Grado en Ingeniería Informática

Trabajo de Final de Grado

Migración de ERP de escritorio a aplicación web

Autor:
Felicitas Agüero Pratto

Supervisor:
Luis Miguel Chulvi Martínez
Tutor académico:
Carlos Granell Canut

Fecha de lectura: 12 de Julio de 2023
Curso académico 2022/2023

Resumen

Este documento se corresponde al Trabajo Final de Grado realizado por Felicitas Agüero Pratto, desarrollado durante la estancia en prácticas en la empresa Infoware Soluciones Informáticas S.L.

En el presente documento se describe el desarrollo de un proyecto de migración de un sistema de planificación de recursos empresariales (ERP) de escritorio a un entorno web. El sistema tiene como objetivo agilizar el flujo de trabajo de talleres mecánicos, proporcionando herramientas para la gestión de las reparaciones y automatizando las actividades administrativas de la organización.

La aplicación web se ha desarrollado mediante el lenguaje de programación C# utilizando el *framework* .NET MVC junto con Entity Framework para la gestión automatizada de la base de datos.

Palabras clave

.NET MVC, Entity Framework, ERP, Desarrollo en cascada, C#.

Keywords

.NET MVC, Entity Framework, ERP, Waterfall model, C#.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Contexto y motivación del proyecto

Cada vez son más las empresas que utilizan herramientas informáticas para la gestión de información [1]. Una de estas herramientas son los sistemas de planificación de recursos empresariales o ERP (por sus siglas en inglés, *enterprise resource planning*). En general, el objetivo de los sistemas ERP consiste en ayudar a las empresas y organizaciones a automatizar y administrar procesos esenciales, como por ejemplo agilizando operaciones entre distintas funciones empresariales al unificar todos los departamentos en una única fuente de información, para así conseguir un rendimiento óptimo de la gestión de la información empresarial [2].

Infoware Soluciones Informáticas S.L. es una empresa con sede en Castellón de la Plana que ofrece servicios de desarrollo de programas de escritorio, páginas web y aplicaciones móviles, así como consultoría informática y mantenimiento de redes [3]. Una de sus especialidades es el desarrollo, puesta en marcha y mantenimiento de aplicaciones ERP a medida [4].

Teniendo en cuenta lo anterior, Infoware Soluciones Informáticas S.L. busca proporcionar una versión renovada de la aplicación ERP de escritorio **+Taller**, cuyo objetivo es facilitar el flujo de información y gestión del trabajo diario en los talleres de reparación. Con este fin, la aplicación proporciona herramientas que facilitan la gestión de trabajadores, clientes, vehículos, órdenes de reparación y materiales. Además, permite administrar las jornadas productivas mediante una agenda de trabajo y calendario laboral, proporciona estadísticas del taller y permite generar automáticamente documentación de las reparaciones.

Como parte de la estancia en prácticas que se describe en este documento, la compañía busca migrar todas las funcionalidades de la aplicación original a una plataforma web, actualizando los módulos obsoletos y añadiendo un sistema de gestión de usuarios que controle el acceso al servicio y sus componentes. Por lo tanto, el presente proyecto aborda las tareas de actualizar la aplicación existente, añadiendo a las capacidades del ERP nuevas tecnologías y características que la hagan más accesible e intuitiva a los usuarios finales.

1.2. Objetivo y alcance del proyecto

El principal objetivo de este proyecto es el desarrollo de una aplicación web que replique el funcionamiento de la aplicación de escritorio **+Taller**. Este objetivo se puede desglosar en los siguientes subobjetivos:

- Estudiar la aplicación de escritorio original para identificar todos los módulos que la componen y sus funcionalidades.
- Migrar los componentes que permiten la gestión de los trabajadores, clientes, vehículos y órdenes de reparación al entorno web.
- Actualizar o reemplazar componentes obsoletos manteniendo todas las funcionalidades de la aplicación de escritorio original en la versión web.
- Diseñar e implementar un nuevo componente para la gestión de usuarios que permita el control de permisos para el acceso a los distintos sectores de la aplicación.
- Implementar una interfaz web con un diseño consistente, fácil de usar, rápido y accesible.

Este proyecto no abarca la implementación completa de la aplicación en su versión web, ya que se lograron desarrollar todos los módulos del sistema excluyendo los componentes para la gestión de las jornadas laborales (agenda de trabajo y calendario laboral), generación de estadísticas y documentación de materiales, por limitaciones de tiempo durante la realización de la estancia en prácticas.

1.3. Objetivo y alcance del producto

El objetivo de este producto —la aplicación web **+Taller**— es agilizar el flujo de trabajo de talleres mecánicos proporcionando herramientas para la gestión de las reparaciones, automatizando actividades administrativas y de trabajo diario de los talleres. Para ello la nueva aplicación debe ajustarse a los siguientes aspectos funcionales, organizativos e informáticos, que se derivan de la versión original si no se indica lo contrario.

En relación con el **alcance funcional**:

- El sistema debe permitir la gestión de usuarios, incluyendo la capacidad de listar todos los usuarios de la aplicación, así como crearlos, editarlos y eliminarlos. Se trata de un nuevo requisito de la aplicación web.
- El sistema debe permitir la gestión de roles y permisos, incluyendo la capacidad de listarlos, así como crearlos, editarlos y eliminarlos. Se debe poder asignar un único rol a cada usuario y varios permisos a cada rol que determinen las áreas de la aplicación que son accesibles para los usuarios con dicho rol. Se trata de un nuevo requisito para facilitar la gestión de usuarios.

- El sistema debe permitir la gestión de clientes, incluyendo la capacidad de listar todos los clientes de la aplicación, así como crearlos, editarlos y eliminarlos. Para cada cliente se deben mostrar sus vehículos y órdenes de reparación.
- El sistema debe permitir la gestión de vehículos, incluyendo la capacidad de listar todos los vehículos de la aplicación, así como crearlos, editarlos y eliminarlos. Para cada vehículo se deben mostrar sus órdenes de reparación.
- El sistema debe permitir la gestión de trabajadores, incluyendo la capacidad de listar todos los trabajadores de la aplicación, así como crearlos, editarlos y eliminarlos. Para cada trabajador se debe mostrar un calendario con la semana actual y las tres próximas donde sea posible registrar las horas ordinarias y extras trabajadas cada día en las tareas de chapa, pintura y mecánica.
- El sistema debe permitir la gestión de las órdenes de reparación, incluyendo la capacidad de listar todas las órdenes de la aplicación, así como crearlas, editarlas y eliminarlas.
- Para cada orden de reparación, el sistema debe:
  - Mostrar un desglose con todas las reparaciones que se deben efectuar.
  - Mostrar un desglose de los materiales necesarios para efectuar las reparaciones.
  - Mostrar un diagrama para visualizar las piezas del vehículo en las que se debe trabajar.
  - Permitir generar automáticamente la documentación del resguardo del depósito.

Con respecto al **alcance organizativo**, la aplicación está dirigida a talleres de reparaciones que buscan agilizar o automatizar procesos administrativos y unificar sus operaciones. La gestión de clientes también se integra en la aplicación, pero los proveedores quedan fuera del alcance organizativo.

Por último, en cuanto al **alcance informático**, para desarrollar la aplicación se utiliza el *framework* .NET [5] de código abierto desarrollo por Microsoft que permite desarrollar aplicaciones web siguiendo un patrón Modelo-Vista-Controlador (MVC) [6].

Para almacenar la información se utiliza una base de datos SQL generada automáticamente mediante Entity Framework [7], un *framework* de mapeo objeto-relacional (ORM) disponible para .NET que permite gestionar una base de datos sin la necesidad de que el programador genere sentencias SQL directamente, abstrayendo así la capa de acceso a los datos de la lógica de la aplicación.

1.4. Estructura de la memoria

En esta sección se describen los seis capítulos que componen esta memoria.

Para empezar, en este Capítulo 1 introductorio se describe el proyecto y el contexto en el que se ha de desarrollar. En el siguiente Capítulo 2, se detalla la planificación inicial del proyecto y se realiza un seguimiento de las tareas realizadas. En el Capítulo 3 se definen y analizan los

requisitos del sistema. Posteriormente, en el Capítulo 4, se describe el diseño de la base de datos y arquitectura del sistema. En el Capítulo 5, se detallan los resultados de la implementación, junto con los desafíos encontrados y las estrategias utilizadas para superarlos. Finalmente, en el Capítulo 6, se presenta una conclusión del proyecto. Se detallan los resultados obtenidos, posibles mejoras del producto y reflexiones personales.

Capítulo 2

Planificación del proyecto

2.1. Metodología

La metodología de desarrollo escogida para el diseño, gestión y desarrollo de este proyecto ha sido la metodología en cascada [8]. Se eligió esta metodología predictiva ya que los requisitos de la aplicación estaban claramente definidos y se tenía conocimiento previo de las tecnologías elegidas para su implementación.

Siguiendo esta metodología se separó el proyecto en las siguientes fases que se completaron de forma secuencial:

1. **Inicio**: Se realiza un estudio del ERP que se debe migrar y se instalan los programas requeridos para llevar a cabo el proyecto.
2. **Planificación**: Se establecen los objetivos, se estiman los costes y se analizan los riesgos del proyecto. A partir de esto, se decide cómo y de qué forma se llevará a cabo el proyecto.
3. **Definición**: Se definen los requisitos del proyecto y se estudian en detalle las tecnologías necesarias para llevarlo a cabo.
4. **Análisis y Diseño**: Se analiza y define la arquitectura del sistema que se va a implementar.
5. **Implementación**: Se realiza la implementación a partir de los requisitos y diseños establecidos en las fases anteriores.
6. **Cierre**: Se realizan las reuniones finales con el supervisor en la que se muestra y entrega el producto final.

2.2. Planificación temporal del proyecto

2.2.1. Estructura de Desglose del Trabajo

Para definir claramente las actividades y tareas que deben llevarse a cabo a lo largo del proyecto para cada una de las fases de la metodología, se creó una estructura de descomposición del trabajo (EDT) que se muestra en la Figura 2.1. A partir del EDT se estimó el tiempo que se tarda en realizar cada tarea y sus dependencias, estos detalles pueden verse en el cuadro 2.1.

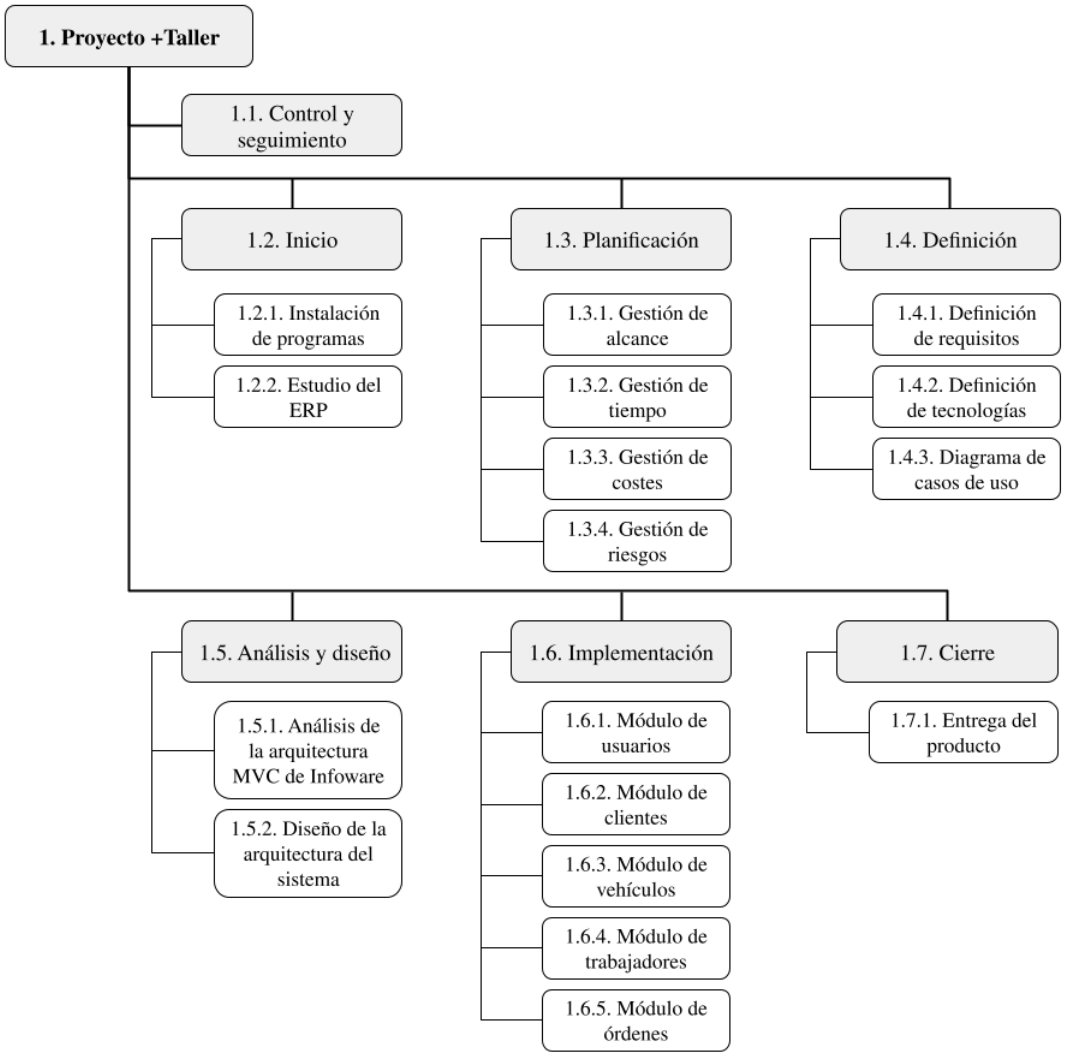


Figura 2.1: EDT del proyecto: fases y desglose de tareas.

Nivel	Identificador	Nombre	Tarea predecesora	Horas dedicadas
1	**1**	**Proyecto +Taller**		**300**
2	**1.1**	**Control y seguimiento**		**300**
2	**1.2**	**Inicio**		**15**
3	1.2.1	Instalación de programas		5
3	1.2.2	Estudio del ERP	1.2.1	10
2	**1.3**	**Planificación**	**1.2**	**20**
3	1.3.1	Gestión del alcance	1.2	5
3	1.3.2	Gestión del tiempo	1.3.1	5
3	1.3.3	Gestión de costes	1.3.2	5
3	1.3.4	Gestión de riesgos	1.3.4	5
2	**1.4**	**Definición**	**1.3**	**20**
3	1.4.1	Definición de requisitos		10
3	1.4.2	Definición de tecnologías	1.4.1	5
3	1.4.3	Diagrama de casos de uso	1.4.1	5
2	**1.5**	**Análisis y diseño**	**1.4**	**15**
3	1.5.1	Análisis de la arquitectura MVC de Infoware		5
3	1.5.2	Diseño de la arquitectura del sistema	1.5.1	10
2	1.6	**Implementación**	1.5	220
3	1.6.1	Módulo de usuarios		40
3	1.6.2	Módulo de clientes	1.6.1	30
3	1.6.3	Módulo de vehículos	1.6.2	30
3	1.6.4	Módulo de trabajadores	1.6.3	50
3	1.6.5	Módulo de órdenes de reparación	1.6.4	70
2	**1.7**	**Cierre**	**1.6**	**10**
3	1.7.1	Entrega del producto		10

Cuadro 2.1: Distribución temporal del proyecto.

Cabe destacar que la fase de "Control y seguimiento" que aparece en la figura y en el cuadro, hace referencia al proceso de seguimiento continuo que se realiza en paralelo con el proyecto, por lo tanto, no está compuesta por tareas individuales que pueden marcarse como finalizadas y se define como una tarea igual de larga que el proyecto. Este proceso abarca principalmente las reuniones con el supervisor, pero también las actividades de gestión diarias como la detección de posibles desviaciones, actualización de la planificación, y ejecución de planes de prevención y corrección, entre otras. Los resultados del seguimiento del proyecto pueden verse en la sección 2.3.

2.2.2. Diagrama de Gantt

Como herramienta para la gestión de la planificación del proyecto se ha desarrollado un Diagrama de Gantt para definir las fechas en las que se realizan las tareas identificadas y realizar así un seguimiento del desarrollo y progreso del proyecto. En la Figura 2.2 puede verse

el diagrama propuesto al inicio del proyecto.

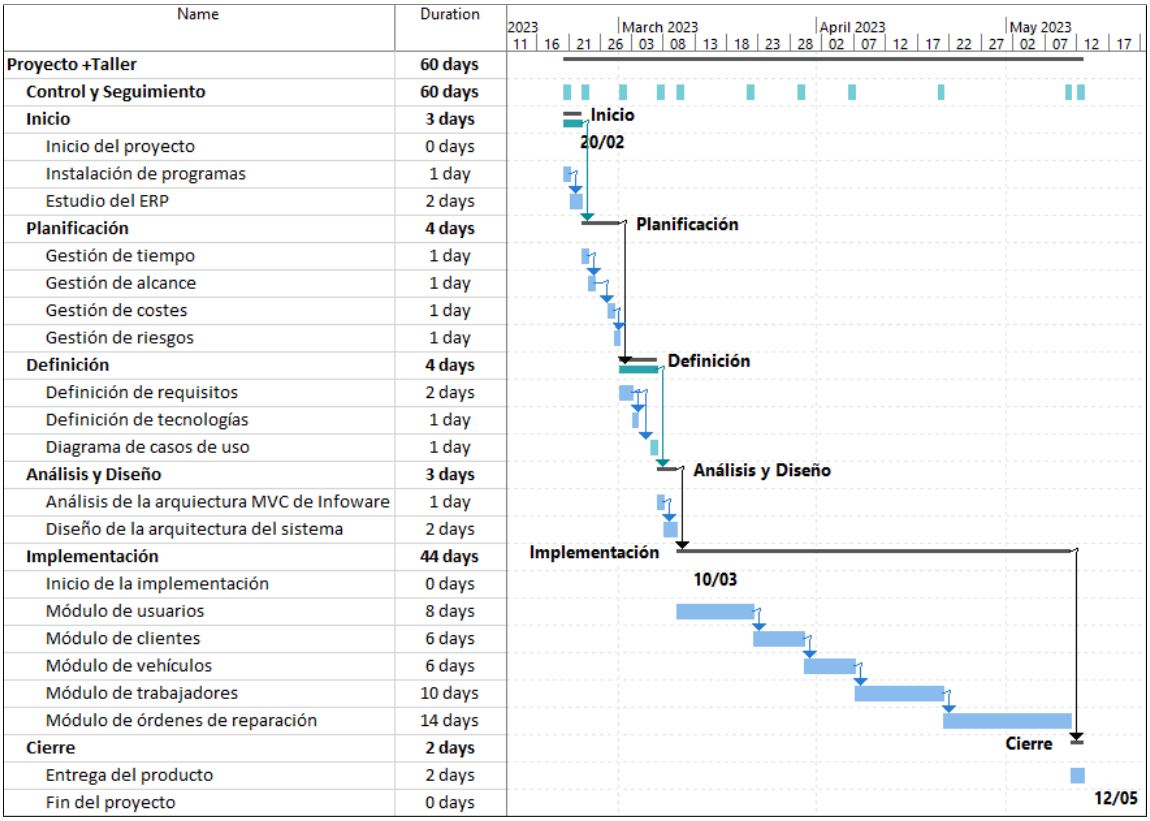


Figura 2.2: Diagrama de Gantt inicial del proyecto.

2.3. Seguimiento del proyecto

Como se mencionó anteriormente, se ha realizado un seguimiento del proyecto principalmente mediante reuniones con el supervisor. Éstas se celebraban al finalizar cada fase del proyecto y al terminar la implementación de cada módulo. En estas reuniones se revisaban los resultados obtenidos, se recibía retroalimentación y se verificaba la planificación. Las únicas desviaciones respecto a la planificación inicial tuvieron lugar en las fases de implementación y cierre. Los detalles de estas modificaciones se muestran en el cuadro 2.2.

Por último, como resultado de las reuniones periódicas de seguimiento a lo largo del proyecto, también se ha ido actualizado el diagrama de Gantt con las fechas finales del proyecto. Este diagrama puede verse en la Figura 2.3.

Id.	Nombre	Duración inicial	Duración final	Motivo de la desviación
1.6	**Implementación**	**44 días**	**45 días**	
1.6.1	Módulo de usuarios	8 días	10 días	Se ha añadido un software intermedio (*middleware*) de autentificación para controlar el acceso a los recursos por pedido del supervisor.
1.6.2	Módulo de clientes	6 días	4 días	Se ha completado la implementación del módulo antes del tiempo planificado.
1.6.5	Módulo de órdenes	14 días	15 días	Se ha aplazado la entrega para modificar el componente para el resguardo del depósito por pedido del supervisor.
1.7	**Cierre**	**2 días**	**1 día**	
1.7.1	Entrega del producto	2 días	1 día	La entrega del producto ha llevado menos tiempo del estimado, ya que no requirió la capacitación de los usuarios y el día anterior se realizó una reunión de seguimiento.

Cuadro 2.2: Desviaciones de la planificación del proyecto.

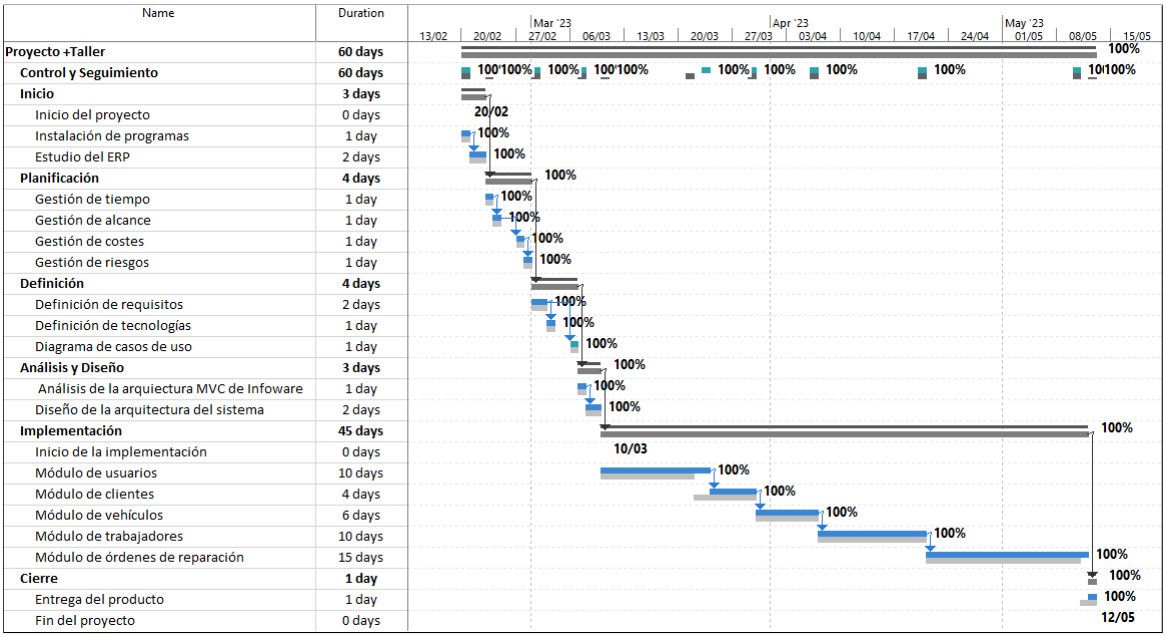


Figura 2.3: Diagrama de Gantt de seguimiento del proyecto.

2.4. Costes

Como parte del análisis para la planificación del proyecto se ha llevado a cabo una estimación de costes necesarios para llevar a cabo el proyecto:

2.4.1. Costes de los recursos humanos

Teniendo en cuenta que el salario medio de un programador en España en 2023 es de 14,62 euros por hora [9] y que el proyecto tiene una duración de 300 horas, podemos calcular el coste de recursos humanos partiendo de la base de que sólo hay un desarrollador:

$$CosteRH = HorasProyecto \cdot SalarioMedio = 300 \cdot 14,62 = 4.386\ euros.$$

Asumiendo que el coste de contratación es el 30 por ciento del valor anterior podemos obtener el coste real asociado a los recursos humanos como:

$$CosteRHReal = CosteRH \cdot 1,3 = 4.386 \cdot 1,3 = 5.701,8\ euros.$$

2.4.2. Costes de los recursos tecnológicos

En cuanto a los recursos *software*, no se han estimados costes ya que todas las aplicaciones utilizadas son de licencia gratuita. Respecto a los recursos *hardware*, se ha utilizado un ordenador HP y un monitor Samsung junto con un teclado y un ratón Logitech. Estos recursos pueden verse en más detalle en el cuadro 2.3.

Recurso	Tipo de recurso	Coste en euros
Visual Studio 2022	Software	0,00
SQL Server Management Studio 19	Software	0,00
Sourcetree	Software	0,00
Ordenador HP	Hardware	400,00
Monitor Samsung	Hardware	200,00
Teclado y ratón Logitech	Hardware	35,00
Total		**635,00**

Cuadro 2.3: Estimación de costes de recursos tecnológicos.

2.4.3. Costes indirectos

Con el fin de realizar una estimación más realista del coste total del proyecto, añadimos los costes indirectos detallados en el cuadro 2.4.

Recurso	Coste en euros
Luz y agua	100,00
Alquiler de espacio	750,00
Gastos de viaje	50,00
Suministros de oficina	20,00
Total	**920,00**

Cuadro 2.4: Estimación de costes indirectos.

2.4.4. Coste total

Mediante la suma de los costes calculados anteriormente podemos estimar el coste total del proyecto. El desglose de este puede consultarse en el cuadro 2.5.

Tipo de coste	Coste en euros
Recursos humanos	5.701,80
Recursos tecnológicos	635,00
Indirectos	920,00
Total	**7.256,80**

Cuadro 2.5: Estimación de coste total.

2.5. Riesgos

Como parte del proceso de planificación se han identificado y analizado con detenimiento seis posibles riegos que pueden afectar la probabilidad de éxito del proyecto. Para cada uno se ha realizado un análisis y se han definido planes de prevención y corrección.

Los riesgos identificados se muestran en el cuadro 2.6 y los análisis realizados para cada uno se detallan en los cuadros 2.7, 2.8, 2.9, 2.10, 2.11 y 2.12. Los planes de prevención y corrección pueden consultarse en el cuadro 2.13.

Id.	Riesgo	Tipo de Riesgo
R01	Insuficiente tiempo de desarrollo	Riesgo del proyecto
R02	Bajas en el equipo de desarrollo	Riesgo del proyecto
R03	Insuficientes reuniones de seguimiento	Riesgo del proyecto
R04	Problemas de rendimiento y tiempo de respuesta	Riesgo del producto
R05	Incorrecto diseño o implementación que dificultan la incorporación de nuevos módulos	Riesgo del producto
R06	Inadecuada comprensión de los requisitos del sistema ERP existente	Riesgo del proyecto

Cuadro 2.6: Riesgos del proyecto.

Riesgo	R01. Insuficiente tiempo de desarrollo
Magnitud	Alta.
Probabilidad	Baja.
Descripción	Como el proyecto se realiza en un contexto de prácticas este tiene un límite inmovible de 300 horas.
Impacto	La falta de tiempo de desarrollo puede ocasionar que el producto final no cumpla todos los requisitos especificados o no tenga la calidad esperada.
Identificadores	Diferencias entre el desarrollo real del proyecto y la planificación estimada.

Cuadro 2.7: Análisis del riesgo R01 - Insuficiente tiempo de desarrollo.

Riesgo	R02. Bajas en el equipo de desarrollo
Magnitud	Alta.
Probabilidad	Media.
Descripción	Uno o varios miembros del equipo no pueden trabajar en el proyecto.
Impacto	Supone un retraso en la planificación que, de no poder ser compensado dentro del tiempo definido, puede hacer que el proyecto fracase.
Identificadores	Aviso de ausencia de un miembro del equipo.

Cuadro 2.8: Análisis del riesgo R02 - Bajas en el equipo de desarrollo.

Riesgo	R03. Insuficientes reuniones de seguimiento
Magnitud	Alta.
Probabilidad	Baja.
Descripción	No se realizan suficientes reuniones de seguimiento o no se le dedica el tiempo necesario para realizar una evaluación adecuada.
Impacto	El equipo no está lo suficientemente comunicado, lo que puede dar lugar a malentendidos, menor capacidad para identificar y resolver problemas, dificultades en la toma de decisiones y retrasos en el proyecto.
Identificadores	Incumplimiento de las reuniones planificadas.

Cuadro 2.9: Análisis del riesgo R03 - Insuficientes reuniones de seguimiento.

Riesgo	R04. Problemas de rendimiento y tiempo de respuesta
Magnitud	Alta.
Probabilidad	Media.
Descripción	La aplicación tiene problemas de rendimiento que ocasionan tiempos de respuestas lentos.
Impacto	El producto final no tiene la calidad esperada en términos de rendimiento lo que afecta negativamente la experiencia del usuario.
Identificadores	Se reciben informes de usuarios mencionando problemas de rendimiento.

Cuadro 2.10: Análisis del riesgo R04 - Problemas de rendimiento y tiempo de respuesta.

Riesgo	R05. Incorrecto diseño o implementación que dificultan la incorporación de nuevos módulos
Magnitud	Alta.
Probabilidad	Baja.
Descripción	El sistema no ha sido diseñado o implementado correctamente, lo que dificulta la incorporación del resto de módulos no desarrollados en este proyecto.
Impacto	Se deben realizar esfuerzos adicionales para refactorizar el sistema, lo cual se retrasa la implementación de nuevos componentes, alarga el tiempo de desarrollo y provoca perdidas.
Identificadores	Dificultades a la hora de añadir funcionalidades provocando continuas refactorizaciones.

Cuadro 2.11: Análisis del riesgo R05 - Incorrecto diseño o implementación que dificultan la incorporación de nuevos módulos.

Riesgo	R06. Inadecuada comprensión de los requisitos del sistema ERP existente
Magnitud	Alta.
Probabilidad	Media.
Descripción	No se ha analizado correctamente el ERP existente lo que provoca una inadecuada comprensión de los requisitos funcionales.
Impacto	Una incorrecta comprensión de los requisitos funcionales puede dar lugar a un diseño o una implementación inadecuados.
Identificadores	Conocimiento limitado del sistema existente.

Cuadro 2.12: Análisis del riesgo R06 - Inadecuada comprensión de los requisitos del sistema ERP existente.

Id.	Plan de Prevención	Plan de Corrección
R01	Se realizan seguimientos regulares para detectar cualquier posible retraso.	Se priorizan las funcionalidades críticas y se ajusta el alcance del proyecto.
R02	Se mantiene una comunicación continua para resolver cualquier problema que pueda afectar la planificación del proyecto. Se preparan herramientas para la comunicación remota entre los miembros del equipo.	Se reasignan el trabajo priorizando las tareas críticas y los plazos se modifican si es necesario. De ser posible, se trabaja remotamente.
R03	Se establecen reuniones regulares desde el inicio del desarrollo y se define una agenda se lo que se debe tratar en cada reunión.	Se realizan reuniones adicionales para abordar las dudas pendientes y comunicar el estado del proyecto.
R04	Se diseña y desarrolla el sistema siguiendo prácticas de programación eficientes. También se utilizan herramientas de monitoreo para identificar y resolver cualquier problema de rendimiento rápidamente.	Se añaden tareas de optimización a la planificación del proyecto para resolver los problemas de rendimiento detectados.
R05	Se diseña el sistema de forma modular y flexible teniendo en cuenta los requisitos de la aplicación. También se siguen buenas prácticas de diseño y desarrollo de software.	Se revisa el diseño del sistema y se realizan tareas de refactorización hasta conseguir un sistema flexible. De ser necesario, se modifica el alcance y se actualiza la planificación del proyecto.
R06	Se establecen reuniones de seguimiento con el supervisor para recibir retroalimentación de los requisitos establecidos y las funcionalidades implementadas.	Se realizan reuniones adicionales para ajustar los requisitos de la aplicación y modificar la planificación de ser necesario.

Cuadro 2.13: Planes de prevención y corrección de los riesgos identificados.

Capítulo 3

Análisis del sistema

3.1. Definición de casos de uso

En esta sección se definen y analizan los requisitos del sistema, que parten necesariamente del análisis de la aplicación original. Para empezar, se ha realizado el Diagrama de Casos de Uso (DCU) del proyecto que puede verse en las Figuras 3.1 y 3.2, con el fin de plasmar los escenarios en los que los distintos tipos de usuarios interactúan con el sistema. Las descripciones de los casos de usos resultantes pueden verse en el cuadro 3.1.

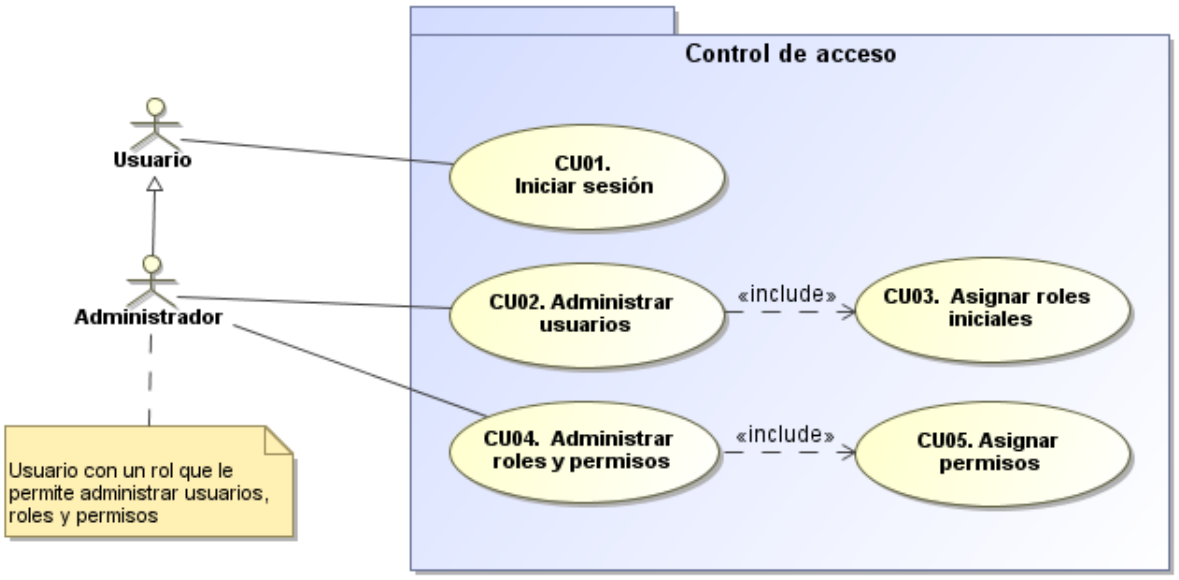


Figura 3.1: DCU del sistema de control de acceso del proyecto.

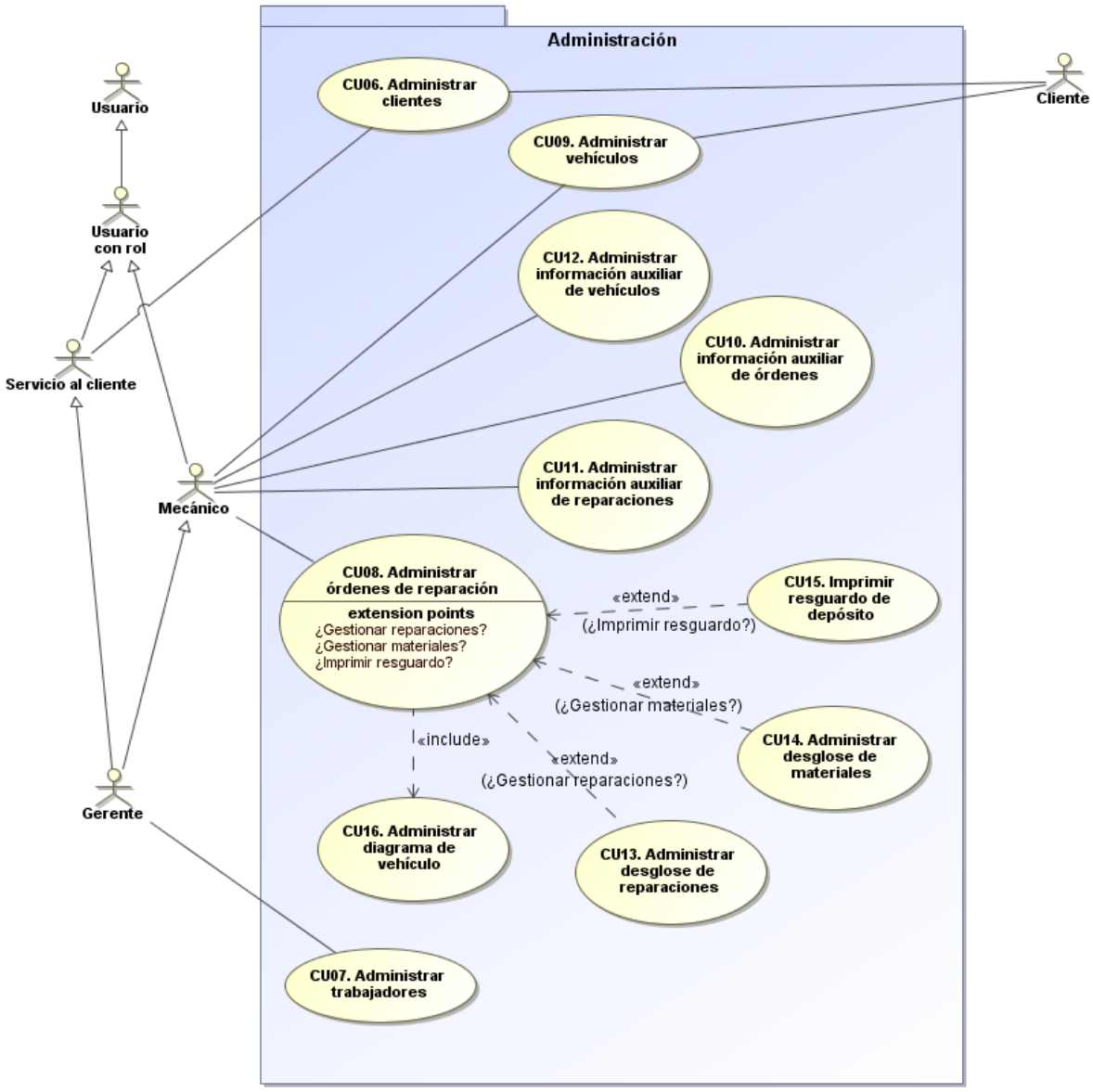


Figura 3.2: DCU de las funcionalidades administrativas del proyecto.

Id.	Nombre	Descripción
CU01	Iniciar sesión	Este caso de uso representa el sistema de autenticación que permite a los usuarios registrados iniciar sesión y proporciona acceso a determinados recursos del sistema en función de sus credenciales.
CU02	Administrar usuarios	Este caso de uso representa las funcionalidades que permiten la gestión de usuarios: listarlos, crearlos, editarlos y eliminarlos.

Cuadro 3.1 continúa de la página anterior.

Id.	Nombre	Descripción
CU03	Asignar roles iniciales	Este caso de uso representa el proceso de asignación de un rol inicial al usuario recién creado para determinar a qué áreas de la aplicación tienen acceso.
CU04	Administrar roles y permisos	Este caso de uso representa las funcionalidades que permiten la gestión de roles y permisos: listarlos, crearlos, editarlos y eliminarlos.
CU05	Asignar permisos	Este caso de uso representa el proceso de asignación de permisos a roles para determinar a qué áreas de la aplicación tienen acceso los usuarios con cada rol.
CU06	Administrar clientes	Este caso de uso representa las funcionalidades que permiten la gestión de clientes: listarlos, crearlos, editarlos y eliminarlos.
CU07	Administrar trabajadores	Este caso de uso representa las funcionalidades que permiten la gestión de trabajadores: listarlos, crearlos, editarlos y eliminarlos.
CU08	Administrar órdenes de reparación	Este caso de uso representa las funcionalidades que permiten la gestión de órdenes de reparación: listarlos, crearlos, editarlos y eliminarlos.
CU09	Administrar vehículos	Este caso de uso representa las funcionalidades que permiten la gestión de vehículos: listarlos, crearlos, editarlos y eliminarlos.
CU10	Administrar información auxiliar de órdenes	Este caso de uso representa las funcionalidades que permiten la gestión de aseguradoras, cantidades de combustible, tipos de órdenes: listarlos, crearlos, editarlos y eliminarlos. Estos datos son utilizados durante el proceso de administración de órdenes.
CU11	Administrar información auxiliar de reparaciones	Este caso de uso representa las funcionalidades que permiten la gestión de piezas y tipos de reparaciones: listarlos, crearlos, editarlos y eliminarlos. Estos datos son utilizados durante el proceso de administración de reparaciones y materiales.
CU12	Administrar información auxiliar de vehículos	Este caso de uso representa las funcionalidades que permiten la gestión de colores, marcas y modelos de vehículos: listarlos, crearlos, editarlos y eliminarlos. Estos datos son utilizados durante el proceso de administración de vehículos.
CU13	Administrar desglose de reparaciones	Este caso de uso representa las funcionalidades que permiten la gestión de reparaciones: listarlos, crearlos, editarlos y eliminarlos.
CU14	Administrar desglose de materiales	Este caso de uso representa las funcionalidades que permiten la gestión de materiales: listarlos, crearlos, editarlos y eliminarlos.

Cuadro 3.1 continúa de la página anterior.

Id.	Nombre	Descripción
CU15	Imprimir resguardo de depósito	Este caso de uso representa la funcionalidad de generar automáticamente documentación sobre el resguardo de depósito de cada orden.
CU16	Administrar diagrama de vehículo	Este caso de uso representa la interacción con un diagrama de un vehículo para señalar las partes con las que se trabaja durante la reparación.

Cuadro 3.1: Casos de uso del proyecto.

A partir del DCU, se han realizado las especificaciones de los casos de uso identificados. Este proceso permite formalizar los requisitos funcionales del sistema, ayudando a su comprensión y facilitando su posterior implantación. Las especificaciones más relevantes pueden verse en los cuadros 3.2, 3.3, 3.4 y 3.5.

Especificación de CU02	
Nombre	Administrar usuarios.
Autor	Felicitas Agüero Pratto.
Versión	1.0
Fuente	+Taller.
Descripción	El sistema permite listar todos los usuarios de la aplicación, así como crearlos, editarlos y eliminarlos. Estas acciones generalmente son realizadas por un administrador.
Actores	Administrador: usuario con permisos para la gestionar la información personal y las cuentas de los usuarios.
Relaciones	CU03. Asignar roles.
Precondición	Estar autentificado en el sistema.
Disparador	El administrador navega a la página de gestión de usuarios.
Secuencia normal	1. El administrador navega a la página de gestión de usuarios. 2. Se muestra una lista con los usuarios registrados en el sistema, junto con la opción de crear un nuevo usuario y editar uno existente. 3. Si decide crear un nuevo usuario es redirigido a un formulario donde debe rellenar todos los campos obligatorios. 4. Si decide editar un usuario existente es redirigido a un formulario con los datos de la cuenta seleccionada y la opción de eliminarlo. 5. El administrador envía el formulario, se validan los datos y se modifica la información almacenada en el sistema.
Secuencia alternativa	4.1. Si el administrador decide eliminar al usuario se le muestra un mensaje para confirmar o cancelar la acción. 4.2. Si decide eliminarlo, se realiza la acción y es redirigido a la lista de usuarios. 4.3. Si decide cancelar la acción, el mensaje desaparece y se muestra el formulario anterior con los datos del usuario.
Postcondición	Se ha añadido o modificado un usuario según el criterio del administrador.

Cuadro 3.2 continúa de la página anterior.

Especificación de CU02	
Excepciones	1. Si el administrador no tiene los permisos suficientes es redirigido a la página de inicio. 5. Si los datos enviados no son válidos se muestran mensajes de error debajo de cada dato incorrecto con los requisitos que se deben completar para que sean aceptados.
Frecuencia esperada	Cinco veces al día.
Importancia	Importante.
Prioridad	Alta.
Comentarios	-

Cuadro 3.2: Especificación de CU02 - Administrar usuarios.

Especificación de CU05	
Nombre	Asignar permisos.
Autor	Felicitas Agüero Pratto.
Versión	1.0
Fuente	+Taller.
Descripción	El sistema permite asignar permisos a los roles existentes para gestionar el acceso a los recursos del sistema. Estas acciones generalmente son realizadas por un administrador para gestionar la seguridad de la aplicación.
Actores	Administrador: usuario con permisos para la gestionar la asignación de permisos.
Relaciones	CU16. Administrar roles y permisos.
Precondición	Estar autentificado en el sistema.
Disparador	El administrador navega a la página de asignaciones y decide añadir un nuevo permiso a un rol existente.
Secuencia normal	1. El administrador navega a la página de asignaciones y decide añadir un nuevo permiso a un rol existente. 2. Se muestra una lista con todos los roles registrados en el sistema y todas las vistas disponibles. 3. El administrador selecciona la combinación deseada. 4. El administrador envía el formulario, se validan los datos y se modifica la información almacenada en el sistema.
Secuencia alternativa	-
Postcondición	Los usuarios con el rol seleccionado tienen acceso un nuevo recurso del sistema.
Excepciones	1. Si el usuario no tiene los permisos suficientes es redirigido a la página de inicio. 4. Si el rol seleccionado ya tiene asignado el permiso seleccionado aparece un mensaje de error.
Frecuencia esperada	Dos veces a la semana.

Cuadro 3.3 continúa de la página anterior.

Especificación de CU05	
Importancia	Importante.
Prioridad	Alta.
Comentarios	-

Cuadro 3.3: Especificación de CU05 - Asignar permisos.

Especificación de CU08	
Nombre	Administrar órdenes de reparación.
Autor	Felicitas Agüero Pratto.
Versión	1.0
Fuente	+Taller.
Descripción	El sistema permite listar todas las órdenes de reparación, así como crearlas, editarlas y eliminarlas.
Actores	Mecánico y Gerente: usuarios con permisos para gestionar las órdenes de reparación.
Relaciones	CU12. Imprimir resguardo de depósito. CU13. Administrar desglose de reparaciones. CU14. Administrar desglose de materiales.
Precondición	Estar autentificado en el sistema.
Disparador	El usuario navega a la página de las órdenes de reparación.
Secuencia normal	1. El usuario navega a la página las órdenes de reparación. 2. Se muestra una lista con las órdenes registradas en el sistema, junto con la opción de crear una nueva orden y editar una existente. 3. Si decide crear una nueva orden es redirigido a un formulario donde debe rellenar todos los campos obligatorios. 4. Si decide editar una orden existente es redirigido a un formulario con los datos de la orden seleccionada, la opción de eliminarla y la opción de imprimir el resguardo del depósito. Asimismo, se muestra un diagrama con las partes del vehículo afectadas y un desglose de las reparaciones y los materiales necesarios para la orden. 5. El usuario envía el formulario, se validan los datos y se modifica la información almacenada en el sistema.
Secuencia alternativa	4.1. Si el usuario decide eliminar la orden de reparación, se le muestra un mensaje para confirmar o cancelar la acción. 4.2. Si decide eliminarlo, se realiza la acción y es redirigido a la lista de órdenes. 4.3. Si decide cancelar la acción, el mensaje desaparece y se muestra el formulario anterior con los datos de la orden.
Postcondición	Se ha añadido o modificado una orden según el criterio del usuario.
Excepciones	1. Si el usuario no tiene los permisos suficientes es redirigido a la página de inicio. 5. Si los datos enviados no son válidos se muestran mensajes de error debajo de cada dato incorrecto con los requisitos que se deben completar para que sean aceptados.

Cuadro 3.4 continúa de la página anterior.

Especificación de CU08	
Frecuencia esperada	Treinta veces al día.
Importancia	Vital.
Prioridad	Alta.
Comentarios	- Las órdenes de reparación de un cliente también pueden ser administradas desde su página de edición en la sección de administración de clientes. - Las órdenes de reparación de un vehículo también pueden ser administradas desde su página de edición en la sección de administración de vehículos.

Cuadro 3.4: Especificación de CU08 - Administrar órdenes de reparación.

Especificación de CU16	
Nombre	Administrar diagrama de vehículo.
Autor	Felicitas Agüero Pratto.
Versión	1.0
Fuente	+Taller.
Descripción	El sistema permite interactuar con un diagrama de un vehículo con el fin de marcar las piezas con las que se tiene que trabajar en la orden de reparación.
Actores	Mecánico y Gerente: usuarios con permisos para gestionar las órdenes de reparación.
Relaciones	CU08. Administrar órdenes de reparación
Precondición	Estar autentificado en el sistema.
Disparador	El usuario navega a la página de las órdenes de reparación y selecciona una orden de la lista mostrada.
Secuencia normal	1. El usuario navega a la página de las órdenes de reparación y selecciona una orden de la lista mostrada. 2. Se muestra un diagrama con las distintas piezas y componentes de un vehículo. 3. El usuario selecciona una parte del diagrama. 4. Si la parte esta desactivada, se activa y cambia su color a rojo. 5. Si la parte esta activada, se desactiva y cambia su color a negro. 6. El sistema actualiza la información del diagrama de acuerdo con la interacción del usuario con las partes.
Secuencia alternativa	4.1. Si el usuario decide eliminar la orden de reparación, se le muestra un mensaje para confirmar o cancelar la acción. 4.2. Si decide eliminarlo, se realiza la acción y es redirigido a la lista de órdenes. 4.3. Si decide cancelar la acción, el mensaje desaparece y se muestra el formulario anterior con los datos de la orden.
Postcondición	Se las partes del vehículo con las que ha interactuado el usuario muestran su nuevo estado.

Cuadro 3.5 continúa de la página anterior.

Especificación de CU16	
Excepciones	1. Si el usuario no tiene los permisos suficientes es redirigido a la página de inicio.
Frecuencia esperada	Quince veces a la semana.
Importancia	Poco importante.
Prioridad	Media.
Comentarios	-

Cuadro 3.5: Especificación de CU16 - Administrar diagrama de vehículo.

Además, para completar el proceso de elicitación de requisitos mediante el Diagrama de Casos de Uso, se han identificado los requisitos de datos junto con los requisitos de calidad del sistema para garantizar una aplicación optimizada, accesible y segura. Los cuadros 3.6, 3.7, 3.8, 3.9 ,3.10,3.11, 3.12, 3.13 y 3.14 muestran los requisitos de datos más importantes y los cuadros 3.15, 3.16 y 3.17 los de requisitos de calidad más destacables.

Requisito de datos RD01	
Nombre	User
Autor	Felicitas Agüero Pratto.
Versión	1.0
Fuente	+Taller.
Requisitos asociados	CU02. Administrar usuarios.
Datos específicos	Identificador de usuario, identificador de rol, nombre, apellidos, correo electrónico, contraseña, comentarios, estado (activo o inactivo) y número de teléfono.
Ocurrencia media	Cinco veces al día.
Ocurrencia máxima	Cincuenta veces al día.
Importancia	Alta.

Cuadro 3.6: Requisito de datos RD01 - User.

Requisito de datos RD02	
Nombre	Client
Autor	Felicitas Agüero Pratto.
Versión	1.0
Fuente	+Taller.
Requisitos asociados	CU06. Administrar clientes.
Datos específicos	Identificador de cliente, DNI, nombre, apellidos, correo electrónico, teléfono, teléfono secundario, dirección, provincia, población, código postal y fecha de alta.
Ocurrencia media	Diez veces al día.
Ocurrencia máxima	Cien veces al día.
Importancia	Alta.

Cuadro 3.7: Requisito de datos RD02 - Client.

Requisito de datos RD03	
Nombre	Vehicle
Autor	Felicitas Agüero Pratto.
Versión	1.0
Fuente	+Taller.
Requisitos asociados	CU09. Administrar vehículos.
Datos específicos	Identificador de vehículo, matrícula, número de chasis, identificador de cliente, marca, modelo, color, código de color y fecha de alta.
Ocurrencia media	Diez veces al día.
Ocurrencia máxima	Cien veces al día.
Importancia	Alta.

Cuadro 3.8: Requisito de datos RD03 - Vehicle.

Requisito de datos RD04	
Nombre	UserType
Autor	Felicitas Agüero Pratto.
Versión	1.0
Fuente	+Taller.
Requisitos asociados	CU04. Administrar roles y permisos.
Datos específicos	Identificador de rol y nombre.
Ocurrencia media	Dos veces a la semana.
Ocurrencia máxima	Veinte veces a la semana.
Importancia	Alta.

Cuadro 3.9: Requisito de datos RD04 - UserType.

Requisito de datos RD05	
Nombre	View
Autor	Felicitas Agüero Pratto.
Versión	1.0
Fuente	+Taller.
Requisitos asociados	CU04. Administrar roles y permisos.
Datos específicos	Identificador de permiso y nombre de recurso.
Ocurrencia media	Dos veces a la semana.
Ocurrencia máxima	Veinte veces a la semana.
Importancia	Alta.

Cuadro 3.10: Requisito de datos RD05 - View.

Requisito de datos RD06	
Nombre	Employee
Autor	Felicitas Agüero Pratto.
Versión	1.0
Fuente	+Taller.
Requisitos asociados	CU07. Administrar trabajadores.
Datos específicos	Identificador de trabajador, código de trabajador, DNI, nombre, apellidos, horas diarias ordinarias y extras trabajando en chapa, horas diarias ordinarias y extras trabajando en pintura y horas diarias ordinarias y extras trabajando en mecánica.
Ocurrencia media	Diez veces al día.
Ocurrencia máxima	Cien veces al día.
Importancia	Alta.

Cuadro 3.11: Requisito de datos RD06 - Employee.

Requisito de datos RD07	
Nombre	Order
Autor	Felicitas Agüero Pratto.
Versión	1.0
Fuente	+Taller.
Requisitos asociados	CU08. Administrar órdenes de reparación.
Datos específicos	Identificador de orden, número de orden, identificador de cliente, identificador de vehículo, tipo de orden, aseguradora, kilómetros del vehículo, cantidad de combustible, descripción, fecha de llegada y salida del vehículo, diagrama asociado, estado (continúa o ha terminado), precio por hora y precio diario de la estancia.
Ocurrencia media	Treinta veces al día.
Ocurrencia máxima	Cien veces al día.
Importancia	Alta.

Cuadro 3.12: Requisito de datos RD07 - Order.

Requisito de datos RD08	
Nombre	RepairBreakdown
Autor	Felicitas Agüero Pratto.
Versión	1.0
Fuente	+Taller.
Requisitos asociados	CU13. Administrar desglose de reparaciones.
Datos específicos	Identificador de reparación, identificador de orden, identificador de trabajador asignado, tipo de reparación, pieza, fecha, coste por hora y horas.
Ocurrencia media	Quince veces al día.
Ocurrencia máxima	Cien veces al día.
Importancia	Alta.

Cuadro 3.13: Requisito de datos RD08 - RepairBreakdown.

Requisito de datos RD09	
Nombre	Material
Autor	Felicitas Agüero Pratto.
Versión	1.0
Fuente	+Taller.
Requisitos asociados	CU14. Administrar desglose de materiales.
Datos específicos	Identificador de material, identificador de orden, pieza, y precio por pieza.
Ocurrencia media	Quince veces al día.
Ocurrencia máxima	Doscientas veces al día.
Importancia	Alta.

Cuadro 3.14: Requisito de datos RD09 - Material.

Requisito de datos RC01	
Nombre	Acceso seguro a recursos mediante inicio de sesión.
Autor	Felicitas Agüero Pratto.
Versión	1.0
Fuente	+Taller.
Requisitos asociados	CU01. Iniciar sesión.
Descripción	El ERP debe controlar el acceso a los recursos de la aplicación mediante una página de inicio de sesión segura.
Importancia	Alta.

Cuadro 3.15: Requisito de calidad RC01 - Acceso seguro a recursos mediante inicio de sesión.

Requisito de datos RC02	
Nombre	Interfaz de usuario adaptable.
Autor	Felicitas Agüero Pratto.
Versión	1.0
Fuente	+Taller.
Requisitos asociados	-
Descripción	El ERP debe ser capaz de adaptar la información mostrada a distintos tamaños de ventana, garantizando la legibilidad y visibilidad de la aplicación.
Importancia	Alta.

Cuadro 3.16: Requisito de calidad RC02 - Interfaz de usuario adaptable.

Requisito de datos RC03	
Nombre	Experiencia de usuario consistente.
Autor	Felicitas Agüero Pratto.
Versión	1.0
Fuente	+Taller.
Requisitos asociados	-
Descripción	El ERP debe garantizar una experiencia de usuario consistente en términos de interfaz, navegación y funcionalidades presentadas en las distintas áreas de gestión.
Importancia	Alta.

Cuadro 3.17: Requisito de calidad RC03 - Experiencia de usuario consistente.

3.2. Modelo conceptual

Por último, utilizando toda la información recolectada de los requisitos se ha definido el modelo conceptual del sistema que puede verse en la Figura 3.3. Este modelo servirá como base para el posterior diseño del sistema.

Este modelo está compuesto de nueve clases separadas en cuatro grupos según su funcionalidad. En primer lugar, en azul pueden verse las clases encargadas de la administración de usuarios. Cada usuario tiene un único rol que puede ser compartido con otros usuarios y cada rol puede tener varios permisos que determinan las áreas a las que tienen acceso.

En segundo lugar, las clases en amarillo permiten la gestión de los clientes y sus vehículos. Un cliente puede tener varios vehículos, pero un vehículo solo pertenece a un cliente.

En tercer lugar, en rojo se muestran las clases necesarias para la administración de las órdenes de reparación. Una orden pertenece a un único cliente y vehículo, pero estos pueden tener varias órdenes a su nombre. Una orden está compuesta de varias reparaciones y puede requerir varios materiales para llevarlas a cabo.

Por último, la clase verde permite la gestión de los trabajadores. Cada trabajador puede estar asignado a diversas reparaciones, pero sólo un empleado se encarga de cada reparación.

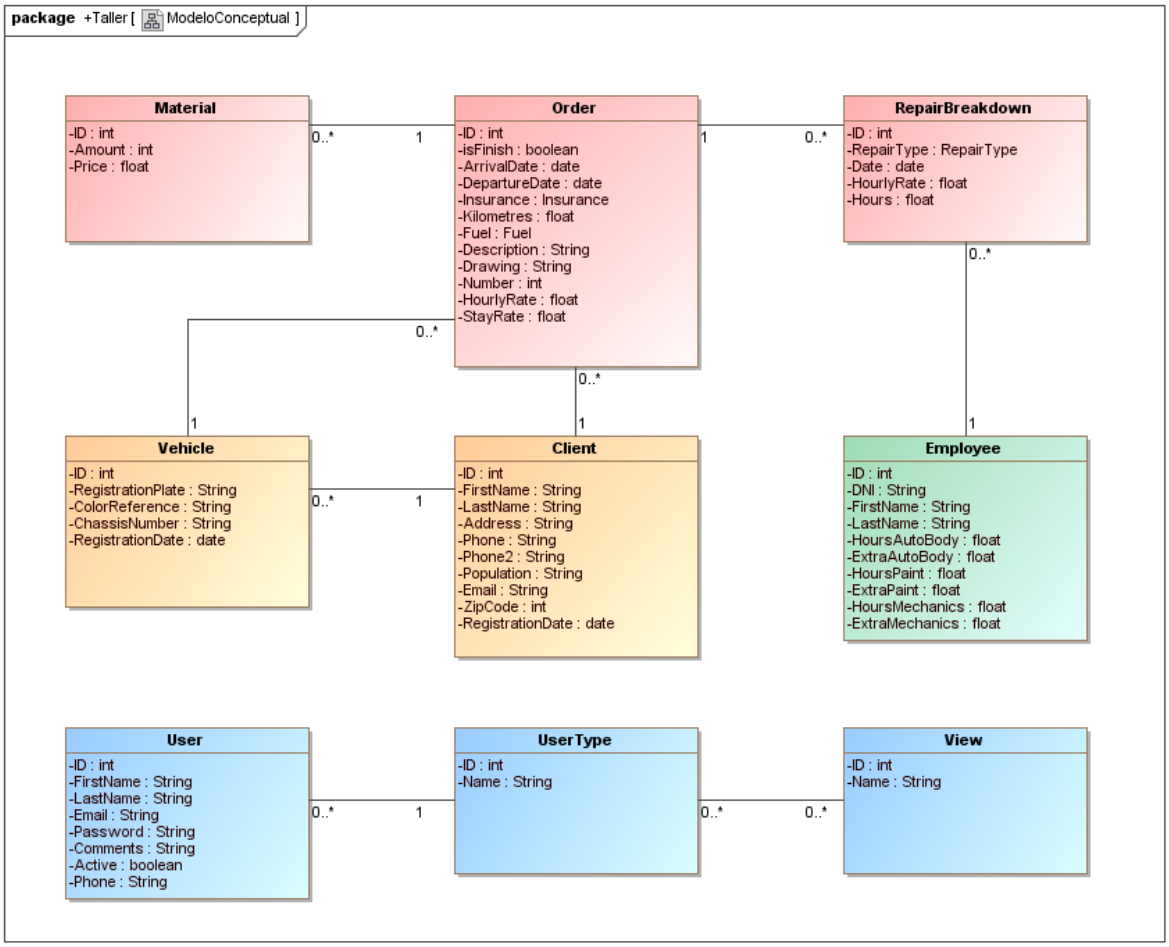


Figura 3.3: Modelo conceptual del sistema.

Capítulo 4

Diseño del sistema

4.1. Diseño de la base de datos

Como parte de los requisitos del sistema, la aplicación debe mantener la misma estructura de la base de datos original, pero debe seguir una estrategia de "código primero" utilizando Entity Framework (EF). Este enfoque de "código primero" significa que primero se programan los modelos de entidad de la aplicación y, a partir de ellos, se crea la base de datos mediante un proceso automatizado.

EF genera y actualiza automáticamente la base de datos a partir de una clase de contexto creada utilizando los modelos de entidad de la aplicación [10]. Esta herramienta EF genera dinámicamente sentencias SQL para gestionar y consultar la base de datos a partir de consultas realizadas en .NET y, como resultado, traduce los registros devueltos en objetos de los modelos [11]. La Figura 4.1 muestra una ilustración del funcionamiento descrito.

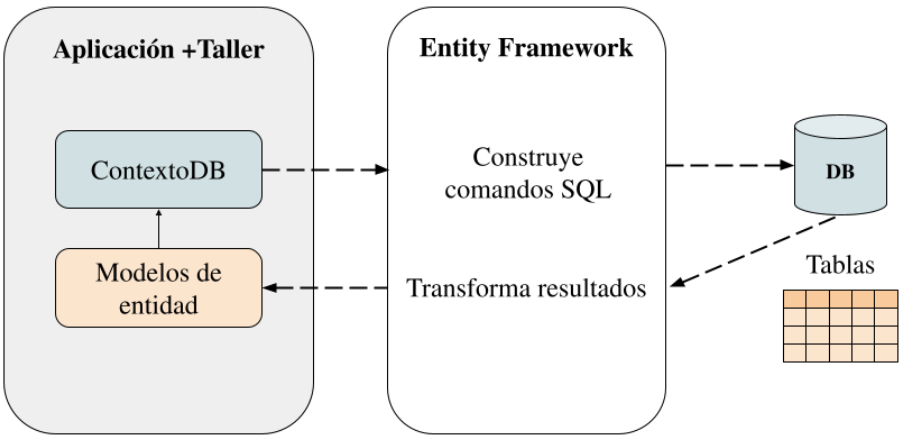


Figura 4.1: Ilustración del funcionamiento de Entity Framework.

El diagrama de clases utilizado para la generación de la base de datos puede verse en la Figura 4.2, el cual es una evolución mucho más detallada del modelo conceptual del capítulo 3. Cabe destacar que este diagrama agrega las clases auxiliares necesarias para la administración de los módulos definidos anteriormente. Por ejemplo, se añadieron las clases de *Brands* y *Models* que permiten especificar la marca y el modelo de un vehículo, respectivamente. Las clases auxiliares están definidas con el prefijo "AUX" en la base de datos.

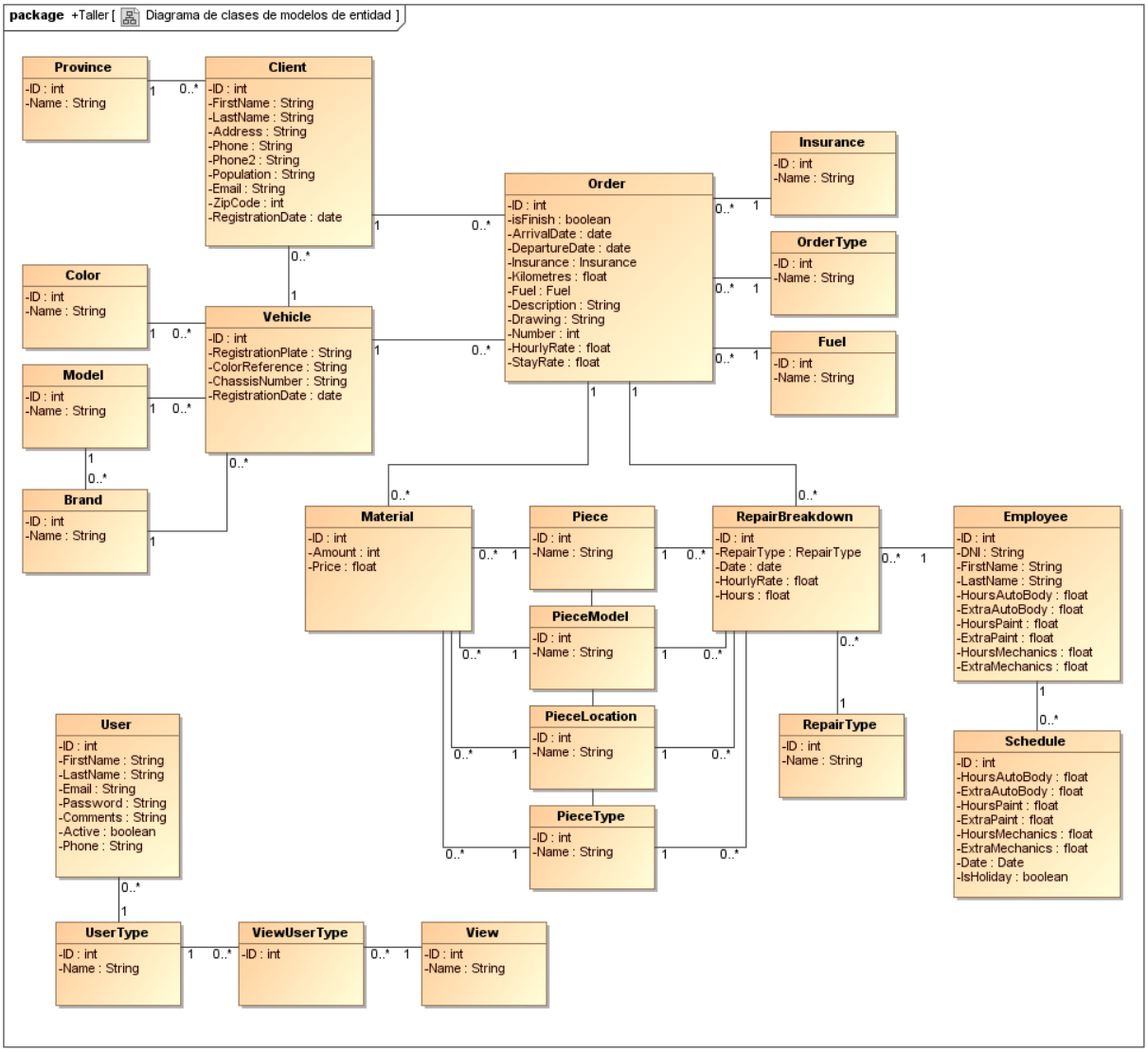


Figura 4.2: Diagrama de clases final de los modelos de entidad del proyecto.

Para clarificar el proceso de transformación que realiza la herramienta EF, en la Figura 4.3, se muestra la implementación del modelo de entidad *User* y el código generado por EF para la creación de la tabla *Users* en la base de datos. En este ejemplo pueden apreciarse entre corchetes las anotaciones necesarias de los atributos que forman parte del modelo de entidad *User*. EF utiliza esta información contextual para generar de forma automática la correspondiente sentencia SQL de creación de tablas.

```csharp
[Index(nameof(Email), IsUnique = true)]
public class User
{
    public int ID { get; set; }

    [Required]
    public int UserTypeID { get; set; }
    public virtual UserType UserType { get; set; }

    [Required]
    [MaxLength(50)]
    public string FirstName { get; set; }

    [Required]
    [MaxLength(100)]
    public string LastName { get; set; }

    [Required]
    [MaxLength(100)]
    public string Email { get; set; }

    [Required]
    [MaxLength(10)]
    public string Password { get; set; }

    public string? Comments { get; set; }

    [Required]
    public bool Active { get; set; }

    [MaxLength(20)]
    public string? Phone { get; set; }
}
```

(a)

```sql
CREATE TABLE [Users] (
    [ID] int NOT NULL IDENTITY,
    [UserTypeID] int NOT NULL,
    [FirstName] nvarchar(50) NOT NULL,
    [LastName] nvarchar(100) NOT NULL,
    [Email] nvarchar(100) NOT NULL,
    [Password] nvarchar(10) NOT NULL,
    [Comments] nvarchar(max) NULL,
    [Active] bit NOT NULL,
    [Phone] nvarchar(20) NULL,
    CONSTRAINT [PK_Users] PRIMARY KEY ([ID]),
    CONSTRAINT [FK_Users_UserTypes_UserTypeID] FOREIGN KEY ([UserTypeID])
        REFERENCES [UserTypes] ([ID]) ON DELETE CASCADE
);
GO

CREATE UNIQUE INDEX [IX_Users_Email] ON [Users] ([Email]);
GO
```

(b)

Figura 4.3: (a) Modelo de entidad *User* y (b) SQL generado por EF para la creación de la tabla *Users*.

Tras el proceso de trasformación, la base de datos resultante puede verse en las Figuras 4.4, 4.5 y 4.6.

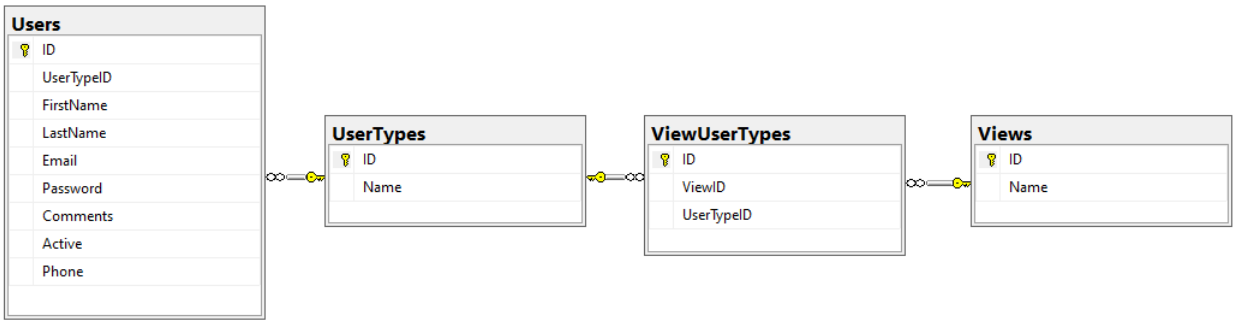


Figura 4.4: Diagrama de bases de datos bloque 1. Usuarios, roles y permisos.

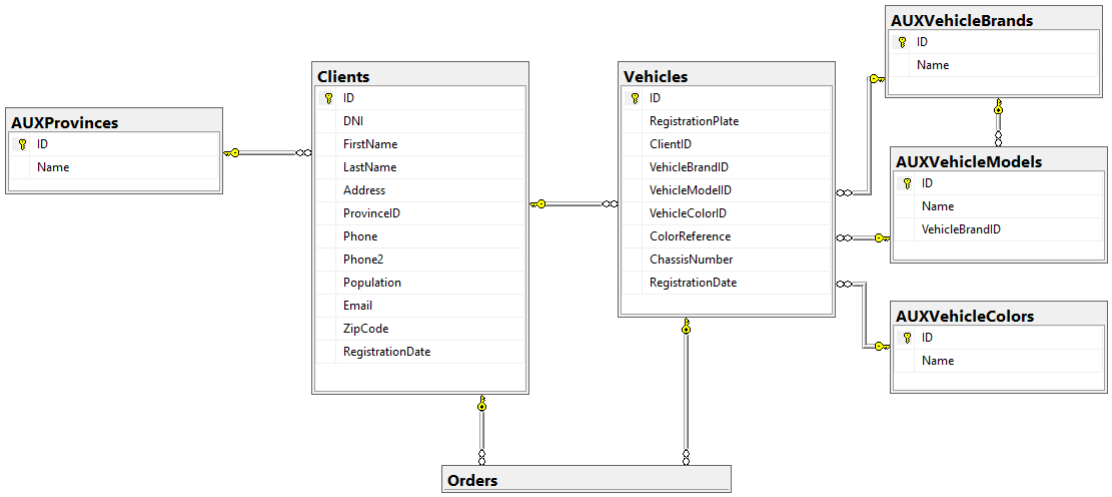


Figura 4.5: Diagrama de bases de datos bloque 2. Clientes y vehículos.

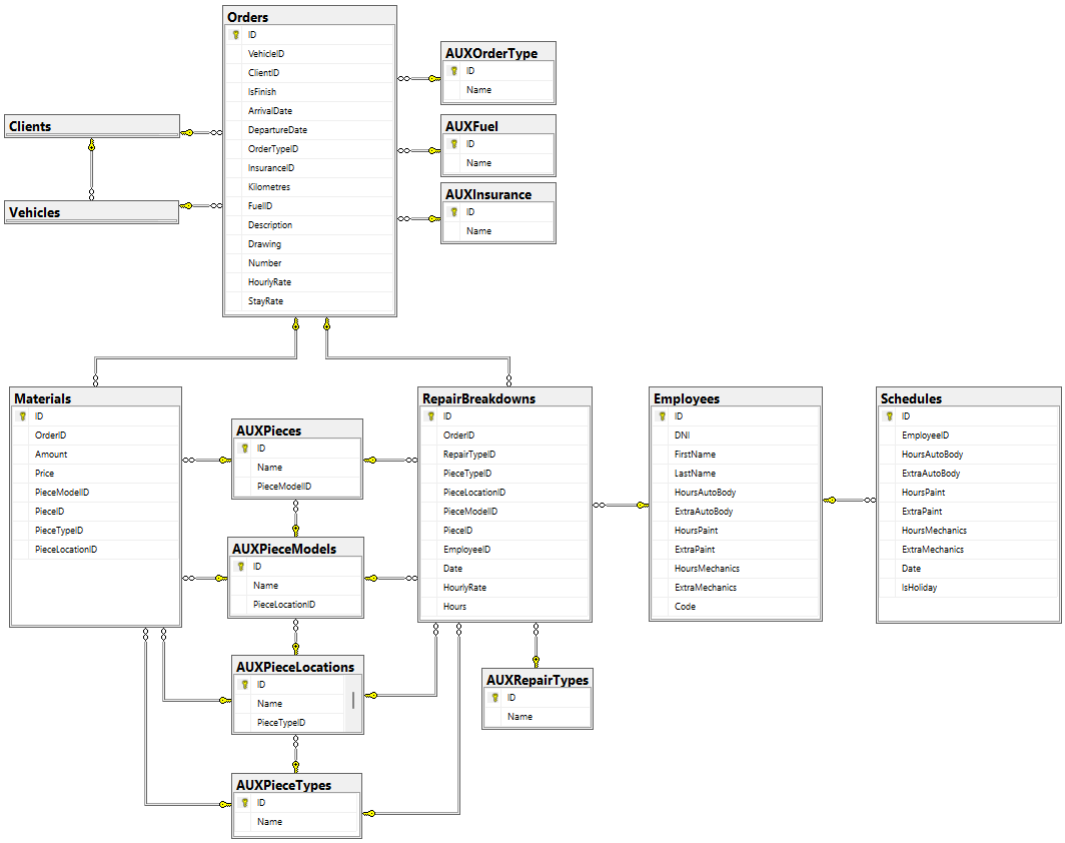


Figura 4.6: Diagrama de bases de datos bloque 3. Órdenes, materiales, reparaciones y trabajadores.

4.2. Diseño de la arquitectura del sistema

A la hora de diseñar la arquitectura del sistema, se ha seguido la arquitectura en capas utilizada actualmente en las aplicaciones web de Infoware. Esta arquitectura de n-niveles propone separar la aplicación en cinco capas independientes: presentación, aplicación, negocio, acceso a datos y datos. Un diagrama de la arquitectura propuesta aparece en la Figura 4.7.

Cada una de estas capas se encarga de las siguientes tareas:

- Capa de presentación: representa las interfaces de usuario de la aplicación. Se encarga de mostrar la información de la aplicación y comunica los datos introducidos por el usuario a la capa de aplicaciones mediante JSON.
- Capa de aplicaciones: representa los controladores de la aplicación. Es la encargada de gestionar las peticiones de la capa de presentación, llamar a los servicios correspondientes para llevarlas a cabo y, si es necesario, enviar una respuesta en formato JSON.
- Capa de negocio: representa los servicios de la aplicación. Esta capa contiene la lógica de

negocio, es decir, los algoritmos necesarios para llevar a cabo los procesos de la aplicación. Con este fin, se comunica con los repositorios correspondientes para consultar información y/o almacenar los resultados de sus operaciones.

- Capa de acceso a datos: representa los repositorios de la aplicación. Proporciona una capa de abstracción entre la lógica de negocio y el almacenamiento de datos. En esta capa se utiliza Entity Framework para realizar operaciones CRUD (crear, leer, actualizar y borrar información) en la base de datos.
- Capa de datos: representa la base de datos SQL de la aplicación. Se encarga de almacenar y manipular la información de la aplicación a partir de las consultas generadas por Entity Framework.

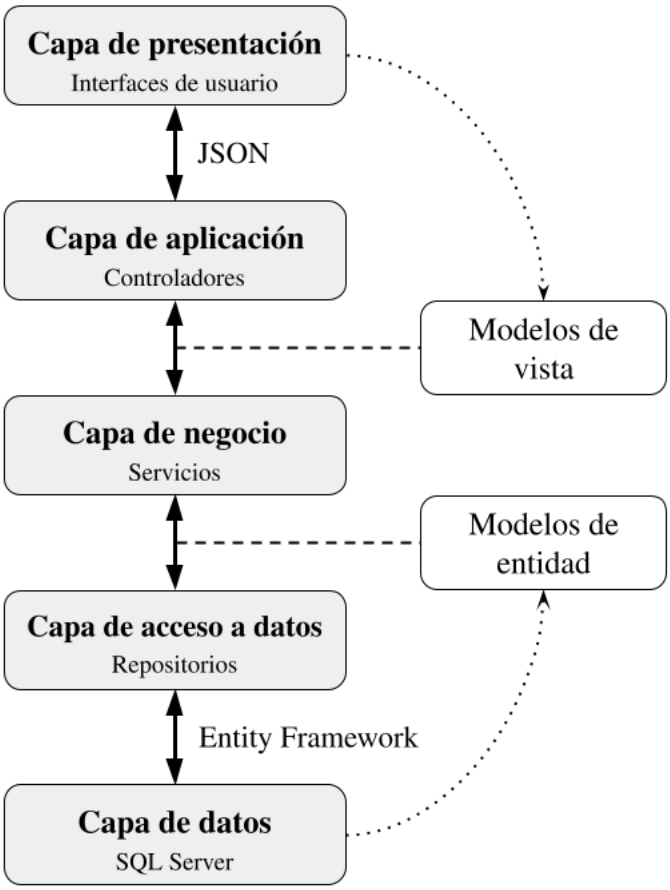


Figura 4.7: Diagrama de la arquitectura del sistema.

Como se ha visto arriba, la definición de modelos son cruciales para permitir la correcta comunicación y paso de información entre las capas de presentación, negocio, y de acceso a datos En particular, se identifican dos tipos de modelos:

- Modelos de entidad: representan la estructura y las relaciones de las tablas de la base de

datos. La base de datos es creada y actualizada a partir de las modificaciones realizadas en los modelos mediante Entity Framework.

- Modelos de vista: representan la información requerida por las vistas, pueden contener los datos de varios modelos. Transfieren información entre las vistas y los controladores.

Esta separación de modelos se debe principalmente al *framework* utilizado para la aplicación. La arquitectura .NET MVC (*Model-View-Controller*) permite pasar un único modelo fuertemente tipado del controlador a la vista [12] haciendo necesaria la creación de un modelo de vista si se quiere mostrar la información de varios modelos.

De esta forma, las capas de presentación y aplicación trabajan con modelos de vista cuando es necesario mostrar la información de un modelo en las interfaces de usuario, mientras que las capas de negocio y acceso a datos trabajan con los modelos de entidad ya que representan la estructura de los datos como están almacenados en la base de datos. Es importante destacar que el proceso de mapeo para pasar de un modelo de entidad a un modelo de vista o viceversa se realiza en las clases de servicios y no en los controladores para seguir el principio de separación de intereses de la arquitectura.

Como ejemplo para ilustrar la diferencia entre los modelos, la Figura 4.8 muestra arriba el modelo de entidad de vehículo utilizado para generar la tabla *Vehicles* en la base de datos y abajo el modelo de vista utilizado para listar a los vehículos en las interfaces de usuario.

Además, como se mencionó anteriormente, la base de datos se crea y actualiza a partir de los modelos de entidad, por lo que se debe evitar su modificación a menos que sea absolutamente necesario. Si un nuevo atributo es necesario para el correcto funcionamiento de una vista, se debe modificar el modelo de vista correspondiente.

4.3. Diseño de las interfaces

Para el diseño de las interfaces se ha adaptado una plantilla dada por la empresa. Debido a esto, muchos aspectos del diseño general de la página, como la tipografía u organización de los componentes, ya estaban definidos.

Sin embargo, para algunas funcionalidades más complejas fuera del marco proporcionado por la plantilla, se han añadido nuevos componentes o se han adaptados los proporcionados. En concreto, supuso modificar el código de las ventanas modales para permitir mostrar tablas y diagramas interactivos, así como modificar las entradas de datos para mostrar dinámicamente los campos obligatorios.

A continuación, en las Figuras 4.9, 4.10 y 4.11, se pueden observar los diseños finales junto con las interfaces originales de la aplicación de escritorio.

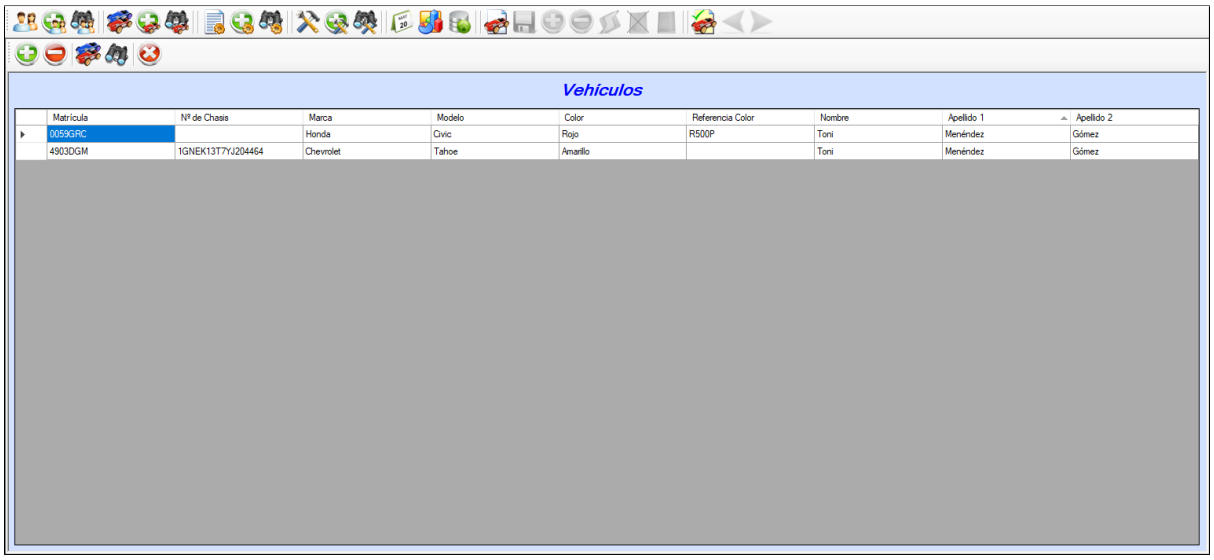
```
[Index(nameof(RegistrationPlate), IsUnique = true)]
public class Vehicle
{
    public int ID { get; set; }

    [Required]
    [MaxLength(50)]
    public string RegistrationPlate { get; set; }

    [Required]
    public int ClientID { get; set; }
    public virtual Client Client { get; set; }

    public int? VehicleBrandID { get; set; }
    public virtual VehicleBrand VehicleBrand { get; set; }

    public int? VehicleModelID { get; set; }
    public virtual VehicleModel VehicleModel { get; set; }

    public int? VehicleColorID { get; set; }
    public virtual VehicleColor VehicleColor { get; set; }

    [MaxLength(50)]
    public string? ColorReference { get; set; }

    [MaxLength(50)]
    public string? ChassisNumber { get; set; }

    public DateTime? RegistrationDate { get; set; }
}
```

(a)

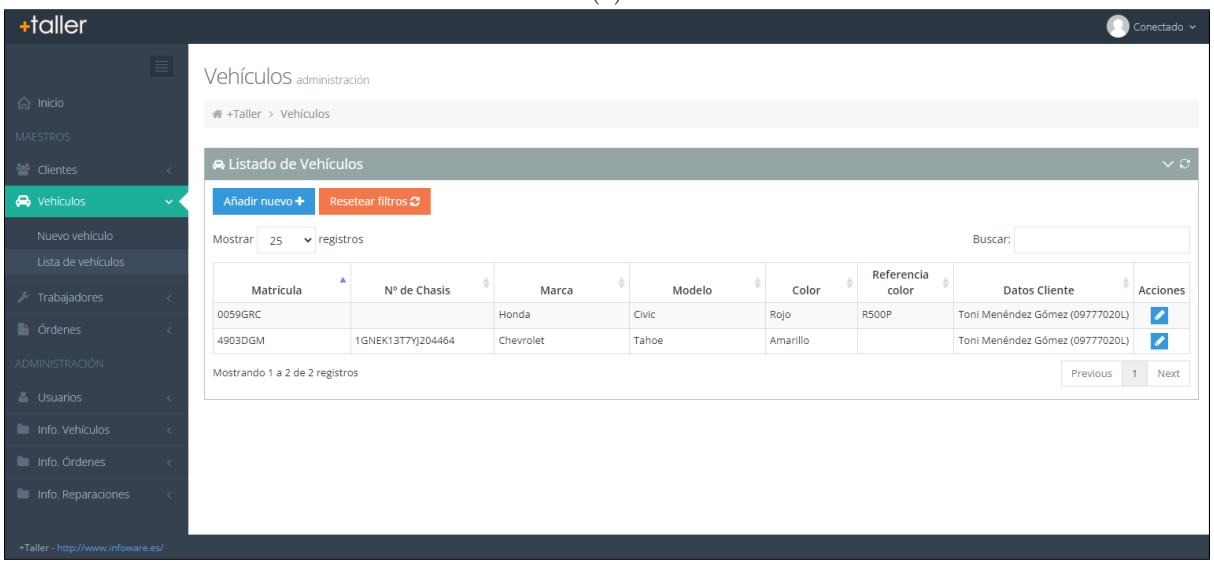
```
public class VehiclesListVM
{
    public int ID { get; set; }

    [Display(Name = "Matrícula")]
    public string RegistrationPlate { get; set; }

    [Display(Name = "Nº de Chasis")]
    public string ChassisNumber { get; set; }

    [Display(Name = "Marca")]
    public string VehicleBrandID { get; set; }

    [Display(Name = "Modelo")]
    public string VehicleModelID { get; set; }

    [Display(Name = "Color")]
    public string VehicleColorID { get; set; }

    [Display(Name = "Referencia Color")]
    public string ColorReference { get; set; }

    [Display(Name = "Datos Cliente")]
    public string ClientID { get; set; }
}
```

(b)

Figura 4.8: (a) Modelo de entidad *Vehicle* y (b) modelo de vista *VehiclesListVM* para el listado de vehículos.

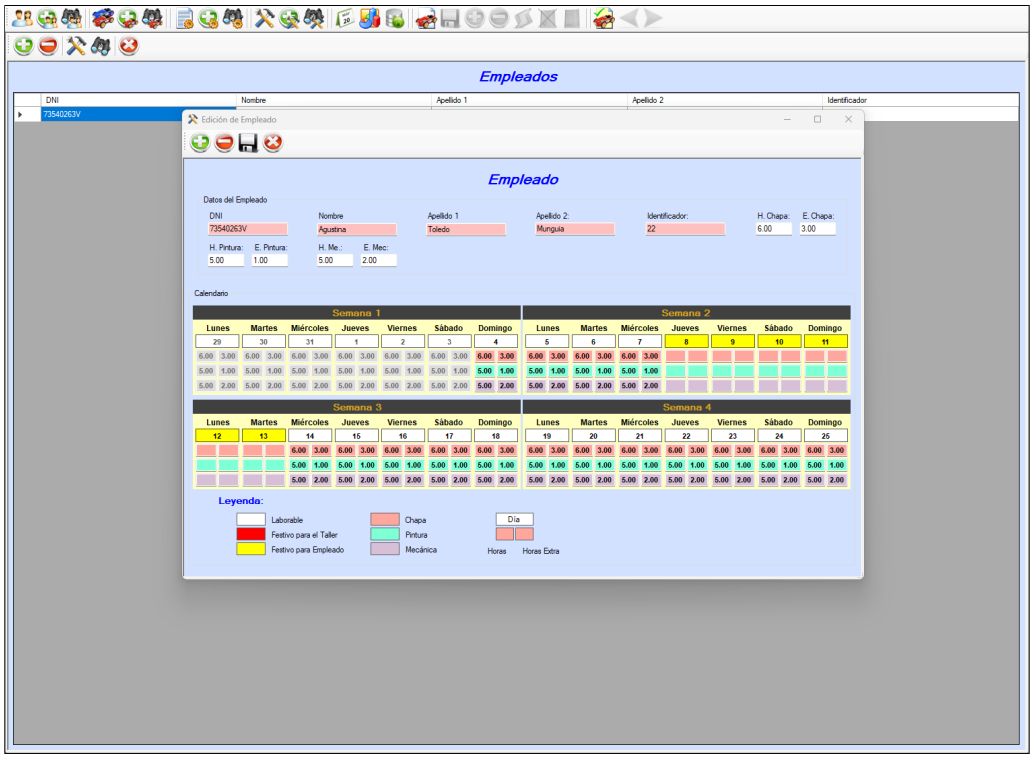


(a)

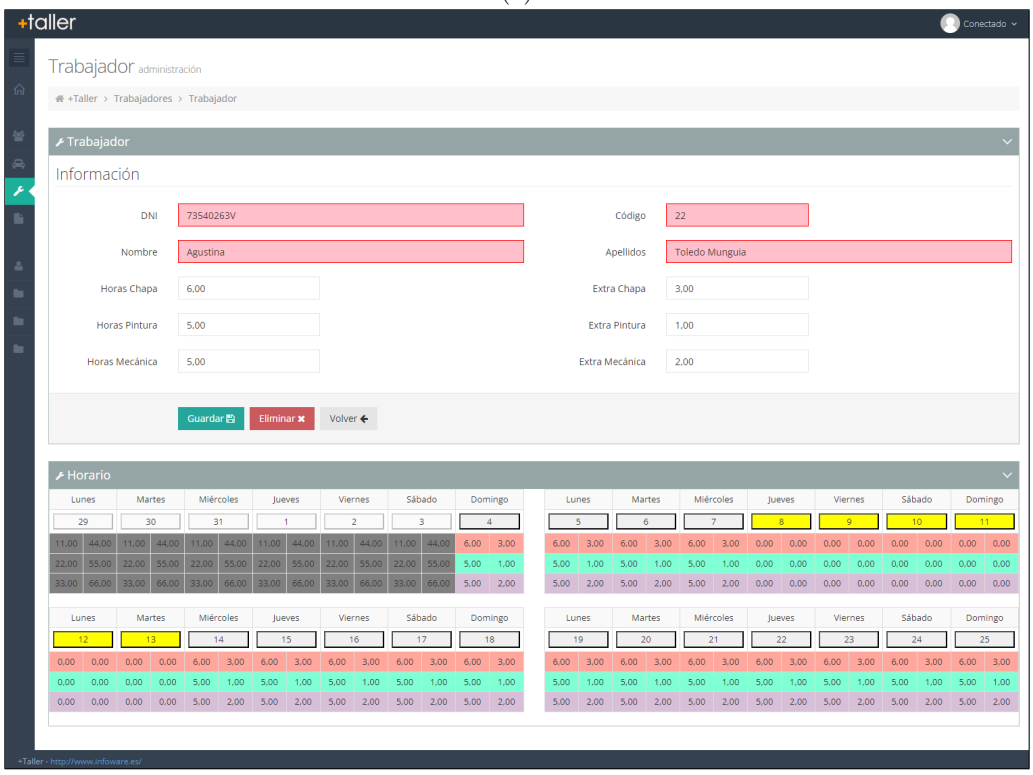


(b)

Figura 4.9: Interfaz del listado de vehículos de (a) la aplicación ERP original y (b) el proyecto de migración.

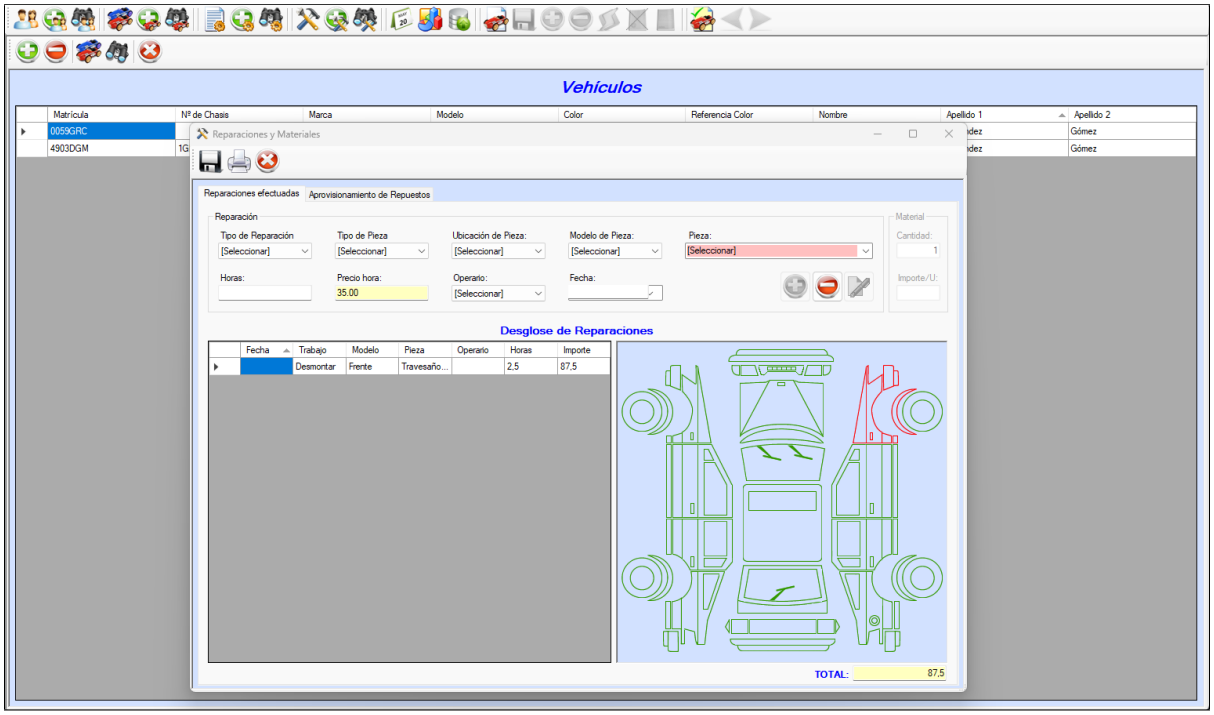


(a)

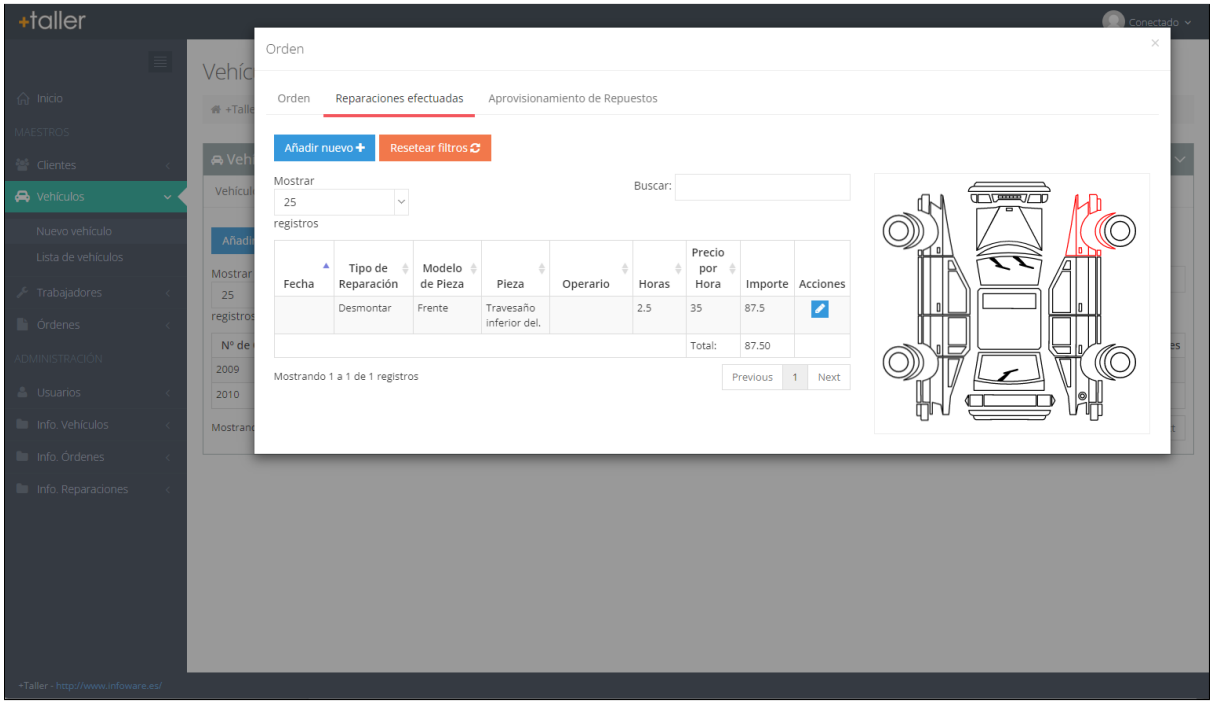


(b)

Figura 4.10: Interfaz del formulario de trabajadores de (a) la aplicación ERP original y (b) el proyecto de migración.



(a)



(b)

Figura 4.11: Interfaz de las órdenes de reparación de los vehículos de (a) la aplicación ERP original y (b) el proyecto de migración.

Capítulo 5

Implementación y pruebas

5.1. Estructura del código

El proyecto se ha organizado siguiendo el estándar de Infoware. Esta estructura, como puede verse en la Figura 5.1, separa las clases en las siguientes carpetas:

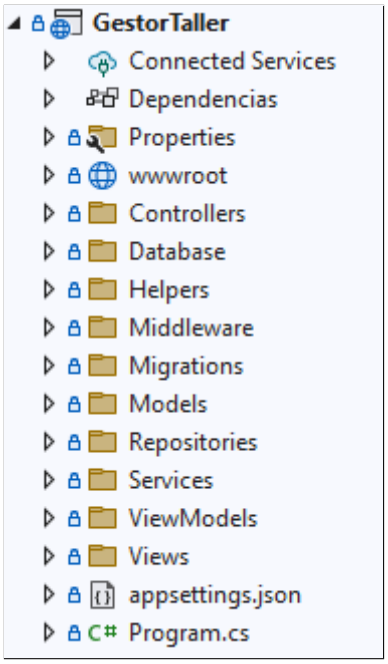


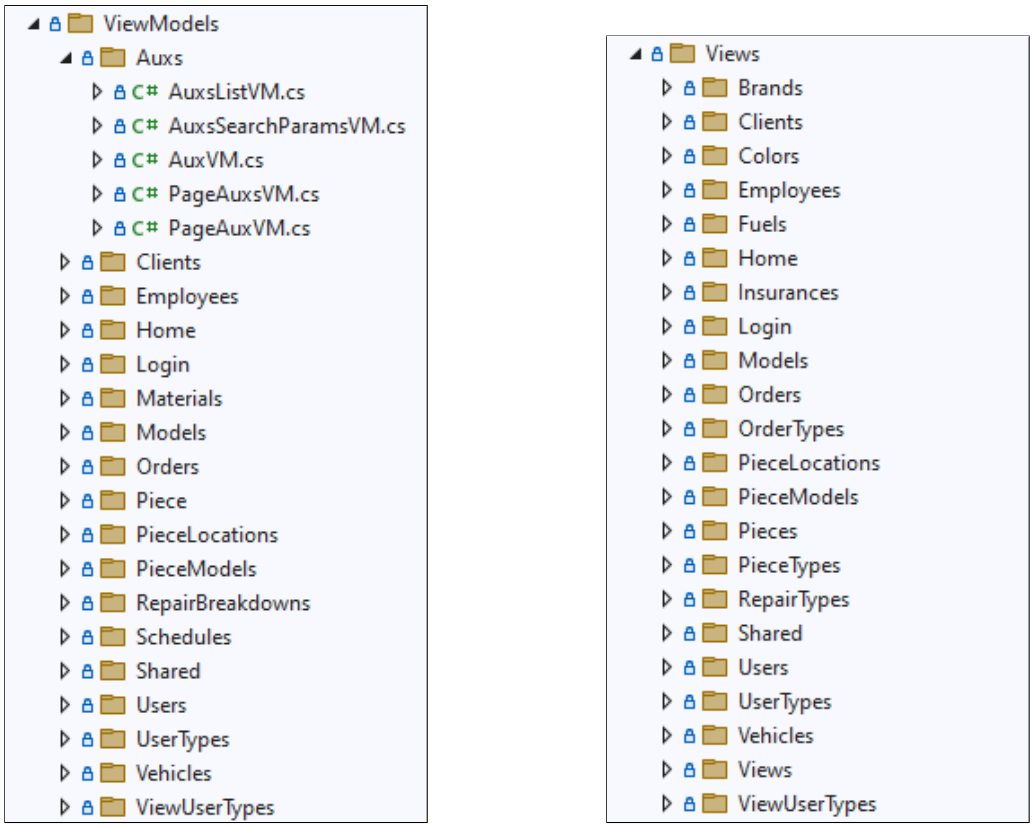
Figura 5.1: Estructura del código del proyecto.

- *Controllers*: Contiene los controladores responsables de gestionar las solicitudes originadas de las vistas. Cada controlador gestiona las solicitudes originadas únicamente en las vistas se su subcarpeta en *Views*.
- *Database*: Contiene los ficheros encargados de traducir las solicitudes de los repositorios

mediante Entity Framework (EF) para que puedan ser ejecutadas en la base de datos.

- *Helpers*: Contiene clases con métodos auxiliares utilizados en varias partes de la aplicación.
- *Middleware*: Contiene las clases necesarias para la implementación del software intermedio que controla los permisos de los usuarios cuando acceden a los distintos recursos del sistema.
- *Migrations*: Contiene los ficheros generados por EF para la gestión de las tablas de la base de datos. Muestran los cambios realizados de forma cronológica y permiten la creación de la base de datos en distintos entornos.
- *Models*: Contiene los modelos de entidad del sistema. Estos modelos son los utilizados por EF para la creación de las tablas en la base de datos.
- *Repositories*: Contiene los repositorios encargados de generar las solicitudes para modificar o acceder a la información almacenada en la base de datos.
- *Services*: Contienen los servicios encargados encapsular la lógica de negocio.
- *ViewModel*: Contienen los modelos de vista, necesarios para mostrar los datos del sistema en los usuarios. Como se puede ver en la Figura 5.2a, la mayoría de los modelos de entidad tienen los siguientes cinco tipos de modelos de vistas:
  - *VM*: Almacena los atributos del modelo de entidad junto con cualquier campo auxiliar necesario para mostrar su información de forma accesible para los usuarios.
  - *ListVM*: Almacena los datos de los objetos cuando son mostrados como filas de una tabla.
  - *SearchParamsVM*: Almacena la información que permite gestionar la tabla del modelo, así como realizar búsquedas y determinar el orden en el que se deben mostrar los datos.
  - *PageVM*: Utilizado en las vistas de modificación y creación de datos. Almacenan el modelo junto con cualquier campo auxiliar necesario para la modificación de datos.
  - *PagesVM*: Utilizado en las vistas de listado de datos. Almacena un grupo de modelos junto con cualquier campo auxiliar necesario para mostrar correctamente los datos.
- *Views*: Contiene las vistas de la aplicación que gestionan la presentación de los datos y las interacciones de los usuarios. Como se puede ver en la Figura 5.2b, contiene una subcarpeta para cada controlador del sistema con las vistas que cada uno de estos deben gestionar.

Los ficheros y las carpetas restantes son generados internamente por .NET para la gestión de la aplicación web.



(a) (b)

Figura 5.2: Estructura del código de (a) los modelos de vista y (b) las vistas del proyecto.

5.2. Descripción técnica de la implementación

A lo largo este proyecto, se desarrollaron veintiún controladores, quince servicios y diecisiete repositorios. Estas clases permiten el listado, creación, edición y eliminación de veintiún tipos de datos, como por ejemplo clientes, vehículos, trabajadores, órdenes de reparación, usuarios, roles de usuario, pantallas, permisos, marcas de vehículos, y modelos de vehículos, por nombrar los más relevantes.

A continuación, en las secciones 5.2.1 y 5.2.2, se describen en más detalle dos de los retos que surgieron durante el desarrollo y las estrategias adoptadas para solucionarlos. En particular, estos dos retos fueron los más destacables, ya que ofrecieron las mayores oportunidades de aprendizaje al involucraban nuevas estructuras y tecnologías.

5.2.1. Unidad de trabajo

La clase de contexto de la base de datos permite a los repositorios de la aplicación realizar operaciones CRUD en la base de datos que luego son traducidas en consultas SQL mediante Entity Framework (EF).

El problema surge cuando una operación de un servicio implica modificar varias tablas de la base datos, es decir, se utilizan varios repositorios a lo largo del procedimiento. Si cada repositorio utiliza una instancia distinta del contexto o la operación CRUD de uno de ellos falla, pueden generar problemas de coherencia e integridad.

Para solucionar esto se crea una unidad de trabajo. Esta clase se encarga, en primer lugar, de crear y almacenar la única instancia del contexto que luego distribuye entre los repositorios. Para ello, inyecta la dependencia en los constructores de todos los repositorios de la aplicación que también se encarga de almacenar.

En segundo lugar, proporciona un método *Commit* que permite a los servicios señalar cuando terminan cada procedimiento, es decir, cuando terminan de realizar operaciones CRUD. De esta forma, la unidad de trabajo puede aplicar todos los cambios realizados o revertirlos si alguno de ellos falla.

Cabe destacar que la unidad tiene el beneficio adicional de crear una capa de abstracción entre la lógica de negocio y la capa de acceso a los datos, ya que todos los servicios deben pasar por la unidad de trabajo para acceder a los repositorios [13]. Esto contribuye a la separación de intereses de la arquitectura facilitando la automatización de pruebas unitarias.

En la Figura 5.3 muestra un diagrama ilustrando la implementación de la unidad de trabajo.

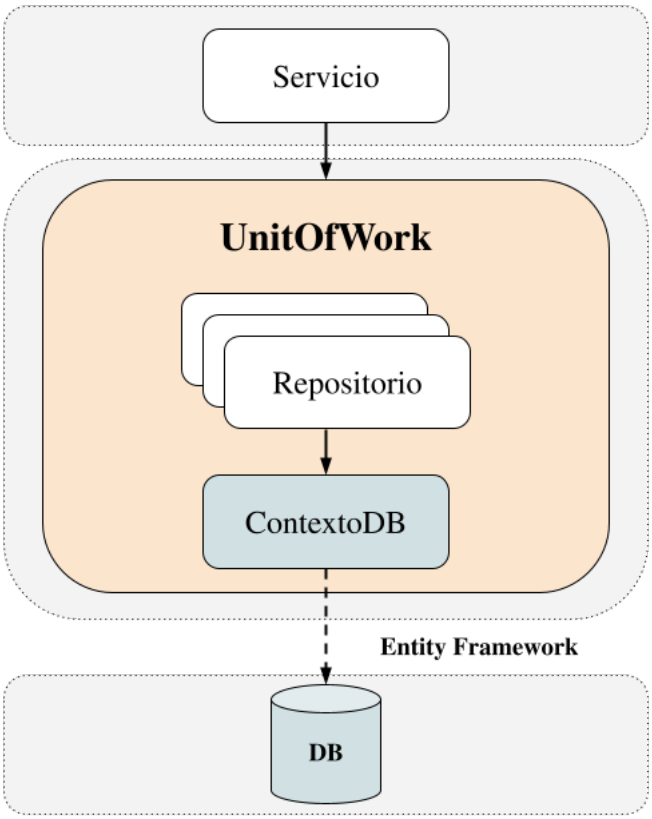


Figura 5.3: Diagrama de la implementación de la unidad de trabajo en el proyecto.

5.2.2. Diagrama interactivo

Como parte de los requisitos del sistema, se debía implementar un diagrama interactivo para visualizar las piezas del vehículo en las que se debe trabajar. Cuando un usuario selecciona una pieza, ésta pasa de desactivada a activada o viceversa, cambiando su color a rojo o negro respectivamente.

Luego de discutirlo con el supervisor, se decidió utilizar *Asynchronous JavaScript and XML* (AJAX). Esta técnica permite intercambiar datos con el servidor de forma asíncrona sin necesidad de actualizar la página [14], proporcionando una forma de modificar el diagrama y la información de la orden de reparación a partir de las acciones del usuario.

Para resolver el problema de incluir diagramas interactivos con actualización asíncrona en la aplicación, se siguieron una serie de pasos. En primer lugar, se creó el diagrama del vehículo como gráfico vectorial escalable (SVG) para poder manipularlo fácilmente por código [15]. Este tipo de formato permite crear figuras y agruparlas en contenedores. Los atributos de un contenedor son heredados por los hijos que contiene, lo que permite cambiar el color de varias figuras al mismo tiempo modificando el contenedor que las agrupa [16]. Debido a esto, un contenedor es equivalente a una pieza en el diagrama. La Figura 5.4 muestra el contenedor para la rueda del vehículo cuando esta desactivado y activado.

```
1  <g stroke="#000" fill="none" stroke-width=".8">
2    <ellipse cx="12.895" cy="12.895" rx="12.495" ry="12.495" />
3    <ellipse cx="12.895" cy="12.895" rx="7.6865" ry="7.6869" />
4  </g>
```

(a)

```
1  <g stroke="#ff0000" fill="none" stroke-width=".8">
2    <ellipse cx="12.895" cy="12.895" rx="12.495" ry="12.495" />
3    <ellipse cx="12.895" cy="12.895" rx="7.6865" ry="7.6869" />
4  </g>
```

(b)

Figura 5.4: Contenedor para una rueda del diagrama (a) desactivado y (b) activado.

En segundo lugar, se implementó un componente responsable de cargar el estado del diagrama cuando el usuario accede a la página, así como de detectar sus interacciones y comunicarlas al servidor mediante AJAX. Para llevar a cabo todo esto, el componente recibe el diagrama en formato SVG y las direcciones URL del servidor para obtener y guardar el estado del dibujo. Donde el estado es una lista de unos y ceros que indican si una pieza esta activada (uno) o desactivada (cero), es decir, si un contenedor es color rojo o negro. Cuando un usuario selecciona una pieza el componente actualiza el color del contenedor y le envía al servidor el nuevo estado del diagrama. El Algoritmo 1 muestra el pseudocódigo de la implementación del componente.

Cabe destacar que las figuras del gráfico son independientes de la implementación del componente, permitiendo así, modificar el dibujo sin necesidad de refactorizar la gestión del diagrama.

Algorithm 1 Algoritmo para la gestión del diagrama interactivo.

```
1: procedure DiagramManager(diagram, urlGetState, urlSaveState)
2:     parts = getPartsSVG(diagram)
3:     state = AjaxCallToServer(urlGetState)                    ▷ Obtiene estado
4:     for i = 0; i < length(state); i++ do                     ▷ Inicializa diagrama
5:         if state[i] == 0 then
6:             Asignar color negro a la pieza parts[i]
7:         else
8:             Asignar color rojo a la pieza parts[i]
9:         end if
10:    end for
11:    while vista abierta do                                   ▷ Espera acciones del usuario
12:        for i = 0; i < length(state); i++ do
13:            if se selecciona la pieza parts[i] then          ▷ Se detectan cambios
14:                if state[i] == 0 then                        ▷ La pieza pasa a estar activada
15:                    Asignar color rojo a la pieza parts[i]
16:                    state[i] == 1
17:                else                                         ▷ La pieza pasa a estar desactivada
18:                    Asignar color negro a la pieza parts[i]
19:                    state[i] == 0
20:                end if
21:                AjaxCallToServer(urlSaveState, state)        ▷ Guarda estado
22:            end if
23:        end for
24:    end while
25: end procedure
```

5.3. Verificación y validación

Si bien la implementación de pruebas automatizadas no formaba parte de este proyecto, se llevó a cabo un proceso de verificación y validación mediante pruebas manuales para completar todas las fases de la metodología de desarrollo. En concreto, se utilizó el enfoque de pruebas exploratorias, que consistía en examinar la aplicación utilizando el conocimiento, la experiencia y la intuición en vez de casos de prueba predefinidos [17].

El objetivo de este proceso era validar que el producto final cumpliera todos los requisitos del proyecto. Con este fin, se realizaron pruebas para detectar problemas funcionales y/o de calidad, así como verificar los avances logrados y proponer mejoras.

Con respecto al proyecto, las pruebas se realizaban al finalizar la implementación de cada módulo. Si el desarrollo duraba varias semanas, como en el caso del módulo de vehículos y el de órdenes de reparación, se efectuaban semanalmente. El cuadro 5.1 muestra un resumen de las pruebas realizadas y los errores encontrados.

Id.	Módulos probados	Funcionalidades probadas	Observaciones
PE01	Usuarios	Listar, crear, editar y eliminar usuarios, permisos, vistas y asignaciones.	Funcionamiento y diseño correcto.
PE02	Clientes	Listar, crear, editar y eliminar clientes.	Funcionamiento y diseño correcto.
PE03	Vehículos	Listar, crear, editar y eliminar vehículos.	Funcionamiento y diseño correcto. Añadir funcionalidad para obtener los vehículos de cada cliente.
PE04	Trabajadores y Vehículos	Listar, crear, editar y eliminar trabajadores. Listar vehículos de cada cliente.	Funcionamiento y diseño correcto.
PE05	Trabajadores	Calendario semanal de trabajadores.	Funcionamiento correcto. Reorganizar interfaz para mostrar dos semanas por fila en vez de una.
PE06	Órdenes de reparación	Listar, crear, editar y eliminar órdenes. Diagrama interactivo del vehículo.	Funcionamiento y diseño correcto.
PE07	Órdenes de reparación	Listar, crear, editar y eliminar reparaciones y materiales.	Funcionamiento correcto. Reorganizar interfaz para mostrar las reparaciones y materiales en pestañas en vez de secciones.
PE08	Órdenes de reparación	Listar, crear, editar y eliminar órdenes, reparaciones y materiales. Diagrama interactivo del vehículo.	Funcionamiento y diseño correcto.

Cuadro 5.1: Pruebas exploratorias realizadas y retroalimentación obtenida.

Cada prueba exploratoria consistía en el supervisor desempeñando el rol de usuario final probando todas las funcionalidades implementadas y proporcionando retroalimentación. Cabe destacar que el supervisor fue quien diseñó e implementó la aplicación ERP original, lo que hizo que este proceso fuera muy provechoso desde el punto de vista de la validación, ya que conocía no solo la arquitectura actual de la aplicación, sino que también tenía experiencia en las partes críticas del sistema.

Posteriormente, se utilizaban las observaciones y comentarios recibidos para mejorar la experiencia de usuario y asegurar la calidad del producto. Si se detectaba algún fallo en la implementación, se tomaba nota para corregirlo y revisarlo en la siguiente reunión.

Capítulo 6

Conclusiones

En lo que respecta al presente proyecto, se han alcanzado todos los objetivos especificados al inicio del mismo a pesar de las desviaciones sufridas en las fases de implementación y cierre. Estos problemas estaban relacionados principalmente con la estimación del tiempo, ya que varias tareas llevaron más o menos del tiempo previsto. A pesar de esto, los módulos de gestión de usuarios, trabajadores, clientes, vehículos y órdenes de reparación se han implementado en su totalidad utilizando las tecnologías definidas, siguiendo los estándares de la empresa y dentro del tiempo definido para el proyecto.

Con respecto al ámbito profesional, Infoware Soluciones Informáticas S.L. ha proporcionado un entorno de trabajo de alta calidad. Mis compañeros han sido de gran ayuda a la hora de aprender las nuevas técnicas y herramientas requeridas para el proyecto, siempre estuvieron disponibles para resolver las dudas que surgían y solucionar cualquier problema encontrado.

Por otra parte, a nivel formativo, este proyecto me ha servido para aplicar, a mayor escala, todos los conocimientos adquiridos a lo largo de la carrera. En particular, me permitió mejorar mis capacidades en gestión de proyectos, junto con las de análisis, diseño y desarrollo de software.

Personalmente, estoy muy contenta con el proyecto y el resultado obtenido. Este trabajo me ha posibilitado experimentar cómo es un entorno laboral real y, al mismo tiempo, trabajar en un proyecto muy interesante. De este modo, me ha permitido mejorar mis habilidades de comunicación y adaptabilidad, al mismo tiempo que me ha ayudado a aprender a desenvolverme profesionalmente.

En conclusión, el proyecto ha finalizado exitosamente y su desarrollo supuso una experiencia inestimable tanto para mis estudios como a nivel personal y profesional. Me ha permitido desenvolverme en nuevos entornos, reforzar mis conocimientos y desarrollar nuevas habilidades en un ambiente de trabajo excepcional.

Bibliografía

[1] Instituto Nacional de Estadística (INE). Encuesta sobre el uso de TIC y del comercio electrónico en las empresas Año 2020 – Primer trimestre de 2021 https://www.ine.es/prensa/tic_e_2020_2021.pdf [Consulta: 25 de Mayo de 2023]

[2] Microsoft Dynamics 365. What is ERP?. https://dynamics.microsoft.com/en-us/erp/what-is-erp/ [Consulta: 25 de Mayo de 2023]

[3] Infoware Soluciones Informáticas S.L. https://www.infoware.es/ [Consulta: 25 de Mayo de 2023]

[4] Infoware Soluciones Informáticas S.L. Desarrollo programas a medida https://www.paginas-web-castellon.com/programas-a-medida/ [Consulta: 25 de Mayo de 2023]

[5] Microsoft. What is .NET? Introduction and overview. https://learn.microsoft.com/en-us/dotnet/core/introduction [Consulta: 25 de Mayo de 2023]

[6] Microsoft. Overview of ASP.NET Core MVC. https://learn.microsoft.com/en-us/aspnet/core/mvc/overview [Consulta: 25 de Mayo de 2023]

[7] Microsoft. Entity Framework overview. https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/ef/overview [Consulta: 25 de Mayo de 2023]

[8] Adobe Communications Team. Waterfall Methodology: A Complete Guide. https://business.adobe.com/blog/basics/waterfall [Consulta: 25 de Mayo de 2023]

[9] Talent. Salario medio para Programador en España, 2023. https://es.talent.com/salary?job=Programador [Consulta: 25 de Mayo de 2023]

[10] Microsoft. Working with DbContext. https://learn.microsoft.com/en-us/ef/ef6/fundamentals/working-with-dbcontext [Consulta: 04 de Junio de 2023]

[11] Microsoft. Entity Framework overview — Access and change entity data. https://learn.microsoft.com/en-us/dotnet/framework/data/adonet/ef/overview#access-and-change-entity-data [Consulta: 04 de Junio de 2023]

[12] Microsoft. Views in ASP.NET Core MVC — Pass data to views. https://learn.microsoft.com/en-us/aspnet/core/mvc/views/overview?view=aspnetcore-6.0#pass-data-to-views [Consulta: 04 de Junio de 2023]

[13] Microsoft. Implementing the Repository and Unit of Work Patterns in an ASP.NET MVC Application (9 of 10) https://learn.microsoft.com/en-us/aspnet/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/

implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application [Consulta: 04 de Junio de 2023]

[14] MDN Web Docs - Mozilla. AJAX: Getting started — What's AJAX?. https://developer.mozilla.org/en-US/docs/Web/Guide/AJAX/Getting_Started#whats_ajax [Consulta: 04 de Junio de 2023]

[15] MDN Web Docs - Mozilla. SVG: Scalable Vector Graphics — Getting Started with SVG. https://developer.mozilla.org/en-US/docs/Web/SVG#getting_started_with_svg [Consulta: 04 de Junio de 2023]

[16] MDN Web Docs - Mozilla. SVG: Scalable Vector Graphics — SVG element reference — $< g >$. https://developer.mozilla.org/en-US/docs/Web/SVG/Element/g [Consulta: 04 de Junio de 2023]

[17] Service Manual - GOV.UK. Technology - Exploratory testing. https://www.gov.uk/service-manual/technology/exploratory-testing [Consulta: 04 de Junio de 2023]