Trabajo Fin de Grado

Automatización del proceso de etiquetado de láminas de vidrio

Automation of the glass sheet labeling process

Autor:

Antonio Santa Isabel Bosqued

Directores:

Dra. Dª. Piedad Garrido Picazo

Ángel Silva Sanahuja

Escuela Universitaria Politécnica de Teruel

2021/2022

RESUMEN Y PALABRAS CLAVE

El documento expone todo el proceso de desarrollo de una App Web para su incorporación real en una fábrica.

En él se detalla por qué dicha aplicación es pertinente y necesaria, que tecnologías se han usado para el desarrollo y como se han incorporado e integrado con respecto al resto de la infraestructura de la fábrica. Así como el proceso de análisis, diseño, implementación y una serie de pruebas a dicha aplicación.

El proyecto está dividido en tres módulos, un front-end siguiendo el modelo de Single Page Application encargado de la interfaz de usuario, un back-end servidor que interactúa con el sistema de persistencia y con el front-end, y un sistema de persistencia en forma de base de datos SQL.

Además, de contar con un apartado donde se revisa la legislación actual referente a accesibilidad y los puntos de usabilidad que la aplicación final cumple.

Palabras clave:
[OCR] [front-end] [REST-API] [Single Page Application][back-end]

Tabla de Contenidos

1.Introducción y Objetivos

El proyecto surge a partir un convenio de prácticas en la empresa Turomas S.L. En el que, tras concluir dicho periodo, se comenta la posibilidad de desarrollar una Aplicación Web para suplir una demanda de uno de los clientes, el cual quería agilizar ciertos pasos del almacenamiento de información en sus instalaciones. Dicho problema, es el expuesto a continuación.

El proyecto desarrollado en este TFG busca dar solución a un problema real de una fábrica de corte de vidrio. En dicha fábrica se reciben paquetes de planchas de vidrio, de diversos tamaños, materiales y número de hojas por paquete, además de ser obtenidas de diferentes proveedores. Estos paquetes son almacenados en caballetes en un almacén mediante un cargador aéreo, cuyo proceso de almacenamiento está automatizado, pero se ha de indicar en un ordenador la orden de almacenamiento, y en otro se hace la actualización de inventario. La orden de almacenamiento se hace cada vez que se va descargando un paquete nuevo, mientras que para el inventario, los operarios de la fábrica van apuntando las características de cada paquete en un papel y tras haber almacenado todo el material, proceden a actualizar el inventario.

Llegados a este punto cabe añadir, que los diferentes paquetes llevan diversas etiquetas en función del proveedor y producto. Por lo que, la estructura de la etiqueta es diferente según la marca, en algunos casos, hay varias etiquetas diferentes para un mismo proveedor. Ver Anexo 1

El objetivo de este proyecto consiste en el desarrollo de una aplicación web que permita al operario obtener, mediante una foto, la información pertinente sobre el paquete a almacenar. Permitiendo editarla e indicar el caballete del almacén donde va a ser almacenado el paquete (ver figura 1).

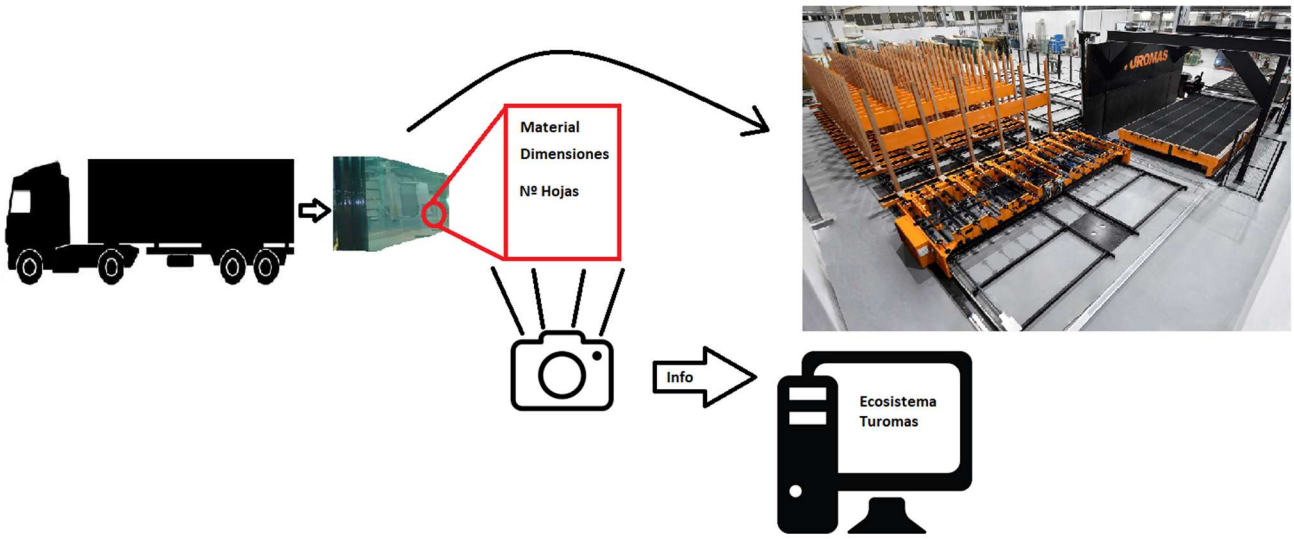

Figura 1. Esquema del Problema

2.Estado del arte

En esta sección se van a analizar y contrastar un conjunto de aplicaciones y herramientas similares, que comparten el uso de técnicas OCR, con la aplicación desarrollada, con la finalidad de concretar la aportación que propone este TFG.

Dichas herramientas son: Nanonets, Acrobat Pro DC, Cognex y Vivino

Producto	Logotipo	Características
Nanonets	Nanonets	-Permite crear flujos de trabajo para extraer únicamente la determinada información dependiendo del tipo de documento con el que se está tratando. -Utiliza IA para leer documentos semiestructurados que no se ven y que no siguen una plantilla estándar como el software de OCR genérico. Valida rápidamente los datos capturados del documento y la IA tiene un proceso de aprendizaje automatizado.

Adobe Acrobat Pro DC	Acrobat Pro DC	-El software se caracteriza por una conversión OCR automáticamente de cualquier documento o imagen en papel escaneado a texto editable dentro de un PDF. -Exportar a documentos de MS Office y transformar a otros tipos como .docx .txt .ppt o .xls. -Incluye la posibilidad de corregir palabras no reconocidas correctamente, mostrando la imagen del reconocimiento OCR dudoso que se puede corregir o confirmar. -Se puede usar desde el teléfono móvil.
Cognex	COGNEX	-*In-Sight*: un sistema de visión que combina una cámara, un software y un procesador en una unidad compacta. -Cuenta con un software de aprendizaje profundo dentro de una cámara inteligente.
Vivino	VIVINO	-Aplicación móvil. -Disponible para Android en la App Store -Permite fotografiar tanto una carta de vinos, como una o más etiquetas botellas simultáneamente. -Tras hacer la foto, muestra la disponibilidad y precios de dichas botellas en el inventario de la empresa y ofrece la posibilidad de compra. -Multilingüe.

Tabla 1. Comparativa entre apps y herramientas OCR

Tras analizar las diferentes opciones se observa, que las herramientas disponibles, o están focalizadas en otros ámbitos (como el caso de Vivino), o sus aplicaciones se centran en modelos industriales de líneas que funcionan a gran velocidad (Cognex) o que trabajan con un alto volumen de documentos (nanonets), siendo estas unas medidas costosas para aplicaciones de menor escala.

Es por todo ello, además de la filosofía de la empresa de desarrollar ad-hoc en lugar de adaptaciones de terceros, lo que hace que el desarrollo de una aplicación dedicada, que consuma pocos recursos y no involucre dependencia de terceras partes, sea lo más deseable.

3.Análisis, Diseño, Implementación y Pruebas

Al igual que en la asignatura de “Ingeniería del software” como manual de referencia se ha usado “Ingeniería del Software 7ª edición. Ian Sommerville”. [8] En este punto se procede a explicar el desarrollo del análisis, diseño, implementación y la batería de pruebas.

Dado que, en las primeras fases del análisis, se llegó a la conclusión de que el proyecto estuviera formado por varios módulos funcionales independientes entre sí, se optó por el desarrollo que siguiese un modelo de ensamblaje de componentes.

3.1.Análisis

En este apartado se define el qué hay que hacer, para ello se ha desarrollado un documento de requisitos, así como un resumen de diversos aspectos que el producto final ha de cumplir.

3.1.1.Perspectiva del producto

Este sistema estará formado por tres partes principales, la aplicación web del operario, el servidor encargado de la conexión con la base de datos(BD) encargada de la persistencia y por último una. Dicho sistema estará organizado siguiendo la estructura front-end y back-end, conocida como cliente-servidor

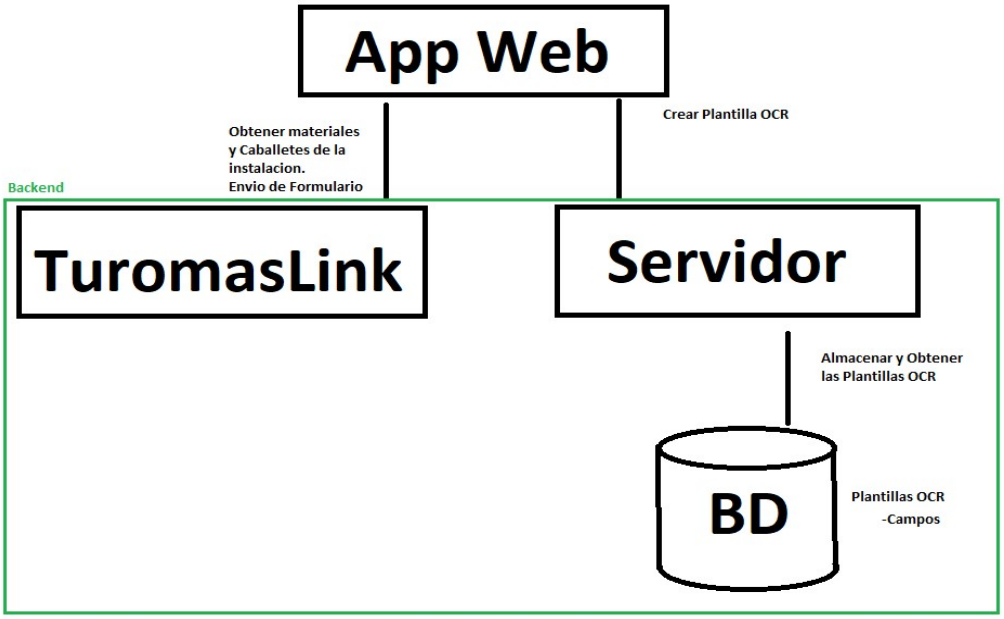

Figura 2. Estructura del programa

Desde la **App Web** el operario interactúa con el sistema, desde ella, el operario introduce en el sistema el nuevo inventario, obteniendo mediante las fotos la información sobre los paquetes de hojas de vidrio. Además, el operario puede modificar dicha información antes de enviarla.

El servidor es el único capaz de acceder y modificar la base de datos. El usuario accede a las plantillas almacenadas en la BD a través del servidor.

Turomas Link son un conjunto de endpoints de los que se obtiene información relativa a los materiales disponibles, así como la disposición de los caballetes de almacenamiento.

Dado que utiliza un sistema de datos centralizado, se usará una base de datos (BD) para el almacenamiento a la que accede el servidor. La App Web accede a estos datos mediante interfaces.

3.1.2. Funcionalidades del producto

La App Web ha de permitir la elección entre las plantillas existentes, después hacer fotos a las etiquetas de los paquetes, de dichas fotos se extraerá la información pertinente. Tras lo cual ha de ser posible editarla antes de ser enviada al resto de la infraestructura del sistema.

3.1.3. Restricciones

Un pilar fundamental de la aplicación es la cámara, es necesario un dispositivo que disponga de ella. Además, debido al modo de funcionamiento del OCR, una cámara de muy baja calidad, no permitirá extraer correctamente la información. Mientras que una cámara de muy alta calidad incrementa los tiempos de procesamiento hasta 30 o 40 veces más.

El dispositivo que se ha usado para el desarrollo de la aplicación es una tablet
Samsung Galaxy note 10.1 con una cámara de 8MP.

3.1.4. Características del usuario

Se asume que el operario que va a usar la App Web, conoce el proceso industrial de la fábrica y entiende la jerga y los tecnicismos relativos a dicha aplicación.

3.1.5.Requisitos

Para comenzar con el desarrollo de la solución, se han pensado en los requisitos que ha de cumplir la App Web siguiendo las directrices de la norma **IEEE 830-1998** [6]

REQUISITOS FUNCIONALES

RQ1. La aplicación ha de ser capaz de identificar etiquetas de diferentes proveedores.

RQ1.1. Tener un listado de diferentes plantillas para los distintos proveedores.

RQ2. El proyecto será una App Web que seguirá el modelo Single Page Application (SPA).

RQ2.1. Mediante una Web API ha de poder dar de alta entradas en el almacén.

RQ3. La aplicación ha de poder ser usada en diferentes dispositivos portátiles.

RQ4. La aplicación usará técnicas OCR para reconocer los diversos elementos de la etiqueta, necesarios en el formulario.

RQ4.1. Los datos a reconocer son: Proveedor, Material, Longitud 1 y 2, Nº Hojas del paquete, nº de Lote y el caballete donde se almacenará el paquete.

RQ4.2. Las longitudes se almacenan en mm.

RQ4.3. Aunque los datos se completan mediante técnicas OCR, el operario debe poder editar dichos datos en todo momento.

RQ5. La aplicación mostrará una plantilla por encima de la cámara, para indicar al operario a donde enfocar para realizar la foto.

RQ5.1. El operario debe poder repetir la foto las veces que desee.

RQ6. La aplicación estará ordenada siguiendo un modelo lineal compuesto de una o dos acciones por pantalla.

RQ6.1. El usuario puede volver sobre sus pasos en todo momento.

RQ6.2. La última pantalla/ventana de un nuevo registro será un listado resumen con los datos a introducir, que el operario podrá editar, y tendrá que confirmar para poder enviarla.

RQ7. La app debe poder mostrar errores al dar de alta las entradas.

RQ8. La app ha de ser capaz de recuperar las plantillas mediante la Web API.

RQ8.1. También ha de poder recuperar los materiales y los caballetes disponibles.

REQUISITOS NO FUNCIONALES

RQ9. La app debe de ser capaz de comunicarse con el resto de sistemas de la fábrica.

RQ10. La app debe de poder ser accedida desde cualquier dispositivo con acceso a la red interna.

3.2.Diseño

El diseño se ha enfocado en tres bloques: el diseño de la BD, el front-end donde se desarrolla la interfaz de usuario (IU) de la App Web y el back-end el encargado de hacer de intermediario entre el front-end y la BD.

3.2.1.Diseño de la Base de Datos

La información sobre la que se necesita persistencia es muy simple, únicamente se almacenan las plantillas, las cuales están compuestas por los atributos mencionados en el requisito RQ4.1. Además de los campos, entidades que representan las áreas a leer por el OCR

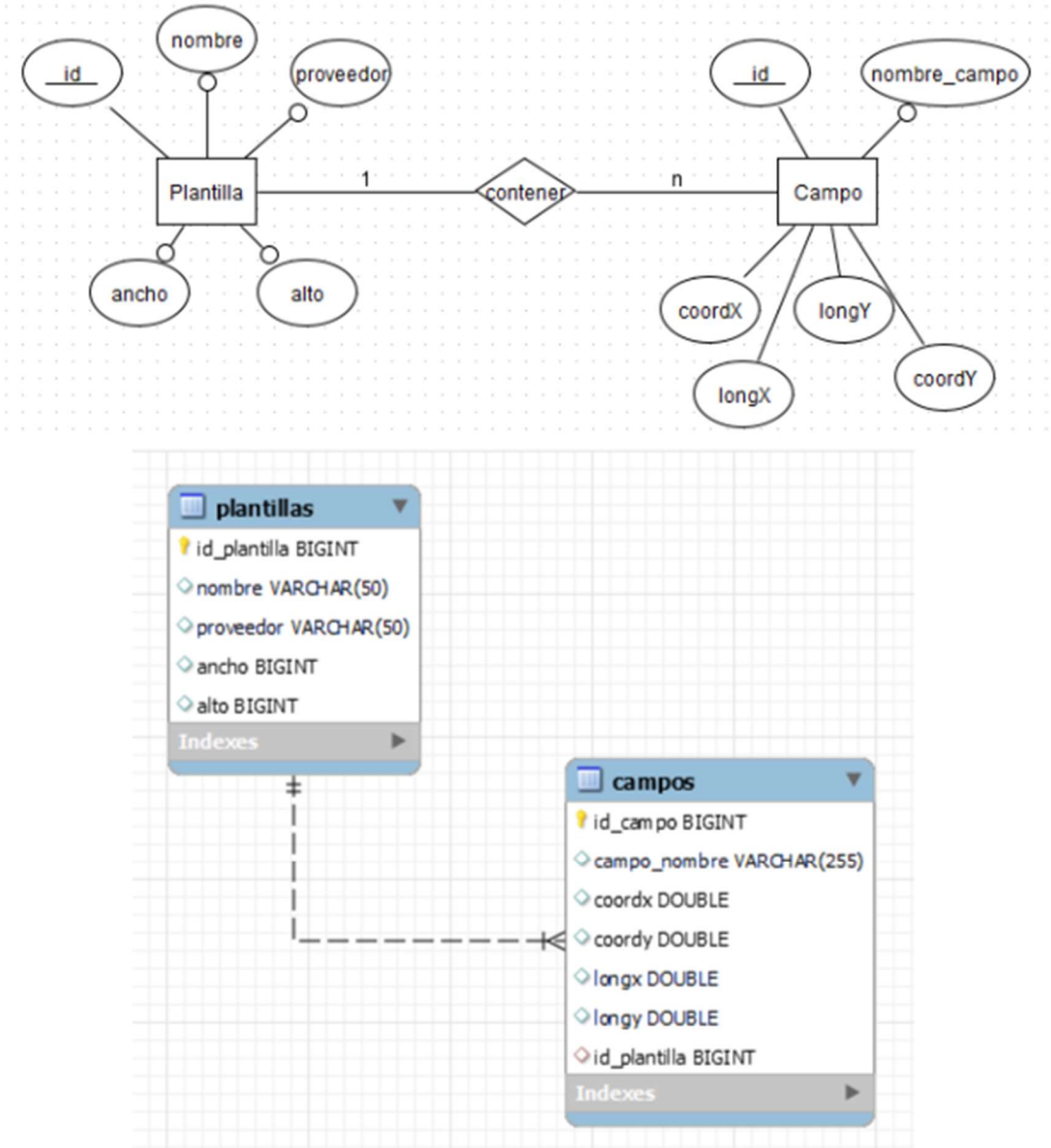

Figura 3. Diagrama E-R y diseño físico de la BD

Además, se ha generado el diseño lógico de la BD, el cual se puede apreciar en la figura 4

```
/*Entidad Plantilla*/
Plantilla       (id: dom_id,
                 nombre: dom_nombre,
                 proveedor: dom_proveedor,
                 ancho: dom_ancho,
                 alto: dom_alto)
        Clave Primaria { id }
        Valor No Nulo { nombre, proveedor,
                        ancho, alto}

/*Entidad Campo*/
Plantilla       (id_campo: dom_id,
                 nombre_campo: dom_nombre,
                 coordX: dom_coordenada,
                 coordY: dom_coordenada,
                 longX: dom_long,
                 longY: dom_long)
        Clave Primaria { id_campo }
        Valor No Nulo { nombre_campo}

/*Relación Contener*/
Contener        (id: dom_id, id_campo: dom_id)
        Clave Primaria { id }
        Valor No Nulo { id_campo }
        Clave Ajena { id } hace referencia a Plantilla
        Clave Ajena { id_campo } hace referencia a Campo
```

Figura 4. Diseño lógico

Por último, para la realización de pruebas, se ha cargado en el sistema de persistencia varias plantillas, con sus respectivos campos (ver figura 5).

id_plantilla	nombre	proveedor	ancho	alto
1	Sisecam Climax	Syscam	8	14
2	Sisecam Ancha	Syscam	12	4
3	G.G. Estandar	Guardian Glass	10	10

id_campo	campo_nombre	coordx	coordy	longx	longy	id_plantilla
1	material	0	0	4	1.5	1
2	dimensiones	3.5	4	5	3	1
3	Num.Hojas	2	6	1.5	1.5	1
4	material	2	2	4	1.5	2
5	dimensiones	6	2	1	3	2
6	Num.Hojas	2	3.5	2.5	1.5	2
7	material	2	2	4	1.5	3
8	dimensiones	6	2	1	3	3
9	Num.Hojas	2	3.5	2.5	1.5	3

Figura 5. Datos de prueba.

3.2.2.Diseño del Front-End

Para el diseño de la interfaz de usuario (IU), se han desarrollado primeramente una serie de prototipos con Balsamiq Wireframe, a modo de guía para el desarrollo de ésta.

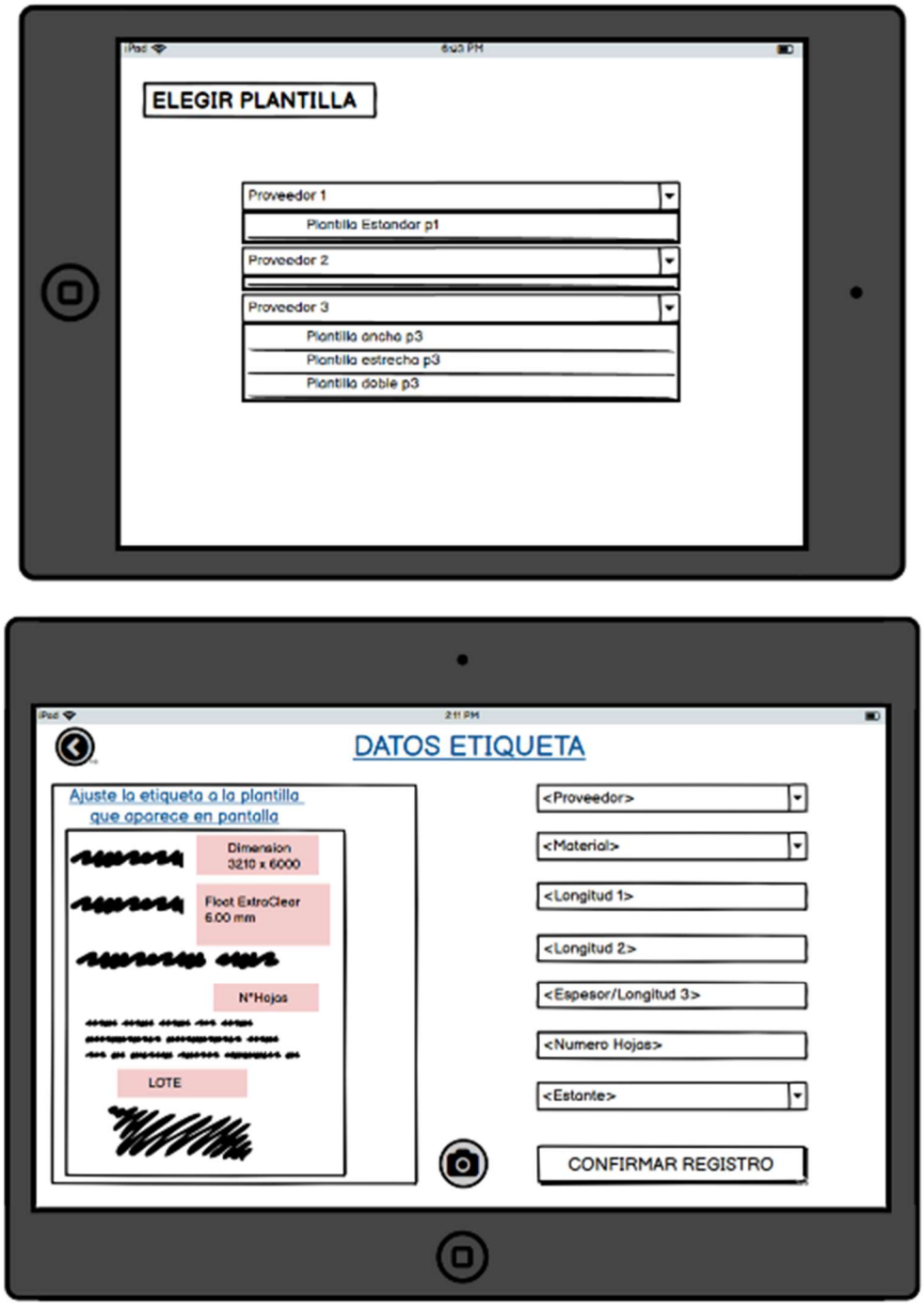

Figura 6. Prototipos

Los diagramas que aparecen en la figura 6 son los diagramas que son más fieles al productor final. Esto es debido a que inicialmente el funcionamiento incluía reconocer el proveedor mediante otra imagen y procesarla por OCR.
Este paso fue finalmente descartado, ya que incrementa considerablemente el tiempo de cada interacción para crear un Formulario nuevo. Cuando esta función se puede realizar fácilmente con un listado de botones, uno por proveedor, los cuales no superan la decena.

Tras tener un boceto, se procede a desarrollar una serie de diagramas de clases, de secuencia y casos de uso. Para ello se ha usado Modelio 4.0, una herramienta aprendida en la asignatura de Ingeniería del Software que permite diseñar diagramas UML [1].

DIAGRAMA DE CLASES

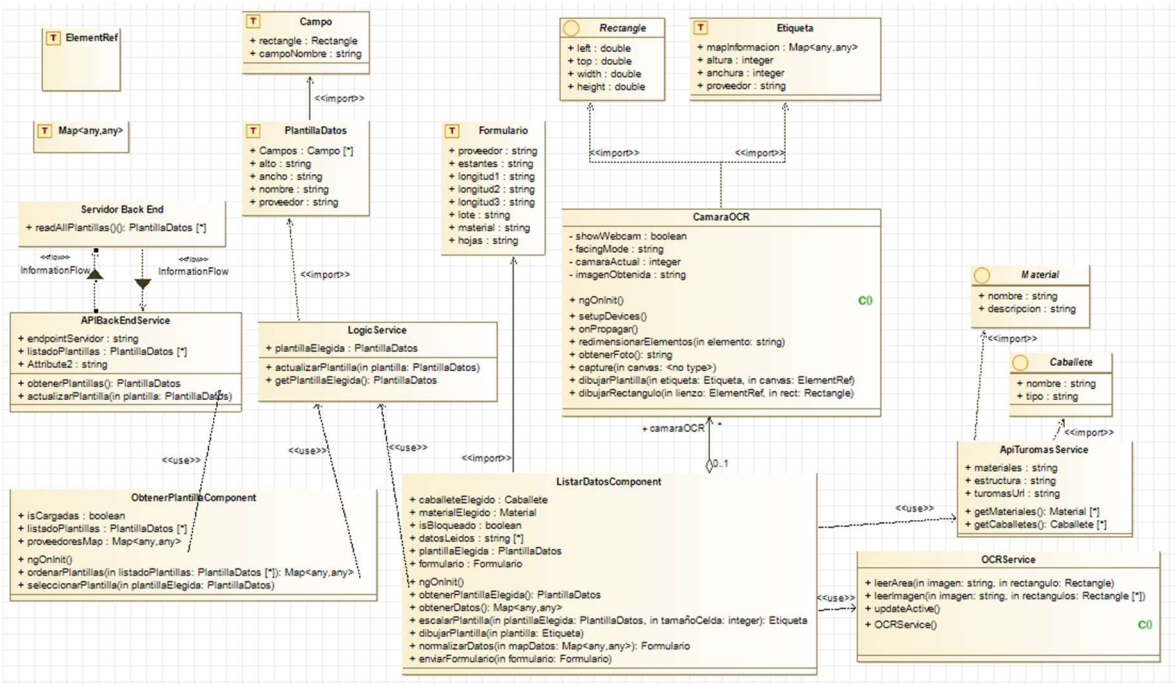

Figura 7. Diagrama de clases del front-end

En este caso, los elementos principales que componen el front-end, son tres:

CamaraOCR, encargado de todo lo relacionado con mostrar la cámara por pantalla y dibujar sobre dicha cámara la plantilla pertinente, así como de obtener la imagen que será procesada.

ObtenerPlantillaComponent, este componente realiza la tarea de obtener las plantillas disponibles y mostrarlas al usuario agrupadas por proveedor.

ListarDatosComponent, este componente hace uso de un componente CamaraOCR para obtener la foto sobre la que extraen los datos para crear el formulario necesario.

Entre las clases de datos diseñadas, es pertinente señalar la diferencia entre PlantillaDatos, Etiqueta y Formulario:

PlantillaDatos, contiene la información referente a la serie de cuadrados a dibujar sobre la cámara, a modo de guía para el operario. Por lo que este objeto contiene las coordenadas y tamaños de los diferentes campos que conforman la plantilla, así como las dimensiones de dichas etiquetas (valores alto y ancho). Estos dos últimos valores son pertinentes para aprovechar mejor el espacio disponible en la cámara, ya que unas etiquetas son más anchas y otras más altas.

Etiqueta, es una clase cuyos objetos contienen una serie de pares de información siguiendo la estructura
<campo leído, [Area a leer, valor leído por el OCR]>.

Formulario, esta clase es la que contiene la información relevante de la etiqueta del paquete, procesada, normalizada y preparada para ser enviada al ecosistema de la fábrica. Además de contener las decisiones del usuario en cuanto al tipo de material y el caballete en el que va a ser almacenado el paquete correspondiente.

DIAGRAMA DE SECUENCIA

El diagrama representado en la figura 8 refleja el proceso de captura de una foto para su reconocimiento OCR y la creación de un Formulario

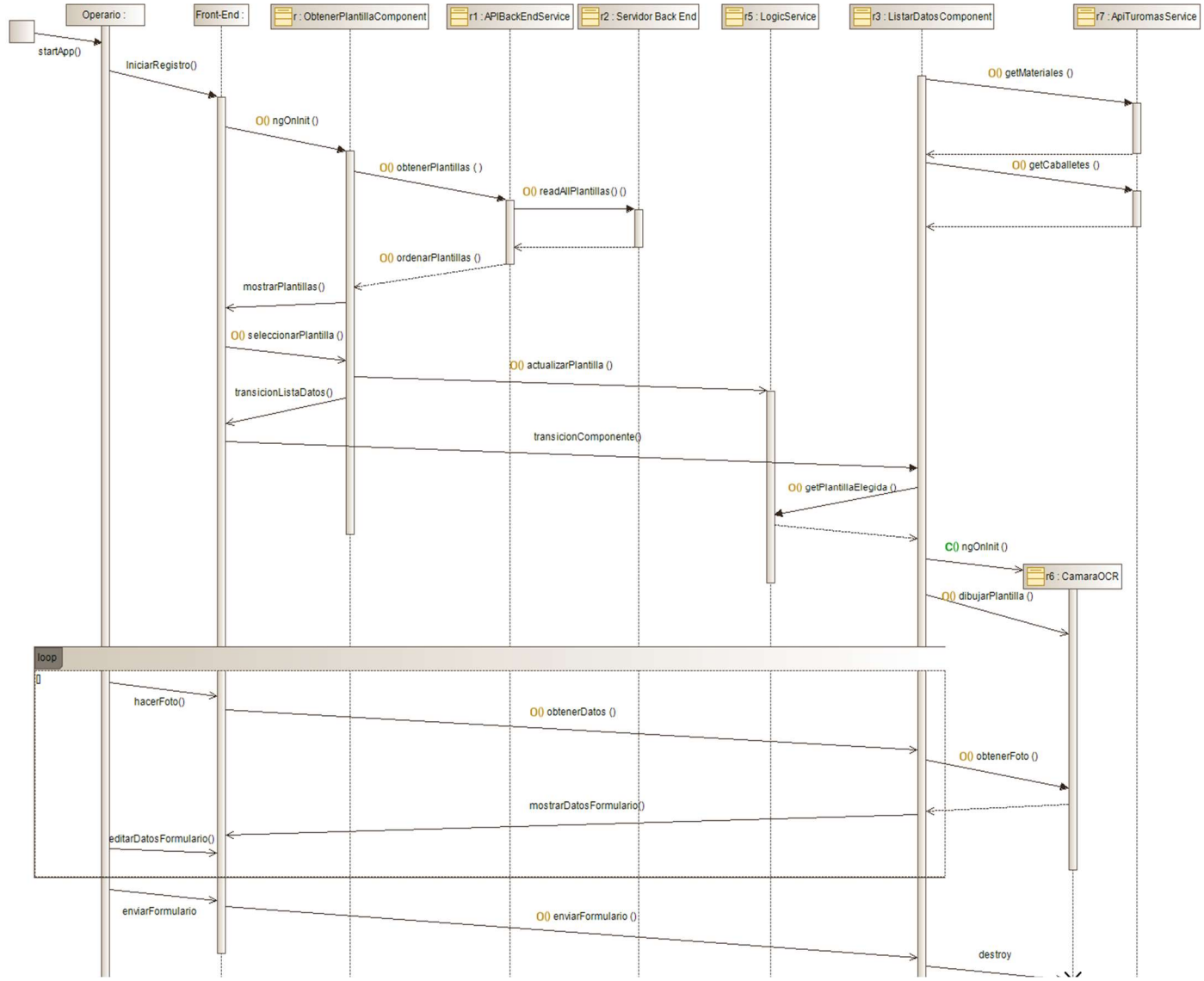

Figura 8. Diagrama de Secuencia. Insertar nuevo Formulario.

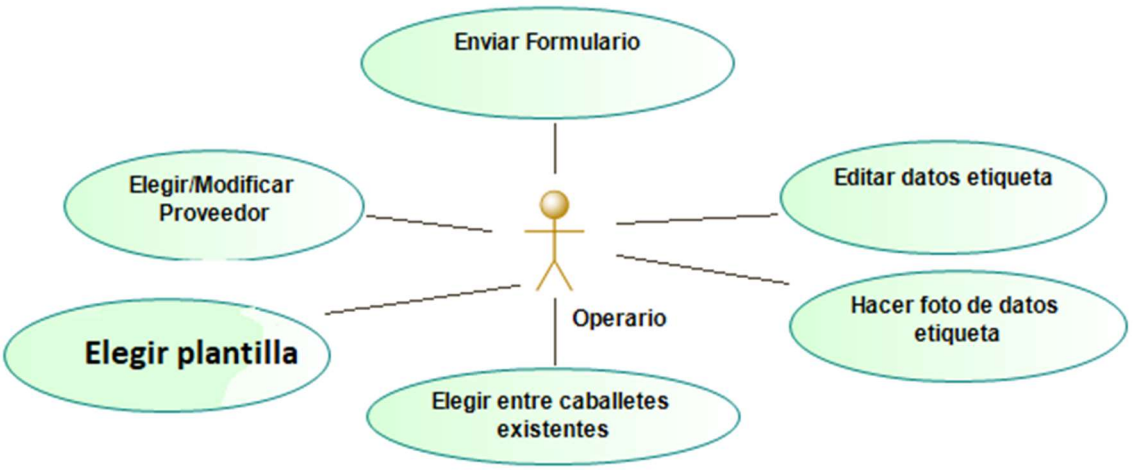

Figura 9. Diagrama de casos de usos

En el diagrama de casos de uso de la figura 9 se observan las diferentes acciones que el operario realiza

3.2.3.Diseño del Back-End

Para el diseño del back-end, se ha usado Visual Paradigm 17.0(VP-17). Esto se debe a que VP-17 se puede integrar con Apache Netbeans, lo que permite integrar el modelo UML diseñado automáticamente en Java, lo cual facilita la implementación.

El back-end se encargará de generar un endpoint API para permitir al front-end solicitar las plantillas almacenadas en el sistema de persistencia. A su vez, es el encargado y único elemento que puede acceder a la BD para acceder/modificar dichos datos (ver figura 10).

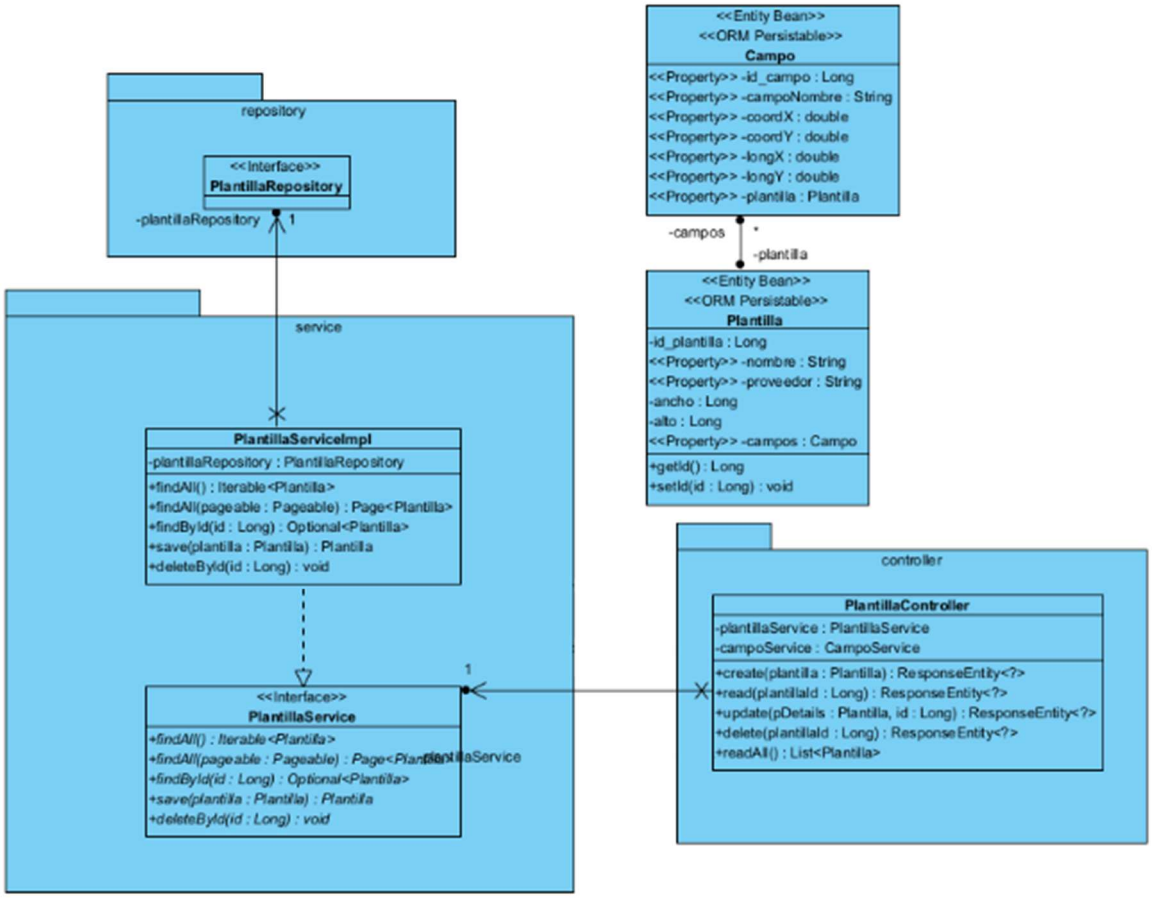

Figura 10. Diagrama de clases del back-end

3.3.Implementación

En este apartado, se explican las tecnologías usadas, tras lo cual, debido a que es un pilar central del trabajo, se explica el funcionamiento del OCR por separado.

3.3.1.Tecnologías

Las tecnologías usadas para la implementación han sido las siguientes para cada una de sus partes:

- Front-End
- Back-End
- Base de Datos

Parte de la implementación	Tecnologías usadas
Front-End	- ID: Visual Studio Code -Framework: Angular Typescript v4.3.2 -Motor OCR: Tesseract OCR
Back-End	-ID: Apache NetBeans 12.6 integrado con Spring Boot 2.6.4 -Lenguaje: Java con JDK 11 -ORM: Spring Data JPA (hace uso de Spring Data y Hibernate)
Base de Datos	-ID: MySQL Workbench -SGBD: MySQL

Tabla 2. Tecnologías usadas

3.3.2.Implementación del Optical Character Recognition

Para la implementación de las técnicas OCR se ha utilizado el motor Tesseract OCR. Dicho motor está escrito en C y es de código libre, liberado bajo la licencia Apache v2.0 y su desarrollo está financiado por Google.

En concreto se ha usado "Tesseract.js", la cual es una librería que soporta alrededor de 100 idiomas. Los motivos para usar esta librería son el hecho de que es código libre, cuenta con una amplia documentación y un gran número de guías y ejemplos.[13]

Las técnicas OCR se basan en un procedimiento de varios pasos: [17] [10]

- Primero se transforma la imagen a una imagen binaria, es decir transformar toda la imagen a dos valores de colores (usualmente blanco y negro, pero puede ser otro dúo)
- Identificación de las zonas oscuras (negras) y zonas iluminadas (blancas), las oscuras son identificadas como glifos que necesitan ser identificados y las iluminadas como fondo de imagen
- Las áreas oscuras son procesadas para encontrar letras alfabéticas o dígitos numéricos. Normalmente, este paso se realiza enfocándose en un único carácter cada vez. Se suele usar uno de estos dos algoritmos, reconocimiento de patrones o extracción de características.
- El reconocimiento de patrones requiere que el programa OCR tenga a su disposición una cantidad mínima de ejemplos de texto en varias fuentes y formatos para comparar y reconocer los caracteres escaneados
- La extracción de características divide o descompone los glifos en características (líneas, circuitos cerrados) y las usa para encontrar la mejor coincidencia o el glifo más cercano entre los glifos almacenados.

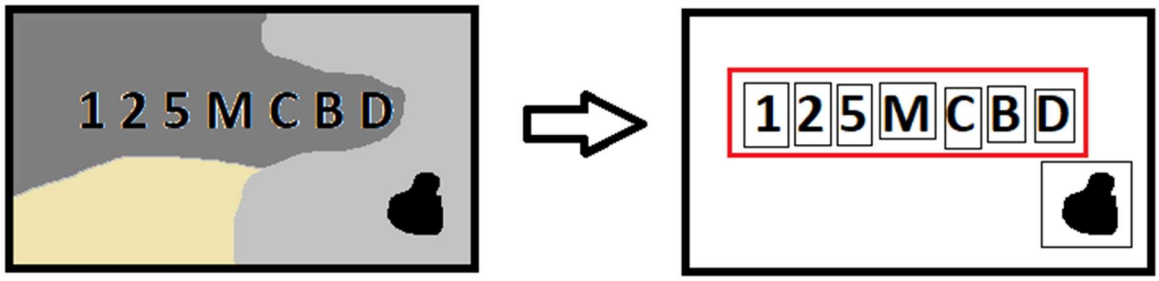

Figura 11. Ejemplo teórico del proceso OCR

Tomando como referencia la figura 11, se ilustra la transformación de la imagen a dos colores. Tras lo cual, las zonas oscuras son representadas con rectángulos negros, mientras que el área indicada por el usuario es el rectángulo rojo. De esta manera, aunque haya más zonas de interés para el OCR, únicamente se reconocen aquellas indicadas por el usuario.

Para comenzar se ha generado un service, un objeto singleton diseñado con el objetivo de controlar la información (obtención, almacenamiento, actualización y compartición), llamado ocr.service.ts.

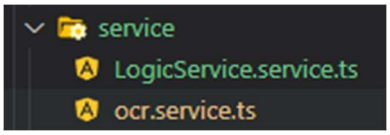

Figura 12. Service del front-end

Este servicio, se encarga de crear objetos "worker" procedentes de una clase proporcionada por la librería tesseract, será el encargado de realizar todos los procedimientos de transformar los campos indicados de la imagen, en texto (ver figura 13).

```
await worker.load();
await worker.loadLanguage('eng');
await worker.initialize('eng');
```

Figura 13. Creación de un OCR worker

Para ello se utiliza el método "readImage", ver figura 15, el cual debe recibir por parámetros la imagen deseada, así como las áreas a leer, acompañadas del filtro de lectura, es decir los carácteres a buscar. Si el filtro está vacío, reconoce todos los carácteres del alfabeto indicado al worker.

Dichas áreas se encuentran dentro de un Map, cuyas keys son los nombres de los campos { Dimensiones, Hojas, Lote} y como valor una dupla de dicha área a leer y el filtro de caracteres a aplicar [rectangle, filtro]. (ver figura 14).

```
("Dimensiones",[areasLectura[1],'0123456789xX']);
("Hojas",[areasLectura[2],'0123456789']);
("Lote", [areasLectura[3],'']);
```

Figura 14. Map para aplicar OCR

```
async readImage(imagen:any, rectangulos:Map<any,any>) {
  this.resMap = new Map();
  try{
      for (let [key, value] of rectangulos) {
        await this.LeerArea(imagen, value[0], value[1]).then(res => (this.resMap.set(key, res)));
      }
    this.updateActive();
  }catch{}
}
```

Figura 15. Método para aplicar OCR sobre una imagen

Siendo el resultante un Map, conformado por la estructura <nombreCampo, valorLeído>, almacenado en el propio service, a la espera de que sea solicitado por la aplicación.

A su vez "readImage" hace uso de "readArea", del cual cabe destacar que primero construye el worker, al cual hay que indicarle un lenguaje que cargar e iniciarlo (se le pueden pasar varios idiomas a la vez). En este caso se ha seleccionado el idioma inglés debido a que varios de los proveedores, independientemente de si son españoles, italianos o de otra nacionalidad, fabrican las etiquetas en inglés.

Se genera un worker para leer cada área, esto es debido a que antes de realizar la lectura, hay que configurar el worker. Para ello, cada par del Map de la Etiqueta contiene la lista de caracteres a identificar, que actúan como configuración. Por ejemplo, el campo de dimensiones está caracterizado por disponer de dos medidas separadas por un x o una X. Por lo que únicamente interesa obtener números y equis minúsculas o mayúsculas. Esto, a su vez, actúa como un filtro de errores, ya que muchas veces las etiquetas disponen de un espacio justo y limitado, con datos muy juntos. Por lo que filtrar los caracteres leídos evita que se estropee la transcripción con la inclusión de datos cercanos.

```
worker.setParameters({tessedit_char_whitelist: filtro});
const { data: { text } } = await worker.recognize(imagen, {rectangle});
```

Figura 16. Configuración del worker y aplicación del OCR

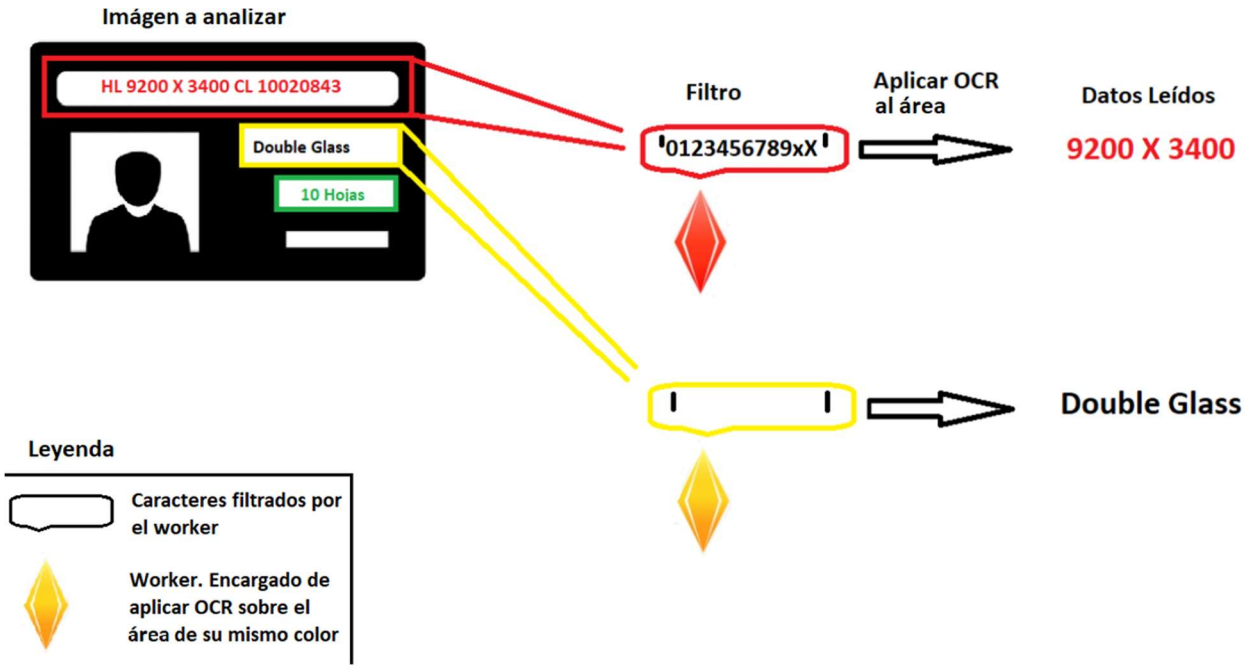

Figura 17. Esquema funcionamiento OCR.

3.3.3.Implementación del Front-End

La estructura del proyecto para la single-page application, que se ha establecido como modelo para el front-end es la siguiente:

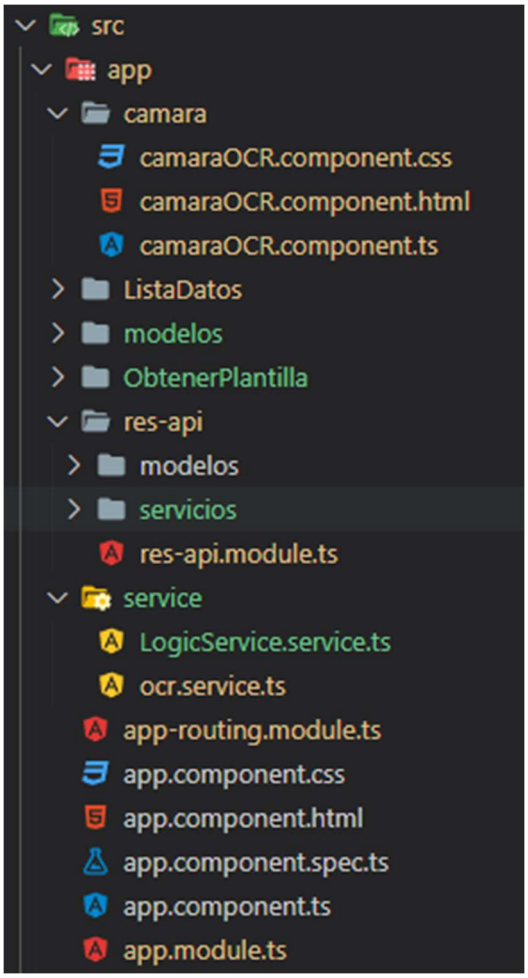

Figura 18. Estructura del front-end

La aplicación se divide en varias secciones, siguiendo el patrón Modelo Vista Controlador. Por un lado, se encuentra la sección res-api, que gestiona la conexión tanto con el back-end, como con los endpoints de la infraestructura de TuromasLink.

En segundo lugar, los componentes, encargados de la vista y las propiedades que van a usar dichas vistas.

Para finalizar, están los modelos y los servicios, donde se contienen las representaciones de los datos. Mientras que los servicios, como se ha comentado anteriormente, cumplirían el papel de controlador, al actuar de intermediario entre los componentes y los modelos.

Res-API

Las resAPI son servicios encargados de las conexiones con diversos endpoints. Se dispone de dos: apiServidor y apiTuromas.

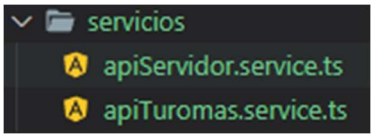

Figura 19. Servicios resAPI creados.

El primer servicio, apiServidor, se comunica con el back-end para solicitar el listado de plantillas. Para ello, se hace una llamada a getListadoPlantillas(), el cual nos retorna un Observable que transmitirá dicho listado una vez sea obtenido.

```
public getListadoPlantillas(): Observable<any> {
  return this.http.get(ApiServidorService.API_ENDPOINT_BACKEND);
}
```

Figura 20. Llamada HTTP al backend.

El segundo servicio se encarga de la conexión con los endpoint ajenos a este proyecto, que se encuentran en la infraestructura de Turomas. Estos son dos, uno contiene el listado de materiales disponibles y en el otro se encuentra la estructura del almacén de la instalación en la que la aplicación está desplegada.

Modelos y servicios

Los modelos contienen la representación de los datos que maneja la aplicación. Son tres: PlantillaDatos, Etiqueta y Formulario.

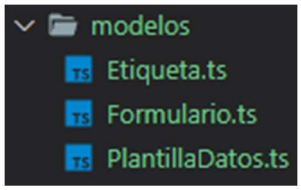

Figura 21. Modelos de datos

Para entender el porqué de estas 3 entidades, es necesario ver cómo se crean las plantillas que más tarde se muestran sobre la cámara a modo de guía. En la figura 22, se muestra el proceso, en primer lugar, se miden los lados de la Etiqueta que se quiere añadir a la BD. Tras esto, se procede a la división de la etiqueta en celdas cuadradas de 1cm de lado.

A continuación, se recogen las coordenadas de la esquina superior izquierda de cada área de interés, así como las longitudes de dichas áreas. Éstas son las correspondientes a los datos de: las dimensiones de sus hojas, el número de hojas y el número de lote.
Las áreas se modelan usando un objeto que usa la librería tesseract, los Rectangle que están formado por valores doubles {left, top, width, height}

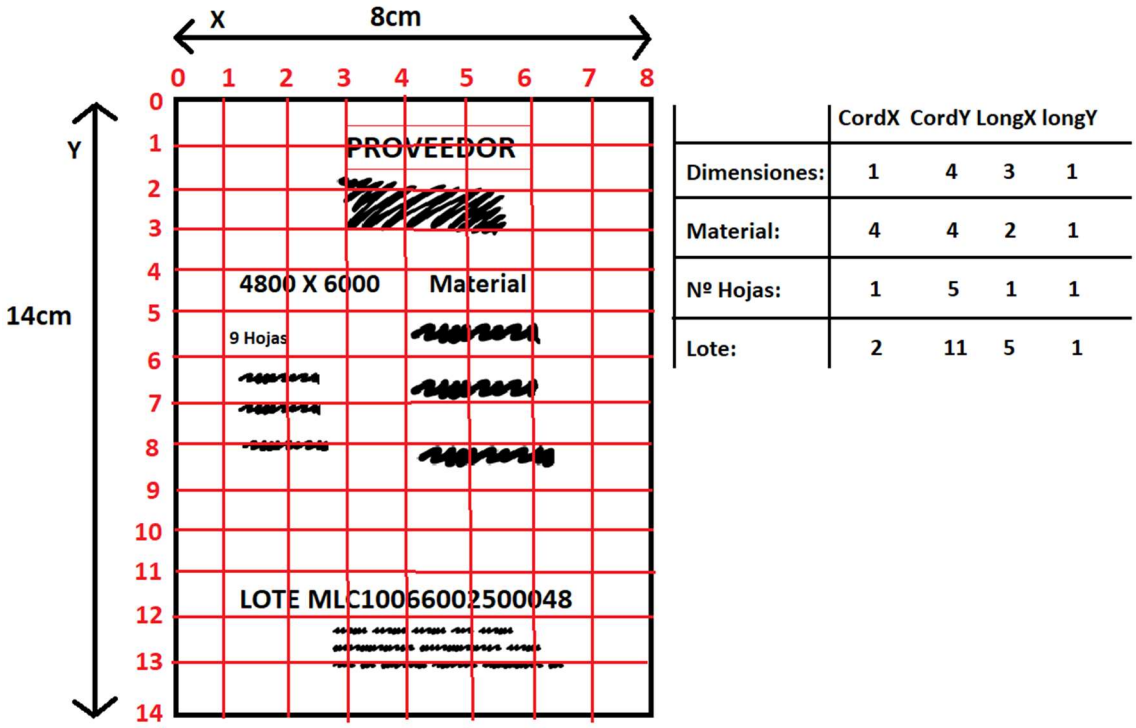

Figura 22. Obtención de coordenadas de una plantilla.

PlantillaDatos almacena las áreas del modelo de etiqueta, las coordenadas son relativas a las medidas de dicha etiqueta. PlantillaDatos guarda el nombre de dicha plantilla, el proveedor, la altura, la anchura y un array de Rectangles.

La Etiqueta es el objeto usado para mostrar la Plantilla por encima de la cámara. Primero es necesario escalar las medidas de la PlantillaDatos obtenida. Para ello, se usan las dimensiones del dispositivo de video. Para ello, se calcula el tamaño de las celdas en las que se divide la plantilla, esto se hace en el componente ListaDatos.

Para este cálculo se busca el lado más corto de la plantilla, del tamaño de la celda se calcula dividiendo la resolución de la cámara entre el lado mayor de la plantilla.

Tomando como ejemplo la figura 23, si el ancho es de 8 cm y el alto es 14 cm, se elige el lado de la altura, y se dividen los píxeles de altura por los centímetros de la altura de la plantilla. El valor resultante permite ajustar la plantilla al mayor tamaño posible por la resolución (Rectángulo verde). Si por el contrario se elige el lado contrario y se usa el valor resultante como tamaño de celda, la plantilla queda sobredimensionada y no cabe entera en la cámara (Rectángulo rojo).

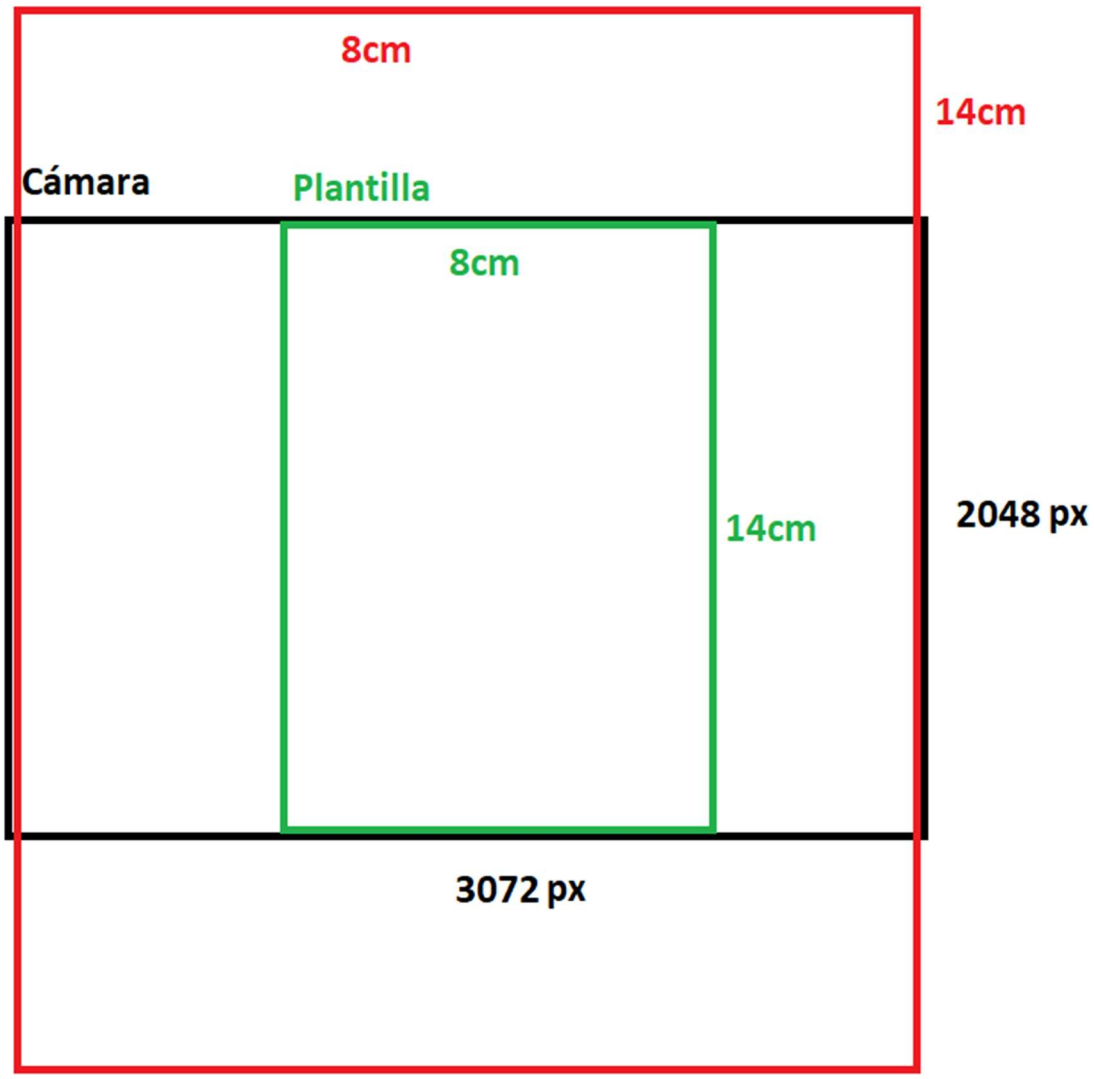

Figura 23. Redimensionado de plantillas a la cámara.

Finalmente, una vez redimensionada la Etiqueta, se guardan los valores de proveedor, las dimensiones de la Plantilla y por último un Map que por key tiene el nombre del campo que se lee {Dimensiones | Hojas | Lote} y por valor una dupla con el área a leer y el filtro que se aplica en el OCR.

Formulario es el objeto que recopila la información leída por el OCR.
Este objeto guarda únicamente el proveedor, que recoge al obtener la PlantillaDatos elegida desde el service, y los valores leídos. Además, guarda también varios valores que el operario introduce manualmente, el grosor de las hojas de los paquetes, el caballete de almacenamiento de destino y el material.

Los servicios usados están agrupados en dos grupos, el primero encargado de las conexiones externas que ya se ha explicado en el apartado res-API de la implementación y el segundo encargado de la información interna del front-end. Este último grupo está compuesto por dos servicios, uno encargado del OCR ya explicado en la implementación del OCR y el LogicService, ver figura 24, encargado de transmitir la plantilla elegida, de un componente a otro de la aplicación. Para ello ambos componentes referencian a este servicio en el constructor y actualizan la plantilla elegida o la obtienen de dicho servicio. Para este proceso se dispone de un observable al que cuando el componente ListaDatos se suscriba, obtendrá en ese momento la PlantillaDatos elegida, que es proporcionada por el componente ObtenerPlantilla.

```
export class LogicServiceService {
  private plantillaElegida: PlantillaDatos = new PlantillaDatos("","",0,0,[]);
  private approvalStageMessage = new BehaviorSubject(this.plantillaElegida);
  currentApprovalStageMessage = this.approvalStageMessage.asObservable();

  updateSelectedPlantilla(message: PlantillaDatos) {
    this.approvalStageMessage.next(message);
  }
```

Figura 24. Servicio de lógica

Componentes

La aplicación sigue el modelo Single Page Application (SPA) en el cual los recursos se cargan una sola vez o dinámicamente según sean necesarios. Los pasos seguidos por la aplicación se dividen en dos componentes principales más uno auxiliar:

a) ObtenerPlantilla: En el que el operario selecciona una plantilla entre las disponibles.
b) ListaDatos: Donde el operario saca una foto y se muestran los datos leídos por OCR, los cuales puede modificar el operario.
c) CamaraOCR: Encargado de la vista de la cámara y de mostrar sobre ella la plantilla de apoyo al operario.

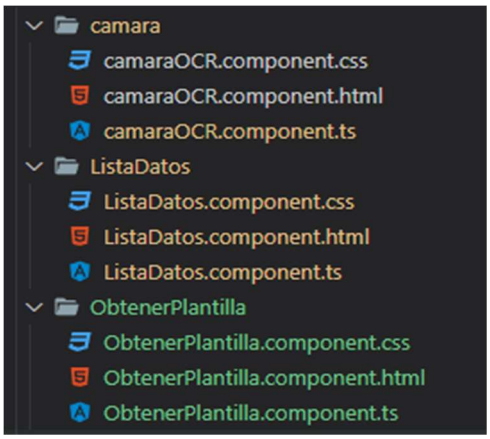

Figura 25. Componentes del front-end

Los componentes están formados por 3 elementos: un archivo html que será la plantilla, donde se visualiza la IU, un archivo de lógica (component.ts), donde se incluye una clase y las propiedades y métodos que usará la vista, un archivo para el CSS que facilita el modelado de la vista mediante los estilos de los elementos HTML.

A lo largo del desarrollo de los componentes, se usan ciertos elementos html obtenidos de las librerías @angular/material/<elemento>. [3] Estos elementos se incluyen en el documento "app.module.ts" (módulo central de la aplicación). Como se muestra en la figura 26.

```
import { MatSliderModule } from '@angular/material/slider';
import { MatExpansionModule } from '@angular/material/expansion';
import { MatListModule } from '@angular/material/list';
import {MatNativeDateModule} from '@angular/material/core';
import {MatSelectModule} from '@angular/material/select';
```

Figura 26. Importación de elementos html adicionales

a) **ObtenerPlantilla**, realiza unas acciones muy lineales. Primero, obtiene las plantillas disponibles en el back-end mediante la suscripción al servicio apiServidor. Esto se ejecuta dentro del método ngOnInit(), ver figura 27, un método que se invoca inmediatamente después de recibir las propiedades de entrada y antes de que cualquier elemento de la vista o contenidos hijos sean creados. Se invoca una sola vez cuando la directiva es instanciada.

```
ngOnInit() {
  /* Obtenemos del sistema de persistencia las plantillas*/
  this.apiServidor.getListadoPlantillas().subscribe(
    plantillas => {
      this.listadoPlantillas = plantillas;
      this.ordenarPlantillas();
    }
  );
```

Figura 27. Suscripción al servicio apiServidor

Tras obtener el listado de plantillas, las agrupa en un map usando como key el proveedor y cómo value todas las plantillas que lo comparten como proveedor. Esto se hace para poder mostrar en la vista las plantillas como una lista anidada.

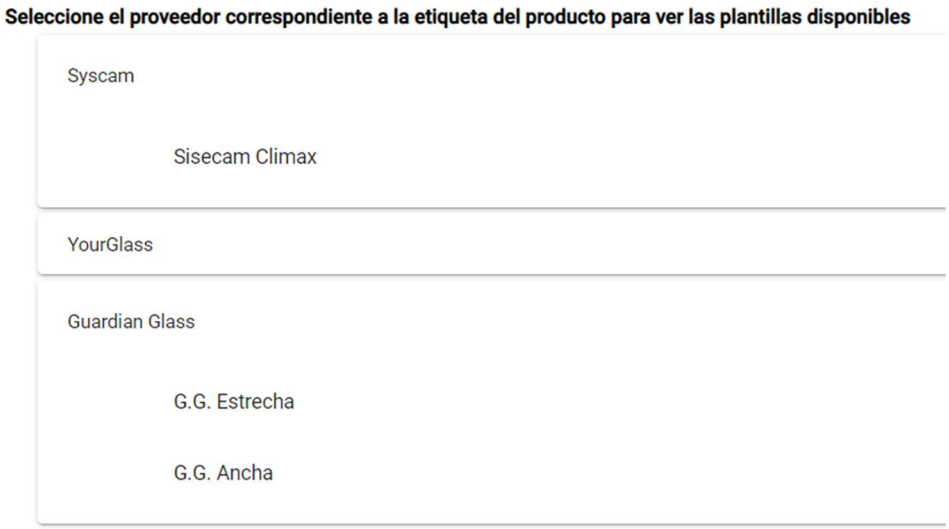

Figura 28. Ejemplo listado de plantillas disponibles.

Cuando el operario pulsa cualquiera de las plantillas disponibles, mediante el servicio LogicService, se almacenan los datos de la plantilla elegida en el service para que, al instanciar el siguiente componente, puedan obtenerse.

b) **CamaraOCR** es un componente auxiliar, donde se gestiona el streaming de la cámara, la sustitución de éste por la imagen obtenida y la eliminación de la imagen si se desea repetir. Además, en este componente se gestiona el dibujado de la plantilla y de sus campos por encima de estos canvas. La idea general se ve en la figura 29.

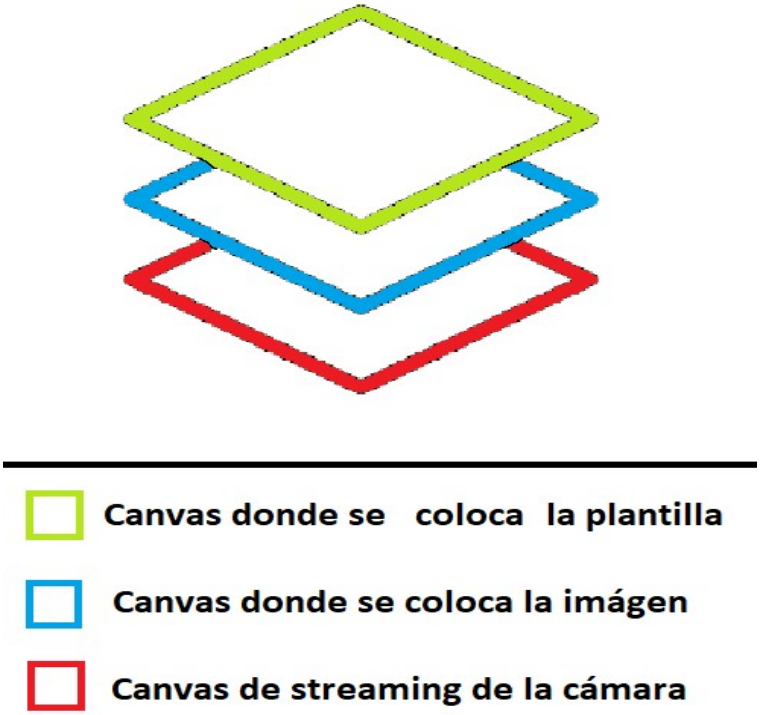

Figura 29. Disposición de los canvas de CamaraOCR

Cuando se quiere hacer la foto, el canvas azul, donde se muestra la futura imagen, se desactiva, mientras que el canvas rojo va mostrando lo que capta la cámara en streaming.

Por encima de ésta, se encuentra un canvas semitransparente, en el que se dibuja la silueta de la plantilla y sus respectivos campos. Este canvas no se desactiva en ningún momento, ya que al obtener la foto, se ha considerado de interés que el operario vea dónde ha enfocado dicha plantilla.

Para mostrar la foto, se desactiva el canvas rojo y en su lugar se activa el azul, donde se muestra la imagen obtenida. Esto se hace a través de una variable booleana que modifican los diferentes métodos del componente según corresponda.

Al iniciarse el componente, se genera la estructura de la vista, creando los elementos mencionados anteriormente. Con el objetivo de hacer dicha vista responsive para diferentes dispositivos, al crearse, no disponen de dimensiones establecidas. Se ha de esperar a que el dispositivo de vídeo esté configurado y listo para transmitir. Cuando esto ocurre, se obtiene la resolución del dispositivo de vídeo, y se escalan todos los elementos pertinentes en función de dichas medidas.

Además, también se dibuja la plantilla, esto ocurre también una vez el componente padre, que engloba a este componente, obtiene una PlantillaDatos y la propia cámara está ya lista, en ese momento se invoca el método DibujarPlantilla, que recursivamente llama a DibujarMarco y dibuja los correspondientes campos de la plantilla. (ver figura 30).

```
public DibujarPlantilla(etiqueta: Etiqueta, canvas: ElementRef<any>): void {
  this.DibujarMarco(this.canvasLienzo, { left:0, top:0, width:etiqueta.anchura*this.CELDA, height:etiqueta.altura*this.CELDA});
  etiqueta.mapCampos.forEach((value) => {
    this.drawRectangle(canvas, value[0]);
  })
}
public DibujarMarco(lienzo: ElementRef, rectangulo: Rectangle): void{
  let ioctx = lienzo.nativeElement.getContext('2d');
  ioctx.fillStyle = 'rgb(0, 255, 0, 0.2)';
  ioctx.fillRect(rectangulo.left + this.DESPLAZAMIENTO, rectangulo.top, rectangulo.width, rectangulo.height);
}
```

Figura 30. Métodos de dibujo de plantillas.

c) **ListaDatos** es el principal componente de la aplicación, ya que sobre él se despliega el componente **CamaraOCR** y es sobre el que se realizan las acciones de obtención de una imagen para aplicar OCR y la revisión, edición y confirmación del Formulario final a ser enviado.

Al crear el componente, se suscribe al servicio de lógica para obtener la plantilla elegida, que está en formato PlantillaDatos. Para dibujarla es necesario primero que el dispositivo cámara esté listo, cuando esto ocurre, salta un evento que llama a los métodos encargados de transformar la PlantillaDatos en un objeto tipo Etiqueta, por lo que primero se transforma en una etiqueta, y luego se escala a la resolución de la cámara, mediante los métodos mostrados en la figura 31.

```
public normalizarPlantilla(pDatos: PlantillaDatos):Etiqueta{
  let camposNormalizados: Rectangle[] = [];
  for (var i in pDatos.campos){
    let rectangleCampo: Rectangle = { left: pDatos.campos[i].left, top: pDatos.campos[i].top,
      width: pDatos.campos[i].width, height: pDatos.campos[i].height};
    camposNormalizados.push(rectangleCampo);
  }
  let etiquetaNueva = new Etiqueta(pDatos.proveedor,pDatos.ancho,pDatos.alto,camposNormalizados);
  return etiquetaNueva;
}

public EscalarPlantilla(plantillaB: Etiqueta, escalado: number):Etiqueta{
  for (const key of plantillaB.mapCampos.keys()) {
    let campo = plantillaB.mapCampos.get(key);
    campo[0] = {left: campo[0].left *escalado, top: campo[0].top *escalado,
      width: campo[0].width *escalado, height: campo[0].height *escalado };
```

Figura 31. Métodos para normalizar y escalar la plantilla

Cuando se llega a dicho componente se encuentra la IU en la situación inicial de la figura 32, con la plantilla ya dibujada sobre la cámara, para guiar al operario en la obtención de la imagen.

Figura 32. Estado inicial componente ListaDatos.

El primer paso, siguiendo el diagrama de secuencia, es que el operario (ajustando la plantilla a los campos correspondientes) haga una foto de la etiqueta a escanear. Mientras se aplica el OCR en la imagen obtenida, se activa sobre toda la pantalla un canvas, semitransparente de color amarillo, que impide que el usuario modifique datos o cancele la foto y la repita en mitad del proceso de OCR, lo que puede generar fallos y/o bloqueos del programa.

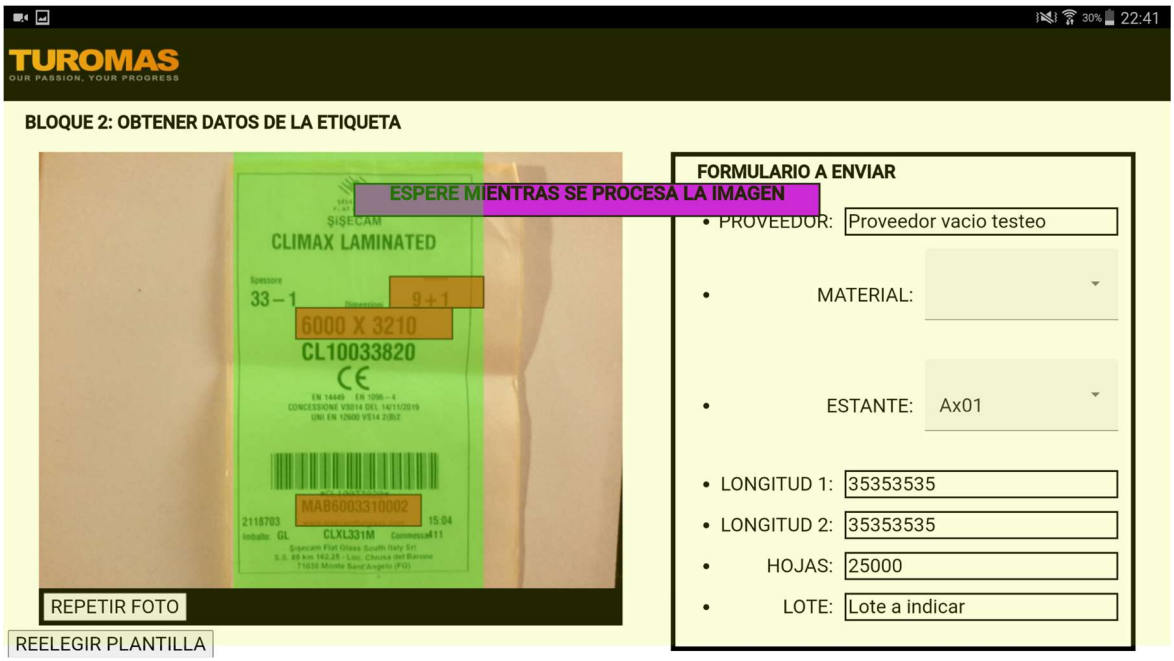

Figura 33. OCR procesando imagen

Una vez obtenida toda la información, ésta aparece sobre el apartado de la IU FORMULARIO A ENVIAR, en cada uno de sus campos.

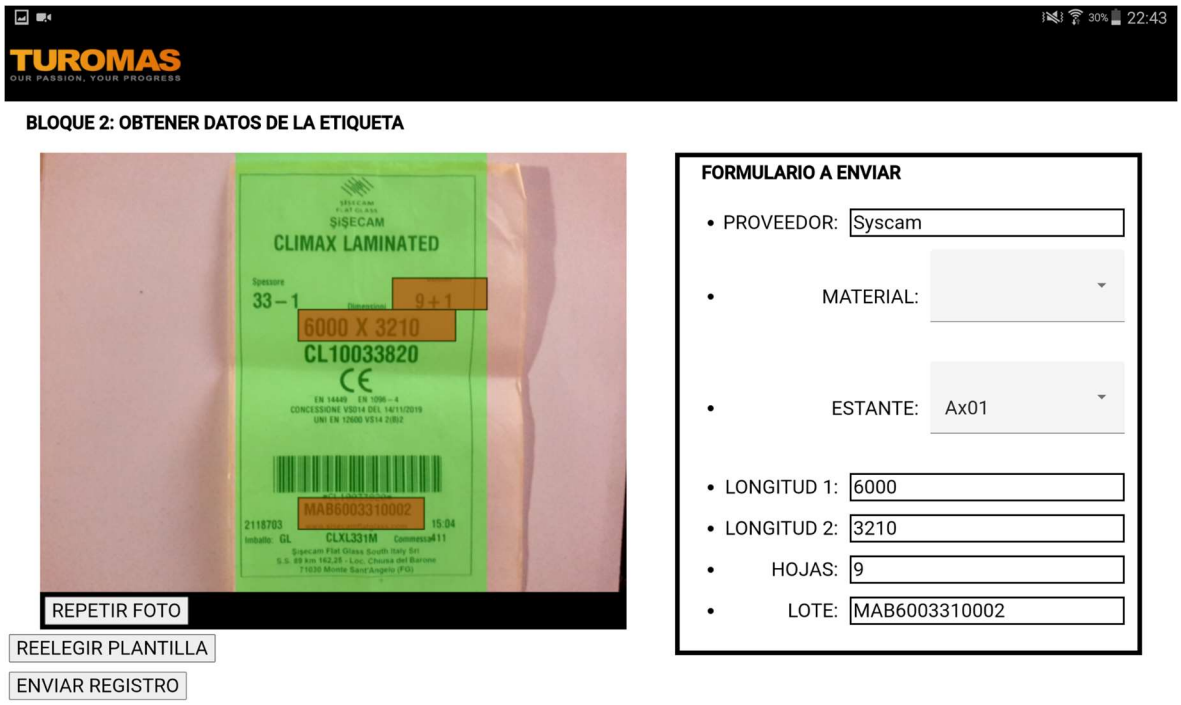

Figura 34. Datos obtenidos por el OCR

Ahora, el operario debe revisar los resultados obtenidos por el OCR y en caso de que alguno sea erróneo, editarlo manualmente. Así mismo, tiene que seleccionar el material

del producto y el caballete de almacenamiento, de dos listas que se han obtenido mediante el servicio apiTuromas, como se muestra en la figura 35.

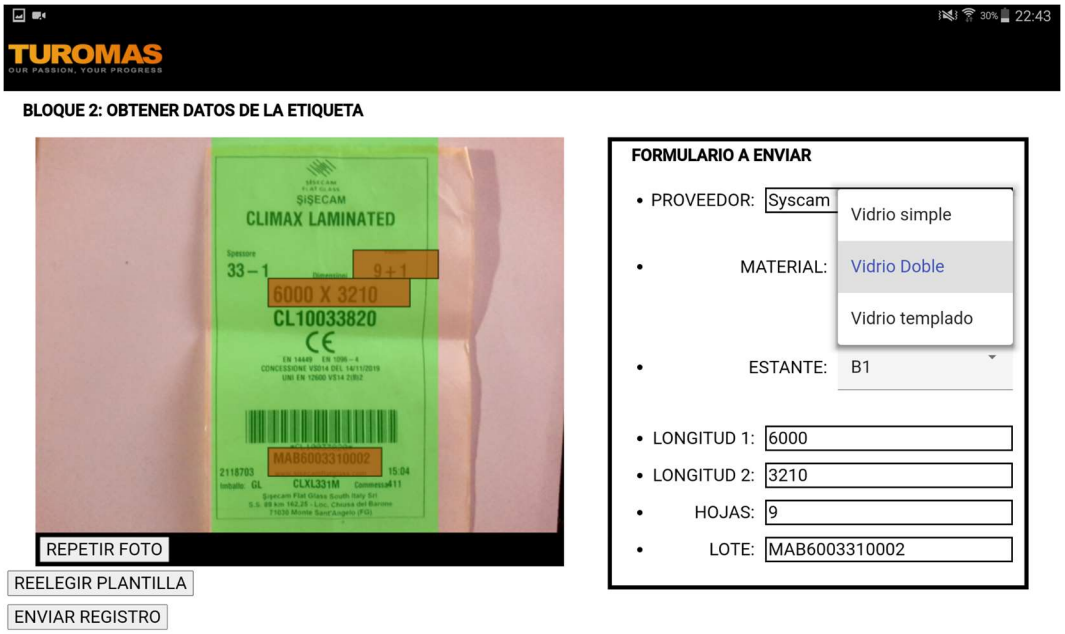

Figura 35. Selección del material

Cuando toda la información es la correcta, se pulsa el botón “ENVIAR REGISTRO” y se manda al ecosistema Turomas la información obtenida, donde es incorporada.

3.3.4.Implementación del Back-End

El back-end se ha desarrollado con SpringBoot, un módulo del proyecto Spring que simplifica el desarrollo de aplicaciones con Spring Framework.[11][12] Esto se debe a que configura automáticamente Spring y las dependencias.

Las dependencias, pertenecen a Spring Framework y a mysql y son:

- spring-boot-starter-data-jpa
- spring-boot-starter-web
- spring-boot-starter-test
- spring-boot-devtools
- mysql-connector-java

Además, se incluye el plugin spring-boot-maven-plugin

La estructura del back-end está agrupada en paquetes, los cuales se observan en la figura 36.

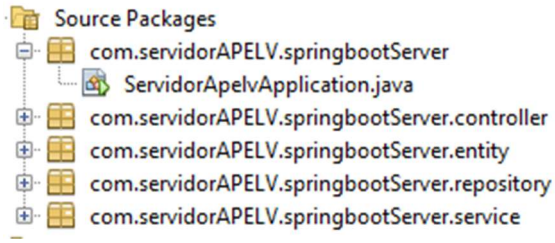

Figura 36. Estructura del back-end

El archivo application.properties define las características de conexión con la base de datos y el comportamiento de Spring con dicha BD. (ver figura 37).

```
spring.datasource.url=jdbc:mysql://localhost:3306/turomas_plantillas
spring.datasource.username=******
spring.datasource.password=******
spring.datasource.driver-class-name =com.mysql.jdbc.Driver
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
spring.jpa.show-sql: true
spring.jpa.hibernate.ddl-auto=validate
logging.level.org.hibernate=debug
```

Figura 37. Archivo de configuración de interacción con la BD

Al desplegarse, Spring mapea las clases indicadas en el paquete entity con las tablas correspondientes de la BD. Remarcar el valor "spring.jpa.hibernate.ddl-auto" que indica el curso de acción de Spring al realizar la conexión con la base de datos, "validate", el valor seleccionado hace que únicamente se verifique el modelo de la BD con el indicado en el back-end. Otros modos pueden destruir la BD existente y crearla de 0 cada vez que se inicia la aplicación.

El paquete entity, contiene las entidades del modelo de negocio que mapean con el ORM Hibernate las tablas de la BD. Son dos, plantilla y campo.
A través de la importación de la librería javax.persistance [9], se da la posibilidad de mapear las entidades mediante anotaciones.

@Entity indica a Spring que registre dicha clase como entidad. También es necesario crear los diferentes atributos de la entidad. Se utiliza @Table(name = "plantillas") para indicar el mapeo a la tabla de nombre "plantillas".
Dentro de las entidades se destacan tres anotaciones:

1) @Id para indicar el identificador de dicha entidad
2) @Column(lenght = X) para añadir un atributo de la longitud determinada por X y del tipo indicado por la variable anotada.

3) @OneToMany permite mapear relaciones entre diferentes entidades. Siguiendo la documentación de Oracle, se recomienda hacer las relaciones bidireccionales, por ello, es necesario incluir el tag @ManyToOne en la otra entidad involucrada en la relación.

```
@Entity
@Table(name = "plantillas")
public class Plantilla implements Serializable{
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name="id_plantilla")
    private Long id_plantilla;

    @Column(length = 50)
    private String nombre;
```

Figura 38. Mapeo de clases como entidades

El paquete repository, contiene los Data Access Object (DAO) encargados de las conexiones con la BD. Para ello se extiende JpaRepository<plantilla/campo,Long>, el cual indica los métodos a usar.

Esto deja un @Repository preparado al que el @Service podrá llamar cuando los necesite. De esta manera el proyecto dispondrá de bajo acoplamiento.

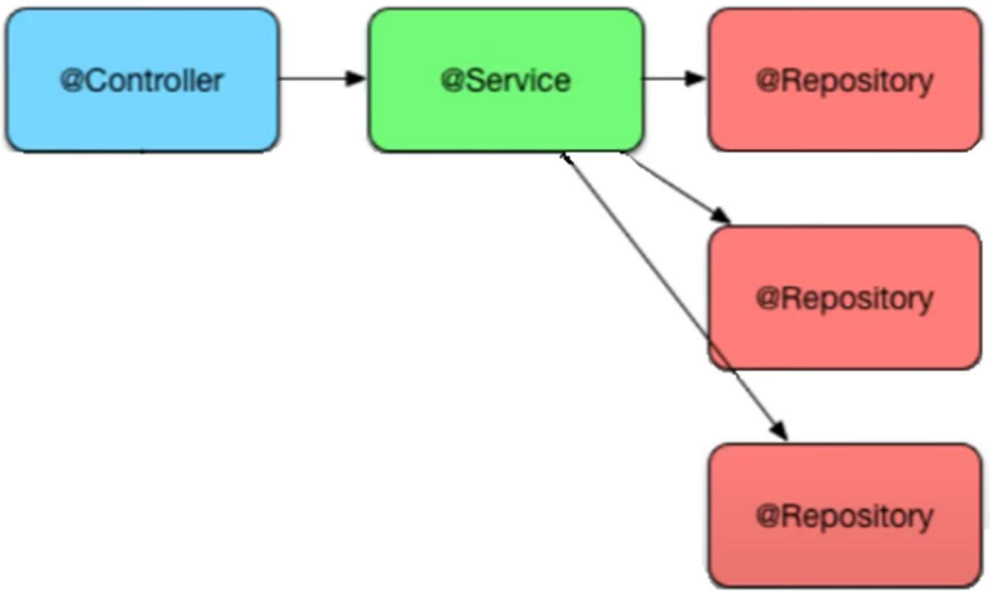

Figura 39. Esquema de paginación desde el servidor.

El paquete service, actúa usando el patrón fachada, en SpringBoot es llamado Service, para evitar poner en el controller y en el repository todo el código que no está relacionado directamente con el controller ni con el repository.

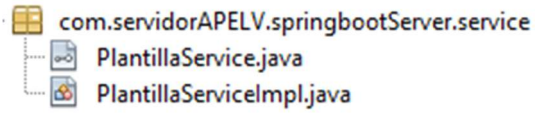

Figura 40. Estructura del paquete service

Para cada entidad se dispone de una interfaz, en la que se declara un listado de métodos. Esta interfaz, luego es implementada, y en dicha clase, además se inyecta el @Repository creado anteriormente. Es necesario indicar la anotación @Transactional y si la transacción es unicamente de lectura o no (para alterar o no la BD).
Ahora, el controlador, mediante inyección de dependencias, podría usar todos los métodos de la interfaz PlantillaService.

```
@Service
public class PlantillaServiceImpl implements PlantillaService{

    @Autowired
    private PlantillaRepository plantillaRepository;

    @Override
    @Transactional(readOnly = true)
    public Iterable<Plantilla> findAll() {
        return plantillaRepository.findAll();
    }

    @Override
    @Transactional(readOnly = true)
    public Page<Plantilla> findAll(Pageable pageable) {
        return plantillaRepository.findAll(pageable);
    }
```

Figura 41. Implementación del service

El paquete controller, dispone de una anotación @RestController, @RequestMapping(“/api/plantillas”)

```java
@RestController
@RequestMapping("/api/plantillas")
@CrossOrigin
public class PlantillaController {

    @Autowired
    private PlantillaService plantillaService;
```

Figura 42. Controlador del Back-End

Como se ve en la figura 42 se inyecta por dependencias el servicio. Este controlador es a la vez un endpoint al que se conectará el front-end y el encargado de solicitar los datos de la BD a través de los métodos en la interface service.

```java
//Obtener todas las Plantillas
@GetMapping
public List<Plantilla> readAll(){
    /*
    * Itera secuencialmente sobre los elementos del Iterable.
    * collect los une en una lista
    */
    List<Plantilla> plantillas = StreamSupport
            .stream(plantillaService.findAll().spliterator(), false)
            .collect(Collectors.toList());

    return plantillas;
}
```

Figura 43. Método Get que devuelve al front end el listado de plantillas

Todo el código se encuentra disponible en el siguiente repositorio de GitHub: https://github.com/737325/TFG_AntonioSantaIsabelBosqued.git

3.4.Pruebas

Para validar el correcto funcionamiento de la aplicación, se han desarrollado diferentes pruebas.

3.4.1.Testeo de diferentes etiquetas

En primer lugar, se dispone de un número de etiquetas obtenidas de la fábrica. Se han agrupado por tipo, ya que se disponen de varias etiquetas del mismo modelo.

Sobre estos grupos se ha hecho lo siguiente:

- Una de las etiquetas, normalmente la que estaba en mejor estado, se ha cogido como referencia de estar en buen estado, sin desperfectos.
- La otra etiqueta, en caso de venir en buen estado también, ha sido sometida a desperfectos que son normales encontrar en el ambiente de la fábrica. Esto puede ser, suciedad, dobleces en la etiqueta o incluso arañazos que eliminan parte de la etiqueta. (ver anexo I)

De esta manera, se dispone de dos tipos etiquetas. Sobre cada una se ha repetido el proceso de OCR hasta 6 veces. Midiendo en cada prueba:

- El tiempo de respuesta del OCR, esto es el tiempo que tarda el programa en actualizar los datos leídos en la pantalla.
- La cantidad de campos que han sido correctos
- En el caso de los campos incorrectos, el fallo ha sido catalogado dentro de 2 categorías. De los cuales, en caso de múltiples fallos en una lectura, se indica el de mayor gravedad.
    + Mínimo, el fallo es de dos o menos caracteres erróneos, ya sean faltantes (no ha reconocido un carácter) o sobrante (ha añadido o duplicado un carácter)
    + Considerable, el OCR reconoce símbolos que se asemejan, la traducción directamente no muestra nada o algún símbolo que no guarda relación con el mensaje (p.ej unas “ o ^ en lugar de un número de 5 dígitos). En esta categoría de fallos cuesta más editar el error para arreglarlos que repetir la foto, y cuando sucede este tipo de fallo suele ocurrir en más de una categoría, ya que suelen producirse cuando la etiqueta está en muy mal estado, o la foto es borrosa.

Los resultados que se muestran están agrupados por modelo, la primera fila de cada subbloque determina los resultados de las pruebas sobre la etiqueta en buen estado, mientras que la segunda representa las pruebas con la etiqueta con desperfectos. (ver tabla 3).

ETIQUETA	Tiempo medio	Prueba	1º	2º	3º	4º	5º	6º
Guardian Glass Ancha. Buen estado	27,83333333	Tiempo de respuesta	25	24,5	30,5	30	31	26
		Campos erroneos	0	0	2	0	0	0
		Tipo de fallo	Ninguno	Ninguno	Considerable	Ninguno	Ninguno	Ninguno
Guardian Glass Ancha. Desperfectos	27,58333333	Tiempo de respuesta	23	28	29	29	30,5	26
		Campos erroneos	1	0	1	2	1	0
		Tipo de fallo	Minimo	Ninguno	Minimo	Considerable	Considerable	Ninguna
Guardian Glass Estrecha. Buen estado	30,5	Tiempo de respuesta	35	35	30	28	27	28
		Campos erroneos	0	0	0	0	0	1
		Tipo de fallo	Ninguno	Ninguno	Ninguno	Ninguno	Ninguno	Minimo
Guardian Glass Estrecha. Desperfectos	31,25	Tiempo de respuesta	30	32	31	31,5	31	32
		Campos erroneos	1	2	1	1	2	2
		Tipo de fallo	Minimo	Considerable	Minimo	Minimo	Considerable	Considerable
Sisecam Climax. Buen Estado	30,66666667	Tiempo de respuesta	32	30	32	29	30	31
		Campos erroneos	0	1	0	0	0	1
		Tipo de fallo	Ninguno	Minimo	Ninguno	Ninguno	Ninguno	Minimo
Sisecam Climax. Desperfectos	30,25	Tiempo de respuesta	32	31	29,5	30	33	26
		Campos erroneos	1	2	1	0	0	2
		Tipo de fallo	Minimo	Considerable	Minimo	Ninguno	Ninguno	Considerable

Tabla 3. Listado de pruebas con distintas plantillas

Las imágenes del estado de las etiquetas se pueden consultar en el Anexo I.

3.4.2.Pruebas con usuarios

Tras las pruebas se han obtenido los siguientes resultados:

- Si las etiquetas están en buen estado, la tasa de fallos (pruebas con fallos de etiquetas en buen estado entre pruebas total de etiquetas en buen estado) está en torno al 22% y de estos fallos en solo un caso ha sido un fallo considerable. El resto de fallos son de un carácter sobrante o que no ha sido leído, estos fallos están provocados porque no se ha enfocado el área entera correctamente o la etiqueta tiene una pequeña marca que tapa algún dígito concreto. Es decir, en solo un 5.5% de los casos sería necesario repetir la foto.

- En el caso de las etiquetas con desperfectos, un 77.7% de los reconocimientos OCR han sufrido al menos un fallo. De los cuales, un 50% son considerables, y el resto mínimos. En estas plantillas, los fallos que más afectan al resultado exitoso son las rasgaduras de etiquetas donde es imposible leer los datos faltantes, después se encuentran las dobleces extremas de las plantillas, las cuales deforman la plantilla al punto que es muy complicado que cuadren todos los campos
- Los tiempos de espera rondan los 30 segundos, este tiempo se reduce ligeramente si la foto es clara y los campos están limpios. Mientras que la suciedad y hacer la foto en movimiento aumenta los tiempos de procesamiento a la vez que empeora los resultados.

Para esta batería de pruebas, se ha usado a personas de distintos rangos de edad, cuatro personas entre los 20 y los 25, cinco personas en el rango de 35 -52 años y dos personas de más de 70 años. Esto es relevante, ya que en la fábrica hay personas que superan los 50 años.

A todas estas personas se les pidió que realizaran varios envíos de formulario sobre etiquetas, simulando la situación de la fábrica, con la etiqueta en posición vertical (los paquetes de hojas de vidrio antes de almacenarse, están o colgando de un puente aéreo o apoyados en caballetes de apoyo) los usuarios debían realizar la foto.

A partir de estas pruebas, se obtuvieron diversas conclusiones y puntos de mejora para la interfaz:

- Los botones deben estar separados entre sí. Ocurría que varios usuarios, al intentar pulsar el botón Foto, pulsaban por error el de “Reelegir Plantilla”
- Es importante ajustar los campos, ciertos campos tienen que ser muy ajustados, ya que pueden estar rodeados de otros textos. Esto ha de ser compensado expandiendo el resto de campos en la medida que sea posible. Por ejemplo, los datos de interés que disponen de espacio es recomendable expandir el área, ya que únicamente se leerá el área en blanco y así se da flexibilidad a ajustar los campos más complicados.
- Los grupos de más de 35 años no tuvieron dificultades más allá de enfocar correctamente los campos (por ello la importancia del punto anterior), el funcionamiento de la aplicación fue fácil para estas personas de entender.
- Hubo dificultades, entre personas pertenecientes a diferentes grupos de edad, de obtener fotos correctamente debido al pulso. Por ello la función de repetir imagen es tan necesaria.

Como posible solución futura la colocación de los campos de manera manual sobre la imagen sería una posible solución.

4.Accesibilidad y Usabilidad

Según el ISO 25010, la accesibilidad es: “Grado en el que los datos pueden ser accedidos en un contexto específico, particularmente por personas que necesiten tecnologías de apoyo o una configuración especial por algún tipo de discapacidad.” [7]

En España, la legislación actual se divide en el marco nacional y el europeo. En el marco nacional se encuentra el Real Decreto 1112/2018. [4] Este RD afecta a las administraciones públicas y otros obligados, estos son, aquellos sitios web y aplicaciones que reciben financiación pública. Fija el estándar aplicable previo español con el europeo, haciendo referencia a las WCAG 2.1 del W3C[16]. A nivel europeo existe la EN 301 549 [2], que cumple parcialmente las pautas del WCAG 2.1.

Tanto el RD 1112/2018 como la EN 301 549 V2.1.2:2018 legislan lo mismo, que las páginas web en el ámbito de la accesibilidad deben ajustarse a la norma WCAG 2.1, emitidas por el W3C. Si bien la mayoría hacen referencia a las administraciones públicas/europeas o a aquellas empresas que cumplan ciertos requisitos que las cataloguen como de especial trascendencia económica, el ámbito del TFG no entraría en ninguna de estas categorías por lo que no es necesario, si bien deseable, cumplirlas.

Considero que la aplicación no cumple los requisitos para ser accesible para todo el mundo, pero, por otro lado, creo que es accesible para todas las personas que pueden trabajar en el ámbito que se va a usar.

Usabilidad

Para valorar la usabilidad de la aplicación se ha usado como medidor los diez principios de Jakob Nielsen[15], de los cuales podemos ver un esquema en la figura 44.

Figura 44. Diez principios de Nielsen

A continuación, se enumeran, junto a una pequeña descripción y un argumento sobre si se considera que la aplicación cumple con cada uno de ellos:

1. **Visibilidad del estado del sistema**. El sistema debe mantener siempre informado al usuario de lo que ocurre. En el caso del sistema desarrollado esto se cumple, ya que el único momento en el que el operario ha de esperar, es cuando se ejecuta el procesamiento OCR.
2. **Relación entre el sistema y el mundo real.** El sitio web o aplicación tiene que utilizar el lenguaje del usuario, con expresiones y palabras que le resulten familiares. El sistema usa el mismo lenguaje técnico que el usado en la fábrica, y como se explica en el apartado "Características del usuario" debido al entorno en el que se va a usar la aplicación, los operarios que harán uso de ella, ya conocen el vocabulario específico. El resto del lenguaje son instrucciones sencillas y claras
3. **Control y libertad del usuario.** En caso de elegir una opción del sitio por error, el usuario agradecerá un método para deshacer el estado no deseado. Las tres acciones que realiza el usuario son: elegir la plantilla, hacer la foto y modificar los atributos. La última acción por sí sola, le permite al usuario hacer y deshacer a gusto. En el caso de la plantilla, si el usuario se equivoca dispone de un botón para volver a reelegir la plantilla. Y en el caso de la foto, tras hacer la foto, aparece un botón que permite repetir la foto.

4. **Consistencia y estándares.** Establecer convenciones lógicas y mantenerlas siempre. La interfaz guarda la misma estructura en ambas pantallas. La cabecera es la misma y las opciones son sencillas al estar indicadas con palabras claras y concisas.
5. **Prevención de errores.** Ayuda al usuario a que no caiga en un error. Al esperar el resultado del OCR, se activa en la pantalla un canvas que bloquea toda la pantalla, esto se hace para evitar fallos de ejecución o bloqueos provocados por el usuario.
6. **Reconocimiento antes que recuerdo.** Se debe hacer visibles acciones y opciones para que el usuario no tenga que recordar información entre distintas secciones o partes del sitio web o aplicación. La información que se comparte entre secciones es la plantilla, según se transiciona a la segunda pantalla, esta aparece dibujada sobre la cámara inmediatamente.
7. **Flexibilidad y eficiencia de uso.** Los aceleradores o atajos de teclado, por ejemplo, pueden hacer más rápida la interacción para usuarios expertos, de tal forma que el sitio web o aplicación sea útil tanto para usuarios básicos como avanzados. Este punto no se cumple completamente, la aplicación es rígida, permite la corrección de fallos, pero no dispone de accesos rápidos, y dependiendo de la plantilla, puede ser laborioso hacer una foto válida (por la dimensión de algún campo, que haya zonas de texto muy juntas)
8. **Estética y diseño minimalista.** Las páginas no deben contener información innecesaria. Este punto se cumple, ya que únicamente se muestra la mínima información. En la primera pantalla, únicamente aparecen las plantillas disponibles. En la segunda, se encuentra la cámara, el formulario a enviar y los tres botones de navegación.
9. **Ayudar a los usuarios a reconocer, diagnosticar y recuperarse de errores.** Los mensajes de error se deben entregar en un lenguaje claro y simple, indicando en forma precisa el problema y sugerir una solución constructiva al problema. Los errores del programa no se presentan al usuario, al finalizar el proceso de OCR, es el propio usuario el que ha de revisar los campos leídos. Así mismo, si el usuario recarga la página en la segunda plantilla, no dibujará ninguna plantilla, por lo que el usuario deberá volver a la primera pantalla para elegir una.
10. **Ayuda y documentación.** Aunque es mejor que el sitio web o aplicación pueda ser usado sin ayuda, puede ser necesario proveer cierto tipo de ayuda. Al ser una aplicación que va a ser usada por operarios de la fábrica, ya se presupone que disponen de la formación necesaria para usar dicha aplicación. Por lo que no se dispone de ayuda extra

5.Licencia Software y Documental

En este apartado, se va a proceder a comentar la licencia de software y la licencia documental.

En cuanto a la licencia de software se va a emplear Berkeley Software Distribution (BSD). Se trata de una licencia de software libre permisiva como puede ser OpenSSL o la MIT License. Existen diferentes tipos de licencias, en el caso de este TFG se ha utilizado la licencia "BSD modificada", "BSD revisada", "BSD-3" o "BSD de 3 cláusulas" [14]

Figura 44. Logotipo de la licencia BSD

Al igual que sucede en el mundo del software, se tienen que buscar formas de garantizar las libertades asociadas al trabajo elaborado y su inviolabilidad futura. Para garantizar que la libertad esté asociada al documento se buscan métodos, uno de ellos es la licencia GNU Free Documentation License (GFDL).

Figura 45. Logotipo de la licencia GNU

El propósito de esta Licencia es hacer que en el caso de este TFG sea 'gratuito' en el sentido de libertad: para asegurar a todos la libertad efectiva de copiarlo y redistribuirlo, con o sin modificarlo, ya sea comercial o no comercialmente. En segundo lugar, esta licencia preserva para el autor y el editor una forma de obtener crédito por su trabajo, sin ser considerado responsable de las modificaciones realizadas por otros. Es una especie de 'copyleft', por lo que las obras derivadas de este han ser libres en el mismo sentido. Si se emplea este documento y se modifica, debe realizar una serie de acciones indicadas en el sitio web oficial de GNU.[5]

Este documento, por defecto, está al amparo de la licencia Creative Commons, por su inclusión en el Repositorio Institucional de Documentos de la Universidad de Zaragoza: ZAGUAN

Figura 46. Licencia de ZAGUAN

6.Conclusiones y Trabajo Futuro

El desarrollo de este trabajo ha servido para ver realmente las aplicaciones de asignaturas que durante su estudio, no tenían sentido para mí. Entender todo el proceso de desarrollo de un proyecto, desde su concepción a su final, pasando por un intenso y laborioso desarrollo.

Además, he podido observar cómo se complementan las diferentes asignaturas entre sí y forman un todo mucho más amplio. Así como el proceso de tener que buscar información verídica y no obtenida de cualquier sitio de internet.

Me ha hecho darme cuenta de muchas problemáticas y puntos que se han de tener en cuenta, para la duración, mantenibilidad y robustez del proyecto, de los cuales no me habría percatado si el proyecto fuese una práctica o ejercicio a entregar del cual no volviese a saber nadie nunca.

El haber participado en el mundo laboral antes de ser titulado, me ha servido para enfocar la recta final de la carrera con mucho más entusiasmo, seguridad y confianza en los conocimientos adquiridos, sabiendo un poco lo que me voy a encontrar y aprendiendo un poco más sobre el mundo informático más allá del ámbito académico.

En cuanto al trabajo en sí, creo que se ha desarrollado el objetivo buscado, ya que se ha creado una herramienta que estoy seguro agilizará el proceso de etiquetado y solventará los problemas que hacerlo a mano traía consigo.

Sobre el trabajo futuro para ampliar este TFG, a la vista hay varios proyectos interesantes. Uno sería la posibilidad de crear plantillas o editarlas de manera dinámica, arrastrando y ajustando los campos por la pantalla.

Siguiendo la retroalimentación obtenida en la batería de pruebas, la opción de colocar manualmente los campos en cada foto obtenida, es muy interesante y abriría la puerta a otros trabajos por sí misma.

7.Referencias bibliográficas

[1] “About the Unified Modeling Language Specification Version 2.5.1,” Omg.org, 2017. https://www.omg.org/spec/UML (accessed Feb. 24, 2022).

[2] “Accessibility requirements for ICT products and services,” Aug. 2018. Accessed: Sep. 05, 2022. Available: https://www.etsi.org/deliver/etsi_en/301500_301599/301549/02.01.02_60/en_301549v020102p.pdf

[3] “Angular,” Angular.io, 2022. https://angular.io/docs (accessed Jun. 15, 2022).

[4] “BOE.es - BOE-A-2018-12699 Real Decreto 1112/2018, de 7 de septiembre, sobre accesibilidad de los sitios web y aplicaciones para dispositivos móviles del sector público.,” Www.boe.es, 2018. https://www.boe.es/eli/es/rd/2018/09/07/1112/con (accessed Sep. 13, 2022).

[5] “El sistema operativo GNU y el movimiento del software libre,” Gnu.org, Sep. 11, 2022. https://www.gnu.org/ (accessed Sep. 13, 2022).

[6]. IEEE 830. (1998). IEEE Recommended Practice for Software Requirements SpeciÞcations. New York: IEEE.

[7] “ISO 25012,” Iso25000.com, 2022. https://iso25000.com/index.php/normas-iso-25000/iso-25012 (accessed Sep. 05, 2022).

[8] I. Sommerville, Software engineering, 7th ed. Harlow: Pearson/Addison-Wesley, 2005.

[9] “Java EE 6,” Oracle.com, 2022. https://docs.oracle.com/javaee/6/api/index.html (accessed Jun. 3, 2022).

[10] “¿Qué es el reconocimiento óptico de caracteres (OCR)? | AWS,” Amazon Web Services, Inc., 2022. https://aws.amazon.com/es/what-is/ocr/ (accessed February. 20, 2022).

[11] “Spring Boot,” Spring.io, 2022. https://spring.io/projects/spring-boot/ (accessed Mar. 13, 2022).

[12] “Spring Guides,” Spring.io, 2022. https://spring.io/guides (accessed Mar. 15, 2022).

[13] “Tesseract User Manual,” tessdoc, 2021. https://tesseract-ocr.github.io/tessdoc/ (accessed Sep. 10, 2022).

[14] “The 3-Clause BSD License | Open Source Initiative,” Opensource.org, 2014. https://opensource.org/licenses/BSD-3-Clause (accessed Sep. 12, 2022).

[15]. “10 Usability Heuristics for User Interface Design,” Nielsen Norman Group, 2020. https://www.nngroup.com/articles/ten-usability-heuristics/ (accessed Aug. 12, 2022).

[16]“Web Content Accessibility Guidelines (WCAG) 2.1,” W3.org, Jun. 05, 2018. https://www.w3.org/TR/2018/REC-WCAG21-20180605/ (accessed Sep. 13, 2022).

[17] “What Is Optical Character Recognition (OCR)?,” Ibm.com, Jan. 05, 2022. https://www.ibm.com/cloud/blog/optical-character-recognition (accessed February. 10, 2022).

Anexo I – Etiquetas de prueba

Las etiquetas usadas para las pruebas se dividen en dos categorías, buen estado y desperfectos. Todas estas etiquetas han sido fotografiadas por el dispositivo usado para las pruebas, la tablet Samsung Galaxy Note 10.1, la cual se puede ver en la figura 47.

Figura 47. Dispositivo usado para las pruebas

Las etiquetas aparecen agrupadas por modelos, en el lado izquierdo la que se encuentra en buen estado y a la derecha la que tiene desperfectos.

Figura 48. Etiquetas Sisecam climax

Figura 49. Etiquetas Guardian Glass Ancha.

Figura 50. Etiquetas Guardian Glass Estrechas

La etiqueta visible en la figura 51, era un modelo único, no se disponía de más unidades, por lo que se hicieron las pruebas sobre ella, y más tarde se procedió a ensuciar con mina de lapicero.

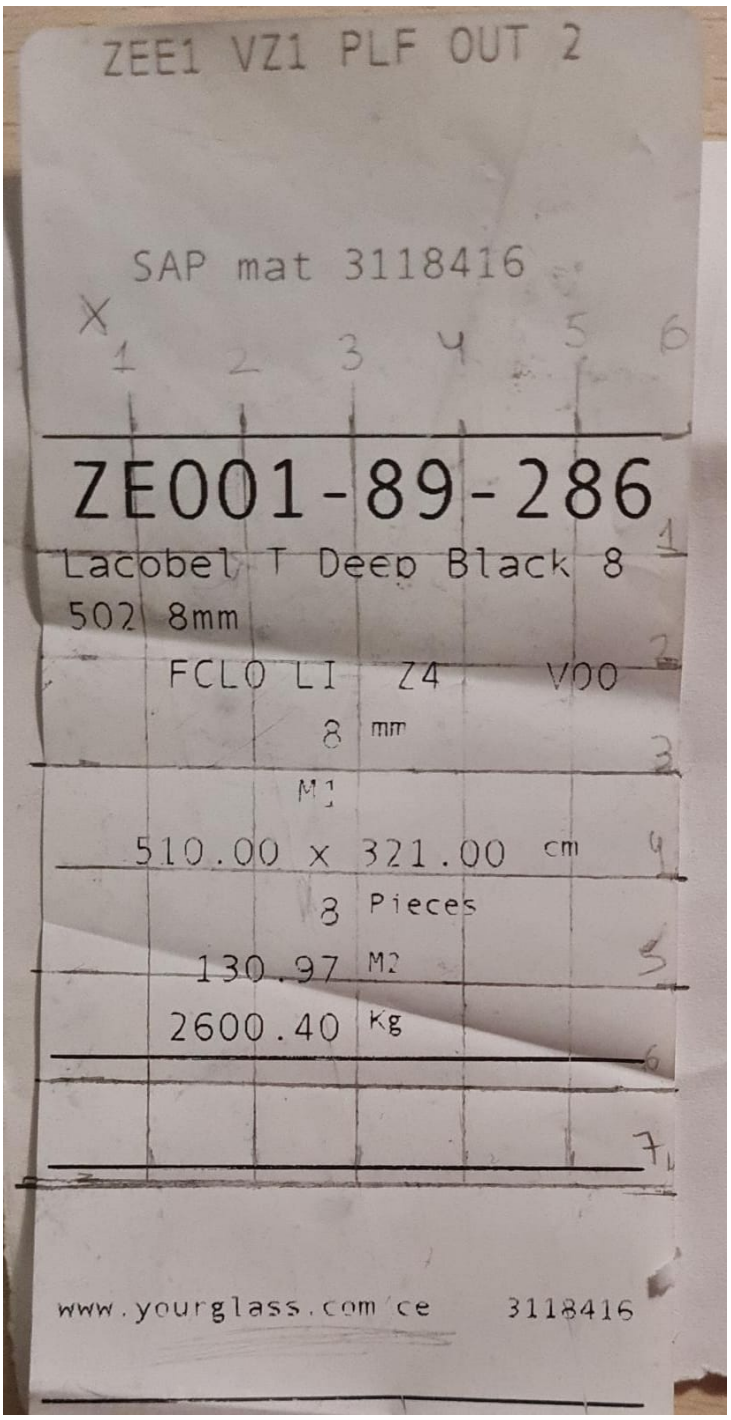

Figura 51. Etiqueta yourglass

Figura 52. Comparación de etiquetas del proveedor Guardian Glass.