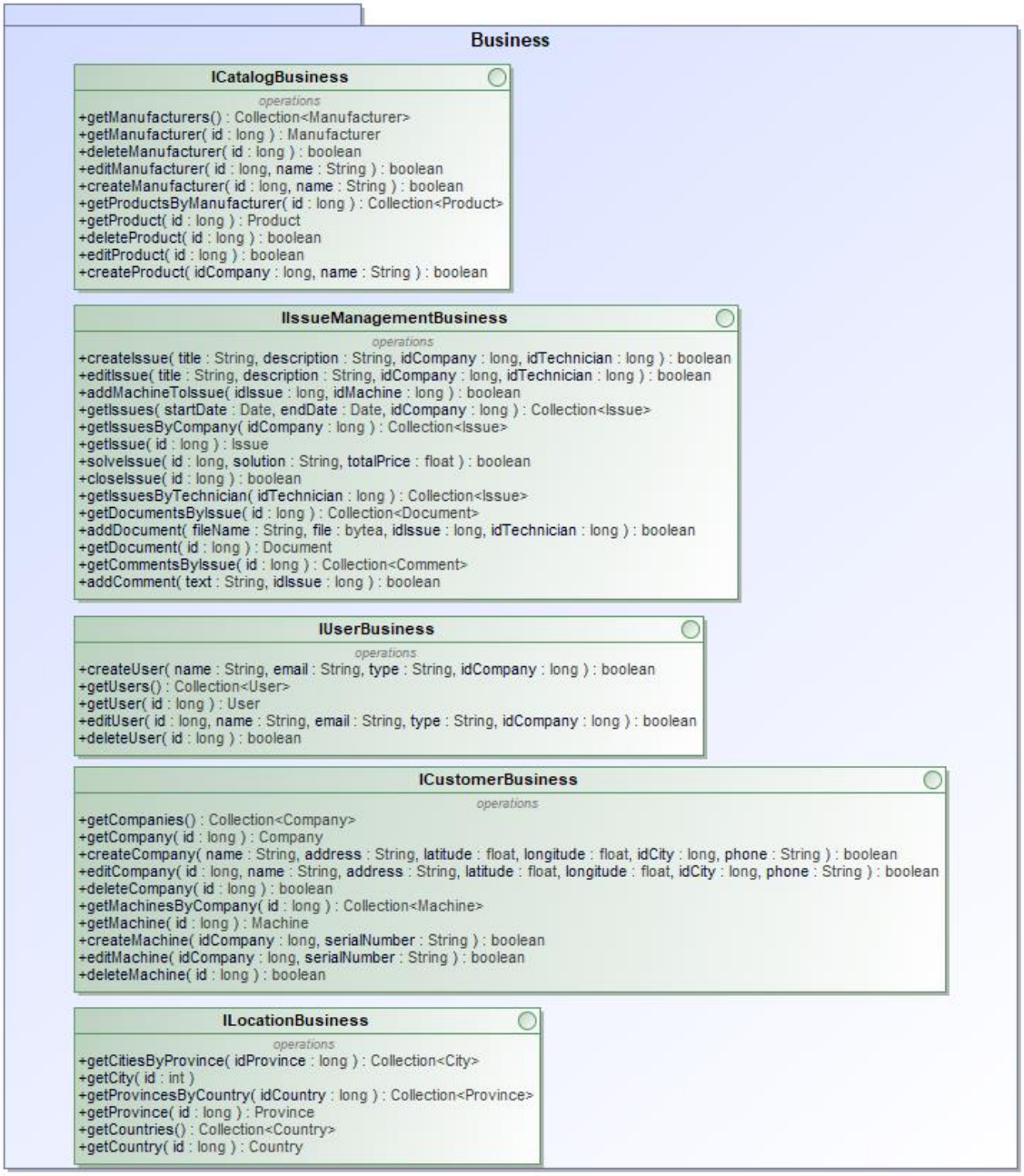


Ilustración 12: Punto de vista de computación. Negocio

3.3.1.3. Capa de Integración

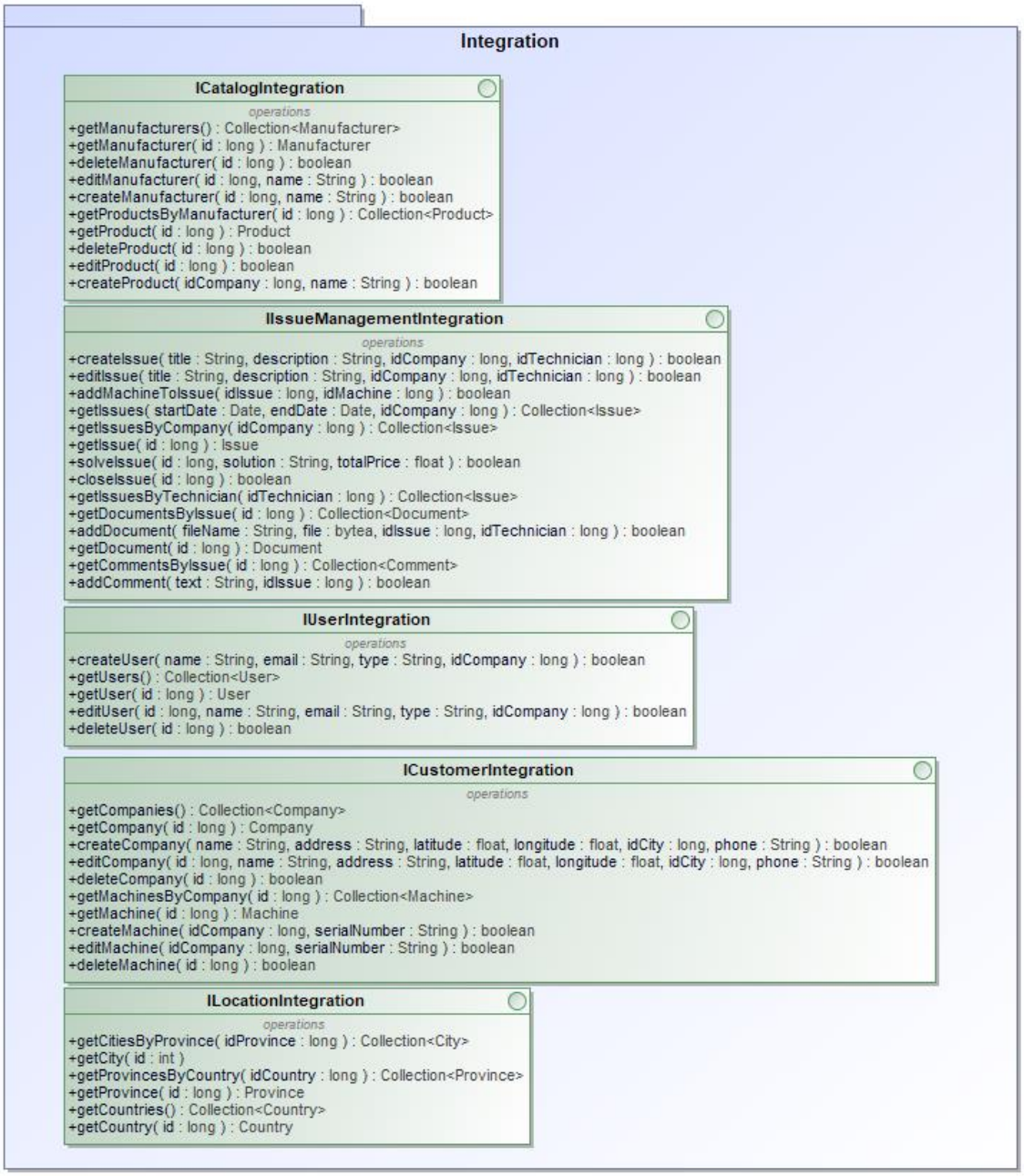


Ilustración 13: Punto de vista de computación. Integración.

3.3.2. Refinamiento del punto de vista de la computación

Tras realizar el diseño de los componentes del proyecto como punto de partida, se van a ir refinando, aplicando distintos patrones de desarrollo.

3.3.2.1. Capa de presentación

Se aplica el patrón MVC en el que se separan las capas de Modelo, Vista y Controlador.

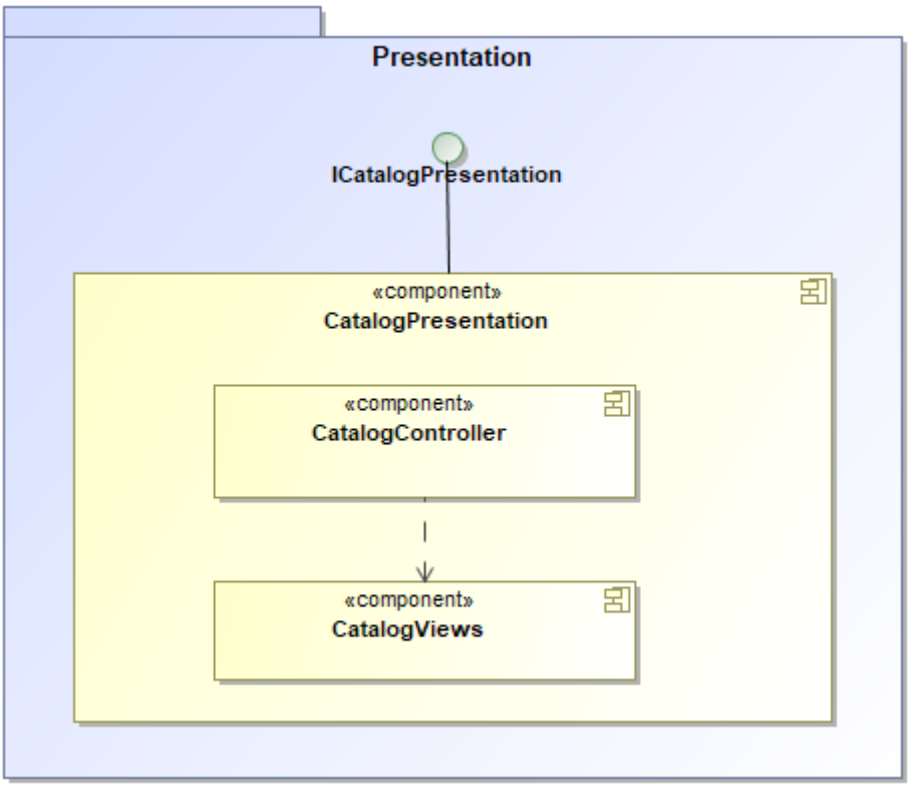


Ilustración 14: Capa de presentación, componente Catálogo

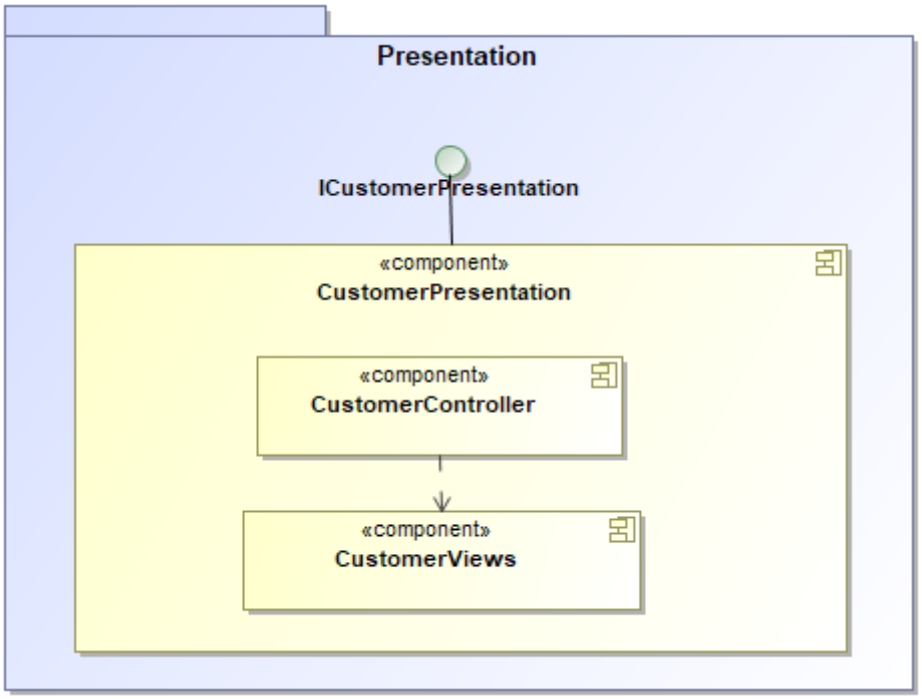


Ilustración 15: Capa de presentación, componente Cliente

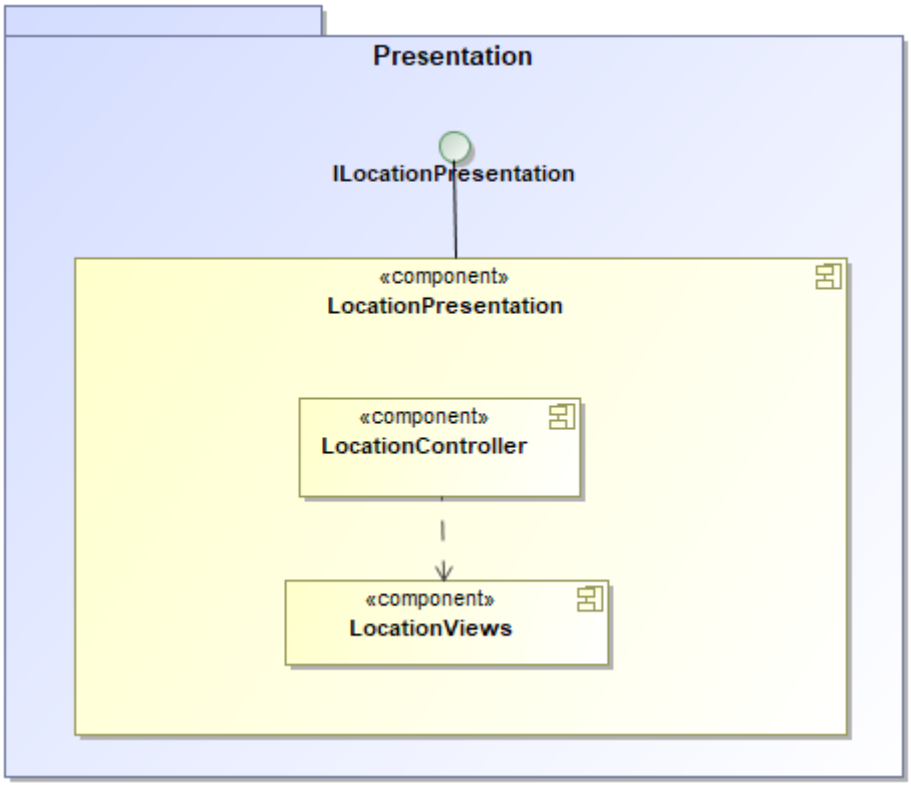


Ilustración 16: Capa de presentación, componente Localización

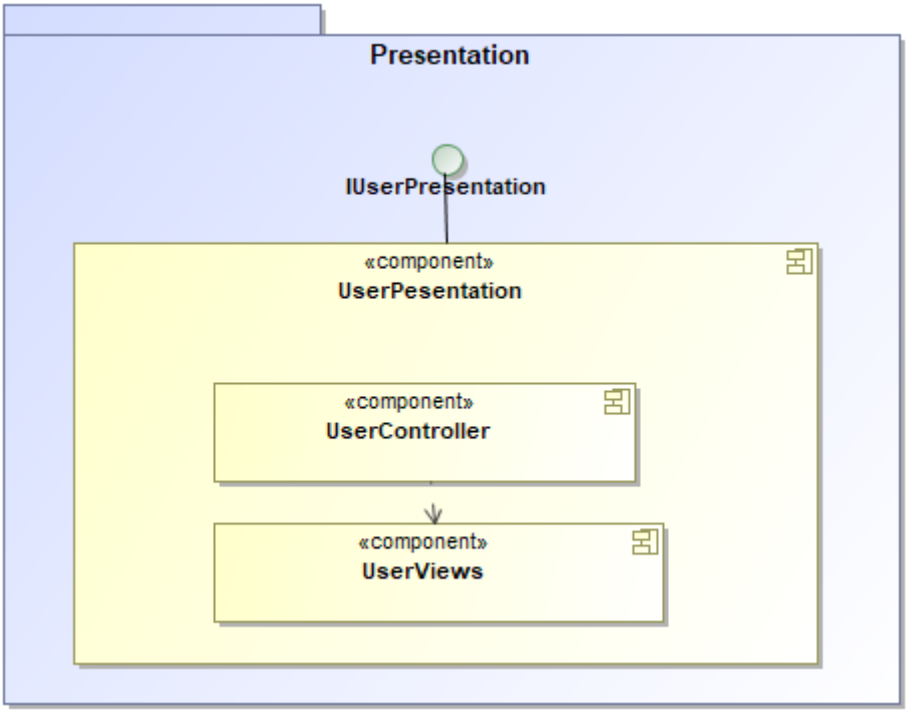


Ilustración 17: Capa de presentación, componente Usuario

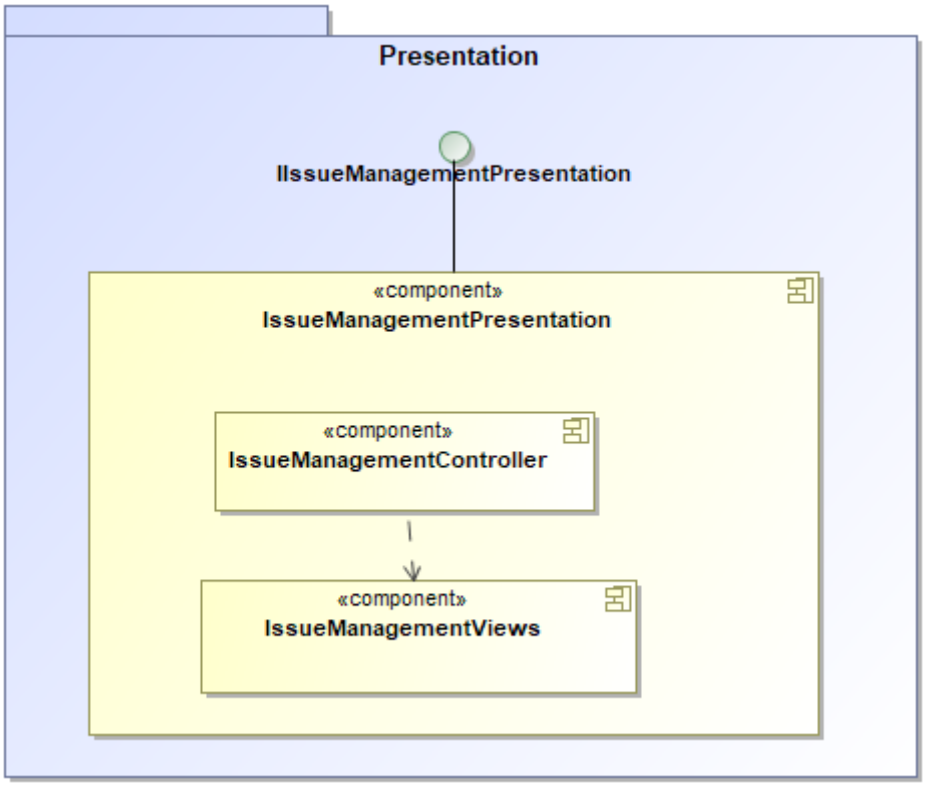


Ilustración 18: Capa de presentación, componente Incidencia

Además, se aplica el patrón “Front Controller”, en el que existe un controlador que es el encargado de recibir todas las peticiones y redirigirlas al controlador correspondiente.

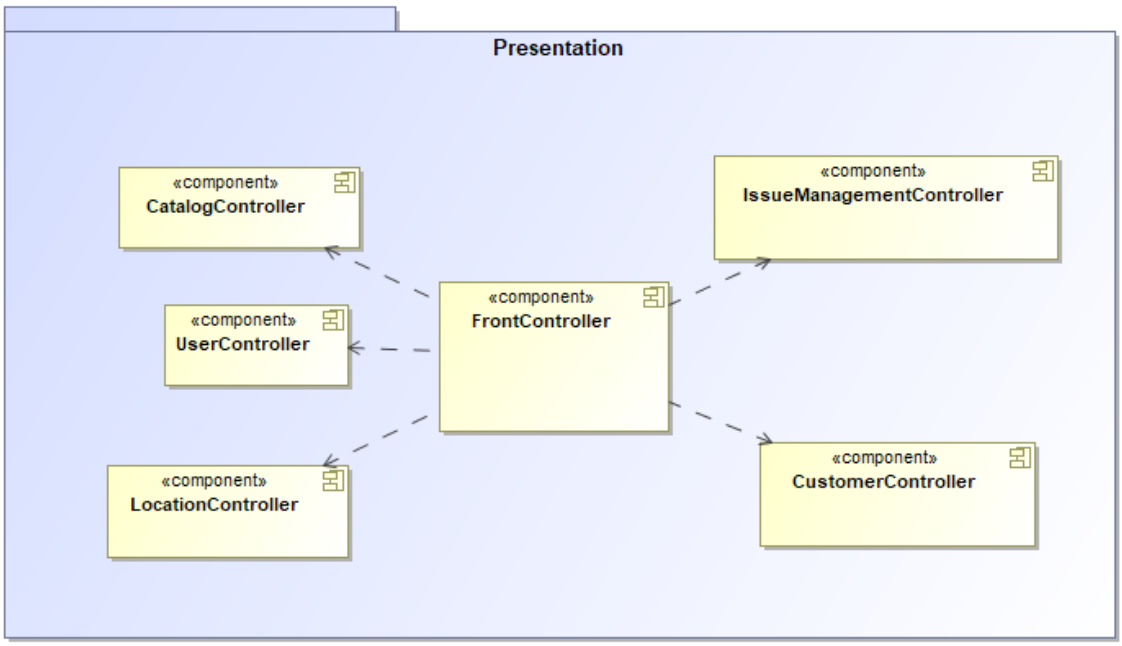


Ilustración 19: Capa de Presentación, Front Controller

3.3.2.2. Capa de negocio

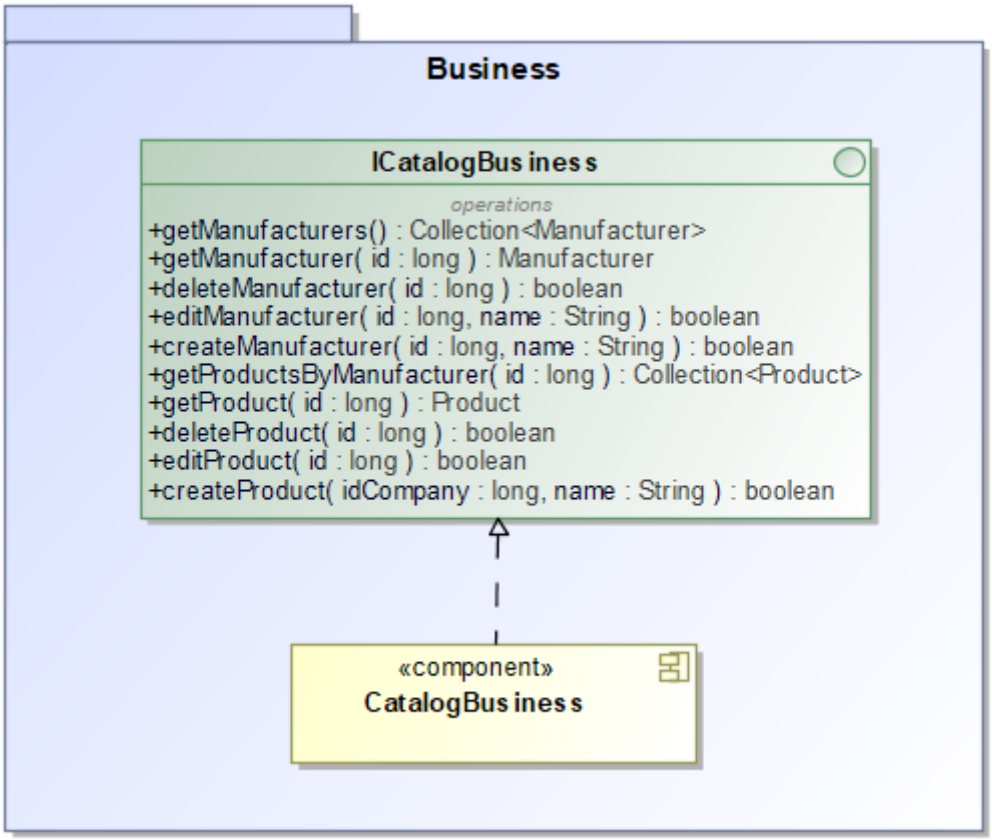


Ilustración 20: Capa de negocio, componente Catálogo

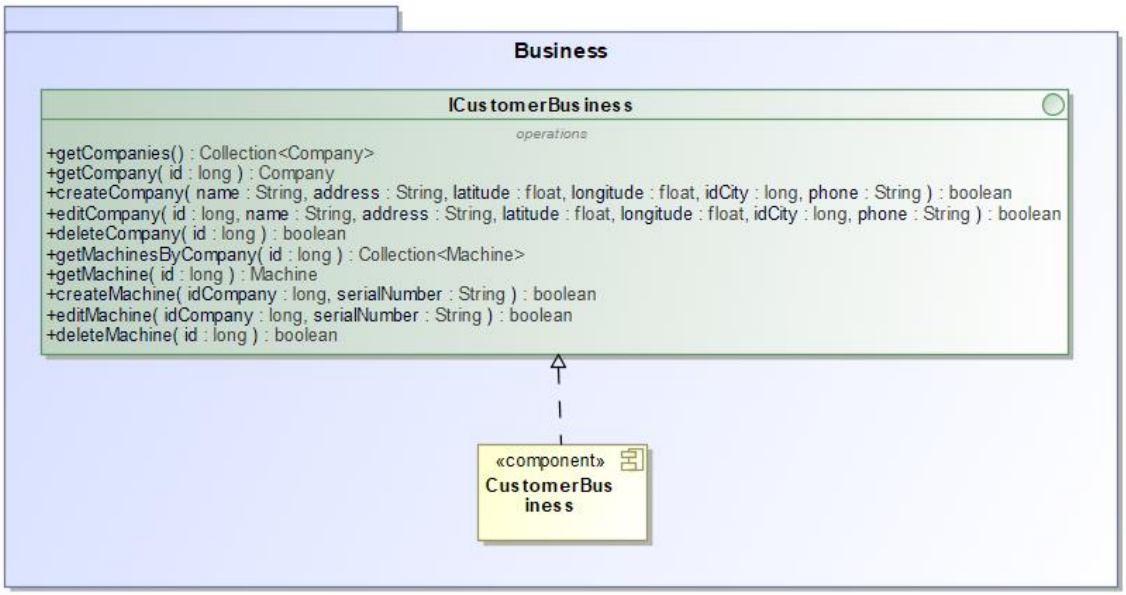


Ilustración 21: Capa de negocio, componente Cliente

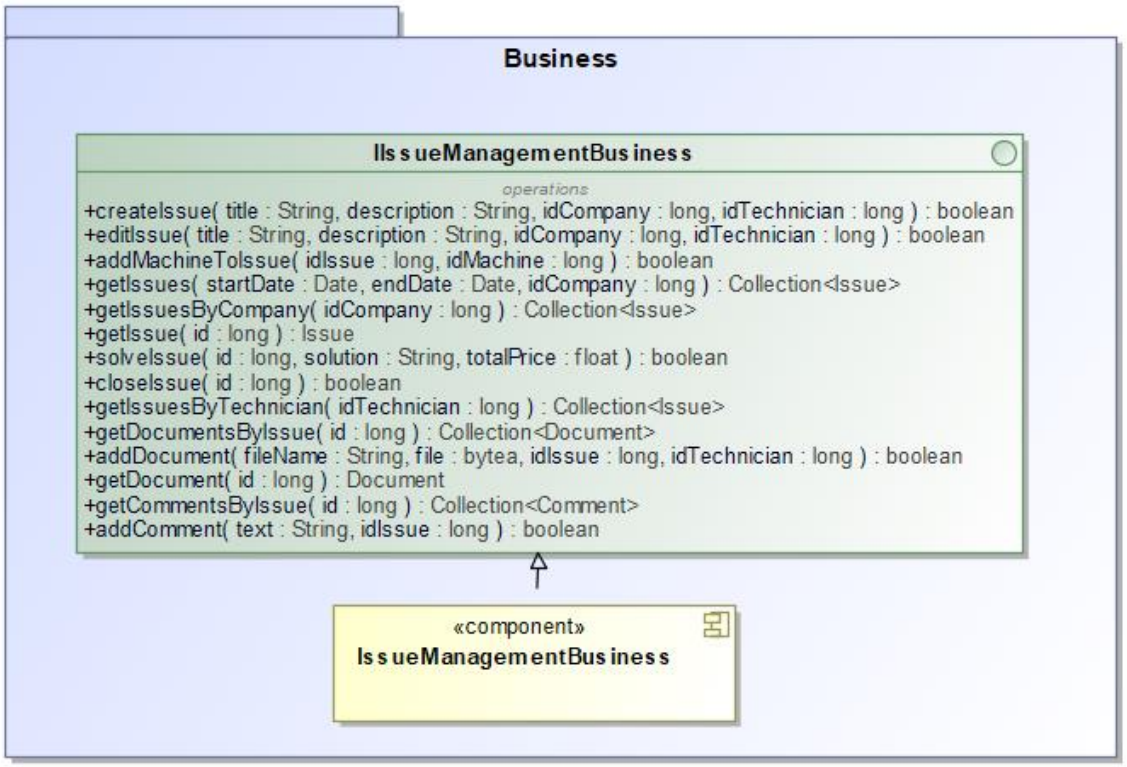


Ilustración 22: Capa de negocio, componente Incidencias

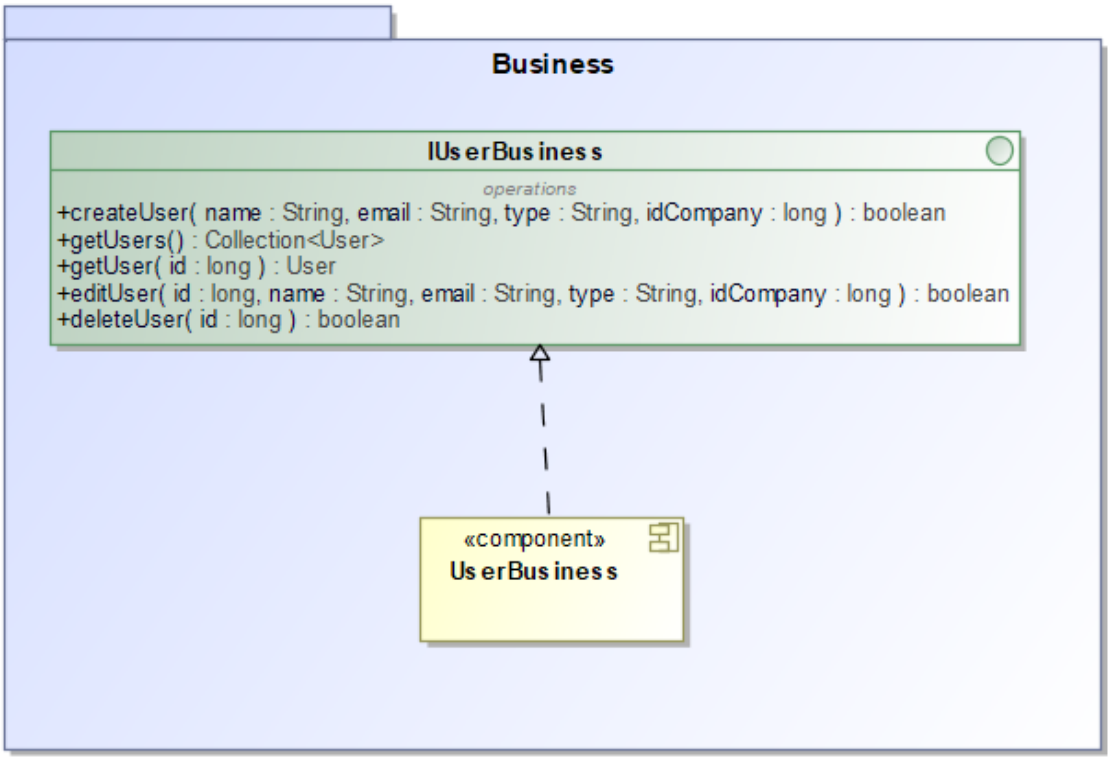


Ilustración 23: Capa de negocio, componente Usuario

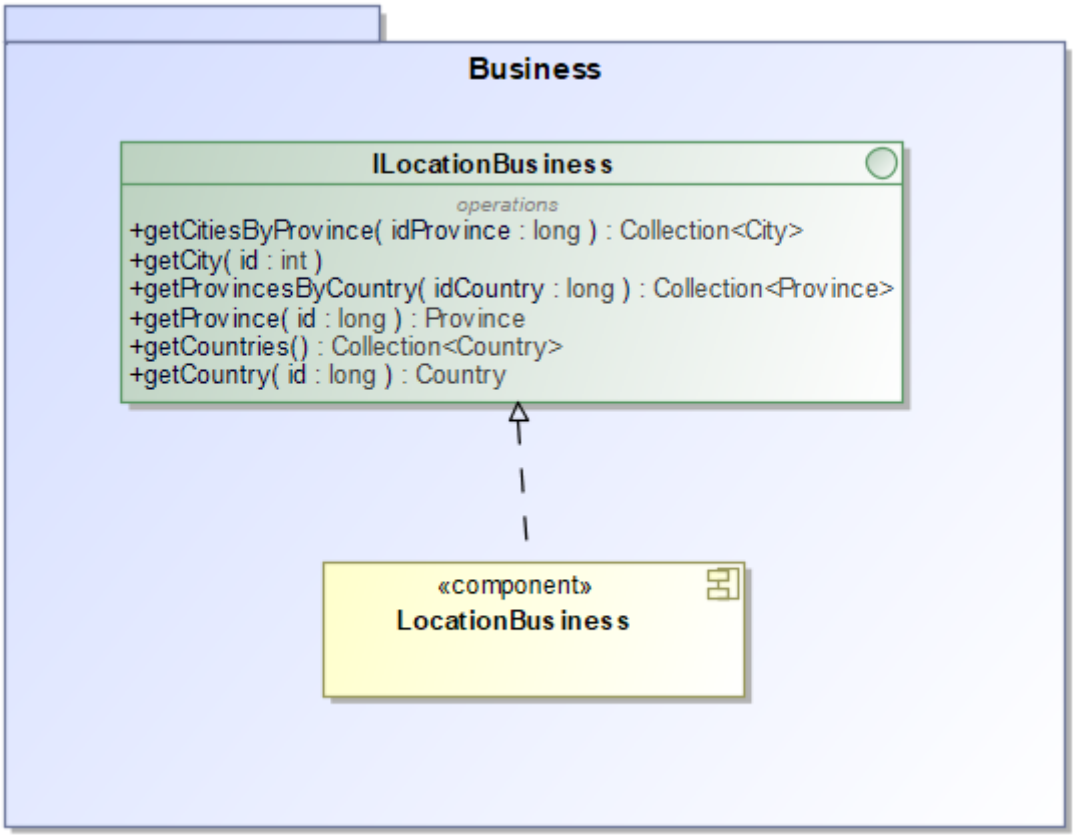


Ilustración 24: Capa de negocio, componente Localización

3.3.2.3. Capa de integración

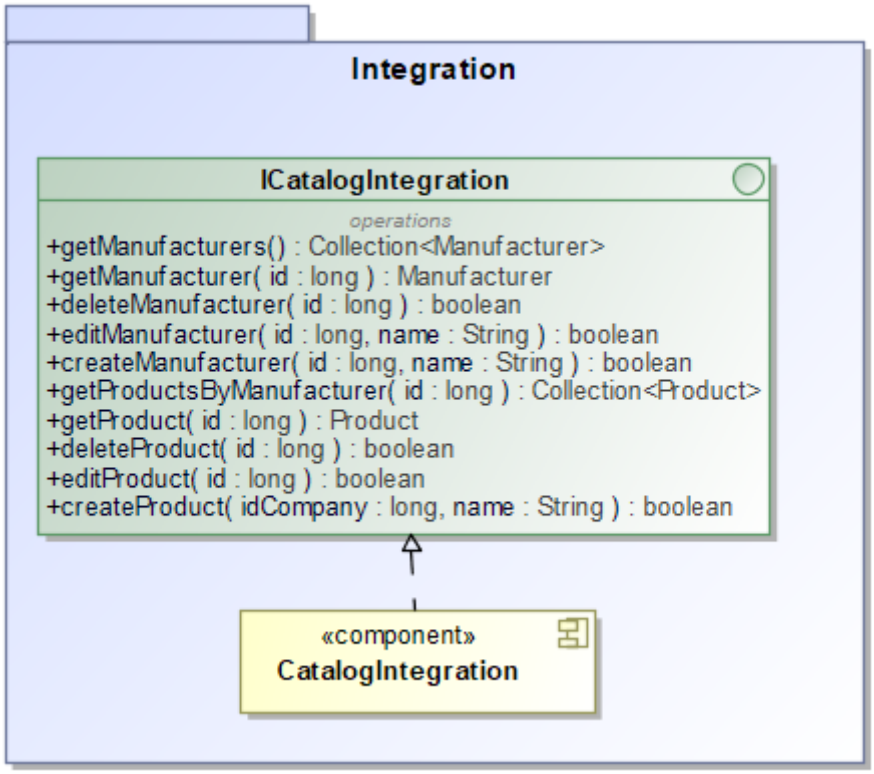


Ilustración 25: Capa de integración, componente Catálogo

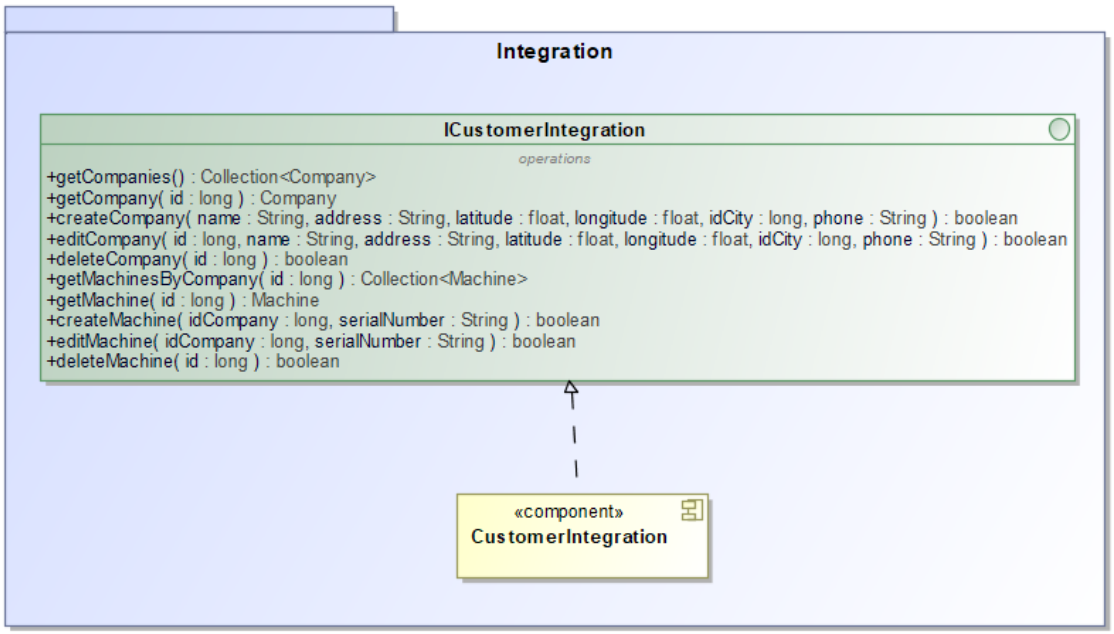


Ilustración 26: Capa de integración, componente Cliente

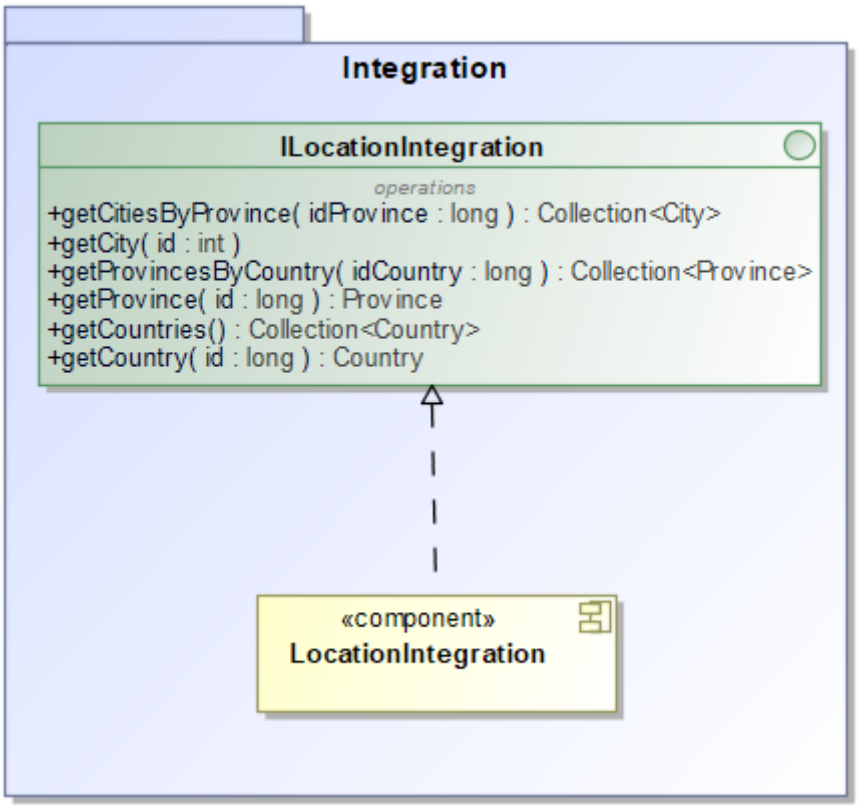


Ilustración 27: Capa de integración, componente Localización

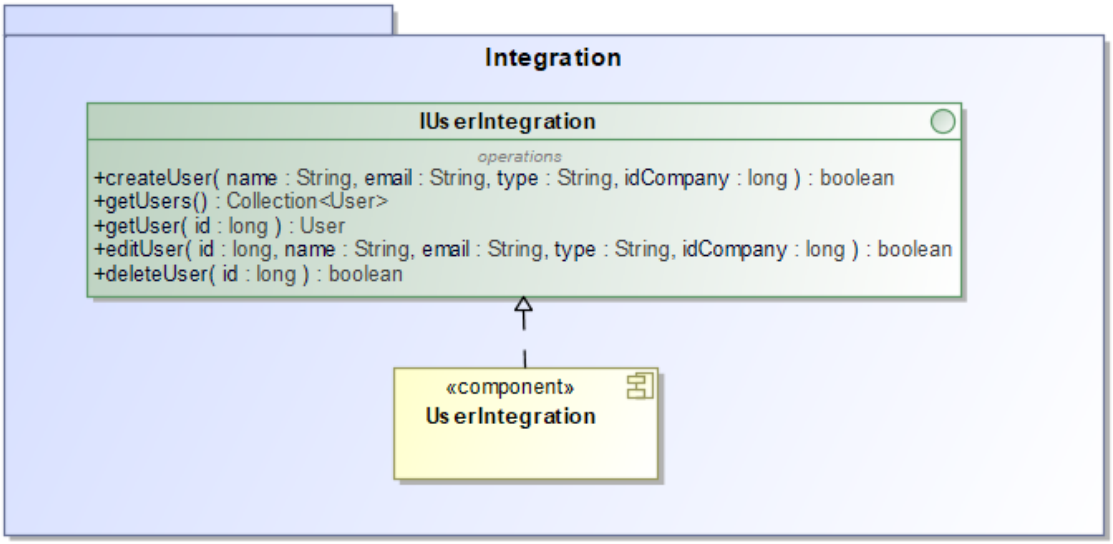


Ilustración 28: Capa de integración, componente Usuario

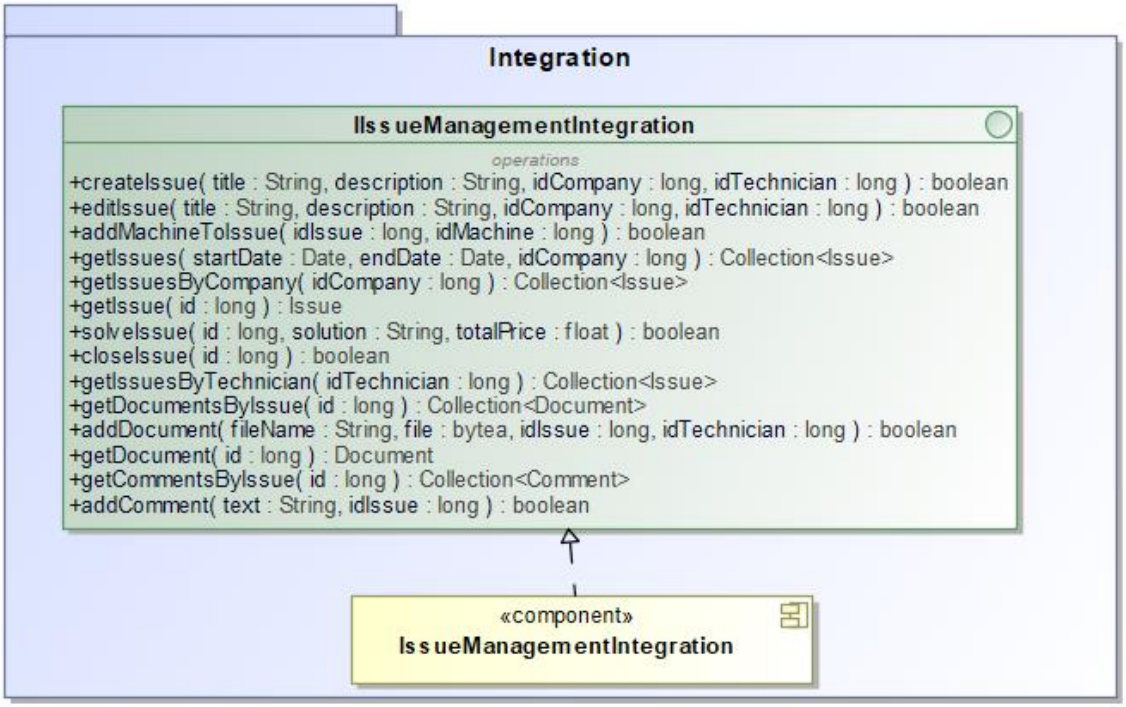


Ilustración 29: Capa de integración, componente Incidencia

3.3.3. Selección de la tecnología

Una vez realizado un primer refinamiento, en el siguiente paso se incluyen los diagramas una vez seleccionada la tecnología a utilizar. Se han tomado las siguientes decisiones:

- Se utiliza el framework Spring Boot con el que se desarrollará una API Rest. En Spring la clase que implementa el patrón "FrontController" es DispatcherServlet.
- Las vistas de la aplicación no se realizan en Java sino con React JS.
- En Spring Boot se desarrollan los controladores y en ellos se exponen los servicios web existentes en la aplicación y a los que se accede desde las vistas en react. Estos estarán anotados mediante @Controller.
- En la capa de negocio se utilizan los componentes denominados Service. Estos estarán anotados mediante @Service.
- En la capa de integración se utilizan los componentes denominados Repository. Estos estarán anotados mediante @Repository. Se implementan utilizando JPA y Spring data, framework incluído dentro de la plataforma Spring

.

Spring aplica el patrón Inyección de Dependencias que permite que sea Spring el encargado de crear las instancias de distintos componentes y proporcionarlas donde sean necesarias. Esto hace posible que se reduzca el acoplamiento entre los componentes.

3.3.3.1. Capa de presentación

Spring es un Framework que implementa el patrón "Front Controller", en concreto con la clase DispatcherServlet.

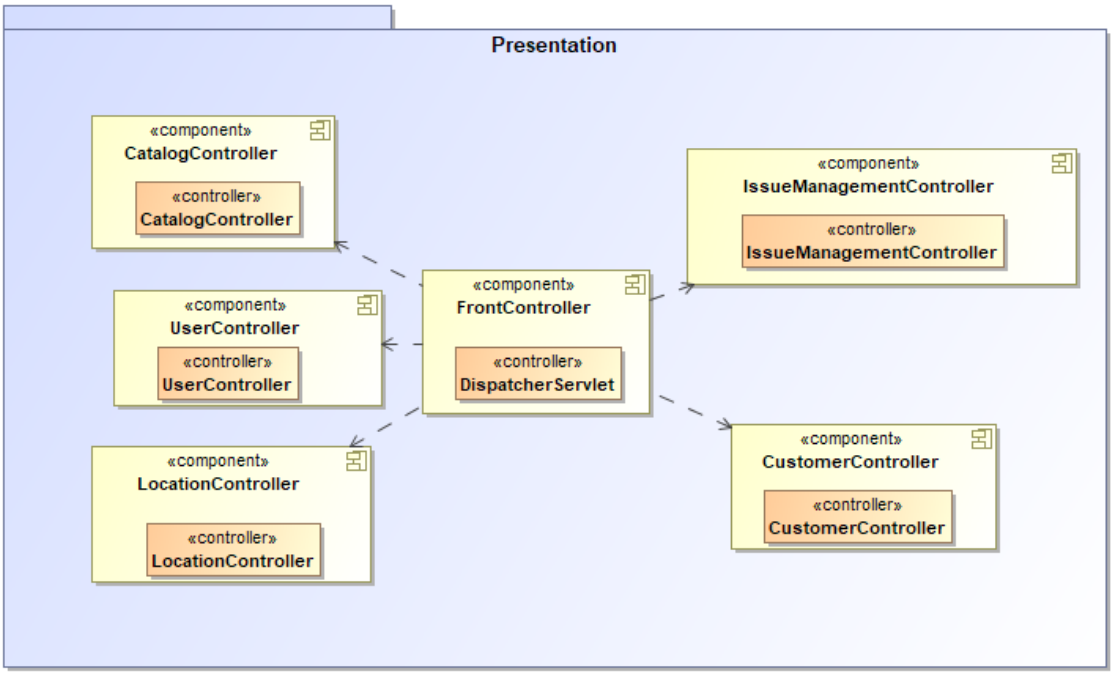


Ilustración 30: Capa de presentación, controladores

3.3.3.2. Capa de negocio

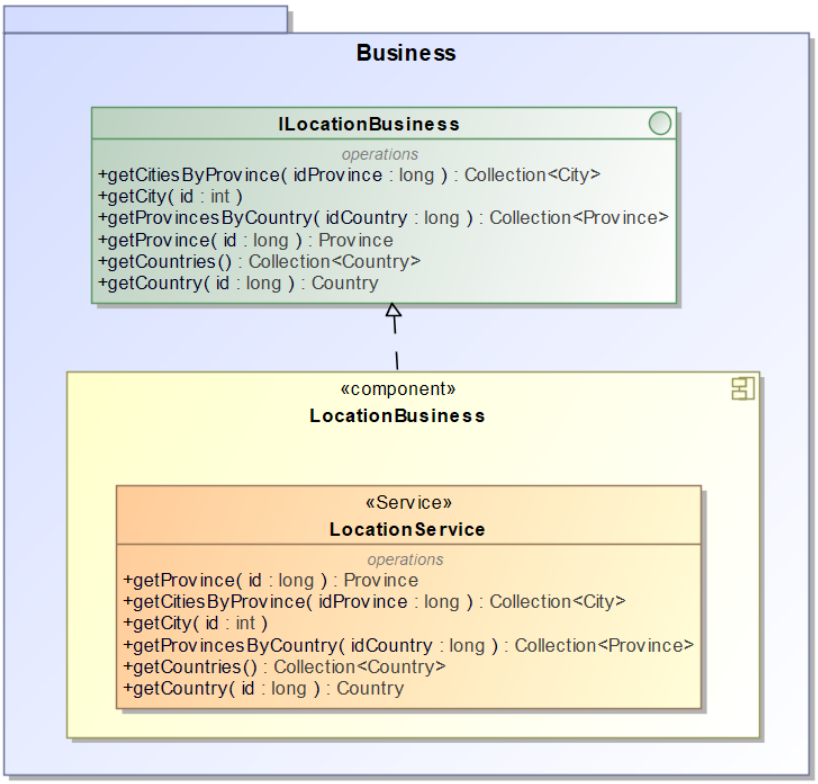


Ilustración 31: Refinamiento capa de negocio, componente Localización

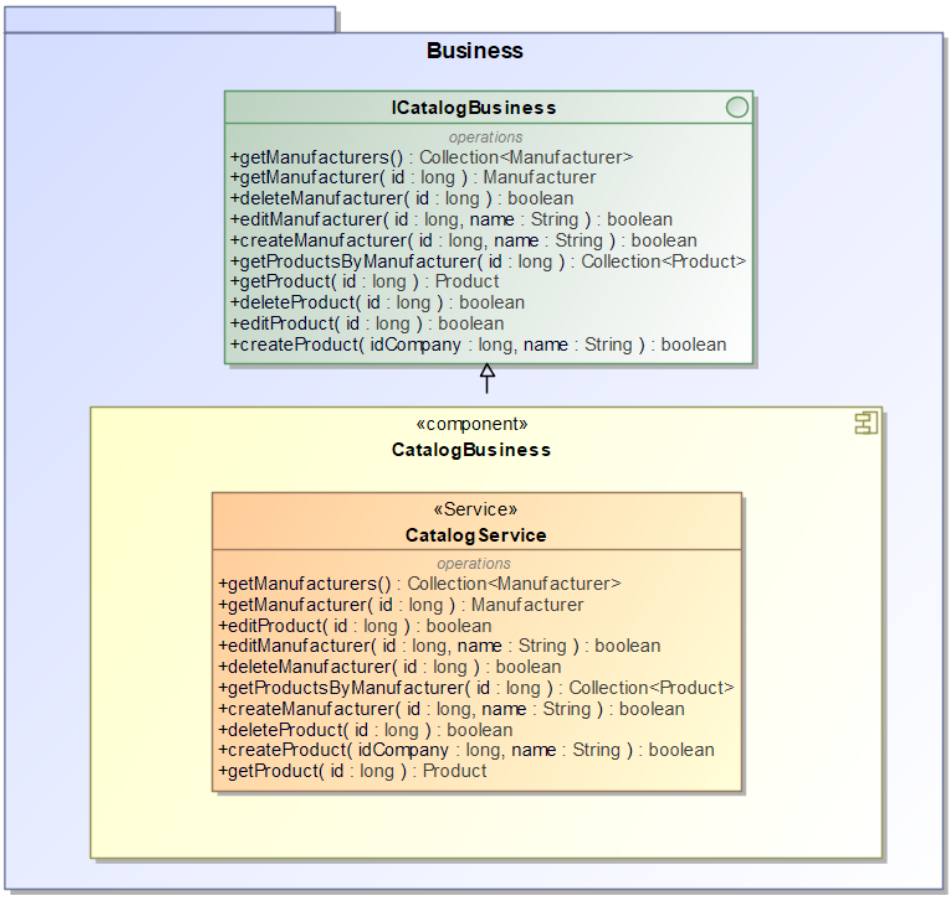


Ilustración 32: Refinamiento capa de negocio, componente Catálogo

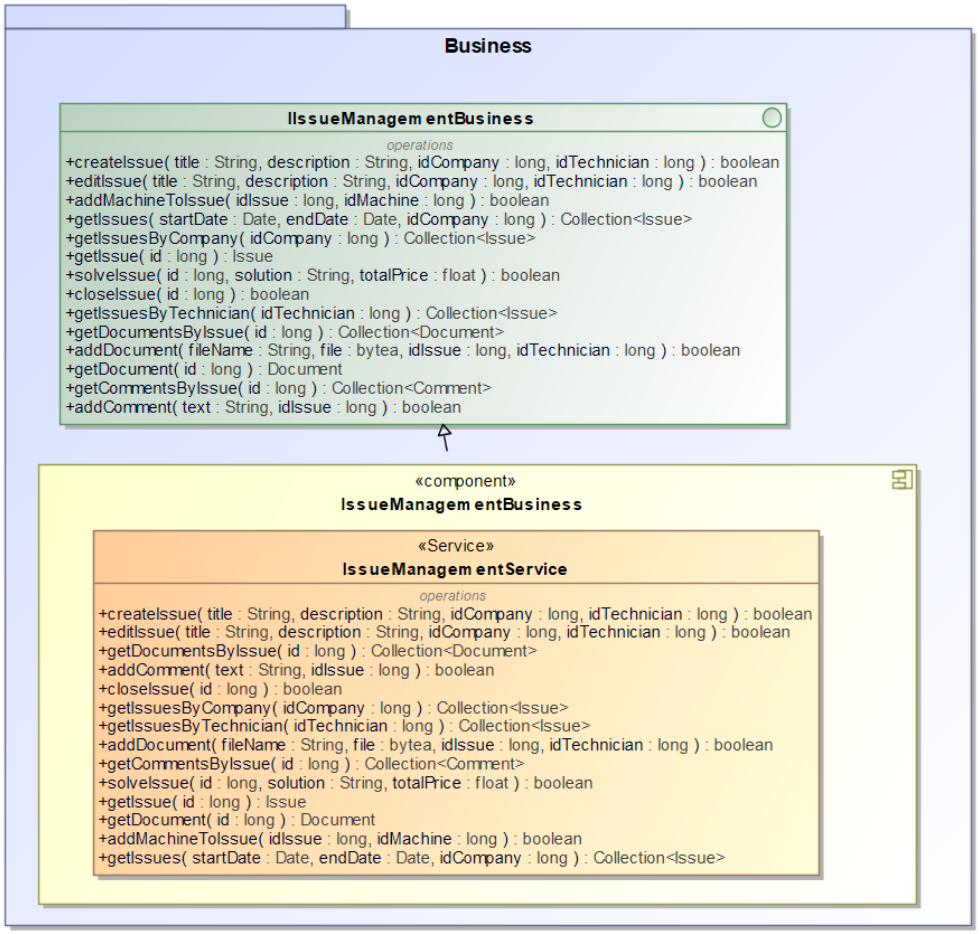


Ilustración 33: Refinamiento capa de negocio, componente Incidencia

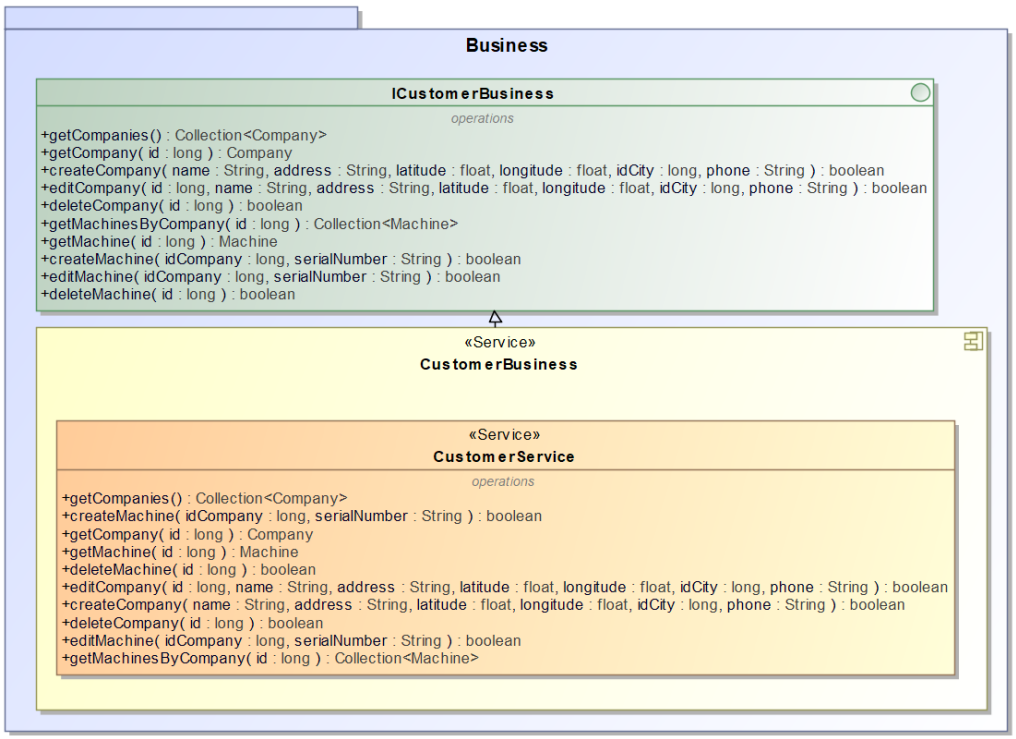


Ilustración 34: Refinamiento capa de negocio, componente Cliente

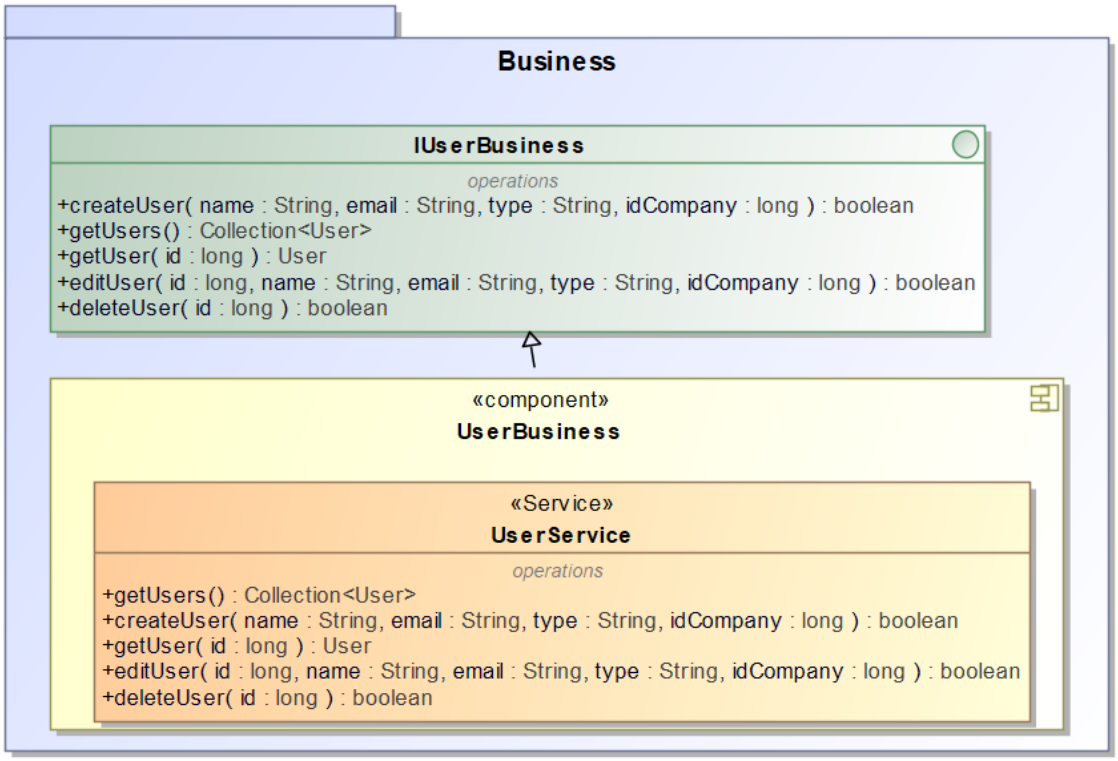


Ilustración 35: Refinamiento capa de negocio, componente Usuario

3.3.3.3. Capa de integración

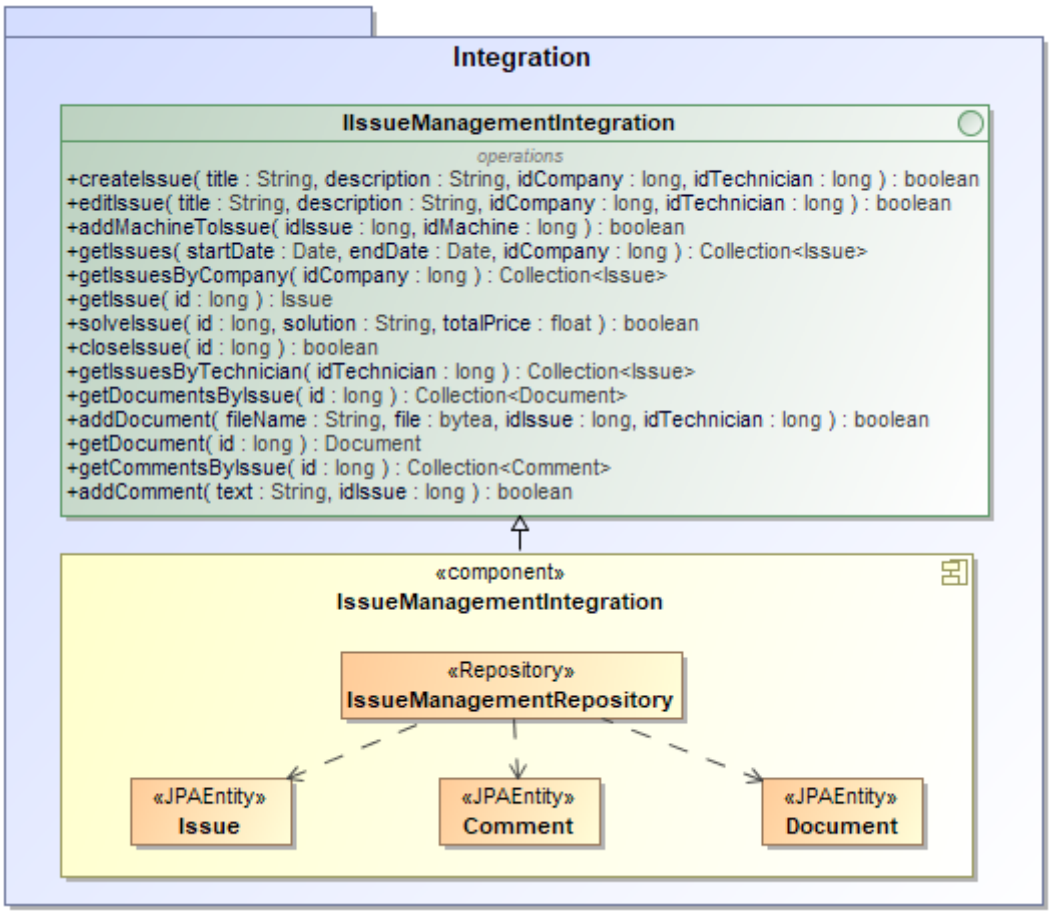


Ilustración 36: Refinamiento capa de integración, componente Incidencia

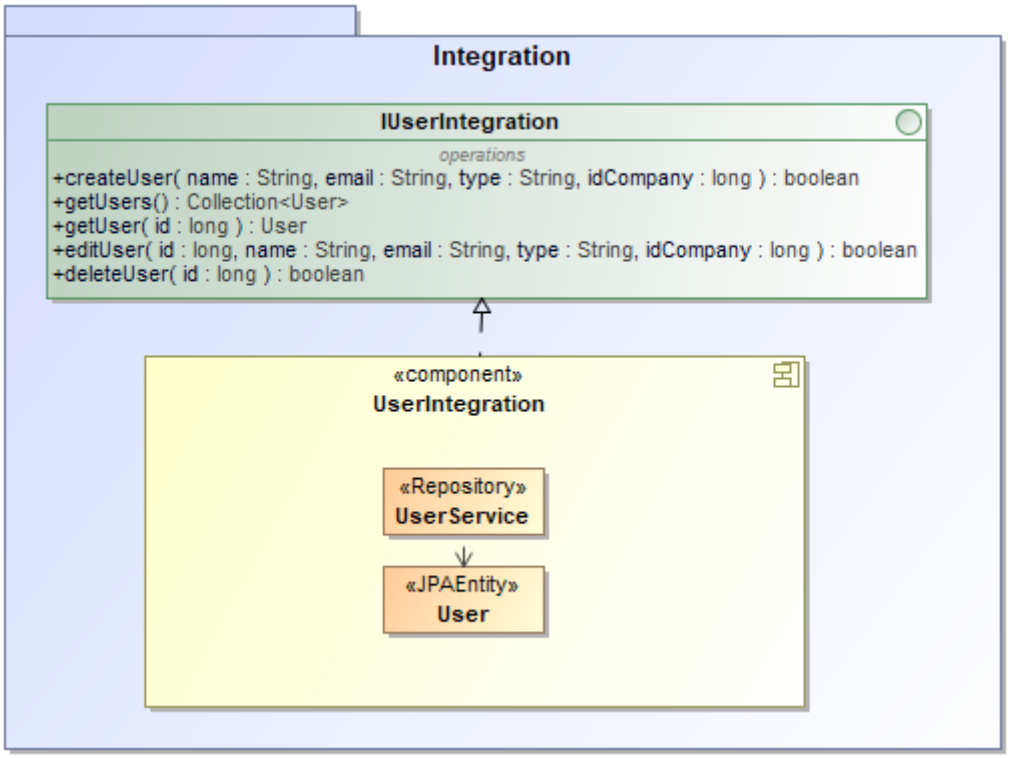


Ilustración 37: Refinamiento componente integración, componente Usuario

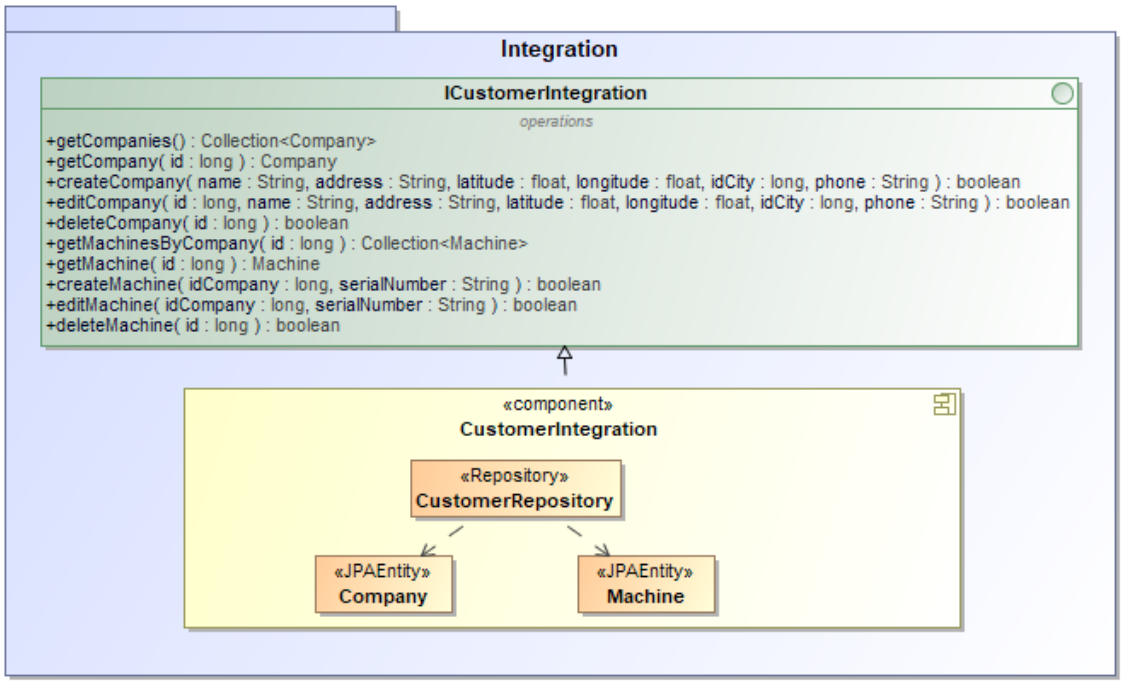


Ilustración 38: Refinamiento componente integración, componente Cliente

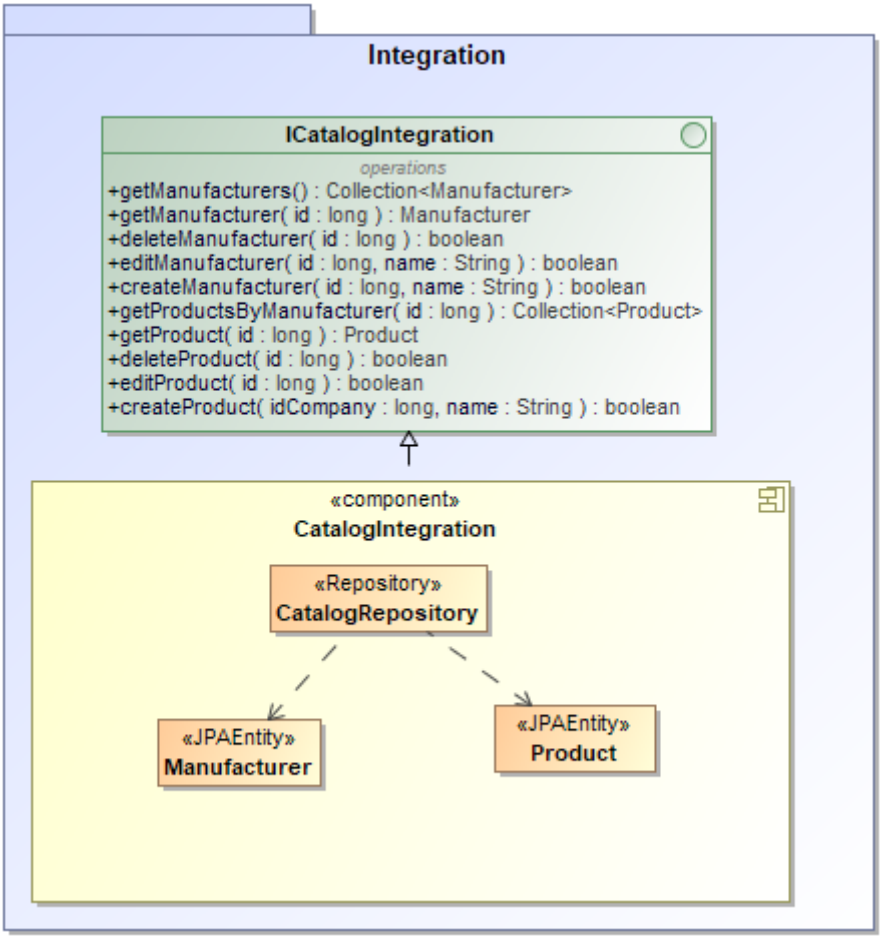


Ilustración 39: Refinamiento componente integración, componente Catálogo

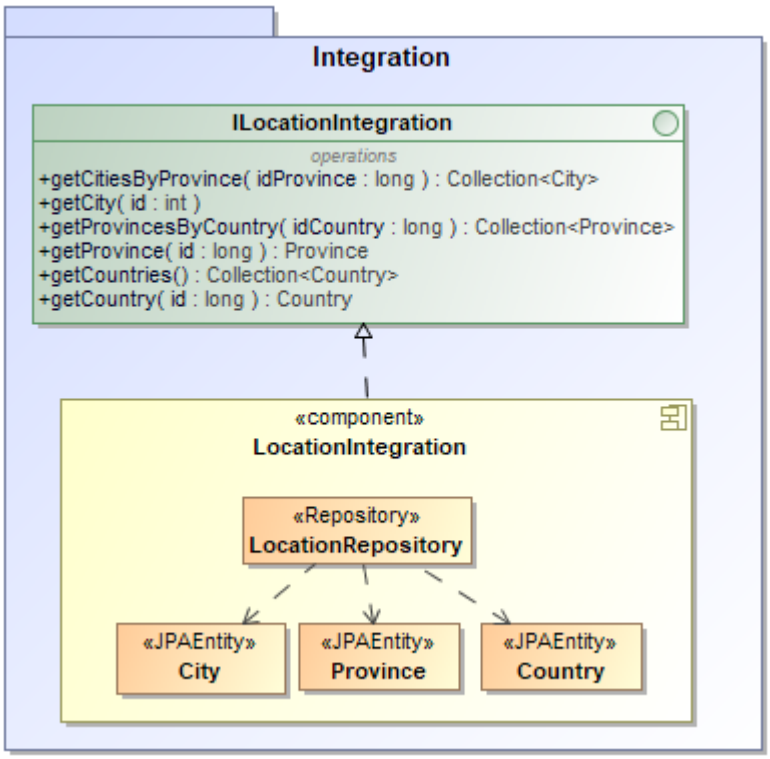


Ilustración 40: Refinamiento componente integración, componente Localización

3.3.4. Frontend

Esta parte se desarrolla utilizando React, que permite construir interfaces de usuario uniendo distintas piezas denominadas componentes. Los componentes más importantes son los siguientes:

- **App**: es el componente principal de la aplicación, contiene el componente Router que es el encargado de gestionar las diferentes rutas de la aplicación y elegir qué componente debe renderizarse en cada momento.

- **Componentes**: en ellos se define qué debe renderizarse en el navegador. Obtienen los datos a mostrar a partir de los servicios o Service.

- **Service**: son los encargados de comunicarse con el backend, realizando las peticiones HTTP a la API REST.

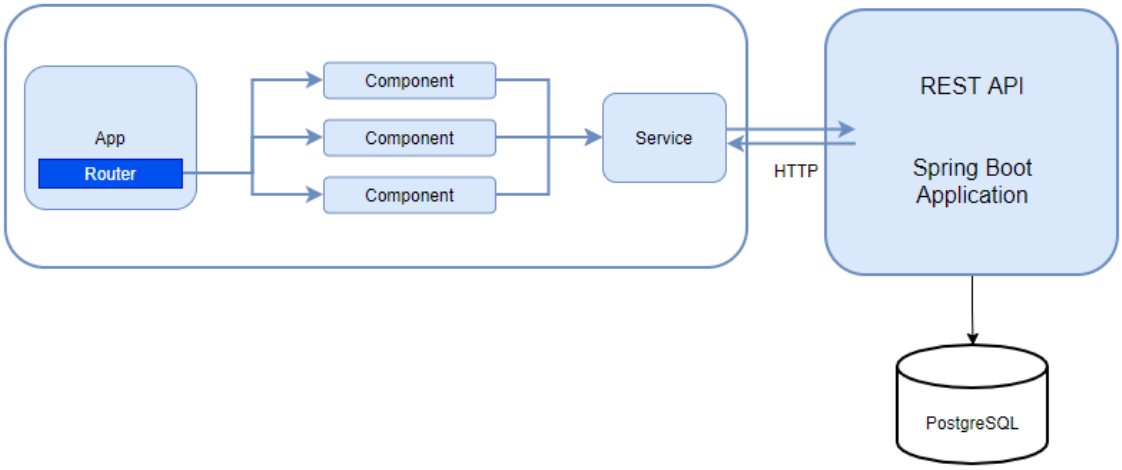


Ilustración 41: Componentes principales de una aplicación React

3.3.5. Securización de la aplicación

Para controlar el acceso a la API Rest es necesario que esta se encuentre securizada. Para ello, se utiliza el mecanismo **JWT o JSON Web Tokens**.

La principal ventaja que ofrece este sistema frente a otros es que no los tokens no tienen estado y el servidor no necesita mantener un registro, cada uno de ellos contiene todos los datos necesarios para poder comprobar que es válido. Esta característica hace que no haya dependencias entre unas peticiones y otras. El hecho de que el propio token contenga la información ahorra mucho trabajo al servidor.

A continuación, se puede ver un diagrama de secuencia con las peticiones entre cliente y servidor a la hora de acceder a los servicios REST:

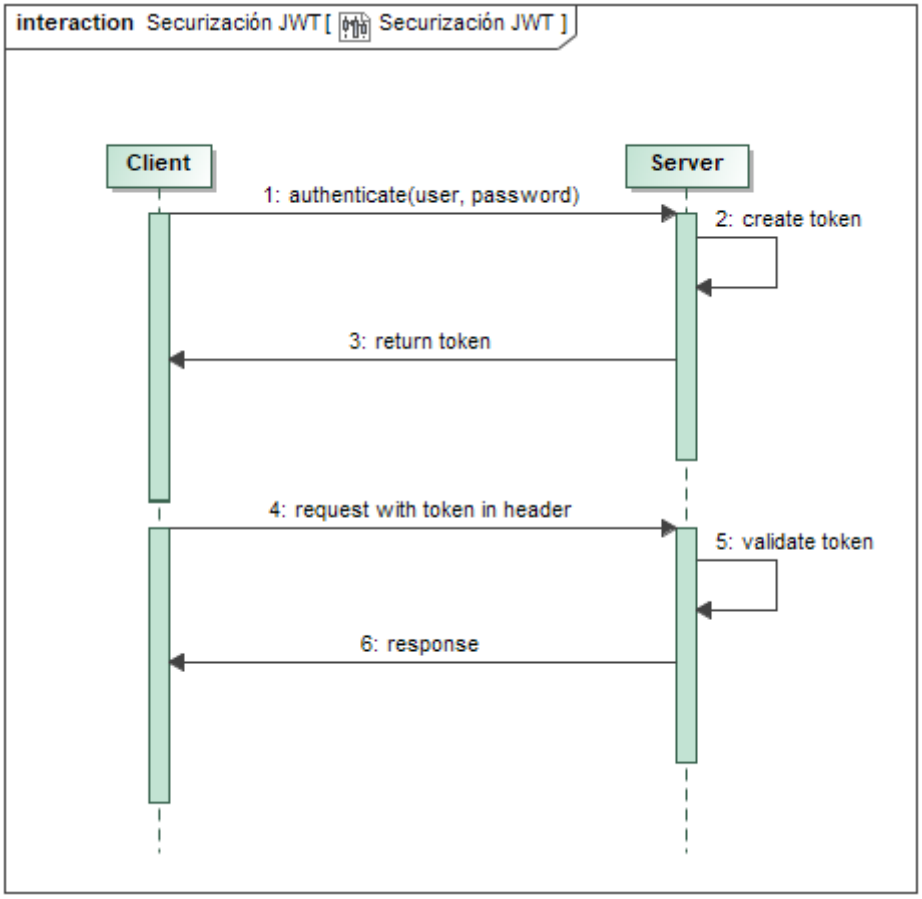


Ilustración 42: diagrama de secuencia securización JWT

4. Desarrollo de la aplicación

En este capítulo se muestran los detalles de la fase de desarrollo del proyecto.

4.1. Preparación del entorno de desarrollo

4.1.1. Preparación de la base de datos

Con el objetivo de evitar tener que realizar la instalación de PostgreSql en el equipo, en su lugar se ha utilizado Docker. Se ha creado un usuario postgres que es el que se utiliza desde la aplicación para acceder a los datos.

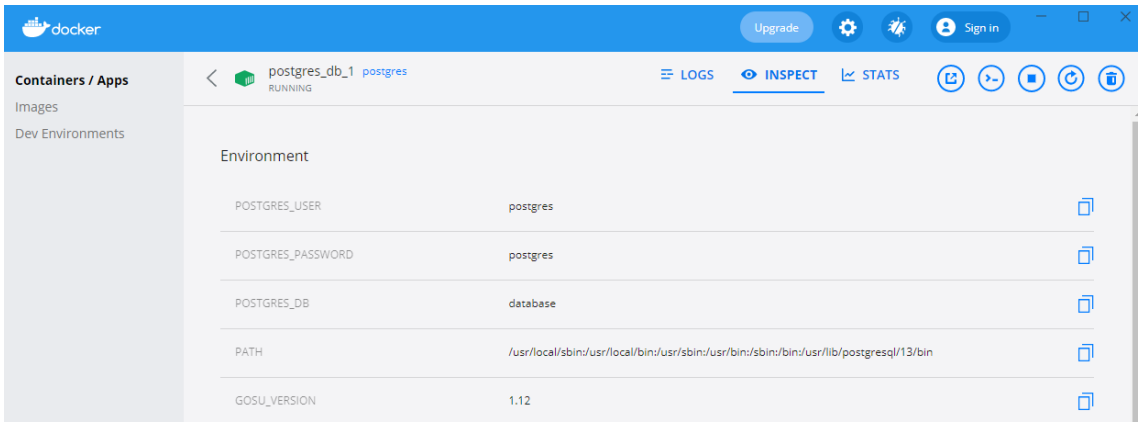


Ilustración 43: Ejecución de la base de datos Postgres mediante Docker

Dado que se va a utilizar Spring y la API de persistencia Spring Data JPA, no se generan las tablas de la base de datos mediante sql. Mediante esta funcionalidad, se crean las tablas automáticamente basándose en la configuración de las entidades realizada en Java.

4.1.2. Preparación del backend

Dado que se ha desarrollado la aplicación mediante Spring Boot, se utiliza Spring Initializr, asistente web de Spring que permite generar un proyecto de manera intuitiva indicando nombre, package a la que pertenecerán las clases y el tipo de proyecto que se quiere realizar. En función del tipo seleccionado el asistente se encarga de asignar las dependencias necesarias. Para la gestión de dependencias permite seleccionar la herramienta Maven, que es la que se ha utilizado en este caso.

La configuración utilizada ha sido la siguiente:

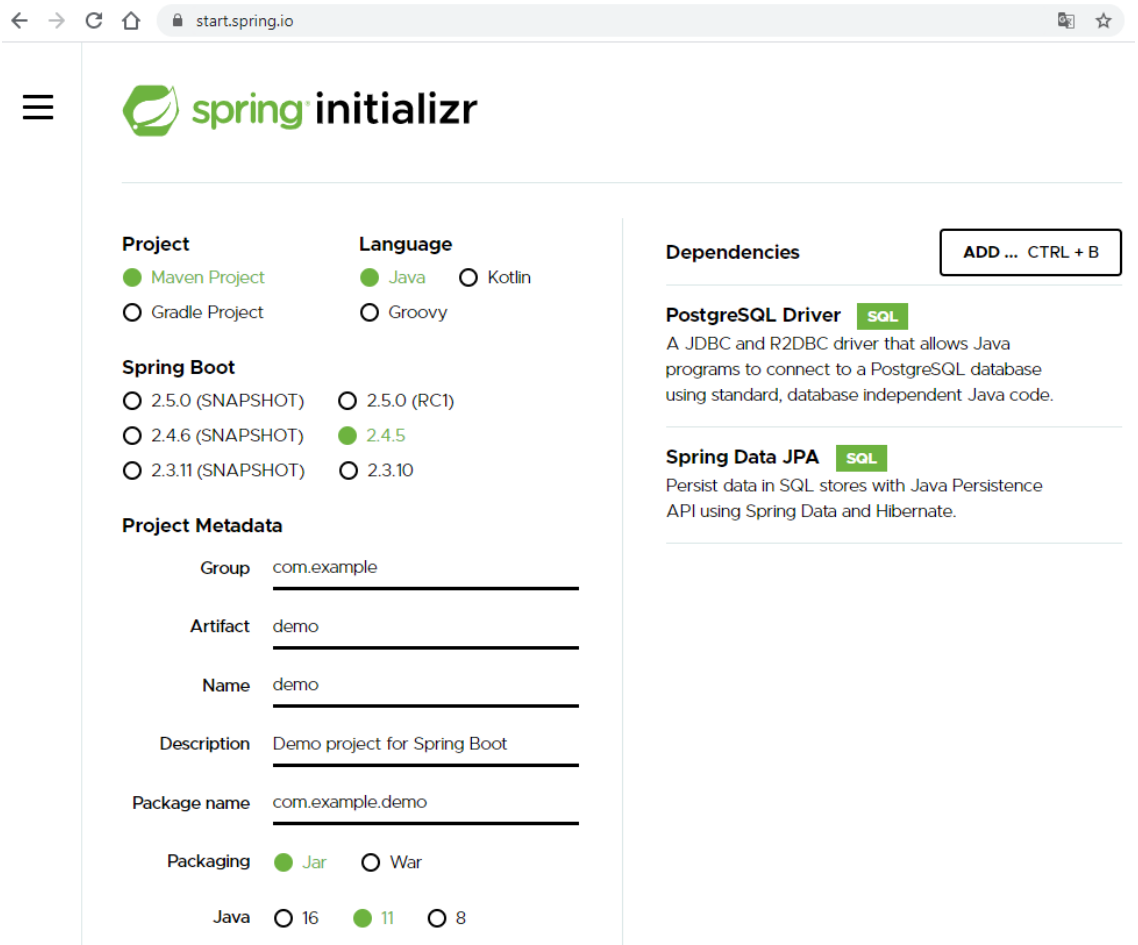


Ilustración 44: Generación de un proyecto mediante Spring Initializr

Tras finalizar la configuración con el asistente, permite descargar un fichero zip con el proyecto Spring Boot configurado y listo para comenzar el desarrollo.

Como herramienta de desarrollo, se ha utilizado el IDE Eclipse for Java Developers Version: 2020-12. Una vez instalado el programa y generado el proyecto solo ha sido necesario importarlo en el IDE indicando que se trata de un proyecto Maven.

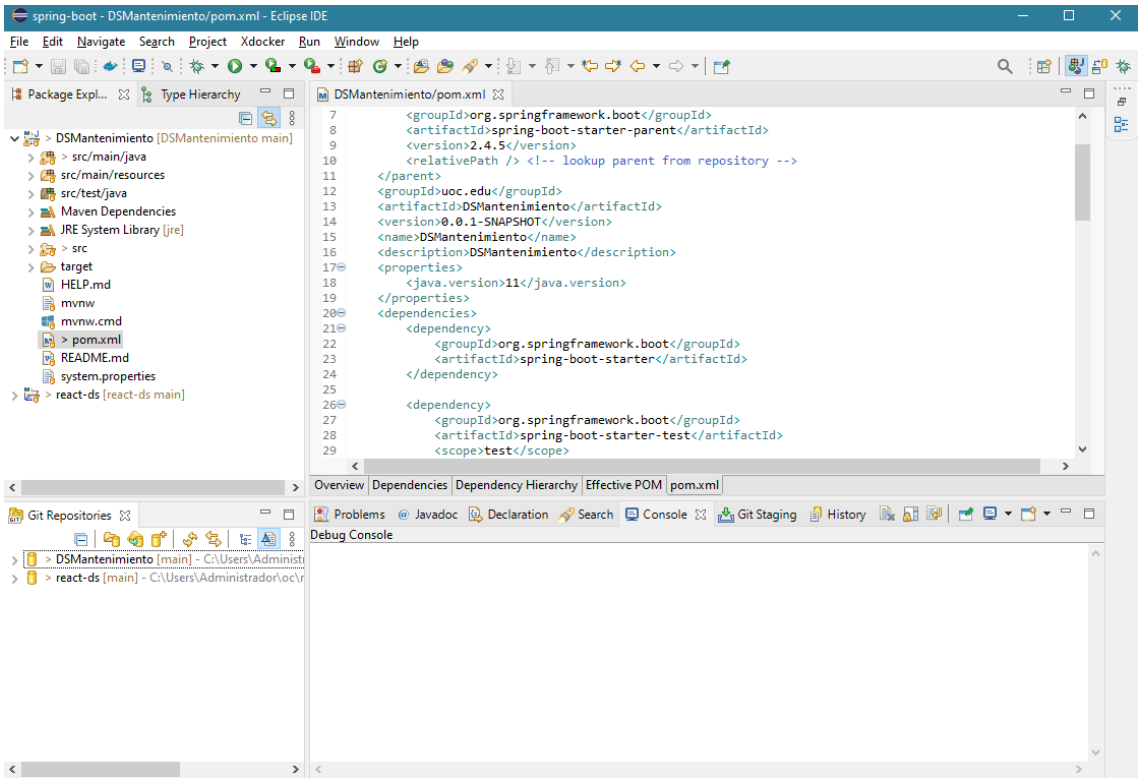


Ilustración 45: Eclipse IDE con el proyecto importado

El proyecto generado con Spring Initializr cuenta con un servidor integrado, por lo que no ha sido necesario instalar ningún servidor.

4.1.3. Preparación del frontend

Para la creación de una aplicación react desde 0 se ha utilizado create-react-app, node.js y npm. Mediante esta herramienta se genera la estructura de un proyecto react ya configurado.

Para el desarrollo de esta parte se ha utilizado el editor de código Atom.

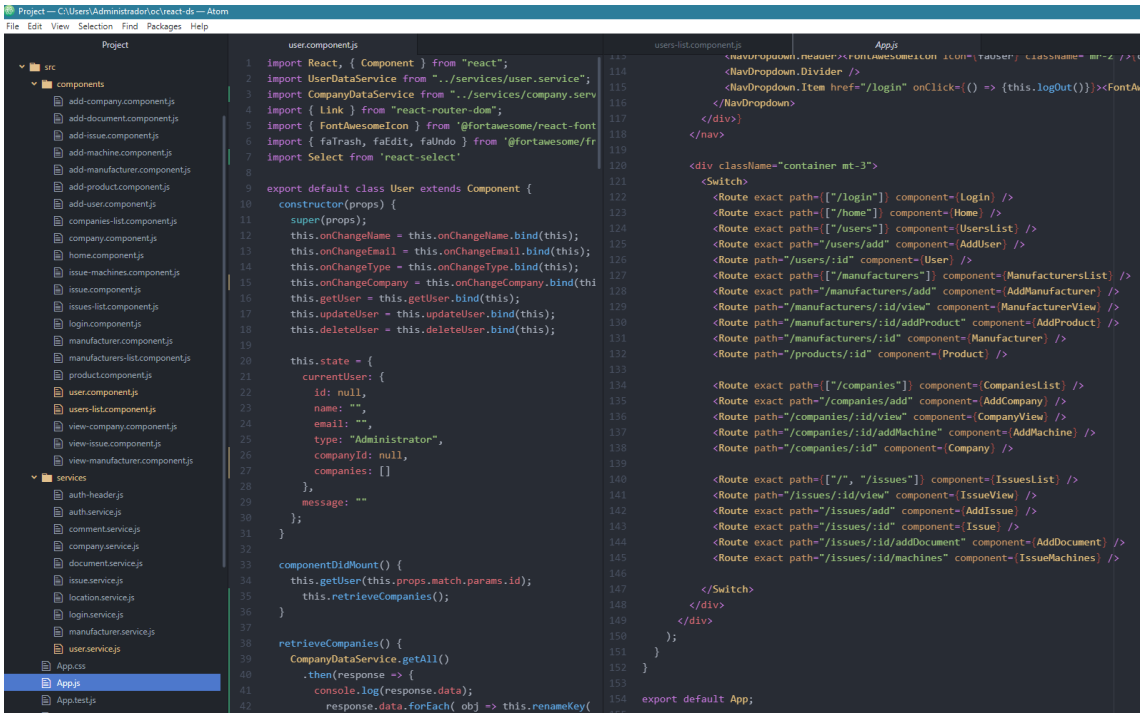


Ilustración 46: proyecto frontend en el editor Atom

Para arrancar la aplicación basta con utilizar npm desde línea de comandos.

4.2. Construcción de la aplicación

Tras la preparación del entorno de desarrollo se ha trabajado en la construcción de la herramienta diseñada.

Se ha seguido la planificación planteada en un inicio, por lo que los pasos seguidos han sido los siguientes:

- Desarrollo de la API REST mediante Spring Boot
- Securización de la API mediante JWT
- Desarrollo del frontend

4.2.1. Construcción de API REST

Se ha incluido en el proyecto la dependencia Swagger que permite generar documentación de manera automática a partir del código. Una vez arrancado el proyecto, puede encontrarse en la siguiente url:
http://localhost:8080/swagger-ui/index.html?configUrl=/api-docs/swagger-config

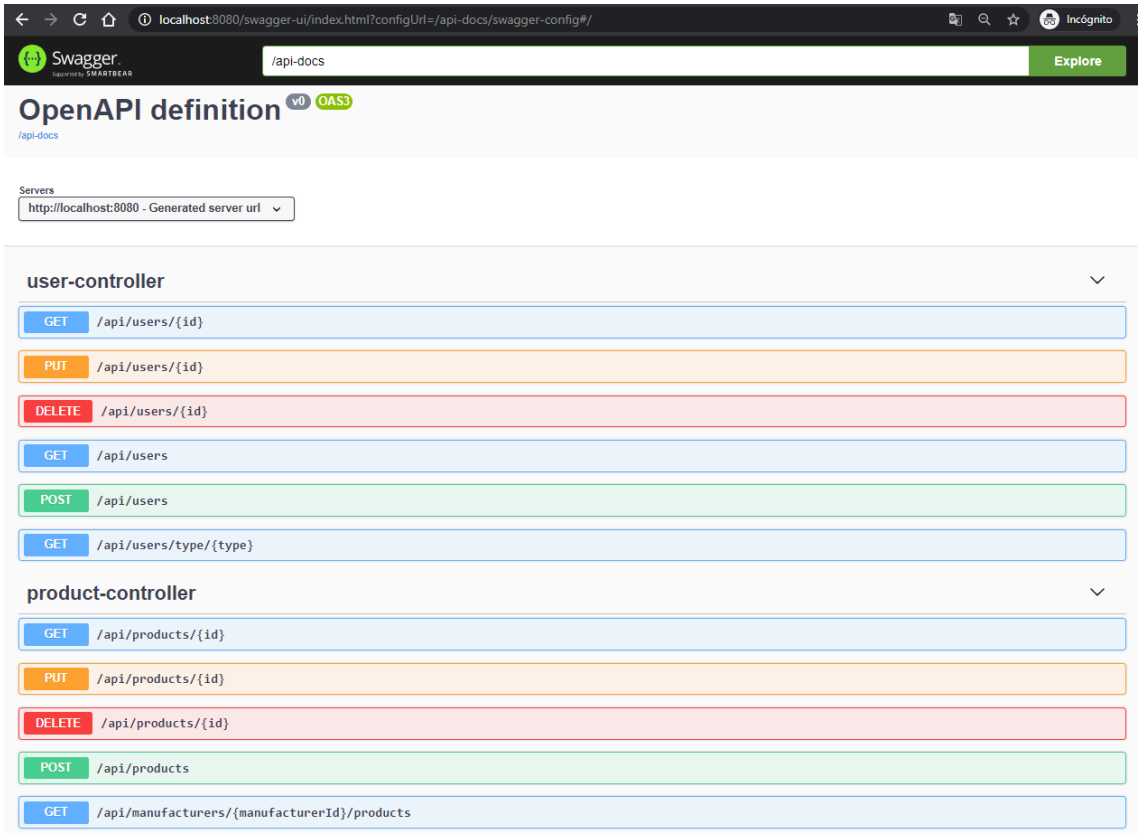


Ilustración 47: Documentación de la API REST

Para cada una de las peticiones que la API soporta se genera un apartado con las características de la petición, los parametros que espera, un ejemplo de cuerpo de la petición, respuestas, etc. Además, da la opción de probar a ejecutar peticiones a la API desde la web.

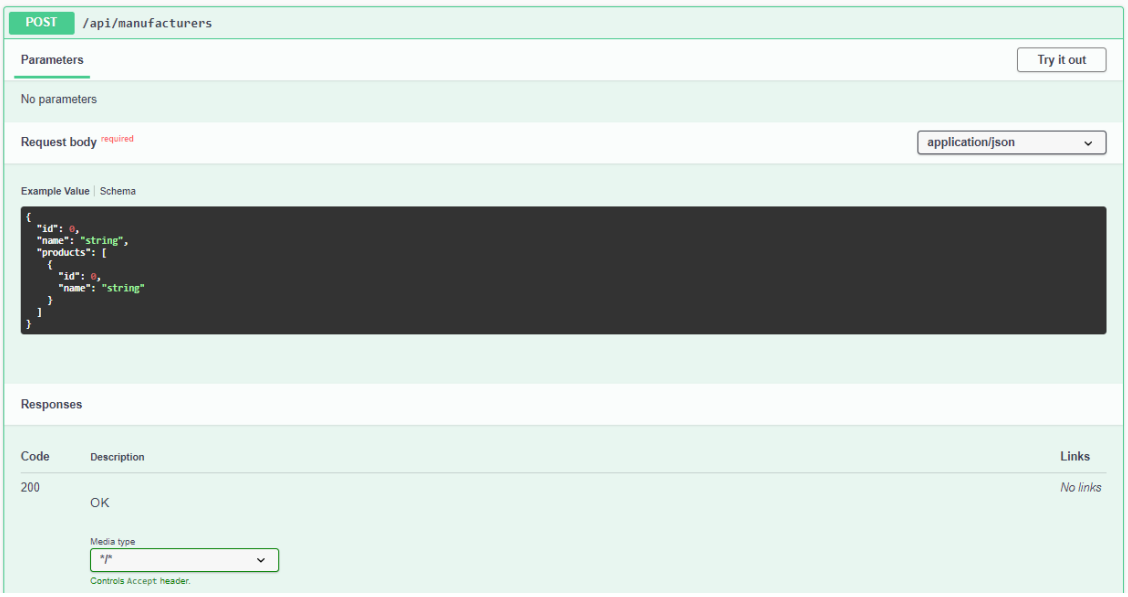


Ilustración 48: Información proporcionada sobre la petición para crear fabricante

4.2.2. Autentificación en la API REST con JWT

Una vez desarrollados los servicios REST, se ha añadido un método de autentificación necesario para consumir los servicios. El método utilizado ha sido la autentificación basada en tokens.

Para permitir la autenticación, se crea un servicio de tipo POST "/api/auth/signin" al que se llama al hacer login en la aplicación. A este servicio se le envía email y password para obtener el token de acceso.

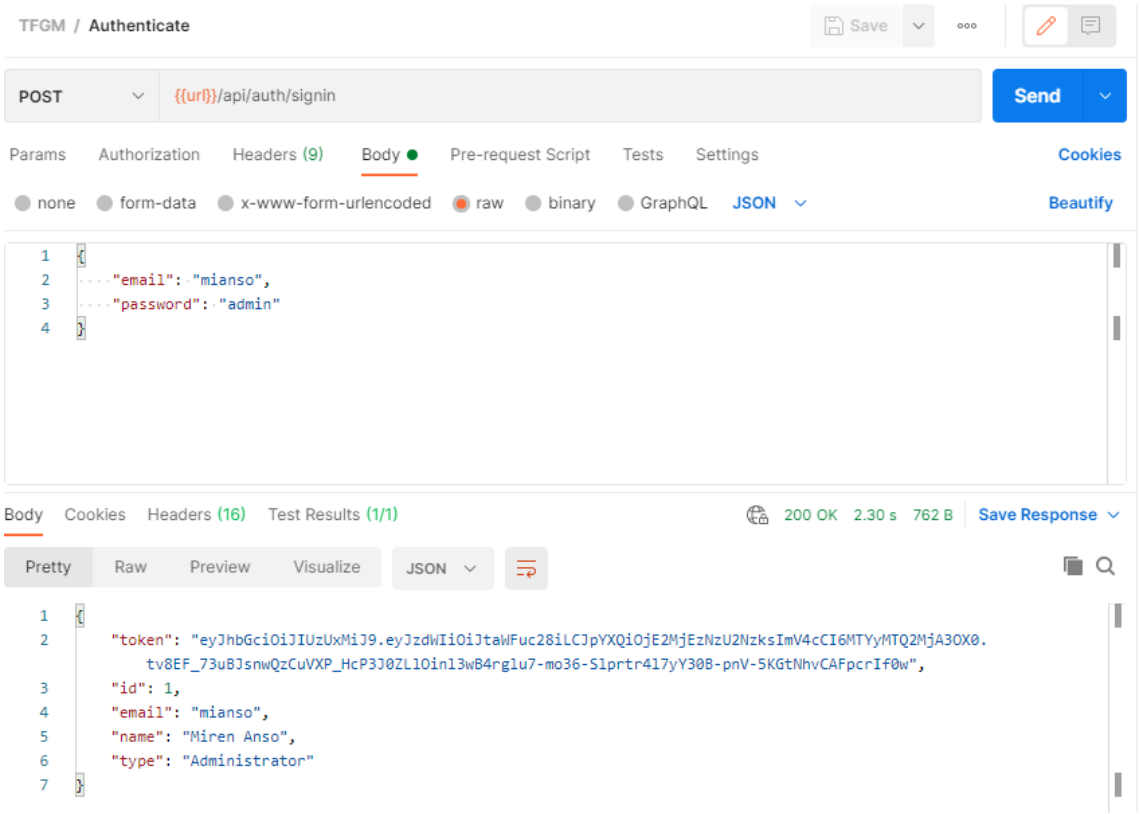


Ilustración 49: Prueba de una petición para obtener un JWT

Una vez que el usuario se haya autenticado en la plataforma, se almacena su información junto con su token de acceso en la Local storage del navegador. Este token se utilizará en cada petición que se realice a la API.

Cuando el usuario cierra sesión en la aplicación, se elimina su información del navegador.

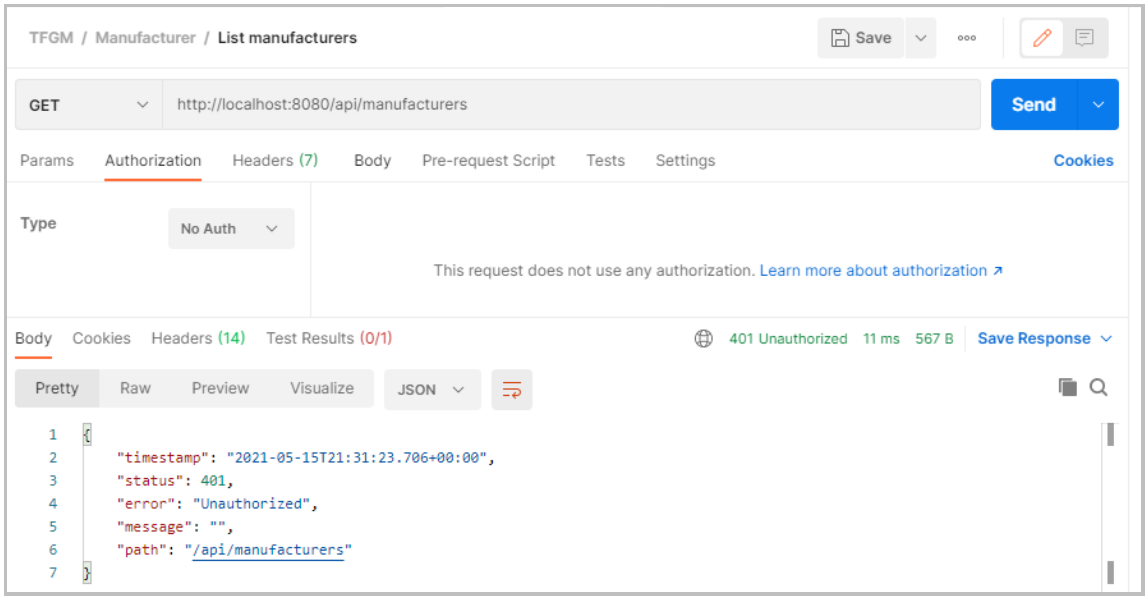


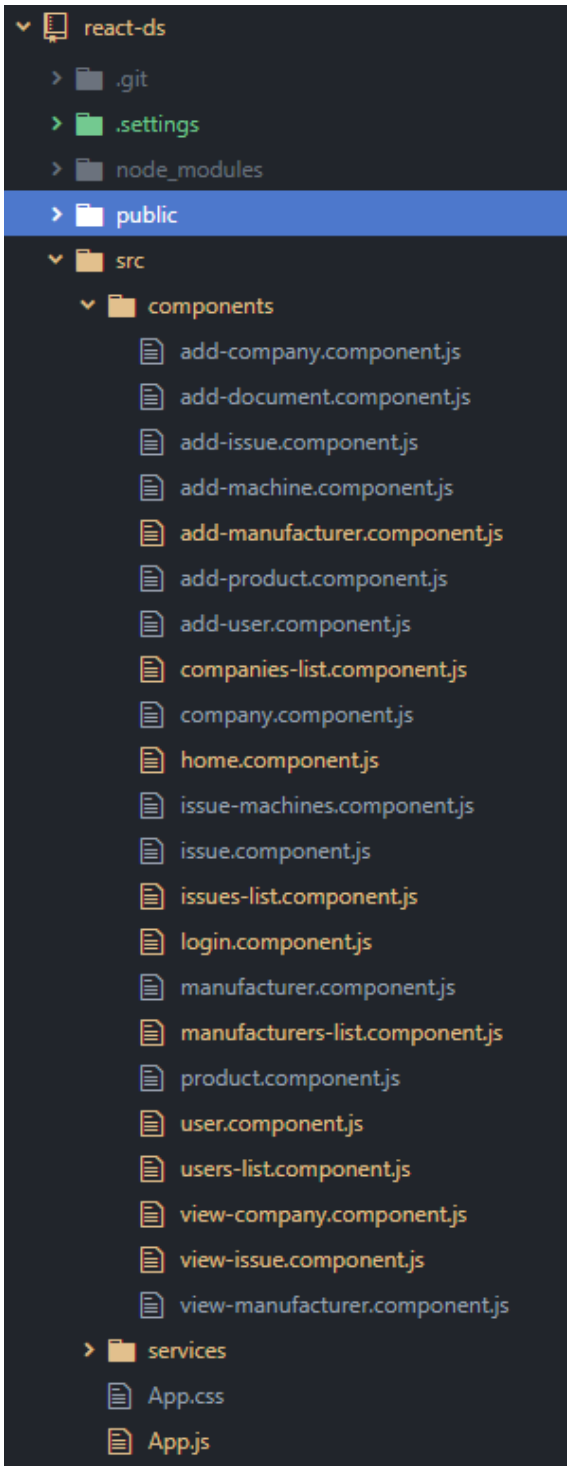
Ilustración 50: Ejemplo de petición no autorizada a la API REST

4.2.3. Construcción del frontend

Como ya se ha dicho anteriormente, la interfaz de la aplicación se ha desarrollado con React. Para facilitar la construcción de las pantallas se ha utilizado el framework Bootstrap, ya que permite crear interfaces intuitivas de forma rápida.

Las carpetas más importantes en la estructura del proyecto son las siguientes:

- **node_modules**: contiene las dependencias necesarias para el funcionamiento del proyecto
- **src**: es la carpeta que contiene el núcleo de la aplicación
    - **components**: contiene los denominados componentes que se corresponden con las distintas vistas de la aplicación.
    - **services**: contiene los servicios que son los encargados de comunicarse con el backend y obtener y enviar los datos.



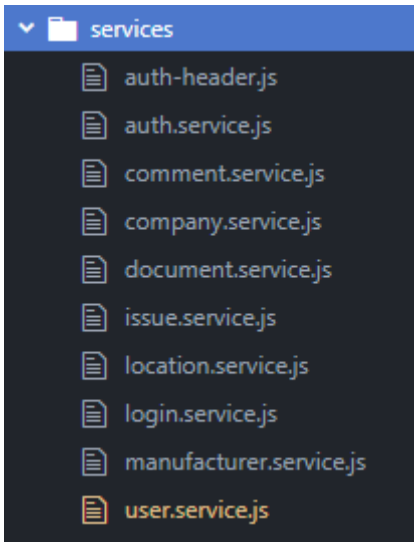


Ilustración 51: Estructura del proyecto React

4.2.4. Herramienta de control de versiones

Para ambos proyectos se ha utilizado Github como repositorio de código. Ambos proyectos son públicos y se encuentran en las siguientes ubicaciones:

- Backend: https://github.com/mianso-uoc/dsm_backend
- Frontend: https://github.com/mianso-uoc/dsm_frontend

4.3. Producto final

En este apartado se pueden ver las principales pantallas de la herramienta desarrollada. Veremos los apartados a los que tienen acceso los usuarios en función de su perfil.

4.3.3. Perfil Administrador

Los usuarios Administradores acceden a una pantalla de inicio con opciones para gestionar Usuarios, Empresas y Fabricantes.

Ilustración 52: Acceso para usuarios administradores

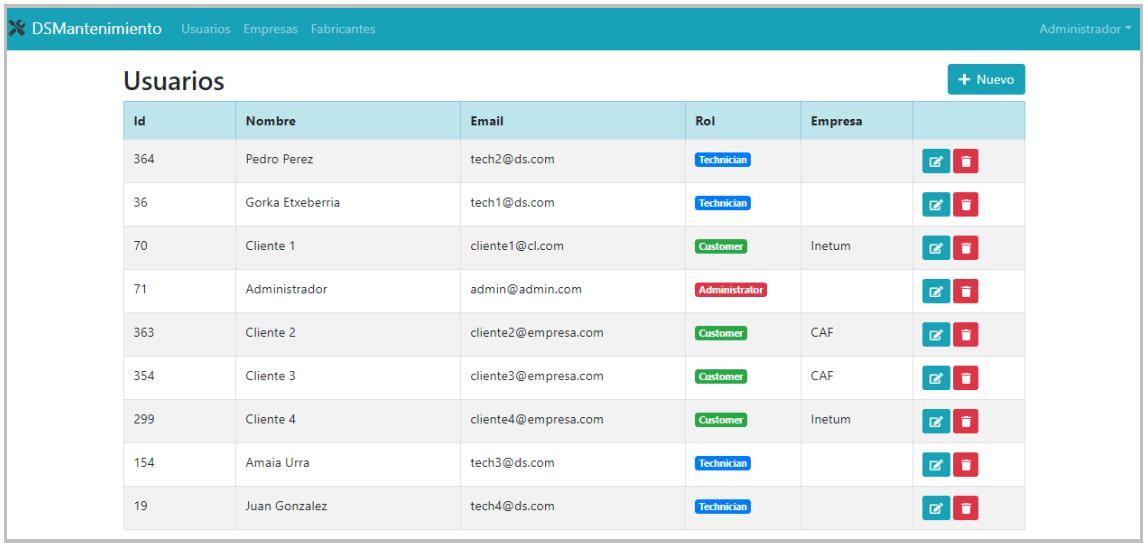


Ilustración 53: Gestión de usuarios

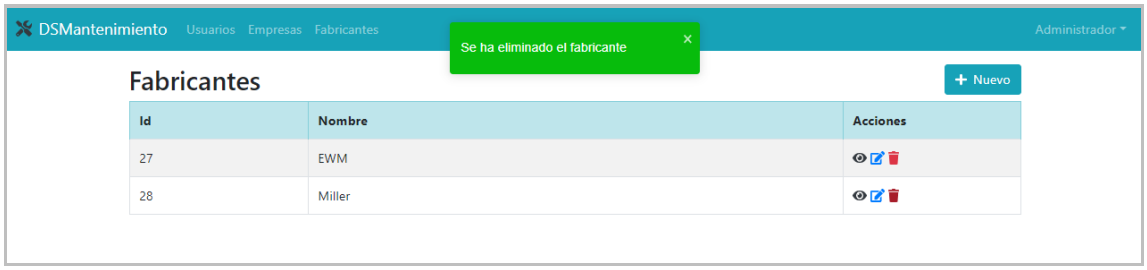


Ilustración 54: Gestión de fabricantes

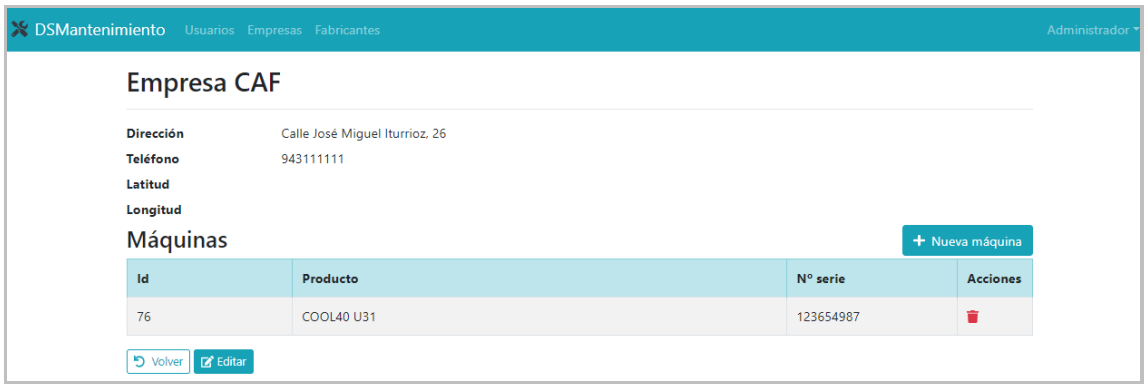


Ilustración 55: Detalle de una empresa

Ilustración 56: Detalle de un fabricante

4.3.4. Perfil Técnico

Los usuarios Técnicos acceden a la pantalla de inicio donde pueden ver el listado de incidencias que tienen asignadas.

Ilustración 57: Listado de Mis incidencias

Desde el apartado de Incidencias tienen opciones para filtrar por fechas y empresa.

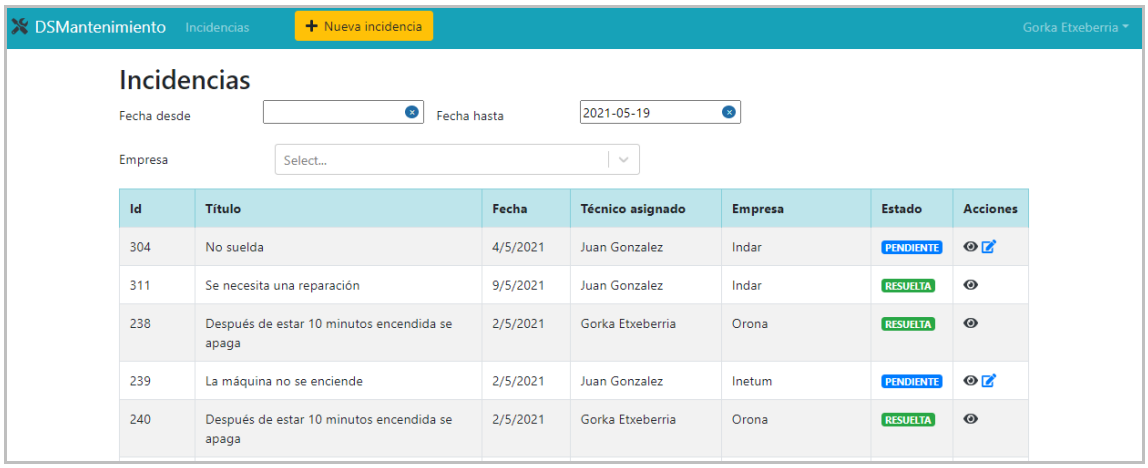


Ilustración 58: Listado de incidencias y opciones de filtro

Desde el detalle de una incidencia un técnico puede ver los documentos y comentarios de la misma, además de poder añadir nuevos. Además, puede añadir a la incidencia las máquinas del cliente que estén afectadas.

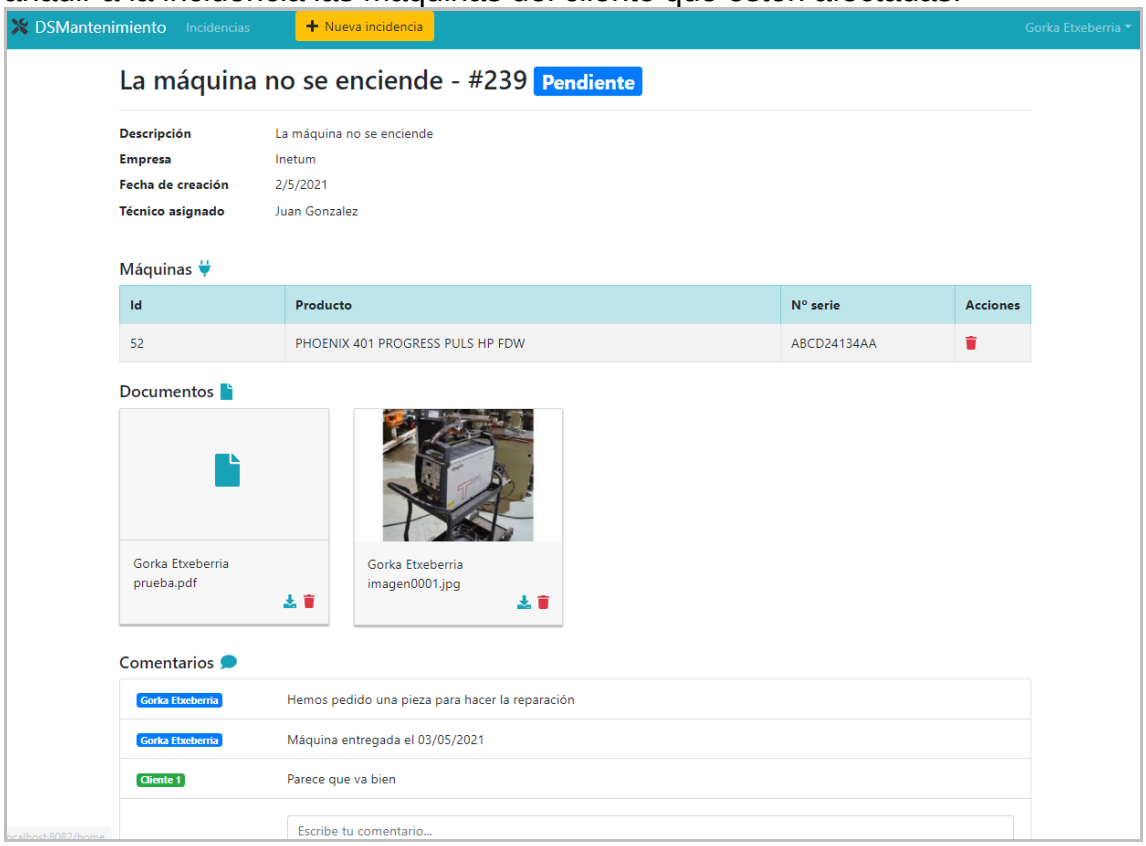


Ilustración 59: Detalle de una incidencia

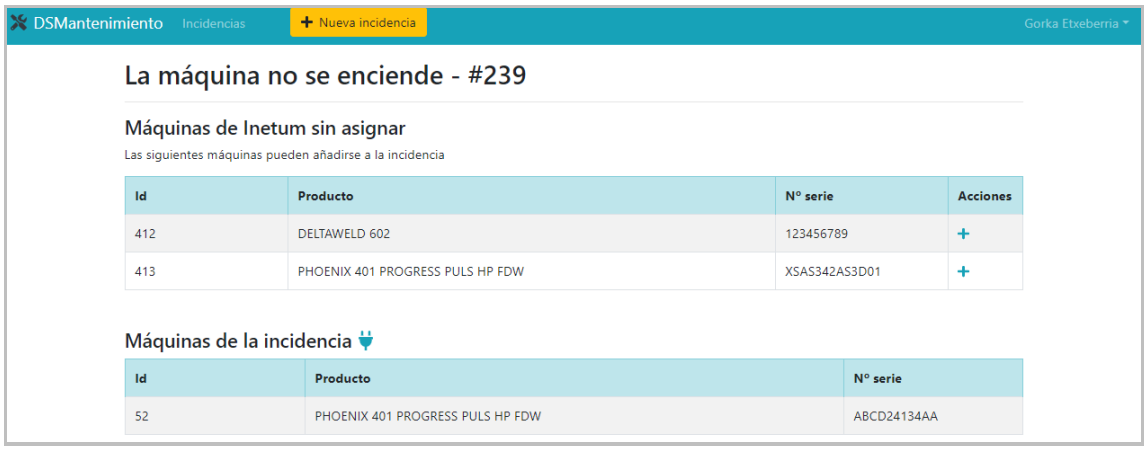


Ilustración 60: Opción de añadir máquinas a una incidencia

Por último, un técnico tiene la opción de cerrar o resolver una incidencia pendiente:

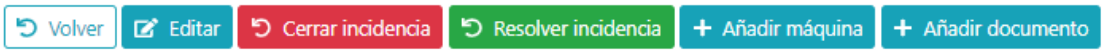


Ilustración 61: Acciones posibles en una incidencia Pendiente

4.3.5. Perfil Cliente

Los usuarios con perfil cliente tienen acceso al listado de incidencias asignadas a su empresa. Además, tienen la opción del filtrar los resultados por fechas.

Ilustración 62: Listado de incidencias de mi empresa

Pueden ver el detalle de una incidencia y añadir comentarios.

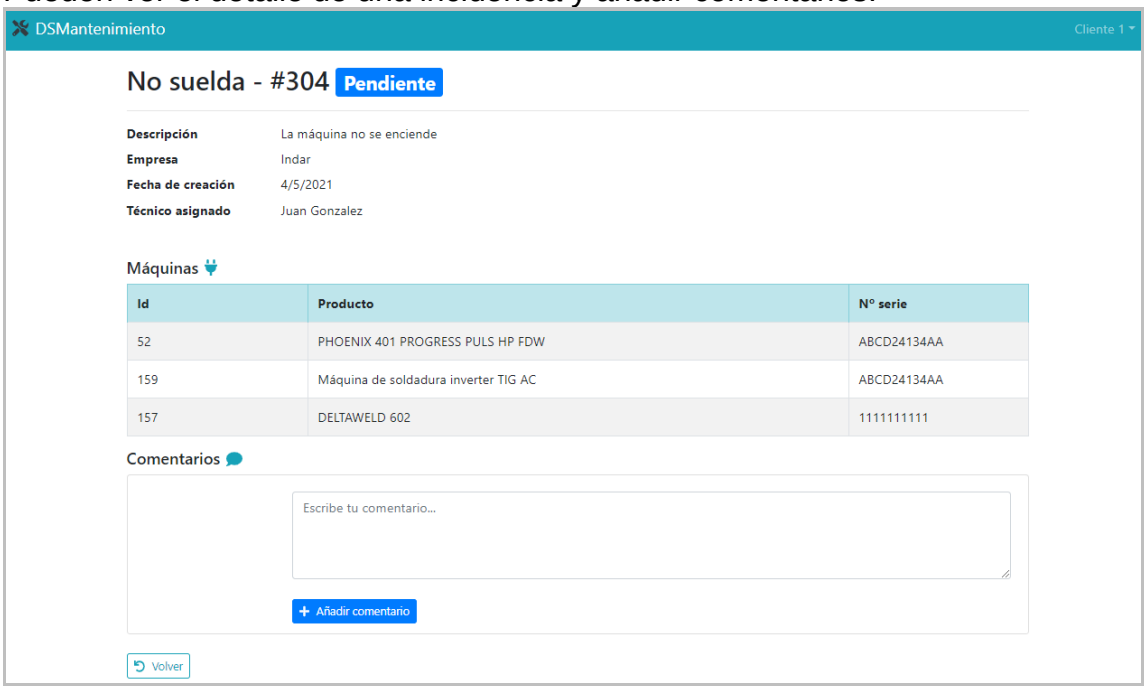


Ilustración 63: Detalle de una incidencia y opción de añadir comentarios

5. Pruebas

5.1. Pruebas de API REST con postman

Postman permite automatizar pruebas para comprobar que la API funciona correctamente. Para ello, se ha creado una colección de peticiones y dentro se han configurado todas las peticiones GET, POST, PUT y DELETE que cubren el funcionamiento de la API desarrollada mediante Spring Boot.

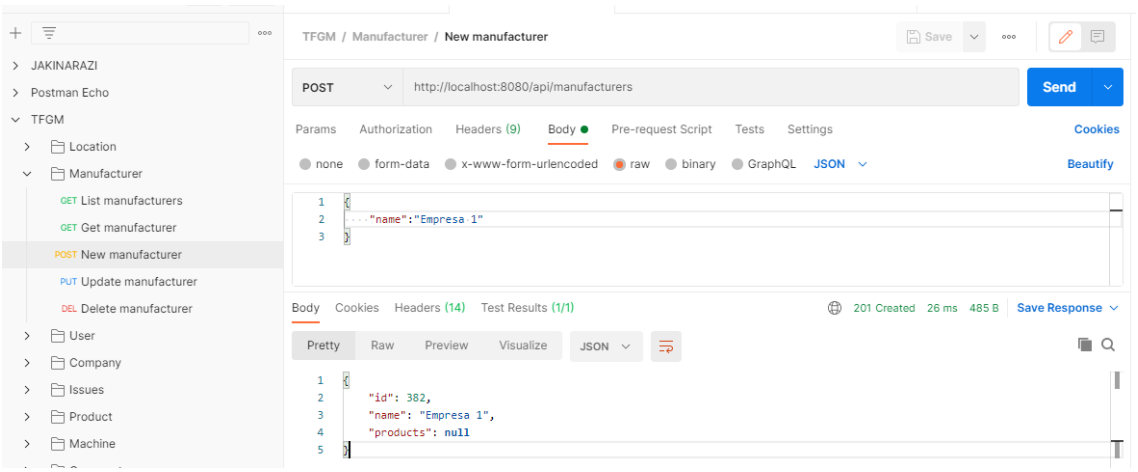


Ilustración 64: Petición en Postman para crear un nuevo fabricante

Colección de pruebas ejecutadas:

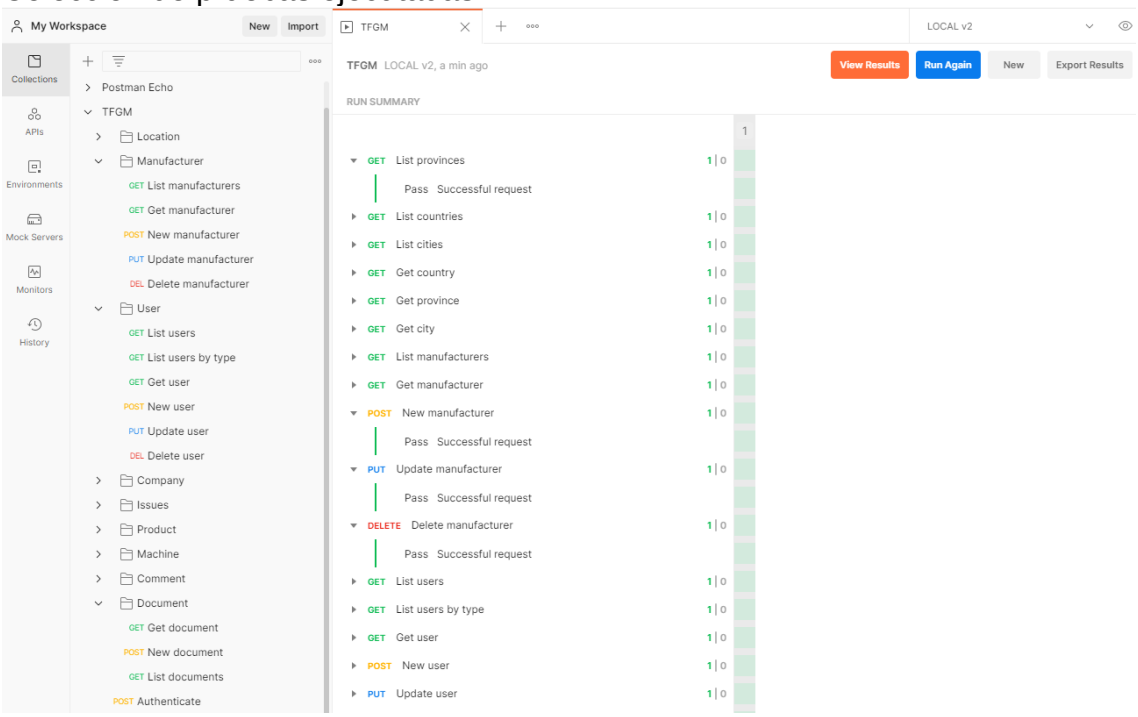


Ilustración 65: Resultado de la ejecución de tests en Postman

5.2. Pruebas frontend

A continuación, se muestra un resumen de las pruebas realizadas, divididas por componentes para las distintas funcionalidades de la herramienta.

5.2.1. Componente Catálogo

Nº	Funcionalidad	Resultado
CA01	Crear un fabricante	OK
CA02	Listar fabricantes	OK
CA03	Editar un fabricante	OK
CA04	Eliminar un fabricante	OK
CA05	Listar los productos de un fabricante	OK
CA06	Crear un producto para un fabricante	OK
CA07	Eliminar un producto	OK
CA08	Editar un producto	KO, no lo edita

Pruebas correctas: 87,5% (7/8)

5.2.2. Componente Incidencias

Nº	Funcionalidad	Resultado
IN01	Crear una incidencia	OK
IN02	Listar incidencias	OK
IN03	Editar una incidencia	OK
IN04	Cerrar una incidencia	OK
IN05	Resolver una incidencia	OK
IN06	Añadir una máquina a la incidencia	OK
IN07	Ver mis incidencias como técnico	OK
IN08	Ver mis incidencias como empresa	OK
IN09	Ver detalle de una incidencia como cliente	KO
IN10	Filtrar incidencias entre fechas	OK
IN11	Filtrar incidencias por cliente	OK
IN12	Añadir documento a una incidencia	OK
IN13	Añadir un comentario a una incidencia como cliente	OK
IN14	Añadir un comentario a una incidencia como técnico	OK

Pruebas correctas: 92,86% (13/14)

5.2.3. Componente Cliente

Nº	Funcionalidad	Resultado
CA01	Crear una empresa	OK
CA02	Listar empresas	OK
CA03	Editar una empresa	OK
CA04	Eliminar una empresa	OK
CA05	Listar las máquinas de una empresa	OK
CA06	Crear una máquina para una empresa	OK
CA07	Eliminar una máquina	OK
CA08	Editar una máquina	OK

Pruebas correctas: 100% (8/8)

5.2.4. Componente Usuario

Nº	Funcionalidad	Resultado
US01	Acceso a la plataforma	OK
US02	Cerrar sesión	OK
US03	Crear un usuario administrador	OK
US04	Crear un usuario técnico	OK
US05	Crear un usuario cliente	OK
US06	Listar usuarios existentes	OK
US07	Editar un usuario	KO, edita bien clientes. Error al editar admin y técnico
US08	Eliminar un usuario	OK

Pruebas correctas: 87,5% (7/8)

5.2.5. Componente Localización

Nº	Funcionalidad	Resultado
LO01	Listar países	OK
LO02	Listar provincias de un país	OK
LO03	Listar municipios de una provincia	OK

Pruebas correctas: 100% (8/8)

Resultado para el total de pruebas: 92,68% (38/41)

6. Despliegue

Se han desplegado ambas aplicaciones en Heroku, una plataforma en la nube que permite alojar aplicaciones sin que tengamos que preocuparnos por la infraestructura que está detrás. Esto permite disponer de la aplicación en marcha para su supervisión y da la opción de que accedan terceros interesados en el proyecto. Heroku integra diferentes servicios dentro de su estructura y cuenta con un plan gratuito para aplicaciones de poco consumo.

Puede accederse a la herramienta DSMantenimiento desde la siguiente dirección:

https://dsm-frontend.herokuapp.com/home

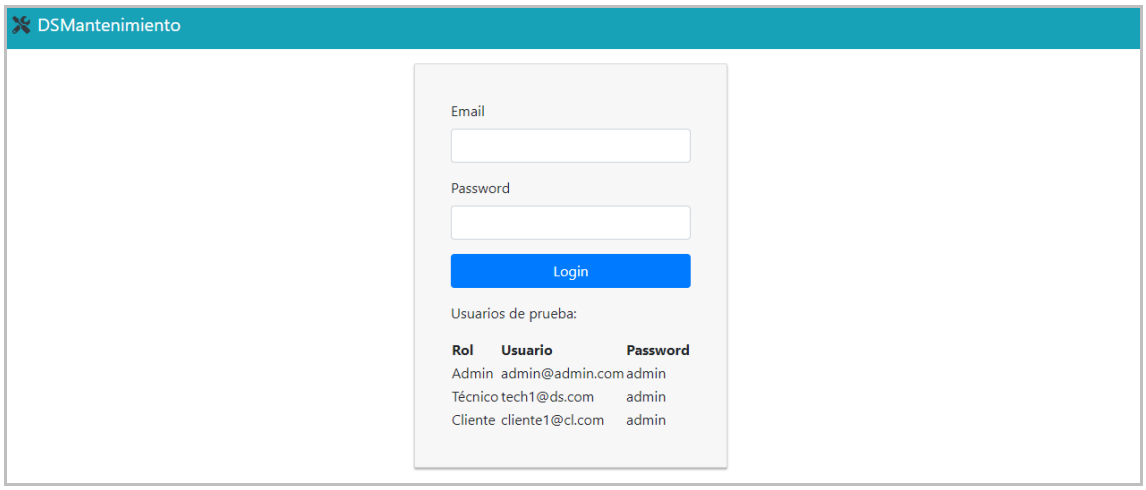


Ilustración 66: Acceso a la aplicación desde Heroku

7. Conclusiones

Este proyecto ha servido como cierre al itinerario de Ingeniería del software en el que he podido adquirir conocimientos sobre las distintas fases que forman el desarrollo de software a través de la realización de distintas asignaturas. Considero que ha sido muy interesante reforzar estos conocimientos realizando un proyecto en el que poder aplicar gran parte de lo aprendido.

Teniendo en cuenta los objetivos planteados al comenzar el proyecto, considero que se han cumplido en gran medida. Se ha desarrollado una aplicación web que permite gestionar las incidencias que surgen en una empresa y se han desarrollado todas las funcionalidades que se plantearon en un inicio.

Considero que la metodología en cascada que se ha seguido ha resultado acertada, tanto por las características propias del proyecto, como por la metodología a seguir en una asignatura como el Proyecto Final de Grado, en la que el desarrollo es realizado por una única persona.

En cuanto a la planificación realizada, no se han introducido grandes cambios respecto al planteamiento inicial y he podido seguir en gran medida el calendario propuesto. En ocasiones, las dificultades de compaginar la vida académica con la laboral han provocado desfases respecto a la planificación de las tareas, pero no han afectado a la consecución de los objetivos.

Tras la finalización de este TFG, han quedado pendientes las siguientes líneas de trabajo que me gustaría seguir desarrollando en una segunda fase del proyecto para poder obtener una herramienta más funcional:

Funcionalidades por desarrollar en una segunda fase:

- La principal tarea por abordar sería permitir al usuario establecer su contraseña. Actualmente es el administrador el que asigna una contraseña al crear un usuario y no debería ser así en un entorno en producción.

- Envío de emails a los técnicos cuando se les asigna una incidencia. También sería interesante avisar por email cuando se publique un comentario en una incidencia, tanto a técnicos como a clientes.

- Búsqueda de incidencias por nº de serie de las máquinas. Sería de gran ayuda para los técnicos de cara a consultar el historial de una máquina concreta y visualizar rápidamente averías previas.

- Opción de añadir máquinas por parte de usuarios con perfil Técnico. Puede ocurrir que no todas las máquinas de un cliente estén dadas de alta en la plataforma por lo que, al dar de alta una incidencia, sería

interesante que un técnico pudiera dar de alta máquinas sin tener que esperar a que lo haga un administrador.

- Campo de observaciones sólo visible para técnicos y no para clientes.

Por último, otro aspecto para mejorar sería la revisión del acceso a la web desde dispositivos móviles y el desarrollo de una app móvil. Esto podría servir para que un técnico pueda utilizar la aplicación también cuando hace alguna salida a una empresa cliente, de forma que pueda gestionar las incidencias in situ.

8. Glosario

API: siglas de Application Programming Interface. Es un conjunto de definiciones y protocolos que permiten la comunicación entre dos aplicaciones a través de un conjunto de reglas.

Backend: se trata de la parte de una aplicación web que se encarga de los procesos necesarios funcionen correctamente.

Docker: es un proyecto que automatiza el despliegue de aplicaciones de software dentro de contenedores de software.

Framework: es una estructura tecnológica que sirve como base para el desarrollo de software. Permite realizar desarrollos de forma más rápida y con código más limpio y evita dedicar tiempo a tareas repetitivas comunes en todos los proyectos.

Frontend: se trata de la parte visible de una página web y es la encargada de mostrar los datos proporcionados por el backend.

IDE: Entorno Integrado de Desarrollo. Es una aplicación que proporciona ayudas al programador para el desarrollo del software y que permite aumentar su productividad. Entre estas ayudas se encuentran el auto completado de código, compiladores, interpretes, etc.

Java EE: Java Enterprise Edition. Es un conjunto de estándares tecnológicos dedicados al desarrollo de aplicaciones empresariales distribuidas en Java.

JWT: estándar basado en JSON para la creación de tokens de acceso que permiten propagar de forma segura la identidad de un usuario.

Repositorio: es el lugar donde se almacena el código de una aplicación. Lo habitual es que cuente con un sistema de control de versiones de forma que se mantenga un registro de los cambios realizados.

IDE: Entorno Integrado de Desarrollo. Es una aplicación que proporciona ayudas al programador para el desarrollo del software y que permite aumentar su productividad. Entre estas ayudas se encuentran el auto completado de código, compiladores, interpretes, etc.

Java EE: Java Enterprise Edition. Es un conjunto de estándares tecnológicos dedicados al desarrollo de aplicaciones empresariales distribuidas en Java.

JWT: estándar basado en JSON para la creación de tokens de acceso que permiten propagar de forma segura la identidad de un usuario.

React: es una librería Javascript que se centra en el desarrollo de interfaces de usuario.

Spring Boot: Spring es un framework de código abierto para el desarrollo de aplicaciones en Java. En concreto Boot es un módulo de Spring que permite simplificar la configuración y despliegue de la aplicación.

Token: es una estructura de datos de una longitud determinada que contiene una secuencia de caracteres.

9. Bibliografía

1. **Pradel Miquel, J.; Raya Martos, J.** – *Introducción a la ingeniería del software*. UOC.

2. **Vallecillo Moreno, A.; Romero Salguero, J. R.; Moreno Vergara, N.; Durán Muñoz, F. J**. – *Diseño de aplicaciones distribuidas*. UOC.

3. **Giménez, A.** - *Aplicaciones web con front-end y back-end desacoplado* - https://www.aunitz.net/aplicaciones-web-front-end-back-end-desacoplado/ (Última visita: marzo del 2021)

4. **JavaTPoint** - *Introduction to RESTful Web Services* - https://www.javatpoint.com/restful-web-services-spring-boot (Última visita: marzo del 2021)

5. **Buna, S.** - *Yes, React is taking over front-end development. The question is why*. - https://www.freecodecamp.org/news/yes-react-is-taking-over-front-end-development-the-question-is-why-40837af8ab76/ (Última visita: marzo del 2021)

6. **Stack Overflow** - *Developer Survey Results 2019* - https://insights.stackoverflow.com/survey/2019 (Última visita: marzo del 2021)

7. **codeinwp** - Angular vs React vs Vue: Which Framework to Choose in 2021 - https://www.codeinwp.com/blog/angular-vs-vue-vs-react/#part-2-community-and-development (Última visita: marzo del 2021)

8. **Java Guides** - Spring Boot + React JS CRUD – https://www.javaguides.net/2020/07/spring-boot-react-js-crud-example-tutorial.html (Última visita: marzo del 2021)

9. **Microolap** - BYTEA vs OID (Large Objects) - https://www.microolap.com/products/connectivity/postgresdac/help/tipsandtricks_byteavsoid.htm (Última visita: marzo del 2021)

10. **Arquitectura Java** - Introducción a Spring Data y JPA - https://www.arquitecturajava.com/introduccion-spring-data-y-jpa/ (Última visita: marzo del 2021)

11. **Morgan, A.J.** - Securizar un API REST utilizando JSON Web Tokens - https://www.adictosaltrabajo.com/2017/09/25/securizar-un-api-rest-utilizando-json-web-tokens/ (Última visita: marzo del 2021)

12. **Álvarez, C**. - ¿Qué es Spring Boot? - https://www.arquitecturajava.com/que-es-spring-boot

13. **Baeldung** - Documenting a Spring REST API Using OpenAPI 3.0 - https://www.baeldung.com/spring-rest-openapi-documentation (Última visita: abril del 2021)

14. **Webb, P., Syer, D., Long, J., Nicoll, S., Winch, R., Wilkinson, A., Overdijk, M., Dupuis, C.** - Spring Boot Reference Guide – Database initialization - https://docs.spring.io/spring-boot/docs/1.1.0.M1/reference/html/howto-database-initialization.html (Última visita: abril del 2021)

15. **Vergara, S.** - ¿Cuál es el mejor método de autentificación en un API REST? - https://www.itdo.com/blog/cual-es-el-mejor-metodo-de-autentificacion-en-un-api-rest/ (Última visita: abril del 2021)

10. Anexos

10.1. Manual de instalación

10.1.1. Base de datos

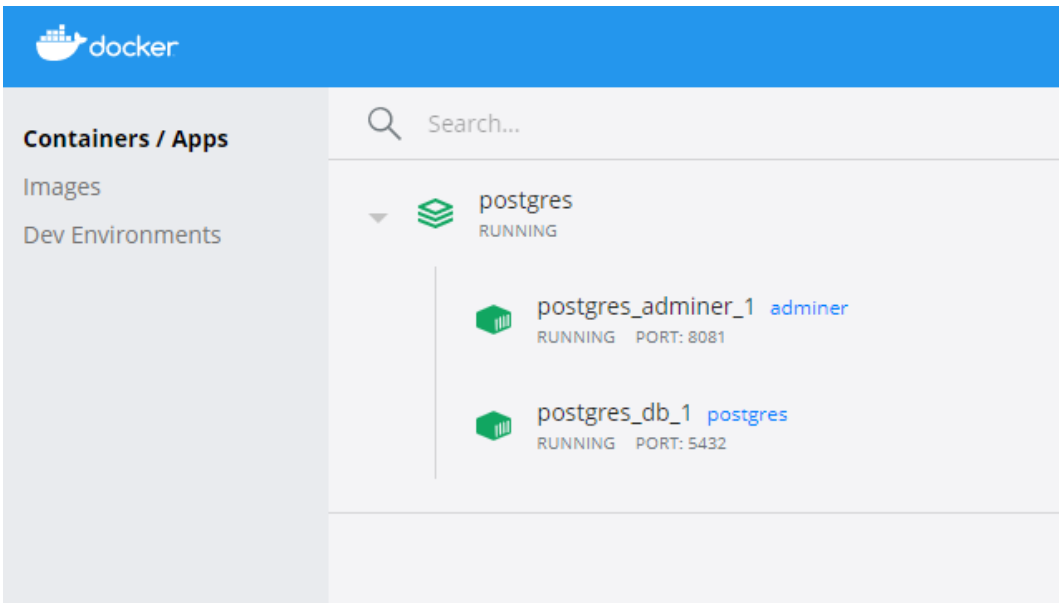
En este proyecto se ha utilizado PostgreSQL 13. La configuración utilizada en la ejecución es la siguiente:

Host	localhost
Puerto	5432
Base de datos	postgres
Usuario	postgres
Contraseña	postgres
Esquema	uoc

Se puede usar Docker para tener una base de datos Postgres con administración adminer web de manera sencilla. Para ello, se ha incluido en la raíz del proyecto dsm-backend.zip el fichero docker-compose.yml.

Se debe ejecutar el comando compose para que se descargue los contenedores de postgres y adminer y los autoconfigure:

”*docker-compose -f docker-compose.yml create*”



10.1.2. Backend

El proyecto utiliza Maven para la gestión de dependencias y despliegue por lo que será necesario para la ejecución del mismo. Se ha utilizado el jdk versión 11 para el desarrollo.

Se debe descomprimir el fichero dsm-backend.zip. En caso de querer configurar los datos de conexión a la base de datos debe editarse el fichero *src/main/resources/application.properties*:

```
application.properties
1 spring.datasource.url= jdbc:postgresql://localhost:5432/postgres
2 spring.datasource.username= postgres
3 spring.datasource.password= postgres
4 spring.datasource.autoCommit=false
5
```

En la raíz del proyecto backend se ha incluído el fichero ***CREATE_SCHEMA.sql*** que deberá ejecutarse antes de la primera ejecución del proyecto.

Dado que se utiliza la API de persistencia Spring Data JPA, no se generan las tablas de la base de datos mediante sql. Se crearán automáticamente en la primera ejecución del proyecto.

Se ha incluído en el proyecto el fichero ***INSERTS.sql*** que proporciona las inserciones básicas para poder comenzar a utilizar la aplicación.

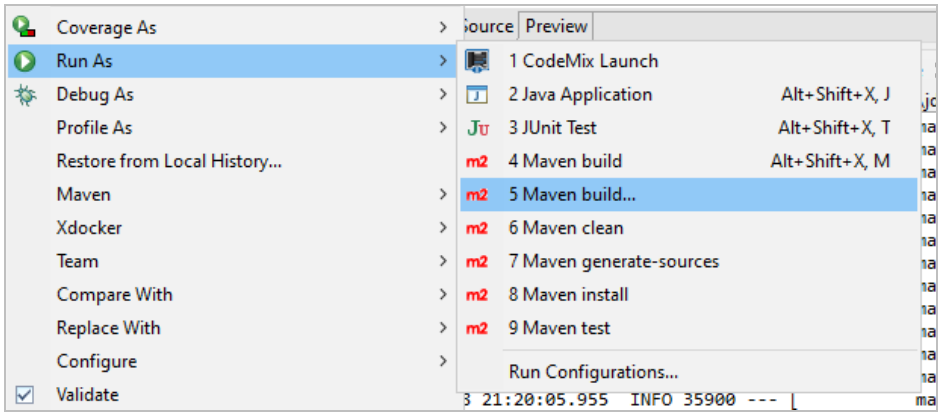
Ejecutando desde línea de comandos:

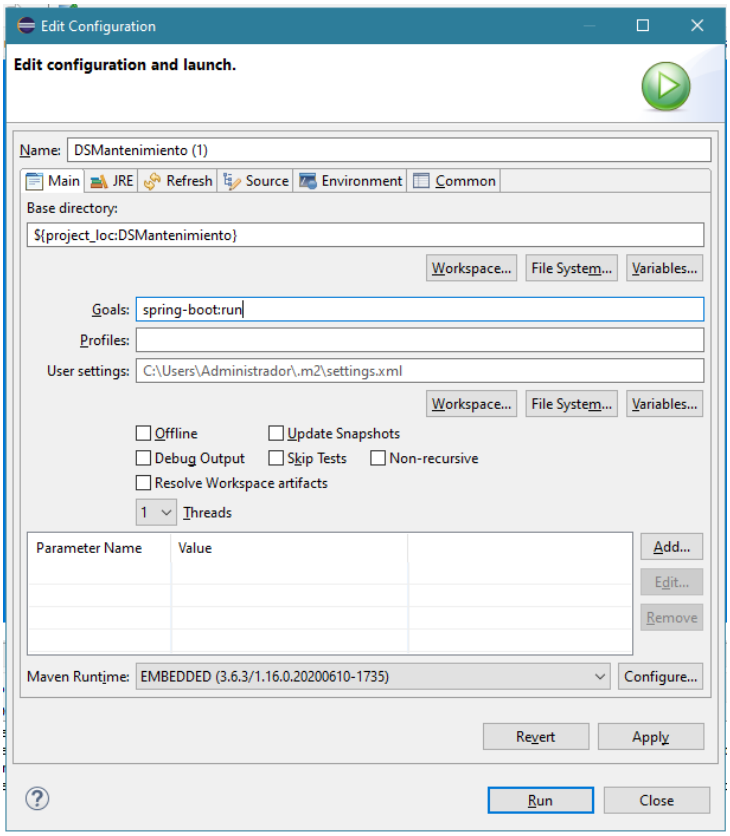
En la carpeta raíz del proyecto se debe ejecutar el siguiente comando:

mvn spring-boot:run

Ejecutando desde eclipse:

Tras importar el proyecto en eclipse como proyecto Maven, puede ponerse en marcha mediante la opción *Run as > Maven build…* y indicando dentro de Goals: *spring-boot:run*.





En la consola, podemos ver la siguiente salida que indica que se ha puesto en marcha correctamente:

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.4.5)

2021-06-08 21:20:03.872  INFO 35900 --- [           main] u.e.d.DsMantenimientoApplication         : Starting DsMantenimientoApplication using Java 11.0.9.1 on GFI12651 with PID
2021-06-08 21:20:03.874  INFO 35900 --- [           main] u.e.d.DsMantenimientoApplication         : No active profile set, falling back to default profiles: default
2021-06-08 21:20:04.516  INFO 35900 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2021-06-08 21:20:04.629  INFO 35900 --- [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 105 ms. Found 11 JPA repository
2021-06-08 21:20:04.844  INFO 35900 --- [           main] trationDelegate$BeanPostProcessorChecker : Bean 'org.springframework.security.access.expression.method.DefaultMethodSec
2021-06-08 21:20:04.848  INFO 35900 --- [           main] trationDelegate$BeanPostProcessorChecker : Bean 'methodSecurityMetadataSource' of type [org.springframework.security.ac
2021-06-08 21:20:05.493  INFO 35900 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2021-06-08 21:20:05.503  INFO 35900 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2021-06-08 21:20:05.503  INFO 35900 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.45]
2021-06-08 21:20:05.601  INFO 35900 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2021-06-08 21:20:05.601  INFO 35900 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1690 ms
2021-06-08 21:20:05.713  WARN 35900 --- [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries m
2021-06-08 21:20:05.815  INFO 35900 --- [           main] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitInfo [name: default]
2021-06-08 21:20:05.849  INFO 35900 --- [           main] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 5.4.30.Final
2021-06-08 21:20:05.955  INFO 35900 --- [           main] o.hibernate.annotations.common.Version   : HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
2021-06-08 21:20:06.031  INFO 35900 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2021-06-08 21:20:06.108  INFO 35900 --- [           main] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2021-06-08 21:20:06.120  INFO 35900 --- [           main] org.hibernate.dialect.Dialect            : HHH000400: Using dialect: org.hibernate.dialect.PostgreSQLDialect
2021-06-08 21:20:06.925  INFO 35900 --- [           main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transacti
2021-06-08 21:20:06.933  INFO 35900 --- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2021-06-08 21:20:07.873  INFO 35900 --- [           main] o.s.s.web.DefaultSecurityFilterChain     : Will secure any request with [org.springframework.security.web.context.reque
2021-06-08 21:20:07.915  INFO 35900 --- [           main] o.s.s.concurrent.ThreadPoolTaskExecutor  : Initializing ExecutorService 'applicationTaskExecutor'
2021-06-08 21:20:08.081  INFO 35900 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2021-06-08 21:20:08.095  INFO 35900 --- [           main] u.e.d.DsMantenimientoApplication         : Started DsMantenimientoApplication in 4.549 seconds (JVM running for 4.92)
```

Tras ejecutarlo, el proyecto estará en marcha en la url siguiente:

http://localhost:8080/

10.1.3. Frontend

Para la ejecución del frontend será necesario tener instalado node.js y npm (https://nodejs.org/en/download/).

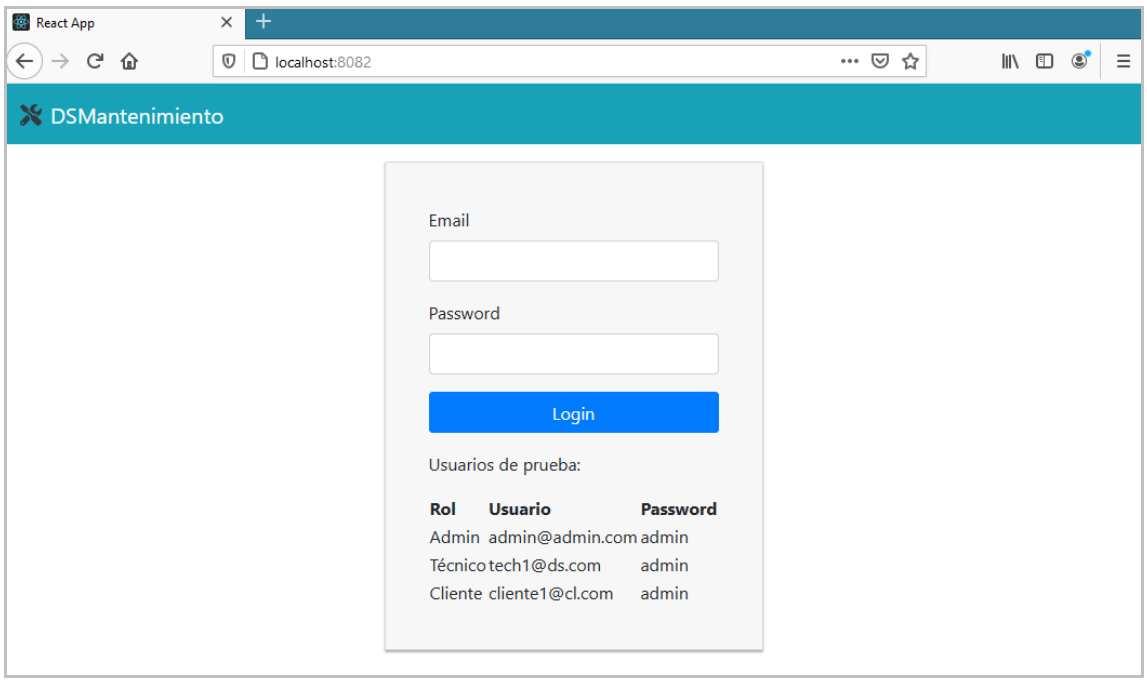
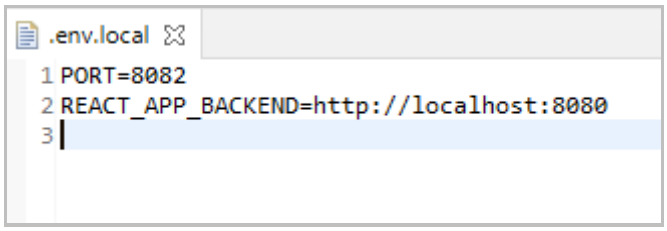
La ejecución de esta parte será similar a la anterior. Se debe descomprimir el proyecto dsm-frontend.zip y desde línea de comandos en el directorio react-ds ejecutar lo siguiente:

npm install

Este comando realiza la instalación de los paquetes indicados en el fichero *package.json*. Una vez completado, debe ejecutarse el siguiente comando para arrancar la aplicación:

npm start

Tras ejecutarlo, la aplicación estará accesible desde http://localhost:8082/



La aplicación cuenta en su raíz con el fichero *.env.local* en el que se indica en qué puerto se ejecutará y la dirección en la que se encuentra el backend: