Memoria del Proyecto

Sistema automático de identificación de productos para el sector retail

Trabajo de Fin de Grado

INGENIERÍA INFORMÁTICA

VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

Julio de 2021

Autor:

Alexander Fuentes Bartolomé

Tutores:

Luis Augusto Silva

Juan Francisco De Paz Santana

Gabriel Villarrubia González

Certificado de los tutores

D. Juan Francisco de Paz Santana y D. Gabriel Villarrubia González, profesores del Departamento de Informática y Automática de la Universidad de Salamanca y D. Luis Augusto Silva, personal investigador de la Universidad de Salamanca.

HACEN CONSTAR:

Que el trabajo titulado “Sistema automático de identificación de productos para el sector retail” ha sido realizado por D. Alexander Fuentes Bartolomé, con el número de DNI 71044796G y constituye la memoria del trabajo realizada para la asignatura Trabajo de Fin de Grado de la Titulación Grado de Ingeniería Informática de esta Universidad.

Y para que así conste a todos los efectos oportunos.

En Salamanca, a 6 de Julio de 2021.

DE PAZ SANTANA JUAN FRANCISCO - 76125754D

Firmado digitalmente por DE PAZ SANTANA JUAN FRANCISCO - 76125754D
Nombre de reconocimiento (DN): c=ES, serialNumber=IDCES-76125754D, givenName=JUAN FRANCISCO, sn=DE PAZ SANTANA, cn=DE PAZ SANTANA JUAN FRANCISCO - 76125754D
Fecha: 2021.07.06 20:44:45 +02'00'

SILVA LUIS AUGUSTO - Y8119893X

Assinado de forma digital por SILVA LUIS AUGUSTO - Y8119893X
Dados: 2021.07.07 02:12:14 +02'00'

VILLARRUBIA GONZALEZ GABRIEL - 71932845T

Firmado digitalmente por VILLARRUBIA GONZALEZ GABRIEL - 71932845T
Nombre de reconocimiento (DN): c=ES, serialNumber=IDCES-71932845T, givenName=GABRIEL, sn=VILLARRUBIA GONZALEZ, cn=VILLARRUBIA GONZALEZ GABRIEL - 71932845T
Fecha: 2021.07.07 11:47:20 +02'00'

Resumen

El proyecto que se ha realizado como trabajo de fin de grado consiste en la creación de un sistema automático de identificación de productos para el sector retail, buscando que con este sistema los productos introducidos por los usuarios en su carro de la compra sean detectados para su posterior consulta o pago. Con esto se consigue que los usuarios sepan en todo momento y sin lugar a duda los productos que han introducido en su carro para que una vez vayan a finalizar su compra sepan de antemano el precio exacto que les costará y no necesitarán extraer los productos del carro para pagar.

La detección de los productos introducidos o extraídos de los carros de la compra se logrará mediante una cámara, la cual utilizará el algoritmo YOLOv4-tiny para la detección de objetos que una vez detectados, realizará una llamada al servidor central para que almacene los productos introducidos en el carro.

Para la gestión de los diferentes apartados del sistema se ha creado una aplicación web a la cual los administradores del sistema se podrán conectar, además podrán ver fácilmente diferente información del sistema a través de esta.

El sistema por medio de una pantalla en los carros mostrará un código QR que los usuarios podrán escanear con la aplicación que se ha creado. Una vez escaneado el código, podrán ver los productos de la compra que se está realizando y cuando hayan acabado de comprar, pagarla de una forma sencilla y rápida desde su dispositivo móvil. Así se conseguiría evitar congestionamientos en los supermercados y prevenir el hastío que sufren algunos compradores al realizar sus compras.

Por tanto, lo que se busca con este proyecto es facilitar las compras a los clientes, ya que muchos de estos no pueden realizar sus compras de forma plena por falta de tiempo al tener que lidiar con eternas filas de clientes dispuestos a pagar por sus compras, además de proveer información de los productos introducidos en sus carros (precio por unidad, cantidad de un mismo producto introducido, precio total de la compra, etc).

Palabras clave: detección de productos, supermercado, cámara, pago autónomo, consulta de productos.

Summary

The project that has been carried out as a final degree project consists of the creation of an automatic product identification system for the retail sector, seeking that with this system the products introduced by users in their shopping cart are detected for later consultation or payment. This allows users to always know and without any doubt the products they have entered in their cart so that once they go to finalize their purchase, they know in advance the exact price it will cost them, and they will not need to extract the products from the cart to pay.

The detection of the products introduced or removed from the shopping carts will be achieved employing a camera, which will use the YOLOv4-tiny algorithm for the detection of objects that, once detected, will make a call to the central server to store the products introduced in the cart.

For the management of the different sections of the system, a web application has been created to which the system administrators will be able to connect and easily view different system information through it.

The system will display a QR code on a screen on the trolleys that users can scan with the application that has been created. Once the code is scanned, they will be able to see the products of the purchase being made and when they have finished shopping, pay for it in a simple and fast way from their mobile device. This would avoid congestion in supermarkets and prevent the boredom that some shoppers suffer when shopping.

Therefore, this project aims to make shopping easier for customers, since many of them cannot entirely make their purchases due to lack of time when having to deal with long lines of customers willing to pay for their purchases, in addition to providing information of the products introduced in their carts (price per unit, quantity of the same product introduced, total price of the purchase, etc.).

Keywords: product detection, supermarket, camera, autonomous payment, consult products.

Tabla de Contenidos

Índice de Figuras

Índice de Tablas

1. Introducción

En la actualidad, existe un gran número de personas que no disponen del tiempo suficiente para realizar tareas de la vida cotidiana, tales como realizar la compra. Con este proyecto, se pretende solventar este problema haciendo que las compras sean más provechosas, más eficientes, (puesto que no hay intervención externa de otras personas, solo el usuario quien introduce los artículos en su carro) y, por lo tanto, más rápidas.

El proyecto cuenta con un carro de la compra capaz de detectar los productos entrantes y salientes de este. Una vez detectado el producto, el carro realizará una llamada al servidor para modificar los productos que se encuentran en dicho carro.

Los clientes podrán escanear el código QR que les mostrará el carro a través de la aplicación creada, desde la cual podrán ver los productos que se hallan en el carro, además de pagar la compra una vez estos la hayan finalizado.

Para poder llevar a cabo las tareas de gestión del sistema, se ha creado una web desde la cual los administradores podrán obtener información del sistema, así como administrarlo.

En este documento se llevará a cabo la explicación de los aspectos más importantes del desarrollo del proyecto. Por lo tanto, la estructura que seguirá el documento será la siguiente:

- **Objetivos:** en este apartado se recogen los objetivos que debe cumplir el proyecto.
- **Estado del arte:** se recogen las aplicaciones actuales que tienen relación con el proyecto creado.
- **Conceptos teóricos:** se recogen los conceptos básicos para poder entender el funcionamiento del proyecto.
- **Técnicas y Herramientas:** es la documentación referida a las técnicas y herramientas que se han utilizado para desarrollar el proyecto.
- **Aspectos relevantes del desarrollo:** es el conjunto de los aspectos más importantes que se han llevado a cabo para realizar este proyecto.
- **Limitaciones del prototipo:** son las limitaciones que se presentan al final del proyecto.
- **Conclusiones y líneas de trabajo futuras:** se explican las conclusiones obtenidas a la hora de realizar el proyecto.
- **Bibliografía**

En el documento también se utilizará información contenida en los siguientes anexos:

- **Anexo I – Plan de Proyecto software:** en este anexo se realiza una estimación del esfuerzo y una planificación temporal del proyecto.
- **Anexo II – Especificación de requisitos Software:** recoge las especificaciones de requisitos software del sistema.
- **Anexo III – Análisis del sistema software:** recoge el análisis del sistema software del sistema.
- **Anexo IV – Diseño del sistema software:** recoge el diseño del sistema software del sistema.
- **Anexo V – Documentación técnica:** se documenta la información relevante del código del sistema desarrollado.

- **Anexo VI – Manual de usuario:** recoge la información sobre el funcionamiento del sistema de forma simplificada para que un usuario logre entenderlo.

2. Objetivos

En este apartado se detallarán los objetivos que debe satisfacer el sistema para poder llevar a cabo el desarrollo del proyecto. Se recogerán tanto los objetivos que deberá cumplir el sistema como los objetivos personales.

2.1 Objetivos del sistema

El objetivo principal del proyecto es el desarrollo de un sistema capaz de detectar los productos introducidos al carro, mostrar dichos productos introducidos a los usuarios a través de una aplicación móvil para permitirles pagar y la creación de una web para la gestión de los diferentes ámbitos del sistema.

Los objetivos del sistema son:

- **Detección de Productos:** el sistema deber ser capaz de detectar tanto los productos introducidos como extraídos del carro de la compra esta se llevará a cabo mediante un algoritmo de reconocimiento de imágenes por medio de Deep learning a través de una webacm.
- **Gestión de Usuarios:** el sistema deberá ser capaz de gestionar la información de los usuarios, permitiendo el registro, inicio de sesión, recuperación de la y la modificación del rol de los usuarios.
- **Gestión de Ventas:** el sistema deberá ser capaz de gestionar las ventas que se realicen en el sistema, pudiendo realizar pagos de la compra, consultar las compras realizadas por los usuarios o consultar las ventas totales del sistema por los administradores.
- **Gestión de Productos:** el sistema deberá ser capaz de gestionar la información de los productos, pudiéndose añadir, modificar o eliminar productos del sistema.
- **Visualización de Productos:** el sistema deberá ser capaz de mostrar los productos que se encuentren dentro del carro mediante el escaneo de un código QR mostrado por el carro de la compra.
- **Gestión de Carros:** el sistema deberá ser capaz de gestionar la información de los diferentes carros de la compra que posea el supermercado, permitiendo el registro automático de carros de la compra, así como actualización de su estado.

2.2 Objetivos personales

En este subapartado se detallarán los objetivos personales por los cuales se ha realizado este proyecto como trabajo fin de grado.

El objetivo principal por el cual he decidido realizar este proyecto es porque cuenta con la tecnología de Deep Learning. Siempre he querido aprender el funcionamiento de este tipo de tecnología y he querido aplicarlo en la realidad. Gracias a la elaboración de este proyecto he podido conocer y adentrarme un poco más en este campo.

Otro de los objetivos personales es el poder realizar un proyecto de gran tamaño y poder aplicar todo lo aprendido durante estos años en la universidad y poder ver la aplicación de ingeniería del software a un proyecto real.

En relación con esto último, me gustaría aprender más sobre patrones arquitectónicos y de diseño, los cuales me servirán en un futuro para saber cómo llevar a cabo otros proyectos de forma más eficiente.

Además, durante el desarrollo del proyecto se podrá profundizar más en algunos lenguajes y tecnologías que se van a utilizar a la hora de desarrollar dicho proyecto.

3. Estado del arte

A continuación, se muestran otros productos que hay en el mercado, los cuales buscan satisfacer las mismas necesidades que nuestro proyecto, aunque no con las mismas tecnologías.

Figura 1: Amazon Dash Cart [1]

En la *Figura 1*, se puede observar el carro de la compra creado por Amazon que aún se encuentra en desarrollo. Amazon Dash Cart se anunció en 2020 como un carro para realizar la compra de forma más automática facilitando así las compras a sus clientes, que es uno de los objetivos de este proyecto. Las diferencias más significativas entre el carro de Amazon y el proyecto realizado es que Amazon muestra por una pantalla los productos que los usuarios han introducido al carro a diferencia de tener una aplicación móvil dedicada a ello. Su detección funciona mediante RFID mientras que la de este proyecto funciona mediante reconocimiento de objetos.[1]

Figura 2: Caper Smart Cart [2]

En la *Figura 2*, se puede observar el carro de la compra creado por Caper AI que aún se encuentra en desarrollo. Caper Smart Cart es un carrito de la compra que se encuentra actualmente en desarrollo por la empresa Caper AI. El carrito de la compra de Caper y el de este proyecto tienen bastantes características en común pues ambos detectan los productos que el usuario introduce en el carro mediante el reconocimiento de los objetos, pero Caper AI además, añade otras funcionalidades para que esta tarea sea más fácil añadiendo al carrito 4 cámaras que apuntan al interior del carro desde distintos ángulos, con el uso de sensores de movimiento que detectan el movimiento dentro de la cesta del carro o los sensores de peso que se encuentran en el fondo de la cesta. También añade la posibilidad de pagar desde el propio carro mediante tarjeta de crédito. Vemos así que el carrito de Caper AI busca satisfacer las mismas necesidades que se intentan conseguir en este proyecto, siendo así el carrito de Caper AI una versión bastante refinada de lo que se busca establecer en este proyecto.[2]

Figura 3: Carrito MOBI [3]

En la *Figura 3*, se puede observar el carro de la compra MOBI diseñado por expertos coreanos. MOBI es el prototipo de un carrito que busca al igual que el desarrollado en el proyecto, hacer las compras más fáciles a los usuarios. Los usuarios al igual que en el carrito de este proyecto se tendrán que conectar al carrito de MOBI para comenzar las compras en el supermercado. También puede detectar los productos que se introducen al interior de este para posteriormente pagarlos mediante el móvil. Una gran diferencia con respecto al carro desarrollado en este proyecto es que MOBI se desplaza de forma autónoma a las zonas en la que se encuentran los productos que el usuario le comunica que desea comprar. MOBI es otro gran ejemplo de un carro autónomo que busca facilitar las compras a los usuarios al igual que el desarrollado en el proyecto.[3]

4. Conceptos teóricos

4.1 REST

REST (Representational State Transfer) no se trata de un protocolo si no de una arquitectura, que se basa en una serie de restricciones que se deben tener en cuenta a la hora de crear un servicio web. Para el envío de la información entre el cliente-servidor se utiliza el formato (XML, JSON).[4]

Las restricciones que se deben seguir para considerarse un servicio REST son:

- **Cliente-servidor:** el cliente y el servidor deben estar débilmente acoplados. El cliente no debe conocer cómo se desarrolla el flujo de ejecución en el servidor cuando hace una petición y el servidor no tiene conocimientos de que hará el cliente con la información que le proporciona.
- **Sin estado:** las peticiones al servidor deberán ser independientes unas de otras.
- **Cacheable:** deberá tener algún sistema de almacenamiento caché.
- **Interfaz uniforme:** define una interfaz genérica para poder administrar de manera uniforme las peticiones del cliente al servidor.
- **Sistema de capas:** el servidor podrá utilizar varias capas a la hora de realizar su implementación, con lo que se consigue una mejora en la escalabilidad, el rendimiento y la seguridad.

4.2 WebSocket

El protocolo Websocket permite la interacción entre un cliente y un servidor a través de un único puerto TCP de forma bidireccional. Gracias a este es posible que el servidor le mande información al cliente sin que este le haya realizado ningún tipo de petición. Las conexiones se hacen como se ha dicho anteriormente sobre un único puerto TCP que suele ser el puerto 80 a diferencia de otros que pueden estar bloqueados en el sistema.

El envío de información a través de este protocolo suele ser casi en tiempo real a diferencia del envió de información con peticiones de HTTP. Además, mantiene la conexión entre el cliente y el servidor abierta en todo momento haciendo posible el envío de información del servidor al cliente antes comentado.[5]

4.3 Código QR

Un código QR o código de respuesta rápida es la evolución de los códigos de barras. En estos, a diferencia de sus predecesores, almacenan la información en una matriz de puntos y así se consigue poder guardar en estos todo tipo de información, ya sean identificadores, URLs, etc. [6]

Se pueden leer con cualquier dispositivo móvil actual que posea un escáner de QR devolviéndonos la información contenida en el mismo.

Figura 4: Ejemplo de Código QR

4.4 YOLO

En la actualidad, gran parte de los sistemas de detección de objetos reutilizan clasificadores de objetos para lograr esta tarea. Para poder detectar un objeto, estos sistemas utilizan un clasificador de objetos teniendo que evaluar la imagen en varias ocasiones y en distintas escalas. Algunos sistemas utilizan el método de ventana deslizante que va analizando la imagen en distintos tramos espaciales hasta que la recorre entera. Otros sistemas usan regiones propuestas en las que se generan unas zonas que potencialmente pueden contener imágenes, luego se analizan con un clasificador de objetos y por último se lleva a cabo un post procesado para refinar los cuadros delimitadores y eliminar repetidos. Todo ello los hace lentos y complejos a la hora de optimizar.

YOLO (You Only Look Once) [7] en cambio es más simple que los anteriormente comentados sistemas de detección de objetos. Primero, YOLO es rápido en comparación con otros, esto es debido a que no tiene un “pipeline” complejo, esto es gracias a que cuenta con una única red neural convolucional que se encarga de todos los pasos para detectar los objetos. Segundo, YOLO analiza la imagen de manera global a diferencia de la ventana deslizante o regiones propuestas, con lo que a la hora del entrenamiento o analizar la imagen cuenta con más información contextual sobre la imagen, lo que resulta de un mayor índice de acierto a la hora de detectar los objetos.

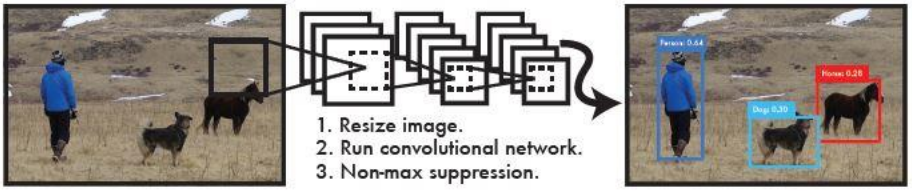

Figura 5: Funcionamiento simplificado de YOLO [7]

YOLO se unifica en una única red neuronal que cuenta con todas las herramientas necesarias para detectar los objetos. La red neuronal utiliza información de la imagen para predecir los cuadrados delimitadores, esto quiere decir que la red actúa de manera global sobre la imagen y las clases.

YOLO divide las imágenes en una cuadrícula de S x S. Si el centro de la imagen está contenido en una celda, esa celda será la encargada de detectar los objetos.

Cada celda predice un número B de cuadrados delimitadores con una confianza para estos. Esta confianza refleja cómo de seguro está el modelo sobre el contenido del cuadrado delimitador en esa celda. El cuadrado delimitador contiene un objeto y determina cómo de precisa es la predicción. La confianza se calcula mediante Pr(Object) * IOUtruth. Si no hay objeto contenido en la celda, la confianza será de 0. Se busca que la confianza sea igual a la intersección sobre la unión (IOU) entre el cuadrado delimitador predicho y el contenido real de la celda.

Cada cuadrado delimitador consiste en 5 predicciones x, y, w, h y confianza. Las coordenadas x e y representan los límites de la cuadrícula de la celda, el ancho y la altura representan los límites de la imagen y la confianza que será la IOU entre el cuadrado delimitador y el contenido real de la celda.

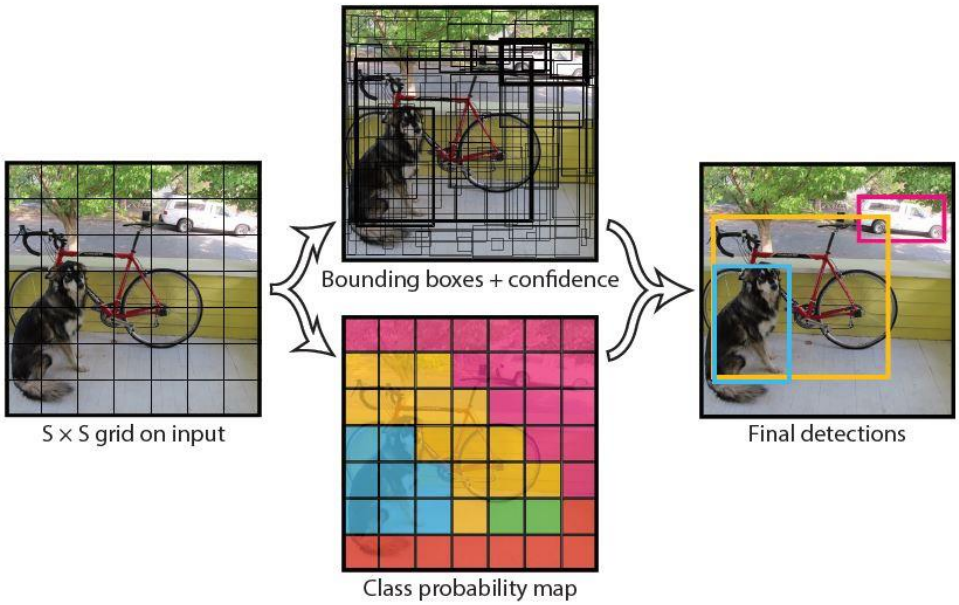

Figura 6: Detección de Objetos con YOLO [7]

Pero YOLO también tiene sus limitaciones a la hora de realizar la detección de objetos en determinadas circunstancias como cuando varios objetos pequeños aparecen en grupos.

En el proyecto se ha utilizado YOLOv4-tiny, que es la última versión de YOLO actualmente. Este difiere en ciertos aspectos sobre YOLO, pero su funcionamiento interno sigue siendo prácticamente el mismo con variaciones que buscan ante todo mejorar su velocidad de funcionamiento y su precisión a la hora de detectar objetos. Así como una mejora a la hora de entrenar modelos propios y su utilización. La diferencia fundamental entre YOLOv4 y YOLOv4-tiny es que la versión tiny sacrifica precisión a cambio de mayores tasas de fps. Se utilizo la verison tiny para el desarrollo de este proyecto porque se busca detectar los productos en tiempo real y para ello se necesita poder procesar una gran cantidad fps, además la perdida de precisión de la versión tiny con respecto a la versión normal no es lo suficientemente significativa como para que afecte a la hora de realizar las detecciones de los productos en este proyecto.

Con YOLOv4-tiny se puede lograr un modelo óptimo capaz de ejecutarse en GPUs convencionales, en las cuales se podrá obtener una detección en tiempo real de alta calidad con una mayor tasa de fotogramas.

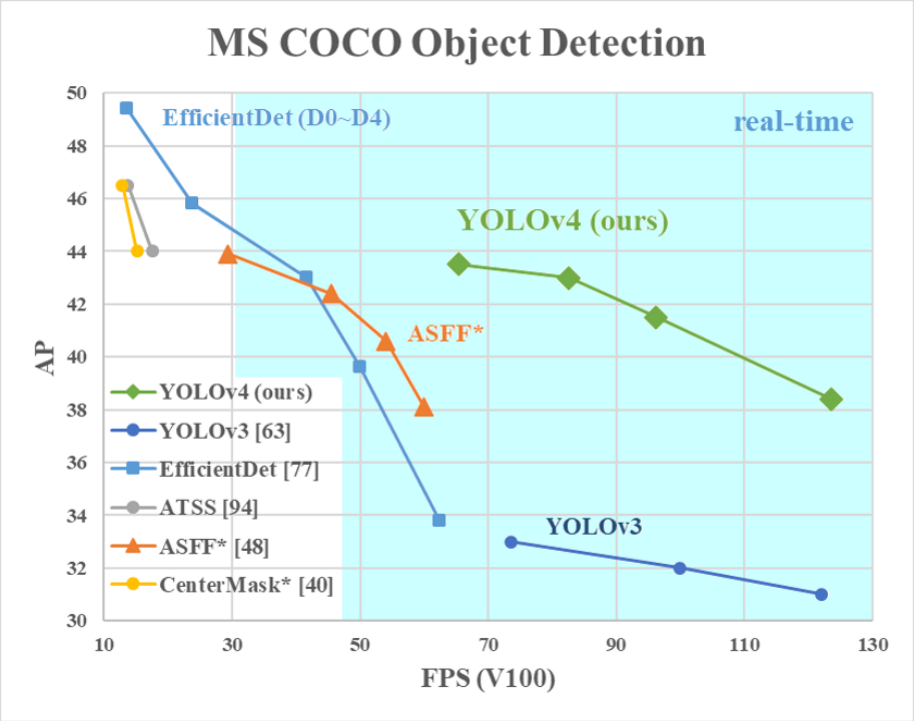

Figura 7: Comparación YOLOv4 con otros detectores de objetos [7]

Se ha entrenado un modelo de YOLOv4-tiny para la detección de los productos. Para poder llevar a cabo este entrenamiento se ha utilizado un modelo pre entrenado, al emplear un modelo pre entrenado conseguimos que la información de las capas intermedias no se pierda y el modelo resultante después del entrenamiento obtenga mejores resultados que si se crease un modelo nuevo con la misma información.

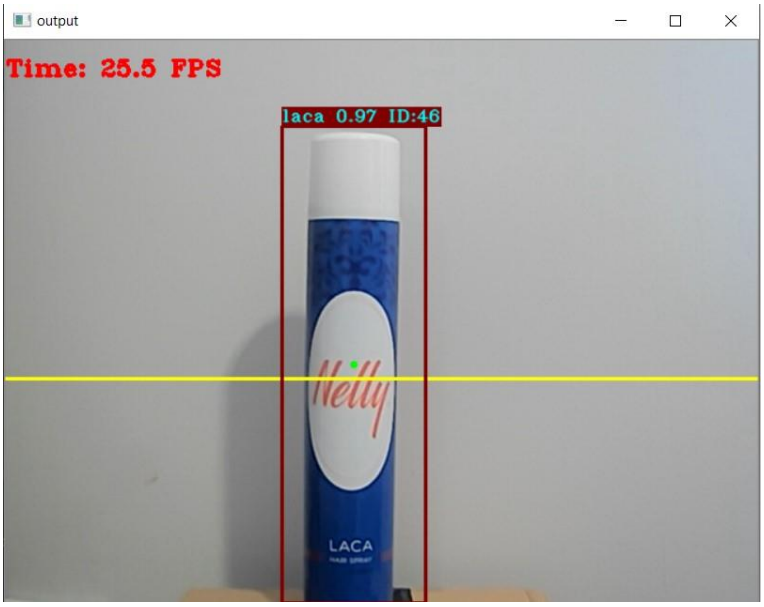

Figura 8: Ejemplo Detección con el modelo ya entrenado

En la *Figura 8*, se puede observar el funcionamiento del modelo que se ha entrenado. Siendo así este capaz de detectar con una confianza del 97% una laca en tiempo real a través de una webcam.

4.5 DeepSORT

Para realizar el seguimiento de los objetos detectados por YOLO, se utilizará el algoritmo de seguimiento DeepSORT [8]. El objetivo de los algoritmos de seguimiento es el de ser capaces de seguir la posición de un objeto en un vídeo cada fotograma asignado les un cuadrado delimitador e identificador a los objetos detectados que les seguirá hasta que salgan del campo de visión del video.

DeepSORT no solo será capaz de realizar el seguimiento de objetos si YOLO no es capaz de detectar un objeto en un cuadrado delimitador, dando lugar esto a cambios en los identificadores de los objetos de los que se está haciendo el seguimiento.

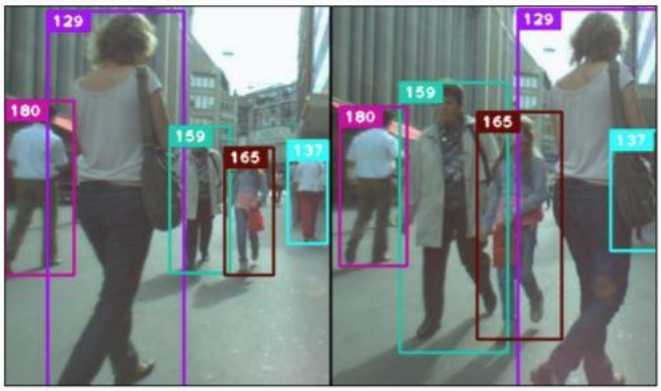

Figura 9: Ejemplo de algoritmo de seguimiento [8]

DeepSORT es una mejora del algoritmo de seguimiento SORT (Simple Online and Realtime Tracking) que es un acercamiento al seguimiento de objetos múltiple con enfoque en la simpleza y eficacia de algoritmos. Sin embargo, su eficiencia a la hora de realizar el seguimiento de objetos se ve reducida debido a la oclusión o cambios del punto de vista de la cámara. DeepSORT consigue un porcentaje de mejora sobrea la pérdida del objeto en seguimiento e intercambio de identificadores del 45%.

El algoritmo de DeepSORT basa su funcionamiento en los siguientes componentes:

- **Estimador de Seguimiento:** continua utilizando al igual que el SORT el filtro de Kalman que definirá un espacio de estados de ocho dimensiones (x,y,z,h,x´,y´,z´,h´). (x, y) contienen la posición central del cuadrado delimitador, z representa la relación de aspecto, la h representa la altura. Al utilizar el filtro de Kalman se toman los estados (x,y,z,h) de la observación del objeto. Los estados (x´,y´,z´,h´) serán una predicción sobre donde se encontrara el objeto en el siguiente frame. El estimador de seguimiento es la métrica de localización que utiliza la intersección entre la posición del cuadrado delimitador (x,y,z,h) y el cuadrado delimitador estimado (x´,y´,z´,h´) para obtenerla.
- **Descriptor de apariencia:** es un componente nuevo que utiliza el algoritmo de DeepSORT. La apariencia es información sobre las características de un objeto que son extraídas de la imagen y diferirán de las características de otros objetos. Esto ayuda a distinguir a unos objetos de otros a la hora de la estimación. El descriptor de apariencia proporciona la denominada métrica de la apariencia.
- **Información asociada:** cada cuadrado delimitador tiene información asociada al seguimiento que se está realizando gracias a la métrica de localización y a la métrica de apariencia. Cada objeto en seguimiento tendrá un identificador propio.
- **Manejo de tareas:** si se detecta un nuevo cuadrado delimitador y no se puede asociar con ningún objeto en seguimiento actual este se pondrá en modo seguimiento tentativo. Si pasados unos fotogramas no se ha conseguido asociar el seguimiento tentativo con ningún seguimiento de un objeto se eliminará este seguimiento, si en cambio se consigue asociar se actualizará el seguimiento quitándolo del modo tentativo.

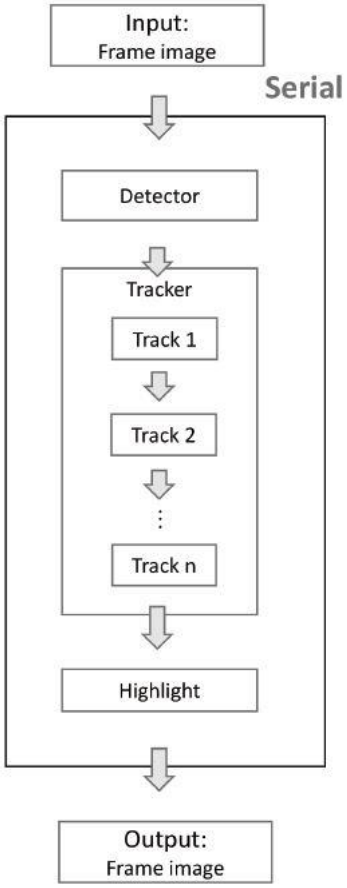

Figura 10: Funcionamiento YOLO y DeepSORT [8]

En la *Figura 10*, se puede observar la arquitectura que se usara a la hora de utilizar conjuntamente YOLO y DeepSORT.

5. Técnicas y herramientas

En esta sección se detallarán las técnicas y herramientas que se han utilizado para llevar a cabo el proyecto.

Se dividirá en 4 apartados:

- Aplicación Web
- Aplicación Móvil
- Servidor Api
- Carro
- Herramientas Case

5.1 Aplicación Web

5.1.1 HTML

Hyper Text Markup Language o HTML es un lenguaje de marcado para el desarrollo de páginas web. Este es un estándar, el cual sirve de referencia del software que conecta con la elaboración de páginas web.

Se ha impuesto como el estándar en internet para la creación y muestra de páginas web. Debido a que ofrece una gran adaptabilidad, estructura lógica y es fácil de interpretar por humanos y máquinas.

Se encarga de definir la estructura y el código básico de una página web, para ello utiliza “etiquetas” las cuales están rodeadas por dos corchetes (< >).[9]

5.1.2 CSS

Cascading Style Sheets o CSS es un lenguaje de diseño con el que se puede definir y crear la presentación de un documento escrito en algún lenguaje marcado como por ejemplo HTML. Se utiliza sobre todo para crear páginas web atractivas dotando de estilo el código de HTML.[10]

5.1.3 JavaScript

Es un lenguaje de programación interpretado, el cual se utiliza habitualmente del lado del cliente, interpretado por los navegadores para conseguir páginas web más interactivas y dinámicas. Se define como un lenguaje orientado a objetos, basado en prototipos e imperativos, el cual es dinámico y esta débilmente tipado.

Es el único lenguaje de programación que entienden los navegadores por lo que se utiliza junto a HTML y CSS para la creación de la mayoría de las páginas web de internet. [11]

5.1.4 Vue.js

Es un marco de trabajo progresivo que utiliza JavaScript para la creación de interfaces de usuario. Se basa en una arquitectura MVVM, centrándose sobre todo en el Modelo de Vista conectando la Vista con Modelo mediante enlace de datos bidireccionales haciendo que cualquier cambio producido en el Modelo se propague a la Vista.

Este trabaja con componentes. Estos componentes se corresponden con un código encapsulado reutilizable. Dentro de cada componente se pueden encontrar HTML, CSS y JavaScript. Mediante el uso de componentes se podrá desarrollar de manera más fácil y rápida una aplicación. [12]

Figura 11: Código ejemplo Vue js

En la *Figura 11*, podemos ver un ejemplo de un componente de Vue.js además de distintas etiquetas en este:

- **Template:** lo que se encuentra dentro de estas etiquetas se representará mediante código HTML
- **Script:** aquí se encontrará contenido el código JavaScript del componente.
- **Style**: se encontrarán los códigos CSS.

5.1.5 Vuetify

Es una librería de Vue que se instala sobre Vue.js y ayuda a desarrollar de manera más rápida y simple interfaces sobre Vue. [13]

5.1.6 Vue Router

Es la librería oficial de router para vue.js, la cual nos permite dirigir a los usuarios por la aplicación de una forma más simple y ordenada. [14]

5.1.7 Axios

Es una librería de JavaScript, la cual nos permite hacer peticiones HTTP al servidor. [15]

5.1.8 Firebase

Es una plataforma en la nube propiedad de Google para el desarrollo de aplicaciones web y móvil. Su función es la de facilitar las tareas de creación de aplicaciones móviles y web. [16]

Nos ofrece una serie de herramientas para el desarrollo de aplicaciones, las utilizadas en este proyecto son:

- **Autentificación de usuarios:** nos ofrece un sistema tanto de autentificación como de registro de usuarios.
- **Firestore Database:** nos ofrece una base de datos no relacional para el almacenamiento y consulta de información. Se utilizará para almacenar la información de los usuarios.

5.2 Aplicación Móvil

En la aplicación móvil se han usado herramientas que ya se han comentado en el apartado anterior como:

- **Axios**
- **Router**
- **JavaScript**
- **Firebase**

5.2.1 React Native

Es un marco de trabajo de JavaScript para el desarrollo de aplicaciones nativas para Android o IOS, es decir, se crea una aplicación en JavaScript que se ejecutará en el dispositivo como si fuera Object-C o java. Esto facilita el desarrollo de las aplicaciones móviles al poder desarrollar una aplicación en un único lenguaje que luego se ejecutará de forma nativa en el dispositivo que la esté efectuando.

Los cambios producidos en el código se mostrarán por pantalla, por lo que no hará falta compilaciones lentas, solo con guardar se actualizará la aplicación, facilitando así la vista de los cambios que realicen.

Se construye sobre React Js el cuál es una biblioteca de JavaScript para el desarrollo de interfaces. [17]

5.2.2 QRCodeScanner

Es una librería de react native que nos permite escanear códigos QR y obtener el resultado de estos. [18]

5.2.3 Socket.io

Es una librería de JavaScript que permite la transmisión de datos en tiempo real de forma bidireccional mediante eventos entre el cliente y el servidor. Se intentará establecer una conexión WebSocket con el servidor para la transferencia de información entre ambos. Cuenta con dos partes separadas, una librería para la parte de los clientes y otra librería para la parte del lado del servidor.

Gracias a esta librería se consigue multidifusión para un grupo de clientes distintos, almacenamiento de información de cada cliente que se conecte y emisiones asíncronas permitiendo continuar la ejecución del programa sin tener que esperar una respuesta del servidor o cliente. [19]

En este caso se usará la librería de la parte de los clientes.

5.3 Servidor API

En el servidor API se han utilizado herramientas que ya se han comentado en los apartados anteriores como:

- **Socket.io:** con la diferencia que esta será la librería del lado del servidor y no la del cliente como en el caso de la aplicación móvil.
- **JavaScript**
- **Firebase**
- **Router**

5.3.1 Node.js

Es un entorno en tiempo de ejecución multiplataforma para la capa del servidor desarrollado en JavaScript. Se basa en el funcionamiento en un único hilo de ejecución en un loop diferenciándolo de otros servidores que crean un hilo para satisfacer las peticiones entrantes, además utiliza un modelo asíncrono de entradas y salidas de información con una arquitectura orientada a eventos. Todo esto mejora el rendimiento y escalabilidad de este permitiendo comunicaciones en tiempo real con el servidor. [20]

5.3.2 Express

Es un marco de trabajo de aplicaciones web para node.js. Se ha convertido en el marco de trabajo estándar para node.js. Este ayuda a desarrollar aplicaciones web y APIS de manera más rápida y fácil. [21]

Nos permite el uso de diferentes middlewares para nuestra API, en el caso de este proyecto se ha utilizado:

- **Cors:** permite habilitar CORS (Cross-Origin Resource Sharing).

5.3.3 MariaDB

Es una de las más populares bases de datos relacional de código abierto. Su velocidad es una de las más importantes características, pudiendo almacenar a la vez cientos de tablas con todo tipo de información contenida en estas. Se basa en SQL para funcionar que es un lenguaje de programación estandarizado utilizado para manejar bases de datos relacionales. [22]

Se utilizará para almacenar la información de los productos, ventas y carros.

5.3.4 mysql

Es un paquete de node que nos permite conectarnos a la base de datos relacional desde el servidor de node.js, habilitando con esto la consulta para guardar, modificar o solicitar información de la base de datos relacional. Este paquete de node comparte el nombre con el gestor de bases relacionales MySQL. [23]

5.4 Carro

En el carro se han usado herramientas que ya se han comentado en los apartados anteriores como:

- **Socket.io**

5.4.1 Python

Es un lenguaje de programación interpretado de alto nivel, multi-paradigma y orientado parcialmente a objetos. [24]

5.4.2 LabelImg

Es una herramienta gráfica de etiquetado de imágenes de código abierto, la cual está escrita en Python. [25]

5.4.3 Roboflow

Es una plataforma Online de servicios de software para permitir una visión por computador más rápida y sencilla. Permitiendo así gestionar tus imágenes, etiquetas, preprocesamiento y aumento. [26]

Para la realización del proyecto solo se ha utilizado la opción de aumento.

5.5 Herramientas Case

5.5.1 Visual Studio Code

Es un editor de código fuente desarrollado por Microsoft, este se basa en el marco de trabajo Electron. [27]

5.5.2 Pycharm

Es un entorno de desarrollo utilizado para la creación de códigos en Python. [28]

5.5.3 Npm

Es un sistema de gestión de archivos que consiste en una base de datos online que puede ser llamada a través de npm para la instalación o actualización automática de paquetes. [29]

5.5.4 Postman

Es una herramienta para la creación y diseño de APIs, permitiendo crear peticiones de manera simple para la creación y prueba de la API. [30]

5.5.5 Microsoft Project

Es un software de administración de proyectos utilizado para el desarrollo de planes, asignación de recursos, etc. [31]

Se ha utilizado para la creación de la planificación temporal.

5.5.6 EZEstimate

Es un programa que nos permite realizar estimaciones de esfuerzo de un determinado proyecto. [32]

5.5.7 Draw.io

Es una herramienta utilizada para la creación de diagramas. [33]

5.5.8 Git

Es un software de control de versiones, creado para el manejo y almacenamiento de las diferentes versiones de un proyecto. [34]

5.5.9 JSDoc

Es una herramienta utilizada para la generación de documentación en el lenguaje de programación JavaScript.[35]

5.5.10 Vue Styleguidist

Es una herramienta utilizada para la generación de documentación de los componentes vue.[36]

5.5.11 Sphinx

Es una herramienta utilizada para la generación de documentación en el lenguaje de programación Python.[37]

6. Aspectos relevantes del desarrollo

En esta sección se detallarán los aspectos más relevantes a la hora de desarrollar el proyecto.

6.1 Marco de trabajo

Para el desarrollo del proyecto se ha seguido el Proceso Unificado como marco de trabajo. [38] Se centra en una serie de características:

- **Iterativo e incremental:** es un marco iterativo e incremental compuesto por diferentes fases. Cade fase está dividida a su vez en un numero de iteraciones, que a su vez se divide en una serie de disciplinas.
- **Dirigido por los casos de uso:** los casos de uso capturan los requisitos funcionales y definen las acciones que se realizan en cada una de las diferentes iteraciones del proceso.
- **Centrado en la arquitectura:** no existe un modelo único que cubra todos los aspectos del sistema. Por tanto, se asume que existen varios modelos y vistas que definirán la arquitectura
- **Enfocado en los riesgos:** se centrará en la identificación de los riesgos en las etapas tempranas del ciclo de vida.

Otra característica importante del Proceso Unificado es que utiliza el Lenguaje Unificado de Modelado (UML).

Como ya se ha comentado, el Proceso Unificado cuenta con una serie de fases que se dividen a su vez en interacciones, estas fases son:

- **Inicio:** se definirá el alcance del proyecto.
- **Elaboración:** se obtiene una idea ya definida de lo que se debe realizar para la creación del proyecto.
- **Construcción:** se llevará a cabo la construcción del proyecto.
- **Transición:** la fase final en la que se suelen realizar pruebas del proyecto completas.

Las disciplinas principales en las que este se dividen son:

- **Modelo de negocio**
- **Requisitos**
- **Análisis**
- **Diseño**
- **Implementación**
- **Pruebas**

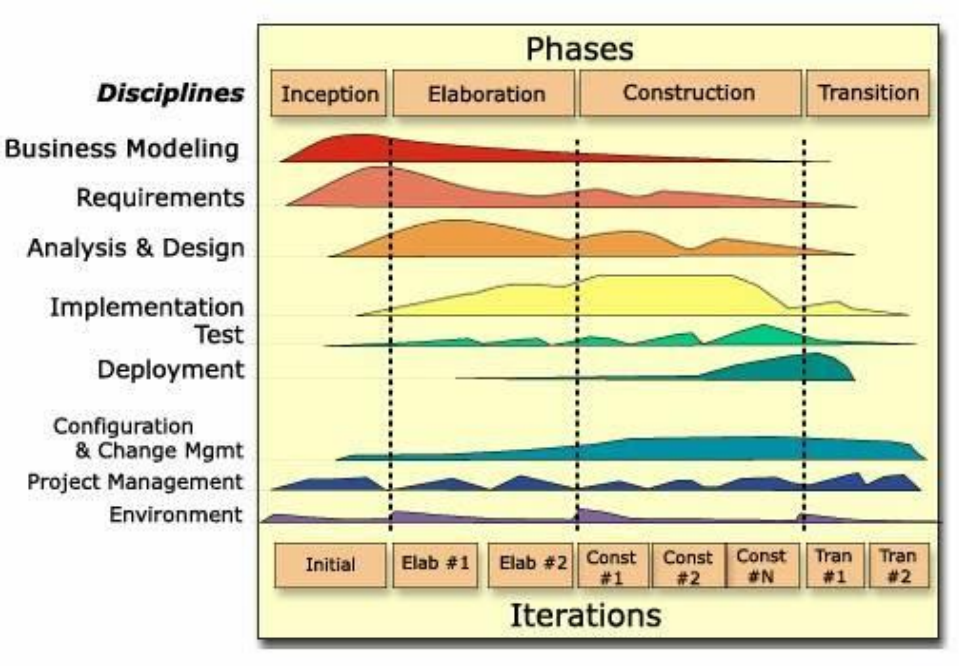

Figura 12: Proceso Unificado

En la *Figura 12*, se pueden observar las diferentes fases, iteraciones y divisiones del Proceso Unificado que se han comentado con anterioridad.

6.2 Estimación del esfuerzo

Se detallará una estimación del esfuerzo requerido para el desarrollo del proyecto software. Para ello se ha utilizado la herramienta EZEstimate que le introduciremos unos valores y nos calculará automáticamente la estimación del esfuerzo. [39]

Para obtener más información sobre la estimación del esfuerzo se puede consultar el anexo I – Plan de Proyecto Software.

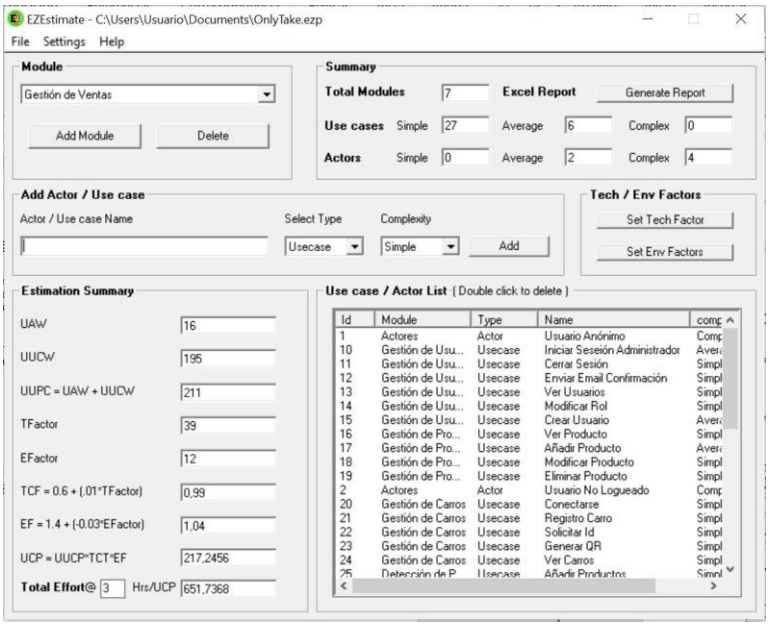

Figura 13: EZEstimate

En la *Figura 13*, se puede observar la estimación del esfuerzo para este proyecto obtenido por EZEstimate.

6.3 Planificación temporal

Se procederá a calcular la planificación temporal del tiempo que requerirá desarrollar el proyecto. Esta planificación se hace siguiendo el esquema del Proceso Unificado. [39]

Para llevar a cabo esta planificación se ha utilizado la herramienta Microsoft Proyect.

La planificación se dividirá en las siguientes fases:

- **Inicio**
- **Elaboración**
- **Construcción**
- **Transición**

Estas fases a su vez se dividen en diferentes iteraciones:

- **Inicio:** 1 iteración
- **Elaboración:** 2 iteraciones
- **Construcción:** 3 iteraciones
- **Transición:** 1 iteración

Las disciplinas principales en las que este se dividen las iteraciones son:

- **Modelo de negocio**
- **Requisitos**
- **Análisis**
- **Diseño**

- **Implementación**
- **Pruebas**

Para obtener más información sobre la planificación temporal se puede consultar el anexo I – Plan de Proyecto Software.

	Inicio	**9 días**	**lun 22/02/21**	**jue 04/03/21**	
	Fin Inicio	0 días	jue 04/03/21	jue 04/03/21	1
	Elaboración	**24,7 días**	**vie 05/03/21**	**jue 08/04/21**	**14**
	Fin Elaboración	0 días	jue 08/04/21	jue 08/04/21	15
	Construcción	**41,3 días**	**jue 08/04/21**	**vie 04/06/21**	**50**
	Fin Implementación	0 días	vie 04/06/21	vie 04/06/21	51
	Transición	**8 días**	**lun 07/06/21**	**mié 16/06/21**	**88**
	Fin Transición	0 días	mié 16/06/21	mié 16/06/21	89

Figura 14: Resultados Planificación temporal

En la *Figura 14*, se pueden observar los tiempos que durarán las diferentes fases para el desarrollo del proyecto.

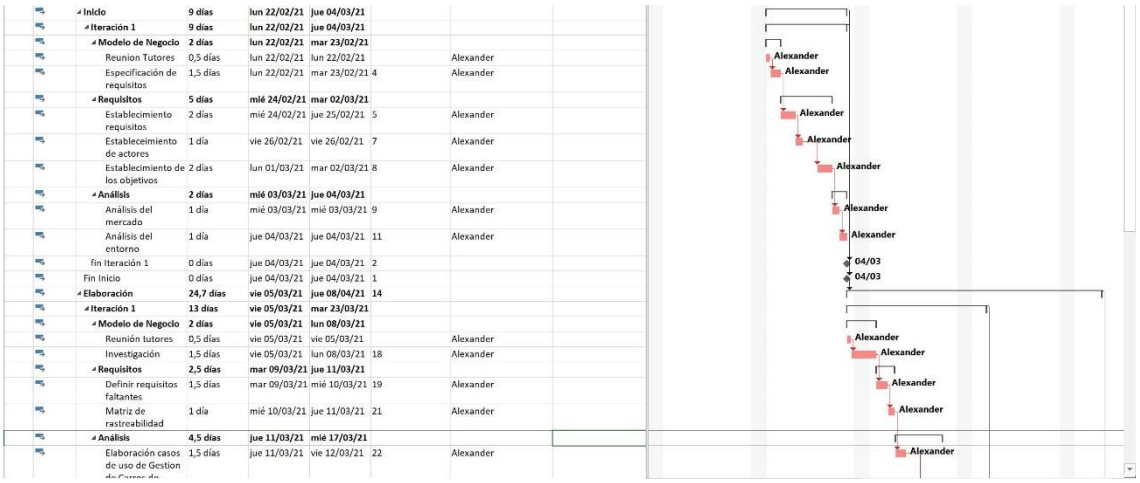

Figura 15: Camino Crítico

En la *Figura 15*, se puede observar el diagrama parte del Gantt, así como el camino crítico del proyecto.

6.4 Especificación de requisitos software

Este apartado tiene como objetivo documentar la especificación de requisitos software del sistema a desarrollar. Por tanto, se recogerán los participantes, objetivos del sistema y el catálogo de requisitos. [40], [41], [42]

Para obtener más información sobre la especificación de requisitos software se puede consultar el anexo II – Especificación de Requisitos Software.

A continuación, se comentarán las diferentes secciones de la especificación de requisitos software.

6.4.1 Participantes

El proyecto cuenta con cuatro participantes, de los cuales tres son tutores y uno el alumno. Todos ellos pertenecen a la misma organización, la Universidad de Salamanca.

Los participantes serán:

- **Alexander Fuentes Bartolomé**
- **Luis Augusto Silva**
- **Juan Francisco de Paz Santana**
- **Gabriel Villarrubia González**

6.4.2 Objetivos del sistema

Los objetivos del sistema son:

- **Detección de Productos**
- **Gestión de Usuarios**
- **Gestión de Productos**
- **Gestión de Carros**
- **Gestión de Ventas**

6.4.3 Requisitos de información

Será la información que el sistema deberá almacenar.

Los requisitos de información del sistema son:

- **Información Usuario**
- **Información Producto**
- **Información Carro**
- **Información Carrito**
- **Información Venta**

6.4.4 Requisitos funcionales

Los requisitos funcionales del sistema nos indicarán cómo se debe comportar el sistema al interactuar con los actores.

Los diferentes actores que interactuarán con el sistema creado serán:

- **Usuario Anónimo**
- **Usuario No Logueado**
- **<<Device>> Carro**
- **<<System>> Gateway**
- **Usuario**
- **Administrador**

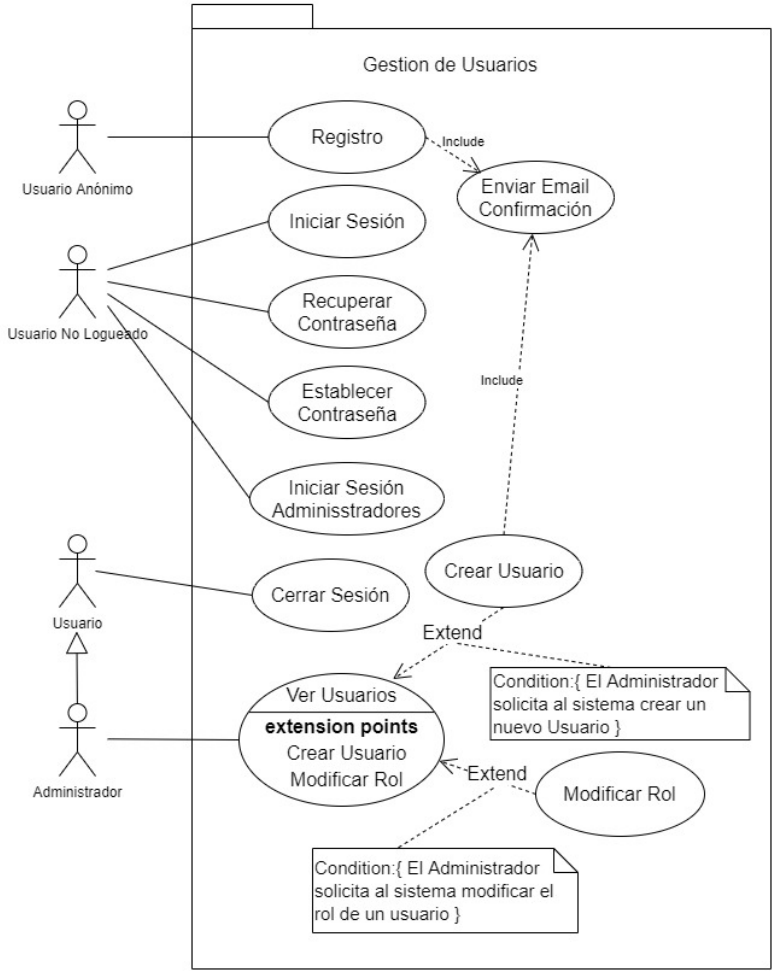

Figura 16: Diagrama de Casos de Uso del paquete Gestión de Usuarios

En la *Figura 16*, se puede observar un ejemplo de un paquete de casos de uso.

UC-001	Registro	
Versión	1.1 (05/06/2021)	
Autores	• Alexander Fuentes Bartolomé	
Fuentes	• Luis Augusto Silva • Juan Francisco de Paz Santana • Gabriel Villarrubia González	
Dependencias	[OBJ-002] Gestión de Usuarios [IRQ-001] Información Usuario	
Descripción	El sistema deberá comportarse como se describe en el siguiente caso de uso cuando un Usuario Anónimo desee registrarse en el sistema.	
Precondición	El Usuario Anónimo no está registrado en el sistema.	
Secuencia normal	**Paso**	**Acción**
	1	El actor Usuario Anónimo (ACT-001) solicita registrarse en el sistema.
	2	El sistema solicita al usuario sus datos para añadirle al sistema.
	3	El actor Usuario Anónimo (ACT-001) introduce sus datos de acceso.
	4	El sistema almacena los datos del nuevo usuario en la base de datos.
Postcondición	El usuario accede al sistema.	
Excepciones	**Paso**	**Acción**
	3	Si el usuario solicita al sistema cancelar el registro, el sistema lo cancela, a continuación, este caso de uso queda sin efecto.
	4	Si alguno de los datos introducidos es incorrecto o ya existe un usuario con ese correo electrónico registrado, se le mostrará un error al usuario, a continuación, este caso de uso quedará sin efecto.
Rendimiento	**Paso**	**Tiempo Máximo**
	-	-
Frecuencia esperada	Ninguno	
Importancia		
Urgencia	Inmediatamente	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Tabla 1: UC-001 Registro

En la *Tabla 1*, se puede observar un ejemplo de cómo se ve una tabla de los casos de uso.

6.4.5 Requisitos funcionales

Los requisitos no funcionales son aquellos requisitos que están relacionados directamente con la funcionalidad del propio sistema y suelen ser más relevantes que los requisitos funcionales para el correcto funcionamiento del sistema.

Los requisitos no funcionales que se tienen son:

- **Tiempo de respuesta**
- **Seguridad de los datos**
- **Concurrencias**
- **Usabilidad**
- **Compatibilidad**
- **Implementación Móvil**
- **Implementación Web**
- **Seguridad**

6.5 Análisis del sistema software

Ya se han especificado los requisitos y la funcionalidad que debe cumplir el sistema. En este apartado se llevará a cabo la etapa de análisis del sistema, que consistirá en un análisis exhaustivo de los requisitos del sistema.[42]

Para obtener más información sobre el análisis de requisitos software se puede consultar el anexo III – Especificación de Requisitos Software.

A continuación, se comentarán las diferentes secciones del análisis de requisitos.

6.5.1 Modelo de dominio

A partir de los requisitos se han obtenido una serie de clases conceptuales del mundo real, las cuales están organizadas en el modelo de dominio de la *Figura 17* que representan todas las relaciones que deberemos mantener en la memoria para preservar los requisitos de información.

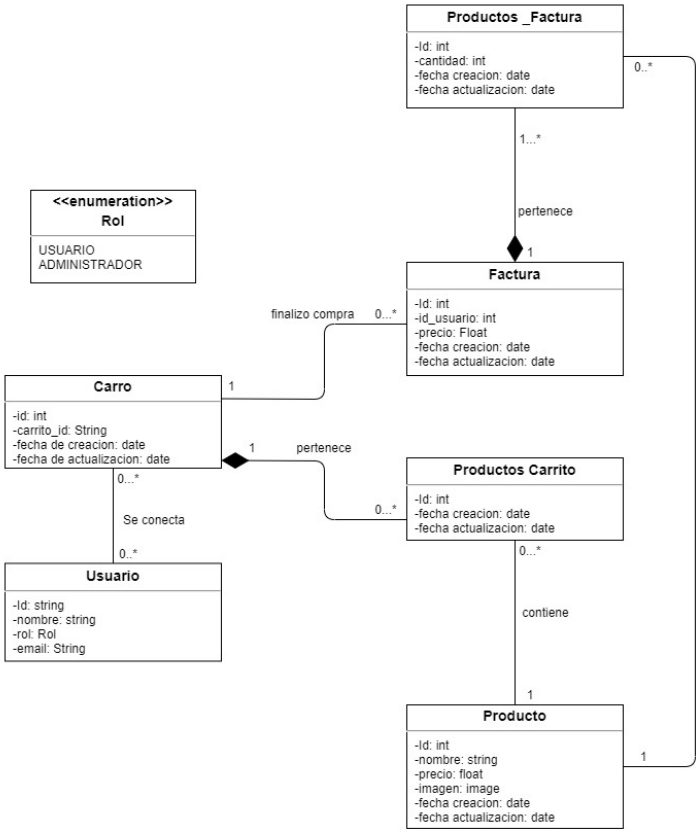

Figura 17: Modelo de Dominio

6.5.2 Realización de caso de uso en el análisis

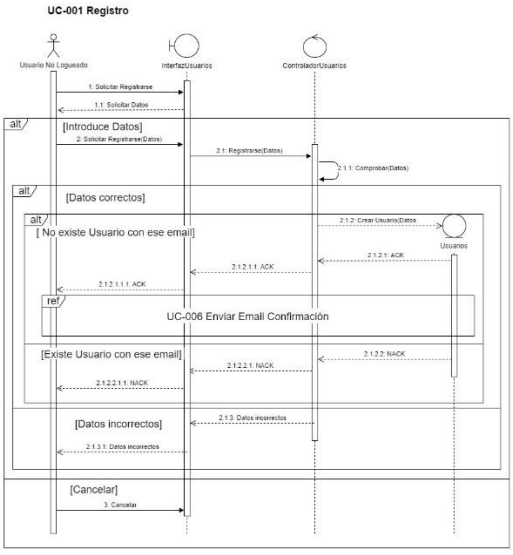

Figura 18: UC-001 Registro

En la *Figura 18*, se puede ver un ejemplo de cómo se han realizado los casos de uso en el análisis mediante diagramas de secuencia.

6.6 Diseño del sistema software

Este apartado tiene como objetivo documentar la fase de diseño del sistema a desarrollar. La fase de diseño se basa en el dominio de la solución por lo que a la hora de realizarla esta se acerca ya bastante a la implementación real del sistema a desarrollar. [43]

Para obtener más información sobre el análisis de requisitos software se puede consultar el anexo IV – Diseño del Sistema Software.

A continuación, se comentarán las diferentes secciones del diseño del sistema software.

6.6.1 Patrones arquitectónicos y de diseño

6.6.1.1 Patrón de Capas

En el patrón de capas, las capas solo se pueden comunicar con la capa de nivel inferior a esta. Con esta separación conseguimos que cada capa se encargue de una tarea bien definida haciendo más fácil el desarrollo y la implementación. Este será el patrón arquitectónico que seguirá el sistema.

Las capas que lo conforman se encargan de:

- **Capa de Presentación:** se encarga de presentar la información, así como de recoger las acciones realizadas por los usuarios.
- **Capa de Negocio:** en esta capa se encuentra la funcionalidad de nuestro sistema.
- **Capa de Acceso a Datos:** esta capa se encarga de acceder y almacenar los datos del sistema.

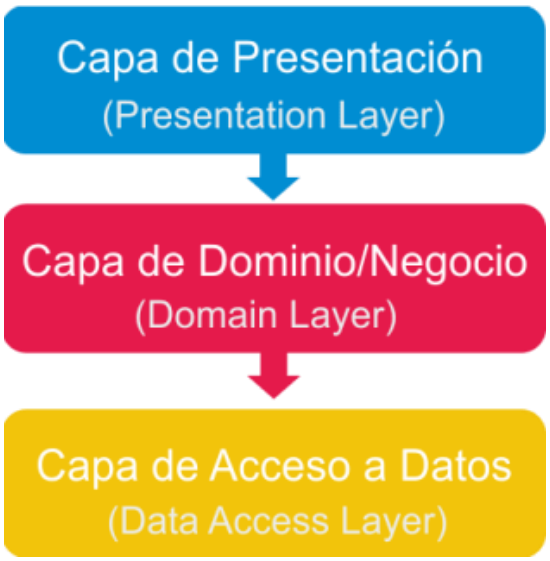

Figura 19: Patrón de Capas

6.6.1.2 Modelo Vista Modelo de Vista

El modelo vista modelo de vista es una variante del patrón arquitectónico modelo vista controlador (MVC), pero en este el controlador se sustituye por ViewModel que busca separar la interfaz de la capa de negocio.

Se encuentra formado por:

- **Modelo:** como en el patrón MVC, representará el modelo de los datos, excepto los métodos o servicios que los manipulan.
- **Vista de Modelo:** es la capa que se encuentra entre el modelo y la vista. Comunica ambas capas de forma automática y bidireccional. En esta se encuentra la lógica de negocio del sistema.
- **Vista:** se encarga de presentar a través de la interfaz la información del modelo, así como de capturar las acciones que realicen los usuarios. La vista en este patrón es más activa que en otros pudiendo llegar a tener eventos, comportamientos y enlaces de datos.

two way data bindings: se encarga de conectar el modelo y la vista, de modo que cualquier cambio producido en el modelo también producirá un cambio en la vista.

Este patrón se sigue a la hora de desarrollar la aplicación web en Vue.js.

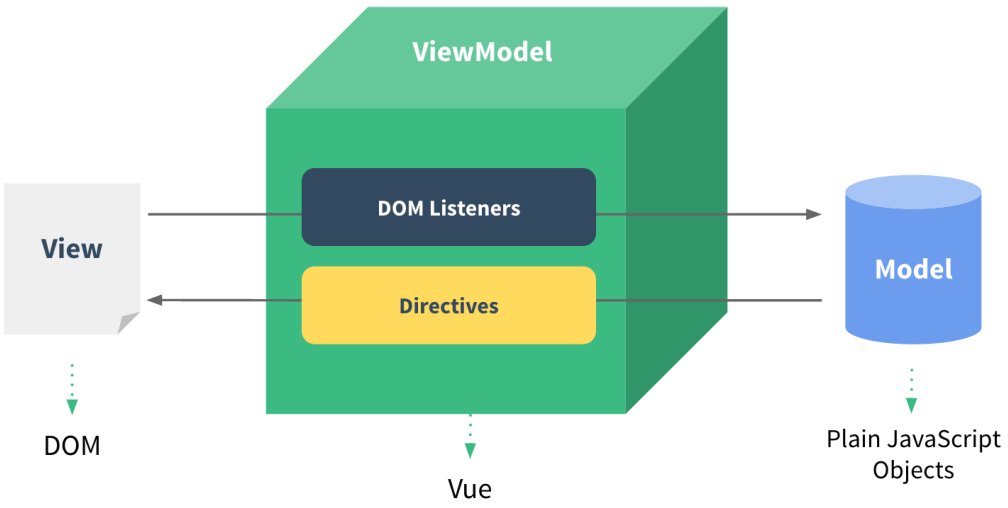

Figura 20: Patrón Modelo Vista Modelo de Vista

6.6.1.3 Patrón DAO

El patrón DAO se encarga de desacoplar la lógica de acceso a la base de datos de la lógica de negocio, consiguiendo así que la lógica de acceso a la base de datos se encuentre encapsulada para el resto de nuestro sistema. Con esto se consigue que las operaciones sobre los datos estén ocultas para el resto de la aplicación.

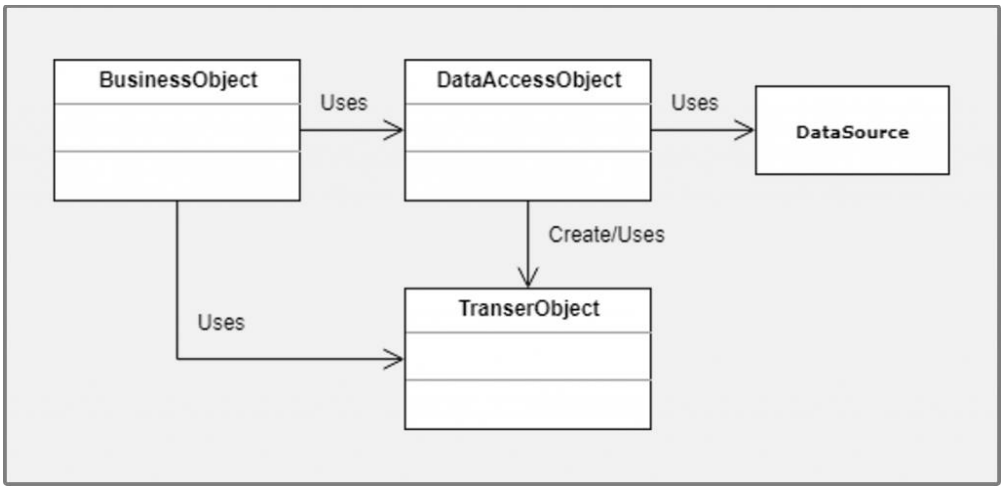

Figura 21: Patrón DAO

6.6.1.4 Publish-Subscribe

El patrón publish-subscribe, es un patrón de mensajería, en el cual los Publisher envían mensajes a un canal bróker que se encarga de entregar dichos mensajes a los receptores que se hayan suscrito al canal. Así vemos que los Publisher no tienen ningún tipo de conocimiento de los subscribers y viceversa.

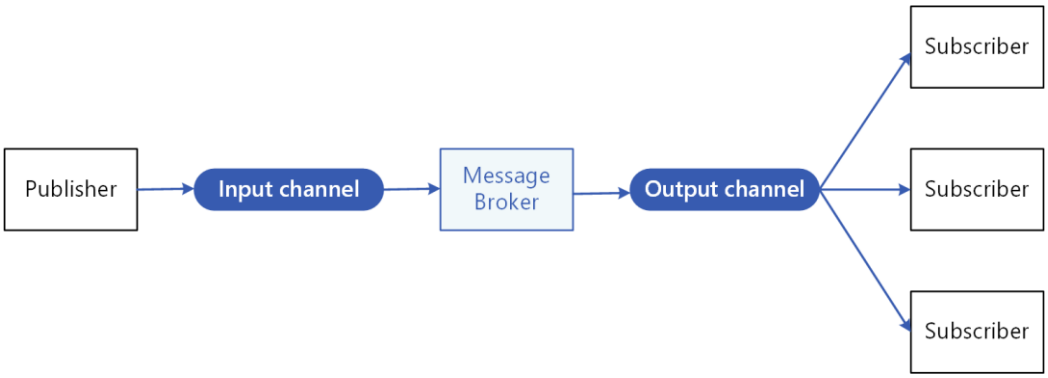

Figura 22: Patrón Publish-Subscribe

6.6.2 Subsistemas de diseño

Se realiza la descomposición del sistema en paquetes de subsistemas para que sean más fáciles de manejar.

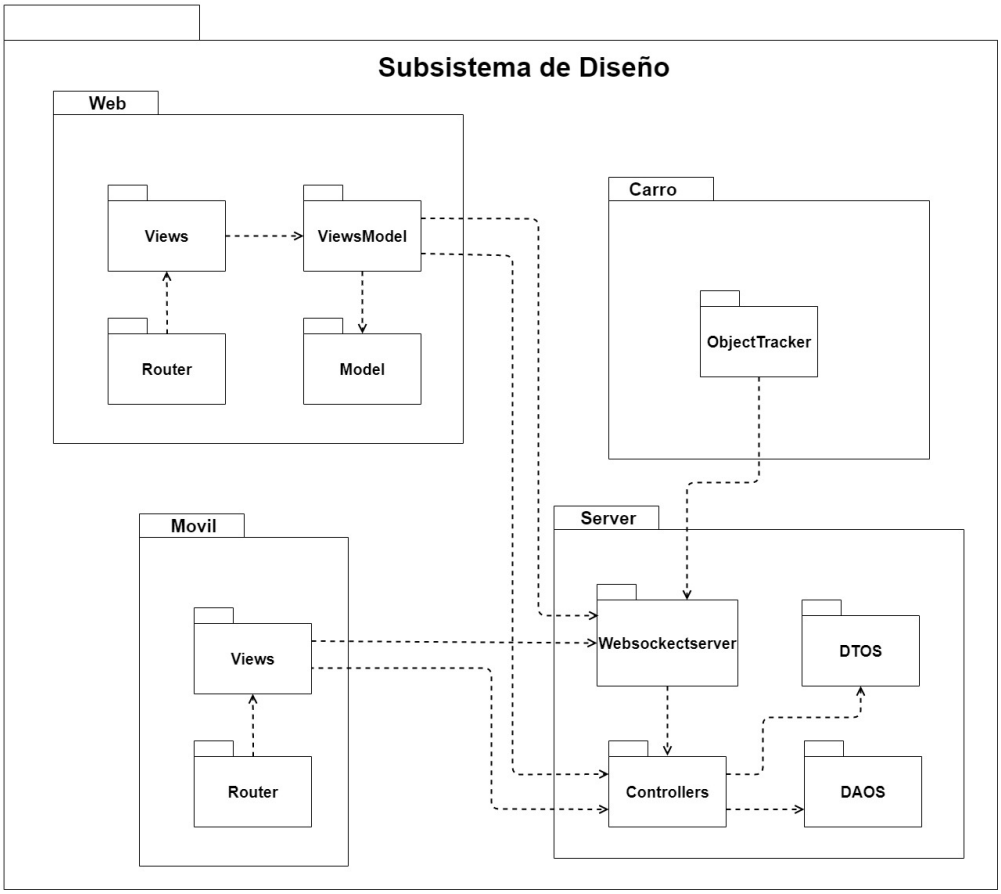

Figura 23: Subsistemas de diseño

Descripción de los subsistemas:

- **Server:** subsistema encargado de procesar las peticiones de los usuarios y acceder a la base de datos.
- **Móvil:** se encarga de dar acceso a los usuarios a nuestro sistema.
- **Web:** se encarga de dar acceso a los administradores a nuestro sistema.
- **Carro:** se encarga de detectar los productos salientes y entrantes al carro de la compra y comunicárselos al servidor.

En la *Figura 23*, se puede observar los diferentes paquetes de subsistemas de diseño del sistema.

6.6.3 Clases de diseño

Se detalla el contenido de los distintos paquetes de los subsistemas de diseño.

6.6.3.1 Subsistema Web

Views

Contiene las distintas vistas de la aplicación web, cada una se corresponde con un fichero.

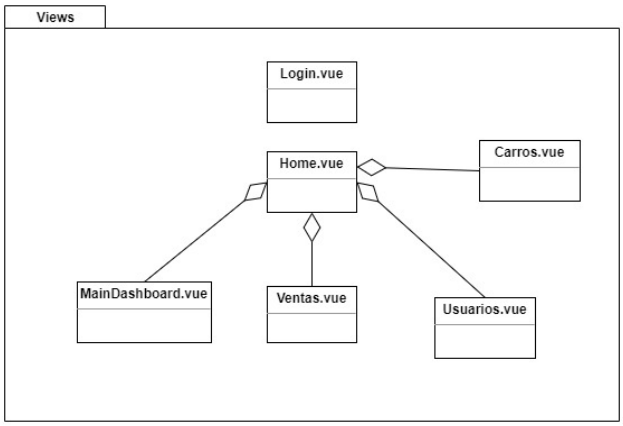

Figura 24: Views

ViewModel

Contiene los métodos encargados de gestionar el flujo de datos entre la vista y el modelo.

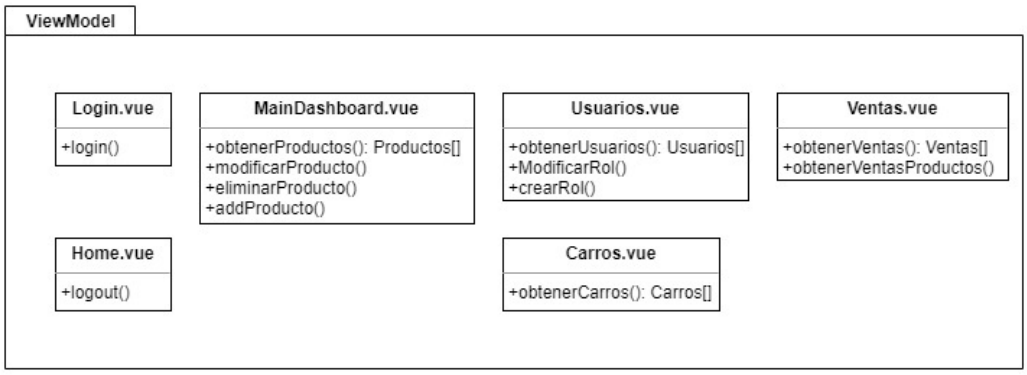

Figura 25: ViewModel

Model

Contiene el modelo que usará la web. Al estar utilizando el MVVM, el modelo de la web serán simples atributos que usará el ViewModel.

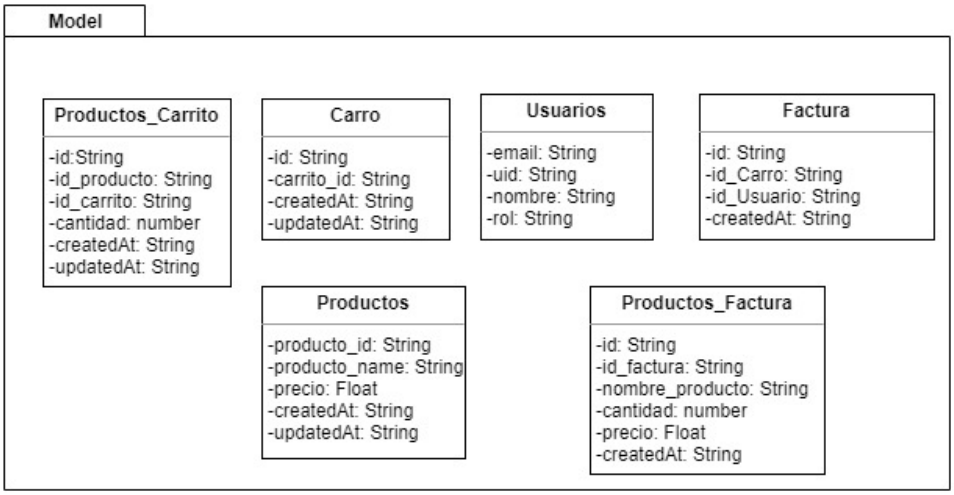

Figura 26: Model

Router

Se encarga de dirigir al usuario dentro de la aplicación, además de comprobar si el usuario está autenticado.

6.6.3.2 Subsistema Móvil

Screens

Contiene las pantallas además de toda la información necesaria para el funcionamiento de estas. Esto es así debido a que react Native es una capa de abstracción entre la Api nativa y el código de react, no teniendo así ni modelo, ni vista ni controlador.

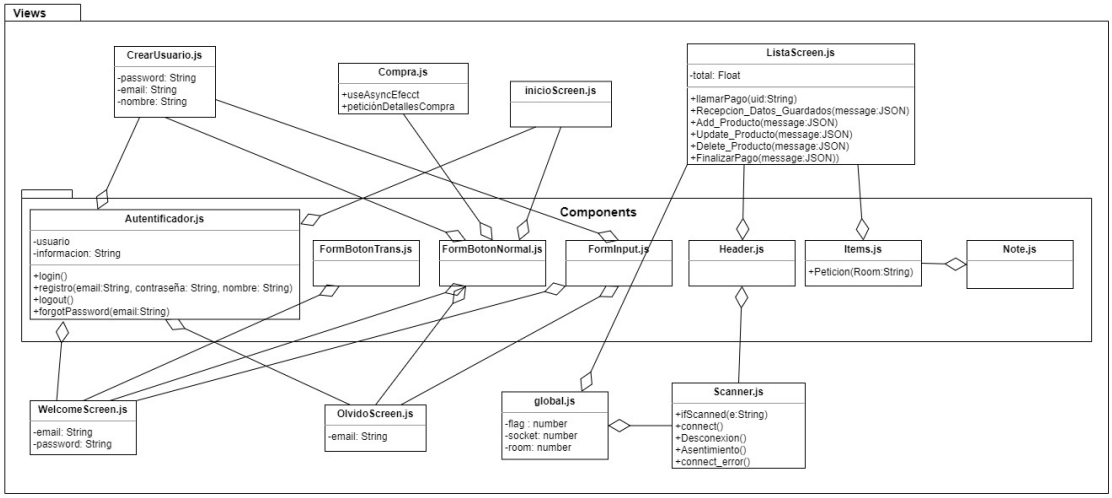

Figura 27: Screens

Router

Se encarga de dirigir al usuario dentro de la aplicación, además de comprobar si el usuario está autenticado.

6.6.3.3 Subsistema Server

WebSocketServer

Contiene los eventos de recepción de socket.io del servidor.

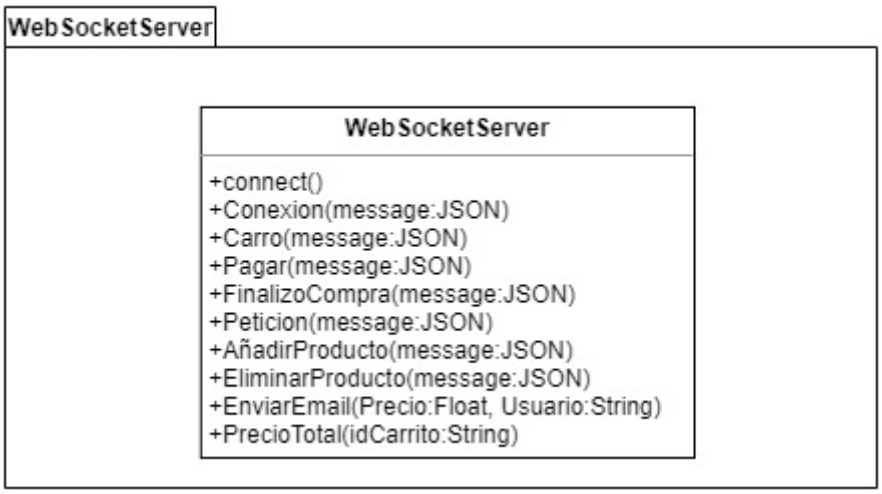

Figura 28: WebSocketServer

Controllers

Contiene los controladores del sistema, encargados de responder a las peticiones de los usuarios.

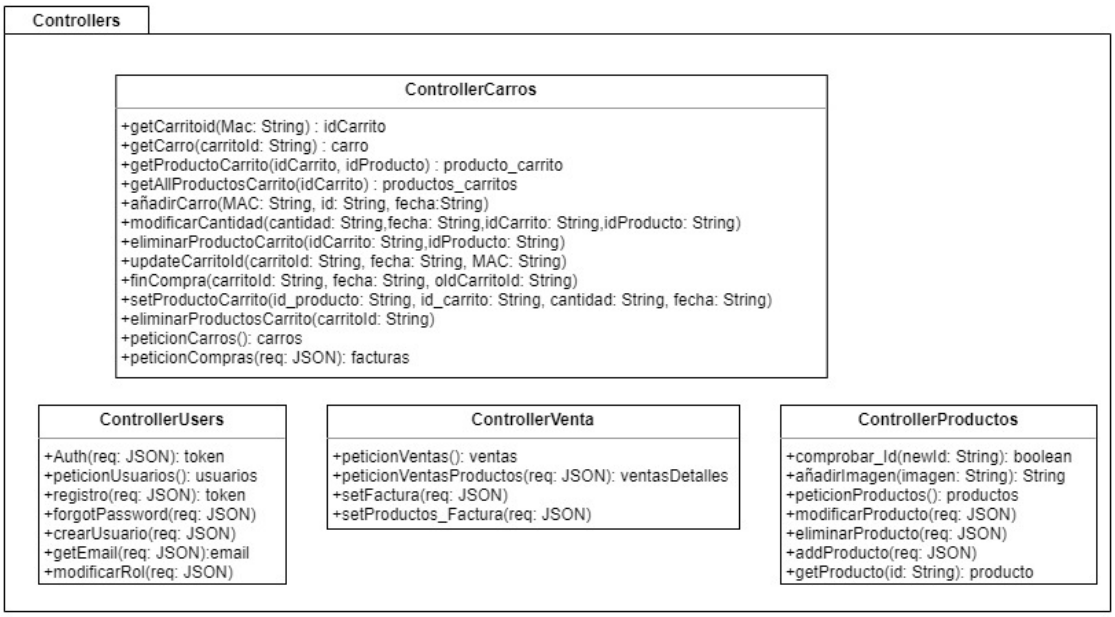

Figura 29: Controllers

DAOS

Contiene los DAOS del sistema, estos se encargan de acceder a la información del sistema.

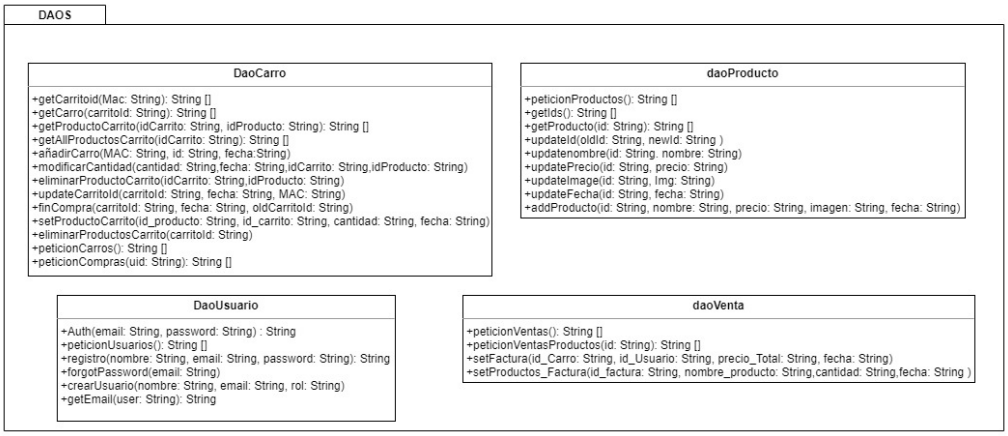

Figura 30: DAOS

DTOS

Contiene los DTOS del sistema, que serán utilizados para transmitir la información contenida en el sistema.

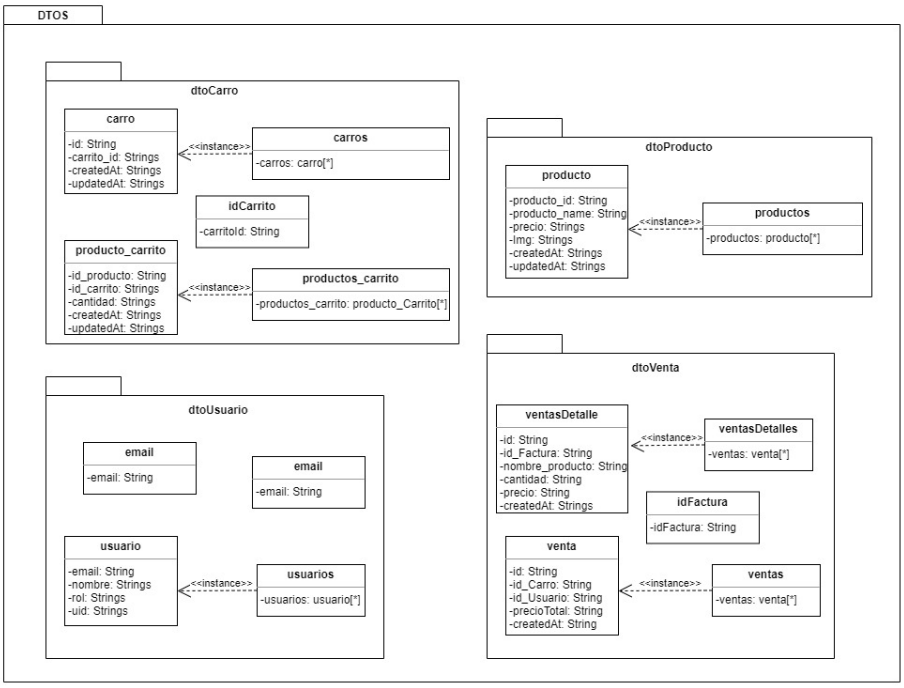

Figura 31: DTOS

6.6.3.4 Subsistema Carro

ObjectTracker

Contiene los métodos y la información para poder detectar los productos.

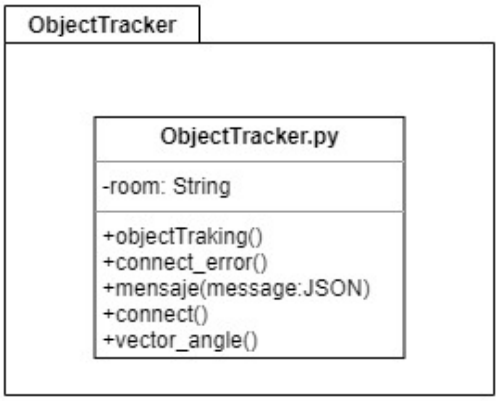

Figura 32: ObjectTracker

6.6.4 Realización de casos de uso en el diseño

Se han realizado los casos de uso en la fase de diseño mediante diagramas de secuencia, solo que ahora las clases son las de los subsistemas al igual que las funciones que las usan.

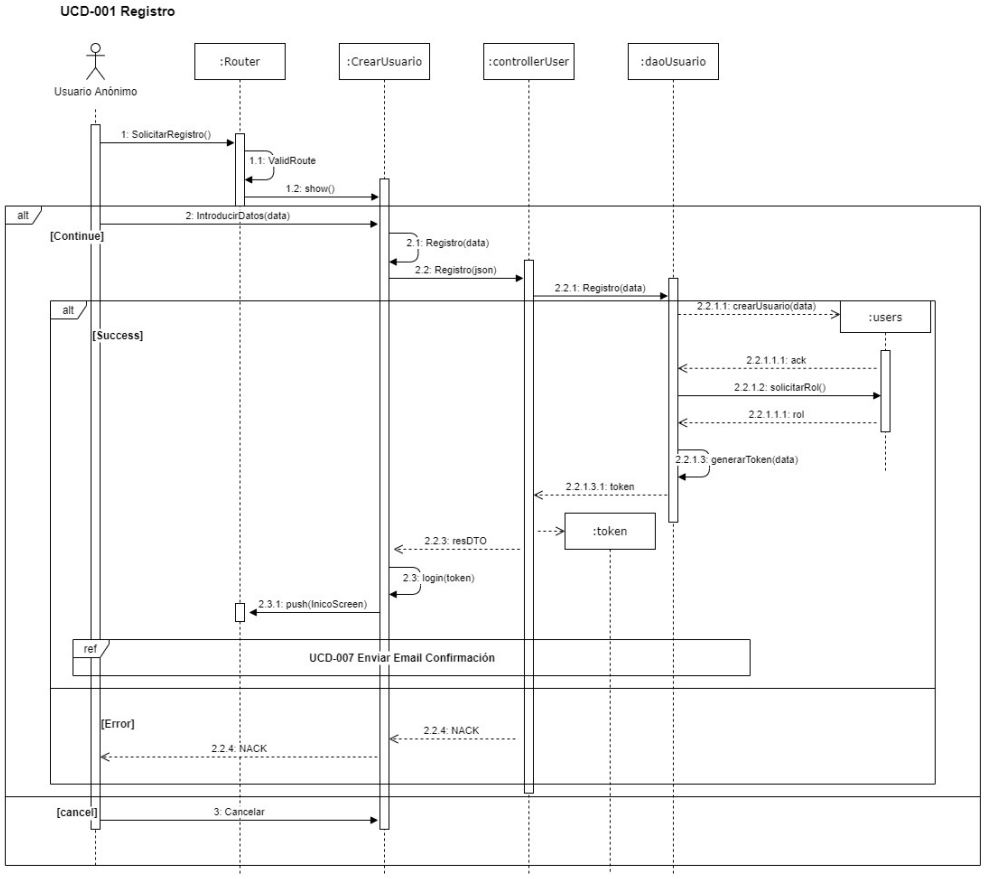

Figura 33: UCD-001 Registro

En la *Figura 33*, se puede observar un ejemplo de la realización de un caso de uso de diseño.

6.6.5 Modelo de despliegue

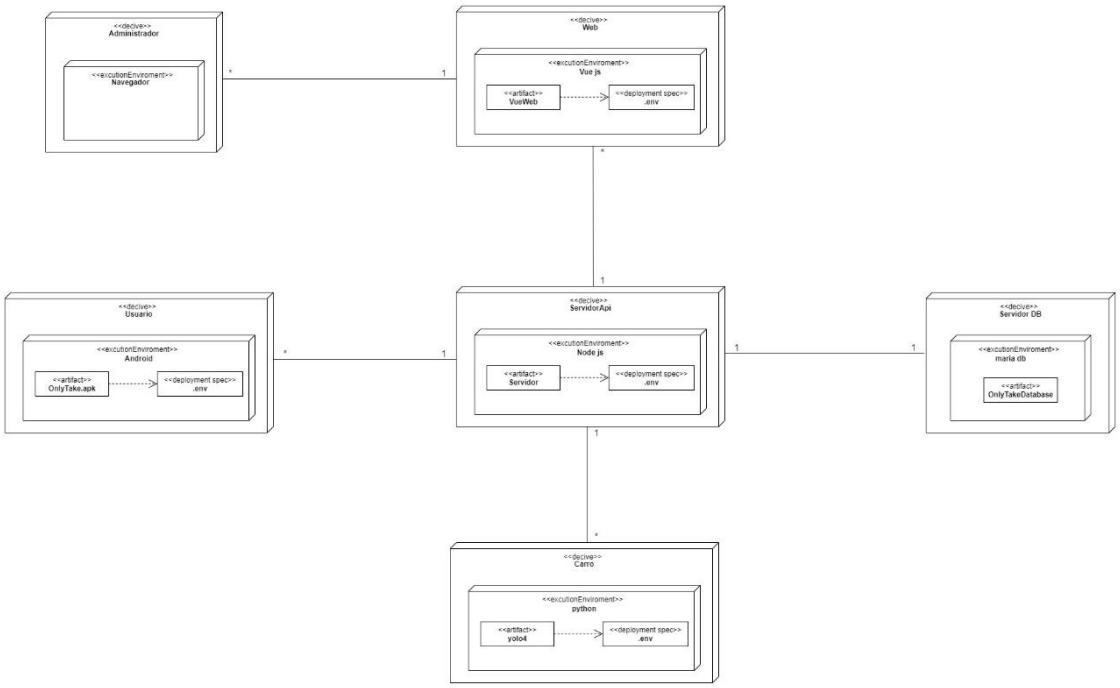

Figura 34: Modelo de despliegue

En la *Figura 34*, se puede ver que el sistema cuenta con 6 nodos distintos:

- **Administrador:** será cualquier dispositivo capaz de conectarse a internet y por lo tanto a la web.
- **Usuario:** será la aplicación desarrollada con la cual los usuarios podrán acceder al sistema.
- **Web:** será la aplicación web a la que acceden los administradores para gestionar el sistema.
- **Carro:** será un carro capaz de lanzar el código con una cámara y conectarse a internet.
- **Servidor DB:** contiene la base de datos principal de nuestro sistema.
- **ServidorApi:** su función es ser la API del sistema y manejar las peticiones que hagan los usuarios.

6.6.6 Descripción de la arquitectura del diseño

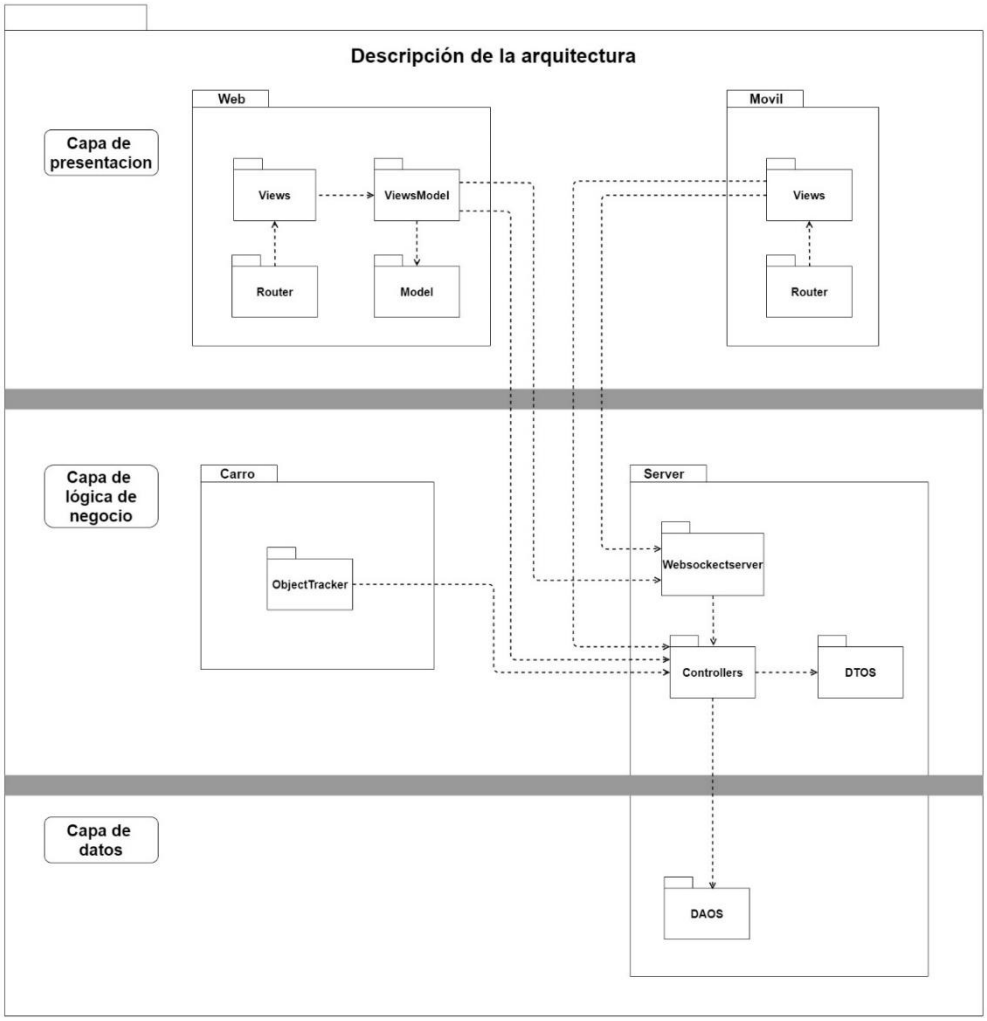

Figura 35: Descripción de la arquitectura del diseño

En la *Figura 35*, se puede observar la arquitectura que sigue el sistema a desarrollar siendo esta el patrón de capas.

6.7 Implementación

La fase de implementación del sistema es la que más tiempo requiere de todo el proyecto, en la que se han usado las técnicas y herramientas que ya se han citado en este documento.

Para que el desarrollo fuera más rápido y sencillo se han seguido los esquemas que se crearon en la fase de diseño.

Para obtener más información sobre los códigos en la fase de implementación puede consultar el Anexo V – Documentación Técnica.

La implementación del sistema se ha llevado a cabo por subsistemas, resultando este subapartado en las siguientes secciones:

- **Server**
- **Carro**
- **Móvil**
- **Web**

6.7.1 Server

Fue la primera parte del sistema que se desarrolló. Al ser una de las partes fundamentales, ya que sobre este que recae la mayor parte de las funcionalidades de todo el sistema, pues es con quien se comunicarán los clientes para hacer peticiones sobre información almacenada en el sistema o modificarlas.

Se encarga de procesar las peticiones que le llegan de los Usuarios. Este se desarrolló sobre el entorno de ejecución de Node.js y mediante el lenguaje de programación JavaScript.

Los principales paquetes y frameworks que se ha utilizado para la creación del servidor son socket.io, mysql, firebase, cors, dotenv, express y nodemailer.

```
async peticionCompras(req,res){
    const respuesta=await daoV.peticionCompras(req.body.uid);
    res.send(respuesta);
},
```

Figura 36: Ejemplo de servicio del servidor API

En la *Figura 36,* se puede observar un ejemplo de servicio al que llamaran los usuarios para obtener las compras que estos han realizado.

Se utilizó el entorno de desarrollo Visual Studio Code para hacer facilitar el desarrollo de esta.

6.7.2 Carro

Fue la segunda parte del sistema que se llevó a cabo puesto que, sin esta, el sistema carecería completamente de utilidad pues no podría detectar los objetos que los usuarios introducen a este.

Se encarga del proceso de detección de los productos que se introducen en el carro mediante la utilización de un modelo propio, se desarrolló mediante el lenguaje de programación Python.

Se utilizó la librería de socket.io para poder realizar la comunicación con el servidor y los usuarios que hayan escaneado el código QR mostrado por el carro. Esta comunicación se da gracias a la librería socket.io que permite la creación de rooms a la que los usuarios se podrán unir para escuchar las emisiones de datos que se envíen.

```
sio.emit('EliminarProducto', {'type': index, 'Room': Room})
```

Figura 37: Ejemplo de emisión desde el carro

En la *Figura 37*, se puede observar un ejemplo de una emisión mediante socket.io del carro al detectar que ha sido extraído un producto del interior de este.

Se utilizó el entorno de desarrollo Pycharm para hacer facilitar el desarrollo de esta.

6.7.2.1 Creación del Modelo

Para la creación del modelo propio lo primero que se hizo fue crear un dataset (conjunto de datos), con este dataset se entrenara un modelo pre entrenado para así obtener mejores resultados a la hora de detectar los productos que si se creara uno desde cero. El modelo pre entrenado que se utilizo fue el de Darknet[44].

La creación del dataset se llevó a cabo tomando fotos de los productos desde distintos algunos y posiciones. Esto se hizo para obtener imágenes de los productos desde distintas perspectivas consiguiendo así un modelo mejor entrenado y por tanto capaz de detectar mejor los productos.

Los productos de los que se tomaron fotos y por tanto el sistema inicialmente es capaz de detectar son:

- Coca cola (lata 330ml)
- Coca cola zero (lata 330ml)
- Botella de agua vidaqua (500ml)
- Swhweppes naranja (lata 330ml)
- Laca Nelly (bote 400ml)

Figura 38: Fotografía de ejemplo del conjunto de datos

En la Figura 38, se puede observar un ejemplo de una fotografía tomada para el entrenamiento del dataset. Se tomaron alrededor 750 fotos de cada producto por separado y 500 fotos de todos los productos en la misma imagen, variando los productos que salían en ellas.

Tras realizar las fotografías de los productos que se iban a utilizar para entrenar el modelo, se llevó a cabo el proceso de etiquetado de estas. Este proceso de etiquetado de las fotos se realizó mediante la herramienta de etiquetado de imágenes labelImg.

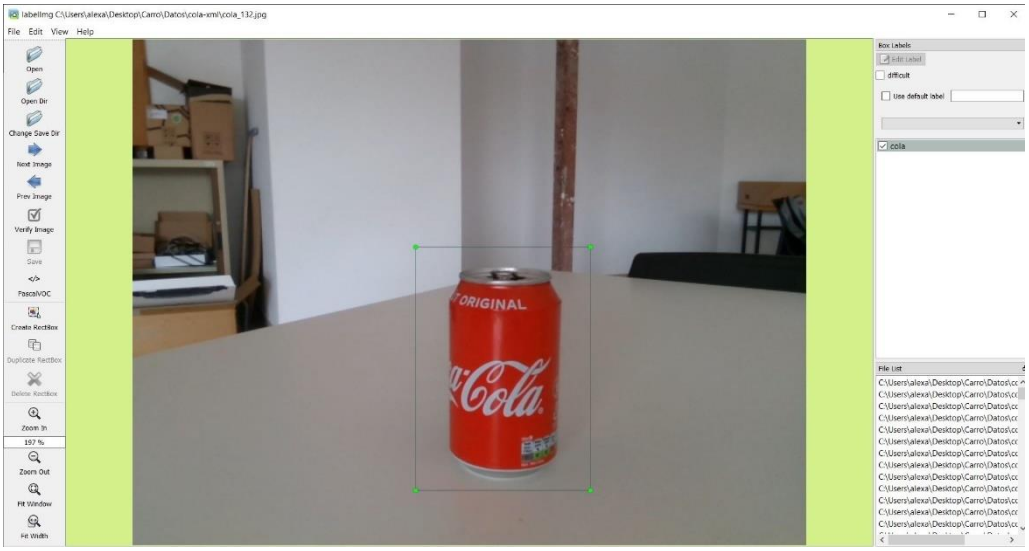

Figura 39: Ejemplo de Imagen etiquetada

En la *Figura 39,* se puede observar una lata de coca cola etiquetada ya etiquetada. Esta etiqueta se guarda en un fichero XML que está asociado a esa imagen.

Una vez todos los productos de las imágenes estén debidamente etiquetados se subirán junto con los XML a la plataforma de ROBOFLOW. Esta generara variaciones de las

imágenes con los productos ya etiquetados en estas mejorando así la capacidad del modelo para detectar productos.

Tras el proceso de aumento llevado a cabo a través de la herramienta Roboflow el dataset contara con 11000 imágenes distintas para el entrenamiento del modelo. De esas imágenes el 80% se utilizarán para la fase de entrenamiento del modelo y el otro 20% se utilizará en la fase de validación del modelo, esta validación se realizará al final de cada epoch para evaluar las métricas. Las imágenes se encontrarán en los ficheros datos_train.txt y datos_text.txt.

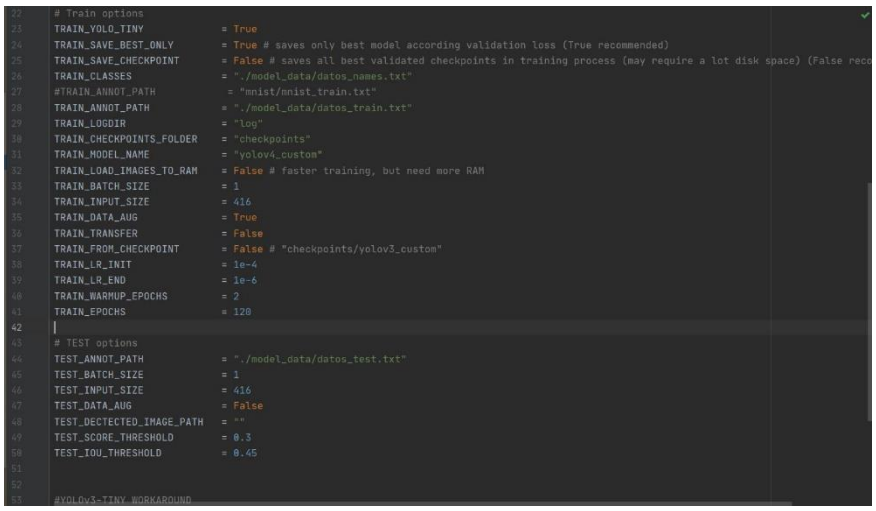

Figura 40: Configs.py

En la *Figura 40*, se puede ver la configuración con la que se ha entrenado el modelo de YOLOv4-tiny. Este modelo se entrenó durante 120 epochs esto se hizo así debido al gran número de imágenes que contiene el dataset utilizado tardando así el entrenamiento en realizarse unas 26 horas en una tarjeta gráfica Nvidia Geforce GTX 1070 GPU. Cabe destacar que se debe tener especial cuidado a la hora de elegir el número de epochs que el modelo realizara a la hora de entrenarse, pues un mayor número de estos no significa mejor detección de objetos ya que habrá un punto donde la red empezara a empeorar si es demasiado alto en relación con el número de imágenes del dataset.

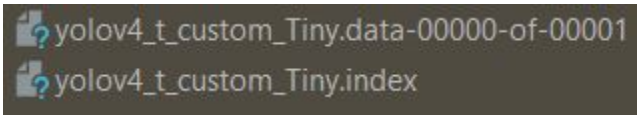

Figura 41: Modelo Entrenado

En la *Figura 41,* se puede observar el modelo resultante tras el proceso de entrenando anteriormente citado.

6.7.3 Móvil

Se llevó a cabo la implementación de este subsistema una vez finalizados los subsistemas de Carro y Server, dado que la aplicación móvil solo será la interfaz a través de la cual los usuarios se comunicarán con el sistema y esta depende de las funcionalidades desarrolladas en los anteriores subsistemas.

Permitirá la interacción de los usuarios con el sistema mediante las diferentes pantallas que esta posee, permitiendo por ejemplo ver el contenido del carro o pagar la compra.

Se utiliza el marco de trabajo React Native para poder llevar a cabo el desarrollo de este subsistema mediante los lenguajes de programación JavaScript, HTML y CSS.

Las librerías más relevantes a la hora de realizar el desarrollo son Firebase, Axios, socket.io, QRCodeScanner y el Gateway de pago Stripe.

Se utilizó el entorno de desarrollo Visual Studio Code para hacer facilitar el desarrollo de esta.

6.7.4 Web

Una vez finalizados todos los anteriores subsistemas, se llevó a cabo el desarrollo de la web desde la que se podrá gestionar el sistema.

Permitirá la interacción de los administradores con el sistema. Permitiéndoles así gestionar los distintos aspectos del mimo.

Se llevó a cabo a través del marco de trabajo de Vue.js que se utiliza para la creación de forma rápida y simple de interfaces de usuario, mediante el uso de los distintos lenguajes de programación como JavaScript, HTML y CSS.

Las principales librerías que se utilizan son Vuetify, Axios y Firebase.

Se utilizó el entorno de desarrollo Visual Studio Code para hacer facilitar el desarrollo de esta.

6.8 Pruebas

La realización de pruebas a lo largo de la creación del proyecto es parte fundamental de este, ya que ayudan a detectar los errores a tiempo antes de convertirse en fallos graves que puedan llegar a afectar al correcto funcionamiento del sistema.

Principalmente se han realizado dos tipos de pruebas:

- **Unitarias:** son pruebas que se realizan durante y la finalización de un componente, se consigue averiguar así si el componente funciona correctamente.
- **Globales:** son las pruebas que se realizan una vez se ha finalizado la construcción de los componentes y se han unido para formar el sistema, con estas se logra comprobar el funcionamiento del sistema en su conjunto.

6.9 Funcionalidad del Sistema

La funcionalidad del sistema se ha dividido en función de los actores que interactúan con cada una de las partes del sistema.

Existen 4 tipos de usuarios en el sistema, por lo cual, este apartado se dividirá a su vez en los siguientes subapartados y constará también de otro subapartado para la detección de objetos:

- **Usuario Anónimo**
- **Usuario No Logueado**
- **Usuario**
- **Administrador**
- **Detección de productos**

El administrador podrá realizar las mismas acciones que el usuario.

Para obtener más información sobre la funcionalidad del sistema se puede consultar el anexo VI – Manual de Usuario.

6.9.1 Usuario Anónimo

Es el tipo de usuario que menos acciones puede realizar en el sistema. Siendo la única acción que puede realizar la de registrarse en el sistema.

6.9.1.1 Pantalla Registro

Figura 42: Pantalla Registro app Móvil

En la *Figura 42*, se puede observar la pantalla de registro de la aplicación móvil. Esta cuenta con cuatro campos que deberán ser rellanados por el usuario y una vez rellenados deberá pulsar el botón de “Registrarse” para poder acceder finalmente al resto de funcionalidades del sistema, que solo están permitidas para los usuarios que han iniciado sesión. Una vez finalizado el registro se le redirigirá a la pantalla de inicio de la aplicación.

También se puede observar que esta pantalla cuenta con otro botón “¿Ya tiene una cuenta?” el cual si lo pulsa le redirigiría a la pantalla de bienvenida de la aplicación.

6.9.2 Usuario No Logueado

Es el tipo de usuario que aún no se ha identificado y, por lo tanto, solo tendrá acceso a las pantallas para autentificar quién es y así acceder al sistema.

6.9.2.1 Pantalla Bienvenida

Figura 43: Pantalla Bienvenida app Móvil

En la *Figura 43*, podemos observar la pantalla de bienvenida de la aplicación móvil. Esta será la primera pantalla que veremos según entremos a la aplicación móvil. Cuenta con 2 campos para poder introducir las claves de acceso del usuario, una vez introducido deberá pulsar el botón de “Iniciar sesión” para poder autentificarse y poder acceder así a las funcionalidades del sistema.

También se puede observar que esta pantalla cuenta con otros dos botones uno “¿Ha olvidado su contraseña?” el cual si lo pulsa le redirigiría a la pantalla de recuperar contraseña de la aplicación y el otro botón “Registrarse” sirve para acceder a la pantalla de registro si aún no se ha registrado.

Esta pantalla tiene el mismo comportamiento que la pantalla registro cuando no se introduce ningún valor en los campos o se introduce erróneamente la contraseña o email.

6.9.2.2 Pantalla Recuperar Contraseña

Figura 44: Pantalla Recuperar Contraseña app Móvil

En la *Figura 44*, podemos observar la pantalla de recuperar contraseña de la aplicación móvil. Esta cuenta con un campo para que el usuario pueda introducir su email para recuperar su contraseña y una vez introducido deberá pulsar el botón de recuperar para que el sistema le envíe un email de recuperación.

También se puede observar que tiene otro botón “Iniciar Sesión”, este se utilizara para regresar a la pantalla de bienvenida.

Esta pantalla tiene el mismo comportamiento que las pantallas anteriores cuando no se introduces ningún valor en los campos o introduces mal un email que no existe.

6.9.3 Usuario

Es la persona que está autentificada en el sistema y tiene acceso a las funcionalidades de la aplicación móvil.

6.9.3.1 Pantalla Inicio

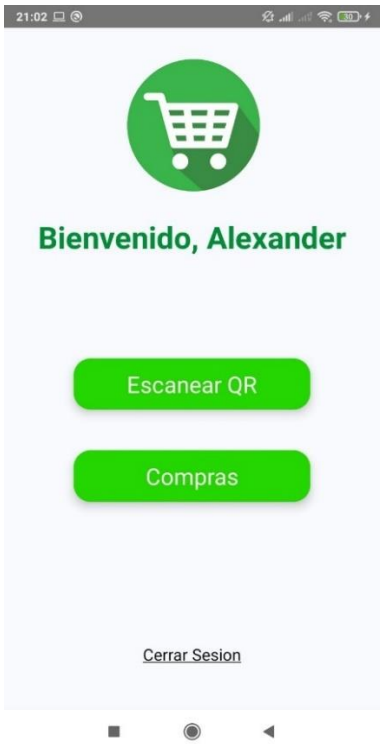

Figura 45: Pantalla Inicio app Móvil

En la *Figura 45*, se puede observar la pantalla de inicio de la aplicación móvil. Esta será la pantalla que verán los usuarios una vez se hayan autentificado en el sistema.

Cuenta con dos botones, uno llamado “Cerrar Sesión” que cierra la sesión activa en la aplicación y redirigirá a los usuarios a la pantalla de bienvenida, otro llamado “Escanear QR” que los redirigirá a la pantalla de Escanear QR y el botón “Compras” que los redirigirá a la pantalla de Compras.

6.9.3.2 Pantalla Compras

Figura 46: Pantalla Compras app Móvil

En la *Figura 46*, se puede observar la pantalla de compras de la aplicación móvil. Esta pantalla mostrará a los usuarios una lista de las compras que han realizado en la aplicación.

La pantalla mostrara en un formato de lista todas las compras que ha realizado ese usuario, pudiendo este presionar cada elemento de la lista para que se le muestren los detalles de esa compra. También contará con un botón llamado “Volver” que redirigirá al usuario a la pantalla anterior.

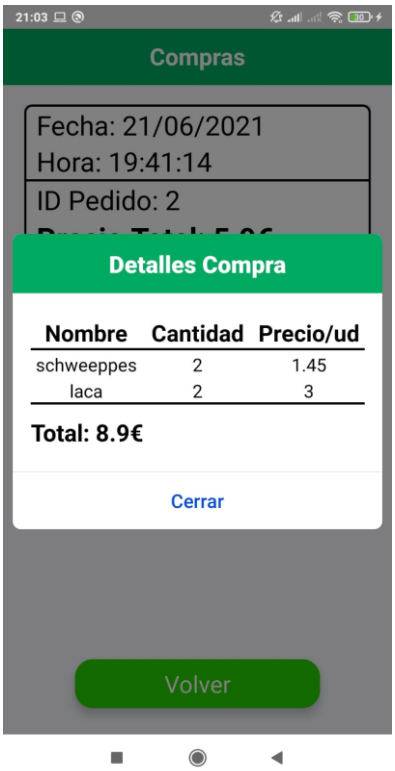

Figura 47: Pantalla Compras app Móvil Detalles Compra

En la *Figura 47*, se puede observar el cuadro de diálogo que se mostrara tras pulsar alguno de los elementos de la lista de las compras realizadas. Este cuadro de diálogo cuenta con un botón llamado “Cerrar” que cerrara el cuadro de diálogo.

6.9.3.3 Pantalla Escáner QR

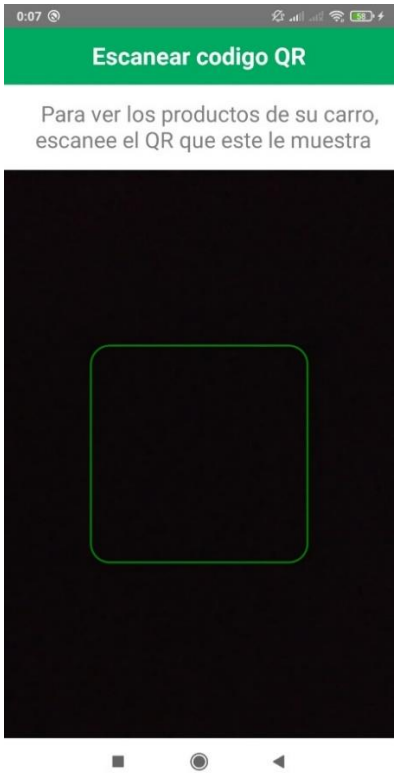

Figura 48: Pantalla Escanear QR app Móvil

En la *Figura 48*, podemos observar la pantalla de escanear QR de la aplicación móvil. Esta pantalla sirve para escanear el QR mostrado por el carro.

6.9.3.4 Pantalla Lista

Figura 49: Pantalla Lista app Móvil

En la *Figura 49*, podemos observar la pantalla Lista de la aplicación móvil. Esta pantalla sirve para ver los productos que hemos introducido en el carro.

Cuenta con un botón que muestra el precio a pagar por los artículos en el carro.

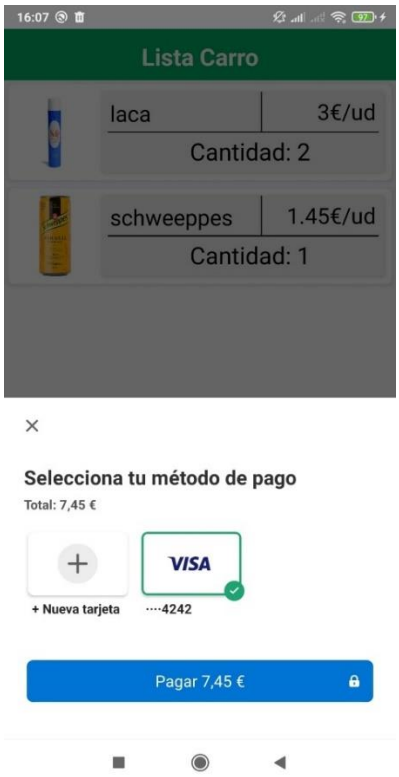

Figura 50: Pantalla Lista pagar app Móvil

En la *Figura 50*, se pueden ver las formas de pago que se le muestran a un usuario que tiene un método de pago almacenado.

6.9.4 Administrador

El administrador tiene acceso a las mismas funcionalidades que el usuario, además, podrá acceder a la web para poder gestionar el sistema desde esta.

6.9.4.1 Web Inicio Sesión

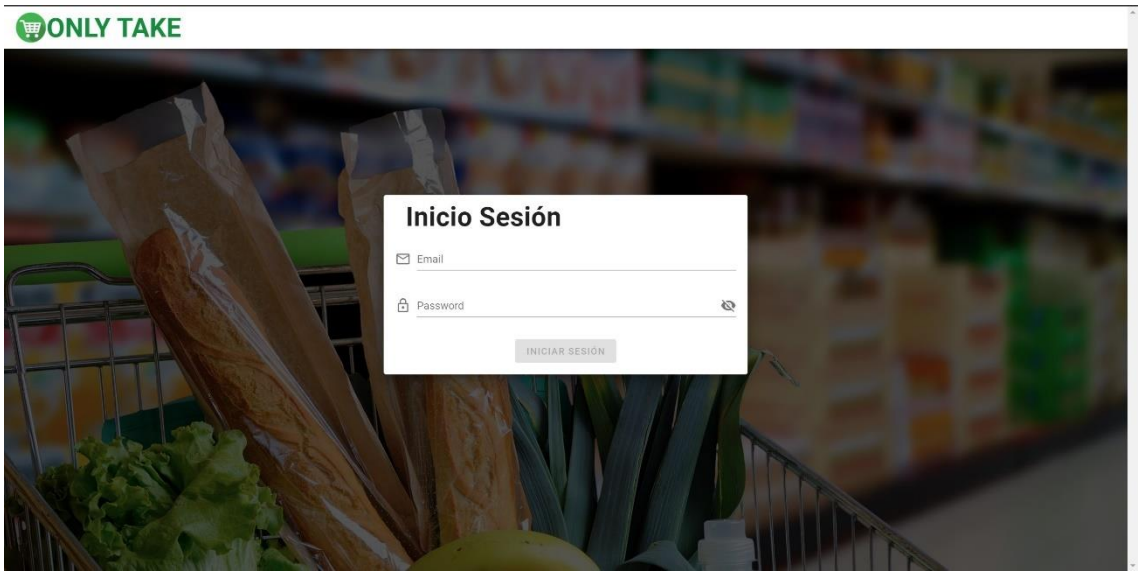

Figura 51: Pantalla Inicio Sesión Web

En la *Figura 51*, podemos observar la pantalla de inicio sesión de la aplicación web. Esta sirve para autentificar a los administradores.

La pantalla cuenta con un botón que estará bloqueado hasta que el usuario rellene los campos.

6.9.4.2 Pantalla Lista

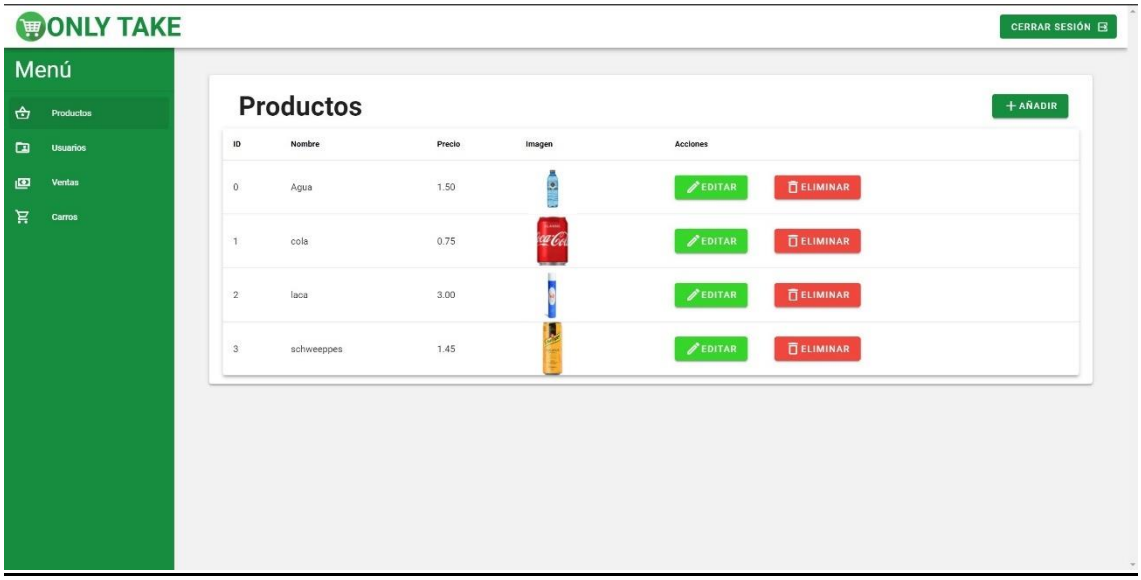

Figura 52: Pantalla Productos Web

En la *Figura 52*, podemos observar una lista de los productos que hay en el sistema.

La barra de navegación de la izquierda se utiliza para poder moverse por la aplicación web, además de indicar en que pantalla nos encontramos. También incluye un botón de “Cerrar sesión” en la parte superior derecha que permitirá salir al usuario en cualquier momento de la aplicación. Estos dos elementos se encontrarán en todas las pantallas de la aplicación una vez el administrador se haya autentificado.

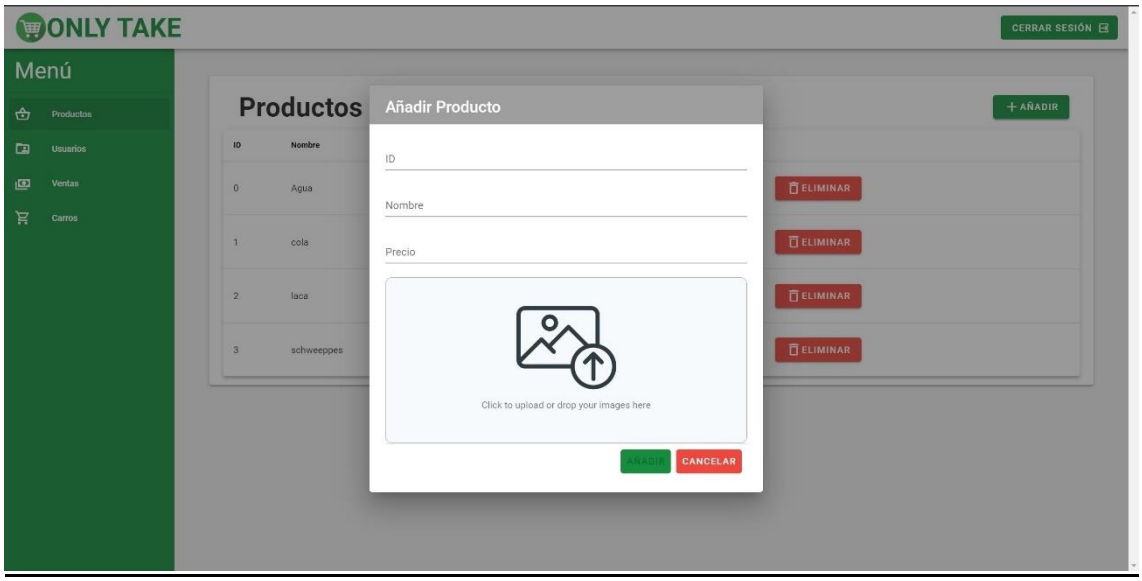

Figura 53: Pantalla Productos Web Añadir Producto

En la *Figura 53*, podemos observar el cuadro de diálogo emergente que surge tras pulsar el botón de añadir. Este cuadro de diálogo cuenta a su vez con dos botones. El botón “Añadir” estará bloqueado hasta que todos los campos del cuadro estén rellenados y el botón “Cancelar” cerrará el cuadro de diálogo. Si se rellenan los campos y se pulsa “Añadir” se agregará ese producto al sistema, pero no por ello significa que el carro sea capaz de detectarlo.

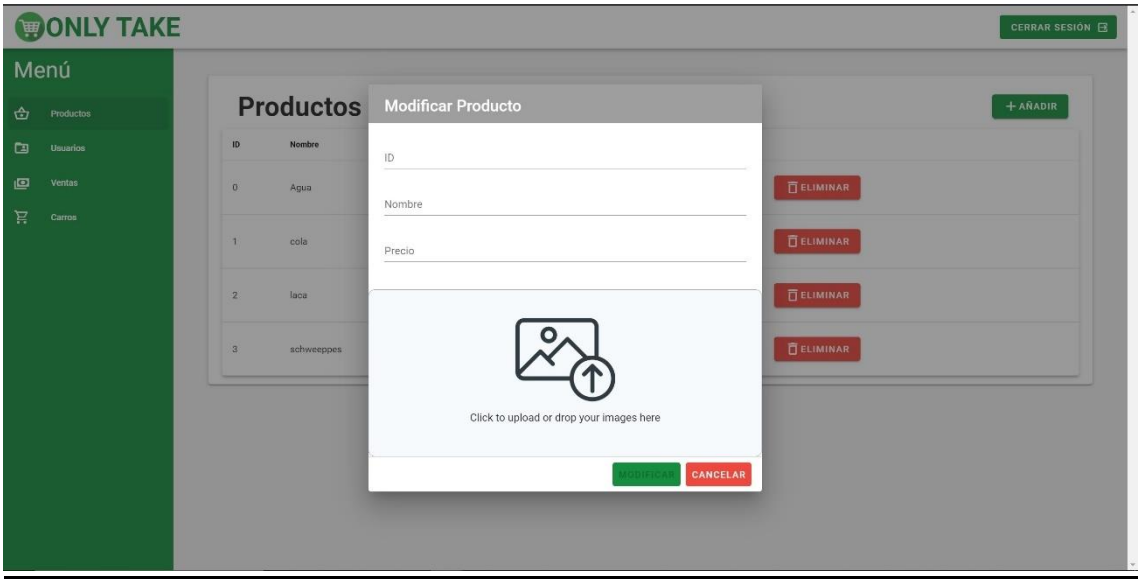

Figura 54: Pantalla Productos Web Modificar Producto

En la *Figura 54*, podemos observar el cuadro de diálogo emergente que surge tras pulsar el botón de “Modificar”. Este cuadro de diálogo cuenta a su vez con dos botones. El botón “Añadir” estará bloqueado hasta que se rellene algún campo, mientras que el botón “Cancelar” cerrará el cuadro de diálogo. El identificar del producto se puede modificar, pero no se recomienda porque puede dar errores al detectar productos.

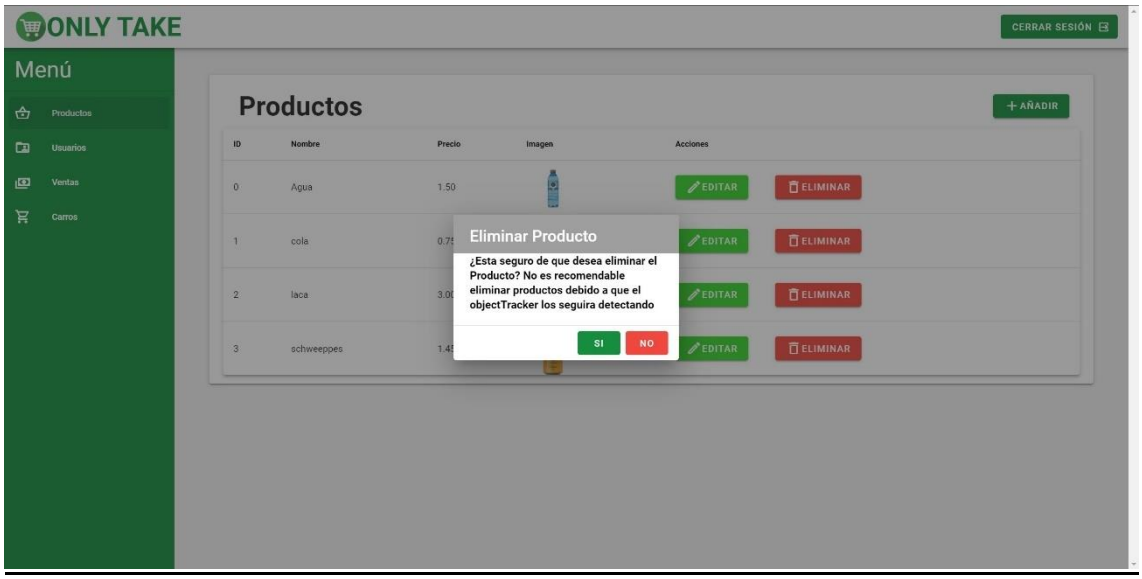

Figura 55: Pantalla Productos Web Eliminar Producto

En la *Figura 55*, podemos observar el cuadro de dialogo emergente que surge tras pulsar el botón de “Eliminar”. Si se pulsa dicho botón se eliminará el producto, pero no se recomienda debido a que dará problemas a la hora de detectar y añadir los productos.

6.9.4.3 Usuarios Web

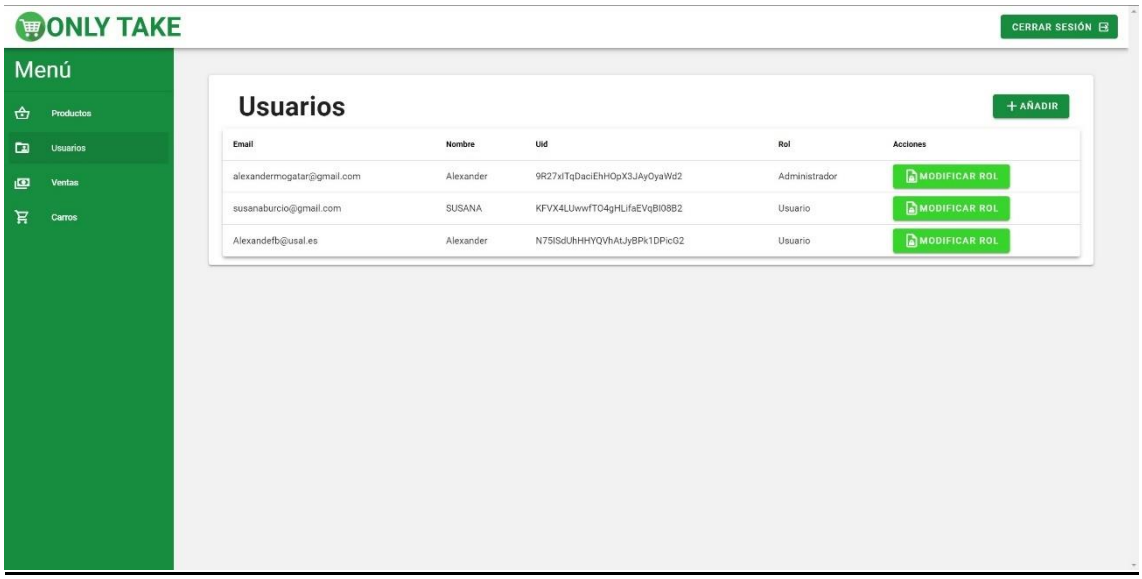

Figura 56: Pantalla Usuarios Web

En la *Figura 56*, se puede observar una lista de los usuarios que hay en el sistema.

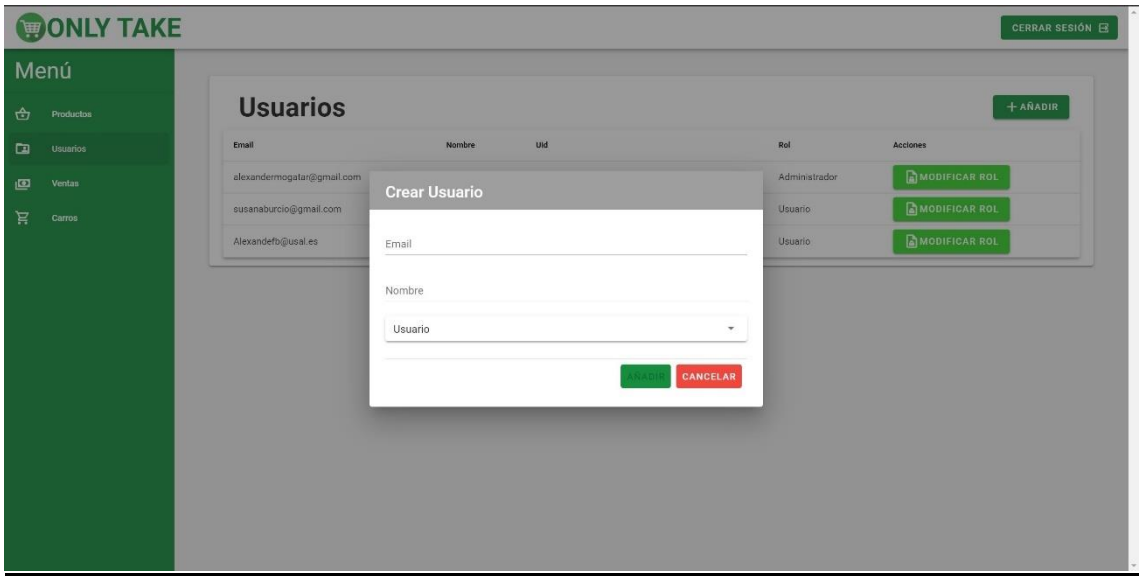

Figura 57: Pantalla Usuarios Web Crear Usuario

En la *Figura 57*, se puede observar el cuadro de dialogo emergente que surge tras pulsar el botón de “Añadir”. Este cuadro de diálogo cuenta a su vez con dos botones. El botón “Añadir” estará bloqueado hasta que se rellenen todos los campos, mientras que el botón “Cancelar” cerrará el cuadro de diálogo.

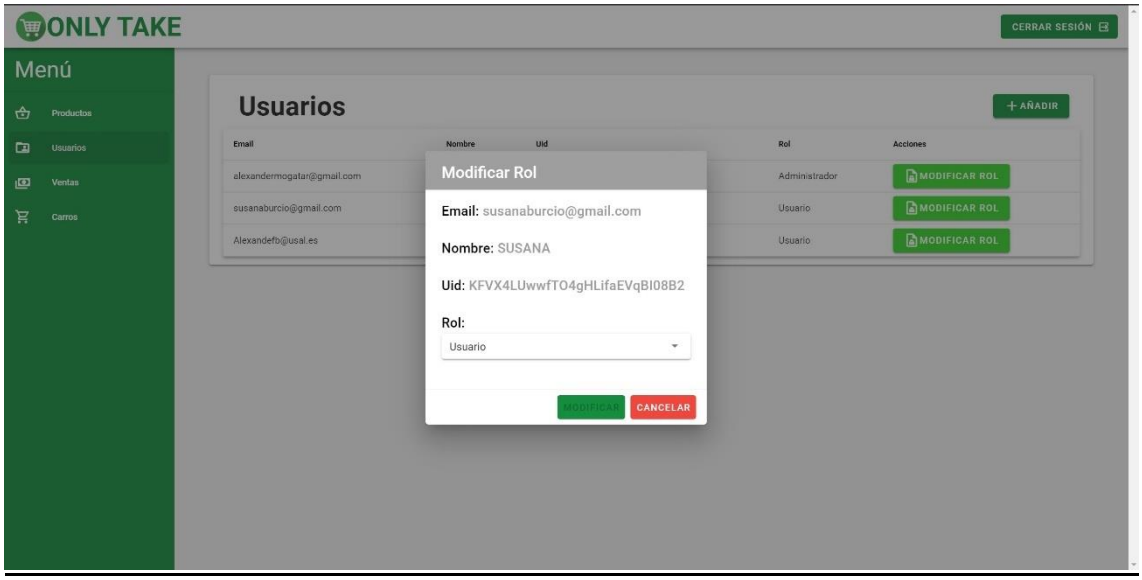

Figura 58: Pantalla Usuarios Web Modificar Rol

En la *Figura 58*, se puede observar el cuadro de dialogo emergente que surge tras pulsar el botón de “Modificar Rol”. Este cuadro de diálogo cuenta a su vez con dos botones. El botón “Añadir” estará bloqueado hasta que se seleccione un rol diferente al que tenía el usuario antes, mientras que el botón “Cancelar” cerrará el cuadro de diálogo.

6.9.4.4 Ventas Web

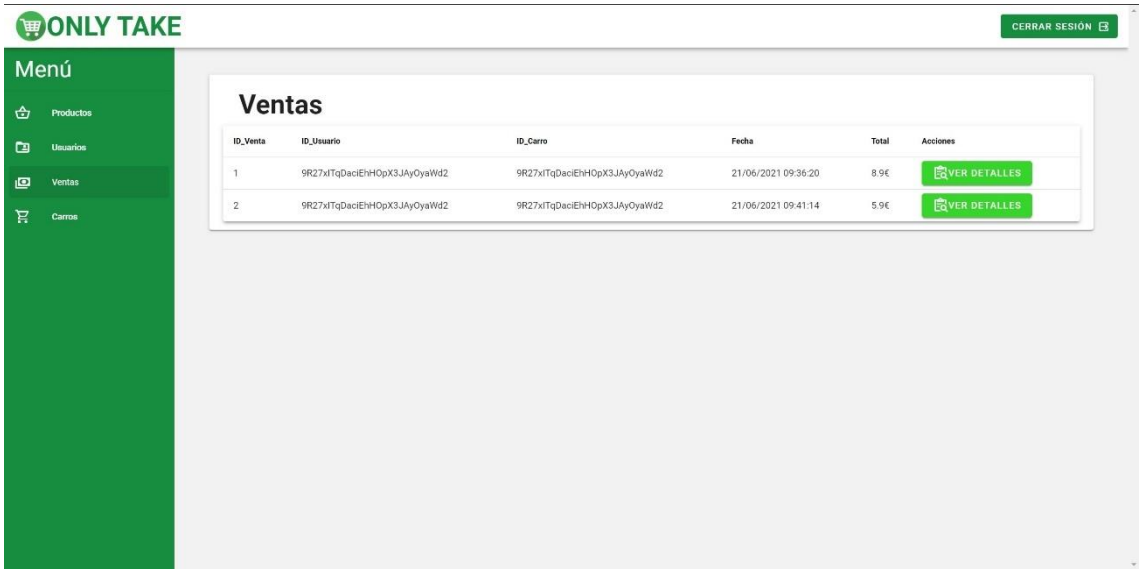

Figura 59: Pantalla Ventas Web

En la *Figura 59*, se puede observar una lista de las ventas que se han realizado.

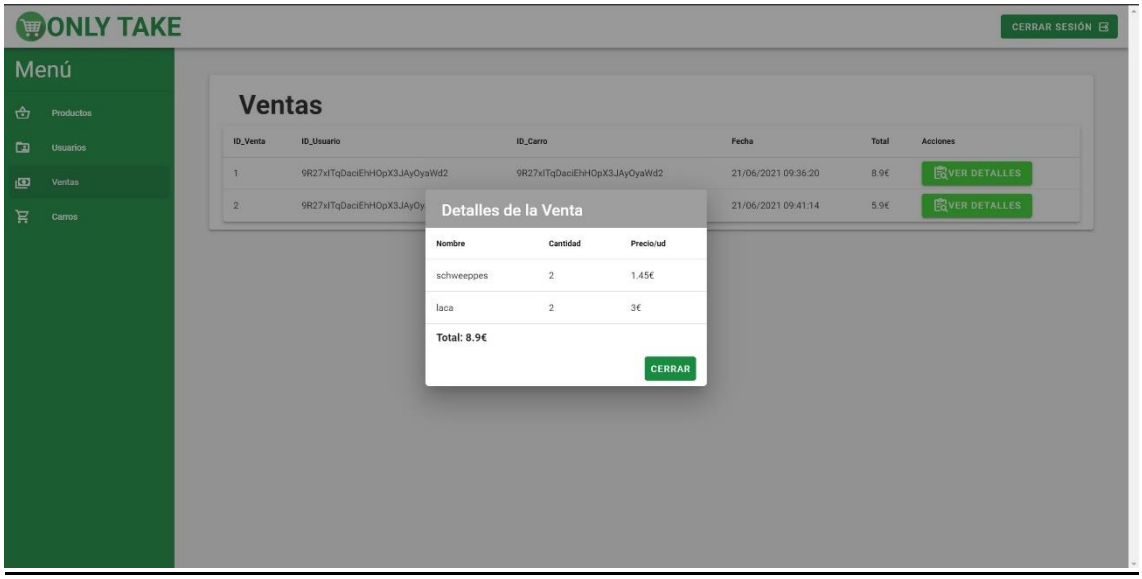

Figura 60: Pantalla Ventas Web Detalles de la Venta

En la *Figura 60*, se puede observar el cuadro de dialogo emergente que surge tras pulsar el botón de “Ver Detalles”. Este cuadro de dialogo cuenta con un botón para cerrar el cuadro de diálogo.

6.9.4.5 Ventas Web

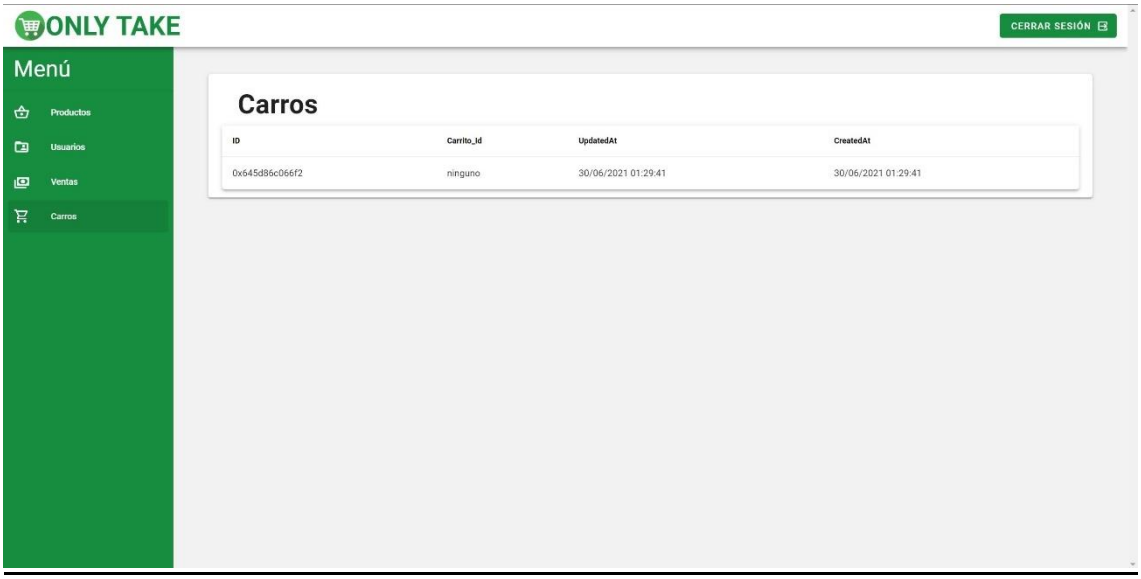

Figura 61: Pantalla Carros Web

En la *Figura 61*, se puede observar una lista de los carros que hay en el sistema.

6.9.5 Detección de productos

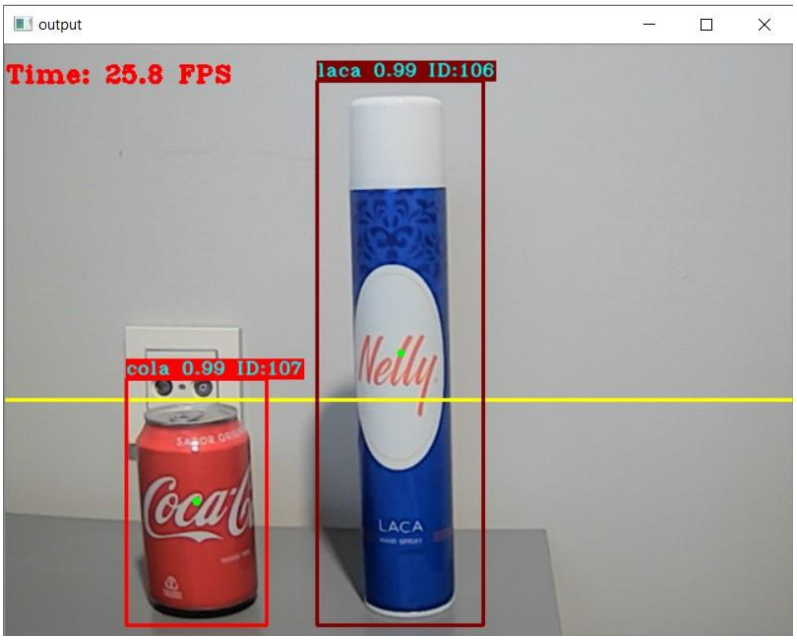

Figura 62:Detección de Productos en tiempo real

En la *Figura 62*, se puede observar una captura de pantalla de la ventana de salida del carro. Se pueden observar dos cuadrados delimitadores, uno para la coca cola y otro

para la laca. En el punto medio de estos cuadrados delimitadores, se encuentra un punto de color verde que se almacenará en cada frame del vídeo. Cuando el punto verde sobrepasa la línea amarilla por la parte superior o por parte inferior, se comprueba el punto anterior para saber la dirección que tiene este producto. De esta forma, sabremos si el usuario ha introducido un producto o lo ha extraído.

7. Limitaciones del prototipo

Como cualquier prototipo que se desarrolla, este tendrá limitaciones, técnicas y apartados que faltan por pulir.

Las limitaciones técnicas del prototipo:

- No se ha podido llevar a cabo la implementación del código del carro a un carro de la compra real por limitaciones en cuanto al hardware, por tanto, los carros de la compra se emulan mediante un portátil conectado a una webcam.
- El modelo aun siendo bastante preciso a la hora de detectar, falla ocasionalmente en el momento de captar los productos si las condiciones lumínicas no son similares a las que había cuando se tomaron las fotos de los objetos para el dataset. Lo mismo sucede si la cámara apunta a un fondo muy colorido haciendo que le cueste más detectar los objetos.
- Si el usuario tapa la cámara no existe mecanismo que evite esto, imposibilitando así la detección de los artículos por parte del carro.

8. Conclusiones y líneas de trabajo futuras

8.1 Conclusiones

Una vez desarrollado el proyecto y estando totalmente funcional, se puede concluir que se han conseguido cumplir todos los objetivos, tanto los objetivos funcionales del proyecto como los personales. A continuación, se muestra cómo se han conseguido realizar todos los objetivos funcionales:

- El proyecto desarrollado consta de un sistema capaz de detectar los productos que introduzcan los usuarios en el carro mediante una cámara web que utiliza YOLOv4-tiny para detectar los objetos introducidos en el carro, que se comunicará con el servidor para poder almacenarlo en el sistema.
- Se ha desarrollado un servidor al que se le podrán realizar distintas peticiones para poder así almacenar los datos de los usuarios, facturas, productos introducidos en el carro, productos o incluso modificar el Rol de los usuarios, así como añadir o eliminar productos. Con esto, podemos concluir que la principal funcionalidad del sistema es la de gestionar toda la información, tanto de los usuarios como de los productos, facturas y ventas a través de las peticiones que recibe de parte de los usuarios del sistema.
- Se ha creado una aplicación móvil desde la cual los usuarios se deben registrar para poder acceder a todas las funcionalidades de esta. Una vez registrados, podrán ver las diferentes compras que se han llevado a cabo a través de la aplicación, como escanear el código QR mostrado por el carro para poder ver una lista con todos los productos de su carro, permitiéndoles pagar la compra desde el móvil.
- Se ha desarrollado una aplicación web en la que los administradores podrán iniciar sesión para gestionar diferentes aspectos del sistema. Podrán así, crear nuevos usuarios, modificar el Rol de los usuarios, ver todas las ventas realizadas en la aplicación, observar los diferentes carros del sistema, listar los productos del sistema, así como añadir, modificar o eliminar dichos productos. Así, desde la aplicación web se enviarán las peticiones al servidor, con el fin de que realice las modificaciones o almacenamientos necesarios de la información.

En cuanto a los objetivos personales, se ha podido aprender más sobre diferentes aspectos a la hora de llevar a cabo este proyecto como la aplicación de un proyecto real de la ingeniería del software que gracias a esta, el desarrollo del proyecto ha estado mucho más organizado, evitando un desarrollo completamente caótico. También se añade el aprendizaje de nuevas técnicas para llevar a cabo de manera más rápida y eficiente el desarrollo de proyectos o el entendimiento determinado de conceptos introducidos en la carreara en los que no se llegó a profundizar demasiado.

Por último, pienso que los conocimientos que he adquirido a la hora de llevar a cabo el proyecto me servirán en un futuro con el fin de no cometer los erros que se han podido desarrollar a lo largo del proyecto, así como poder desarrollar un proyecto de manera más eficiente, estructurada y rápida.

8.2 Líneas de trabajo futuras

Algunas funcionalidades que se pueden aplicar en un futuro al proyecto podrían ser:

- El entrenamiento de un mejor modelo para poder detectar de forma más eficiente los productos.
- El desarrollo de un carro de la compra en el cual poder implementar el sistema de detección de productos creados.
- Separación en diferentes supermercados del sistema, de tal forma que diferentes supermercados pudieran utilizar a la vez sin interferir unos con otros.
- Mostrar sugerencias a la hora de realizar la compra en base a los productos que se han comprado en otras ocasiones.

Estas serán algunas funcionalidades que podrían mejorar en gran medida el sistema que se ha creado.

9. Referencias

[1] Amazon Dash Cart

- https://www.amazon.com/b?ie=UTF8&node=21289116011

[2] Caper Smart Cart

- https://www.caper.ai/cart

[3] MOBI

- https://www.lavanguardia.com/motor/innovacion/20210131/6211115/carro-compra-electrico-autonomo-acompana-supermercado-mobi.html

[4] REST

- https://es.wikipedia.org/wiki/Transferencia_de_Estado_Representacional

[5] WebSocket

- https://es.wikipedia.org/wiki/WebSocket

[6] Código QR

- https://es.wikipedia.org/wiki/C%C3%B3digo_QR

[7] YOLO

- https://arxiv.org/pdf/1506.02640.pdf
- https://arxiv.org/pdf/1804.02767.pdf
- https://arxiv.org/pdf/2004.10934.pdf

[8] DeepSORT

- https://arxiv.org/pdf/1703.07402.pdf
- https://www.researchgate.net/publication/344099630_Object_Tracking_Using_Improved_Deep_Sort_YOLOv3_Architecture

[9] HTML

- https://es.wikipedia.org/wiki/HTML

[10] CSS

- https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada

[11] JavaScript

- https://developer.mozilla.org/es/docs/Web/JavaScript
- https://es.wikipedia.org/wiki/JavaScript

[12] Vue.js

- https://012.vuejs.org/guide/
- https://vuejs.org/v2/guide/
- https://es.wikipedia.org/wiki/Vue.js

[13] Vuetify

- https://vuetifyjs.com/en/
- https://vuetifyjs.com/en/introduction/why-vuetify/

[14] Vue Router

- https://router.vuejs.org/

[15] Axios

- https://axios-http.com/docs/intro

[16] Firebase

- https://firebase.google.com/
- https://firebase.google.com/docs/storage/web/start
- https://firebase.google.com/docs/auth/web/start

[17] React Native

- https://reactnative.dev/
- https://en.wikipedia.org/wiki/React_Native

[18] QRCodeScanner

- https://github.com/moaazsidat/react-native-qrcode-scanner

[19] Socket.io

- https://socket.io/docs/v4/index.html
- https://en.wikipedia.org/wiki/Socket.IO

[20] Node.js

- https://nodejs.org/es/about/
- https://es.wikipedia.org/wiki/Node.js

[21] Express

- https://en.wikipedia.org/wiki/Express.js
- https://expressjs.com/es/
- https://developer.mozilla.org/es/docs/Learn/Server-side/Express_Nodejs/Introduction

[22] MariaDB

- https://es.wikipedia.org/wiki/MariaDB
- https://mariadb.org/about/

[23] mysql

- https://www.npmjs.com/package/mysql

[24] Python

- https://www.python.org/about/
- https://es.wikipedia.org/wiki/Python

[25] LabelImg

- https://github.com/tzutalin/labelImg

[26] Roboflow

- https://blog.roboflow.com/getting-started-with-roboflow/

[27] Visual Strudio Code

- https://es.wikipedia.org/wiki/Visual_Studio_Code
- https://code.visualstudio.com/docs

[28] Pycharm

- https://www.jetbrains.com/es-es/pycharm/

[29] Npm

- https://docs.npmjs.com/about-npm

[30] PostMan

- https://www.postman.com/

[31] Microsoft Project

- https://www.microsoft.com/es-es/microsoft-365/project/project-management-software

[23] EZEstimate

- https://sites.google.com/site/ezestimation/

[33] Draw.io

- https://app.diagrams.net/

[34] Git

- https://git-scm.com/
- https://es.wikipedia.org/wiki/Git

[35]JSdoc

- https://jsdoc.app/about-getting-started.html

[36] Vue Styleguidist

- https://vue-styleguidist.github.io/docs/Documenting.html

[37] Sphinx

- https://www.sphinx-doc.org/en/master/

[38] Proceso Unificado

- https://es.wikipedia.org/wiki/Proceso_unificado

[39] Moreno García M. N. - Transparencias de Gestión de Proyectos

[40] García Peñalvo, F.J., Moreno García, M.N. - Transparencias de Ingeniería del Software I.

[41] Duran Toro, A., Bernárdez Jiménez, B. -Metodología para la Elicitación de Requisitos de Sistemas Software.

[42] Pressman, R. S. - Ingeniería del Software: Un Enfoque Práctico.

[43] Moreno García, M.N. - Transparencias de Ingeniería del Software II.

[44] Darknet

- https://pjreddie.com/darknet/yolo/