

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Fixtrack: Gestión de seguimiento de reparaciones electrónicas

Autor

Tomás Hidalgo Martín

Directores

Dr. Javier Sánchez Monedero

D. Sergio Gómez Bachiller

Mayo 2024



UNIVERSIDAD DE CÓRDOBA

Licencias

Licencia de documentación: Los manuales de este Trabajo Fin de Grado están distribuidos bajo la licencia GNU Free Documentation License versión 1.3. Puede encontrar una copia de la licencia en la web de la Free Software Fundation: <http://www.gnu.org/copyleft-fdl.html>.

Licencia de código: El código de este proyecto está distribuido bajo la GNU Affero General Public License versión 3.0. Puede encontrar una copia de la licencia en la web de la Free Software Fundation: <https://www.gnu.org/licenses/agpl-3.0.en.html>.

Agradecimientos

Me gustaría agradecer a mi familia, pilar constante de apoyo y sacrificio. Gracias por brindarme la mejor educación posible y por proveerme con todos los recursos necesarios para mi trayectoria académica. Papa, mamá, lo he logrado. Aquel inicio difícil, lleno de tropiezos y dudas, se transformó en esta realidad de la que estoy inmensamente orgulloso. Todo esto fue posible gracias a su amor y apoyo incondicional.

No puedo dejar de mencionar a mis amigos, compañeros de innumerables aventuras académicas y personales. Con ustedes, he compartido desde resolver complejas integrales hasta analizar los aspectos financieros de un proyecto financiero. Más allá de los conocimientos académicos, me llevo amistades valiosas que espero duren toda la vida.

Mi gratitud también se extiende a mis directores, y en particular a ti, Sergio. Desde aquel encuentro en el Aula de Software Libre, has sido una influencia transformadora en mi vida. Para mí, has sido más que un mentor en programación; eres un ejemplo de integridad y paciencia. Gracias por todo lo que me has enseñado, lo cual ha sido fundamental en mi formación.

Por último, agradezco a todas las personas que han compartido este camino conmigo. A cada uno de ustedes, mi más sincero agradecimiento.

Índice general

1. Introducción	1
2. Definición del problema	3
2.1. Problema real	3
2.2. Problema técnico	3
2.2.1. Funcionamiento	4
2.2.2. Entorno	4
2.2.3. Vida esperada	5
2.2.4. Ciclo de mantenimiento	5
2.2.5. Competencia	5
2.2.6. Aspecto externo	6
2.2.7. Estandarización	6
2.2.8. Calidad y fiabilidad	7
2.2.9. Programa de tareas	7
2.2.10. Pruebas	8
2.2.11. Seguridad	8
3. Objetivos	11
4. Antecedentes	13
4.1. RepairDesk	13
4.2. RepairShopr	14
4.3. mHelpDesk	15

4.4. Tabla comparativa	16
5. Restricciones	17
5.1. Factores dato	17
5.2. Factores estratégicos	17
5.2.1. Herramientas para desarrollar la interfaz de usuario	18
5.2.2. Herramientas para desarrollar el backend	18
5.2.3. Herramientas para desarrollar el frontend	19
5.2.4. Herramientas para desarrollar el proyecto	20
5.2.5. Persistencia de los datos	20
6. Recursos	23
6.1. Recursos humanos	23
6.2. Recursos software	23
6.3. Recursos hardware	24
7. Especificación de requisitos	25
7.1. Requisitos de información	25
7.2. Requisitos funcionales	27
7.3. Casos de uso	28
7.3.1. Actores	30
7.3.2. CU01: Registro de usuario	30
7.3.3. CU02: Acceso al sistema	31
7.3.4. CU03: Cambiar Contraseña	32
7.3.5. CU04: Crear usuario	32
7.3.6. CU05: Eliminar usuario	33
7.3.7. CU06: Lista de usuarios	34
7.3.8. CU07: Crear Reparación	35
7.3.9. CU08: Editar Reparación	36
7.3.10. CU09: Eliminar Reparación	36
7.3.11. CU10: Lista de Reparaciones	37
7.3.12. CU11: Crear Dispositivo	38

7.3.13. CU12: Eliminar Dispositivo	39
7.3.14. CU13: Lista de Dispositivos	39
7.4. Requisitos no funcionales	40
8. Análisis de la información	43
8.1. Descripción del modelo conceptual de datos	43
8.2. Descripción de las clases	44
8.2.1. Clase Cliente	44
8.2.2. Clase Dispositivo	44
8.2.3. Clase Técnico	45
8.2.4. Clase Reparación	45
9. Análisis funcional	47
9.1. Descripción del comportamiento	49
9.1.1. Contexto usuario	49
9.1.1.1. Diagrama de secuencia: creación de un usuario	49
9.1.1.2. Diagrama de secuencia: cambiar contraseña de un usuario	49
9.1.1.3. Diagrama de secuencia: eliminación de un usuario	50
9.1.1.4. Diagrama de secuencia: obtención de un usuario	50
9.1.1.5. Diagrama de secuencia: obtención de todos los usuarios	51
9.1.2. Contexto dispositivos	51
9.1.2.1. Diagrama de secuencia: creación de un dispositivo	51
9.1.2.2. Diagrama de secuencia: eliminación de un dispositivo	52
9.1.2.3. Diagrama de secuencia: obtención de un dispositivo	52
9.1.2.4. Diagrama de secuencia: obtención de todos los dispositivos	53
9.1.3. Contexto reparaciones	53
9.1.3.1. Diagrama de secuencia: creación de una reparación	53
9.1.3.2. Diagrama de secuencia: edición de una reparación	54
9.1.3.3. Diagrama de secuencia: eliminación de una reparación	54
9.1.3.4. Diagrama de secuencia: obtención de una reparación	55
9.1.3.5. Diagrama de secuencia: obtención de todas las reparaciones	55

10. Diseño del sistema	57
10.1. Modelado	57
10.2. Arquitectura del sistema	65
10.2.1. Diagrama de despliegue	67
10.3. Datos	68
11. Diseño de la interfaz	69
11.1. Características generales de la interfaz	69
11.2. Interfaces de la aplicación de administrador	70
11.2.1. Interfaz de inicio	70
11.2.2. Interfaz de inicio de sesión	71
11.2.3. Interfaz de <i>Home</i> del administrador	72
11.2.4. Interfaz de página de reparaciones del administrador	73
11.2.4.1. Listado de reparaciones	73
11.2.4.2. Información de una reparación	74
11.2.4.3. Crear una reparación	75
11.2.5. Interfaz de página de dispositivos del administrador	76
11.2.5.1. Listado de dispositivos	76
11.2.5.2. Crear un dispositivo	77
11.2.6. Interfaz de página de usuarios del administrador	78
11.2.6.1. Listado de usuarios	78
11.2.6.2. Crear un usuario	79
11.3. Interfaces de la aplicación de técnico	81
11.3.1. Interfaz de inicio	81
11.3.2. Interfaz de inicio de sesión	82
11.3.3. Interfaz de <i>Home</i> del técnico	83
11.3.4. Interfaz de página de reparaciones del técnico	84
11.3.4.1. Listado de reparaciones	84
11.3.4.2. Información de una reparación	85
11.3.5. Interfaz del perfil del técnico	86

11.4. Interfaces de la aplicación del cliente	87
11.4.1. Interfaz de inicio	87
11.4.2. Interfaz de inicio de sesión	88
11.4.3. Interfaz de <i>Home</i> del cliente	89
11.4.4. Interfaz del perfil del cliente	90
12. Pruebas	91
13. Conclusiones y futuras mejoras	95
13.1. Análisis de objetivos	95
13.1.1. Objetivo 1: Gestión de partes de reparaciones	95
13.1.2. Objetivo 2: Gestión de dispositivos electrónicos	95
13.1.3. Objetivo 3: Sistema de notificaciones	95
13.1.4. Objetivo 4: Gestión de roles de usuario que permitan acceder a las diferentes partes del sistema	96
13.1.5. Objetivo 5: Gestión de registro de clientes	96
13.1.6. Objetivo 6: Gestión de registro de técnicos	96
13.1.7. Objetivo 7: Uso de KPIs	96
13.2. Problemas encontrados y soluciones	97
13.2.1. Empleo de dos bases de datos de lectura	97
13.2.2. Reconstrucción de eventos con la base de datos de <i>EventStore</i>	97
13.2.3. Complejidad en configurar Docker	97
13.3. Conclusiones	98
13.4. Futuras mejoras	98
Referencias	99
A. Manual de administrador	103
A.1. Instalación del software	103
A.1.1. Requisitos previos	103
A.1.2. Instalación del software	104
A.1.3. Comprobación de la instalación	105

A.1.4. Desinstalación del software	105
B. Manual de usuario	107
B.1. Interfaces de la aplicación de administrador	107
B.1.1. Interfaz de inicio	107
B.1.2. Interfaz de inicio de sesión	108
B.1.3. Interfaz de Home del administrador	109
B.1.4. Interfaz de página de reparaciones del administrador	110
B.1.4.1. Listado de reparaciones	110
B.1.4.2. Información de una reparación	111
B.1.4.3. Crear una reparación	112
B.1.5. Interfaz de página de dispositivos del administrador	113
B.1.5.1. Listado de dispositivos	113
B.1.5.2. Crear un dispositivo	114
B.1.6. Interfaz de página de usuarios del administrador	115
B.1.6.1. Listado de usuarios	115
B.1.6.2. Crear un usuario	116
B.2. Interfaces de la aplicación de técnico	117
B.2.1. Interfaz de inicio	117
B.2.2. Interfaz de inicio de sesión	118
B.2.3. Interfaz de Home del técnico	119
B.2.4. Interfaz de página de reparaciones del técnico	120
B.2.4.1. Listado de reparaciones	120
B.2.4.2. Información de una reparación	121
B.2.5. Interfaz del perfil del técnico	122
B.3. Interfaces de la aplicación del cliente	123
B.3.1. Interfaz de inicio	123
B.3.2. Interfaz de inicio de sesión	124
B.3.3. Interfaz de Home del cliente	125
B.3.4. Interfaz del perfil del cliente	126

C. Manual de código	127
C.1. Introducción	127
C.2. Requisitos previos	127
C.3. Descripción modular	128
C.3.1. Backend	128
C.3.2. <i>Frontend</i>	133
C.3.3. Contratos	133
C.4. Dependencias	134
C.4.1. Dependencias para el entorno de desarrollo	134

Índice de figuras

4.1. Captura de pantalla de RepairDesk	13
4.2. Captura de pantalla de RepairShopr	14
4.3. Captura de pantalla de mHelpDesk	15
7.1. Diagrama de casos de uso	29
8.1. Diagrama de clases	43
9.1. Esquema de arquitectura hexagonal	48
9.2. Diagrama de secuencia de la creación de un usuario	49
9.3. Diagrama de secuencia de la edición de un usuario	50
9.4. Diagrama de secuencia de la eliminación de un usuario	50
9.5. Diagrama de secuencia de la obtención de un usuario	51
9.6. Diagrama de secuencia de la obtención de todos los usuarios	51
9.7. Diagrama de secuencia de la creación de un dispositivo	52
9.8. Diagrama de secuencia de la eliminación de un dispositivo	52
9.9. Diagrama de secuencia de la obtención de un dispositivo	53
9.10. Diagrama de secuencia de la obtención de todos los dispositivos	53
9.11. Diagrama de secuencia de la creación de una reparación	54
9.12. Diagrama de secuencia de la edición de una reparación	54
9.13. Diagrama de secuencia de la eliminación de una reparación	55
9.14. Diagrama de secuencia de la obtención de una reparación	55
9.15. Diagrama de secuencia de la obtención de todas las reparaciones	56

10.1. Diseño y modelado del flujo ideal del negocio con <i>Event Modeling</i>	60
10.2. Diseño y modelado de los distintos estados de reparaciones	61
10.3. Diseño y modelado del registro de usuarios con <i>Event Modeling</i>	62
10.4. Diseño y modelado de crear reparación con <i>Event Modeling</i>	63
10.5. Diseño y modelado de actualizar una reparación con <i>Event Modeling</i>	64
10.6. Esquema <i>Domain-Driven Design</i>	66
10.7. Diagrama de despliegue	67
 11.1. Interfaz de la página de inicio	70
11.2. Interfaz de la página de inicio de sesión	71
11.3. Interfaz de página principal del administrador	72
11.4. Interfaz de página de listado de reparaciones	73
11.5. Interfaz de página de información de una reparación	74
11.6. Interfaz de página de crear una reparación	75
11.7. Interfaz de página de listado de dispositivos	76
11.8. Interfaz de página de crear un dispositivo	77
11.9. Interfaz de página de listado de usuarios	78
11.10. Interfaz de página de crear un usuario	79
11.11. Interfaz de la notificación de alta de usuario	80
11.12. Interfaz de la página de inicio	81
11.13. Interfaz de la página de inicio de sesión	82
11.14. Interfaz de página principal del técnico	83
11.15. Interfaz de página de listado de reparaciones	84
11.16. Interfaz de página de información de una reparación	85
11.17. Interfaz de página del perfil del técnico	86
11.18. Interfaz de la página de inicio	87
11.19. Interfaz de la página de inicio de sesión	88
11.20. Interfaz de página principal del cliente	89
11.21. Interfaz de página del perfil del cliente	90
 B.1. Interfaz de la página de inicio	107

B.2. Interfaz de la página de inicio de sesión	108
B.3. Interfaz de página principal del administrador	109
B.4. Interfaz de página de listado de reparaciones	110
B.5. Interfaz de página de información de una reparación	111
B.6. Interfaz de página de crear una reparación	112
B.7. Interfaz de página de listado de dispositivos	113
B.8. Interfaz de página de crear un dispositivo	114
B.9. Interfaz de página de listado de usuarios	115
B.10. Interfaz de página de crear un usuario	116
B.11. Interfaz de la página de inicio	117
B.12. Interfaz de la página de inicio de sesión	118
B.13. Interfaz de página principal del técnico	119
B.14. Interfaz de página de listado de reparaciones	120
B.15. Interfaz de página de información de una reparación	121
B.16. Interfaz de página del perfil del técnico	122
B.17. Interfaz de la página de inicio	123
B.18. Interfaz de la página de inicio de sesión	124
B.19. Interfaz de página principal del cliente	125
B.20. Interfaz de página del perfil del cliente	126

Índice de tablas

4.1. Tabla comparativa entre FixTrack y plataformas de reparación existentes	16
12.1. Ficheros principales de testing relacionados con los dispositivos	91
12.2. Ficheros principales de testing relacionados con los dispositivos	92
12.3. Ficheros principales de testing relacionados con los dispositivos	93
A.1. Tabla de requisitos para la instalación de las dependencias del sistema	103
C.1. Tabla de requisitos para la instalación de las dependencias del sistema	128
C.2. Ficheros principales del backend	128
C.3. Directorios principales de los módulos de la aplicación	129
C.4. Directorios user	130
C.5. Directorios device	131
C.6. Directorios work-order	132
C.7. Directorios auth	132
C.8. Directorios apps/web	133
C.9. Directorios libs/contracts	134
C.10. Dependencias de desarrollo	136
C.11. Dependencias de producción	137

CAPÍTULO

1

Introducción

En el contexto actual, la reparación de dispositivos electrónicos se ha convertido en un servicio crucial. Dependemos de forma creciente de dispositivos como teléfonos móviles, tabletas, computadoras y sistemas de entretenimiento en el hogar. Sin embargo, el proceso de reparación a menudo es opaco y estresante para los clientes, quienes usualmente se encuentran en la oscuridad respecto al estado de sus dispositivos y deben depender de actualizaciones intermitentes.

Por un lado, los clientes buscan rapidez y fiabilidad en el servicio de reparación. Valorizan estar informados continuamente sobre el estado de sus dispositivos y los posibles contratiempos que puedan surgir durante la reparación. Por otro lado, las tiendas de reparación enfrentan desafíos en el seguimiento eficiente de los dispositivos en reparación y en la comunicación efectiva con los clientes. Un sistema ineficiente puede llevar a la insatisfacción del cliente y a pérdidas económicas para la tienda.

En este escenario, el proyecto “FixTrack” se presenta como una solución innovadora. Esta plataforma web permitirá a los clientes ver el estado de las reparaciones de sus dispositivos y recibir observaciones del técnico encargado. Para las tiendas, ofrecerá herramientas para optimizar sus procesos internos y mantener una comunicación fluida con los clientes.

Asimismo, la plataforma se enfocará en la gestión eficiente de partes de reparación, un elemento esencial para documentar el trabajo realizado y las piezas reemplazadas. Esto es crucial para la transparencia y eficiencia del proceso de reparación. Además, incluirá la monitorización de Indicadores Clave de Rendimiento (KPI), proporcionando a las tiendas una visión clara de la eficacia de sus operaciones y permitiéndoles identificar áreas de mejora.

Con “FixTrack”, se busca facilitar el proceso de reparación tanto para clientes como para tiendas, mejorando la comunicación, acelerando las reparaciones y aumentando la satisfacción general. Este proyecto, realizado como Trabajo de Fin de Grado de la titulación de Ingeniería Informática en la Universidad de Córdoba, no solo aporta una herramienta tecnológica útil, sino también una solución práctica y relevante para un problema de actualidad y de importancia creciente.

Definición del problema

2.1. Problema real

Este *Trabajo de Fin de Grado* supone un problema en la actualidad y que nos afecta a todos. Como clientes queremos tener constancia en todo momento de cómo avanza la reparación de un dispositivo electrónico, en especial si dicho aparato electrónico lo usamos con mucha frecuencia al cabo del día como puede ser un teléfono móvil. Es por ello que este proyecto busca proveer de una monitorización del estado de la reparación de un dispositivo electrónico al cliente.

Por otro lado de cara al propietario de la compañía reparadora de dispositivos electrónicos, posee la capacidad de monitorizar distintas estadísticas del negocio e indicadores de rendimiento de cada uno de los técnicos, muy útil desde el punto de vista empresarial.

2.2. Problema técnico

En este apartado se expone el enfoque técnico que se ha llevado a cabo para la implementación de la aplicación con el objetivo de solventar el problema presentado en la sección anterior. Se han desarrollado tres portales web, el primero de ellos se trata del portal de usuarios por donde se observan las reparaciones del cliente y su estado, el segundo se corresponde al portal de gestión de reparaciones desde el punto de vista del técnico y el tercero el portal para gestión y monitorización desde el punto de vista del administrador. Se hará uso de la metodología *DDD (Domain Driven Design)* para la descripción del problema.

2.2.1. Funcionamiento

El funcionamiento de este proyecto se basará en una herramienta software que permita la gestión y seguimiento del proceso de reparación de un dispositivo electrónico.

Desde el punto de vista del administrador del sistema:

- Instalación y configuración de la aplicación.
- Control sobre los distintos roles de usuarios.
- Administración de clientes, técnicos, reparaciones y estadísticas de negocio.
- Administración de la base de datos.

Desde el punto de vista del usuario:

- Identificación. El usuario podrá iniciar sesión y registrarse en el sistema.
- Gestión de su perfil.
- Visualización de las reparaciones de sus dispositivos electrónicos.
- Notificación en alta en el sistema.

Desde el punto de vista del técnico:

- Identificación. El usuario podrá iniciar sesión y registrarse en el sistema.
- Gestión de su perfil.
- Visualización y modificación de las reparaciones de sus dispositivos electrónicos.
- Notificación en alta en el sistema.

2.2.2. Entorno

El entorno de ejecución del *backend* de la aplicación será en servidores con soporte de lenguajes de programación web y acceso de internet. Los entornos de ejecución del *frontend* de las aplicaciones se realizará mediante el uso de unas aplicaciones web *responsive* que se comunicarán con el entorno *backend*.

2.2.3. Vida esperada

No se ha estimando una vida útil exacta para el proyecto al tratarse de un sistema de gestión de reparaciones de dispositivos electrónicos, es decir, algo bastante recurrente en estos tiempos, por lo tanto se ha previsto que los distintos aplicativos son lo suficientemente flexibles para poder iterar en el futuro, asumiendo nuevas necesidades y nuevos criterios. Por lo que se determina que la vida del proyecto será extensa y debe seguir activa mientras sea útil.

2.2.4. Ciclo de mantenimiento

La cantidad de revisiones a la aplicación se marcará dependiendo del nivel de mantenimiento necesario que se necesite en los entornos de desarrollo de la aplicación, tanto del administrador del sistema *backend* de la aplicación como del el administrador del sistema *frontend*. Ambos entornos de desarrollo serán actualizados de manera recurrente con el objetivo de mantener la aplicación en versiones estables que garanticen el correcto funcionamiento de la aplicación y, de este modo, evitar posibles brechas de seguridad.

En cualquier caso, se tratará de llevar un seguimiento del código para estudiar alguna cambio en la aplicación. Puede establecerse un chequeo del entorno para comprobar que la herramienta va a seguir siendo compatible con el entorno descrito anteriormente, independientemente de que alguna notificación de seguridad urgente obligue a hacer una operación de mantenimiento antes.

2.2.5. Competencia

Este **Trabajo Fin de Grado** es una solución a un problema conocido el cual se ha intentado resolver múltiples veces de diferentes formas, sin embargo es difícil encontrar un sistema que unifique en un mismo ecosistema el seguimiento, monitorización y empleo de estadísticas del rendimiento de una aplicación de gestión de reparaciones de dispositivos.

2.2.6. Aspecto externo

El *frontend* del sistema será presentando mediante tres aplicaciones web, la primera de ellas se trata del portal de usuarios que usarían los clientes, la segunda se corresponde al panel de gestión de reparaciones de técnicos y la tercera la plataforma para gestión de reparaciones de dispositivos electrónicos., cumpliendo con estándares de accesibilidad y uso para permitir que cualquier usuario pueda utilizar la aplicación.

El proyecto será distribuido en *Github*, una de las plataformas más populares de desarrollo de software. En dicha plataforma podrá ser accesible por cualquier usuario interesado, pudiendo descargar cualquiera de los entornos de desarrollo del proyecto, incluyendo también los manuales de instalación y de usuario para que puedan ser consultados ante cualquier duda.

2.2.7. Estandarización

La estandarización del proyecto seguirá las directrices estudiadas durante los años de grado, permitiendo legibilidad y compresión del código, para que cualquier persona interesada en el proyecto no tenga ninguna complicación en entender el flujo de funcionamiento de los entornos.

Por un lado, el *backend* será realizado en *NestJS* [1] un *framework* que utiliza como lenguaje de programación *TypeScript*, superconjunto de *Javascript*, por lo que se hará uso de los estándares marcados en ***ECMA Script*** para la estandarización del código fuente de este entorno. Este será a su vez construido a través de ***REST***, un estilo de arquitectura de software que define una serie de pautas para crear servicios web distribuidos como la *World Wide Web*, por lo que se seguirán sus recomendaciones.

Por otro lado, el *frontend* se desarrollará también con *NextJS* un *framework* para la librería *React* [2], específica para construir interfaces de usuario. Al tratarse de tres aplicaciones web, todas se desarrollarán del mismo modo utilizando librerías nativas para algunos casos como la autenticación con *JWT* [3] y se hará uso de las guías de estilo publicadas por dicha compañía para el desarrollo de aplicaciones web.

2.2.8. Calidad y fiabilidad

Para la evaluación de la calidad y fiabilidad del proyecto será necesario el control periódico por parte de los directores y el alumno del proyecto, con el objetivo de comprobar que se están cumpliendo las directrices establecidas acerca de la estandarización del proyecto y que el producto en construcción sea sostenible y cumpla los objetivos establecidos a cumplir.

Es importante tener en cuenta los siguientes objetivos especificados:

- El *backend* tendrá el cometido de verificar que la información y la lógica impuesta por los casos de uso es veraz y consistente, haciendo que se cumplan la lógica de negocio establecida para el proyecto.
- El *frontend* deberá de mostrar la información procesada del *backend* de forma que cumpla los estándares establecidos anteriormente.

2.2.9. Programa de tareas

Para la realización de este proyecto se ha propuesto el empleo de SCRUM[4] como base para la planificación del tiempo y la implementación de éste. Ésta es una metodología ágil que permite una mayor flexibilidad en los procesos y aumenta la calidad del software entregado. El uso de SCRUM implica un conjunto de buenas prácticas donde se realizan entregas parciales y regulares del producto final, lo que conlleva beneficios muy significativos. Para diseñar dicho dominio se hará uso del enfoque de construcción de software llamado *Domain-driven Design* [5] y *Event Modeling* [6].

Esta metodología está basada en *sprints* o iteraciones, las cuales permiten establecer el ritmo de trabajo del proyecto, siendo la duración habitual entre 2 y 3 semanas. El objetivo es que a final de cada *sprint*, todo lo realizado por el equipo de desarrollo aporte un valor al producto que está siendo desarrollado.

Dentro de esta metodología podemos diferenciar las siguientes etapas:

1. Análisis y estudio del problema. En esta etapa se verá el alcance del proyecto, definiendo el dominio de éste, los objetivos que se pretenden contemplar y las tecnologías a utilizar para alcanzar dichas metas. Para diseñar dicho dominio se hará uso del enfoque de
-

construcción de software llamado *Domain-driven Design* [5]. Y para representar el flujo de los distintos procesos de negocio se usa *Event Modeling* [6].

2. Especulación. Fase que se repetirá por cada iteración del proyecto, llevando cabo las siguientes tareas:
 - Desarrollo y supervisión de requisitos generales.
 - Comprobación de funcionalidades esperadas.
 - Plan de entrega. Se proponen las fechas de las diferentes versiones e iteraciones a realizar.
3. Implementación. Una vez realizadas las tareas de la fase anterior, se incluye la nueva funcionalidad al producto.
4. Revisión de la iteración. Se comprueba que todo lo incluido corresponde con la planificación y que se ha realizado correctamente. Se volvería en este momento a la fase 2, repitiendo el bucle hasta la última iteración.
5. Entrega. Correspondrá con la última iteración, entregando el producto final y la documentación correspondiente.

2.2.10. Pruebas

Durante el desarrollo del proyecto se ha empleado la prueba manual punto a punto desde la interfaz de usuario, con esto se asegura el correcto funcionamiento de la conexión *frontend-backend*. Y además se han realizado pruebas unitarias para probar el correcto funcionamiento de la lógica de negocio del *backend*.

2.2.11. Seguridad

La mayoría de la seguridad del sistema ha recaído en la parte del *backend* ya que, es la encargada de asegurar la integridad de los datos en la base de datos como las distintas restricciones de negocio. Los puntos más críticos de seguridad son los siguientes:

- La autenticación debe ser segura y capaz de reportar errores en la autenticación de usuarios.
-

- La información en la base de datos de escritura posee los datos y metadatos encriptados, por lo que solo se puede leer desde la proyección de ésta en otra base de datos.
- Deben seguirse los mantenimientos de seguridad escrupulosamente, ya que son la principal causa de ataques a este tipo de sistemas.

CAPÍTULO

3

Objetivos

El objetivo principal de este *Trabajo Fin de Grado* es desarrollar una plataforma web, “FixTrack”, que permita a los clientes monitorizar el estado de la reparación de sus dispositivos electrónicos, mejorar la comunicación con el técnico de reparación, y finalmente, mejorar la transparencia y eficiencia del proceso de reparación de dispositivos electrónicos.

La realización de esta aplicación será cumplida mediante otros objetivos más específicos:

- Gestión de partes de reparaciones.
- Gestión de dispositivos electrónicos.
- Sistema de notificaciones.
- Gestión de roles de usuario que permitan acceder a las diferentes partes del sistema.
- Gestión de registro de clientes.
- Gestión de registro de técnicos.
- Uso de KPIs.

CAPÍTULO

4

Antecedentes

4.1. RepairDesk

RepairDesk es una plataforma de gestión de tiendas de reparación que permite a las empresas llevar un registro eficiente de las reparaciones de los dispositivos, las ventas de productos y las facturas. Ofrece la posibilidad de generar informes detallados y personalizados que pueden ayudar a los propietarios de las tiendas a tomar decisiones informadas. Sin embargo, no ofrece seguimiento en tiempo real de las reparaciones ni permite observaciones directas de los técnicos que están haciendo la reparación. Además, no es software libre.

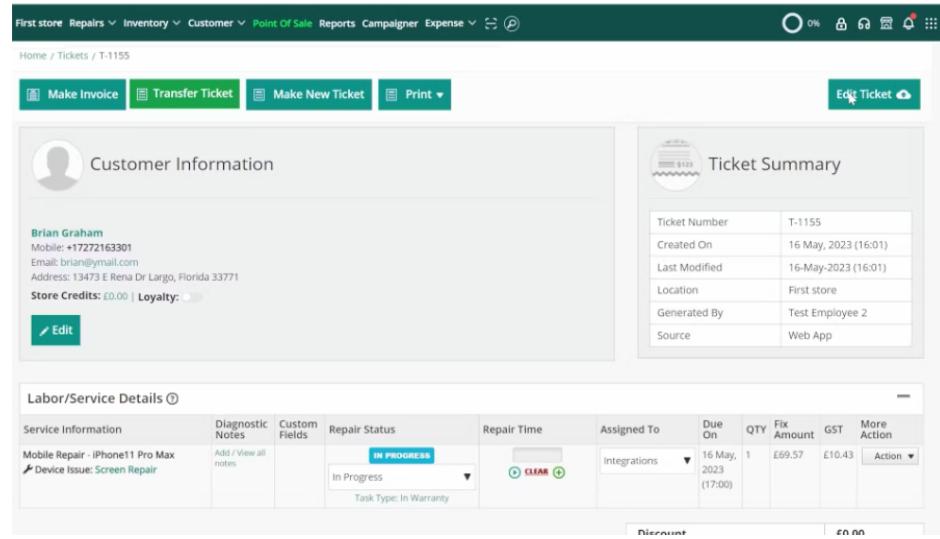


Figura 4.1: Captura de pantalla de RepairDesk

4.2. RepairShopr

RepairShopr es un software de gestión para tiendas de reparación que combina funciones de CRM (gestión de relaciones con los clientes), POS (sistema de punto de venta) y gestión de tickets en una única plataforma. Ofrece funcionalidades como la gestión de inventario, facturación y seguimiento de trabajo. Al igual que RepairDesk, no ofrece seguimiento en tiempo real de las reparaciones ni permite que los técnicos dejen observaciones directas en las reparaciones. RepairShopr tampoco es software libre.

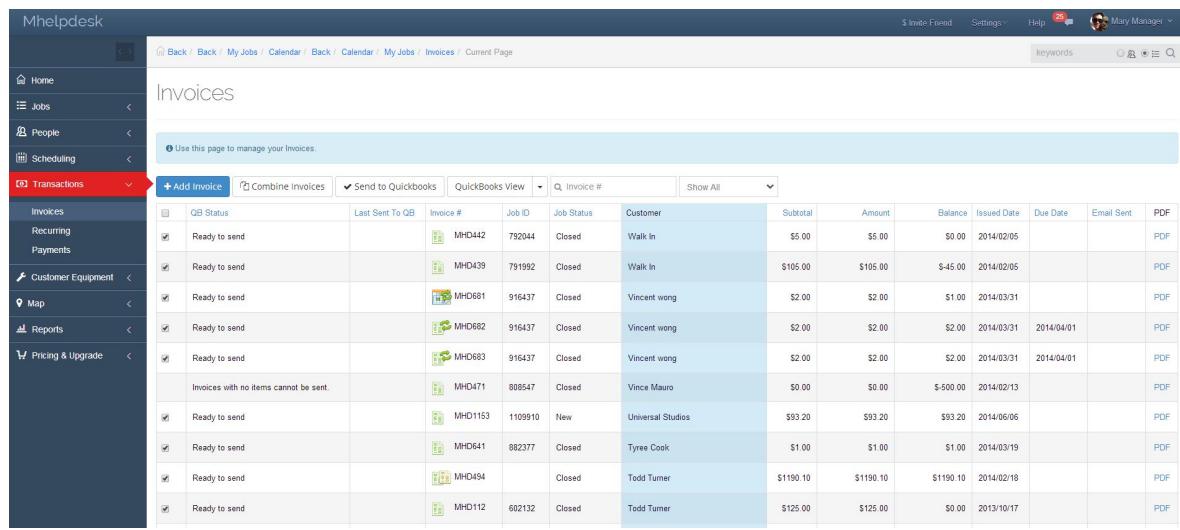
The screenshot shows the RepairShopr Demo software interface. At the top, there is a navigation bar with icons for Customers, Invoices, Estimates, Tickets (highlighted in orange), Logistics, Inventory, Field Jobs, POS, Leads, Marketr (highlighted in green), and Admin. The Tickets icon has a red notification badge with the number '4'. Below the navigation bar, the main title is 'Tickets' with a sub-count of '58'. There are several search and filter options: 'Containing' (with a search input field), 'Status is Not' (set to 'Resolved'), 'Status is' (dropdown menu), 'Hide Invoiced' (checkbox), 'Priority is' (dropdown menu), 'Type is' (dropdown menu set to 'Tech'), and a 'Search' button. A 'New Ticket' button is also visible. To the right of the search area, there is a 'feedback' button with a speech bubble icon. The main content area displays a table of ticket records:

#	CUSTOMER	SUBJECT	CREATED	STATUS	ISSUE TYPE	TECH	TYPE	PRIORITY	LAST UPDATED
11218	Buck Lubowitz	dell laptop with will not power on	01-06-15	Waiting for Parts	Hardware	Demo Admin		not set	12 days
11219	Leatha Thiel	iphone with cannot get online	01-07-15	New	Virus	click to add		not set	about 1 month
11220	Buck Lubowitz	iphone with fbi virus	01-07-15	New	Software	click to add		not set	about 1 month
11222	David Briggs	iphone with will not power on	01-08-15	New	Virus	click to add		not set	about 1 month
11223	Brannon Willms	desktop with cannot get online	01-08-15	New	Virus	click to add		not set	about 1 month
11230	Edgar Bins	macbook with fbi virus	01-09-15	New	Virus	click to add		not set	about 1 month
11231	Jimmy Briggs	macbook with broken screen	01-09-15	New	Software	click to add		not set	about 1 month
11234	Merle Conroy	dell laptop with needs a tuneup	01-10-15	New	Hardware	click to add		not set	about 1 month
11235	Lora Murazik	dell laptop with broken screen	01-10-15	New	Software	click to add		not set	about 1 month
11236	Lora Murazik	macbook with cannot get online	01-11-15	New	Software	click to add		not set	about 1 month
11237	Surendra Goud	macbook with will not power on	01-11-15	New	Virus	click to add		not set	about 1 month

Figura 4.2: Captura de pantalla de RepairShopr

4.3. mHelpDesk

mHelpDesk es un software de gestión de servicios de campo que se utiliza en una variedad de industrias, incluyendo limpieza, fontanería, y por supuesto, reparación de dispositivos. Mientras que ofrece características como la programación y despacho de trabajos, facturación y seguimiento de trabajo, no es específico para la reparación de dispositivos y carece de características importantes como el seguimiento en tiempo real de las reparaciones y las observaciones de los técnicos. Además, al igual que los otros dos, no es software libre.



La captura de pantalla muestra la interfaz de usuario de mHelpDesk. En la parte superior, hay un menú con enlaces a Back, My Jobs, Calendar, Invoices y Current Page. A la derecha del menú, se encuentran los botones para Invitar Amigo, Configuración, Ayuda y el nombre del usuario Mary Manager. La barra de búsqueda tiene un campo para keywords y un icono de búsqueda.

En la izquierda, hay un menú lateral desplegable que incluye Home, Jobs, People, Scheduling, Transactions (selecciónada), Invoices, Recurring Payments, Customer Equipment, Map, Reports y Pricing & Upgrade.

El contenido principal se titula "Invoices". Una barra azul indica que se puede usar esta página para administrar las facturas. Hay botones para "Add Invoice", "Combine Invoices", "Send to Quickbooks" y "QuickBooks View".

La tabla de facturas muestra los siguientes datos:

QB Status	Last Sent To QB	Invoice #	Job ID	Job Status	Customer	Subtotal	Amount	Balance	Issued Date	Due Date	Email Sent	PDF
<input checked="" type="checkbox"/> Ready to send		MHD442	792044	Closed	Walk In	\$5.00	\$5.00	\$0.00	2014/02/05			PDF
<input checked="" type="checkbox"/> Ready to send		MHD439	791992	Closed	Walk In	\$105.00	\$105.00	\$-45.00	2014/02/05			PDF
<input checked="" type="checkbox"/> Ready to send		MHD681	916437	Closed	Vincent wong	\$2.00	\$2.00	\$1.00	2014/03/31			PDF
<input checked="" type="checkbox"/> Ready to send		MHD682	916437	Closed	Vincent wong	\$2.00	\$2.00	\$2.00	2014/03/31	2014/04/01		PDF
<input checked="" type="checkbox"/> Ready to send		MHD683	916437	Closed	Vincent wong	\$2.00	\$2.00	\$2.00	2014/03/31	2014/04/01		PDF
Invoices with no items cannot be sent.		MHD471	808547	Closed	Vince Mauro	\$0.00	\$0.00	\$500.00	2014/02/13			PDF
<input checked="" type="checkbox"/> Ready to send		MHD1153	1109910	New	Universal Studios	\$93.20	\$93.20	\$93.20	2014/06/06			PDF
<input checked="" type="checkbox"/> Ready to send		MHD641	882377	Closed	Tyree Cook	\$1.00	\$1.00	\$1.00	2014/03/19			PDF
<input checked="" type="checkbox"/> Ready to send		MHD494	602132	Closed	Todd Turner	\$1190.10	\$1190.10	\$1190.10	2014/02/18			PDF
<input checked="" type="checkbox"/> Ready to send		MHD112	602132	Closed	Todd Turner	\$125.00	\$125.00	\$0.00	2013/10/17			PDF

Figura 4.3: Captura de pantalla de mHelpDesk

4.4. Tabla comparativa

A continuación se muestra una tabla con un resumen de las características más relevantes ofrecidas por las aplicaciones mostradas anteriormente:

	RepairDesk	RepairShopr	mHelpDesk	FixTrack
Gestión de roles de usuario	✓	✓	✓	✓
Notificaciones al cliente	✓	✓	✓	✓
Específico para reparación de dispositivos	✓	✓		✓
Seguimiento de reparación en tiempo real				✓
Observaciones del técnico				✓
Software Libre				✓

Tabla 4.1: Tabla comparativa entre FixTrack y plataformas de reparación existentes

5.1. Factores dato

Las siguientes restricciones son aquellas que vienen a raíz de la naturaleza del problema y se debe de cumplir con ellas obligatoriamente.

- Será liberado con una licencia de *Software Libre* para que cualquier entidad pueda utilizarlo. Dentro de las distintas opciones disponibles entre la multitud de licencias existentes (Free Software Foundation, 2012) se ha elegido la licencia AGPL debido a que es la más extendida y la que garantiza las cuatro libertades que definen al software libre (Free Software Foundation, 1996).
- Esta restricción se ajusta a las directrices impuestas por la Universidad de Córdoba acerca de los *Trabajo de Fin de Grado* en la Escuela Politécnica de Córdoba donde el total de horas empleadas para la realización del proyecto no debe superar 300 horas (12 créditos ECTS).
- Debido a los distintas casuísticas el *frontend* del sistema deberá estar compuesto por una aplicación web *responsive* con distintos roles: una para el administrador, otra para los técnicos y otra para los clientes.

5.2. Factores estratégicos

En esta sección se exponen las herramientas que se han decidido emplear para solventar el problema, así como posibles alternativas y algunas ventajas frente a otras no seleccionadas

finalmente.

5.2.1. Herramientas para desarrollar la interfaz de usuario

El diseño de las interfaces de usuario de las aplicaciones se ha desarrollado de la misma manera en todos los casos, usando librerías de diseño y componentes, destacando especialmente ***Chakra UI***[7]. Esta herramienta provee al desarrollador múltiples componentes, fáciles de usar y que se ajustan a los estándares de diseño comunes en el diseño de aplicaciones móviles y web.

5.2.2. Herramientas para desarrollar el backend

En cuanto al backend, se pretendió buscar entornos de desarrollo que facilitasen la creación de una *API* de tipo *REST*. A su vez, el *framework* escogido debía de ser lo suficientemente flexible para poder recrear un sistema que satisficiera los requisitos de la aplicación, apoyado también de una buena documentación.

Por la versatilidad, la alta demanda en el mercado, la escalabilidad que plantean y la favorable curva de aprendizaje de desarrollo que ofrecen, en primer lugar se decidió debatir entre los dos siguientes *frameworks* principalmente, aunque también cupo la posibilidad de haber planteado algún otro parecido que se comentarán posteriormente:

- **Express:** Es uno de los *frameworks* más conocidos de *Node.js* por su versatilidad, velocidad y minimalismo. Provee un conjunto robusto de herramientas para aplicaciones web y móviles. Sin embargo, su enfoque minimalista puede ser tanto una fortaleza como una limitación; su sencillez implica que menos funcionalidades están integradas de manera directa, lo que podría requerir que los desarrolladores inviertan más tiempo en la configuración y el desarrollo de componentes adicionales que en otros *frameworks* que ya incluyen estas capacidades de forma predeterminada.
 - **NestJS:** Es un framework de *Node.js* pensado para el desarrollo de aplicaciones progresivas del lado del servidor, basado en *TypeScript* y *Javascript*. Este framework está apoyado bajo los pilares fundamentales de frameworks HTTP robustos como Express
-

y Fastify, por lo que añade funcionalidades nuevas que permiten a los desarrolladores una mayor rapidez en la creación de sus aplicaciones. NestJS provee un nivel de abstracción por encima de los dos frameworks mencionados anteriormente pero expone la configuración de la API a los desarrolladores, permitiendo ajustar su funcionalidad con total flexibilidad. Este framework permite la separación en diferentes capas de su funcionalidad, tales como la capa de negocio, la capa de aplicación y la lógica del servidor, facilitando la implementación de patrones de diseño útiles como CQRS(Command Query Responsibility Segregation) [8] o Event Sourcing [9], y siguiendo una arquitectura de software limpia como la arquitectura hexagonal, interesantes para el desarrollo de software de mayor calidad.

Entre las opciones expuestas anteriormente, se determinó que, por la escalabilidad que ofrece, flexibilidad y por la buena experiencia de desarrollo que plantea integrando metodologías como *EventSourcing*, *CQRS* entre otras, se ha decidido utilizar NestJS como framework de desarrollo del backend.

Cabe destacar por último que durante el desarrollo del *backend*, pretendiendo responder a ciertos inconvenientes que se generaban a la hora de la reconstrucción de eventos por el uso de *EventSourcing*, se ha utilizado un módulo desarrollado por el *Aula de software Libre* que resuelve dicho problema y el cual es de código abierto y está alojado en Github¹.

5.2.3. Herramientas para desarrollar el frontend

A la hora de elegir las posibles herramientas de desarrollo que más comunidad y estabilidad tienen detrás a día de hoy, se debatió entre las siguientes opciones:

- **NextJS:** Este *framework* detallado previamente en la sección 5.2.2, tiene múltiples beneficios, tanto para las aplicaciones de nuestros clientes como para el propio desarrollo, siendo uno de ellos la simplificación del enrutamiento basado en páginas con soporte de rutas dinámicas, además de la capacidad para generar sitios estáticos, usar *server-side rendering (SSR)* o una combinación de ambos según la necesidad de cada página.

¹<https://github.com/aulasoftwarelibre/nestjs-eventstore>

- **Vue.js:** Se autodenomina como un *framework progresivo* de Javascript. Esta calificación nos indica que en realidad con este framework podemos crear todo tipo de desarrollos. Podrían ser componentes sencillos, que implementan una parte determinada de una aplicación web, pero también aplicaciones *frontend* completas, con su sistema de enrutamiento y pequeñas librerías que completan el uso de este framework.

Finalmente, debido a las características presentadas anteriormente y a la amplia documentación que ofrece, se ha decidido que NextJS es la herramienta que mejor rendimiento desempeñará y brindará un experiencia de desarrollo idónea.

5.2.4. Herramientas para desarrollar el proyecto

Dado que la aplicación va a usar el mismo lenguaje en *frontend* y *backend*, Typescript, se ha decidido usar un monorepositorio en Github. La principal ventaja para esta toma de decisión es compartir tipos de datos y código entre frontend y backend, evitando así la duplicación de código. Para una facilidad de uso y mejor control de la configuración de ambos ecosistemas se ha usado *NX*[10] como motor de creador de monorepositorios.

5.2.5. Persistencia de los datos

Al igual que en el diseño del sistema, para el almacenamiento de datos también se ha seguido el patrón de diseño CQRS el cual separa las lecturas y escrituras en diferentes modelos. Tenemos el modelo de escritura y el modelo de lectura, la fuente que consume del modelo de escritura para ofrecer proyecciones de los datos en función de las necesidades de los usuarios.

Se han usado bases de datos diferentes para cada uno de los modelos:

- Modelo de lectura: *MongoDB*. Esta base de datos nos ofrece una mayor flexibilidad a la hora de generar las proyecciones de manera independiente lo cual conlleva a una mejor experiencia de desarrollo y una oportunidad factible de escalabilidad en caso de que el sistema lo requiera.
 - Modelo de escritura: *EventStoreDB*. Es una base de datos basada en el almacenamiento de los eventos que suceden en el sistema y absolutamente enfocada para usar con *EventSourcing*. Entre las características por las que se ha seleccionado esta base de
-

datos para el almacenamiento podemos destacar que está perfectamente integrada con el *framework* NestJS, el almacenamiento inmutable de los eventos ejecutados a lo largo del tiempo y que garantiza la escritura, es decir, los eventos ejecutados persisten en la fuente de verdad, además de encriptación de los datos y metadatos.

CAPÍTULO

6

Recursos

6.1. Recursos humanos

El autor del proyecto encargado de realizar la solución al problema propuesto, junto con el análisis de los requisitos, diseño, codificación, pruebas y generación de documentación corresponde al alumno del Grado de Ingeniería Informática, Tomás Hidalgo Martín.

Los directores encargados de la coordinación y supervisión del proyecto son: Dr. Javier Sánchez Monedero: Investigador “Beatriz Galindo” del Área de Ciencias de la Computación e Inteligencia Artificial del Departamento de Informática y Análisis Numérico; y D. Sergio Gómez Bachiller: Ingeniero en Informática. Colaborador del Aula de Software Libre de la Universidad de Córdoba.

6.2. Recursos software

Para la realización de este proyecto se hará uso de los siguientes recursos software:

- Sistema operativo: Ubuntu¹.
- Entorno de desarrollo: VSCode².
- Sistema de control de versiones: Git³.

¹<https://ubuntu.com>

²<https://code.visualstudio.com>

³<https://git-scm.com>

- Editor de documentos: Overleaf⁴.

Para el desarrollo del backend del proyecto se ha utilizado:

- Nodejs, entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor⁵.
- Nestjs como framework de NodeJS⁶.
- Express como framework de NodeJS⁷.

Para el desarrollo del frontend del proyecto se ha utilizado:

- Reactjs, librería de Javascript para construir interfaces de usuario⁸.
- Nextjs como framework de ReactJS⁹.

6.3. Recursos hardware

Para la realización de este proyecto se hará uso del ordenador personal con las siguientes características:

- Dispositivo: Lenovo Ideapad 5.
- Procesador: AMD Ryzen 7 5700u.
- Memoria: 16GB RAM DDR4.
- Almacenamiento: 512GB SSD.

⁴<https://www.overleaf.com>

⁵<https://nodejs.org>

⁶<https://nestjs.com>

⁷<https://expressjs.com>

⁸<https://reactjs.org>

⁹<https://nextjs.org>

Especificación de requisitos

Este **Trabajo de Fin de Grado** se llevará a cabo el desarrollo de tres portales web, uno para usuarios público, otro para el administrador y un último para técnicos. En cada uno de estos portales se podrá consumir una **API REST**(Application Programming Interface Representational State Transfer) capaz de identificar los distintos roles y permisos de un usuario.

7.1. Requisitos de información

A continuación, se recoge la información relevante al almacenamiento y gestión de los datos que las aplicaciones llevarán a cabo. Los requisitos de información son los siguientes:

Requisito 1: Información sobre el usuario

El sistema almacenará la siguiente información sobre los usuarios de la aplicación.

Datos específicos:

- Identificador
- Correo electrónico
- Contraseña

- Rol del usuario

Requisito 2: Información sobre el dispositivo electrónico

El sistema almacenará la siguiente información sobre un dispositivo electrónico de la aplicación.

Datos específicos:

- Identificador
- Tipo de dispositivo
- Marca del dispositivo
- Modelo de dispositivo

Requisito 3: Información sobre una reparación

El sistema almacenará la siguiente información sobre una reparación de un dispositivo electrónico de la aplicación.

Datos específicos:

- Identificador
 - Identificador del técnico asociado a dicha reparación
 - Identificador del usuario propietario del dispositivo
 - Identificador del dispositivo asociado a dicha reparación
 - Estado de la reparación
 - Descripción de la reparación
-

- Precio de la reparación

7.2. Requisitos funcionales

Los requisitos funcionales describirán el comportamiento que del sistema, es decir, lo que debe poder hacer. Se usa la siguiente nomenclatura: **RFnn**, donde **nn** indica el número que identifica al requisito en cuestión.

A continuación se muestran los requisitos funcionales identificados para el proyecto

- **RF01** Un cliente debe poder registrarse mediante el método de identificación federada proporcionado.
- **RF02** Un técnico debe poder registrarse mediante el método de identificación federada proporcionado.
- **RF03** Un administrador debe poder registrarse mediante el método de identificación federada proporcionado.
- **RF04** El sistema permitirá acceso a administradores, técnicos y clientes.
- **RF05** El administrador del sistema debe poder crear un nuevo cliente en el sistema.
- **RF06** El administrador del sistema debe poder eliminar un cliente en el sistema.
- **RF07** El administrador del sistema debe poder ver la lista de todos los clientes en el sistema.
- **RF08** El administrador del sistema debe poder crear un nuevo técnico en el sistema.
- **RF09** El administrador del sistema debe poder eliminar un técnico en el sistema.
- **RF10** El administrador del sistema debe poder ver la lista de todos los técnicos en el sistema.
- **RF11** El administrador del sistema debe poder crear un nuevo administrador en el sistema.

- **RF12** El administrador del sistema debe poder eliminar un administrador en el sistema.
- **RF13** El administrador del sistema debe poder ver la lista de todos los administradores en el sistema.
- **RF14** El administrador del sistema debe poder crear una reparación y asignarla a un técnico.
- **RF15** El administrador del sistema debe poder editar una reparación.
- **RF16** El administrador del sistema debe poder eliminar una reparación.
- **RF17** El administrador del sistema debe poder ver la lista de todas las reparaciones en el sistema.
- **RF18** Un usuario debe poder cambiar la contraseña en su perfil.
- **RF19** El administrador del sistema debe poder crear un nuevo dispositivo en el sistema.
- **RF20** El administrador del sistema debe poder eliminar un dispositivo en el sistema.
- **RF21** El administrador del sistema debe poder ver la lista de todos los dispositivos en el sistema.
- **RF22** El técnico del sistema debe poder editar una reparación en el sistema.
- **RF23** El cliente del sistema debe poder visualizar el estado de sus reparaciones asignadas.

7.3. Casos de uso

El modelado ha sido realizado siguiendo las pautas de diseño de casos de uso recomendadas de **IBM, 2008**[11]. Se hará uso de la siguiente nomenclatura: **CUnn**, donde **nn** indica el número que identifica al caso de uso en cuestión.

En la figura 7.1 puede verse el esquema general de los casos de uso y los actores que los llevan a cabo.

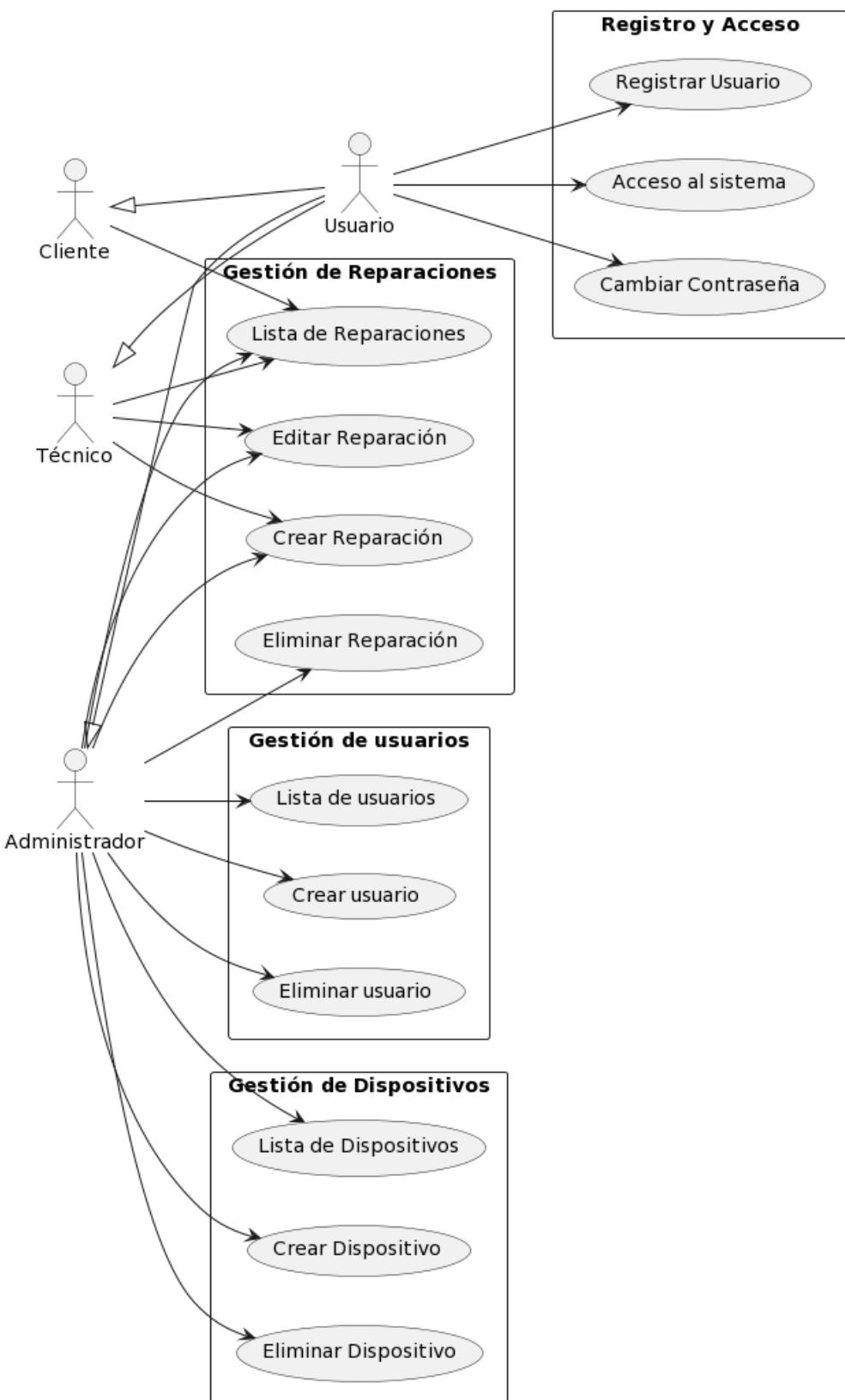


Figura 7.1: Diagrama de casos de uso

7.3.1. Actores

- **Administrador:** Usuario identificado como administrador del sistema.
- **Técnico:** Usuario identificado como técnico de reparaciones del sistema.
- **Cliente:** Usuario identificado como cliente del sistema.

7.3.2. CU01: Registro de usuario

Descripción.

Este caso de uso permite registrar información acerca de los usuarios identificado a través de la API y gestionar la información recogida.

Actores.

Actor principal: Administrador del sistema, Técnico, Cliente.

Precondiciones.

El usuario debe tener una cuenta de correo electrónico existente.

Postcondiciones.

Los cambios realizados al guardar los datos del usuario se almacenan en la base de datos.

1. Flujo de eventos.

1.1 IDENTIFICACIÓN DEL USUARIO

El usuario debe de autenticar su identidad pulsando el botón de iniciar sesión a través del portal de registro.

1.2 REGISTRO DEL USUARIO

El sistema comprueba que no existe ningún usuario registrado con la información recibida por el servicio de autenticación y en caso de no haber ningún resultado existente, registra al usuario en la plataforma. Una vez realizado este proceso el usuario podría disfrutar de los servicios del sistema.

1.3 CERRAR SESIÓN:

El usuario identificado puede cerrar sesión en cualquier momento.

2. Flujo alternativo

2.1 INICIO DE SESIÓN

Se produce en el caso de que el usuario ya exista en el sistema. En este caso se le da acceso a la aplicación de inmediato.

2.2 AUTENTICACIÓN CANCELADA

Sucede cuando el usuario cancela la autenticación en el proceso de identificación y, por lo tanto, no llega a finalizarlo.

7.3.3. CU02: Acceso al sistema

Descripción.

Este caso de uso permite a usuarios registrados acceder al sistema mediante sus credenciales establecidas.

Actores.

Actor principal: Administrador del sistema, Técnico, Cliente..

Precondiciones.

El usuario ya está registrado en el sistema.

Postcondiciones.

El usuario accede al sistema y puede utilizar sus funcionalidades.

1. Flujo de eventos.

1.1 INTRODUCIR CREDENCIALES

El usuario introduce su nombre de usuario y contraseña en la página de inicio de sesión.

1.2 VALIDAR ACCESO

El sistema valida las credenciales y permite el acceso al usuario.

1.3 INGRESO AL SISTEMA

El usuario ingresa al sistema y puede comenzar a utilizar las funcionalidades disponibles.

2. Flujo alternativo

2.1 CREDENCIALES INCORRECTAS:

Si las credenciales no son válidas, el sistema notifica al usuario y le permite intentarlo de nuevo.

7.3.4. CU03: Cambiar Contraseña

Descripción.

Este caso de uso permite a un usuario cambiar su contraseña actual por una nueva.

Actores.

Actor principal: Administrador del sistema, Técnico, Cliente..

Precondiciones.

El usuario está registrado y ha iniciado sesión en el sistema.

Postcondiciones.

La contraseña del usuario es actualizada en la base de datos.

1. Flujo de eventos.

1.1 SOLICITAR CAMBIO DE CONTRASEÑA

El usuario selecciona la opción para cambiar su contraseña en su perfil.

1.2 INTRODUCIR NUEVA CONTRASEÑA

El usuario introduce y confirma la nueva contraseña.

1.3 ACTUALIZAR CONTRASEÑA

El sistema valida y actualiza la contraseña del usuario.

2. Flujo alternativo

2.1 CONTRASEÑA INCORRECTA:

Si la nueva contraseña no cumple con los requisitos de seguridad, el sistema notifica al usuario y solicita una nueva.

7.3.5. CU04: Crear usuario

Descripción.

Este caso de uso permite al administrador del sistema crear un nuevo usuario en el sistema.

Actores.

Actor principal: Administrador del sistema.

Precondiciones.

El administrador ha iniciado sesión en el sistema.

Postcondiciones.

El nuevo usuario se crea y se añade a la base de datos del sistema.

1. Flujo de eventos.**1.1 ACCEDER A GESTIÓN DE USUARIOS**

El administrador selecciona la opción de crear un nuevo usuario dentro del panel de administración.

1.2 INTRODUCIR DATOS DEL USUARIO

El administrador ingresa la información necesaria para la creación del nuevo usuario.

1.3 CONFIRMAR Y CREAR USUARIO

El sistema valida la información y crea la nueva cuenta de usuario.

2. Flujo alternativo**2.1 DATOS INVÁLIDOS:**

Si la información proporcionada es inválida, el sistema notifica al administrador y solicita la corrección de los datos.

7.3.6. CU05: Eliminar usuario**Descripción.**

Este caso de uso permite al administrador del sistema eliminar un usuario existente.

Actores.

Actor principal: Administrador del sistema.

Precondiciones.

El administrador ha iniciado sesión y el usuario a eliminar existe en el sistema.

Postcondiciones.

El usuario se elimina de la base de datos del sistema.

1. Flujo de eventos.**1.1 SELECCIONAR USUARIO A ELIMINAR**

El administrador busca y selecciona al usuario que desea eliminar.

1.2 CONFIRMAR ELIMINACIÓN

El administrador confirma la acción de eliminación del usuario.

1.3 ELIMINAR USUARIO

El sistema elimina al usuario de la base de datos.

2. Flujo alternativo**2.1 CANCELAR ELIMINACIÓN:**

El administrador puede cancelar la eliminación antes de confirmar la acción.

7.3.7. CU06: Lista de usuarios**Descripción.**

Este caso de uso permite al administrador del sistema visualizar una lista de todos los usuarios registrados en el sistema.

Actores.

Actor principal: Administrador del sistema.

Precondiciones.

El administrador ha iniciado sesión en el sistema.

Postcondiciones.

El administrador visualiza la lista de usuarios.

1. Flujo de eventos.**1.1 ACCEDER A LISTA DE USUARIOS**

El administrador selecciona la opción para visualizar la lista de usuarios.

1.2 VISUALIZAR USUARIOS

El sistema muestra la lista de todos los usuarios registrados.

2. Flujo alternativo

2.1 NO HAY USUARIOS:

Si no hay usuarios para mostrar, el sistema informa al administrador que la lista está vacía.

7.3.8. CU07: Crear Reparación

Descripción.

Este caso de uso permite al administrador del sistema registrar una nueva reparación en el sistema y asignarla a un técnico.

Actores.

Actor principal: Administrador del sistema.

Precondiciones.

El administrador y el técnico están registrados en el sistema.

Postcondiciones.

La reparación se crea y se asigna correctamente en el sistema.

1. Flujo de eventos.

1.1 ACCEDER A GESTIÓN DE REPARACIONES

El administrador accede a la sección de gestión de reparaciones.

1.2 INTRODUCIR DATOS DE LA REPARACIÓN

El administrador introduce los detalles de la nueva reparación y selecciona un técnico para asignarla.

1.3 CONFIRMAR CREACIÓN DE REPARACIÓN

El sistema registra la nueva reparación y notifica al técnico asignado.

2. Flujo alternativo

2.1 TÉCNICO NO DISPONIBLE:

Si el técnico seleccionado no está disponible, el sistema notifica al administrador para que seleccione otro técnico.

7.3.9. CU08: Editar Reparación

Descripción.

Este caso de uso permite al administrador del sistema o al técnico asignado modificar los detalles de una reparación existente.

Actores.

Actor principal: Administrador del sistema, Técnico.

Precondiciones.

La reparación existe en el sistema y el actor ha iniciado sesión.

Postcondiciones.

Los detalles de la reparación se actualizan en el sistema.

1. Flujo de eventos.

1.1 SELECCIONAR REPARACIÓN A EDITAR

El actor busca y selecciona la reparación que desea editar.

1.2 MODIFICAR DETALLES DE LA REPARACIÓN

El actor realiza los cambios necesarios en los detalles de la reparación.

1.3 GUARDAR CAMBIOS

El sistema valida y guarda los cambios en la reparación.

2. Flujo alternativo

2.1 CANCELAR EDICIÓN:

El actor puede cancelar la edición en cualquier momento antes de guardar los cambios.

7.3.10. CU09: Eliminar Reparación

Descripción.

Este caso de uso permite al administrador del sistema eliminar una reparación registrada.

Actores.

Actor principal: Administrador del sistema.

Precondiciones.

El administrador ha iniciado sesión y la reparación a eliminar está registrada en el sistema.

Postcondiciones.

La reparación se elimina del sistema.

1. Flujo de eventos.**1.1 SELECCIONAR REPARACIÓN A ELIMINAR**

El administrador selecciona la reparación que desea eliminar.

1.2 CONFIRMAR ELIMINACIÓN

El administrador confirma la acción de eliminación de la reparación.

1.3 ELIMINAR REPARACIÓN

El sistema elimina la reparación de la base de datos.

2. Flujo alternativo**2.1 CANCELAR ELIMINACIÓN:**

El administrador puede cancelar la eliminación antes de confirmar la acción.

7.3.11. CU10: Lista de Reparaciones**Descripción.**

Este caso de uso permite a los actores visualizar una lista de todas las reparaciones registradas en el sistema.

Actores.

Actor principal: Administrador del sistema, Técnico, Cliente.

Precondiciones.

Los actores han iniciado sesión en el sistema.

Postcondiciones.

Los actores pueden visualizar la lista de reparaciones.

1. Flujo de eventos.**1.1 ACCEDER A LISTA DE REPARACIONES**

El actor accede a la sección donde se listan todas las reparaciones.

1.2 VISUALIZAR REPARACIONES

El sistema muestra la lista de reparaciones con su estado actual.

2. Flujo alternativo

2.1 NO HAY REPARACIONES:

Si no hay reparaciones para mostrar, el sistema informa que la lista está vacía.

7.3.12. CU11: Crear Dispositivo

Descripción.

Este caso de uso permite al administrador del sistema añadir un nuevo dispositivo al inventario del sistema.

Actores.

Actor principal: Administrador del sistema.

Precondiciones.

El administrador ha iniciado sesión en el sistema.

Postcondiciones.

El dispositivo nuevo se añade al sistema y se refleja en el inventario.

1. Flujo de eventos.

1.1 ACCEDER A GESTIÓN DE DISPOSITIVOS

El administrador accede al módulo de gestión de dispositivos.

1.2 INTRODUCIR DATOS DEL DISPOSITIVO

El administrador introduce la información necesaria del nuevo dispositivo.

1.3 CONFIRMAR Y AÑADIR DISPOSITIVO

El sistema registra el nuevo dispositivo en el inventario.

2. Flujo alternativo

2.1 DATOS INVÁLIDOS:

Si los datos son inválidos, el sistema notifica al administrador y solicita la corrección de la información.

7.3.13. CU12: Eliminar Dispositivo

Descripción.

Este caso de uso permite al administrador del sistema retirar un dispositivo del inventario del sistema.

Actores.

Actor principal: Administrador del sistema.

Precondiciones.

El administrador ha iniciado sesión y el dispositivo a eliminar está registrado en el sistema.

Postcondiciones.

El dispositivo se elimina del inventario del sistema.

1. Flujo de eventos.

1.1 SELECCIONAR DISPOSITIVO A ELIMINAR

El administrador selecciona el dispositivo que desea eliminar.

1.2 CONFIRMAR ELIMINACIÓN

El administrador confirma la acción de eliminación del dispositivo.

1.3 ELIMINAR DISPOSITIVO

El sistema retira el dispositivo del inventario.

2. Flujo alternativo

2.1 CANCELAR ELIMINACIÓN:

El administrador puede cancelar la eliminación antes de confirmar la acción.

7.3.14. CU13: Lista de Dispositivos

Descripción.

Este caso de uso permite al administrador del sistema visualizar la lista completa de dispositivos en el inventario del sistema.

Actores.

Actor principal: Administrador del sistema.

Precondiciones.

El administrador ha iniciado sesión en el sistema.

Postcondiciones.

El administrador obtiene una lista actualizada de todos los dispositivos en el sistema.

1. Flujo de eventos.**1.1 ACCEDER A LA LISTA DE DISPOSITIVOS**

El administrador accede al módulo donde se listan los dispositivos.

1.2 VISUALIZAR DISPOSITIVOS

El sistema muestra la lista completa de dispositivos con su información correspondiente.

2. Flujo alternativo**2.1 NO HAY DISPOSITIVOS:**

Si no hay dispositivos para mostrar, el sistema informa que la lista está vacía.

7.4. Requisitos no funcionales

En último lugar, se presentarán los requisitos no funcionales analizados del sistema, haciendo uso de la siguiente nomenclatura: **RNFnn**, donde **nn** identifica un número consecutivo que sirve para identificar al requisito no funcional en cuestión.

- **RNF01** Las interfaces de usuario deben estar diseñadas de tal modo que ofrezcan una experiencia llevadera para todos los usuarios, cuenten con experiencia previa de las tecnologías o no.
 - **RNF02** El sistema debe manejar de manera eficaz los posibles errores que se den en la comunicación entre el *textit{frontend}* y el *backend* mostrando mensajes de error legibles y comprensibles para el usuario.
 - **RNF03** El sistema deberá ser fácil de instalar y configurar.
-

- **RNF04** El sistema deberá garantizar la privacidad y la integridad de los datos registrados, asegurando que usuarios no autorizados consuman datos personales de otros usuarios.

Análisis de la información

8.1. Descripción del modelo conceptual de datos

En esta sección se muestra el modelo de datos necesarios para gestionar la información en el sistema, a la hora de definir su esquema se ha optado por usar la metodología UML.

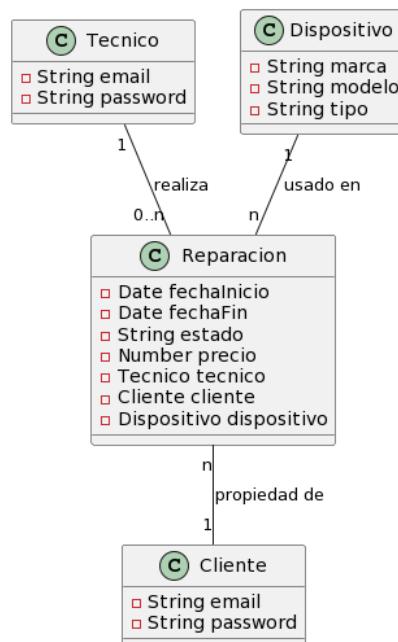


Figura 8.1: Diagrama de clases

8.2. Descripción de las clases

8.2.1. Clase Cliente

Descripción: La clase *Cliente* representa a los individuos o entidades que poseen dispositivos y solicitan servicios de reparación. Esta clase es responsable de almacenar la información de contacto del cliente.

Atributos:

- **email:** La dirección de correo electrónico del cliente.
- **password:** La contraseña para acceder a la cuenta del cliente en el sistema.

Relaciones:

- Cada *Cliente* puede tener varias *Reparaciones* asociadas.
- Un *Cliente* es el propietario de uno o varios *Dispositivos*.

8.2.2. Clase Dispositivo

Descripción: La clase *Dispositivo* contiene la información sobre los aparatos electrónicos o maquinaria que requieren reparación o mantenimiento.

Atributos:

- **marca:** La marca comercial del dispositivo.
- **modelo:** El modelo específico del dispositivo.
- **tipo:** La categoría o tipo de dispositivo (por ejemplo, smartphone, portátil, electrodoméstico).

Relaciones:

- Un *Dispositivo* está asociado con cero o más *Reparaciones*.
 - Cada *Dispositivo* es propiedad de un único *Cliente*.
-

8.2.3. Clase Técnico

Descripción: La clase *Técnico* se encarga de representar a los profesionales que realizan las reparaciones. Incluye información de contacto y autenticación para el sistema.

Atributos:

- **email:** La dirección de correo electrónico del técnico.
- **password:** La contraseña que utiliza el técnico para acceder al sistema.

Relaciones:

- Un *Técnico* puede llevar a cabo varias *Reparaciones*.

8.2.4. Clase Reparación

Descripción: La clase *Reparación* registra cada intervención o servicio de reparación solicitado por un *Cliente* y realizado por un *Técnico* sobre un *Dispositivo*.

Atributos:

- **fechaInicio:** La fecha y hora en que se inició la reparación.
- **fechaFin:** La fecha y hora en que se completó la reparación.
- **estado:** El estado actual de la reparación (por ejemplo, en curso, completada, cancelada).
- **precio:** El costo asociado con la reparación.

Relaciones:

- Una *Reparación* es realizada por un único *Técnico*.
 - Una *Reparación* está asociada con un único *Dispositivo*.
 - Una *Reparación* está solicitada por un único *Cliente*.
-

CAPÍTULO

9

Análisis funcional

En este capítulo se describirá la arquitectura que se ha aplicado para manejar la lógica de negocio, así como estará detallado el análisis del comportamiento de nuestro sistema y las distintas interacciones entre objetos los objetos que lo componen mediante diagramas de secuencia.

El concepto principal para entender la arquitectura aplicada es el de arquitectura *hexagonal*. Como se muestra en la Figura 9.1, obtenida del libro *Clean Architecture* de Robert C. Martin [12], la arquitectura hexagonal es una arquitectura del software en la que se busca es separar el núcleo lógico de la aplicación, dejándolo en el centro totalmente aislado del exterior, del cliente y de otras interacciones.

- La sección *Enterprise Business Rules* se corresponde con la capa de dominio.
- La sección *Application Business Rules* se corresponde con la capa de aplicación.
- Las secciones *Interface adapters* y *Frameworks & Drivers* se corresponden con la capa de infraestructura.

En el modelo de arquitectura *hexagonal* se diferencian las siguientes capas para cada uno de los contextos identificados en nuestro sistema:

- **Capa de dominio.** Corresponde con la capa central de la arquitectura y representa el dominio y las reglas de negocio. Los objetos de dominio deberán de estar en esta capa junto con las interfaces que serán implementadas en las demás capas para comunicarse con ésta. Las entidades dentro del dominio no deben de tener ninguna dependencia

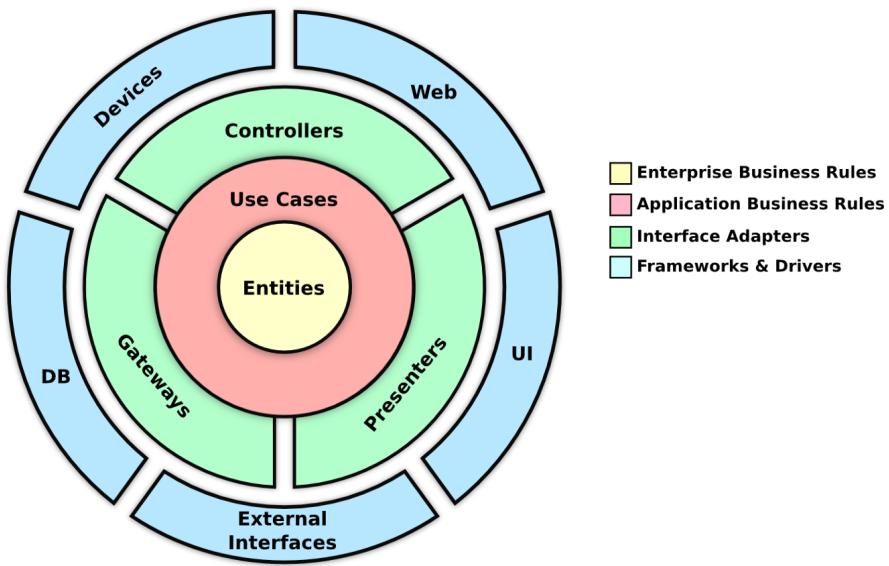


Figura 9.1: Esquema de arquitectura hexagonal

externa de otras capas. Dentro de esta capa se encontrarán entidades que representan los modelos de datos del sistema.

- **Capa de aplicación.** Crea una abstracción entre las entidades de dominio y la lógica de negocio de la aplicación mediante la orquestación de los servicios encargados de realizar los casos de uso. Cada servicio se activará tras recibir la petición del usuario mediante un comando específico, que será recogido por los manejadores de comandos o *handlers* e iniciarán el proceso para cumplir el caso de uso concreto.
- **Capa de infraestructura.** Es la encargada de proveer la implementación de las interfaces definidas en el dominio y de proveer de aquellos servicios externos a las capas interiores. Dentro de esta capa se encontrará la implementación de los repositorios que establecerán la comunicación con la base de datos, de la comunicación con la interfaz de la aplicación mediante los controladores y los servicios auxiliares de las entidades del dominio.

9.1. Descripción del comportamiento

A continuación se muestran los diferentes diagramas de secuencia del proyecto.

9.1.1. Contexto usuario

9.1.1.1. Diagrama de secuencia: creación de un usuario

La Figura 9.2 muestra la secuencia de creación de un usuario. El comando *CreateUserCommand*, es escuchado y manejado por su manejador correspondiente, *CreateUserHandler*. Es el manejador el encargado de llamar a los métodos de la clase **User**, en este caso el método *add()*. Una vez creado el usuario, se persisten los datos en la base de datos a través del método *save()* del repositorio del agregado **User**.

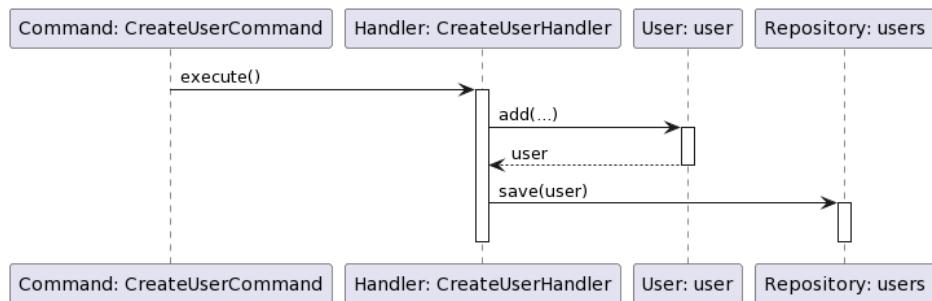


Figura 9.2: Diagrama de secuencia de la creación de un usuario

9.1.1.2. Diagrama de secuencia: cambiar contraseña de un usuario

La Figura 9.3 muestra la secuencia de edición de un usuario. El comando *PasswordChangeCommand*, es escuchado y manejado por su manejador correspondiente, *PasswordChangeHandler*. Es el manejador el encargado de llamar al método *find()* del repositorio para devolver el usuario en cuestión y a continuación, llamar al método *updatePassword()* donde se autentica al usuario también. Finalmente, se persisten los datos en la base de datos a través del método *save()* del repositorio del agregado **User**.

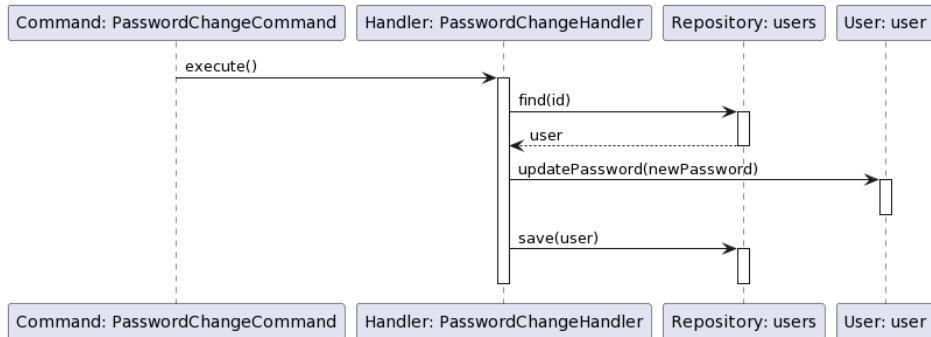


Figura 9.3: Diagrama de secuencia de la edición de un usuario

9.1.1.3. Diagrama de secuencia: eliminación de un usuario

La Figura 9.4 muestra la secuencia de eliminación de un usuario. El comando *DeleteUserCommand*, es escuchado y manejado por su manejador correspondiente, *DeleteUserHandler*. Es el manejador el encargado de llamar al método *find()* del repositorio para devolver el usuario en cuestión y a los métodos de la clase *User*, en este caso al método *delete()*. Finalmente, se eliminan los datos de la base de datos a través del método *delete()* del repositorio del agregado *User*.

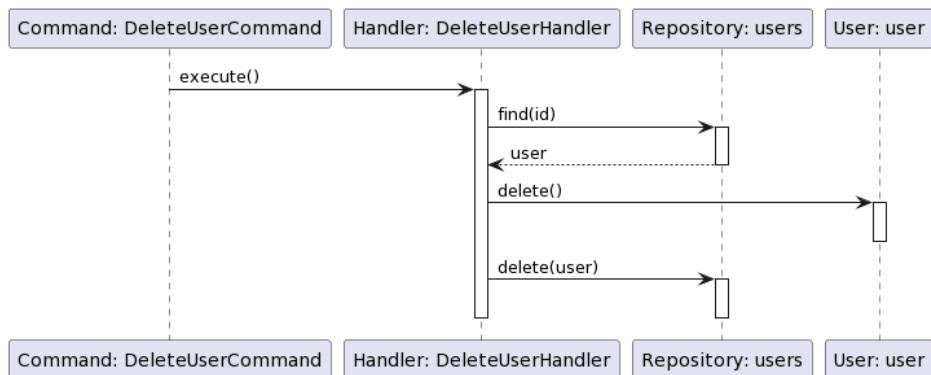


Figura 9.4: Diagrama de secuencia de la eliminación de un usuario

9.1.1.4. Diagrama de secuencia: obtención de un usuario

La Figura 9.5 muestra la secuencia de obtención de un usuario. La consulta *GetUserQuery*, es escuchada y manejada por su manejador correspondiente, *GetUserHandler*. Es el manejador el encargado de llamar al método *find()* del servicio *finder* para devolver al usuario en cuestión

a partir de su identificador.

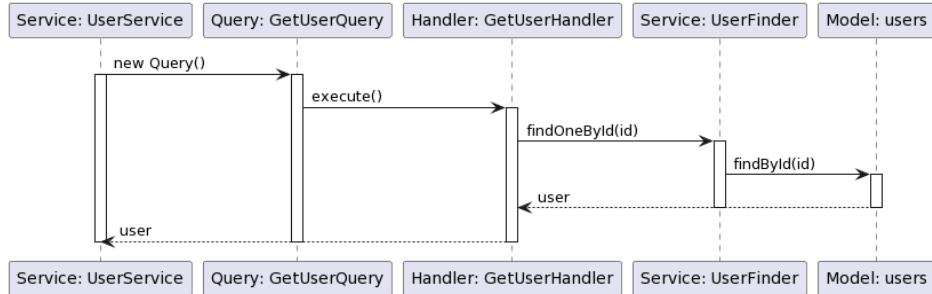


Figura 9.5: Diagrama de secuencia de la obtención de un usuario

9.1.1.5. Diagrama de secuencia: obtención de todos los usuarios

La Figura 9.6 muestra la secuencia de obtención de todos los usuarios. La consulta *GetUsersQuery*, es escuchada y manejada por su manejador correspondiente, *GetUsersHandler*. Es el manejador el encargado de llamar al método *findAll()* del servicio *finder* para devolver a todos los usuarios registrados en el sistema.

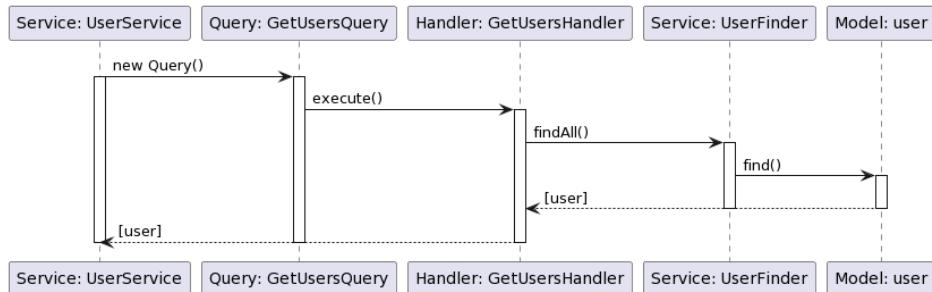


Figura 9.6: Diagrama de secuencia de la obtención de todos los usuarios

9.1.2. Contexto dispositivos

9.1.2.1. Diagrama de secuencia: creación de un dispositivo

La Figura 9.7 muestra la secuencia de creación de un dispositivo. El comando *CreateDeviceCommand*, es escuchado y manejado por su manejador correspondiente, *CreateDeviceHandler*. Es el manejador el encargado de llamar a los métodos de la clase **Device**, en este caso el método *add()*. Una vez creado el dispositivo, se persisten los datos en la base de datos a través del método *save()* del repositorio del agregado **Device**.

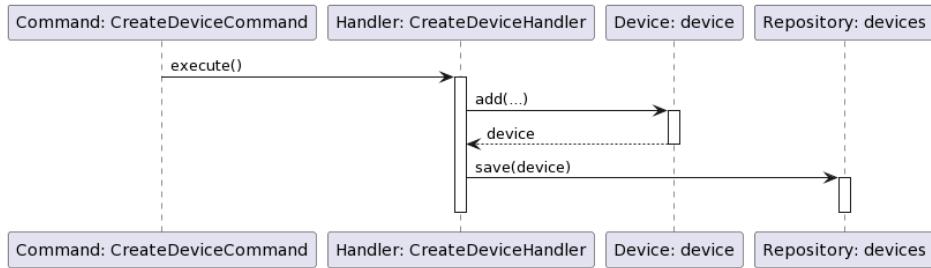


Figura 9.7: Diagrama de secuencia de la creación de un dispositivo

9.1.2.2. Diagrama de secuencia: eliminación de un dispositivo

La Figura 9.8 muestra la secuencia de eliminación de un dispositivo. El comando *DeleteDeviceCommand*, es escuchado y manejado por su manejador correspondiente, *DeleteDeviceHandler*. Es el manejador el encargado de llamar al método *find()* del repositorio para devolver el dispositivo en cuestión y a los métodos de la clase **Device**, en este caso al método *delete()*. Finalmente, se eliminan los datos de la base de datos a través del método *delete()* del repositorio del agregado **Device**.

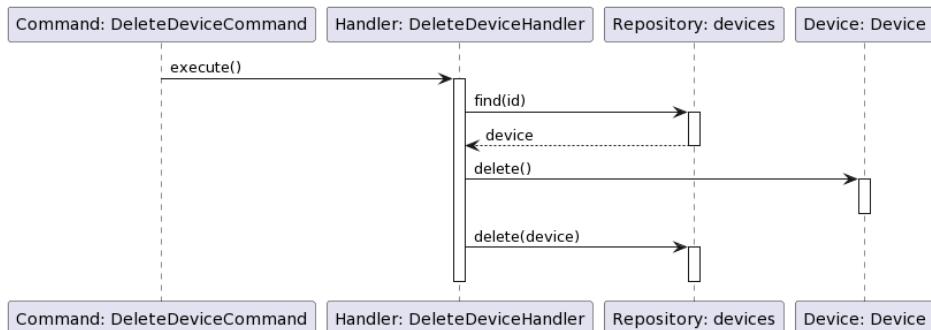


Figura 9.8: Diagrama de secuencia de la eliminación de un dispositivo

9.1.2.3. Diagrama de secuencia: obtención de un dispositivo

La Figura 9.9 muestra la secuencia de obtención de un dispositivo. La consulta *GetDeviceQuery*, es escuchada y manejada por su manejador correspondiente, *GetDeviceHandler*. Es el manejador el encargado de llamar al método *find()* del servicio *finder* para devolver al dispositivo en cuestión a partir de su identificador.

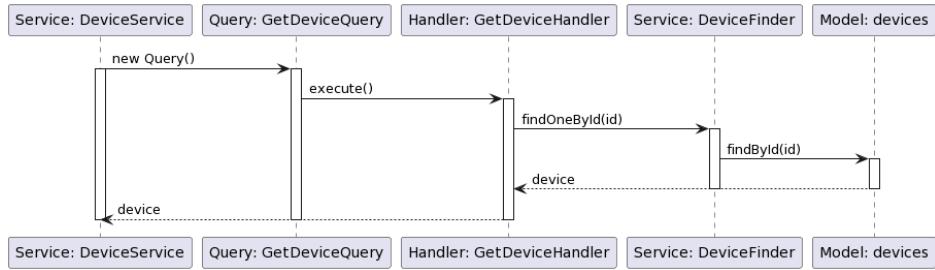


Figura 9.9: Diagrama de secuencia de la obtención de un dispositivo

9.1.2.4. Diagrama de secuencia: obtención de todos los dispositivos

La Figura 9.10 muestra la secuencia de obtención de todos los dispositivos. La consulta `GetDevicesQuery`, es escuchada y manejada por su manejador correspondiente, `GetDevicesHandler`. Es el manejador el encargado de llamar al método `findAll()` del servicio `finder` para devolver a todos los dispositivos registrados en el sistema.

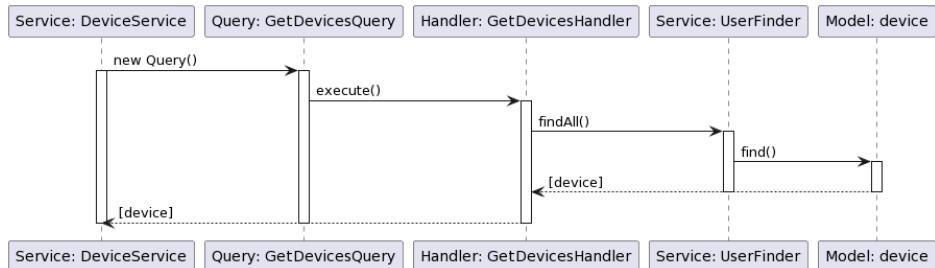


Figura 9.10: Diagrama de secuencia de la obtención de todos los dispositivos

9.1.3. Contexto reparaciones

9.1.3.1. Diagrama de secuencia: creación de una reparación

La Figura 9.11 muestra la secuencia de creación de una reparación. El comando `CreateWorkOrderCommand`, es escuchado y manejado por su manejador correspondiente, `CreateWorkOrderHandler`. Es el manejador el encargado de llamar a los métodos de la clase `WorkOrder`, en este caso el método `add()`. Una vez creada la reparación, se persisten los datos en la base de datos a través del método `save()` del repositorio del agregado `WorkOrder`.

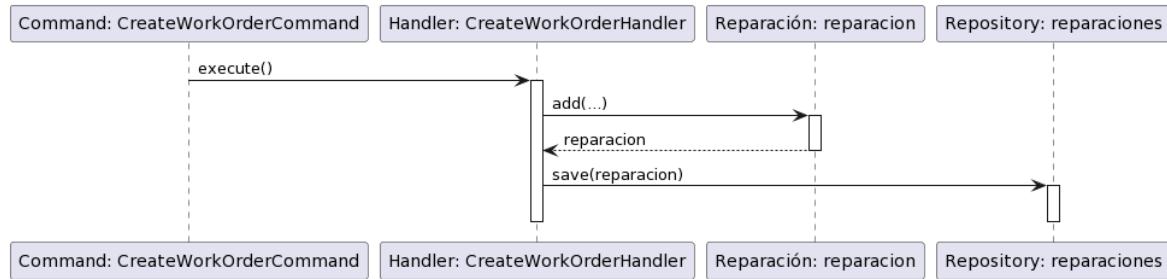


Figura 9.11: Diagrama de secuencia de la creación de una reparación

9.1.3.2. Diagrama de secuencia: edición de una reparación

La Figura 9.12 muestra la secuencia de edición de una reparación. El comando `UpdateWorkOrderCommand`, es escuchado y manejado por su manejador correspondiente, `UpdateWorkOrderHandler`. Es el manejador el encargado de llamar al método `find()` del repositorio para devolver la reparación en cuestión y a continuación, al método, `update(...)`. Finalmente, se persisten los datos en la base de datos a través del método `save()` del repositorio del agregado `WorkOrder`.

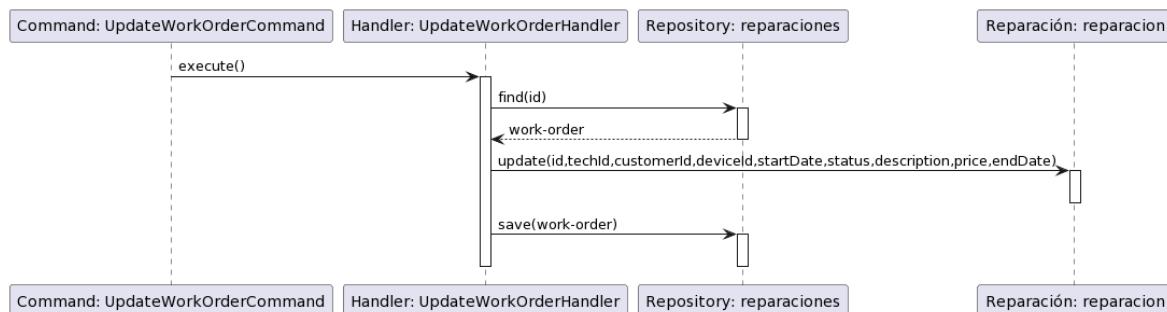


Figura 9.12: Diagrama de secuencia de la edición de una reparación

9.1.3.3. Diagrama de secuencia: eliminación de una reparación

La Figura 9.13 muestra la secuencia de eliminación de una reparación. El comando `DeleteWorkOrderCommand`, es escuchado y manejado por su manejador correspondiente, `DeleteWorkOrderHandler`. Es el manejador el encargado de llamar al método `find()` del repositorio para devolver la reparación en cuestión y a los métodos de la clase `WorkOrder`, en este caso al método `delete()`. Finalmente, se eliminan los datos de la base de datos a través del método `delete()` del repositorio del agregado `WorkOrder`.

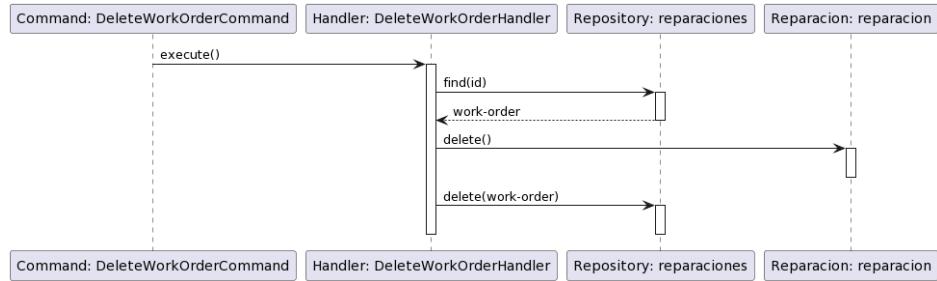


Figura 9.13: Diagrama de secuencia de la eliminación de una reparación

9.1.3.4. Diagrama de secuencia: obtención de una reparación

La Figura 9.14 muestra la secuencia de obtención de una reparación. La consulta *GetWorkOrderQuery*, es escuchada y manejada por su manejador correspondiente, *GetWorkOrderHandler*. Es el manejador el encargado de llamar al método *find()* del servicio *finder* para devolver a la reparación en cuestión a partir de su identificador.

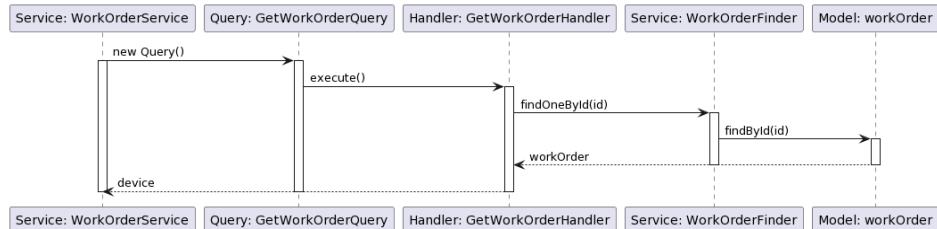


Figura 9.14: Diagrama de secuencia de la obtención de una reparación

9.1.3.5. Diagrama de secuencia: obtención de todas las reparaciones

La Figura 9.15 muestra la secuencia de obtención de todas las reparaciones. La consulta *GetWorkOrdersQuery*, es escuchada y manejada por su manejador correspondiente, *GetWorkOrdersHandler*. Es el manejador el encargado de llamar al método *findAll()* del servicio *finder* para devolver a todas las reparaciones registradas en el sistema.

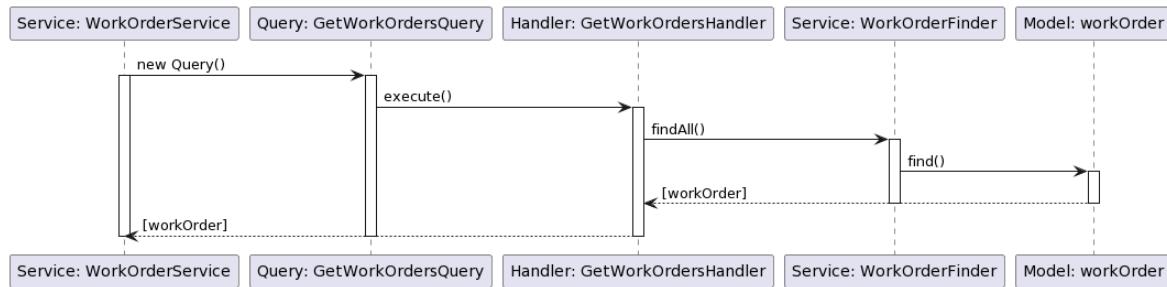


Figura 9.15: Diagrama de secuencia de la obtención de todas las reparaciones

10.1. Modelado

Para modelar el dominio de nuestro sistema y la serie de eventos que se darán en él, se ha seguido una metodología denominada *Event Modeling* [6], la cual cuenta con una serie de técnicas de modelado basadas en eventos, interfaces y entidades que interactúan entre sí cuando algo en el dominio del problema sufre un cambio de estado. Esta técnica de modelado está directamente relacionada con patrones de diseño como *Domain-Driven Design*, *Event-Sourcing* o los modelos de escritura/lectura que plantea *Command-Query Responsibility Segregation* (CQRS) a la hora de la persistencia de datos en el sistema.

Para representar la información modelada se han agrupado en diferentes paquetes, correspondiendo con cada uno de los contextos identificados del dominio de nuestra aplicación.

Los datos del dominio corresponden con la información de los agregados, es decir, los *Value Objects* y los métodos que los componen. Para ello se realizará una descripción de los atributos de cada raíz del agregado.

Las fases a seguir planteadas por esta metodología (*Event Modeling*) son los siguientes:

- **Identificar los eventos.** Todos los miembros del equipo tratan de tener una visión de lo que será el sistema y proponen los eventos que sucederán en él.
- **Línea temporal.** Una vez se hayan recogido todos los eventos del sistema se ordenarán

cronológicamente.

- **Storyboard.** Una vez estén los eventos ordenados cronológicamente, se piensan los posibles casos de uso para el usuario con el sistema y se definen los *wireframes* pertinentes. Los *wireframes* son bocetos de las interfaces que sirven para facilitar de manera visual al equipo cuales serán los posibles casos de uso que existirán en el sistema que disparen algún evento.
- **Consultas y comandos.** Aquellas que se lanzarán cuando el usuario interactúe con el sistema.
- **Diseño de proyecciones.** Se identifica la información que será requerida para el usuario cuando un evento suceda.

A continuación, en las figuras 10.1, 10.2, 10.3, 10.4 y 10.5 se muestran varias partes del modelado de eventos y diseño del sistema de reparaciones que se ha llevado a cabo para este *Trabajo de Fin de Grado*. Para su mejor comprensión se va a explicar lo que representa cada tarjeta según el color en el diagrama:

- **Tarjetas de color rosa.** Estas tarjetas representan los datos enviados desde la interfaz de usuario, comúnmente a través de una aplicación web, hacia la interfaz de programación de aplicaciones (API). Suelen ser formularios llenados por usuarios que inician interacciones y procesos dentro del sistema. Por ejemplo, un formulario de solicitud de reparación que el usuario rellena para iniciar un proceso de servicio técnico.
- **Tarjetas de color azul.** Estas tarjetas indican una acción que se quiere llevar a cabo. En el contexto del event modeling, describen la función o método que se invoca en respuesta a un evento o una solicitud, como por ejemplo, "Registrar Usuario" o "Crear Orden de Trabajo". Representan la lógica de negocio que procesa las entradas recibidas.
- **Tarjetas de color naranja.** Estas tarjetas representan eventos que son generados y luego almacenados en bases de datos. Un evento puede ser algo como "Usuario Registrado" o "Orden de Trabajo Creada". En un sistema basado en eventos, estos actúan como registros inmutables de que algo ha sucedido en el sistema y pueden desencadenar flujos de trabajo adicionales.

- **Tarjetas de color verdes.** Estas tarjetas simbolizan las vistas o representaciones de la información que se generan tras procesar los datos enviados. Por ejemplo, después de que un usuario se registra, la vista generada podría ser la página de perfil del usuario o un dashboard actualizado.
- **Tarjetas de color roja.** Estas tarjetas se utilizan para mostrar notificaciones externas que se originan desde el sistema. Podrían ser correos electrónicos enviados al usuario, como una confirmación de registro o una notificación de que su orden de trabajo ha sido procesada.

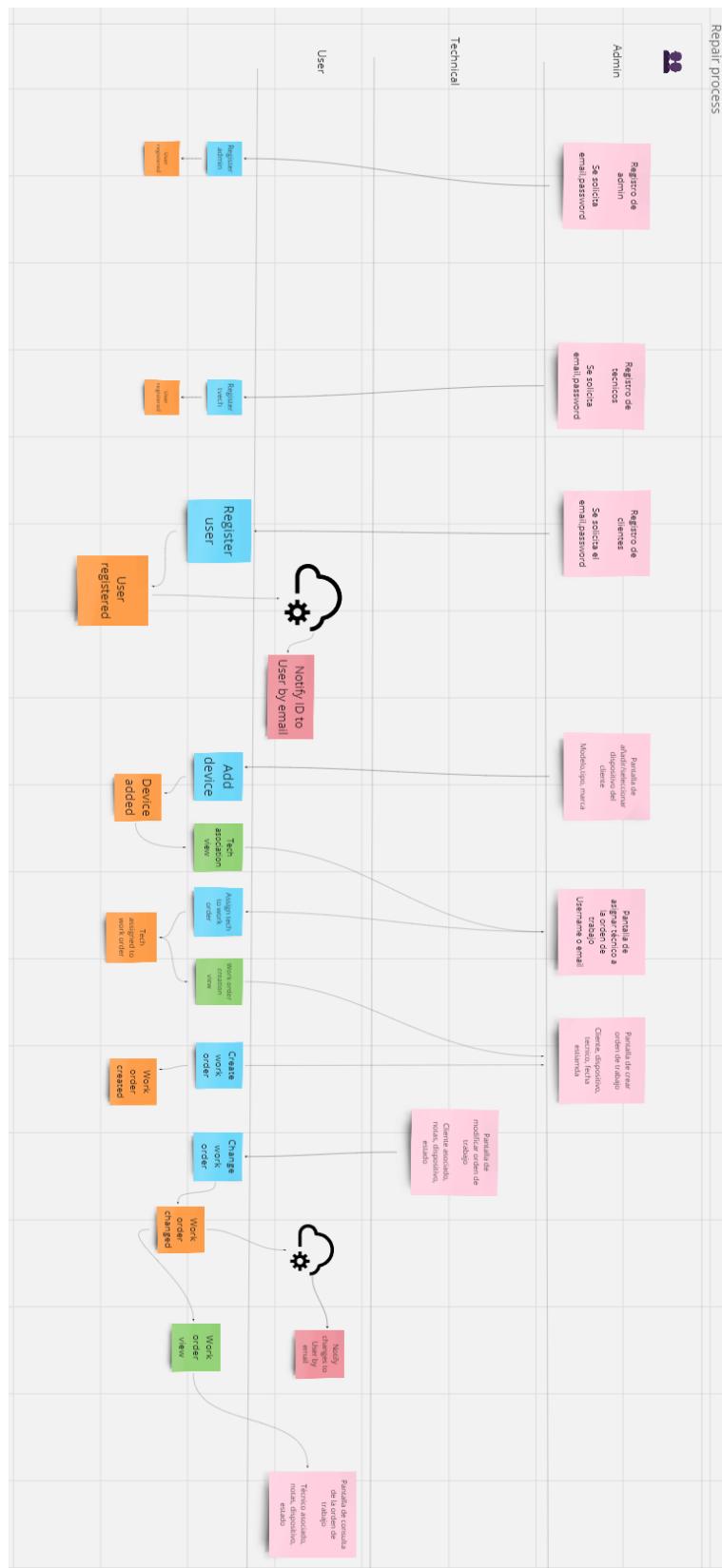


Figura 10.1: Diseño y modelado del flujo ideal del negocio con *Event Modeling*

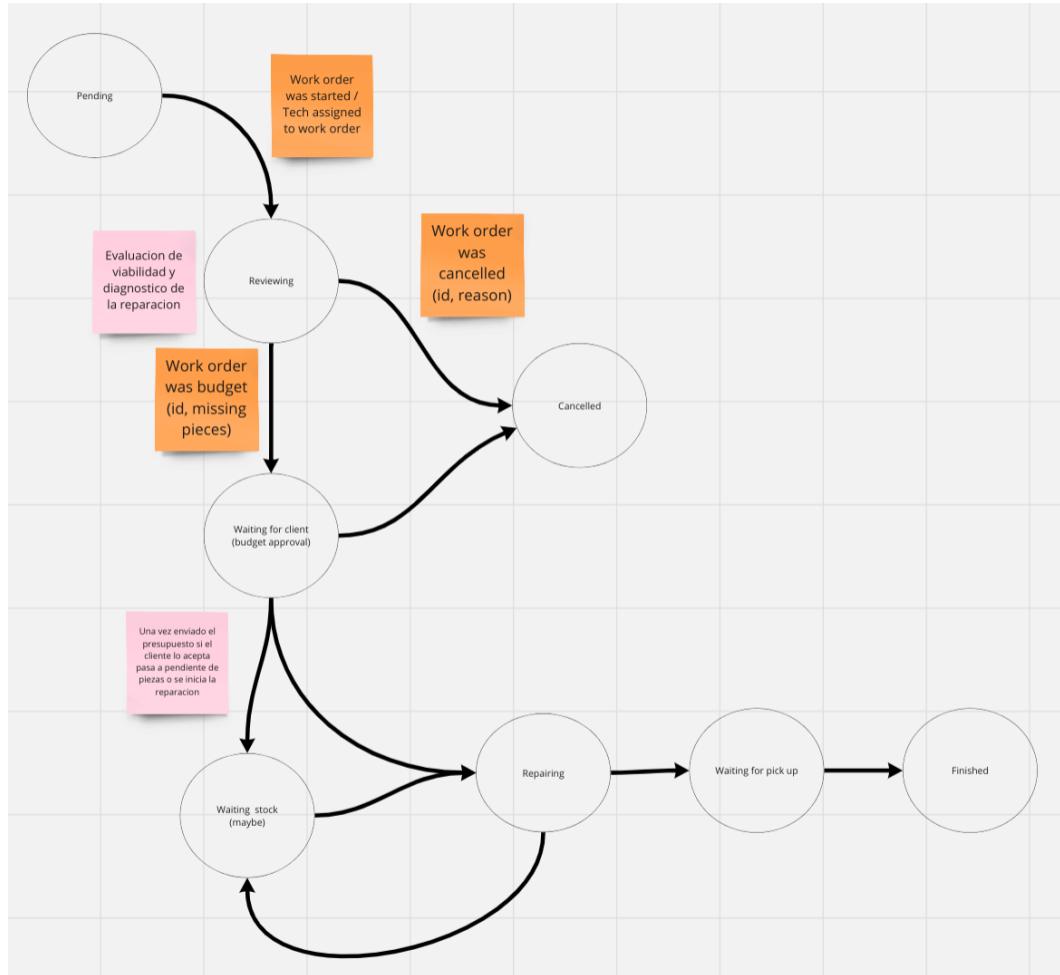


Figura 10.2: Diseño y modelado de los distintos estados de reparaciones

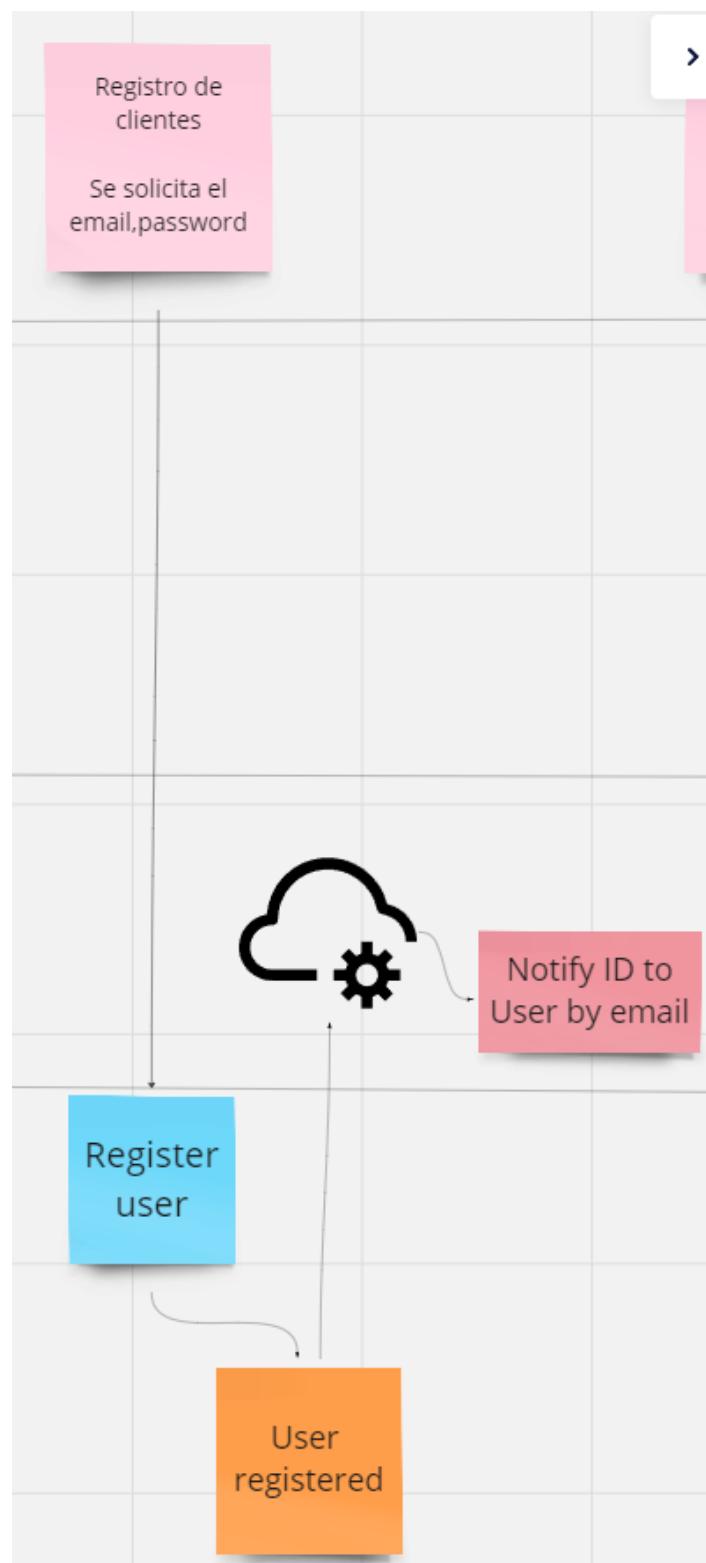


Figura 10.3: Diseño y modelado del registro de usuarios con *Event Modeling*



Figura 10.4: Diseño y modelado de crear reparación con *Event Modeling*

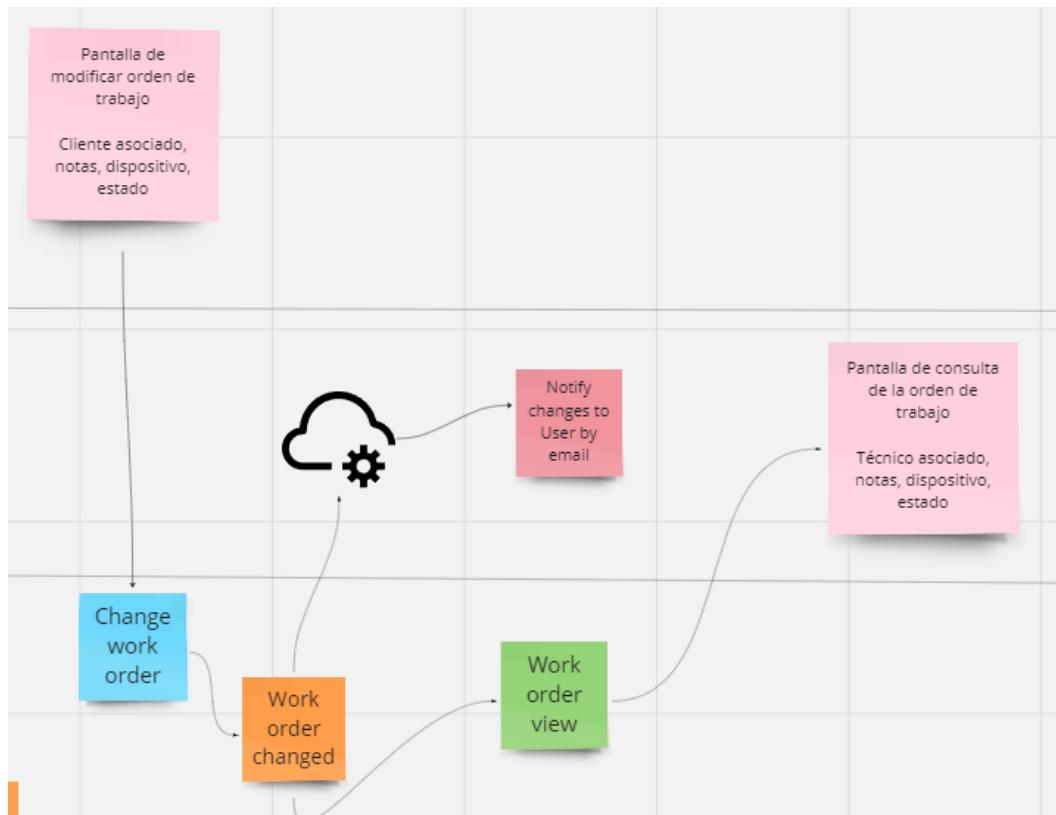


Figura 10.5: Diseño y modelado de actualizar una reparación con *Event Modeling*

10.2. Arquitectura del sistema

Con el objetivo de ir conociendo cada vez más el dominio del problema a resolver conforme el desarrollo el proyecto, el cual está basado en iteraciones, se ha decidido seguir el enfoque presentado por el conocido concepto de diseño orientado al dominio (*Domain-Driven Design*), ya que se adecuaba de mejor manera a la resolución de nuestro problema.

Vaugh Vernon definió el diseño orientado al dominio como una metodología para el desarrollo de sistemas complejos donde el centro de atención recae en la esquematización de las actividades, eventos, tareas e información que conforma el dominio del problema. El punto principal de esta metodología es entender el dominio del problema para poder crear un modelo abstracto que pueda ser implementado mediante cualquier tecnología [13].

El problema más común durante el desarrollo de un proyecto tecnológico es la falta de entendimiento entre las personas que diseñan y construyen ese sistema y las personas que necesitan el sistema. Esta metodología pretende solucionar esto mediante el establecimiento del **lenguaje ubicuo**, un lenguaje compartido entre estas personas que permita expresar las necesidades del usuario y poder discutir acerca de cómo solucionar el problema de modo que todos los involucrados se entiendan entre sí.

Domain-Driven Design se refiere con dominio a las reglas de negocio que vienen dadas por los posibles casos de uso que tendrá el sistema a lo largo del tiempo, lo cual generará los denominados Eventos de dominio. Realmente estos últimos tienen por qué perteneces a la arquitectura estricta que predicen los principios del diseño basado en el dominio, pero sí forma parte de un tipo de metodología denominada *Event sourcing* la cual se integra de manera más que natural a una arquitectura basada en *Domain-Driven Design*, también conocida como arquitectura hexagonal.

En la Figura 10.6 obtenida del libro *Implementing Domain-Drive Design* [13] se muestra un esquema con los principales componentes que hay en *Domain-Driven Design*.

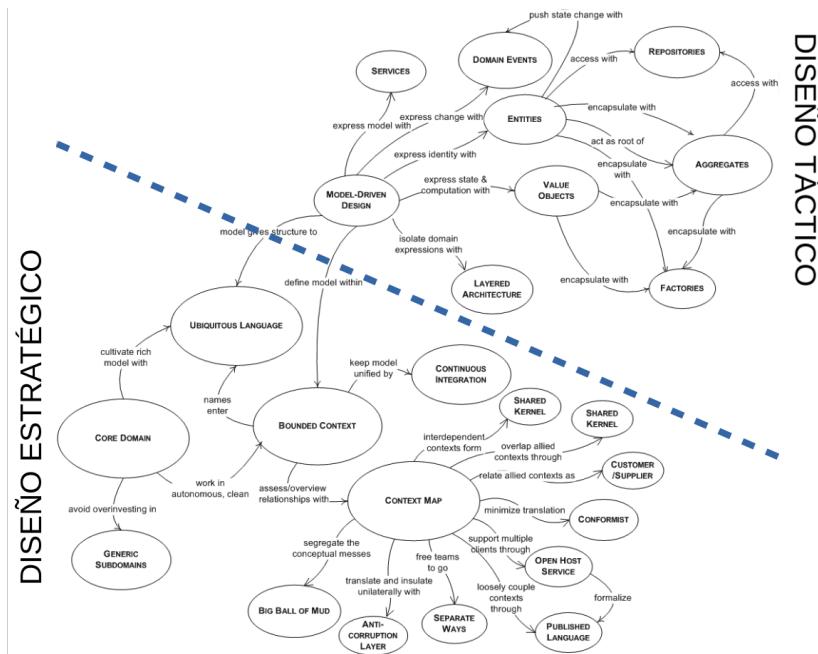


Figura 10.6: Esquema *Domain-Driven Design*

10.2.1. Diagrama de despliegue

En el siguiente diagrama se observa la arquitectura del sistema y la comunicación *frontend-backend*.

Los datos recogidos por la aplicación web entran en nuestro servidor por una ruta del servidor *backend* (*endpoint*) que pertenece a un controlador en nuestro servidor.

Una vez llegan al controlador, si es una operación de escritura el *CommandBus* gestionará los datos mediante el *CommandHandler* y persistirá los datos en la *EventStore* y en sus proyecciones respectivas(MongoDB y Redis).

En caso de que sea una operación de lectura, el *QueryBus* es quién llama al *QueryHandler* que es quien consulta en las proyecciones del *EventStore* los datos requeridos por la solicitud.

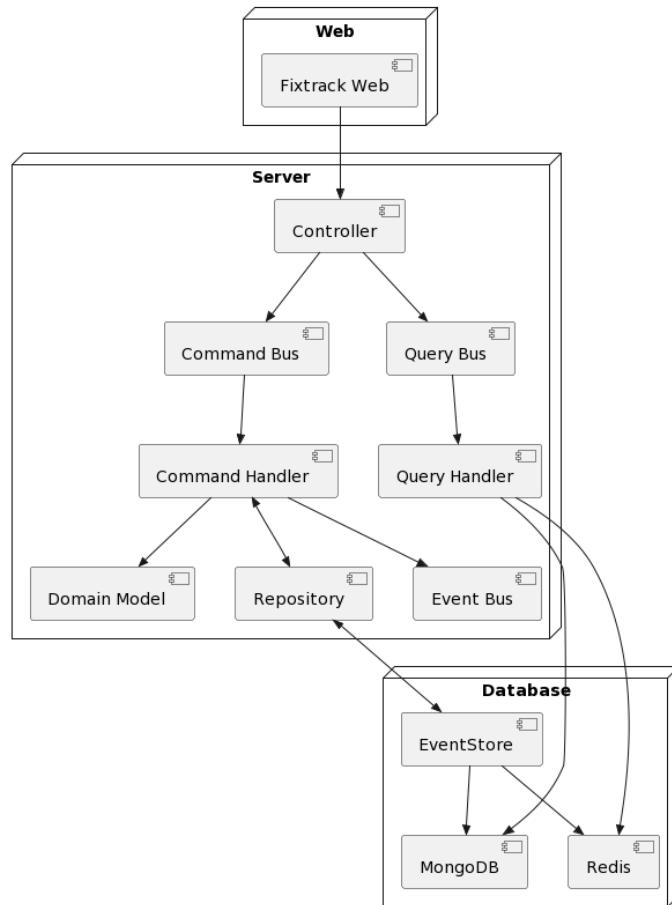


Figura 10.7: Diagrama de despliegue

10.3. Datos

En la arquitectura de nuestro proyecto se implementa el patrón CQRS, que implica una estrategia distinta para el manejo de los datos en operaciones de escritura (comandos) frente a las de lectura (consultas). Esta separación nos permite optimizar ambas operaciones de manera independiente, mejorando así el rendimiento y la escalabilidad del sistema.

En la mayoría de los contextos de dominio dentro de nuestro sistema, mantenemos esquemas de datos distintos para las operaciones de escritura y lectura. Por ejemplo, mientras que los comandos pueden requerir estructuras de datos más normalizadas para evitar redundancias y garantizar la integridad referencial, las consultas pueden beneficiarse de esquemas desnormalizados que agilizan la recuperación de los datos. Sin embargo, esta separación no se aplica de la misma manera en el contexto de los datos de los usuarios debido a las restricciones del RGPD.

El RGPD estipula que los datos personales deben ser eliminables a solicitud del usuario, asegurando así su derecho al olvido. En sistemas que utilizan bases de datos basadas en eventos, como es el caso de nuestro proyecto, eliminar registros es una operación compleja, ya que las bases de datos de eventos están diseñadas para ser inmutables para preservar el rastro completo de las actividades del sistema. Por lo tanto, en lugar de borrar eventos, debemos implementar estrategias alternativas que cumplan con el RGPD, como la anonimización de los datos, para desvincular los eventos de los datos personales del usuario.

Para abordar esta obligación del RGPD sin contravenir los principios de las bases de datos basadas en eventos, hemos decidido tratar los datos de los usuarios de manera diferente en nuestro diseño de datos. Cuando un usuario ejerce su derecho al olvido, en lugar de eliminar los eventos, los anonimizamos para retirar toda la información personal, cumpliendo así con la legislación sin comprometer la integridad de nuestra base de datos de eventos.

CAPÍTULO

11

Diseño de la interfaz

El sistema ofrecerá tres interfaces de usuario para cada una de las partes involucradas en una reparación de un dispositivo electrónico (administrador, técnico y cliente).

11.1. Características generales de la interfaz

Desde un principio se le dio prioridad a dos objetivos primordiales: que la interfaz fuese usable e intuitiva, y que la experiencia de usuario fuese idónea. Para lograr estas metas, se han tenido en cuenta las siguientes características para el diseño de la interfaz.

- No cargar de información innecesaria pantallas en las que el usuario pueda llegar a confusión a la hora de realizar cualquier acción, focalizando la atención del usuario en la información importante.
- Formularios simples: los formularios deben de describir explícitamente cual es su propósito, indicando que campos son obligatorios y proporcionando *feedback* al usuario tanto en caso de error como de éxito.
- Los elementos de la interfaz deben de ser visibles, se elegirán colores que contrasten la información y consigan una cómoda legibilidad de la aplicación.
- Priorizar que la experiencia de usuario sea limpia y tenga un sentido tanto cronológico como ordenado.

11.2. Interfaces de la aplicación de administrador

11.2.1. Interfaz de inicio

En la figura 11.1 se muestra la pantalla de la página de inicio.

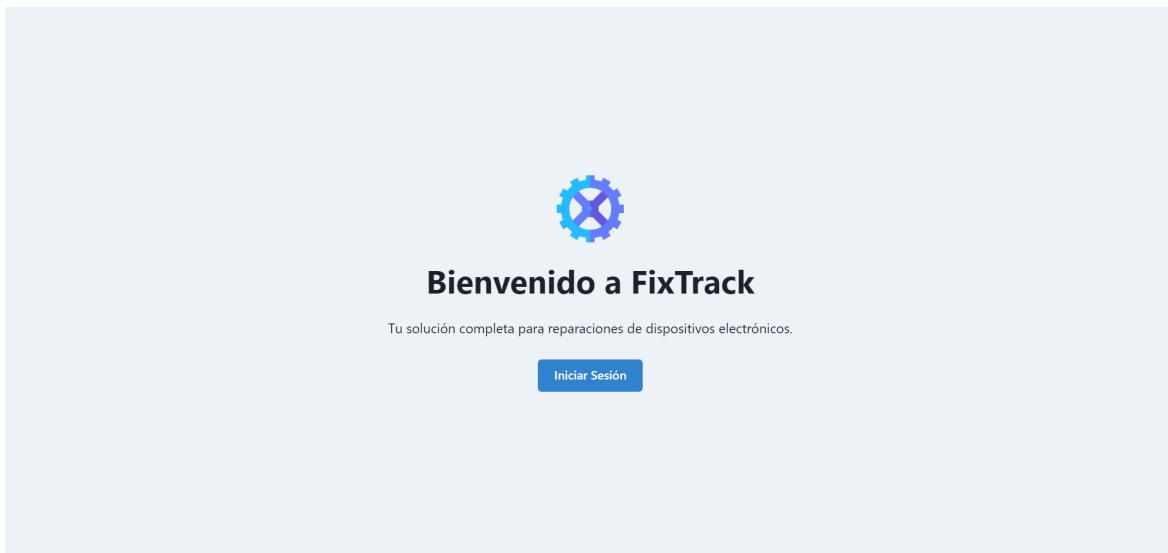


Figura 11.1: Interfaz de la página de inicio

11.2.2. Interfaz de inicio de sesión

En la figura 11.2 se muestra la pantalla de la página de inicio de sesión.

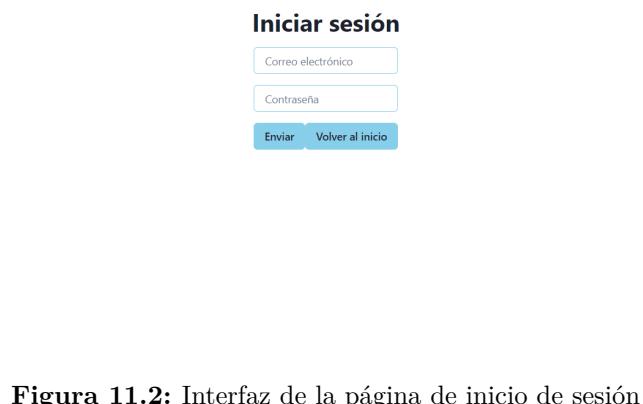


Figura 11.2: Interfaz de la página de inicio de sesión

11.2.3. Interfaz de *Home* del administrador

En la figura 11.3 se muestra la pantalla de inicio de la aplicación donde se mostrarán un listado de estadísticas relacionadas con el negocio.



Figura 11.3: Interfaz de página principal del administrador

11.2.4. Interfaz de página de reparaciones del administrador

11.2.4.1. Listado de reparaciones

En la figura 11.4 se muestra la pantalla de listado de reparaciones.

Lista de órdenes de trabajo		
	Crear orden de trabajo	Buscar
#1 - 5/5/2021 - 5/5/2021	CANCELLED	100€ ▾
#2 - 18/9/2023	REVIEWING	5€ ▾
#3 - 22/9/2023	PENDING	5€ ▾
#4 - 22/9/2023	PENDING	50€ ▾
#5 - 22/9/2023	PENDING	5€ ▾
#6 - 22/9/2023	PENDING	45€ ▾
#7 - 22/9/2023	PENDING	12€ ▾

Figura 11.4: Interfaz de página de listado de reparaciones

11.2.4.2. Información de una reparación

En la figura 11.5 se muestra la pantalla de información de una reparación.

The screenshot shows a user interface for managing repair orders. At the top, there are navigation icons for Home, Cart, User, Device, and Logout. Below that is a header with the title "Lista de órdenes de trabajo". A green button labeled "Crear orden de trabajo" is on the left, and a search bar with placeholder "Buscar" and a dropdown for "10 por página" are on the right. The main area displays a table of repair orders:

Orden	Fecha	Estado	Precio
#1 - 5/5/2021 - 5/5/2021		CANCELLED	100€ ^
Cliente: cliente123@gmail.com			
Dispositivo: PHONE-APPLE-IPHONE 14			
Técnico: tecnico123@gmail.com			
Descripción: The device is broken			
		<input type="button" value="Actualizar"/>	<input type="button" value="Eliminar"/>
#2 - 18/9/2023		REVIEWING	5€ ▾
#3 - 22/9/2023		PENDING	5€ ▾
#4 - 22/9/2023		PENDING	50€ ▾

Figura 11.5: Interfaz de página de información de una reparación

11.2.4.3. Crear una reparación

En la figura 11.6 se muestra la pantalla de crear una reparación.

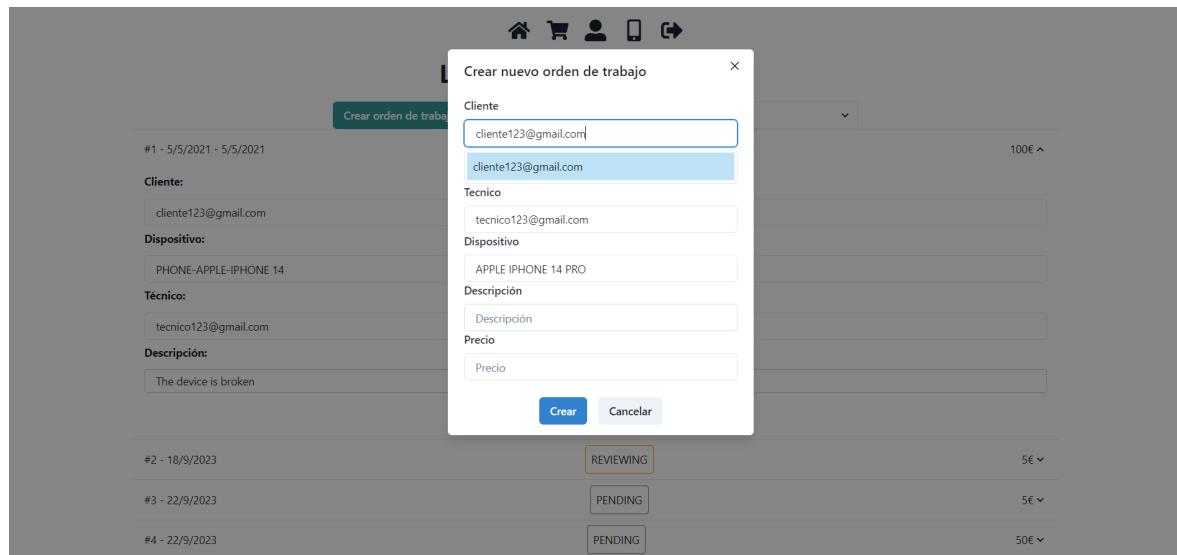


Figura 11.6: Interfaz de página de crear una reparación

11.2.5. Interfaz de página de dispositivos del administrador

11.2.5.1. Listado de dispositivos

En la figura 11.7 se muestra la pantalla de listado de dispositivos.

The screenshot shows a user interface for managing devices. At the top, there are icons for home, shopping cart, user profile, and navigation. Below that is a title 'Lista de dispositivos' (Device List) with a subtitle 'Dispositivos registrados' (Registered Devices). A search bar and a dropdown for page size (set to 10) are also at the top. The main area is a table with the following data:

ID	TIPO	MARCA	MODELO	ACCIONES
1	PHONE	APPLE	IPHONE 14 PRO	<button>Eliminar</button>
2	PHONE	APPLE	IPHONE 14	<button>Eliminar</button>
3	PHONE	SAMSUNG	GALAXY S22	<button>Eliminar</button>
4	LAPTOP	APPLE	MACBOOK AIR 2020	<button>Eliminar</button>
5	TABLET	APPLE	IPAD AIR PRO 2020	<button>Eliminar</button>
6	PHONE	LG	G7 PLUS	<button>Eliminar</button>
7	PHONE	SAMSUNG	S22 ULTRA	<button>Eliminar</button>
8	PHONE	SAMSUNG	GALAXY S22 ULTRA	<button>Eliminar</button>

Figura 11.7: Interfaz de página de listado de dispositivos

11.2.5.2. Crear un dispositivo

En la figura 11.8 se muestra la pantalla de crear un dispositivo.

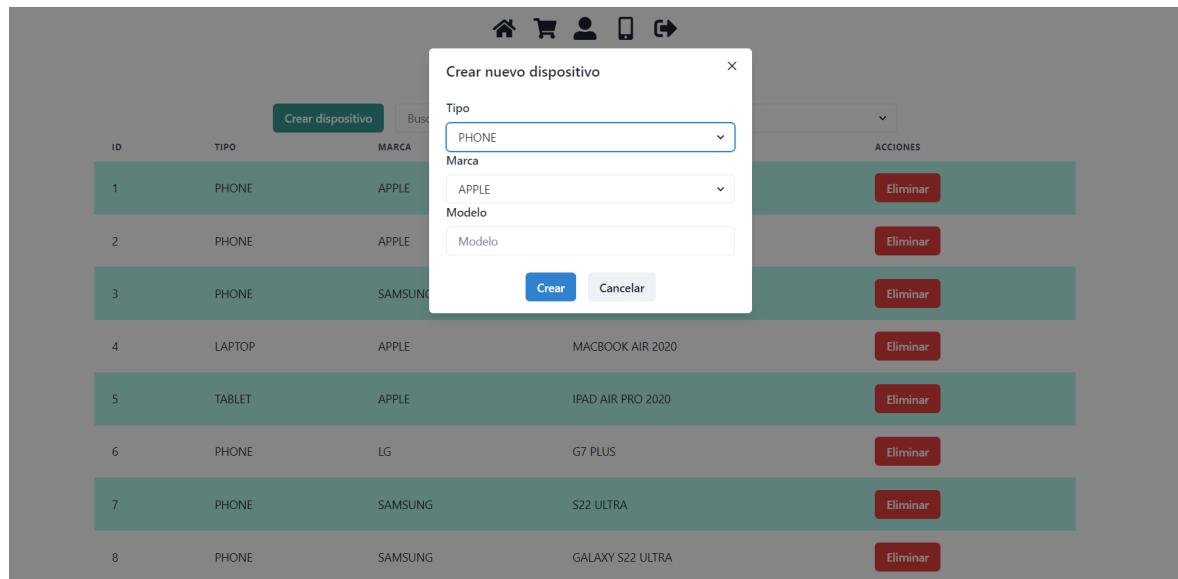


Figura 11.8: Interfaz de página de crear un dispositivo

11.2.6. Interfaz de página de usuarios del administrador

11.2.6.1. Listado de usuarios

En la figura 11.9 se muestra la pantalla de listado de usuarios.

ID	EMAIL	ROLE	ACCIONES
1	admin123@gmail.com	ADMIN	<button>Eliminar</button>
2	cliente123@gmail.com	CLIENTE	<button>Eliminar</button>
3	tecnico123@gmail.com	TECNICO	<button>Eliminar</button>
4	ejemplo123@gmail.com	CLIENTE	<button>Eliminar</button>
5	ejemplito12@gmail.com	CLIENTE	<button>Eliminar</button>
6	juanjo@gmail.com	ADMIN	<button>Eliminar</button>

[Anterior](#) Página 1 de 1 [Siguiente](#)

Figura 11.9: Interfaz de página de listado de usuarios

11.2.6.2. Crear un usuario

En la figura 11.10 se muestra la pantalla de crear un usuario. Y en la figura 11.11 se muestra la notificación enviada al usuario que desea registrarse en el sistema.

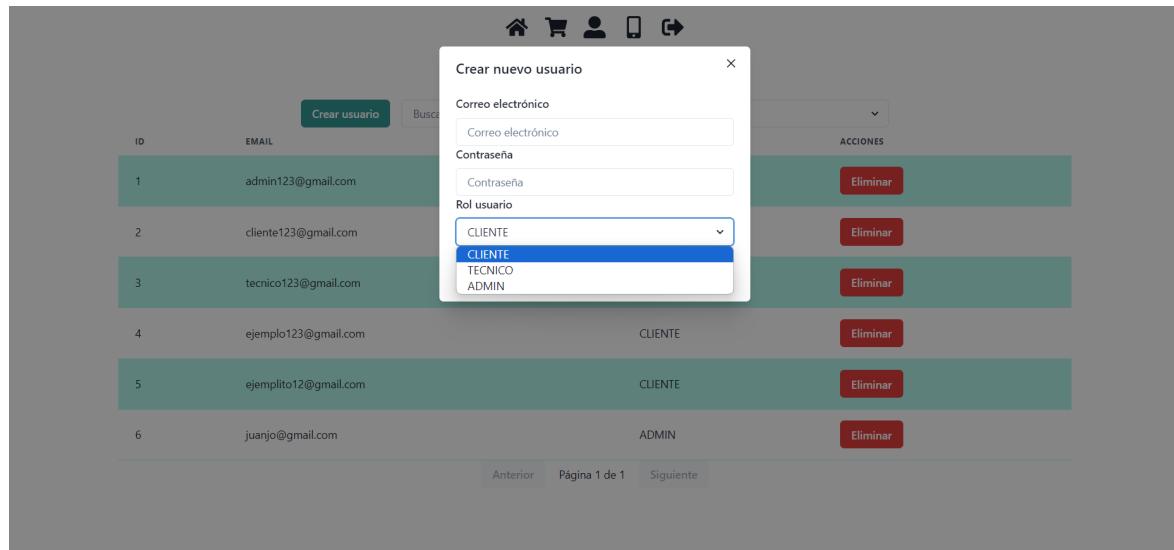
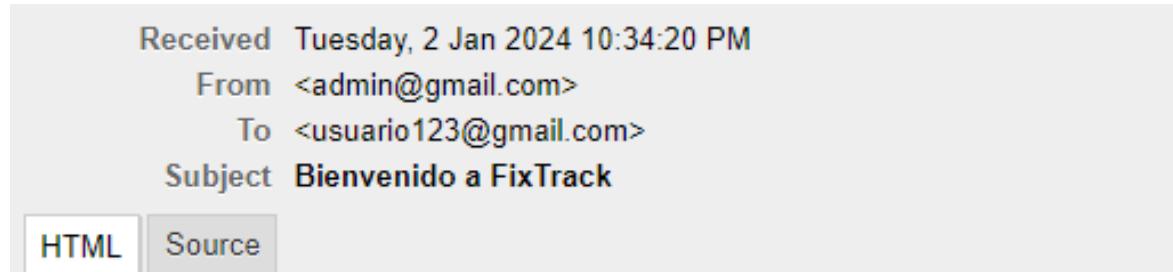


Figura 11.10: Interfaz de página de crear un usuario



Bienvenido a FixTrack

Gracias por registrarte en FixTrack

Tus contraseña generada aleatoriamente es esta:

usuario123

Cambia tu contraseña al iniciar sesión con el siguiente enlace:

[Clíckeme](#)

Si no has creado una cuenta en FixTrack, ignora este correo.

Saludos,

El equipo de FixTrack

Figura 11.11: Interfaz de la notificación de alta de usuario

11.3. Interfaces de la aplicación de técnico

11.3.1. Interfaz de inicio

En la figura 11.12 se muestra la pantalla de la página de inicio.



Figura 11.12: Interfaz de la página de inicio

11.3.2. Interfaz de inicio de sesión

En la figura 11.13 se muestra la pantalla de la página de inicio de sesión.

La imagen es una captura de pantalla de un formulario de inicio de sesión. El título "Iniciar sesión" está centrado en la parte superior. Abajo de él hay dos cuadros de texto: el primero para "Correo electrónico" y el segundo para "Contraseña". A continuación, hay un botón azul con el texto "Enviar" y un enlace azul "Volver al inicio".

Iniciar sesión

Correo electrónico

Contraseña

Enviar Volver al inicio

Figura 11.13: Interfaz de la página de inicio de sesión

11.3.3. Interfaz de Home del técnico

En la figura 11.14 se muestra la pantalla de inicio de la aplicación donde se mostrarán un listado de estadísticas relacionadas con el técnico y el negocio.



Figura 11.14: Interfaz de página principal del técnico

11.3.4. Interfaz de página de reparaciones del técnico

11.3.4.1. Listado de reparaciones

En la figura 11.15 se muestra la pantalla de listado de reparaciones.

#	Fecha	Estado	Precio
1	18/9/2023	PENDING	5€
2	22/9/2023	PENDING	50€
3	22/9/2023	PENDING	5€

Figura 11.15: Interfaz de página de listado de reparaciones

11.3.4.2. Información de una reparación

En la figura 11.16 se muestra la pantalla de información de una reparación.

#	Fecha	Estado	Precio
#1 - 18/9/2023	REVIEWING	5€	
#2 - 22/9/2023	PENDING	5€	
#3 - 22/9/2023	PENDING	50€	
#4 - 22/9/2023	PENDING	5€	

Figura 11.16: Interfaz de página de información de una reparación

11.3.5. Interfaz del perfil del técnico

En la figura 11.17 se muestra la pantalla de editar el perfil.

The screenshot shows a user interface for changing a password. At the top, there are four icons: a house, a shopping cart, a person, and a gear. Below them is the text "Cambiar contraseña". There are two input fields: "Contraseña actual" and "Nueva contraseña", both with placeholder text. A blue "Cambiar" button is located below the fields.

Contraseña actual
Nueva contraseña

Cambiar

Figura 11.17: Interfaz de página del perfil del técnico

11.4. Interfaces de la aplicación del cliente

11.4.1. Interfaz de inicio

En la figura 11.18 se muestra la pantalla de la página de inicio.



Figura 11.18: Interfaz de la página de inicio

11.4.2. Interfaz de inicio de sesión

En la figura 11.19 se muestra la pantalla de la página de inicio de sesión.

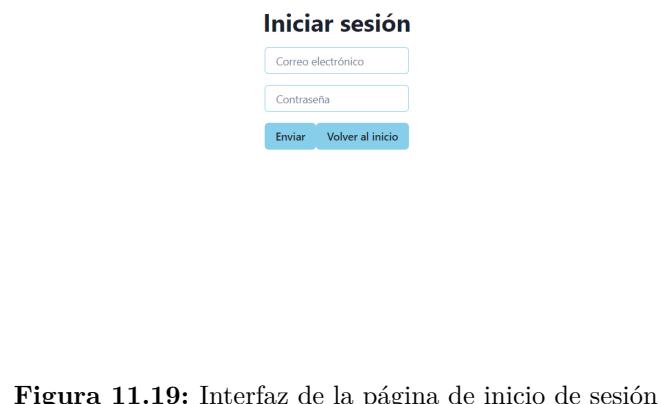


Figura 11.19: Interfaz de la página de inicio de sesión

11.4.3. Interfaz de *Home* del cliente

En la figura 11.20 se muestra la pantalla de inicio de la aplicación donde se mostrarán un listado de reparaciones del cliente.

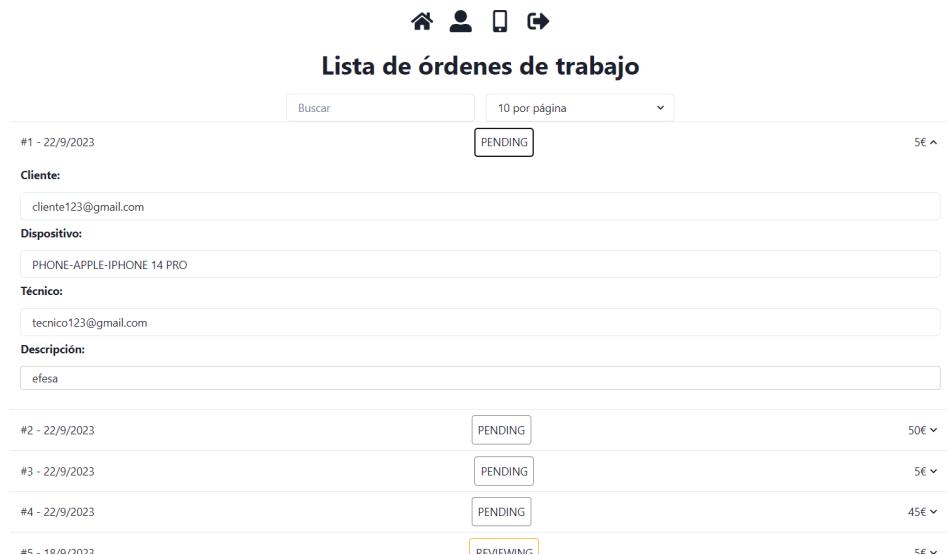


Figura 11.20: Interfaz de página principal del cliente

11.4.4. Interfaz del perfil del cliente

En la figura 11.21 se muestra la pantalla de editar el perfil.

The screenshot shows a user interface for changing a password. At the top, there are four icons: a house, a person, a gear, and a right-pointing arrow. Below them is the text "Cambiar contraseña". There are two input fields: "Contraseña actual" and "Nueva contraseña", both with placeholder text. A blue "Cambiar" button is positioned below the new password field.

Contraseña actual
Nueva contraseña

Cambiar

Figura 11.21: Interfaz de página del perfil del cliente

CAPÍTULO

12

Pruebas

Durante el desarrollo de este *Trabajo de Fin de Grado* se ha ido testeando a medida que había nuevas funcionalidades. Las pruebas realizadas han sido de manera manual (interacción *frontend-backend*) y de manera automatizada (tests unitarios).

No se ha seguido ninguna estrategia en específico al realizar los tests, simplemente cuando se desarrollaba una nueva funcionalidad testeó con distintas casuísticas. En cuanto a los tests automáticos, son tests unitarios utilizando *Jest* que es la biblioteca de Typescript la cual ejecuta e interpreta los tests. Los tests unitarios prueban pequeñas funciones, en el caso del proyecto se han empleado para testear los *valueObjects*(tipo de dato que en DDD representa un valor descriptivo con su comportamiento) del dominio de la aplicación. Con esto se asegura que la lógica de negocio funciona como se espera.

Los ficheros que han sido generados son:

Fichero/Propósito

test/unit/user/email.spec.ts

Contiene los tests unitarios para el el valueobject UserEmail de User

test/unit/user/password.spec.ts

Contiene los tests unitarios para el el valueobject UserPasssword de User

test/unit/user/rol.spec.ts

Contiene los tests unitarios para el el valueobject UserRole de User

test/unit/user/userId.spec.ts

Contiene los tests unitarios para el el valueobject UserId de User

Tabla 12.1: Ficheros principales de testing relacionados con los dispositivos

Fichero/Propósito

test/unit/device/brand.spec.ts

Contiene los tests unitarios para el el valueobject DeviceBrand de Device

test/unit/device/deviceId.spec.ts

Contiene los tests unitarios para el el valueobject DeviceId de Device

test/unit/device/model.spec.ts

Contiene los tests unitarios para el el valueobject DeviceModel de Device

test/unit/device/type.spec.ts

Contiene los tests unitarios para el el valueobject DeviceType de Device

Tabla 12.2: Ficheros principales de testing relacionados con los dispositivos

Fichero/Propósito

test/unit/work-order/description.spec.ts

Contiene los tests unitarios para el el valueobject WorkOrderDescription de WorkOrder

test/unit/work-order/work-order-end-date.spec.ts

Contiene los tests unitarios para el el valueobject WorkOrderEndDate de WorkOrder

test/unit/work-order/work-order-id.spec.ts

Contiene los tests unitarios para el el valueobject WorkOrderId de WorkOrder

test/unit/work-order/work-order-idCustomer.spec.ts

Contiene los tests unitarios para el el valueobject WorkOrderCustomerId de WorkOrder

test/unit/work-order/work-order-idDevice.spec.ts

Contiene los tests unitarios para el el valueobject WorkOrderDeviceId de WorkOrder

test/unit/work-order/work-order-idTechnician.spec.ts

Contiene los tests unitarios para el el valueobject WorkOrderTechnicianId de WorkOrder

test/unit/work-order/work-order-price.spec.ts

Contiene los tests unitarios para el el valueobject WorkOrderPrice de WorkOrder

test/unit/work-order/work-order-start-date.spec.ts

Contiene los tests unitarios para el el valueobject WorkOrderStartDate de WorkOrder

test/unit/work-order/work-order-status.spec.ts

Contiene los tests unitarios para el el valueobject WorkOrderStatus de WorkOrder

Tabla 12.3: Ficheros principales de testing relacionados con los dispositivos

CAPÍTULO

13

Conclusiones y futuras mejoras

Al haber finalizado todas las etapas de este *Trabajo de Fin de Grado*, a continuación se detalla un análisis de desarrollo del proyecto y verificar si ha sido posible el cumplimiento de los objetivos propuestos.

13.1. Análisis de objetivos

13.1.1. Objetivo 1: Gestión de partes de reparaciones

El sistema final permite la alta, edición, visualización y eliminación de reparaciones con el rol de administrador.

13.1.2. Objetivo 2: Gestión de dispositivos electrónicos

El sistema final permite la alta, visualización y eliminación de dispositivos electrónicos con el rol de administrador.

13.1.3. Objetivo 3: Sistema de notificaciones

El sistema final utiliza un sistema de notificaciones de mail para notificar al usuario que se ha registrado satisfactoriamente en el sistema.

13.1.4. Objetivo 4: Gestión de roles de usuario que permitan acceder a las diferentes partes del sistema

El sistema final permite controlar los roles y permisos identificados previamente (administrador, técnico y cliente).

13.1.5. Objetivo 5: Gestión de registro de clientes

El sistema final permite la alta, visualización y eliminación de clientes con el rol de administrador.

13.1.6. Objetivo 6: Gestión de registro de técnicos

El sistema final permite la alta, visualización y eliminación de técnicos con el rol de administrador.

13.1.7. Objetivo 7: Uso de KPIs

El sistema final permite al administrador y técnico visualizar estadísticas y KPIs relacionadas con el negocio. Esto se observa en la página home del administrador y técnico.

13.2. Problemas encontrados y soluciones

13.2.1. Empleo de dos bases de datos de lectura

Una de las decisiones más complicadas ha sido emplear dos bases de datos de lectura (MongoDB y Redis). Al principio fue difícil mantener la consistencia entre ambas bases de datos, lo cual ocupó varias semanas. Una vez todo funcionaba perfectamente y ambas bases de datos de lectura reflejaban en tiempo real una proyección del EvenStoreDB se pudo continuar con otros retos. Principalmente la elección de Redis ha sido por su rapidez y eficiencia en búsqueda indexada, lo cual es útil para leer datos de usuarios, ya que éstos se buscan por índice. Por otro lado, MongoDB es útil cuando hay mucha información relacionada, por lo que es muy potente y eficiente en lecturas de información de reparaciones de dispositivos electrónicos.

13.2.2. Reconstrucción de eventos con la base de datos de *EventStore*

Durante el desarrollo del proyecto se descubrió que el módulo que ofrece el *framework* de *NestJS* para integrar una base de datos basada en eventos como *EventStoreDB*[14] a la hora de implementar y llevar un seguimiento de los eventos disparados de *EventSourcing*[9] no funcionaba correctamente a la hora de reconstruir ciertos eventos en base a las proyecciones generadas. Para solucionar este problema el *Aula de Software Libre* desarrolló un módulo que podemos encontrar en la plataforma *Github* de código libre, en el siguiente enlace: <https://github.com/aulasoftwarelibre/nestjs-eventstore>, el cual integraba módulos nativos de *Eventsore* en vez de usar los propuestos por *NestJS*.

13.2.3. Complejidad en configurar Docker

Uno de los grandes retos era utilizar Docker para la creación de la API por muchas ventajas que ofrece esta tecnología, la principal de ellas es la facilidad de instalación y uso entre distintos sistemas operativos y ordenadores en poco tiempo. El principal problema fue configurar los puertos, instalación y despliegue de las tecnologías empleadas en el mismo. Pero con varias semanas de búsqueda y adquisición de nuevo conocimiento se pudo solventar.

13.3. Conclusiones

Podemos comprobar que se han cumplido todos los objetivos que nos propusimos al principio del proyecto y que hemos creado un sistema que, además, ha pasado todas las pruebas del sistema.

13.4. Futuras mejoras

Algunas de las mejoras que se esperan realizar en este proyecto próximamente son:

- Integración con una pasarela de pago para permitir a los clientes pagar desde la aplicación.
- Integración de servicio de entrega y recogida de dispositivos.
- Implementación de un chat interno para clientes, técnicos y administradores.
- Añadir nuevas estadísticas tanto para el administrador como técnico.
- Añadir notificaciones multicanal (email o mensajería instantánea) para los clientes.

Referencias

- [1] “Nestjs.” <https://nestjs.com/>. Fecha de consulta: diciembre 2023.
- [2] “Reactjs.” <https://es.react.dev/>. Fecha de consulta: diciembre 2023.
- [3] “Jwt.” <https://jwt.io/>. Fecha de consulta: diciembre 2023.
- [4] “Qué es scrum.” <https://proyectosagiles.org/que-es-scrum/>. Visitado el 03/12/2023.
- [5] E. Evans, *Domain-Driven Design: Tackling Complexity in the Heart of Software*. Addison-Wesley Professional, 2004.
- [6] “Event modeling.” <https://eventmodeling.org/posts/what-is-event-modeling/>. Visitado el 03/08/2023.
- [7] “Chakraui.” <https://chakra-ui.com/>. Visitado el 05/12/2023.
- [8] “CQRS.” <https://docs.microsoft.com/es-es/azure/architecture/patterns/cqrs>. Visitado el 05/08/2023.
- [9] “Event sourcing.” <https://martinfowler.com/eaaDev/EventSourcing.html>. Visitado el 03/08/2023.
- [10] “Nx.” <https://nx.dev/>. Fecha de consulta: diciembre 2023.
- [11] “Ibm 2008 - casos de uso.” <https://www.ibm.com/docs/en/engineering-lifecycle-management-suite/lifecycle-management/6.0.1?topic=cases-use>. Fecha de consulta: diciembre 2023.

- [12] R. C. Martin, *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Pearson Education, 2017.
- [13] V. Vernon, *Implementing Domain-driven Design*. Addison-Wesley, 2013.
- [14] “Event store.” <https://microservices.io/patterns/data/event-sourcing.html>. Visitado el 03/08/2023.
- [15] C. de Coordinación de Alertas y Emergencias Sanitarias, “Enfermedad por coronavirus, covid-19.” <https://www.mscbs.gob.es/profesionales/saludPublica/ccayes/alertasActual/nCov/documentos/ITCoronavirus.pdf>, Noviembre 12, 2023.
- [16] “API.” <https://www.mulesoft.com/resources/api/what-is-an-api>. Visitado el 03/12/2023.
- [17] “Resident advisor.” <https://www.residentadvisor.net/>. Visitado el 01/12/2023.
- [18] “Vampr.” <https://www.vampr.me/>. Visitado el 01/12/2023.
- [19] “Git.” <https://git-scm.com/>. Visitado el 03/12/2023.
- [20] “Miro.” <https://miro.com/>. Visitado el 05/08/2023.
- [21] “Visual studio code.” <https://code.visualstudio.com/>. Visitado el 03/12/2023.
- [22] “Overleaf.” <https://es.overleaf.com/>. Visitado el 03/12/2023.
- [23] “Responsive.” <https://www.arimetrics.com/glosario-digital/responsive>. Visitado el 05/10/2023.
- [24] “Docker.” <https://www.docker.com/>. Visitado el 10/07/2023.
- [25] “Gitub.” <https://github.com/>. Visitado el 10/07/2023.
- [26] “Bearer token.” <https://swagger.io/docs/specification/authentication/bearer-authentication/>. Visitado el 10/07/2023.
- [27] “Swagger.” <https://swagger.io/>. Visitado el 10/07/2023.
-

- [28] RepairShopr, “Repairshopr - software de gestión de reparaciones.” Fecha de consulta: agosto 2023.
- [29] RepairDesk, “Repairdesk - software de gestión de reparaciones.” Fecha de consulta: agosto 2023.
- [30] mHelpDesk, “mhelpdesk - field service software.” Fecha de consulta: agosto 2023.
- [31] “Nextjs.” <https://nextjs.org/docs>. Fecha de consulta: diciembre 2023.

A.1. Instalación del software

Todas las aplicaciones de este proyecto están ubicadas en el mismo repositorio. Para facilitar la gestión de las dependencias, compartición de código y arranque de los distintos aplicativos se ha hecho uso del *framework Nx*.

El repositorio se puede encontrar en el siguiente enlace: <https://github.com/tomashm01/fixtrack>

A.1.1. Requisitos previos

El *backend* de la aplicación está desarrollado con *NestJS*, framework de *TypeScript*. Por otro lado, el *frontend* se ha desarrollado haciendo uso de *Next.js* un *framework* para la librería de desarrollo de componentes web en *Javascript*, *React*.

En la figura A.1 podemos ver el *software* a instalar junto a sus versiones mínimas.

Software	Versión mínima
npm	9.8.1
node	16.7.3
nestjs	16.7.3
docker	24.0.5
react	16.7.3
next	16.7.3

Tabla A.1: Tabla de requisitos para la instalación de las dependencias del sistema

Para instalar las dependencias de la aplicación necesitamos instalar **npm**, un gestor de paquetes para *Node.js*. Para ello, ejecutaremos los siguientes comandos:

```
$ sudo apt-get update && sudo apt-get install nodejs  
$ sudo apt-get install npm
```

Además, será necesaria la instalación del sistema de despliegue de contenedores **Docker**, siguiendo los pasos mostrados a continuación:

```
$ sudo curl -sSL https://get.docker.com/ | sh  
$ sudo apt install docker.io  
$ sudo systemctl start docker  
$ sudo systemctl enable docker
```

A.1.2. Instalación del software

Lo primero será descargar la última versión del software del repositorio oficial y descomprimir el paquete. Una vez descargado el código podremos descomprimirlo de la siguiente manera:

```
$ unzip fixtrack.zip  
$ cd fixtrack
```

Ahora podemos proceder a instalar las dependencias de nuestro proyecto.

```
$ npm i
```

Tras esto será necesario instalar el servicio de **Docker** para ejecutar múltiples aplicaciones utilizando los contenedores definidos en el fichero *docker-compose.yaml*. Con esto ejecutaremos los contenedores necesarios para la **Event Store**, las base de datos de lectura (**MongoDB** y **Redis**) y la API REST de la aplicación:

```
$ sudo apt install docker-compose  
$ sudo docker-compose up -d
```

A.1.3. Comprobación de la instalación

Para comprobar que se ha instalado todo correctamente se arranca el proyecto, se inicializará la aplicación *frontend*, habrá que ejecutar los siguientes comandos:

```
1$ nx serve web
```

También podremos ejecutar todas las pruebas del proyecto con el comando:

```
1$ nx run test
```

A.1.4. Desinstalación del software

Para la desinstalación del software los pasos a seguir son los siguientes:

- 1. Borrar la configuración y contenedores de *Docker*.
- 2. Borrar el directorio de instalación y el contenido del éste.

CAPÍTULO

B

Manual de usuario

B.1. Interfaces de la aplicación de administrador

B.1.1. Interfaz de inicio

En la figura B.1 se muestra la pantalla de la página de inicio.

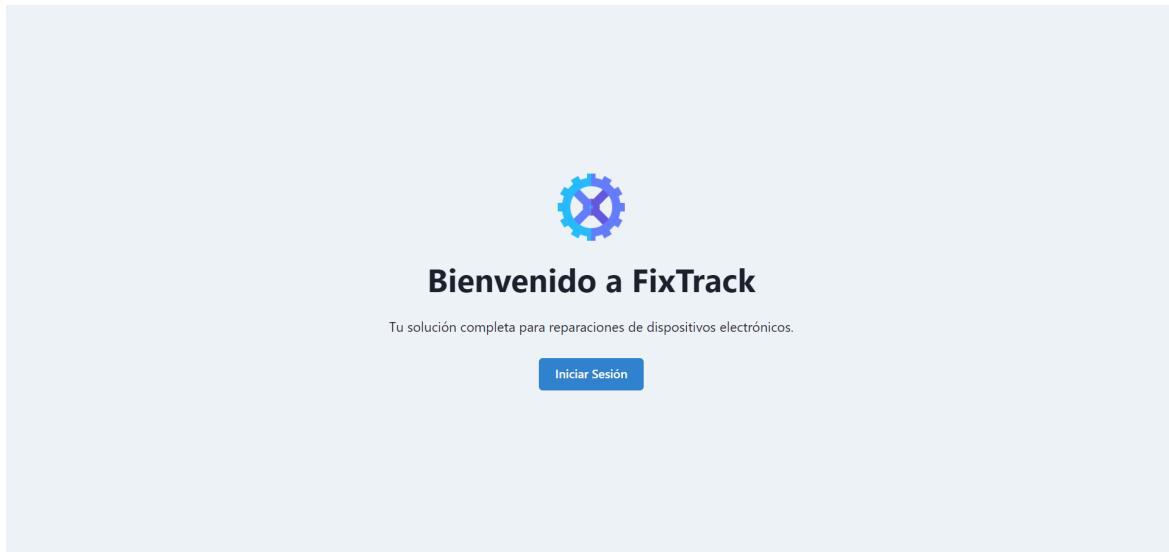


Figura B.1: Interfaz de la página de inicio

B.1.2. Interfaz de inicio de sesión

En la figura B.2 se muestra la pantalla de la página de inicio de sesión.

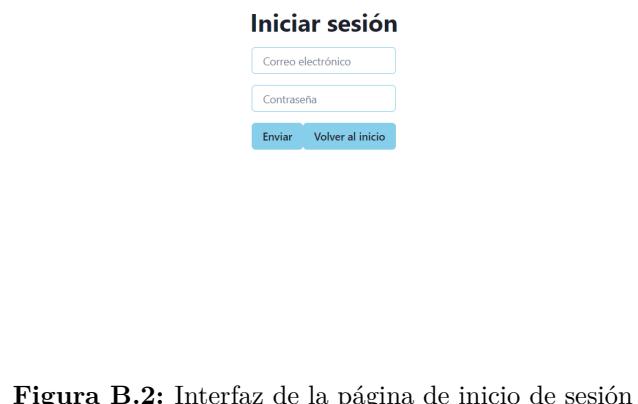


Figura B.2: Interfaz de la página de inicio de sesión

B.1.3. Interfaz de Home del administrador

En la figura B.3 se muestra la pantalla de inicio de la aplicación donde se mostrarán un listado de estadísticas o KPIs relacionadas con el negocio. Éstas estadísticas pueden ser modificadas y sustituidas por otras, es por ello que este apartado de la aplicación depende de negocio.



Figura B.3: Interfaz de página principal del administrador

B.1.4. Interfaz de página de reparaciones del administrador

B.1.4.1. Listado de reparaciones

En la figura B.4 se muestra la pantalla de listado de reparaciones. Principalmente se observan el estado, precio y fecha inicio de la reparación, ya que se han considerado que son los datos más relevantes para mostrar.

Lista de órdenes de trabajo			
		Buscar	10 por página
#1 - 5/5/2021 - 5/5/2021	CANCELLED	100€	v
#2 - 18/9/2023	REVIEWING	5€	v
#3 - 22/9/2023	PENDING	5€	v
#4 - 22/9/2023	PENDING	50€	v
#5 - 22/9/2023	PENDING	5€	v
#6 - 22/9/2023	PENDING	45€	v
#7 - 22/9/2023	PENDING	12€	v

Figura B.4: Interfaz de página de listado de reparaciones

B.1.4.2. Información de una reparación

En la figura B.5 se muestra la pantalla de información de una reparación. Aquí se detalla más datos relacionados con una reparación. El administrador puede tanto eliminar como editar la información.

The screenshot shows a user interface for managing work orders. At the top, there are navigation icons: a house, a shopping cart, a person, a phone, and a refresh symbol. Below them is the title "Lista de órdenes de trabajo". On the left, there's a button labeled "Crear orden de trabajo". To the right are search and filter fields, including a dropdown for "10 por página".

Work order #1 details:

- Cliente:** cliente123@gmail.com
- Dispositivo:** PHONE-APPLE-IPHONE 14
- Técnico:** tecnico123@gmail.com
- Descripción:** The device is broken

Below these details are two buttons: "Actualizar" (Update) and "Eliminar" (Delete). The status of this work order is "CANCELLED".

Below the details, there's a table listing four other work orders:

ID	Fecha	Estado	Precio
#2	18/9/2023	REVIEWING	5€
#3	22/9/2023	PENDING	5€
#4	22/9/2023	PENDING	50€

Figura B.5: Interfaz de página de información de una reparación

B.1.4.3. Crear una reparación

En la figura B.6 se muestra la pantalla de crear una reparación. El usuario administrador rellenará los datos para crear una reparación en pocos pasos.

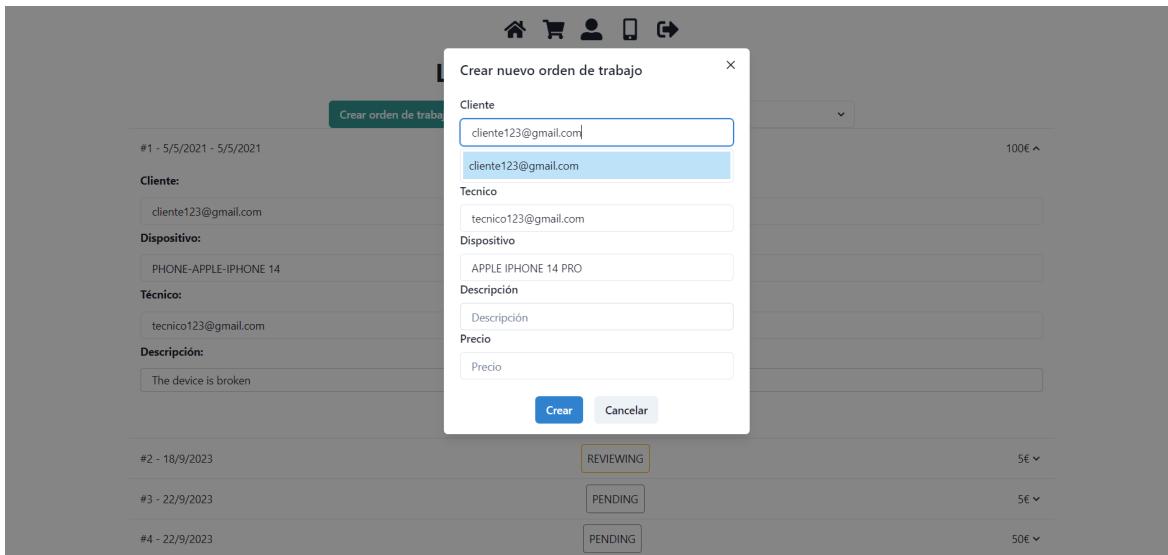


Figura B.6: Interfaz de página de crear una reparación

B.1.5. Interfaz de página de dispositivos del administrador

B.1.5.1. Listado de dispositivos

En la figura B.7 se muestra la pantalla de listado de dispositivos.

ID	TIPO	MARCA	MODELO	ACCIONES
1	PHONE	APPLE	IPHONE 14 PRO	<button>Eliminar</button>
2	PHONE	APPLE	IPHONE 14	<button>Eliminar</button>
3	PHONE	SAMSUNG	GALAXY S22	<button>Eliminar</button>
4	LAPTOP	APPLE	MACBOOK AIR 2020	<button>Eliminar</button>
5	TABLET	APPLE	IPAD AIR PRO 2020	<button>Eliminar</button>
6	PHONE	LG	G7 PLUS	<button>Eliminar</button>
7	PHONE	SAMSUNG	S22 ULTRA	<button>Eliminar</button>
8	PHONE	SAMSUNG	GALAXY S22 ULTRA	<button>Eliminar</button>

Figura B.7: Interfaz de página de listado de dispositivos

B.1.5.2. Crear un dispositivo

En la figura B.8 se muestra la pantalla de crear un dispositivo.

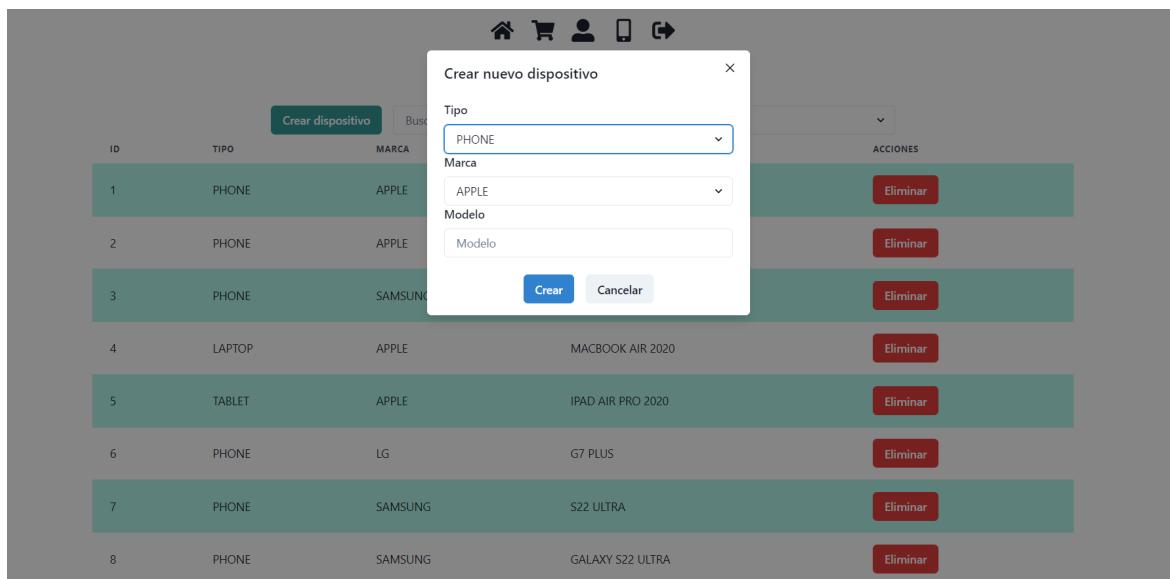


Figura B.8: Interfaz de página de crear un dispositivo

B.1.6. Interfaz de página de usuarios del administrador

B.1.6.1. Listado de usuarios

En la figura B.9 se muestra la pantalla de listado de usuarios.

The screenshot shows a user management interface titled "Lista de usuarios". At the top, there are icons for home, shopping cart, user profile, and a right-pointing arrow. Below the title is a search bar labeled "Buscar" and a dropdown menu set to "10 por página". A green button labeled "Crear usuario" is located on the left. The main area contains a table with the following data:

ID	EMAIL	ROLE	ACCIONES
1	admin123@gmail.com	ADMIN	<button>Eliminar</button>
2	cliente123@gmail.com	CLIENTE	<button>Eliminar</button>
3	tecnico123@gmail.com	TECNICO	<button>Eliminar</button>
4	ejemplo123@gmail.com	CLIENTE	<button>Eliminar</button>
5	ejemplito12@gmail.com	CLIENTE	<button>Eliminar</button>
6	juanjo@gmail.com	ADMIN	<button>Eliminar</button>

At the bottom, there are navigation links: "Anterior", "Página 1 de 1", and "Siguiente".

Figura B.9: Interfaz de página de listado de usuarios

B.1.6.2. Crear un usuario

En la figura B.10 se muestra la pantalla de crear un usuario.

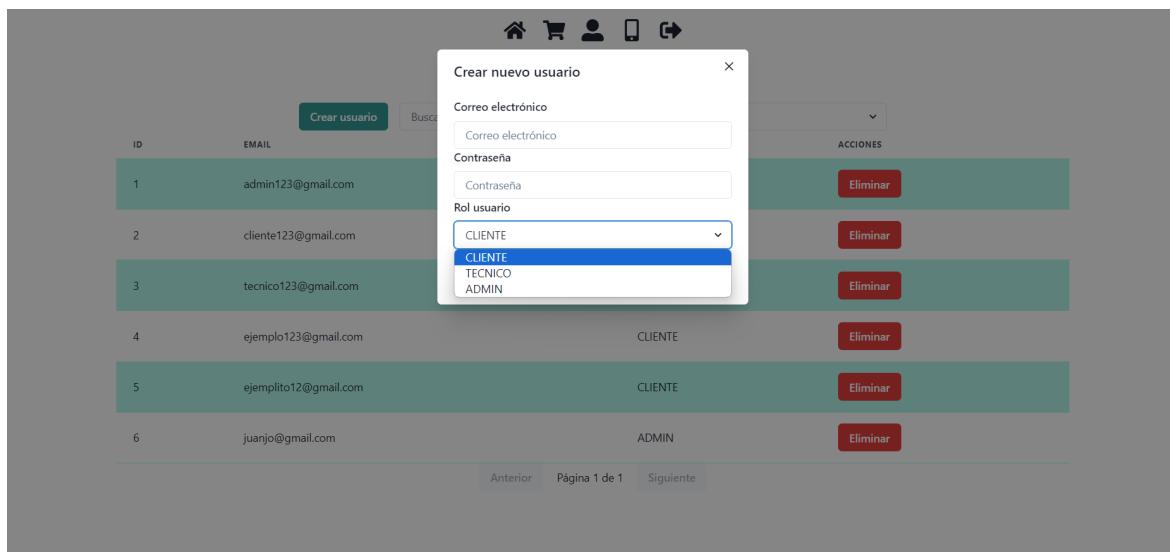


Figura B.10: Interfaz de página de crear un usuario

B.2. Interfaces de la aplicación de técnico

B.2.1. Interfaz de inicio

En la figura B.11 se muestra la pantalla de la página de inicio.



Figura B.11: Interfaz de la página de inicio

B.2.2. Interfaz de inicio de sesión

En la figura B.12 se muestra la pantalla de la página de inicio de sesión.

La imagen es una captura de pantalla de un formulario de inicio de sesión. El título "Iniciar sesión" está centrado en la parte superior. Abajo de él hay dos cuadros de texto: el primero para "Correo electrónico" y el segundo para "Contraseña". A continuación, hay un botón azul con el texto "Enviar" y un enlace azul que dice "Volver al inicio".

Correo electrónico	
Contraseña	
Enviar	Volver al inicio

Figura B.12: Interfaz de la página de inicio de sesión

B.2.3. Interfaz de Home del técnico

En la figura B.13 se muestra la pantalla de inicio de la aplicación donde se mostrarán un listado de estadísticas relacionadas con el técnico y el negocio. Estos datos también pueden ser modificados y sustituidos por otros que requiera el negocio.



Figura B.13: Interfaz de página principal del técnico

B.2.4. Interfaz de página de reparaciones del técnico

B.2.4.1. Listado de reparaciones

En la figura B.14 se muestra la pantalla de listado de reparaciones.

Lista de órdenes de trabajo

Buscar 10 por página

#1 - 18/9/2023 REVIEWS 5€ ^

Cliente: cliente123@gmail.com

Dispositivo: PHONE-APPLE-IPHONE 14 PRO

Técnico: tecnico123@gmail.com

Descripción: ejemplo

Actualizar

#2 - 22/9/2023 PENDING 5€ ^

#3 - 22/9/2023 PENDING 50€ ^

#4 - 22/9/2023 PENDING 5€ ^

Figura B.14: Interfaz de página de listado de reparaciones

B.2.4.2. Información de una reparación

En la figura B.15 se muestra la pantalla de información de una reparación.

#	Fecha	Estado	Precio
#1 - 18/9/2023	REVIEWING	5€	
#2 - 22/9/2023	PENDING	5€	
#3 - 22/9/2023	PENDING	50€	
#4 - 22/9/2023	PENDING	5€	

Figura B.15: Interfaz de página de información de una reparación

B.2.5. Interfaz del perfil del técnico

En la figura B.16 se muestra la pantalla de editar el perfil.

The screenshot shows a user interface for changing a password. At the top, there are four icons: a house, a shopping cart, a person, and a right-pointing arrow. Below these is a link labeled "Cambiar contraseña". Underneath the link are two input fields: one for "Contraseña actual" and another for "Nueva contraseña". A blue "Cambiar" button is positioned below the new password field.

Cambiar contraseña

Contraseña actual

Nueva contraseña

Cambiar

Figura B.16: Interfaz de página del perfil del técnico

B.3. Interfaces de la aplicación del cliente

B.3.1. Interfaz de inicio

En la figura B.17 se muestra la pantalla de la página de inicio.



Figura B.17: Interfaz de la página de inicio

B.3.2. Interfaz de inicio de sesión

En la figura B.18 se muestra la pantalla de la página de inicio de sesión.

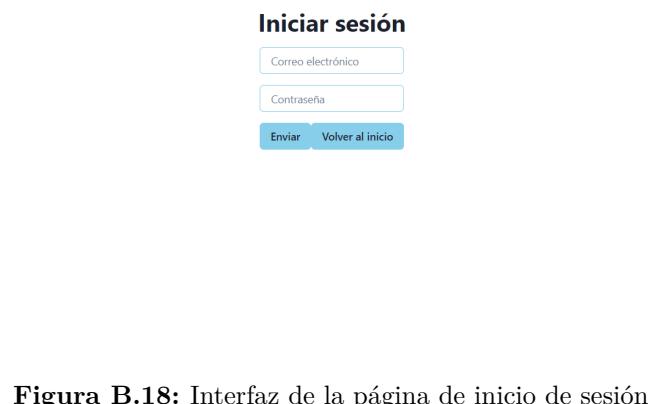


Figura B.18: Interfaz de la página de inicio de sesión

B.3.3. Interfaz de Home del cliente

En la figura B.19 se muestra la pantalla de inicio de la aplicación donde se mostrarán un listado de reparaciones del cliente.

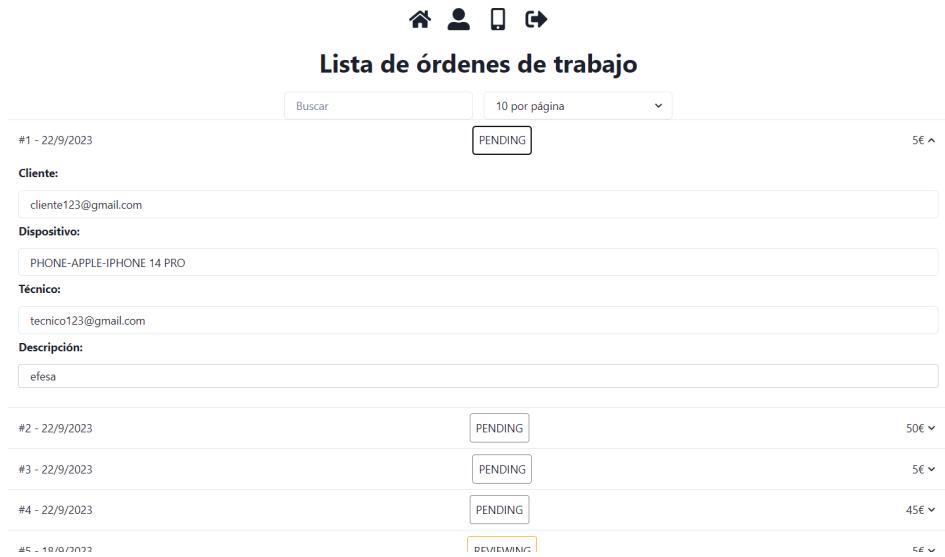


Figura B.19: Interfaz de página principal del cliente

B.3.4. Interfaz del perfil del cliente

En la figura B.20 se muestra la pantalla de editar el perfil.

The screenshot shows a user interface for changing a password. At the top, there are four icons: a house, a person, a document, and a gear. Below them is the text "Cambiar contraseña". There are two input fields: "Contraseña actual" and "Nueva contraseña", both with placeholder text. A blue "Cambiar" button is located below the new password field.

Contraseña actual
Nueva contraseña

Cambiar

Figura B.20: Interfaz de página del perfil del cliente

C.1. Introducción

Este manual de código corresponde con el sistema de gestión de reparaciones de dispositivos electrónicos de *Fixtrack*. A continuación, se detallan los requisitos previos para la implementación del sistema y se presentan todos los archivos que forman parte de la aplicación, organizados según su funcionalidad y describiendo su contenido.

El código fuente está disponible en la siguiente dirección: <https://github.com/tomashm01/fixtrack>

C.2. Requisitos previos

El *backend* de la aplicación está desarrollado con *NestJS*, framework de *TypeScript*. Por otro lado, el *frontend* se ha desarrollado haciendo uso de *Next.js* un *framework* para la librería de desarrollo de componentes web en *Javascript*, *React*.

En la figura C.1 podemos ver el *software* a instalar junto a sus versiones mínimas.

Software	Versión mínima
npm	9.8.1
node	16.7.3
nestjs	16.7.3
docker	24.0.5
react	16.7.3

next	16.7.3
------	--------

Tabla C.1: Tabla de requisitos para la instalación de las dependencias del sistema

C.3. Descripción modular

La estructura principal corresponde a la recomendada en los proyectos de *Nx*, diferenciando las aplicaciones dentro de una carpeta *apps* en la raíz del proyecto, y en una carpeta *libs* las librerías de componentes, unidades de código o contratos que convivan entre varias aplicaciones, ya sean *backend* o *frontend*.

C.3.1. Backend

El módulo principal del *backend* de la aplicación se sitúa en el directorio *apps/api*. Con el objetivo de hacer el proyecto modular, cada uno de los paquetes que conforman la aplicación del *backend* se han desarrollado como aplicaciones de *NestJS* independientes para que se puedan integrar en cualquier otro proyecto de manera sencilla sin dependencias externas de otros módulos, son estas aplicaciones las que se inyectan en el fichero principal del directorio *apps/fixtrack*.

Ficheros principales de *apps/fixtrack/src*

Fichero/Propósito

main.ts

Contiene la configuración principal sobre la api de nestjs, recomendada por la documentación del framework para usar OpenAPI .

app.module.ts

Es el módulo principal de toda la aplicación, en él se inyectan los módulos de cada paquete (user, work-order, auth, device) además de las configuraciones de las bases de datos (EventStore, MongoDB y Redis).

Tabla C.2: Ficheros principales del backend

A continuación se muestran los módulos que contienen la lógica de negocio, los cuales se

inyectan en el fichero principal de la aplicación `app.module.ts`. Estos módulos los encontramos a partir del directorio raíz `apps/fixtrack/src`, cada uno como una aplicación de *NestJS* independiente. Esta estructura se ha seguido basándose en las recomendaciones de la documentación oficial del *framework*.

Directarios de `apps/fixtrack/src`

Directorio/Propósito

`auth/src/`

Contiene el *bounded context* de género autenticación.

`device/src/`

Contiene el *bounded context* de dispositivo.

`user/src/`

Contiene el *bounded context* de usuario.

`work-order/src/`

Contiene el *bounded context* de reparaciones de dispositivos electrónicos.

Tabla C.3: Directarios principales de los módulos de la aplicación

Directarios de `apps/fixtrack/src/user/`

Directorio/Propósito

`user/domain/`

Piezas de la capa de dominio del contexto usuario.

`user/domain/model`

Agregado raíz y *value objects* de la entidad usuario.

`user/domain/event`

Eventos de dominio del contexto usuario.

`user/domain/exception/`

Excepciones de dominio del contexto usuario.

`user/application/`

Piezas de la capa de aplicación del contexto usuario.

`user/application/command/`

Comandos y manejadores de los casos de uso de escritura del contexto de usuario.

user/application/query/

Consultas y manejadores de los casos de uso de lectura del contexto de usuario.

user/infrastructure/

Piezas de la capa de infraestructura del contexto usuario.

user/infrastructure/controller

Controlador del contexto de usuario.

user/infrastructure/service

Servicios de infraestructura del contexto de usuario.

user//infrastructure/projection

Proyecciones realizadas desde el EventStore a MongoDB y/o Redis.

user/infrastructure/user.module.ts

Fichero de configuración del módulo que carga las dependencias del mismo.

user/infrastructure/user.provider.ts

Fichero de configuración de inyección de dependencias.

Tabla C.4: Directorios user

Directarios de *apps/fixtrack/src/device*

Directorio/Propósito

device/domain/

Piezas de la capa de dominio del contexto dispositivos.

device/domain/model

Agregado raíz y *value objects* de la entidad dispositivos.

device/domain/event

Eventos de dominio del contexto dispositivos.

device/domain/exception/

Excepciones de dominio del contexto dispositivos.

device/application/

Piezas de la capa de aplicación del contexto dispositivos.

device/application/command/

Comandos y manejadores de los casos de uso de escritura del contexto de dispositivos.

device/application/query/

Consultas y manejadores de los casos de uso de lectura del contexto de dispositivos.

device/infrastructure/

Piezas de la capa de infraestructura del contexto dispositivos.

device/infrastructure/controller

Controlador del contexto de dispositivos.

device/infrastructure/service

Servicios de infraestructura del contexto de dispositivos.

device/infrastructure/projection

Proyecciones realizadas desde el EventStore a MongoDB y/o Redis.

device/infrastructure/user.module.ts

Fichero de configuración del módulo que carga las dependencias del mismo.

device/infrastructure/user.provider.ts

Fichero de configuración de inyección de dependencias.

Tabla C.5: Directorios device

Directarios de *apps/fixtrack/src/work-order*

Directorio/Propósito**work-order/domain/**

Piezas de la capa de dominio del contexto reparaciones.

work-order/domain/model

Agregado raíz y *value objects* de la entidad reparaciones.

work-order/domain/event

Eventos de dominio del contexto reparaciones.

work-order/domain/exception/

Excepciones de dominio del contexto reparaciones.

work-order/application/

Piezas de la capa de aplicación del contexto reparaciones.

work-order/application/command/

Comandos y manejadores de los casos de uso de escritura del contexto de reparaciones.

work-order/application/query/

Consultas y manejadores de los casos de uso de lectura del contexto de reparaciones.

work-order/infrastructure/

Piezas de la capa de infraestructura del contexto reparaciones.

work-order/infrastructure/controller

Controlador del contexto de reparaciones.

work-order/infrastructure/service

Servicios de infraestructura del contexto de reparaciones.

work-order/infrastructure/projection

Proyecciones realizadas desde el EventStore a MongoDB y/o Redis.

work-order/infrastructure/user.module.ts

Fichero de configuración del módulo que carga las dependencias del mismo.

work-order/infrastructure/user.provider.ts

Fichero de configuración de inyección de dependencias.

Tabla C.6: Directorios work-order

Directarios de *apps/fixtrack/src/auth/*

Directorio/Propósito

auth/service/

Servicios de autenticación.

auth/jwt.strategy

Fichero strategy de la verificación JWT.

Tabla C.7: Directorios auth

C.3.2. *Frontend*

En cuanto al *frontend* encontramos una aplicación dentro del directorio ***apps/web***, que identifica el rol y permisos del usuario que ha iniciado sesión y se comunica con los controladores correspondientes de la API REST. **Directorios de *apps/web***

Directorio/Propósito
components/
Componentes creados que se reutilizan en varias páginas de la aplicación web.
hoc/
Componentes de orden superior.
pages/
Páginas de la aplicación web.
public/
Ficheros públicos de la aplicación web.
services/
Servicios que integran la API REST con la aplicación web.

Tabla C.8: Directorios *apps/web*

C.3.3. **Contratos**

El ***frontend* y el *backend*** de la aplicación usan unos contratos (conjunto de especificaciones o interfaces acordada entre las dos partes) compartidos para simplificar la dependencia existente entre ambas partes, además de minimizar el error a la hora de enviar una petición o esperar una respuesta. Estos contratos se encuentran en el directorio ***libs/contracts***, en la raíz del proyecto.

Directorios de *libs/contracts*

Directorio/Propósito
contracts/auth
Objetos de petición, respuesta y reglas de negocio relacionados con la autenticación.
contracts/user

Objetos de petición, respuesta y reglas de negocio relacionados con el usuario.

contracts/device

Objetos de petición, respuesta y reglas de negocio relacionados con el dispositivo.

contracts/work-order

Objetos de petición, respuesta y reglas de negocio relacionados con la reparación.

Tabla C.9: Directorios libs/contracts

C.4. Dependencias

C.4.1. Dependencias para el entorno de desarrollo

Las dependencias para el entorno de desarrollo son las siguientes:

Dependencia	Versión
@aulasoftwarelibre/nestjs-eventstore	^0.8.0
@nestjs/config	^3.0.0
@nestjs/cqrs	^10.2.5
@nestjs/elasticsearch	^10.0.1
@nestjs/jwt	^10.1.0
@nestjs/microservices	^10.2.1
@nestjs/mongoose	^10.0.1
@nestjs/schematics	^10.0.1
@nestjs/swagger	^7.1.8
@nestjs/testing	^10.0.2
@nrwl/cli	^15.9.3
@nrwl/nest	^16.7.3
@nrwl/next	^16.7.3
@nx/eslint-plugin	16.7.3
@nx/jest	16.7.3
@nx/js	16.7.3
@nx/linter	16.7.3

@nx/nest	16.7.3
@nx/next	16.7.3
@nx/node	16.7.3
@nx/react	16.7.3
@nx/webpack	16.7.3
@nx/workspace	16.7.3
@testing-library/react	14.0.0
@types/jest	[~] 29.4.0
@types/node	18.14.2
@types/react	18.2.14
@types/react-dom	18.2.6
@types/uuid	[~] 9.0.3
@typescript-eslint/eslint-plugin	[~] 5.60.1
@typescript-eslint/parser	[~] 5.60.1
babel-jest	[~] 29.4.1
eslint	8.46.0
eslint-config-next	13.4.1
eslint-config-prettier	8.1.0
eslint-plugin-import	2.27.5
eslint-plugin-jsx-ally	6.7.1
eslint-plugin-react	7.32.2
eslint-plugin-react-hooks	4.6.0
event-sourcing-nestjs	[~] 1.1.4
jest	[~] 29.6.4
jest-environment-jsdom	[~] 29.4.1
jest-environment-node	[~] 29.4.1
nestjs-console	[~] 9.0.0
nx	[~] 16.8.1
nx-cloud	latest
prettier	[~] 2.6.2

shallow-equal-object	^1.1.1
swagger-ui-express	^5.0.0
ts-jest	^29.1.0
ts-node	10.9.1
typescript	5.1.3
uuid	^9.0.1

Tabla C.10: Dependencias de desarrollo

Las dependencias para el entorno de producción son las siguientes:

Dependencia	Versión
@chakra-ui/icons	^2.1.0
@chakra-ui/react	^2.8.0
@emotion/react	^11.11.1
@emotion/styled	^11.11.0
@nestjs/common	^10.0.2
@nestjs/core	^10.0.2
@nestjs/passport	^10.0.1
@nestjs/platform-express	^10.0.2
@types/cookie	^0.5.2
@types/js-cookie	^3.0.3
axios	^1.0.0
bcrypt	^5.1.1
class-validator	^0.14.0
cookie	^0.5.0
dotenv	^16.3.1
formik	^2.4.3
framer-motion	^10.16.2
ioredis	^5.3.2
js-cookie	^3.0.5
mongoose	^7.5.1

nest-winston	^1.9.4
next	13.4.1
nodemailer	^6.9.5
passport-jwt	^4.0.1
react	18.2.0
react-dom	18.2.0
react-icons	^4.10.1
react-router-dom	^6.15.0
reflect-metadata	^0.1.13
rxjs	^7.8.0
semver	^7.5.4
tslib	^2.3.0
winston	^3.10.0
winston-loki	^6.0.7

Tabla C.11: Dependencias de producción