Análisis, diseño y desarrollo de una aplicación web para la gestión de un taller mecánico de motocicletas

Memoria de Proyecto Final de Grado/Máster

Máster Universitario en Aplicaciones multimedia

TFM/Itinerario Profesional

Autor: David Alberto Casado González

Consultor: Mikel Zorrilla Berasategui

Profesor: Laura Porta Simó

07 de junio de 2019

Créditos/Copyright

A) CreativeCommons:

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Análisis, diseño y desarrollo de una aplicación web para la gestión de un taller mecánico de motocicletas*
Nombre del autor:	*David Alberto Casado González*
Nombre del consultor/a:	*Mikel Zorrilla Berasategui*
Nombre del PRA:	Laura Porta Simó
Fecha de entrega:	06/2019
Titulación:	*Máster Universitario en Aplicaciones multimedia*
Área del Trabajo Final:	*Área TFM Profesionalizadora*
Idioma del trabajo:	*Castellano*
Palabras clave	*Taller mecánico, sistema de gestión, OCR*

Resumen del Trabajo

Este trabajo se ha realizado con la finalidad de desarrollar una aplicación que satisfaga las necesidades que implica la gestión de un pequeño taller mecánico, específicamente de motocicletas ya que es en ese pequeño sector donde se ha detectado la necesidad. La metodología utilizada es iterativa, pero, en vez de entregar partes totalmente funcionales que cumplan las necesidades end to end, ha sido por capas, de forma que el aprendizaje necesario para desarrollar las capas futuras se ha ido intercalando en el desarrollo de las anteriores.

Tras las diferentes iteraciones se ha desarrollado un backend en JAVA, que gestiona todas las peticiones recibidas mediante interacciones con la base de datos y un frontend en Angular 7 accesible tanto desde equipos de sobremesa como tablets y móviles gracias a los estilos SCSS desarrollados.

El producto final cumple con los objetivos marcados, de forma que el dueño de un taller mecánico de un tamaño reducido pueda realizar las gestiones de admisión de vehículos, generar órdenes de reparación y facturas, gestionar la flota de vehículos, clientes y stock de productos e incluso podría gestionar la posibilidad de escalar esta administración a varios talleres de su propiedad.

Abstract

This Project has been done with the main goal of developing an application that satisfies the needs that implies managing a small garage, specifically, a motorcycle one, because it's in this specific sector where I detected the need. The development methodology that has been used is iterative, but in an unusual way, because each iteration wasn't delimited by functional needs. Instead, it was delimited by technological layers. The reason for that is the need of learning the technology that was essential to complete the next layer.

After each iteration, a JAVA backend (in charge of managing all the requests interacting with the data base) and an Angular 7 frontend (that can be accessed from pcs and mobile devices thanks to the SCSS implemented) were developed.

The final application stands with the established objectives, so the owner of a small garage is able to manage the entrance of vehicles, generate work orders and invoices and manage the list

of vehicles, clients and product stock. Within this application, this management can even be done for several garages that he/she owns.

Dedicatoria

Este proyecto final de máster está dedicado a Laura, mi novia y en pocos días mi esposa, por la paciencia demostrada ante mi plena dedicación a este proyecto.

Agradecimientos

Quería agradecer a Pablo, fuente de conocimiento interminable, que me ha enseñado tantas y tantas cosas vitales para el desempeño de este TFM.

Abstract

This Project has been done with the main goal of developing an application that satisfies the needs that implies managing a small garage, specifically, a motorcycle one, because it's in this specific sector where I detected the need. The development methodology that has been used is iterative, but in an unusual way, because each iteration wasn't delimited by functional needs. Instead, it was delimited by technological layers. The reason for that is the need of learning the technology that was essential to complete the next layer.

After each iteration, a JAVA backend (in charge of managing all the requests interacting with the data base) and an Angular 7 frontend (that can be accessed from pcs and mobile devices thanks to the SCSS implemented) were developed.

The final application stands with the established objectives, so the owner of a small garage is able to manage the entrance of vehicles, generate work orders and invoices and manage the list of vehicles, clients and product stock. Within this application, this management can even be done for several garages that he/she owns.

Palabras clave

Garage, management system, OCR.

Resumen

Este trabajo se ha realizado con la finalidad de desarrollar una aplicación que satisfaga las necesidades que implica la gestión de un pequeño taller mecánico, específicamente de motocicletas ya que es en ese pequeño sector donde se ha detectado la necesidad. La metodología utilizada es iterativa, pero, en vez de entregar partes totalmente funcionales que cumplan las necesidades end to end, ha sido por capas, de forma que el aprendizaje necesario para desarrollar las capas futuras se ha ido intercalando en el desarrollo de las anteriores.

Tras las diferentes iteraciones se ha desarrollado un backend en JAVA, que gestiona todas las peticiones recibidas mediante interacciones con la base de datos y un frontend en Angular 7 accesible tanto desde equipos de sobremesa como tablets y móviles gracias a los estilos SCSS desarrollados.

El producto final cumple con los objetivos marcados, de forma que el dueño de un taller mecánico de un tamaño reducido pueda realizar las gestiones de admisión de vehículos, generar órdenes de reparación y facturas, gestionar la flota de vehículos, clientes y stock de productos e incluso podría gestionar la posibilidad de escalar esta administración a varios talleres de su propiedad.

Palabras clave

Taller mecánico, sistema de gestión, OCR.

Notaciones y Convenciones

- Código XML:
    - Tipo: Courier New.
    - Tamaño: 9.
    - Ejemplo: `<tag property='value'>`

Índice

Figuras y tablas

Índice de figuras

Índice de tablas

Capítulo 1: Introducción

1 Introducción

En la era digital en la que vivimos hay multitud negocios en los que los sistemas informáticos han penetrado o lo están haciendo muy lentamente. Esto se puede apreciar en la mayoría de los negocios de barrio, en los que, por falta de medios o de formación (o ambas), su informatización se limita al uso de procesadores de texto y hojas de cálculo para realizar sus tareas de gestión. Esta situación es un nicho de mercado muy interesante tanto para el desarrollo de aplicaciones a medida como aplicaciones genéricas por sector, donde, en base a las necesidades de diversos negocios similares se puede realizar una aplicación que pueda ser usada por todos ellos.

Basándonos en este escenario, se va a desarrollar una aplicación básica para llevar la gestión de un pequeño taller de barrio tomando como referencia uno real, fruto del cual surgió la inspiración para realizar este proyecto.

2 Descripción/Definición

El proyecto relatado en este documento versa sobre la realización de un sistema de gestión de talleres de motocicletas de tamaño reducido. Con este trabajo se pretende cubrir la necesidad de talleres de barrio donde actualmente utilizan software básico como tablas de Excel tanto para la gestión de los usuarios, stock y flota de vehículos, como para la emisión de facturas. Esto hace que cualquier consulta en el histórico, generación de factura o recepción de un vehículo se traduzca en un proceso tedioso, donde, por lo general se piden datos innecesarios al cliente con el fin de localizar/generar la información.

Aunque los puntos expuestos para justificar el desarrollo no parezcan razones de peso, realmente la inclusión de una aplicación informática que garantice el correcto almacenamiento de los datos y la rápida gestión de estos puede ser crucial para la supervivencia de un negocio pequeño como el que hemos definido como objetivo. Un taller que sea capaz de evaluar la reincidencia de averías en un vehículo consultando su historial rápidamente o que sea capaz de gestionar las interacciones con el usuario de forma ágil y sin hacerle perder tiempo, pueden marcar el éxito o fracaso del negocio.

El segmento de aplicaciones que existen actualmente sobre este sector es muy escaso, siendo las pocas herramientas que hay disponibles de pago. Estas, a pesar de su condición de no gratuidad, son prácticamente inmanejables debido a la multitud de opciones y características que barajan.

Bajo estas premisas, se va a desarrollar una aplicación que cubra las necesidades elementales del dueño del taller, incluyendo sólo lo imprescindible para realizar las operaciones más habituales que conlleva la propia gestión del taller. Estas operaciones recogen el tratamiento de:

- Clientes: pudiendo acceder a un listado de estos, teniendo disponible la visualización de un cliente específico, la modificación de sus datos o acceso al listado de vehículos que le pertenecen.
- Vehículos: pudiendo acceder a un listado de estos, teniendo disponible la visualización de un vehículo específico, la modificación de sus datos o acceso al listado de flujos de reparación que le afectan o le han afectado.
- Productos: teniendo acceso a un listado de estos, y a sendos formularios de visualización/edición por producto para tener actualizado el stock del taller.
- Talleres: aunque a prori los dueños de un taller de barrio sólo cuentan con un único establecimiento, se ha decidido no cerrar las puertas a una posible expansión de sus negocios utilizando la herramienta, de forma que se pueda gestionar diversos talleres desde la propia aplicación. Por lo tanto, la herramienta permitirá obtener un listado de los talleres y la visualización/actualización de los datos de cada uno de ellos. Los flujos de reparación que se gestionan en la herramienta estarán vinculados a un único taller, por lo que se podrá en todo momento ver los paneles de resumen de los flujos seleccionando el taller escogido.
- Resguardos: generados a partir de la recepción de un vehículo en un taller, los resguardos podrán ser accesibles a través de un panel de resumen de resguardos del taller escogido, pudiendo generar un documento PDF para su impresión o bien generar una vista previa online.

- Órdenes de reparación: generadas a partir de un resguardo, las órdenes de reparación podrán ser accesibles a través de un panel de resumen de órdenes de reparación del taller escogido, pudiendo generar un documento PDF para su impresión o bien generar una vista previa online.
- Facturas: generadas a partir de una orden de reparación, las facturas podrán ser accesibles a través de un panel de resumen de facturas del taller escogido, pudiendo generar un documento PDF para su impresión o bien generar una vista previa online.

3 Objetivos generales

Listado y descripción de los objetivos del TF, ordenados por relevancia.

3.1 Objetivos principales

Objetivos de la aplicación:

- Implementar una aplicación que permita llevar la gestión de las operaciones administrativas básicas de un taller de motocicletas.
- Adaptar la interfaz a los diferentes tipos de pantalla de los dispositivos de los que se puede utilizar.
- Seguir un diseño sencillo sin sobrecarga de información.
- Procurar una comunicación rápida entre el frontal y la base de datos a la hora de visualizar, modificar o insertar información.
- Incorporar tecnología OCR para la lectura de matrículas con el fin de aportar una alternativa al usuario a la hora de introducir la matricula con el fin de obtener datos los datos necesarios del vehículo al iniciar un flujo de reparación (generar un resguardo).
- Seguir un criterio estético uniforme, utilizando los colores corporativos.

Objetivos para el cliente:

- Ser capaz de gestionar de forma sencilla los inventarios de productos, clientes, vehículos y talleres, de forma que pueda visualizar los listados y ver, modificar y crear cualquiera de las entidades especificadas.
- Acceder fácilmente a los paneles de resumen de los flujos de reparación que están en curso de forma que de un simple vistazo pueda saber qué trabajo está pendiente y la urgencia que tiene teniendo en cuenta la fecha prevista de fin.

Objetivos personales del autor del TF:

- Dominar la exposición de servicios a través de un backend realizado en Java.
- Adquirir conocimientos de Angular 7 para realizar frontales sencillos que sirvan de interfaz para los servicios expuestos en el backend.

3.2 Objetivos secundarios

Objetivos adicionales que enriquecen el TF.

- Envío de correos a los posibles usuarios del taller con el fin de comunicar avances en el flujo de reparación de sus vehículos o bien promociones u otras comunicaciones personalizadas.

4 Metodología y proceso de trabajo

Teniendo en cuenta el estado del arte (que se verá en el apartado 1.Estado del arte) dentro del sector de las aplicaciones dedicadas a la gestión de talleres, la opción de adaptar o mejorar un producto existente nunca se contempló, especialmente porque no se encontraron proyectos de código libre en los que poder trabajar. Por lo tanto, la opción escogida para satisfacer las necesidades del proyecto ha sido la de crear un producto nuevo que cumpla con los requisitos especificados en el diseño de la solución.

Dentro de las opciones de las que se dispone a la hora de realizar un nuevo producto tenemos:

- Usar un CMS (Wordpress, Drupal…):
  - Como frontal y backend: utilizar la solución global ofrecida por el CMS, de forma que se utilice tanto la interfaz del propio CMS como el almacenamiento de las entidades.
  - Como backend: utilizar únicamente el CMS como bakend de la solución, de forma que se construya únicamente un frontal que consuma la información almacenada en el CMS y expuesta a través de servicios REST.
- Desarrollar una aplicación nueva:
  - Usando un framework de aplicaciones web como Laravel o Ruby on Rails.
  - Construyendo una aplicación web entera, utilizando una tecnología actual para desarrollar el front y el backend.

De entre estas cuatro opciones, se descartó rápidamente la primera de todas, ya que para un proyecto de esta envergadura interesa tener un frontal propio y personalizado con eventos que modifiquen la estructura del HTML.

La segunda opción también fue descartada porque, aunque sí que podría resultar útil para los inventarios de la aplicación, hay servicios REST que no podría aportarnos el CMS (como la lectura de matrículas a través de un servicio externo, o la generación de documentos a través de plantillas).

De entre las dos opciones restantes, se decantó finalmente por la segunda, aunque la primera de las dos hubiese sido interesante (aunque habría que estudiar su viabilidad, especialmente por la integración de la librería OCR). La decisión se tomó en base al especial interés en aprender tecnologías como Angular ya que su expansión hoy en día es notable y el salto hacia otras tecnologías de frontend punteras como React o Vue no sería excesivamente costoso.

En cuanto a la metodología de desarrollo seguida para realizar este proyecto, se ha seguido una estructura ágil, pero no tal y como esta se define, ya que al final de cada iteración no se obtiene un producto estable y funcional. Cada iteración representa el fin de una de las capas que tiene la aplicación: backend, frontend funcional y frontend estético. Estas iteraciones se han delimitado así debido a la necesidad de formación para poder completar el producto final, ya que la primera fase era a priori la que más se dominaba (la última también, pero es imprescindible contar con la anterior para su comienzo), pero en la siguiente se sufrían grandes carencias. No obstante, a medida que el producto se realizaba en las diferentes capas que lo compone, se han realizado ajustes en las anteriores por errores, detalles no contemplados en la especificación o mejoras.

5 Planificación

La planificación inicial del proyecto contempla la ejecución del mismo desde el 25 de febrero hasta el 7 de junio de 2019. Podemos ver el desglose de tareas con los hitos principales marcados en negrita a continuación:

Nombre	Duración	Inicio	Final
PEC 1	**12 días**	25/02/2019	08/03/2019
Estudio y debate de la propuesta	3 días	25/02/2019	27/02/2019
Aprobación por parte del consultor	1 día	26/02/2019	26/02/2019
Informe de trabajo	10 días	27/02/2019	08/03/2019
PEC 2	**10 días**	09/03/2019	18/03/2019
Informe de trabajo	10 días	09/03/2019	18/03/2019
Estudio de las tecnologías	8 días	09/03/2019	16/03/2019
Estudio del estado del arte	3 días	12/03/2019	14/03/2019
Definición de objetivos	2 días	17/03/2019	18/03/2019
PEC 3	**28 días**	**19/03/2019**	**15/04/2019**
Informe de trabajo	28 días	19/03/2019	15/04/2019
Análisis de requisitos	5 días	19/03/2019	23/03/2019
Estudio detallado Backend+OCR	5 días	19/03/2019	23/03/2019
Configuración del entorno	2 días	23/03/2019	24/03/2019
Diseño del modelo de datos	1 día	24/03/2019	24/03/2019
Creación de la base de datos	1 día	25/03/2019	25/03/2019
Diseño del Backend	3 días	26/03/2019	28/03/2019
Diseño de pruebas del Backend	4 días	26/03/2019	29/03/2019
Desarrollo del Backend	14 días	30/03/2019	13/04/2019
Realización pruebas Backend	14 días	30/03/2019	13/04/2019
Estudio Frameworks Frontend	28 días	19/03/2019	15/04/2019
PEC 4	**28 días**	**16/04/2019**	**13/05/2019**
Informe de trabajo	28 días	16/04/2019	13/05/2019
Reajuste planificación	6 días	16/04/2019	21/04/2019
Diseño de interfaces gráficas	3 días	22/04/2019	24/04/2019
Configuración del entorno Front	2 días	25/04/2019	26/04/2019
Desarrollo del Frontend	17 días	27/04/2019	12/05/2019
Realización pruebas Frontend	17 días	27/04/2019	12/05/2019
PEC 5	**25 días**	**14/05/2019**	**07/06/2019**
Informe de trabajo	25 días	14/05/2019	07/06/2019
Reajuste planificación	6 días	14/05/2019	19/05/2019
Desarrollo de la interfaz	9 días	20/05/2019	28/05/2019

Pruebas multidispositivo	9 días	20/05/2019	28/05/2019
Presentación Académica	10 días	29/05/2019	07/06/2019
Presentación Pública	7 días	01/06/2019	07/06/2019

Tabla 1: Planificación

Esta planificación se ha visto variada entre los diferentes hitos marcados debido a diversos contratiempos en el desarrollo. Los detalles de la planificación de cada hito se pueden consultar en los informes de trabajo. A continuación, se puede observar el diagrama de Gantt que muestra la planificación a partir del hito PEC 3, es decir, a partir del inicio del desarrollo del producto.

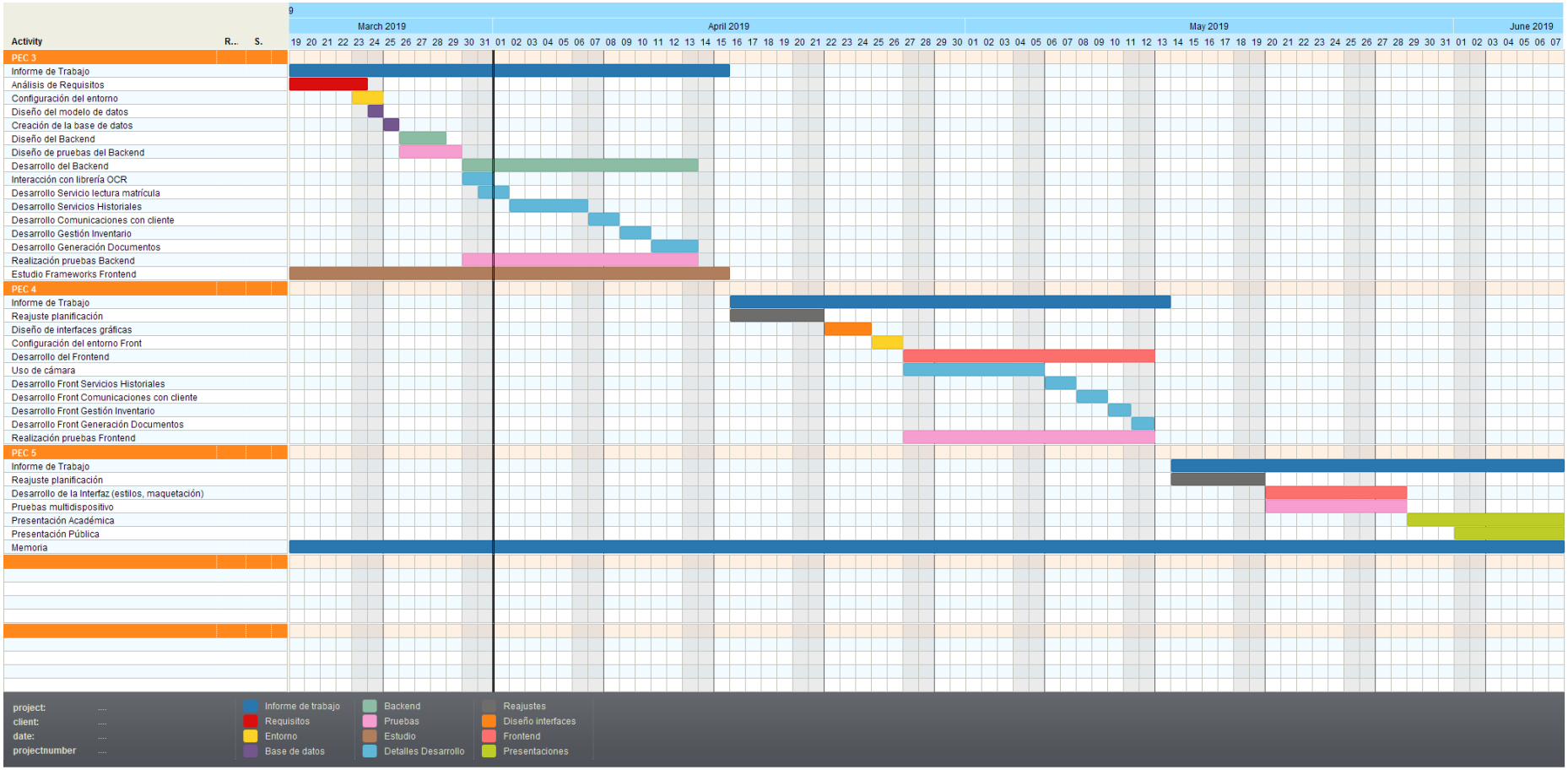


Ilustración 1: Planificación del proyecto desde el hito "PEC3"

6 Presupuesto

A continuación, se detalla la estimación de costes del proyecto teniendo en cuenta una aproximación de las horas dedicadas y registradas en los diferentes informes de trabajo detallados en cada uno de los hitos del proyecto. En todo momento se considera que el presupuesto aquí definido corresponde al esfuerzo realizado por un único desarrollador. Todo impuesto asociado al trabajo no está incluido en el precio.

CONCEPTO	HORAS DEDICADAS	PRECIO/HORA	TOTAL
ANÁLISIS			
Estudios previos	65	30€	1.950€
Estado del arte	9	30€	270€
Objetivos, requisitos y casos de uso	7	30€	210€
DISEÑO			
Backend	10	25€	250€
Frontend	8	25€	200€
DESARROLLO			
Documentación	15	15€	225€
Configuraciones	8	15€	120€
Backend+pruebas	90	15€	1.350€
Frontend+pruebas	108	15€	1.620€
Interfaz gráfica	78	15€	1.170€
EQUIPAMIENTO			
Amortización PC	618	0.009€ (1)	56€
Licencia Intellij Idea	---	---	150€
Gastos de oficina (2)	---	---	105€
TOTAL			7.676€

Tabla 2: Presupuesto

(1) Considerando que el PC, cuyo coste fue de 1500€, tiene una vida útil de 8 años utilizándolo 6 horas al día durante los 365 días del año.

(2) En gastos de oficina se contabiliza el coste del material gráfico y la proporción de las facturas de la luz e internet, teniendo en cuenta que el proyecto se alarga durante alrededor de 3’5 meses.

7 Estructura del resto del documento

En la presente memoria se van a detallar los siguientes capítulos:

- **Análisis**: en él se detallan las características del entorno tecnológico en el momento de realizar el proyecto y el condicionamiento que estas han supuesto en su realización. Además, se especifican tanto los objetivos específicos que debe cumplir la aplicación y las especificaciones y características que incorporará para alcanzar dichos objetivos.
- **Diseño**: es el capítulo en el que se describe el modelado la solución, pasando desde la arquitectura hasta los aspectos de la interfaz y los lenguajes de programación usados para la implementación.
- **Implementación**: en este capítulo se explican los requisitos previos que se deben satisfacer, así como la preparación de los mismos para para poder ejecutar el producto.
- **Demostración**: en él se detallan las instrucciones de uso de la aplicación. Este apartado se verá ampliamente detallado en la guía de usuario adjunta y especificada en al Anexo B.
- **Conclusiones**: es el capítulo destinado a relatar las experiencias y las lecciones aprendidas con la realización de este proyecto. Además, se enumeran las diferentes vías de mejora que se podrían aplicar en un futuro sobre el producto para su mejora y ampliación de funcionalidad.

Capítulo 2: Análisis

1 Estado del arte

Dentro de este apartado abordaremos los diferentes puntos clave a tener en cuenta en la realización del proyecto. En cada uno de ellos, plantearemos diferentes formas de abordarlos con el objetivo de aportar una base sólida para demostrar que la alternativa escogida en cada una de las categorías es la más acertada para el proyecto en cuestión. Además, se estudiará el universo actual dentro de la temática escogida, con el propósito de extraer sus puntos fuertes y débiles. Esto condicionará el diseño del producto, el cual perseguirá imitar los puntos fuertes de cada una de ellas, así como mejorar los débiles, de forma que se realice un producto de valor.

1.1 Aplicaciones actuales

En el mercado existen muy pocos productos especializados en el sector de los talleres mecánicos, siendo muchos de los productos encontrados genéricos, de forma que se pueda llevar prácticamente cualquier tipo de negocio. Estos productos han sido excluidos de este estudio, ya que no se consideran aptos para satisfacer las necesidades específicas del cliente objetivo. La comparativa y crítica de estas aplicaciones se hace en la sección: *2.1. Comparativa de la competencia*.

1.2 Optical Character Recognition

Como uno de los puntos fuertes de la aplicación, la elección de la librería que realizará la lectura de las matrículas es muy importante y puede llegar a condicionar el resto del desarrollo por posibles incompatibilidades. Para este estudio se incluyen las librerías más utilizadas hoy en día tanto en OCR como en librerías específicas de lectura de matrículas:

1.3 Tecnología Backend

Aunque a priori parece que no tiene por qué ser un punto determinante, parece claro que, dependiendo de qué tecnología se escoja, el desarrollo del producto puede tener un grado de dificultad diferente. Partiendo de que para realizar este proyecto se dispone de un servidor en el que se puede instalar cualquier tecnología, habrá que tener en cuenta las compatibilidades con la librería OCR a escoger y, en gran medida, con el nivel de conocimiento de las tecnologías por parte del equipo de desarrollo o bien la curva de aprendizaje teniendo en cuenta su perfil. En la sección *4.1.1. Backend* se describen las tecnologías estudiadas.

1.4 Tecnología Frontend

Como el producto va a ser multiplataforma, se debe escoger una tecnología con la que sea fácil crear un frontal para los diferentes tipos de dispositivos (PCs, móviles y tablets). Además, su integración con la tecnología backend escogida debe ser perfecta, permitiendo comunicaciones rápidas y sencillas. En la sección *4.1.2. Frontend* se describen las tecnologías estudiadas.

1.5 Aspectos legales

Los talleres tienen su relación contractual con los clientes regulada por una serie de leyes. El proyecto se ve afectado por el R.D. 1457/1986 de 10 de enero (Boletín Oficial del Estado, 2010) debido a que especifica la información a incluir en los diferentes documentos que se proporcionan al cliente (resguardos, partes de reparación y facturas). Concretamente, en este Real Decreto se especifica que:

- El resguardo debe incluir como mínimo:
  - El número de identificación fiscal y el domicilio del taller.
  - El nombre y domicilio del usuario.
  - La identificación del vehículo, con expresión de marca, modelo, matrícula y número de kilómetros recorridos, así como si el depósito del vehículo se efectúa para la confección del presupuesto o para la reparación del vehículo.
  - Descripción sucinta de la reparación y/o servicios a prestar, con sus importes, si fueran ya conocidos, en el caso de que el vehículo se entregue para reparación.
  - Fecha prevista de entrega, bien del presupuesto solicitado, bien del vehículo reparado.
  - Fecha y firma del prestador del servicio.
- La orden de reparación/factura debe incluir:
  - El número de identificación fiscal y el domicilio del taller.
  - El nombre y domicilio del usuario.
  - La identificación del vehículo, con expresión de marca, modelo, matrícula y número de kilómetros recorridos.
  - Reparaciones a efectuar, elementos a reparar o sustituir y/o cualquier otra actividad, con indicación del precio total desglosado a satisfacer por el usuario.
  - La fecha y la firma del prestador del servicio.
  - La fecha prevista de entrega del vehículo ya reparado, a partir de la aceptación del presupuesto (sólo aplicable a la orden de reparación).
  - Indicación del tiempo de validez del presupuesto (sólo aplicable a la orden de reparación).
  - Espacio reservado para la fecha y la firma de aceptación por el usuario.

2 Análisis del mercado

2.1 Comparativa de la competencia

A continuación, se incluye una breve descripción de los productos que se ofrecen en la red para solucionar el problema descrito en este proyecto. Una vez vista cada alternativa, se ofrece una tabla resumen para compararlas.

2.1.1 TallerAlpha

La aplicación que tiene una mayor presencia en la web es TallerAlpha (TallerAlpha, s.f.). Se trata de una aplicación de pago para dispositivos móviles y tablets disponible únicamente para Android (a pesar de anunciar que también hay versión para iOS, no es así).
TallerAlpha se compone de una serie de módulos que engloban las diferentes operaciones que pueden necesitar cualquier empleado del taller. Estos módulos son:

- Recepción del vehículo: para registrar la entrada de vehículos al taller.
- Reportes: para controlar las tareas ejecutadas asignadas a cada mecánico.
- Citas: para agendar visitas al taller.
- Cotizaciones: para gestionar la facturación de los servicios ofrecidos.
- Créditos: para gestionar las ventas a crédito de los servicios.

La aplicación en su inicio nos muestra una especie de cuadro de mando de resumen donde podemos ver el estado actual del sistema. Se enumeran las órdenes, las ventas, los créditos activos, las órdenes pendientes (el número de vehículos tratándose en el taller en ese momento), etc. Entrando en cada una de las secciones se desplegará un gráfico donde se aporta un detalle más afinado, pudiendo escoger la vista de lista para enumerar los elementos de la sección escogida.
En todo momento contamos con un menú de hamburguesa situada en la esquina izquierda donde se nos ofrecen las diferentes opciones agrupadas por categoría (una categoría no es necesariamente un módulo de la aplicación). Además, aquí podemos encontrar el acceso a la configuración de la aplicación, el perfil y una opción de incluir la firma digital del usuario que luego se utilizará en la emisión de facturas o partes de reparación.

2.1.2 Doscar

Según su web (Doscar, 2010-2019), Doscar es un sistema de gestión de talleres completo y sencillo de usar para equipos de sobremesa/TPVs. Sin embargo, la web no ofrece en su dominio un instalador, por lo que no se ha podido testear.

2.1.3 FuturoInformática

El programa ofrecido por FuturoInformática (GestFuturo) es de pago, por lo que el análisis se realizará a través de los vídeos demostrativos que incluyen en su web (FuturoInformática, 2017).

GestFuturo es una aplicación de sobremesa dedicada a la gestión de talleres mecánicos. La aplicación consta de una pantalla de inicio con un menú superior dividido en pestañas (podríamos considerar cada una de ellas como los módulos principales). Estas son:

- Ventas: para gestionar los presupuestos, órdenes de reparación (la recepción del vehículo) y facturaciones.
- Ficheros: principalmente usado para gestionar el inventario de clientes, proveedores y vehículos.
- Almacén: para llevar el stock del taller.
- Compras: para gestionar las compras a proveedores.
- Contabilidad: para controlar los aspectos económicos del taller.
- Tesorería: para gestionar las compras y ventas pendientes de pagar.
- Empresas: para llevar la gestión de los recursos humanos de la propia empresa.

2.1.4 Crítica productos analizados

2.1.4.1 TallerAlpha

Lo más reseñable a destacar negativamente es el nivel de detalle es altísimo (y no sólo en el proceso de recepción donde puedes indicar con todo lujo de detalle el estado de cada una de las partes del vehículo, incluso el estado de la pintura) y por ello se hace difícil su manejo, especialmente porque es un volumen de información muy grande para un tamaño de pantalla tan pequeño como es la de un dispositivo móvil. Para talleres mecánicos pequeños, manejar una aplicación como esta es totalmente insostenible.

No obstante, me ha parecido muy buena idea el introducir el concepto de la firma digital, por lo que podría ser una buena ampliación para el proyecto.

2.1.4.2 FuturoInformática

De nuevo se trata de una aplicación muy extensa, en la que se cubre una cantidad de funcionalidades abrumadora para un taller mecánico pequeño. Gran parte de la información que aporta estaría fuera del alcance de este proyecto. Además, el aspecto es muy recio, hecho que denota que, aparte de tener unos años, no se trata de una aplicación web, si no de un cliente pesado.

Un aspecto a destacar es el inventario de artículos y su relación con la generación de partes de reparación/facturas. Este aspecto no se tenía en consideración inicialmente, pero es un punto que facilitaría en gran medida el trabajo del mecánico por lo que va a ser incluido dentro de los objetivos del proyecto.

3 Definición de objetivos/especificaciones del producto

3.1 Objetivos

- Implementar una aplicación que permita llevar la gestión de las operaciones administrativas básicas de un taller de motocicletas. Estas son:
    - La recepción de los vehículos a partir de un formulario cuya generación se puede realizar a partir de la lectura de la matrícula utilizando OCR en dispositivos móviles o tablets y a través de un input en equipos de sobremesa.
    - Creación de un historial de resguardos, partes de reparación y facturas de los clientes accesibles partiendo del identificador único del vehículo (su matrícula).
    - Establecimiento de un medio de contacto directo con el cliente para las comunicaciones básicas (alertas programables de necesidad de revisión, presupuestos, fin de reparación o promoción) por correo electrónico.
    - Control del inventario necesario para la reparación de motocicletas y relacionado con la generación de partes de reparación y facturas.
- Crear una interfaz clara, intuitiva y accesible a la par que atractiva.

3.2 Especificaciones/características del producto

- **Multiplataforma**: la aplicación muestra diferentes configuraciones dependiendo del tamaño de la pantalla del dispositivo. En la gran mayoría de ocasiones el diseño es responsive, es decir, el contenido es el mismo, pero se despliega de forma diferente para encajar en el tamaño que se le permite. En ciertos casos se ha tendido a una estrategia adaptative, de forma que los componentes están duplicados y se muestra uno u otro dependiendo del tamaño de la pantalla.
- **Registro de clientes, vehículos, productos y talleres**: la aplicación cuenta con una serie de formularios que permiten la gestión de estas entidades.
- **Tramitación del flujo de reparación de una motocicleta**: la aplicación dispone tres paneles de resumen que indican el estado de la reparación de las motocicletas. Desde estos paneles el usuario podrá transitar los flujos, y por lo tanto generar un nuevo documento, cancelarlos, ver el documento correspondiente al estado actual del flujo o generar su pdf.
- **Generación de documentos en PDF**: cualquiera de los tres documentos que conforman el flujo de reparación de un vehículo (resguardo, orden de reparación y factura) es exportable a un fichero PDF para permitir su impresión de forma sencilla.
- **Lectura de matrículas por OCR**: para facilitar la recepción del vehículo se permite al usuario utilizar la tecnología OCR para iniciar el trámite en dispositivos móviles con cámara. No obstante, contarán con un input donde poder escribir la matrícula, o corregir la que ha cargado el sistema OCR.

Capítulo 3: Diseño

1 Arquitectura general de la aplicación/sistema/servicio

El proyecto consta de dos partes, un backend desarrollado en JAVA que expone servicios a través de unos endpoints determinados y un frontend, desarrollado en Angular que ataca dichos servicios para mostrar la información por pantalla. Este sería el esquema general de la solución, siendo la parte azul el backend y la naranja el frontend:

Ilustración 2: Arquitectura general de la solución

2 Arquitectura de la información y diagramas de navegación

Como la aplicación cuenta con dos partes totalmente diferenciadas (backend y frontend), se comentará la estructura de cada una de ellas en una sección independiente. Por último, se detalla el diagrama de navegación de la aplicación, donde se expondrán las diferentes pantallas que la componen y la interacción existente entre ellas.

2.1 Backend

Para desarrollar el backend de la aplicación se han seguido dos patrones de diseño:

- **DAO**: patrón cuyo objetivo es aislar la capa de persistencia de la base de datos de forma que la aplicación se abstraiga de todas las operaciones CRUD (Crate, Read, Update y Delete).
- **Repository**: patrón que permite gestionar el intercambio de información entre la lógica de negocio y la capa de acceso a datos (DAO). Con este patrón se abstrae la lógica de la aplicación de las operaciones de base de datos, permitiendo desarrollar una aplicación de negocio "genérica" cuya conexión a base de datos queda perfectamente aislada y por tanto fácilmente reemplazable si el proyecto lo requiriese.

Además, la estructura del proyecto viene definida por una interpretación de la arquitectura Clean. Esta arquitectura divide la arquitectura de la aplicación en cuatro (y proporciona los mecanismos de interconexión): entidades de negocio, casos de uso, adaptadores y agentes externos (base de datos, interfaz gráfica, dispositivos…). Para explicar el uso de esta arquitectura en la aplicación, se utilizará la siguiente imagen como soporte (Deutsch, 2018):

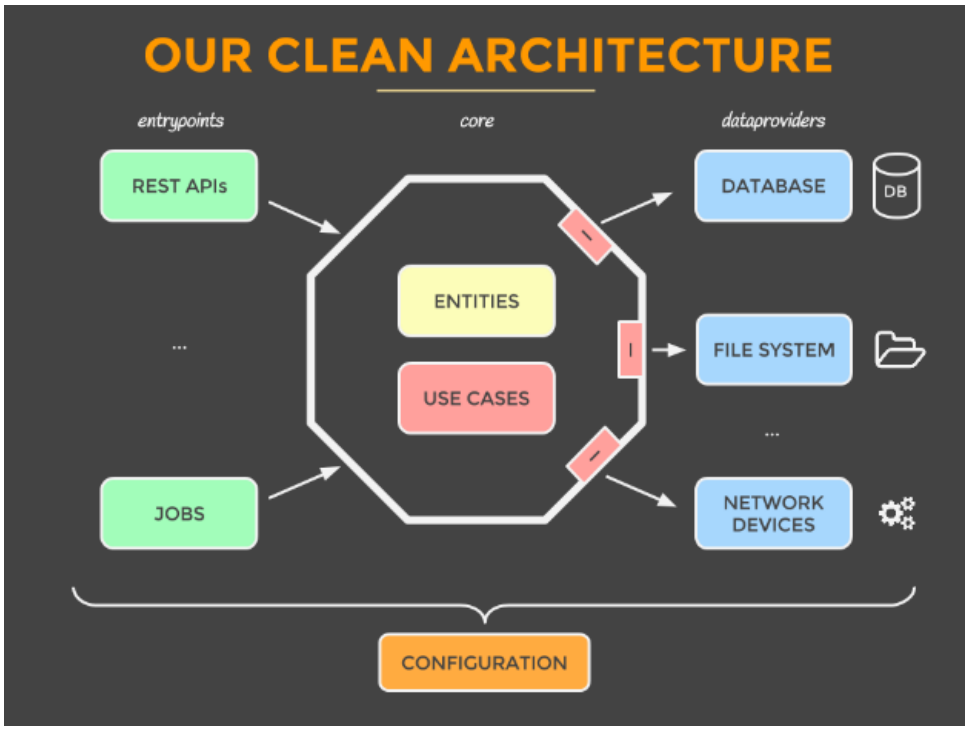


Ilustración 3: Arquitectura Clean

Para acceder a los servicios que satisface el backend se hacen llamadas a través de una REST Api. Estas llamadas serán recogidas a través del Rest Resource, el cual encaminará la petición al caso de uso concreto (Use Case) que usará entidades de la lógica de la aplicación en sus llamadas a las interfaces (los símbolos "-") para su salida al exterior (base de datos, generación de ficheros, acceso a librerías externas…).

Los Jobs remarcados en la imagen serían aquellos procesos batch que hacen reajustes en la aplicación. En el caso de este proyecto, se necesitaría un job en el crontab del servidor para eliminar los ficheros creados en el día por la solicitud de generación de PDFs.

2.1.1 Diagrama de la base de datos

A continuación, se expone un diagrama de la base de datos relacional empleada para realizar el proyecto, especificando las entidades creadas, así como las relaciones entre ellas.

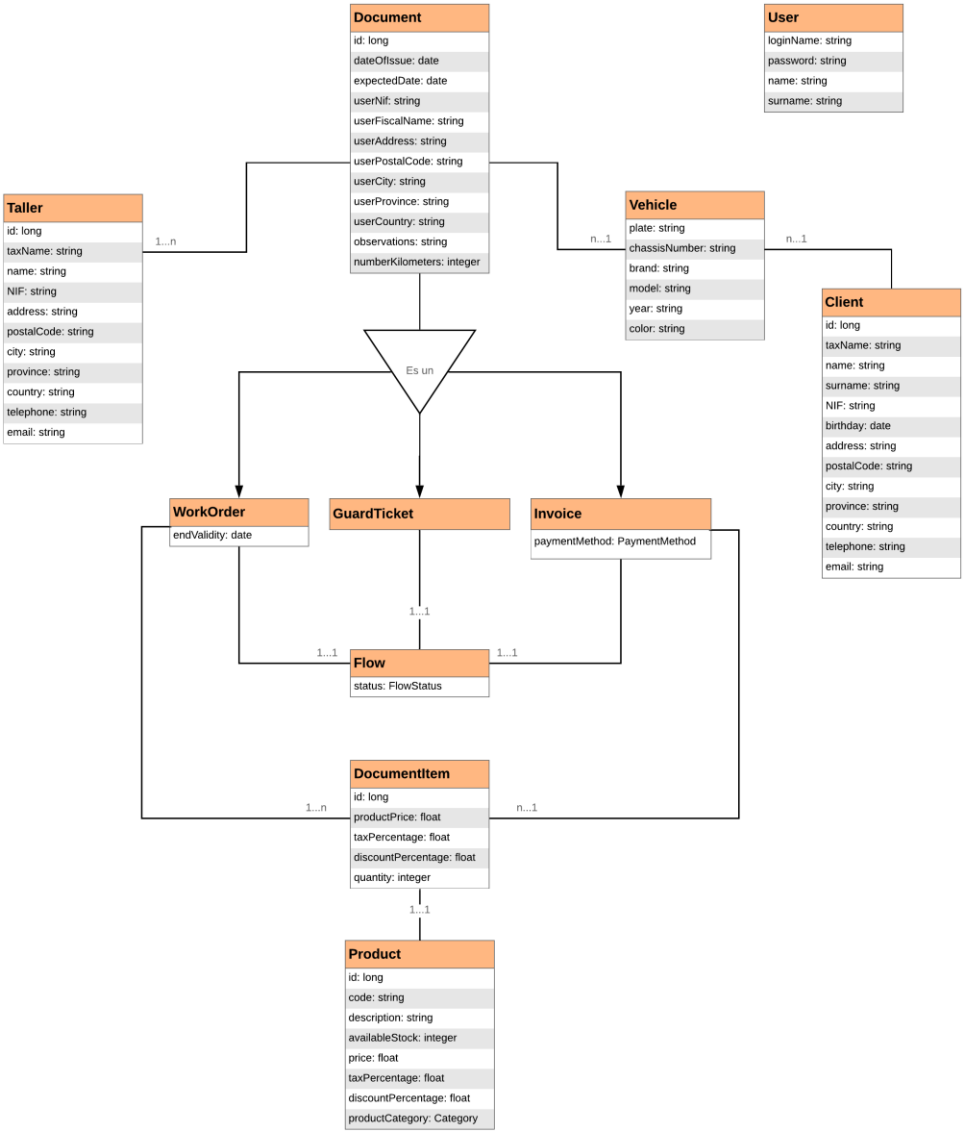


Ilustración 4: Diagrama de Entidad-Relación de la Base de Datos

2.2 Frontend

Esta capa se ha organizado bajo las directrices de Angular, por lo que la capacidad de decisión se ha limitado a meros criterios de agrupamiento por tipo de elemento. Por lo tanto, para explicar el planteamiento escogido, se expone primero la arquitectura propuesta por Angular y posteriormente se explica los criterios de agrupamiento:

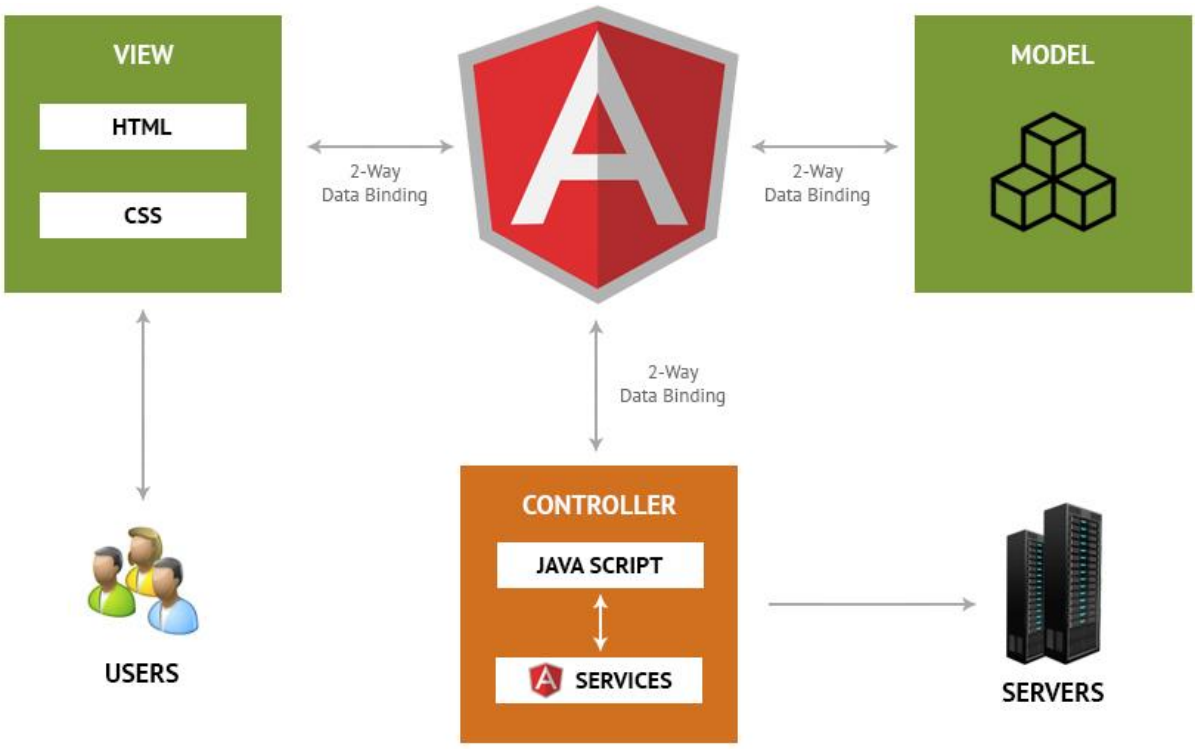


Ilustración 5: Arquitectura Angular

Antes de explicar la imagen, conviene saber qué es el concepto de componente en Angular, ya que se mencionará a continuación. Un componente en Angular es un módulo que consta de una plantilla HTML, una página de estilos CSS/SASS/SCSS y un JavaScript/TypeScript que le otorga inteligencia. Se puede considerar un componente como un ente gráfico (con más o menos inteligencia) susceptible de ser utilizado varias veces en la aplicación. Teniendo en cuenta la ilustración anterior, un componente tendría presencia tanto en el bloque *View* (la plantilla HTML y la página de estilos) como en el *Controller* (JavaScript/TypeScript).

Partiendo de esta introducción, se puede observar como en la imagen tenemos tres bloques diferenciados (Modelo-Vista-Controlador):

- El modelo: son las entidades que se manejan en Angular. Son los objetos a tratar, cuyos datos se cargarán a través de llamadas al backend (el cual accederá a la base de datos) o se rellenarán a través de un formulario.
- La vista: son las diferentes páginas web más sus estilos que configuran la interfaz gráfica de la aplicación. Normalmente mostrarán información precargada en entidades del modelo, o formularios para cargar los datos en dichas entidades.

- El controlador: conformado por los diferentes elementos que componen la lógica de la aplicación: los servicios, que son los objetos que se utilizarán para hacer llamadas al backend y obtener/persistir la información, y los JavaScript (TypeScript en este proyecto) que otorgan inteligencia a los componentes Angular.

Tras esta breve explicación, se comenta la solución propuesta en cuanto a la estructura de ordenación del proyecto. En la carpeta de la aplicación se han distribuido los diferentes componentes agrupados por el objeto del modelo que principalmente utilizan. Así el componente encargado de reflejar el listado de clientes, el del formulario de creación de clientes y el de visualización/edición de estos, se agrupan en la carpeta *clients*. Los servicios se han metido en la carpeta *services* dentro de la carpeta de aplicación, en una carpeta con el nombre de la entidad que utilizan, donde además se han insertado, las entidades del modelo que referencian los propios servicios. Este es el diagrama que resume la estructura:

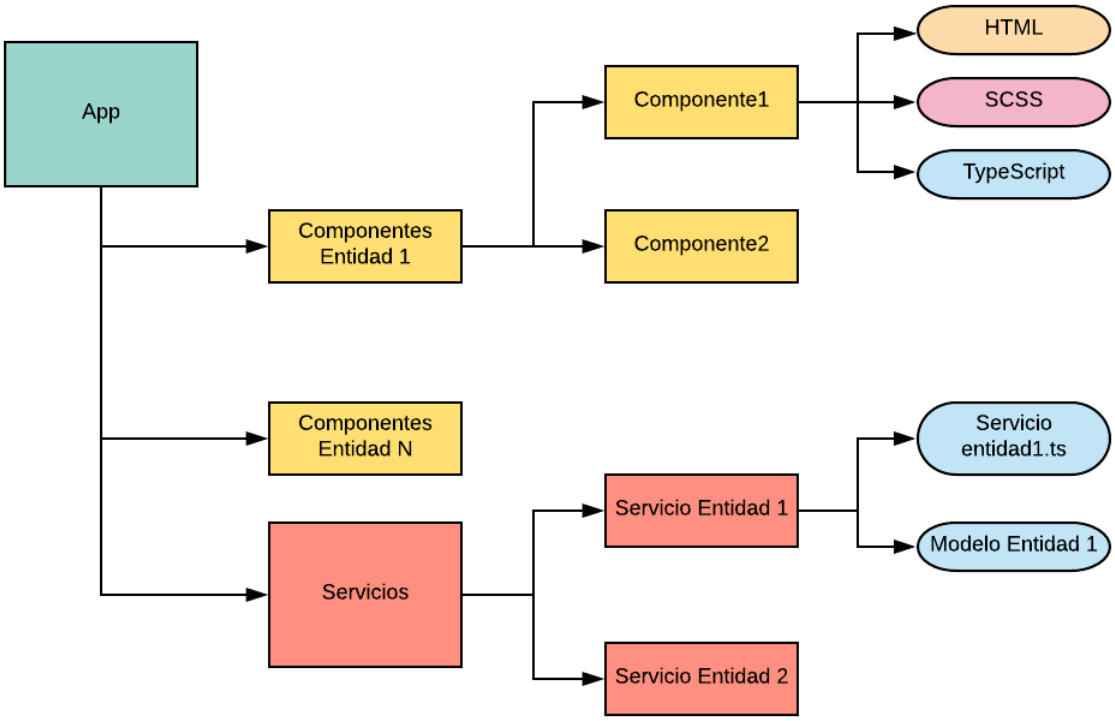


Ilustración 6: Estructura del frontend

2.3 Diagrama de navegación

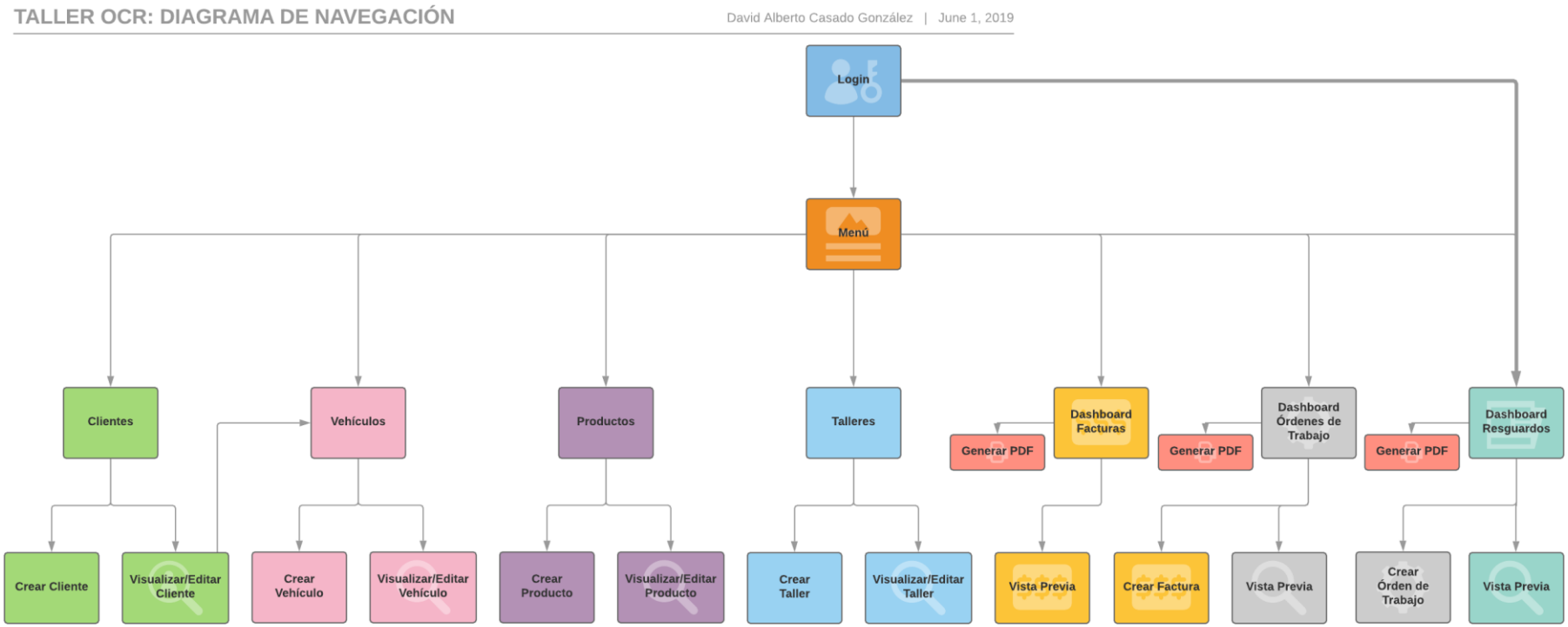


Ilustración 7: Diagrama de navegación

3 Diseño gráfico e interfaces

En este proyecto se han realizado bocetos en las fases delimitadas por dos hitos de la planificación para definir cómo sería visualmente la aplicación una vez finalizada:

- El primero de ellos fue el hito "PEC4", en el que se finaliza la lógica del frontend sin tener apenas en claro la maquetación deseada. Los bocetos fueron realizados con la herramienta Balsamiq (Balsamiq, s.f.) y representan una idea generalizada de cómo podrían ser las pantallas, las cuáles distan bastante de lo que se ha realizado finalmente. Estos bocetos se pueden observar en el fichero "balsamiq_sketches.pdf" que se incluye dentro del apartado de bocetos y sketches mencionados en el Anexo B de este documento.
- El segundo de ellos se trata del hito "PEC5", en el que se da por finalizada la aplicación. Al contar con un tiempo muy limitado por primar el desarrollo de la propia aplicación, estos bocetos de lo que a priori es la interfaz definitiva se han realizado sin un software de apoyo como podría ser Justinmind (Justinmind, s.f.) o Axure (Axure, s.f.). Este hecho penaliza el tener una idea más realista de lo que puede ser la aplicación final, pero agiliza la ejecución del proyecto notoriamente. Se pueden observar los bocetos de las principales pantallas de la aplicación en la documentación asociada al apartado de bocetos y sketches mencionados en el Anexo B de este documento.

3.1 Estilos

En todo momento el diseño de la interfaz ha perseguido la sencillez, evitando exceso de información en pantalla para no causar rechazo por el usuario. Además, se ha utilizado las clases proporcionadas por Bootstrap (Bootstrap, s.f.) en todas aquellas pantallas en las que fue posible para, además de agilizar el desarrollo, permitir una vista responsive y sencilla.
A continuación, se exponen otros aspectos que merece la pena reseñar dentro del apartado estilístico:

3.1.1 Logotipo

En la aplicación hay dos logotipos y se muestra uno u otro dependiendo del fondo en el que se presenta. Así pues, el primero de ellos (el oscuro) se presenta en la pantalla de login, que tiene un fondo claro, mientras que el segundo se muestra en el menú lateral que tiene fondo oscuro.

Ilustración 8: Logotipo con fondo oscuro

Ilustración 9: Logotipo con fondo claro

3.1.2 Paleta de colores

Para escoger los colores se ha buscado un conjunto que combine, con un color destacado para la cabecera (que se convierte por ende en el color a identificar como propio de la compañía), un menú de un color oscuro y destacable frente al fondo de pantalla, que siempre va a ser blanco. Los colores exactos de la plantilla son:

Menú zona del logo (#23282e)	Menú (#2e353d)	Cabecera (#9acd32)	Menú hover (#4f5b69)

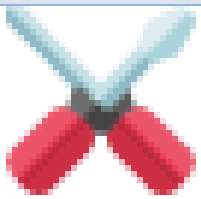
Tabla 3: Paleta de colores

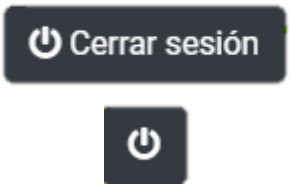
3.1.3 Tipografía

En toda la aplicación se utiliza la misma fuente: Roboto (Google, s.f.) ya que es una fuente clara y profesional. Por lo general, el tamaño de la fuente de los elementos textuales es heredado de la librería Bootstrap. No obstante, y especialmente para aclimatar las fuentes a los tamaños más pequeños de pantalla, en ocasiones se reduce el tamaño de la fuente a un 80% de su tamaño en pantalla grande.

3.1.4 Elementos gráficos

En este apartado contamos tanto con iconos como con botones de acción. A continuación, los enumeramos:

Elemento	¿Dónde se encuentra?	¿Qué significa? ¿Para qué sirve?
	Se encuentra en la parte superior del menú lateral presente en resoluciones superiores a 768px	Se trata del logo de la compañía.
	Se encuentra en la página de Login.	Se trata del logo de la compañía.
	En la pantalla del listado de talleres en resoluciones superiores a 1200px de anchura	No tiene un significado especial. Representa un icono que sirve de logo del taller al que precede
	Se encuentra en la cabecera en resoluciones superiores a 480px de anchura	No tiene un significado especial. Representa un icono que sirve de logo del taller escogido en el selector
	En la cabecera en resoluciones superiores a 506px de anchura	Indica que el usuario está logado. Además, se añade posteriormente un saludo indicando el nombre del usuario
SPARKSUITE	En la vista previa y facturas de los diferentes documentos	Representa el logo del taller.
	En la cabecera en todas las resoluciones	Es el botón que actualiza el listado de talleres a seleccionar

		para indicar el taller actual. Se debe pulsar si se modifica o crea un nuevo taller
	En la cabecera. El primero de ellos en resoluciones superiores a 768px mientras que el segundo en las inferiores	Son los botones encargados de cerrar la sesión del usuario, hecho que le devolverá al login de la aplicación
	En la pantalla del dashboard de resguardos en todas las resoluciones.	Es el botón para efectuar la recepción de un nuevo vehículo
	En las pantallas de los dashboards de los diferentes documentos en todas las resoluciones	Sirve para generar un fichero PDF del documento
	En las pantallas de los dashboards de los diferentes documentos en todas las resoluciones y en los listados de clientes, vehículos, productos y talleres en resoluciones reducidas	Se utiliza para abrir una vista previa del documento o para ver el formulario de la entidad seleccionada
	En el dashboard de resguardos y en el de órdenes de reparación en todas las resoluciones	Sirve para avanzar en el flujo de reparación, de forma que de un resguardo saltemos a la creación de una orden de reparación, y de ésta última a la creación de una factura
	En las pantallas de los dashboards de los diferentes documentos en todas las resoluciones y en el listado de clientes. En este último caso únicamente en las resoluciones inferiores a 992px	Se utiliza para enviar un correo electrónico al usuario notificándole la creación del documento específico y adjuntándoselo en el proceso. En el caso del botón mostrado en la lista de clientes, se utiliza para enviar un correo electrónico definido en una ventana modal

	En el dashboard de resguardos y en el de órdenes de reparación en todas las resoluciones	Cancela el flujo de reparación de forma que no salga en los dashboards.
Generar PDF	En las vistas precios de los tres tipos de documentos en todas las resoluciones	Genera el pdf del documento en visualización
Volver	En los diferentes formularios de la aplicación, además de en las vistas previas de los documentos, en todas las resoluciones	Se utiliza para retroceder a la pantalla anterior.
Nuevo cliente	En la pantalla del listado de clientes en todas las resoluciones	Redirige al formulario de creación de un cliente
Enviar email	En la pantalla del listado de clientes en todas las resoluciones inferiores a 992px de anchura	Abre una ventana modal donde, estableciendo un asunto y un cuerpo de mensaje, se envía un correo electrónico al cliente afectado.
Ver/Actualizar	En las pantallas de los listados de clientes y productos en resoluciones superiores a 992px en el primer caso y 768px en el segundo	Redirige al formulario de visualización/edición de la entidad seleccionada
Ver Vehículos	En la pantalla del listado de clientes. En el primer caso en resoluciones superiores a 992px de anchura y en el segundo en las inferiores	Redirige al listado de vehículos del cliente seleccionado
Guardar	En los formularios de creación y edición de clientes, vehículos, productos y talleres.	Guarda la entidad seleccionada con los datos insertados en el formulario.
Nuevo vehículo	En la pantalla destinada al listado de vehículos	Redirige al formulario de creación de un nuevo vehículo
Nuevo producto	En la pantalla del listado de productos	Redirige al formulario de creación de un nuevo producto

+ Nuevo taller	En la pantalla del listado de talleres	Redirige al formulario de creación de un nuevo taller

Tabla 4: Elementos gráficos presentes en la aplicación

3.2 Usabilidad/UX

Desde un comienzo (en parte gracias al estudio de las alternativas en el estado del arte) se planteó el diseño de una interfaz sencilla, intuitiva y de fácil navegación. Por ello se ha estructurado la aplicación de forma que no haya una acumulación excesiva de datos, con un menú sencillo y visible en todo momento en PC, y desplegable en versión móvil con el objetivo de mostrar el panel de información al mayor tamaño posible.

La paleta de colores ofrece contrastes en los contenidos para facilitar la lectura de los elementos y no presentar problemas a los posibles usuarios con defectos visuales.

Todos los botones vienen acompañados de iconos que ayudan a interpretar la acción del propio botón. Además, todos ellos contienen tooltips para que, en la versión de PC, los usuarios puedan leer la información asociada al botón de forma que solvente cualquier duda.

Gracias a incluir sólo los elementos clave de la gestión de un taller mecánico de un tamaño reducido, el aprendizaje es muy rápido. El usuario, gracias al menú, se ubicará rápidamente en la sección relacionada con la operación que quiere realizar. Una vez dentro de la sección se le presentará la información de forma ordenada, por lo que encontrar la funcionalidad no le costará en exceso, siendo fácil de memorizar y recordar la siguiente vez que necesite dicha funcionalidad.

Las urls, aun siendo en inglés son amigables (es decir, descriptivas), de forma que el usuario sabe en todo momento qué pantalla está visualizando y/o qué operación se está llevando a cabo.

En cuanto a las formas de interacción, en todo momento se incluye un pie de página con el buzón de soporte para ponerse en contacto con el administrador de la herramienta, con el objetivo de comunicar errores y aportar sugerencias de mejora.

4 Lenguajes de programación y APIs utilizadas

4.1 Software de desarrollo

4.1.1 Backend

De entre las dos opciones que veremos a continuación se decidió desarrollar el producto en la segunda de ellas (Java), porque, además de dar más versatilidad a los lenguajes utilizados en el proyecto, da mayor apariencia de robustez para dar soporte de backend a aplicaciones web.

4.1.1.1 Node.js

Como se puede observar en su página (Node.js, s.f.):
"Concebido como un entorno de ejecución de JavaScript orientado a eventos asíncronos, Node está diseñado para construir aplicaciones en red […]"
Por lo tanto, el JavaScript generado por este entorno tiene un objetivo notoriamente diferente al que utilizamos habitualmente, ya que en este caso es ejecutado en el lado del servidor.
La instalación de Node.js suele venir acompañada de la instalación del gestor de paquetes "npm" ya que facilita la instalación y compilación de los módulos a utilizar en Node.

4.1.1.2 Java

Java es uno de los lenguajes de programación más utilizados en el mundo. Se trata de un lenguaje orientados a objetos caracterizado por ejecutarse sobre una máquina virtual (JVM), permitiendo la ejecución de sus programas ya compilados en cualquier dispositivo que cuenta con dicha máquina virtual instalada.
Este lenguaje cuenta con una infinidad de librerías que, una vez importadas, aumentan sus posibilidades enormemente. Existen herramientas como Maven que nos permiten gestionar estas librerías (dependencias del proyecto), construir paquetes para desplegar en el servidor, etc.

4.1.2 Frontend

Para elegir la tecnología con la que realizar el frontend del producto hubo más problemas ya que las tres alternativas estudiadas eran totalmente válidas y las tres exigían un alto componente de aprendizaje para su desarrollo. No obstante, se contaba con una mínima experiencia laboral en Angular (y por lo tanto los conocimientos adquiridos se podrían aplicar en proyectos futuros a corto-medio plazo), y, además, dicho framework cuenta con una gran comunidad de desarrolladores para poder aclarar dudas que puedan surgir durante el desarrollo, hecho que hizo decantar la balanza a su favor.
A continuación, se incluyen las tres tecnologías que formaron parte del estudio inicial:

4.1.2.1 React

Se trata de un framework creado por Facebook que nos permite crear interfaces de usuario a partir de componentes que poseen un ciclo de vida (esto permite refrescar la información desplegada por pantalla si alguna propiedad del componente varía). Aporta una característica específica: la posibilidad de utilizar el lenguaje JSX, cuya sintaxis es una especie de mezcla entre JavaScript y HTML, que facilita la programación de los componentes e interpretación de su código.

4.1.2.2 Vue.js

Vue es un framework progresivo, lo que implica que podamos usarlo para partes específicas de un proyecto, o bien para todo el desarrollo del frontal. Es muy similar a React aunque tienen diferencias de sintaxis y, lo que es más importante, de arquitectura (como por ejemplo la distribución template-script-style por fichero vue versus JSX+imports CSS). Aunque en aplicaciones sencillas no es necesario utilizar componentes, estos son los que le dan potencia al framework y que será imprescindible utilizar en aplicaciones de mayor envergadura.

4.1.2.3 Angular

Angular es un framework mantenido por Google basado en componentes y diseñado en TypeScript (los dos frameworks anteriores utilizan EcmaScript 5-6). Es un framework mucho más sólido y menos flexible que sus competidores. No obstante, tiene una característica muy ventajosa y es que cuenta con inyección de dependencias, de forma que podamos inyectar servicios dinámicamente al componente que los necesite.

4.2 Software de diseño

Para la realización de los sketches iniciales del frontend antes de su implementación tras el hito “PEC3”, se utilizó Balsamiq (Balsamiq, s.f.), un software muy sencillo de utilizar donde se pueden hacer diseños de las pantallas a un nivel de conceptualización, de forma que se pueda tener una vaga idea de cuál será el aspecto del producto final.
Los diagramas de navegación desarrollados para este documento se han realizado con Lucidchart (Lucidchart, s.f.), herramienta muy sencilla online para la realización de diagramas.

4.3 Apis de terceros y herramientas

4.3.1 Optica Character Recognition (OCR)

Para escoger la tecnología de reconocimiento de matrículas se optó por la librería más especializada para ello, OpenAlpr. Además, la inclusión de un API REST para poder hacer peticiones online utilizando una cuenta gratuita hizo que la elección fuera mucho más sencilla, ya que facilita enormemente el trabajo. A continuación, se expone las dos alternativas que se valoraron:

4.3.1.1 Tesseract

Esta librería de código abierto desde 2005, es una de las más populares actualmente en lo que a OCR respecta. Está codificada en C++, pero existen varios proyectos como JavaCPP Presets (The missing bridge between Java and native C++ libraries ,2013-2019), proyecto donde se conecta el mundo Java con las librerías más utilizadas de C/C++, entre ellas Tesseract. Además, existe una versión en JavaScript denominada Tesseract.js (Pure Javascript OCR for 62 Languages, 2015-2019), que, como podemos ver en How to convert images to text with pure JavaScript using Tesseract.js (2018, 25 de diciembre) permite digitalizar textos utilizando “training data” para interpretar el texto en una lengua específica.

4.3.1.2 OpenAlpr

Se trata de una librería especializada en la lectura de matrículas. Está basada en Tesseract y OpenCV y tiene tanto una vertiente de pago como una de código abierto. Se ha codificado en C++, pero existen conectores en múltiples lenguajes como Java, C# o Node.js.

Para este proyecto se utilizará la vertiente de pago, la cual tiene una demo gratuita que ofrece consultas a una API que proporcionan en la nube. El número de consultas es bastante elevado, por lo que se considera adecuado para el proyecto actual.

4.3.2 Librería de creación de PDF/HTML

Uno de los aspectos clave de la aplicación es la generación de documentos para plasmar los resguardos, órdenes de reparación y facturas. Para ello se probó en primera instancia la librería IText (Yáñez Novo, s.f.). El problema de utilizar esta librería es que el diseño es muy laborioso, teniendo que implementar, usando clases propias de la librería, cada uno de los elementos de los que constará el PDF.

Por ello se decidió probar con otro sistema: ThymeLeaf + FlyingSaucer (Uhrig, s.f.). A través de esta alternativa se generan una serie de plantillas HTML donde se inyectarán tanto los datos como las etiquetas de los mismos (que se han hecho configurables a partir de un JSON) y posteriormente se generará el PDF a partir de ellas. La solución es mucho más flexible y permite modelar mucho más fácilmente un modelo más atractivo.

4.4 Software base

Dentro del software que se ha utilizado para realizar el producto, se distinguen dos categorías:

4.4.1 IDEs

Para realizar la codificación de la solución se han utilizado dos aplicaciones diferentes. Esta decisión es meramente organizativa, ya que parecía más ordenado tratar el backend con una aplicación y el frontend con otra de forma que la navegación por los diferentes objetos para su edición fuese más rápida. Estas han sido las dos aplicaciones utilizadas:

- Intellij Idea: Se trata de un IDE propietario muy completo, lleno de funcionalidades que ayudan a incrementar la productividad del desarrollador. Por ello fue el software escogido para realizar el backend.
- VS Code: se trata de la alternativa gratuita de Microsoft a la hora de desarrollar aplicaciones. Incluye multitud de add-ons instalables que personalizan el comportamiento y funcionalidades incluidas. Este ha sido la aplicación escogida para desarrollar el frontend.

4.4.2 Servidor de aplicaciones

Para servir una aplicación JAVA hay múltiples opciones, pero de entre las alternativas de software libre WildFly es una opción con garantías. Desplegar un war en esta aplicación es muy sencillo y exige una configuración prácticamente nula. Además, la integración con IDEs como Intellij Idea para activar la

depuración es realmente fácil. Por ello, este software ha sido el escogido desde el primer instante para realizar este cometido.

Capítulo 4: Implementación

1 Detalles de la implementación del producto

El desarrollo de la solución ha supuesto, como es lógico, el grueso del tiempo empleado en el proyecto. Por ello, ha sido una fuente constante de investigación y pruebas para acercar el producto gradualmente a lo que significaba cumplir los objetivos marcados desde el inicio del proyecto. A continuación, se exponen detalles más concretos de aspectos que se consideran más novedosos o bien que se ha implementado previa investigación, tanto en el back como en el frontend.

1.1 Backend

1.1.1 Seguridad y encriptación

Para este proyecto se ha decidido utilizar un sistema de encriptación de contraseñas avanzado, con un mecanismo de defensa ante ataques de fuerza bruta a través de una ralentización a la hora de generar el hasheo de la contraseña. En concreto se va a utilizar PBKDF2, siguiendo los pasos marcados en el artículo de Gupta, Lokesh (Gupta, s.f.).
Siguiendo el hilo de la seguridad, se estudió la posibilidad de securizar las operaciones REST utilizando JSON Web Tokens. La implementación de este sistema se declinó ya que añadía una complejidad extra que, con el reducido tiempo para completar el proyecto del que se dispone, ha sido imposible asumir. En su lugar se utiliza un método de autenticación básico. No obstante, este punto se propone como mejora del producto para una posible futura versión.

1.1.2 DozerBeanMapper

Un aspecto clave para seguir los patrones de diseño escogidos en el diseño del backend y comentados anteriormente en la sección 2.1 Backend, es la utilización del DozerBeanMapper para facilitar el mapeo entre las entidades de negocio y las de base de datos. Esta librería Java permite mapear unas clases con otras por “convención”. Esto quiere decir que, si tenemos dos clases que queremos mapear cuyos atributos y getters/setters se llaman igual, DozerBeanMapper puede hacer la transformación sin configuración alguna. De necesitarse una configuración especial (por ejemplo, no llevarse un campo en la transformación de tipos, como puede ser la contraseña del usuario alojada en base de datos) se debe configurar en un XML que debe seguir una estructura específica (Baeldung, 2018).

1.1.3 Servicios REST

Aunque no sea una funcionalidad “novedosa” se considera interesante su explicación ya que se podría decir que estos conforman el núcleo del backend. Teniendo en mente la Ilustración 3 donde se describe la arquitectura Clean, se va a proceder a detallar los pasos que se sigue cada vez que una llamada REST ataca el backend, tomando como ejemplo la obtención de los datos de un cliente:

1. El front (o el Rest Client) ataca al endpoint [host]/taller/rs/clients/{nº id de cliente} con una llamada tipo GET (obtención de datos).

2. El Resource que responde a las peticiones efectuadas a los endpoint [host]/taller/rs/clients es el ClientsResource. En él se ven los métodos:

Método	Tipo	Path	Cuerpo del mensaje
getClient	**GET**	**/{nº de id de cliente}**	Vacío, es un get
getAll	GET	/	Vacío, es un get
update	PUT	/{nº de id de cliente}	Datos modificados
persist	POST	/	Datos a insertar
getByClientId	GET	/{nº de id de cliente}/vehicles	Vacío, es un get

Tabla 5: Métodos con endpoints dentro del endpoint de clientes

Como se puede observar, sólo hay un método cuya ruta sea exactamente */nº* y que sea de tipo GET, por lo que ese será el método que responda a la llamada.

3. El Resource utiliza el UseCase "GetClient" para encaminar la solicitud.
4. UseCase recibe la solicitud y la redirige al Repository de clientes.
5. El Repository, llama al DAO, que es el encargado de hacer la petición a la base de datos. En este punto actúa el DozerBeanMapper ya que el DAO le devuelve una entidad del modelo de base de datos, y se espera que este método devuelva una entidad de la lógica de negocio.
6. Los datos (en forma de Cliente) suben hasta el Resource, el cual construye un JSON y lo envía como respuesta.

Para ver todas las llamadas que admite el sistema se puede consultar el documento "Servicios REST" incluido en el Anexo B: Entregables del proyecto.

1.2 Frontend

1.2.1 Uso de localStorage

Para mantener ciertos datos almacenados durante la sesión iniciada al logarse en la aplicación se utiliza el almacenamiento proporcionado por el navegador en el localStorage. Gracias a él podemos saber en todo momento si estamos logados (gracias a la variable currentUser) y el taller seleccionado en la cabecera (para filtrar los documentos de los dashboards en función del taller seleccionado).

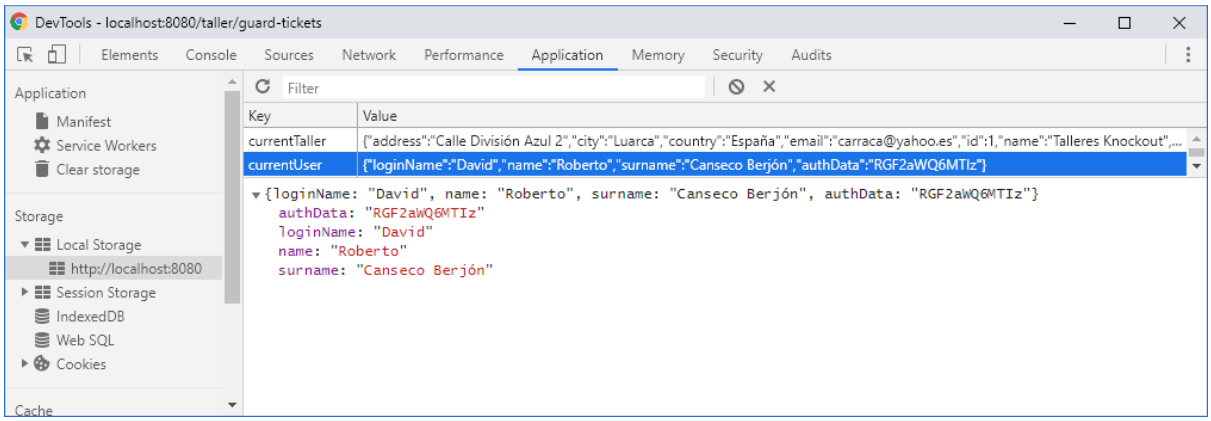


Ilustración 10: LocalStorage

1.2.2 CanActivate

Como se puede ver en la página de Angular (Angular, s.f.), CanActivate es una interfaz implementable por los componentes y que determina si el encaminador de Angular puede o no abrir la página. En el caso de este proyecto, todas las pantallas son accesibles sí y sólo sí el usuario está logado en el sistema y por tanto hay un registro en el localStorage para respaldarlo. De no haberlo, independientemente de la página a la que se quiera acceder, siempre se redireccionará al usuario a la pantalla de login (almacenándose la url de retorno si el login se efectúa correctamente).

1.2.3 Buscadores y Date Inputs

Los buscadores son los componentes gráficos más sencillos, pero con la funcionalidad más compleja y específica de Angular de la aplicación. Estos son inputs que, al cumplimentarlos, hacen una llamada a un servicio y retornan las entidades que son el resultado de la petición realizada.

Entrando en términos más técnicos, la mayor dificultad radica en la comunicación entre los dos componentes que intervienen: el padre, es decir, el componente que incluye el buscador, y el hijo, el propio buscador. Para realizar dicha comunicación nos basamos en etiquetas de Angular como @Output, que, apoyándose en la emisión de entidades que proporciona la clase EventEmitter, se encarga de devolver los resultados al padre.

Los Date Inputs de HTML5 se han insertado en los diferentes componentes Angular de la misma forma que los buscadores: con componentes incrustados siguiendo la estructura padre-hijo. Esto se ha tenido que realizar así por la necesidad de convertir el valor que devuelve este tipo de estructuras en un valor que sepa interpretar el backend. Se ha decidido utilizar la fecha en milisegundos para la transferencia de datos entre el front y el back (en ambos sentidos) por su precisión y sencillez en la traducción.

2 Requisitos de instalación

Para ejecutar el producto entregado en este proyecto, se debe contar con un servidor de aplicaciones JAVA para desplegar el archivo *.war* que empaqueta la compilación de los ficheros fuente. En la carpeta determinada en la sección Anexo B asociada al WildFly, se encuentra la versión 16.0.0 de WildFly (WildFly, s.f.) utilizada tanto en el desarrollo como en las pruebas y preparado para, con un

mínimo de ajustes en la configuración, ser exportada a otros equipos. Esto nos permite pues seguir dos vías de instalación:

- La más sencilla: utilizar el WildFly proporcionado que ya contiene la última versión del software.
- La más completa: instalar el servidor de aplicaciones desde cero (preferiblemente WildFly por contar con una base de datos embebida H2 que es la que utiliza la aplicación), configurarlo y desplegar el *.war* proporcionado. Aunque parezca mucho más tedioso, el software es muy fácil de instalar, y las configuraciones son básicas. No obstante, la aparición de problemáticas es superior a la que ofrece la anterior solución, la cual ha sido probada en numerosas ocasiones.

3 Instrucciones de instalación

3.1 Instalación sencilla

Se debe copiar la carpeta *wildfly* proporcionado en la entrega en una ubicación donde se disponga de plenos permisos de escritura. Posteriormente se deben hacer las siguientes modificaciones en la configuración del servidor:

- Modificar la ubicación del log: se debe acceder al fichero ubicado en [UBICACIÓN ESCOGIDA]\wildfly-16.0.0.Final\standalone\configuration\logging.properties y modificar la sentencia handler.FILE.fileName de forma que la ruta absoluta sea coherente con la ubicación que se ha escogido.
- En el caso de que WildFly dé un error "java.net.BindException: Address already in use: bind /0.0.0.0:9990" se debe cambiar el puerto en el que la consola de administración de WildFly se despliega. Para ello se debe modificar el uso del puerto 9990 por uno que no esté en uso en [UBICACIÓN ESCOGIDA]\wildfly-16.0.0.Final\standalone\configuration\standalone.xml. Esta es la fila que se encarga de la asignación de puertos:

```
<socket-binding name= "management-http" interface="management" port=
"${jboss.management.http.port:9990}"/>
```

Una vez hecho esto, y ubicándonos en la carpeta [UBICACIÓN ESCOGIDA]\wildfly-16.0.0.Final\bin desde la consola de comandos (**nunca** desde el explorador de Windows ya que se tendrían problemas a la hora de generar ficheros) ejecutamos *standalone.bat.* Una vez el proceso acabe se debería mostrar una fila como la siguiente:

```
XX:XX:XX,XXX INFO  [org.jboss.as.server] (ServerService Thread Pool -- 43)
WFLYSRV0010: Deployed "taller.war" (runtime-name : "taller.war")
```

Posteriormente se debe abrir una instancia del navegador Chrome (la aplicación está optimizada para el uso en este navegador) e introducir la url: http://localhost:8080/taller/.

3.2 Instalación completa

Accediendo a la página de descargas de WildFly (WildFly, s.f.) se puede observar las diferentes versiones del software. Para replicar exactamente el entorno que se ha utilizado para desarrollar y publicar el producto, se debe utilizar la versión 16.0.0 (cuya descripción es Java EE Full & Web Distribution).

Una vez descargado y descomprimido se debe modificar el fichero standalone.xml ubicado en [UBICACIÓN ESCOGIDA]/standalone/configuration/ con el fin de poder habilitar las conexiones a través de terminales móviles conectados a la misma red local. En este fichero, se debe localizar el bloque de interfaces "<interface name="public">" de forma que quede de la siguiente forma:

```
<interface name="public">
      <any-address/>
</interface>
```

Una vez realizada esta configuración, se debe copiar el fichero .war proporcionado en la carpeta designada en el apartado WAR enunciada en el Anexo B: entregables del proyecto en [UBICACIÓN ESCOGIDA]/standalone/deployments. Acto seguido, y desde la consola de comandos, se debe ejecutar el standalone.bat ubicado en [UBICACIÓN ESCOGIDA]\wildfly-16.0.0.Final\bin. Una vez el proceso acabe se debería mostrar una fila como la siguiente:

```
XX:XX:XX,XXX INFO  [org.jboss.as.server] (ServerService Thread Pool -- 43)
WFLYSRV0010: Deployed "taller.war" (runtime-name : "taller.war")
```

Posteriormente, se debe abrir una instancia del navegador Chrome (la aplicación está optimizada para el uso en este navegador) e introducir la url: http://localhost:8080/taller/

Capítulo 5: Demostración

1 Instrucciones de uso

Una vez tenemos levantado el servicio en el servidor local (ver Instrucciones de Instalación) se debe abrir una instancia de Google Chrome e introducir en ella la url: http://localhost:8080/taller/. Si el usuario no tiene una sesión abierta, se le redireccionará a la página de login, donde, a menos que haya creado un usuario por REST creando el suyo propio (ver el documento “manual de instalación” indicado en el apartado Anexo B: Entregables del proyecto), deberá introducir el login de ejemplo y que se muestra por pantalla: David/123. Una vez logados, el usuario puede operar con total normalidad dentro de la aplicación.

2 Ejemplos de uso del producto (o guía de usuario)

Se ha desarrollado una extensa guía de usuario explicando las principales operaciones que se pueden realizar en la aplicación. Esta se encuentra accesible en la carpeta indicada en el apartado “Guía de Usuario” enunciado en el apartado Anexo B: Entregables del proyecto.

Capítulo 6: Conclusiones y líneas de futuro

1. Conclusiones

1.1 Lecciones aprendidas

Este proyecto se ha desarrollado de forma simultánea a diversas tareas como la realización de otras asignaturas, el trabajo, una serie de eventos ajenos al estudio y, ante todo, el aprendizaje de las tecnologías que se iban a emplear. A pesar de que suelo pecar de optimista en mis planificaciones y que, además de intentar abarcar mucho, tiendo a ser bastante perfeccionista con el resultado obtenido, el producto ha salido en base a horas y horas de esfuerzo.

El mayor problema que he tenido ha sido el tener que autoformarme para poder desarrollar la tarea, por lo que creo que, en futuros proyectos en los que precise de formación, priorizaré ésta colocándola en una fase previa al desarrollo del proyecto. Esta secuenciación facilitará en gran medida el desarrollo del proyecto y generará menos estrés a la hora de desarrollar y completar los entregables en fecha. Aplicando esta mejora se podría orientar el proyecto hacia una metodología ágil siguiendo los cánones establecidos y realizando entregas periódicas plenamente funcionales.

1.2 Reflexión completitud objetivos

Considero que los objetivos inicialmente marcados se han satisfecho a grandes rasgos (especialmente los personales). No obstante, siempre tengo mis dudas acerca del apartado de usabilidad. Siempre intento seguir las recomendaciones establecidas (utilizar colores con contrastes, tamaño de la fuente adecuado, no incluir elementos con movimiento, agregar iconos fácilmente reconocibles a los botones…) pero aun así nunca estoy seguro de haber cumplido con lo que se espera en este apartado.

He de indicar que, aunque no se reflejase en los objetivos, estaba dentro de mis planes incluir una base de datos “física” en el despliegue, de forma que los datos persistieran en base de datos y no se generen con cada arranque del servidor al utilizar una base de datos embebida. Finalmente se ha desistido ya que, además de complicar la instalación del sistema, aumentaría en gran medida el tamaño de la entrega. Este punto se ha destacado como posible mejora futura.

1.3 Seguimiento de la planificación

A grandes rasgos la planificación se ha seguido. No obstante, ha hecho falta replanificar más de un apartado, por motivos de autoformación o por imprevistos que han obligado a retrasar el inicio de las tareas. Aun así, en un alto porcentaje de replanificaciones se han corregido empleando más horas de las planificadas en los días consecutivos.

La metodología seguida, aun a día de hoy, creo que es la única que podría haber seguido si quería utilizar tecnologías en las que pudiese aprender algo que fuese reutilizable en futuros proyectos personales. Al haber hitos de entrega establecidos, el adelantar la formación no se vio viable, por lo que parece que la estrategia seguida fue la correcta. De todas formas, considero que no es una estrategia válida para proyectos reales ya que al final de cada hito el producto entregado no es un

parcial del total plenamente funcional, pero, al tratarse de un proyecto formativo (y más cuando he tenido que autoformarme en su desempeño) podría considerarse viable.

El establecimiento de períodos de reajuste de planificación al comienzo de cada fase marcada entre hitos fue clave para garantizar el éxito de la planificación. Gracias a ellos no ha habido que hacer demasiados cambios en la planificación más allá de ciertos retrasos en los inicios de ciertas tareas.

2. Líneas de futuro

Desde la conceptualización del proyecto hasta el desarrollo de la interfaz gráfica, se han desechado numerosas funcionalidades que servirían para mejorar el producto y/o añadir nuevas funcionalidades interesantes. A continuación, se enumeran todas ellas:

- Creación de usuarios desde la interfaz.
- Reinicio/recordatorio de contraseñas.
- Implementar perspectiva cliente.
- Crear base de datos dedicada, dejando de usar la base de datos H2 embebida en el WildFly.
- Utilizar JSON Web Tokens para securizar las llamadas REST.
- Mejorar el servicio OCR de forma que las matrículas se detecten en la propia cámara sin necesidad de hacer una fotografía.
- Paginación desde el backend.
- Incluir input “número de elementos a incluir” en los listados para limitar la paginación.
- Vincular los usuarios con los talleres y que estos sean a la vez el punto de referencia para el resto de las entidades de forma que, cuando se loga un usuario concreto, sólo salga la información relativa a él. Esto permitiría gestionar cuentas de usuario diferentes.
- Exportación de listados en CSV.
- Importación de logos de talleres.
- Zona de usuario: avatar, datos personales, selector de notificaciones a recibir por correo electrónico…
- Personalizar los correos enviados desde el frontend para que utilicen plantillas.

Bibliografía

Angular. (s.f.). *Angular - CanActivate*. Obtenido de https://angular.io/api/router/CanActivate

Axure. (s.f.). *Axure RP 9 - Prototypes, Specifications, and Diagrams in One Tool*. Obtenido de https://www.axure.com/

Baeldung. (2 de Noviembre de 2018). *A Guide to Mapping With Dozer*. Obtenido de https://www.baeldung.com/dozer

Balsamiq. (s.f.). *Balsamiq Cloud*. Recuperado el 22 de Abril de 2019, de https://balsamiq.cloud/

Boletín Oficial del Estado. (30 de Abril de 2010). *Documento consolidado BOE-A-1986-18896*. Recuperado el 17 de Marzo de 2019, de https://www.boe.es/buscar/act.php?id=BOE-A-1986-18896

Bootstrap. (s.f.). *Bootstrap · The most popular HTML, CSS, and JS library in the world.* Obtenido de https://getbootstrap.com/

Deutsch, D. (13 de Agosto de 2018). *A quick introduction to clean architecture*. Recuperado el 26 de Marzo de 2019, de https://medium.freecodecamp.org/a-quick-introduction-to-clean-architecture-990c014448d2

Doscar. (2010-2019). *Doscar Software de Gestión*. Recuperado el 12 de Marzo de 2019, de https://www.doscar.com/programa-taller/

FuturoInformática. (2017). *Videos formativos programa de talleres*. Recuperado el 12 de Marzo de 2019, de https://www.futuroinformatica.com/videos-formativos-talleres.html

Google. (s.f.). *Roboto - Google Fonts*. Obtenido de https://fonts.google.com/specimen/Roboto

Gupta, L. (s.f.). *Java Secure Hashing - MD5, SHA256, SHA512, PBKDF2, BCrypt, SCrypt*. Obtenido de https://howtodoinjava.com/security/how-to-generate-secure-password-hash-md5-sha-pbkdf2-bcrypt-examples/#PBKDF2WithHmacSHA1

Justinmind. (s.f.). *Free prototyping tool for web & mobile apps - Justinmind*. Obtenido de https://www.justinmind.com/

Lucidchart. (s.f.). *Sitio Oficial de Lucidchart*. Recuperado el 30 de Mayo de 2019, de https://www.lucidchart.com/

Node.js. (s.f.). *Acerca de Node.js*. Recuperado el 15 de Marzo de 2019, de https://nodejs.org/es/about/

TallerAlpha. (s.f.). *Sistema para Taller de Motos*. Recuperado el 12 de Marzo de 2019, de https://www.talleralpha.com/sistema-para-taller-de-motos/

Uhrig, T. (s.f.). *Generating PDFs with Java, Flying Saucer and Thymeleaf (Part 1)*. Recuperado el 05 de Abril de 2019, de https://tuhrig.de/generating-pdfs-with-java-flying-saucer-and-thymeleaf/

WildFly. (s.f.). *WildFly Downloads · WildFly*. Obtenido de https://wildfly.org/downloads/

Yáñez Novo, J. (s.f.). *Java iText PDF – Creando un pdf en Java con iText*. Recuperado el 03 de Abril de 2019, de http://codigoxules.org/java-itext-pdf-creando-pdf-java-itext/

Anexos

1 Anexo A: Glosario

Flujo de reparación: es el conjunto de estados por los que pasa una motocicleta desde su entrada al garaje (Resguardo) hasta su posible recogida (Factura).

OCR: es una tecnología que permite la lectura de caracteres a través de un medio óptico.

Resguardo: es el estado (y el documento asociado al mismo) correspondiente al primer paso del flujo de reparación: la recepción del vehículo en el taller.

Orden de reparación: es el estado (y el documento asociado al mismo) correspondiente al segundo paso del flujo de reparación: evaluación realizada y presupuesto de la reparación preparado.

Factura: es el estado (y el documento asociado al mismo) correspondiente al tercer y último paso del flujo de reparación: motocicleta reparada y reparación abonada.

2 Anexo B: Entregables del proyecto

Guía de Usuario: se puede encontrar en la carpeta *Documentos* del entregable. En este documento se especifican los procesos a seguir en la interfaz para realizar los procesos más importantes de la herramienta.

WAR: se puede encontrar en la carpeta *war* del entregable. Es el paquete de los ficheros fuente del proyecto compilados. Es el fichero que se debe desplegar en caso de realizar una instalación limpia del servidor de aplicaciones JAVA.

Fuentes: se pueden encontrar en la carpeta *fuentes* del entregable. Se trata de los ficheros de código fuente que se han desarrollado a lo largo del proyecto.

Informes de trabajo: estos documentos son los entregables que relatan el proceso de desarrollo desde los inicios de este. Se pueden encontrar en la carpeta con ese mismo nombre.

Sketches y bocetos: se pueden encontrar en la carpeta *Bocetos y Sketches* del entregable, siendo el PDF el diseño realizado en la fase de realización del frontend, mientras que las imágenes que se pueden encontrar en esta misma carpeta son los bocetos realizados en la última fase del proyecto, previo al desarrollo de la interfaz gráfica.

Servicios REST: se pueden encontrar en la carpeta *Documentos.* En este documento se especifican todos los servicios REST expuestos por el backend y los endpoints a los que hay que remitir para ejecutarlos.

WildFly: se puede encontrar en la carpeta con el mismo nombre del entregable. Se trata del servidor de aplicaciones JAVA que se ha utilizado tanto en las pruebas como en el desarrollo del proyecto. Para utilizar esta instalación, se deben seguir los pasos especificados en 3.1 Instalación sencilla.

3 Anexo C: Currículum Vitae

Nacido en 1987 y trabajador desde los 18, he ido compaginando estudios con trabajo hasta ser graduado en Ingeniería Informática de Gestión por la Universidad de Oviedo en 2012. Posteriormente,

en 2013, desempeñé el máster de Formación al Profesorado para, posiblemente en un futuro, instruir a alumnos de FP y/o secundaria en materias relativas a la informática.

Teniendo en cuenta sólo los trabajos asociados al sector informático, tras un breve periodo de unos dos años trabajando en Fi2Net en Gijón, desde hace más de 5 años llevo desempeñando mi trabajo en el área de Procesos de Telefónica España administrando y gestionando los desarrollos de varias herramientas de gestión de la dirección.