
Proyecto de Desarrollo de Aplicaciones Multiplataforma

Control De Acceso De Vehículos

CavAll

CICLO FORMATIVO DE GRADO SUPERIOR
Desarrollo de Aplicaciones Multiplataforma (IFCS02)

Curso 2021-22



Autor:
Adrian Lozano Lastras

Tutor:
Javier Palacios

Departamento de Informática y Comunicaciones
I.E.S. Luis Vives



1	INTRODUCCIÓN	4
1.1	OBJETIVO	6
1.2	ALCANCE	7
1.3	JUSTIFICACIÓN	8
2	IMPLEMENTACIÓN	9
2.1	ANÁLISIS DE LA APLICACIÓN	9
2.1.1	CLIENTE	10
2.1.2	SERVIDOR	12
2.2	DISEÑO	14
2.3	IMPLEMENTACIÓN	20
2.3.1	CLIENTE	20
2.3.2	SERVIDOR	22
2.4	IMPLANTACIÓN	35
2.5	DOCUMENTACIÓN	36
2.6	PROCESO DE DESARROLLO Y PLANIFICACIÓN	38
3	RESULTADOS Y DISCUSIÓN	41
4	TRABAJO FUTURO	42
5	CONCLUSIONES	43
6	BIBLIOGRAFÍA	44
6.1	CLIENTE	44
6.2	SERVIDOR	45
	ANEXOS	46
I.	PROCESAMIENTO Y TRATAMIENTO DE IMÁGENES	46
II.	CÁMARAS	48
III.	DESCARGAS	50

1 INTRODUCCIÓN

El presente proyecto pretende desarrollar un sistema para el reconocimiento de matrículas, utilizando visión artificial, que permita detectar automáticamente las matrículas para poder controlar el acceso a una instalación de los diferentes vehículos que deseen acceder, verificando si estos están autorizados o no para poder acceder a un aparcamiento, mejorando de esta forma la seguridad en una empresa, centro, edificio, institución ,etc.

El sistema cuenta con una parte cliente la cual se encargará de reconocer la matrícula para enviársela al servidor el cual se encargará de cotejar dicha matrícula en una base de datos para verificar que tiene permiso para acceder al parking, realizando un registro de las entradas y salidas realizadas por los vehículos que tienen acceso y poder consultar la información tanto de los vehículos y de sus propietarios, como de las entradas y salidas realizadas por estos. También se realizarán diferentes logs registrando tanto las entradas como las salidas permitidas y denegadas con toda la información necesaria.

Para la realización de la parte cliente se ha optado por realizar un programa desarrollado en Python para poder realizar el reconocimiento de las matrículas mediante diferentes filtrados en la imagen captada por la cámara, el cual dará visual de lo que este captando la cámara y al reconocer una matrícula, realizará una petición al servidor para que este la verifique con la base de datos.

Para ello se ha realizado un estudio de las diferentes técnicas de detección y reconocimiento para saber que tecnologías, métodos de filtrado y librerías escoger. Al final se ha optado por emplear para su realización el lenguaje de programación Python junto con las librerías de OpenCv y Pytesseract para el reconocimiento de las matrículas. Pudiendo reconocer las matrículas con OpenCv, obteniendo la imagen del vehículo con su matrícula para posteriormente pasarla por diferentes filtros obteniendo una imagen más limpia de la matrícula y mediante Pytesseract poder leer la matrícula obteniendo sus caracteres.

En cuanto al servidor se ha realizado una Aplicación Web con una arquitectura (Modelo - Vista - Controlador) empleando .Net Core en su versión 5 y usando como lenguaje de programación C#. El servidor estará conectado a una base de datos SQL Server para poder realizar los CRUD necesarios y filtrado de datos desde la página web, a parte de poder obtener los datos en Excel y estará funcionando como servidor en una intranet mediante el administrador de Internet Information Services.

Por último, se han realizado pruebas de validación para la comprobación de su buen funcionamiento, empleando, imágenes de vehículos y matrículas, vídeos grabados de vehículos en movimiento y estacionados, vídeos con webcam empleando matrículas de internet y a papel, y pruebas en entornos más reales, como puede ser con vehículos en un garaje y en la calle.

Reconocimiento Automático de Matrículas

Automatic number plate recognition (ANPR) o Licence Plate Recognition (LPR), consiste en un método de vigilancia y control de accesos que permite detectar en una imagen la matrícula de un vehículo para poder utilizar Reconocimiento Óptico de Caracteres (OCR) para así obtener los valores alfanuméricos que componen dicha matrícula y de esta forma poder realizar el registro de entrada y salida en ciertos lugares como estacionamientos, empresas, centros educativos, residencias o en peajes, entre otros.

Estos sistemas a menudo utilizan iluminación infrarroja para hacer posible que la cámara pueda tomar fotografías en cualquier momento del día. La tecnología ANPR tiende a ser específica para una región, debido a la variación entre matrículas de un lugar a otro.

El software del sistema se ejecuta sobre un hardware de PC estándar y puede ser enlazado con otras aplicaciones o bases de datos. Primero utiliza una serie de técnicas de manipulación de la imagen para detectar, normalizar y realzar la imagen del número de la matrícula, y finalmente reconocimiento óptico de caracteres para extraer los alfanuméricos de la matrícula.

Visión Artificial

La visión artificial es una disciplina científica que incluye métodos para adquirir, procesar, analizar y comprender las imágenes del mundo real con el fin de producir información numérica o simbólica para que puedan ser tratados por un ordenador para tratar de conseguir que las máquinas puedan percibir y comprender una o varias imágenes y actuar de una manera determinada.

La visión artificial engloba todas las aplicaciones industriales y no industriales en las que una combinación de hardware y software proporciona orientación operativa a los dispositivos en la ejecución de sus funciones basándose en la captura y el procesamiento de imágenes.

El objetivo de la visión artificial es conseguir el desarrollo de estrategias automáticas para el reconocimiento de patrones complejos en imágenes de múltiples dominios

Reconocimiento Óptico de Caracteres

El reconocimiento óptico de caracteres, Optical Character Recognition (OCR) es un proceso automático mediante el cual se extrae texto de una imagen y se transforma a un formato digital, almacenando caracteres, números y símbolos en forma de datos, a los cuales poder acceder y manipular.

Por ejemplo se podría usar OCR para digitalizar el contenido de un documento, de este modo se podría ahorrar el tener que pasar manualmente todo ese contenido. Se podría también extraer los caracteres y dígitos de las matrículas de vehículos, documentos, escritos, certificados, registros, entre otros.

1.1 OBJETIVO

El Objetivo Principal del proyecto será:

Afianzar los conocimientos adquiridos durante el curso, conocer y emplear nuevas tecnologías, lenguajes de programación, frameworks y herramientas para el desarrollo de aplicaciones las cuales poder desarrollar e implementar en proyectos aplicables en la vida real.

En este caso poder aplicarlo en una instalación, desarrollando un sistema automatizado para poder monitorear y permitir el control de acceso de forma segura por parte de los vehículos que según el reconocimiento de su matrícula estén autorizados a acceder.

Los Objetivos Específicos serán:

- Desarrollar un sistema que permita reconocer y validar las diferentes matrículas de los vehículos.
- Desarrollo de una base de datos en la cual poder almacenar la información de los vehículos los cuales tienen acceso, de sus propietarios y un registro de las entradas y salidas realizadas por estos.
- Desarrollo de una aplicación en la que poder dar de alta y baja tanto a los propietarios como a sus vehículos y en la que poder consultar la información tanto de estos como de sus entradas y salidas.

1.2 ALCANCE

El presente proyecto pertenece al área de Seguridad abordando el Control de Acceso de Vehículos. Se pretende desarrollar un Sistema que permita el ingreso a una instalación de forma automática, mediante el reconocimiento de matrículas de los diferentes vehículos que deseen acceder, empleando para ello en la parte cliente un sistema para el reconocimiento y validación de matrículas y como servidor una aplicación con la que comunicarse para verificar la autenticación de las matrículas y registrarlas.

A la hora de que un vehículo intente acceder, mediante visión artificial se realizará una detección de la matrícula del vehículo, si el vehículo que va acceder esta registrado en la base de datos, entonces se le dará permiso para entrar al parking, en caso contrario se le denegará dicho permiso.

Esta pensado su uso en instalaciones donde se desee tener un control de los vehículos y en donde poder realizar un registro de sus entradas y salidas, pudiéndose implementar en aparcamientos o parking de viviendas, comunidades de vecinos, centros, empresas e instituciones, centros educativos, o incluso para instalaciones privadas como pueden ser la policía o las fuerzas militares.

1.3 JUSTIFICACIÓN

El presente proyecto esta enfocado principalmente en reconocer las diferentes matriculas de vehículos para comprobarlas y permitir su acceso en el caso de que tengan el permiso necesario, pudiéndose para ello emplear en parkings y garajes de diferentes, hogares, comunidades de vecinos, instalaciones, urbanizaciones, centros de trabajo, instituciones, etc.

El sistema desarrollado permitirá monitorizar los movimientos de vehículos que tienen acceso, a través de un control de seguridad, para mejorar la seguridad y de esta forma gestionar las entradas y salidas de los vehículos pudiendo de esta forma saber cuando entró y salió un vehículo determinado.

El sistema al tener un registro de las entradas y salidas realizadas por los diferentes vehículos permitirá poder facilitar la búsqueda de los vehículos en caso de robo, pudiéndose saber en que momento lo sacaron según el registro y de esta forma poder comprobar por las cámaras de seguridad quién lo robó.

También puede utilizarse para facilitar más a los propietarios de los vehículos el acceso a su hogar o a su trabajo en caso de que a estos se les olviden las llaves del garaje en el trabajo o en casa pudiendo acceder sin necesidad de las mismas y sin tener que preocuparse por ellas.

En caso de que al propietario le llegue alguna multa y resulte que en dicha fecha tenia el vehículo estacionado en el aparcamiento, el propietario podría recurrir la multa pidiendo al encargado el registro de sus entradas y salidas en una fecha determinada.

Pero sobre todo el sistema esta pensado para restringir el acceso de los vehículos a un aparcamiento, teniendo control tanto de las entradas y salidas realizadas por los vehículos con permiso.

2 IMPLEMENTACIÓN

2.1 ANÁLISIS DE LA APLICACIÓN

El presente proyecto consta de dos partes, una parte de cliente y otra parte de servidor.

El cliente será el encargado de realizar el reconocimiento de las matrículas, validándolas y realizando peticiones al servidor para saber si dichas matrículas tienen o no acceso a la instalación. Para ello se ha realizado un programa ejecutable que conectado a una cámara permita el reconocimiento de matrículas.

El servidor se encargará de obtener las matrículas enviadas por el cliente para cotejarlas con la base de datos y verificarlas, una vez hecho le devuelve una respuesta al cliente y en caso de que la matrícula este cotejada se registrará su entrada o salida correspondiente. Para ello se ha realizado una aplicación web que permita consultar las entradas y salidas realizadas, y los vehículos y propietarios registrados.

Windows

Tanto la parte del cliente como la parte del servidor se han desarrollado con el sistema operativo de Windows 10, por ende el proyecto tendrá total compatibilidad con aquellos equipos con un sistema operativo de Windows. Esto no quiere decir que no se pueda usar en otros sistemas operativos ya que el cliente mientras el sistema operativo tenga instalado Python y sus librerías necesarias se podrá ejecutar correctamente y en cuanto al servidor, este al realizarse en .Net Core 5 es compatible con Linux y Mac, aunque para ello habrá que tener instalado las herramientas necesarias para su funcionamiento.

Todas las herramientas y software empleados para el desarrollo del proyecto son Open-Source (código abierto), con lo cual no ha sido necesario ningún tipo de licencia para su desarrollo.

A continuación se explicarán las tecnologías, framework, lenguajes y herramientas más adecuadas y empleadas para el desarrollo del presente proyecto, tanto en la parte del cliente como en la del servidor, indicando sus funcionalidades y comentando cada una de las tecnologías, explicando también donde se van a encontrar y usar tanto el cliente como el servidor.

2.1.1 CLIENTE

El cliente se ha desarrollado con el lenguaje de programación Python, empleando para ello el entorno de desarrollo de Microsoft, Visual Studio Code, para ello se han hecho uso de diferentes librerías de Python necesarias para la realización del cliente.

El cliente esta pensado para utilizarse en un equipo situado en la caseta de vigilancia la cual suele encontrarse en la entrada del aparcamiento o zona en la que se desee acceder, para así realizar el control de acceso a una instalación. El cliente deberá estar conectado a una cámara con la que poder captar los diferentes vehículos que deseen acceder, para ello se proporcionará un programa .exe el cual al ejecutarse en el equipo situado en la garita se conectará a la cámara y empezará a realizar el reconocimiento de las matrículas proporcionando visual de la cámara para ver los diferentes vehículos que deseen acceder.

En caso de que el vehículo tenga acceso se registrará su entrada o salida en la base de datos y se mostrará una notificación en el equipo indicando que ha accedido un vehículo mostrando su matrícula.

En caso de que el vehículo no tenga acceso no se realizará ningún registro de la entrada o salida y se mostrará una notificación en el equipo indicando que el acceso ha sido denegado mostrando la matrícula del vehículo que ha querido acceder.

Visual Studio Code

Como entorno de desarrollo integrado se ha optado por emplear el editor de código fuente de Microsoft Visual Studio Code el cual tiene soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código, haciéndolo un Ide muy completo y perfecto para programar en lenguajes como puede ser Python.

Para poder usar Visual Studio Code con Python antes es necesario tener el compilador de Python instalado y descargar la extensión de Python de Visual Studio Code.

Este editor de código es muy recomendado para programar en Python ya que a parte de ser ligero, fácil de usar y gratuito, proporciona algunas ventajas como pueden ser el coloreado del código para diferenciar los distintos elementos del lenguaje, IntelliSense para ayudar con los errores y otros elementos del código, la posibilidad de ejecutar código en Python observando el resultado en el terminal y poder hacer uso de este para la ejecución de comandos

Python

Para la realización del cliente en el presente proyecto se ha escogido como lenguaje de programación Python en su versión 3.8.2 por ser el lenguaje con mayor documentación y librerías especializadas para el procesamiento de imágenes y para visión artificial necesarias para el desarrollo del cliente, también es un buen lenguaje para la realización de scripts y programas, a parte de ser un lenguaje de programación multiplataforma e interpretado más sencillo y limpio que otros. Es necesaria una versión igual o mayor a Python 3.7 siendo necesario para la compatibilidad con algunas librerías como puedan ser Pytesseract. También se podría desarrollar el cliente en otros lenguajes como pueden ser Java, C++ o incluso en Matlab ya que estos tienen compatibilidad con algunas librerías como puede ser OpenCv.

Librerías de Python utilizadas

Cv2: es una librería libre de visión artificial originalmente desarrollada por Intel, que contiene funciones para el procesado de imagen. En este proyecto, dicha librería ha sido utilizada para el procesado de las imágenes capturadas por la cámara, para posteriormente procesarlas con filtros.

Para poderla usar se necesita instalar el paquete de OpenCV

Pytesseract: se empleará como motor de reconocimiento óptico de caracteres (OCR), ya que es uno de los mejores motores OCR actuales y está desarrollado por Google. Para poder utilizar tesseract en Python se necesita instalar el paquete de pytesseract.

Pueden producirse errores en el reconocimiento por la distancia entre caracteres o la conexión de algunos píxeles de diferentes caracteres.

Re: es una librería de Python que permite realizar expresiones regulares. Se ha utilizado para verificar si es una matrícula comprobando que empiece por una letra mayúscula seguida por cuatro números y termine con dos letras o que empiece por cuatro números y termine con tres letras en mayúsculas.

Time: permite la gestión del tiempo en Python. Dicha librería ha sido utilizada en este proyecto para realizar una espera de tiempo al detectar un vehículo con una matrícula verificada y validada, dando tiempo al vehículo para acceder y no seguir reconociendo la matrícula mientras esté entrando o saliendo.

Request: para realizar las consultas a una API con Python. Es empleada para realizar las peticiones al servidor una vez se obtenga la matrícula para cotejarla con la base de datos y obtener una respuesta.

Notify: es una librería del paquete Notifypy que permite crear notificaciones para cualquier sistema operativo. Es utilizado para notificar en el equipo una vez obtenida la respuesta del servidor los accesos permitidos y denegados junto con la matrícula del vehículo que se quiere acceder y la hora de acceso.

2.1.2 Servidor

El servidor se ha desarrollado con el lenguaje de programación C# para el back-end y HTML para el front-end, empleando para ello la plataforma de desarrollo de .NET Core 5 y utilizándose como entorno de desarrollo Microsoft Visual Studio 2022, para ello se ha hecho uso de los diferentes paquetes NuGets necesarios para la realización del servidor. También se ha usado SQL como lenguaje de base de datos y utilizando como sistema de gestión de bases de datos SQL Server.

El servidor esta pensado para utilizarse en un equipo situado en la zona de servidores de la instalación y conectado en la intranet mediante el Internet Information Service (IIS), para poder acceder a el desde cualquier equipo conectado a la Intranet siempre y cuando se tengan las credenciales necesarias para acceder. También podría estar en la caseta de vigilancia junto al programa de cliente aunque no seria lo recomendable, siendo lo correcto que el encargado de la caseta se conecte al servidor de forma remota.

Para acceder al servidor antes será necesario iniciar sesión, para ello se proporcionará un usuario con una contraseña por defecto, pudiendo posteriormente crear otros usuarios con dos roles diferente administrador y usuario, siendo estos los únicos con acceso al servidor.

Los administradores (admin) tendrá acceso a todo, desde consultar, crear y eliminar tanto los vehículos, propietarios y usuarios como entradas y salidas.

Por el contrario los usuarios (user) solo tendrán acceso a consultar, crear y eliminar vehículos y propietarios.

El cliente al estar conectado en la intranet, cualquiera conectado puede acceder, para evitarlo se han protegido cada ruta mediante autenticación, es decir, si al acceder al servidor no te has autenticado iniciando sesión no podrás acceder a ninguna de las rutas del servidor.

Visual Studio 2022

Como entorno de desarrollo integrado para la realización de la parte del cliente se a optado por emplear el entorno de desarrollo integrado de Microsoft Visual Studio en su versión más reciente 2022 y usando la edición Community siendo esta muy completa y gratuita. Se ha escogido Visual Studio ya que se puede usar para editar, depurar y compilar código y, después, publicar una aplicación, siendo uno de los IDE más completo para el desarrollo de software por todas sus características y permitir desarrollar aplicaciones y servicios web multiplataforma de ASP.NET Core, aparte de ser el mejor IDE para programar en C#.

También se ha escogido este entorno de desarrollo por haberse utilizado en la FCT realizada.

En conjunto con Visual Studio se ha utilizado el framework de .NET 5 y se ha programado en C#.

.Net 5

Como entorno de trabajo se a utilizado la plataforma de .NET Core en su versión 5, denominada .NET 5, el cual es un framework informático administrado, gratuito y de código abierto para los sistemas operativos Windows, Linux y macOS.

Se ha escogido .Net 5 y no .NET Framework u otras versiones ya que a parte de haber sido utilizada en la realización de las FCT, también es la versión más completa y estable, la cual añade cambios y mejoras sobre sus versiones anteriores.

Este framework ha sido utilizado para la realización de una aplicación web API multiplataforma con controladores y vistas de ASP.NET Core MVC, creando también un servicio RESTful HTTP como API para obtener la peticiones del cliente, hacer las consultas a la base de datos y enviar una respuesta al cliente.

C#

El entorno de desarrollo Visual Studio junto con la plataforma de .NET utilizan los leguajes de C#, F# y Visual Basic, siendo C# el más utilizado y popular.

C# es un lenguaje de programación multiparadigma desarrollado y estandarizado por Microsoft como parte de la plataforma .NET. Es uno de los lenguajes de programación diseñado para la infraestructura de lenguaje común, y para compilar diversas aplicaciones que se ejecutan dentro de la familia de .NET. Es un lenguaje de programación moderno, simple, eficaz y con tipado, a parte de ser orientado a objetos.

Es un lenguaje que siempre se encuentra entre los 5 lenguajes de programación más utilizados.

Es el lenguaje utilizado para la realización de toda la parte de back-end del servidor.

Razor

Para la realización de la parte front-end del servidor se han empleado diferentes vistas Razor que son las vistas utilizadas en las aplicaciones de ASP.NET Core MVC. La vista se encarga de la presentación de los datos y de la interacción del usuario haciendo uso de los métodos del controlador correspondiente.

Es usado para ello un fichero .cshtml que permite añadir código C# dentro del marcado del HTML.

SQL Server

Microsoft SQL Server es un sistema de gestión de base de datos relacional, desarrollado por Microsoft.

El lenguaje de desarrollo utilizado es Transact-SQL, el cual es una implementación del estándar ANSI del lenguaje SQL, utilizado para manipular y recuperar datos, crear tablas y definir relaciones entre ellas.

Es el sistema de base de datos empleado para la realización del proyecto, usándose la edición desarrollador 2022, y en donde se almacenará toda la información correspondiente en las tablas de Propietario, Vehículo, Entradas, Salidas y Usuarios, para poderla consultar y hacer los CRUD necesarios.

IIS

IIS (Internet Information Server) es un servidor web extensible que provee un conjunto de servicios para sistemas operativos de Windows. Esta característica permite convertir una máquina en un servidor web para poder publicar un sitio web en Internet o en una red interna.

Es donde se realizará la publicación del servidor para poderse usar dentro de la Intranet de la instalación.

2.2 DISEÑO

Diagrama de la Base de Datos del Control de Acceso de Vehículos

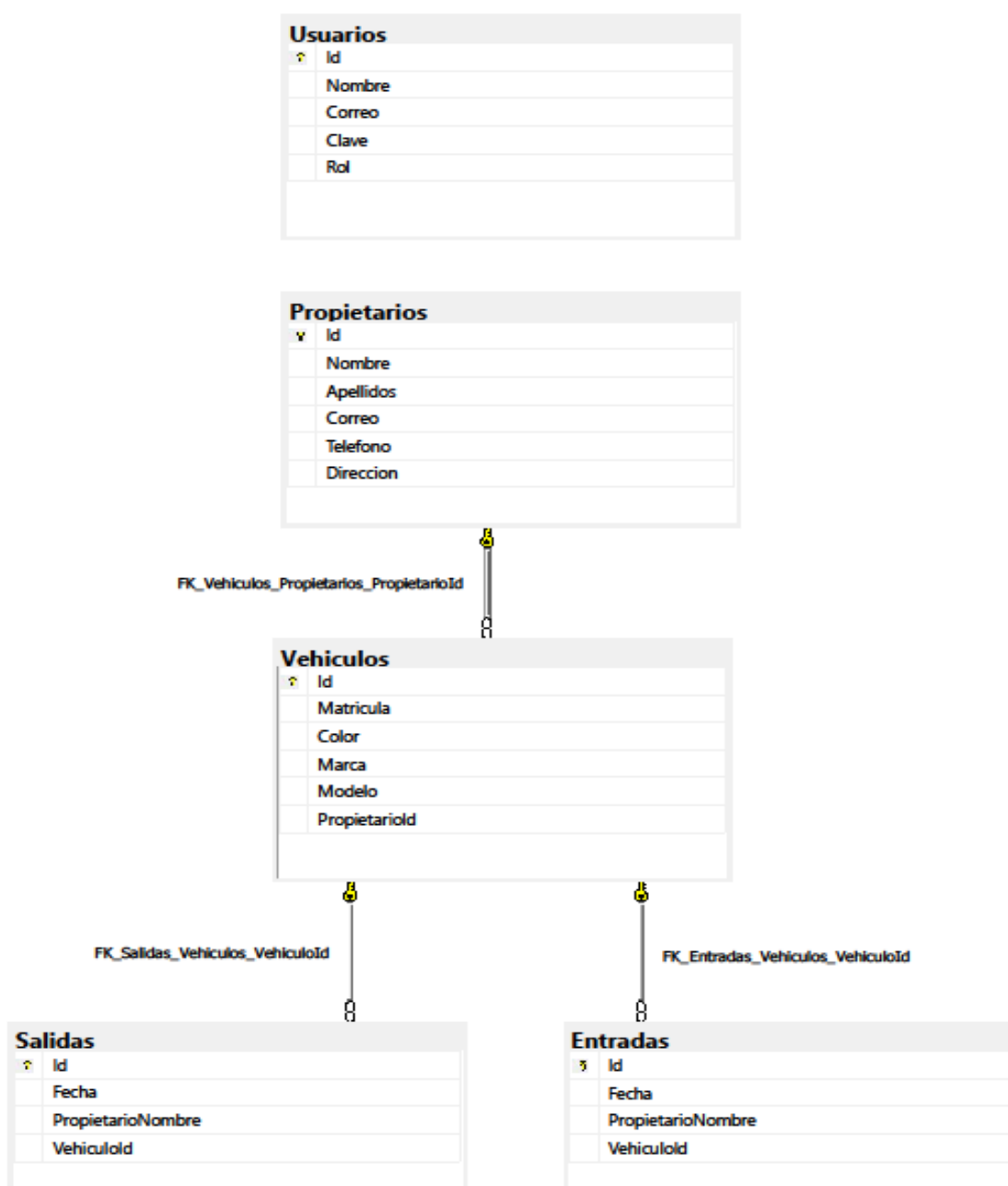
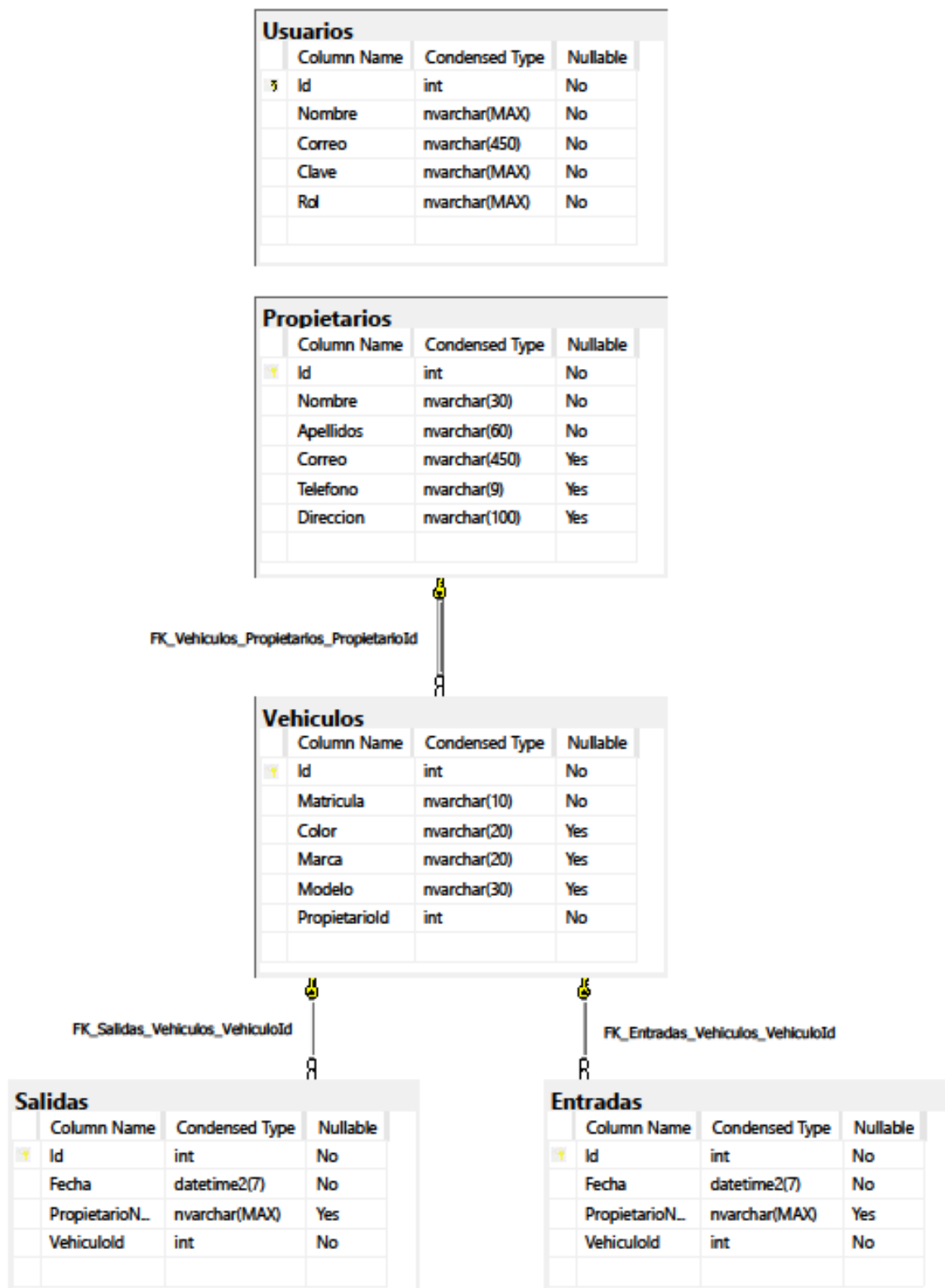


Diagrama de la Base de Datos del Control de Acceso de Vehículos



CONTROLADORES

EntradaController

Clase
 ↗ Controller

Campos

- _context
- log

Métodos

- Create (+ 1 sobrecarga)
- Delete
- Details
- EntradaController
- Entrar
- Exportar_Excel
- Index

SalidaController

Clase
 ↗ Controller

Campos

- _context
- log

Métodos

- Create (+ 1 sobrecarga)
- Delete
- Details
- Exportar_Excel
- Index
- SalidaController
- Salir

HomeController

Clase
 ↗ Controller

Campos

- _logger

Métodos

- Error
- HomeController
- Index

PropietarioController

Clase
 ↗ Controller

Campos

- _context

Métodos

- Create (+ 1 sobrecarga)
- DeleteConfirmed
- Details
- Edit (+ 1 sobrecarga)
- Exportar_Excel
- Index
- PropietarioController
- PropietarioExists

VehiculoController

Clase
 ↗ Controller

Campos

- _context

Métodos

- Create (+ 1 sobrecarga)
- DeleteConfirmed
- Details
- Edit (+ 1 sobrecarga)
- Exportar_Excel
- Index
- VehiculoController
- VehiculoExists

UsuarioController

Clase
 ↗ Controller

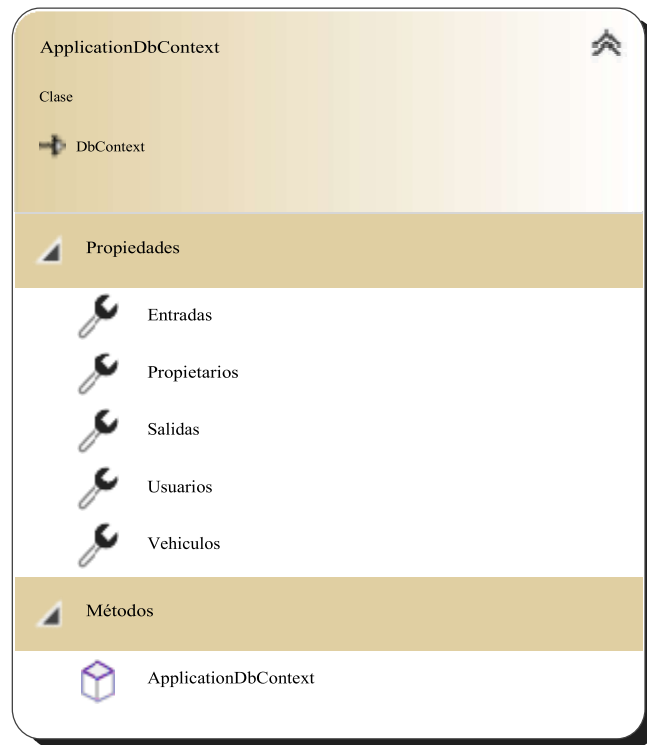
Campos

- _context

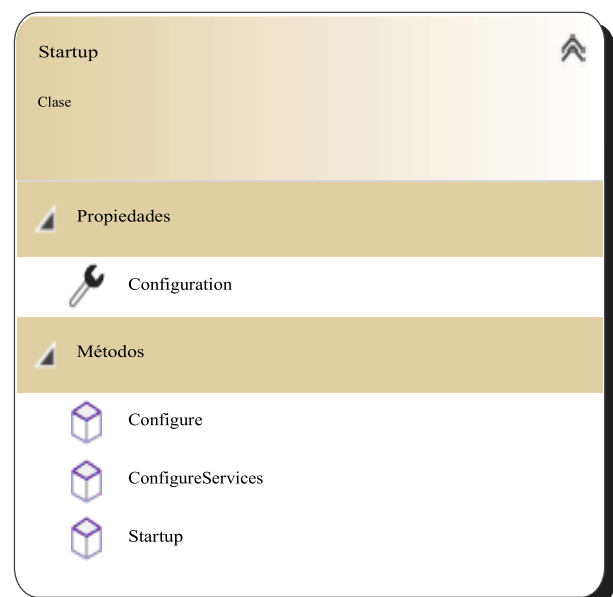
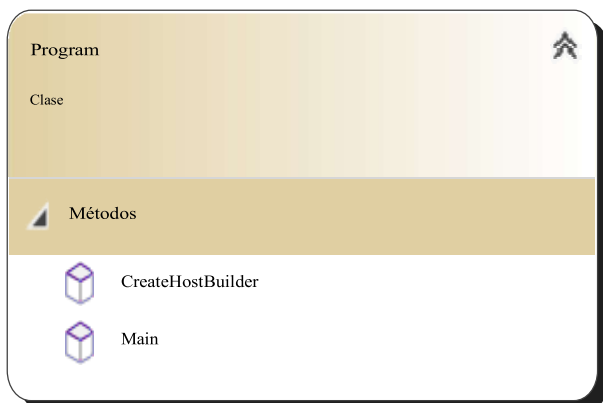
Métodos

- ConvertirSha256
- Create
- CreateAsync
- DeleteConfirmed
- Details
- Index
- Login (+ 1 sobrecarga)
- Salir
- UsuarioController
- ValidarUsuario

CONTEXTO DE LA BASE DE DATOS



PROGRAM Y STARTUP



ENTIDADES

Entrada

Clase

▲ Propiedades

- Fecha
- Id
- PropietarioNombre
- Vehiculo
- Vehiculold

Salida

Clase

▲ Propiedades

- Fecha
- Id
- PropietarioNombre
- Vehiculo
- Vehiculold

ErrorViewModel

Clase

▲ Propiedades

- RequestId
- ShowRequestId

Propietario

Clase

▲ Propiedades

- Apellidos
- Correo
- Direccion
- Id
- Nombre
- NombreCompleto
- Telefono

Vehiculo

Clase

▲ Propiedades

- Color
- Id
- Marca
- Matricula
- Modelo
- Propietario
- Propietariold

Usuario


Clase

▲ Propiedades


- Clave
- Correo
- Id
- Nombre
- Rol


LOG Y PAGINACIÓN


Log
Clase




Métodos

 EntradaLog


 Log


 SalidaLog


Pager
Clase





Propiedades


 CurrentPage

 EndPage


 PageSize

 StartPage

 TotalItems

 TotalPages

Métodos

 Pager (+ 1 sobrecarga)

2.3 IMPLEMENTACIÓN

2.3.1 CLIENTE

El cliente será el encargado de obtener el vídeo de una cámara para procesar cada fotograma y así obtener la matrícula, validarla y enviársela al servidor para comprobarla con la base de datos.

Para ello lo primero será conectar al equipo en el cual se encuentre el cliente y desde el cual se vaya a ejecutar, una [cámara](#) con la cual poder visualizar los diferentes vehículos que deseen acceder, obteniendo una captura en vídeo con la cual el programa cliente se encargará de almacenar cada uno de los fotogramas captados por la cámara para poder [procesarlos y realizar los diferentes tipos de filtrado o emplear en su lugar un clasificador en cascada](#).

Una vez procesada y tratada la imagen se procede a calcular tanto el contorno como el área donde se encuentra la matrícula, pudiendo de esta forma calcular la relación de aspecto obtenido así en la imagen el rectángulo que forma la matrícula en caso de que se encuentre en la imagen.

Tras haber obtenido el rectángulo que conforma la matrícula se procede a obtener el texto de la matrícula mediante el reconocimiento óptico de caracteres, intentando sacar los caracteres de la matrícula.

En cuanto el texto obtenido de la matrícula tenga más de siete caracteres se procede a validar dicho texto para comprobar que la matrícula sea válida, para ello se comprueba que empiece por cuatro número y termine por tres letras o que empiece por una letra con cuatro números en el medio y termine por dos letras, estas matrículas corresponden a España, en otros países la validación será diferente.

A continuación se realiza una petición al servidor pasándole la matrícula validada y según la respuesta recibida por el servidor el cliente mostrará una notificación con el estado de la matrícula notificando si el acceso a sido permitido o denegado según la respuesta recibida por el servidor. Solo en caso de que la matrícula tenga acceso permitido el programa se pausará durante un minuto para dejar de buscar matrículas hasta que el vehículo pase el control, en caso contrario, seguirá buscando matrículas.

Todo este proceso lo realiza el cliente continuamente hasta que el usuario pulse sobre la tecla de escape(ESC) o desconecte el equipo en el que se encuentre el cliente, finalizando así el programa.

A continuación se hablará sobre los métodos más importantes de cada librería que se han empleado para la realización del cliente.

OpenCV

- **VideoCapture(0):** captura el vídeo con una cámara para almacenar cada frame para procesarlo.
- **cvtColor:** `dst = cv2.cvtColor(src, flag)` Convierte una imagen `src` de un espacio de color a otro y guarda dicha conversión en la variable `dst`. El segundo parámetro indica el tipo de conversión `cv2.COLOR_BGR2GRAY` de RGB a gris.
- **threshold:** `res, dst = cv2.threshold(src, thresh, maxval, type)` Guarda en la variable `dst` la binarización de una imagen `src`. Los píxeles de intensidad mayor a `thresh` toman el valor de intensidad `maxval`. El parámetro `type` indica el tipo de binarización usándose `BINARY` junto con `OTSU`.
- **findContours:** `contours, hierarchy = cv2.findContours(src, mode, method)` Guarda en la variable `contours` los contornos de una imagen `src` utilizando un modo `mode`, y un método `method`. Para ello se ha usado como modo `RETR_TREE` y como método `CHAIN_APPROX_SIMPLE`.
- **contourArea:** `area = cv2.contourArea(contour)` Guarda en la variable `area` el área encerrada en un determinado contorno `contour`.
- **boundingRect:** `x, y, w, h = cv2.boundingRect(cnt)` Guarda en las variables `x, y, w, h` las coordenadas del rectángulo mínimo que encierra a un determinado contorno `cnt`. Las variables `x` e `y` hacen referencia a las coordenadas de la esquina superior izquierda del rectángulo y `w, h` son el largo y alto del rectángulo respectivamente.

Pytesseract:

- **image_to_string:** `string = pytesseract.image_to_string(img)` Realiza el reconocimiento óptico de caracteres de una imagen `img`, y los resultados son guardados en una cadena de texto `string`.

Re:

- **re.search:** `re.search(regex, src)` Mediante una expresión regular `regex` válido que dicha matrícula `src` sea correcta en España comprobando que empiece por cuatro números y termine por tres letras o que empiece por una letra con cuatro números en el medio y termine por dos letras.

Request:

- **request.get:** `res = request.get(url, verify)` Se realiza una llamada a la api pasándole la matrícula validada y según la respuesta obtenida del servidor recibirá una notificación indicando el estado de acceso.

Notify:

Permite mostrar notificaciones en el escritorio del equipo pudiendo mostrarlas con un título, mensaje y logo. Si la respuesta del servidor es correcta se mostrará una notificación indicando acceso permitido con la matrícula correspondiente, en caso contrario mostrará acceso denegado junto con la matrícula.

Time:

- **sleep:** `time.sleep(seg)` Retardo de `seg` segundos. En caso de que la respuesta del servidor sea correcta el programa se parará durante 60 segundos para dejar de buscar matrículas.

2.3.2 SERVIDOR

El servidor será el encargado de comprobar las matriculas en la base de datos, realizar el registro tanto de las entradas como de las salidas y gestionar los vehículos con acceso permitido y sus propietarios.

El servidor esta desarrollado con una aplicación web de ASP.NET Core (Modelo-Vista-Controlador), en donde la capa modelo corresponde con las clases asociadas a las tablas de la base de datos con su información, la capa de vista corresponde con las vistas que se mostraran en la aplicación web y la capa de controlador será la encargada de comunicar los modelos con las vistas obteniendo los datos de los modelos desde la base de datos para tratarlos y pasarlos a la vista y al contrario obtener los datos ingresados en las vistas para insertarlos en las tablas correspondientes de la base de datos.

Modelos:

Como se ha mencionado en esta capa se encuentran las clases correspondientes con las tablas de la base de datos, en el apartado de [Diseño](#) en Entidades se puede observar como están formadas.

```
namespace ControlAccesoVehiculosAdrianLozano.Models
{
    [Index(nameof(Matricula), IsUnique = true)]
    29 referencias
    public class Vehiculo
    {
        [Key]
        16 referencias
        public int Id { get; set; }

        [Required(ErrorMessage = "Se necesita la matrícula del Vehículo")]
        [StringLength(10)]
        [Display(Name="Matrícula")]
        29 referencias
        public string Matricula { get; set; }

        [StringLength(20)]
        15 referencias
        public string Color { get; set; }

        [StringLength(20)]
        20 referencias
        public string Marca { get; set; }

        [StringLength(30)]
        20 referencias
        public string Modelo { get; set; }

        [Required(ErrorMessage = "Se necesita el nombre del Propietario")]
        12 referencias
        public int PropietarioId { get; set; }

        [ForeignKey("PropietarioId")]
        35 referencias
        public Propietario Propietario { get; set; }
    }
}
```

Controladores:

Se podría decir que es la capa más importante y en la que se encuentra la lógica del programa. Se ha creado un controlador por cada entidad, estando asociado cada uno de los controladores con su modelo correspondiente. Serán los encargados de comunicar la capa de modelos con la de las vistas y viceversa. Algunos de los métodos utilizados han sido:

Index: el cual es el método encargado de devolver la vista principal con todos los datos obtenidos del modelo asociado, incorporando el filtrado y paginación correspondientes para emplearlas en la vista.

```
/// <summary>
///Método que devuelve una vista con todos los vehículos añadiendo filtros y paginación
/// GET: Vehiculo
/// </summary>
/// <param name="matricula"></param>
/// <param name="pg"></param>
/// <returns>Vista con todos los vehículos</returns>
4 referencias
public async Task<IActionResult> Index(string matricula, int pg = 1)
{
    //Filtrado por matricula y nombre
    ViewData["BuscarMatricula"] = matricula;
    var vehiculo = from v in _context.Vehiculos.Include(v => v.Propietario) select v;

    if (!String.IsNullOrEmpty(matricula))
    {
        vehiculo = vehiculo.Where(v => v.Matricula.Equals(matricula)
            || (v.Propietario.Nombre + " " + v.Propietario.Apellidos).Equals(matricula));
    }

    //Paginación
    List<Vehiculo> vehiculos = await vehiculo.AsNoTracking().ToListAsync();
    const int pageSize = 10;
    if (pg < 1)
        pg = 1;
    int recsCount = vehiculos.Count();
    var pager = new Pager(recsCount, pg, pageSize);
    int recSkip = (pg - 1) * pageSize;
    var data = vehiculos.Skip(recSkip).Take(pager.PageSize).ToList();
    this.ViewBag.Pager = pager;
    return View(data);
}
```

Details: es el método que devolverá la vista con los datos de un elemento del modelo correspondiente.

```
/// <summary>
/// Método para obtener los detalles de un vehículo devolviendo la vista correspondiente
/// GET: Vehiculo/Details/ID
/// </summary>
/// <param name="id"></param>
/// <returns>Vista con los detalles de un vehículo</returns>
1 referencia
public async Task<ActionResult> Details(int? id)
{
    if (id == null)
    {
        return NotFound();
    }

    var vehiculo = await _context.Vehiculos
        .Include(v => v.Propietario)
        .FirstOrDefaultAsync(m => m.Id == id);
    if (vehiculo == null)
    {
        return NotFound();
    }

    return View(vehiculo);
}
```

Create: permite crear un nuevo elemento del modelo asociado con los valores correspondientes.

```
/// <summary>
/// Método para realizar la creación de un vehículo devolviendo la vista y almacenando los datos
/// POST: Vehiculo/Create
/// </summary>
/// <param name="vehiculo"></param>
/// <returns>Vehículo creado</returns>
[HttpPost]
[ValidateAntiForgeryToken]
1 referencia
public async Task<ActionResult> Create([Bind("Id,Matricula,Color,Marca,Modelo,PropietarioId")] Vehiculo vehiculo)
{
    if (ModelState.IsValid)
    {
        _context.Add(vehiculo);
        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }
    ViewData["PropietarioId"] = new SelectList(_context.Propietarios, "Id", "NombreCompleto", vehiculo.PropietarioId);
    return View(vehiculo);
}
```


Edit: solo disponible en Vehículos y Propietarios para editar los valores de un elemento en concreto.

```
/// <summary>
/// Método para editar los campos de un registro de un vehículo
/// POST: Vehiculo/Edit/ID
/// </summary>
/// <param name="id"></param>
/// <param name="vehiculo"></param>
/// <returns>Vehículo editado</returns>
[HttpPost]
[ValidateAntiForgeryToken]
0 referencias
public async Task<IActionResult> Edit(int id, [Bind("Id,Matricula,Color,Marca,Modelo,PropietarioId")] Vehiculo vehiculo)
{
    if (id != vehiculo.Id)
    {
        return NotFound();
    }

    if (ModelState.IsValid)
    {
        try
        {
            _context.Update(vehiculo);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!VehiculoExists(vehiculo.Id))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    ViewData["PropietarioId"] = new SelectList(_context.Propietarios, "Id", "NombreCompleto", vehiculo.PropietarioId);
    return View(vehiculo);
}
```

Delete: permite eliminar un elemento del modelo asociado o en caso de Entradas y Salidas todos.

```
/// <summary>
/// Método para relizar el borrado de un vehículo
/// POST: Vehiculo/Delete/ID
/// </summary>
/// <param name="id"></param>
/// <returns>Confirmación para eliminar un vehículo</returns>
[HttpPost, ActionName("Delete")]
[ValidateAntiForgeryToken]
1 referencia
public async Task<IActionResult> DeleteConfirmed(int id)
{
    var vehiculo = await _context.Vehiculos.FindAsync(id);
    _context.Vehiculos.Remove(vehiculo);
    await _context.SaveChangesAsync();
    return RedirectToAction(nameof(Index));
}
```

Exportar_Excel: es el método encargado de exportar los datos de la entidad asociada en un fichero Excel.

```
/// <summary>
/// Método para exportar los vehículos en un archivo Excel
/// </summary>
/// <returns>Excel con todos los vehículos</returns>
0 referencias
public IActionResult Exportar_Excel()
{
    DataTable tabla_vehiculos = new DataTable();
    var vehiculo = from v in _context.Vehiculos.Include(v => v.Propietario) select v;
    List<Vehiculo> vehiculos = vehiculo.ToList();

    //Meter elementos en el datatable
    tabla_vehiculos.Columns.AddRange(new DataColumn[5] {
        new DataColumn("Matricula"),
        new DataColumn("Color"),
        new DataColumn("Marca"),
        new DataColumn("Modelo"),
        new DataColumn("Propietario") });

    foreach (var lista in vehiculos)
    {
        tabla_vehiculos.Rows.Add(lista.Matricula, lista.Color, lista.Marca, lista.Modelo, lista.Propietario.NombreCompleto);
    }

    using (var libro = new XLWorkbook())
    {
        tabla_vehiculos.TableName = "Vehiculos";
        var hoja = libro.Worksheets.Add(tabla_vehiculos); //Creo la hoja Excel
        hoja.ColumnsUsed().AdjustToContents();

        using (var memoria = new MemoryStream()) //Memoria a exportar
        {
            libro.SaveAs(memoria);
            var nombreExcel = string.Concat("Reporte Vehiculos ", DateTime.Now.ToString(), ".xlsx");

            return File(memoria.ToArray(), "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet", nombreExcel);
        }
    }
}
```

Entrar: método correspondiente con Entradas el cual es el empleado por el cliente para realizar el registro y comprobación de las matrículas para realizar el registro de entradas, comprobando la matrícula en la base de datos y devolviendo una respuesta según el resultado.

```
/// <summary>
/// Método empleado por el cliente para realizar el registro y comprobación de las matrículas para
/// realizar el registro de entradas, comprobando la matrícula en la base de datos y devolviendo una respuesta
/// GET: Entrada/Entrar/Matricula
/// </summary>
/// <param name="id"></param>
/// <param name="entrada"></param>
/// <returns>OK o Not Found según la respuesta</returns>
[HttpGet]
1 referencia
public async Task<IActionResult> Entrar(string? id, [Bind("Id, Fecha, PropietarioNombre, VehiculoId")] Entrada entrada)
{
    if (id == null)
    {
        return NotFound();
    }

    //Obtengo el vehículo
    var vehiculo = await _context.Vehiculos
        .Include(v => v.Propietario)
        .FirstOrDefaultAsync(m => m.Matricula == id);

    if (vehiculo == null)
    {
        log.EntradaLog("Acaba de intentar acceder el vehículo con matrícula " + id.ToString(), false);
        return NotFound();
    }

    //Se realiza el Insert en la Tabla Entradas
    if (vehiculo != null)
    {
        entrada.Fecha = DateTime.Now;
        entrada.PropietarioNombre = vehiculo.Propietario.Nombre + " " + vehiculo.Propietario.Apellidos;
        entrada.VehiculoId = vehiculo.Id;
        entrada.Vehiculo = vehiculo;

        _context.Add(entrada);
        await _context.SaveChangesAsync();
        log.EntradaLog("Acaba de acceder el vehículo con matrícula " + vehiculo.Matricula.ToString() + " perteneciente al propietario " + entrada.PropietarioNombre.ToString(), true);
    }

    return Json(entrada);
}
```

Salir: método correspondiente con Salidas el cual es el empleado por el cliente para realizar el registro y comprobación de las matriculas para realizar el registro de salidas, comprobando la matrícula en la base de datos y devolviendo una respuesta según el resultado.

```
/// <summary>
/// Método empleado por el cliente para realizar el registro y comprobación de las matriculas para
/// realizar el registro de salidas, comprobando la matrícula en la base de datos y devolviendo una respuesta
/// GET: Salida/Salir/Matricula
/// </summary>
/// <param name="id"></param>
/// <param name="salida"></param>
/// <returns>OK o Not Found según la respuesta</returns>
[HttpGet]
public async Task<IActionResult> Salir(string? id, [Bind("Id, Fecha, PropietarioNombre, VehiculoId")] Salida salida)
{
    if (id == null)
    {
        return NotFound();
    }

    //Obtengo el vehiculo
    var vehiculo = await _context.Vehiculos
        .Include(v => v.Propietario)
        .FirstOrDefaultAsync(m => m.Matricula == id);

    if (vehiculo == null)
    {
        log.SalidaLog("Acaba de intentar salir el vehiculo con matricula " + id.ToString(), false);
        return NotFound();
    }
    //Se realiza el Insert en la Tabla Entradas
    if (vehiculo != null)
    {
        salida.Fecha = DateTime.Now;
        salida.PropietarioNombre = vehiculo.Propietario.Nombre + " " + vehiculo.Propietario.Apellidos;
        salida.VehiculoId = vehiculo.Id;
        salida.Vehiculo = vehiculo;

        _context.Add(salida);
        await _context.SaveChangesAsync();
        log.SalidaLog("Acaba de salir el vehiculo con matricula " + vehiculo.Matricula.ToString() + " perteneciente al propietario " + salida.PropietarioNombre.ToString(), true);
    }

    return Json(salida);
}
```

Login: método empleado en Usuarios para realizar el logado de la página y comprobar la autenticación y autorización del usuario que va acceder según su correo y clave introducidas.

```
/// <summary>
/// Método para realizar el logado de la página y comprobar la autenticación y autorización
/// </summary>
/// <param name="_usuario"></param>
/// <returns>Logado de usuario</returns>
[HttpPost]
public async Task<IActionResult> Login(Usuario _usuario)
{
    var user = ValidarUsuario(_usuario.Correo, _usuario.Clave);

    List<Usuario> usuarioList = await _context.Usuarios.ToListAsync();

    Usuario usuario = null;

    foreach (Usuario u in usuarioList)
    {
        if (u.Correo == user.Correo && u.Clave == user.Clave)
        {
            usuario = u;
        }
    }

    if (usuario != null)
    {
        //CONFIGURACION DE LA AUTENTICACION
        #region AUTENTICACION
        var claims = new List<Claim>
        {
            new Claim(ClaimTypes.Name, usuario.Nombre),
            new Claim("Correo", usuario.Correo),
            new Claim(ClaimTypes.Role, usuario.Rol)
        };

        var claimsIdentity = new ClaimsIdentity(claims, CookieAuthenticationDefaults.AuthenticationScheme);

        await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, new ClaimsPrincipal(claimsIdentity));
        #endregion

        return RedirectToAction("Index", "Home");
    }
    else
    {
        return View();
    }
}
```

Salir: este método correspondiente a Usuarios permitirá realizar el logout del usuario que inicio sesión.

```
/// <summary>
///Método para salir de la sesión
/// </summary>
/// <returns>Vista de login</returns>
0 referencias
public async Task<IActionResult> Salir()
{
    //CONFIGURACION DE LA AUTENTICACION
    #region AUTENTICACION
    await HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);
    #endregion

    return RedirectToAction("Login", "Usuario");
}
```

ValidarUsuario: método de Usuario para la validación del usuario pasando la contraseña introducida a SHA256 para una mayor seguridad y así almacenar en la base de datos la clave cifrada.

```
/// <summary>
///Método para realizar la validación de los usuarios
/// </summary>
/// <param name="correo"></param>
/// <param name="clave"></param>
/// <returns>Usuario</returns>
1 referencia
public static Usuario ValidarUsuario(string correo, string clave)
{
    Usuario usuario = new Usuario();
    usuario.Correo = correo;
    usuario.Clave = ConvertirSha256(clave);
    return usuario;
}

/// <summary>
///Método para convertir la clave en SHA256 para almacenarla en la base de datos
/// </summary>
/// <param name="texto"></param>
/// <returns>Clave en SHA256</returns>
2 referencias
public static string ConvertirSha256(string texto)
{
    StringBuilder Sb = new StringBuilder();
    using (SHA256 hash = SHA256Managed.Create())
    {
        Encoding enc = Encoding.UTF8;
        byte[] result = hash.ComputeHash(enc.GetBytes(texto));

        foreach (byte b in result)
            Sb.Append(b.ToString("x2"));
    }

    return Sb.ToString();
}
```

Vistas:

Son la parte visual, es decir la parte web donde se muestra toda la información y en la cual nada más iniciar la página web se pedirá un inicio de sesión, una vez iniciada la sesión se mostrará la ventana principal con una barra superior donde se encontrarán los diferente apartados para gestionar. Estos apartados son:

Iniciar Sesión

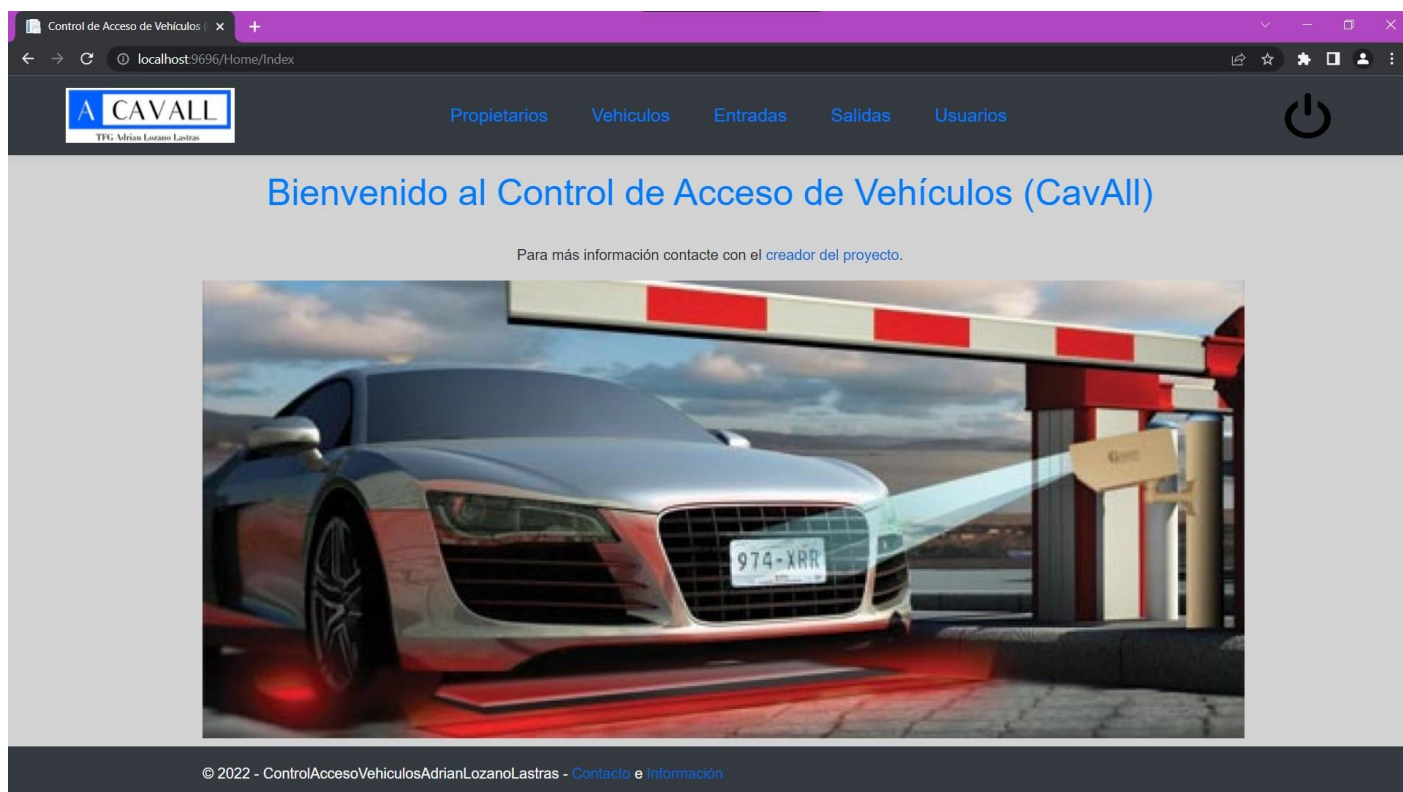
adrianll@admin.com

.....

Acceder

Contacto

Información



Propietarios: permitirá gestionar los diferentes propietarios de los vehículos, pudiendo crear nuevos propietarios con su información correspondiente como puede ser el nombre, apellidos, correo, teléfono y dirección. También se podrá tanto mostrar la información de un propietario como editar sus datos o incluso eliminarlo (en caso de que se elimine un propietario se eliminarán sus vehículos y entradas asociadas). La página donde se muestren los propietarios estará paginada por cada diez propietarios registrados pudiéndolos buscar filtrando por su nombre o filtrando por su nombre y apellidos. A parte también se podrá descargar un archivo Excel con todos los propietarios y sus datos.

Propietarios De Los Vehiculos

Nombre	Apellidos	Correo	Teléfono	Dirección			
Wanda	Alvarez	wanda.alvarez@example.com	687985847	6785 Stevens Creek Blvd	<input type="button" value="Editar"/>	<input type="button" value="Información"/>	<input type="button" value="Eliminar"/>
Suzanne	Stanley	suzanne.stanley@example.com	676786474	5283 College St	<input type="button" value="Editar"/>	<input type="button" value="Información"/>	<input type="button" value="Eliminar"/>
Eli	Morgan	eli.morgan@example.com	685476343	8536 Elgin St	<input type="button" value="Editar"/>	<input type="button" value="Información"/>	<input type="button" value="Eliminar"/>
Paco	Ramirez Segovia	paco@ramirez.com	696758644	Calle de la Piruleta	<input type="button" value="Editar"/>	<input type="button" value="Información"/>	<input type="button" value="Eliminar"/>

Vehículos: en la página de gestión de los vehículos, se mostrará la información de cada uno de los vehículos registrados como puedan ser la matrícula, el color, la marca, el modelo y el propietario al que pertenece el vehículo. Los vehículos se mostrarán de diez en diez con paginación pudiendo buscar un vehículo filtrando por la matrícula de este o por el nombre completo del propietario al que pertenece. También se podrán crear vehículos e incluso mostrar la información, editar los datos o eliminar un vehículo en concreto. A parte se podrá descargar la información de los vehículos en un archivo Excel.

Vehículos Registrados

Matrícula	Color	Marca	Modelo	Propietario			
8915 DWB	Blanco	Citroen	C5	Wanda Alvarez	<input type="button" value="Editar"/>	<input type="button" value="Información"/>	<input type="button" value="Eliminar"/>
2340 LWM	Negro	Seat	Arona	Suzanne Stanley	<input type="button" value="Editar"/>	<input type="button" value="Información"/>	<input type="button" value="Eliminar"/>
0735 GPT	Rojo	Citroen	C7	Paco Ramirez Segovia	<input type="button" value="Editar"/>	<input type="button" value="Información"/>	<input type="button" value="Eliminar"/>
2759 GPR	Blanco	Volkswagen		Paco Ramirez Segovia	<input type="button" value="Editar"/>	<input type="button" value="Información"/>	<input type="button" value="Eliminar"/>
6363 DDK	Negro	Audi	A6	Paco Ramirez Segovia	<input type="button" value="Editar"/>	<input type="button" value="Información"/>	<input type="button" value="Eliminar"/>
4324 BTY	Gris	Citroen		Wanda Alvarez	<input type="button" value="Editar"/>	<input type="button" value="Información"/>	<input type="button" value="Eliminar"/>
7943 KBH	Negro	Toyota		Suzanne Stanley	<input type="button" value="Editar"/>	<input type="button" value="Información"/>	<input type="button" value="Eliminar"/>
3480 HYM	Blanco	Renault		Eli Morgan	<input type="button" value="Editar"/>	<input type="button" value="Información"/>	<input type="button" value="Eliminar"/>
8806 KZS	Azul			Wanda Alvarez	<input type="button" value="Editar"/>	<input type="button" value="Información"/>	<input type="button" value="Eliminar"/>
1234 BCD	Blanco			Suzanne Stanley	<input type="button" value="Editar"/>	<input type="button" value="Información"/>	<input type="button" value="Eliminar"/>

Entradas: se mostrarán las entradas realizadas por los vehículos los cuales tengan permiso junto con la fecha y hora en la que se realizó la entrada y el nombre completo del propietario del vehículo que ha accedido. Pudiéndose también mostrar la información de una entrada en concreto y buscar entradas según un día o mes determinados según un año en concreto e incluso pudiéndose filtrar según el nombre completo del propietario del vehículo que realizó la entrada. Las entradas se mostrarán por cada treinta entradas con paginación para facilitar la búsqueda.

No se permitirá eliminar una entrada en concreto pero esta la posibilidad de eliminar todas las entradas registradas.

En caso de que ocurra algún error en el cliente se podrán crear entradas registrándolas manualmente.

Por último se podrá descargar un fichero Excel con todas las entradas registradas con su información.

Control De Entradas

Fecha	Propietario	Vehículo	
28/05/2022 15:19:39	Wanda Alvarez	8915 DWB	<input type="button" value="Información"/>
28/05/2022 15:22:55	Suzanne Stanley	2340 LWM	<input type="button" value="Información"/>
01/06/2022 19:05:31	Wanda Alvarez	8915 DWB	<input type="button" value="Información"/>
01/06/2022 19:22:59	Paco Ramirez Segovia	0735 GPT	<input type="button" value="Información"/>
01/06/2022 19:25:57	Paco Ramirez Segovia	2759 GPR	<input type="button" value="Información"/>
01/06/2022 19:28:48	Paco Ramirez Segovia	6363 DDK	<input type="button" value="Información"/>
01/06/2022 19:30:25	Wanda Alvarez	4324 BTY	<input type="button" value="Información"/>
03/06/2022 17:44:18	Eli Morgan	3480 HYM	<input type="button" value="Información"/>

Salidas: las salidas serán mostradas de treinta en treinta con paginación, mostrando las salidas realizadas por aquellos vehículos cuyos propietarios tengan permiso, junto a cada salida se mostrará también la fecha y hora correspondientes a la salida y el nombre completo del propietario del vehículo que la realizo. También se permitirá mostrar la información correspondiente de una salida concreta y buscar salidas filtrándolas según el día o mes del año en que se realizo e incluso por el nombre completo del propietario que las realizo.

En caso de que el cliente falle se podrán crear salidas registradas de forma manual completando los campos correspondientes. No se podrá eliminar una salida en concreto pero si se podrán eliminar todas las salidas registradas. En caso de querer guardar las salidas registradas antes de eliminarlas todas se da la posibilidad de descargarlas con toda su información en un archivo Excel.

Control De Salidas			
<input type="text" value="Dia"/>	<input type="text" value="Mes"/>	<input type="text" value="Año"/>	<input type="text" value="Nombre del Propietario"/>
<input type="button" value="Buscar"/>			
<input type="button" value="Mostrar Todas Las Salidas"/>	<input type="button" value="Crear Entrada"/>	<input type="button" value="Eliminar todas las Salidas"/>	<input type="button" value="Exportar Excel"/>
Fecha	Propietario	Vehículo	
28/05/2022 15:26:59	Wanda Alvarez	8915 DWB	<input type="button" value="Información"/>
28/05/2022 15:30:05	Suzanne Stanley	2340 LWM	<input type="button" value="Información"/>
01/06/2022 22:39:05	Suzanne Stanley	7943 KBH	<input type="button" value="Información"/>
01/06/2022 22:40:11	Eli Morgan	3480 HYM	<input type="button" value="Información"/>
01/06/2022 22:41:47	Suzanne Stanley	2340 LWM	<input type="button" value="Información"/>
01/06/2022 22:44:19	Wanda Alvarez	8915 DWB	<input type="button" value="Información"/>

Usuarios: en la vista de usuarios se mostraran los usuarios registrados con su nombre, correo y rol de cada usuario siendo este admin o user, también se permitirá mostrar la información correspondiente a un usuario concreto e incluso poder eliminar un usuario en concreto. Los usuarios se dividen en administradores que tienen acceso a todo y usuarios los cuales solo podrán acceder a las vistas de propietarios y vehículos. Por defecto vendrá con un usuario administrador ya creado.

Usuarios

[Crear Nuevo Usuario](#)

Nombre	Correo	Rol		
Adrian Lozano Lastras	adrianll@admin.com	admin	Información	Eliminar
Administrador	administrador@admin.com	admin	Información	Eliminar
Usuario	usuario@user.com	user	Información	Eliminar

2.4 IMPLANTACIÓN

Para realizar el despliegue de la aplicación se han empaquetado todos los archivos necesarios para el correcto funcionamiento del proyecto tanto del cliente como del servidor, generando con ellos un instalador, el cual al ejecutarlo como administrador procederá a realizar la instalación del proyecto generando una carpeta con el nombre CavAll en el directorio de C:\ en el cual tras realizar la instalación se encontrarán todos los archivos necesarios para el funcionamiento correcto del presente proyecto. Este archivo instalable será el que se le entregue al usuario final junto con la presente documentación.

Para el despliegue tanto del cliente como del servidor se recomienda que cada uno se encuentre en un equipo diferente aunque se pueden tener los dos funcionando en el mismo equipo.

Cliente

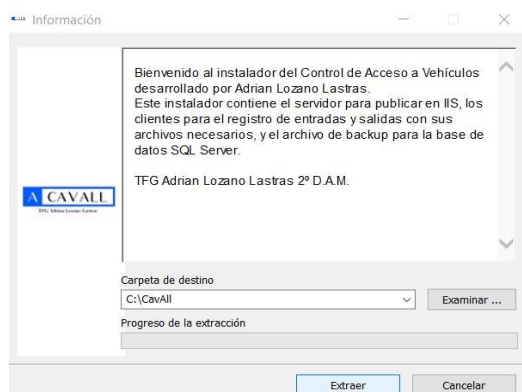
En cuanto al cliente se refiere, una vez instalado el programa CavAll.exe se crearán dos accesos directos en el escritorio uno para la realización de los registros de las entradas y otro para realizar el registro de las salidas. En la carpeta CavAll situada en C:\ se habrán generado el ejecutable de las entradas y el ejecutable de las salidas, junto con el archivo del clasificador necesario para el reconocimiento de las matrículas y las imágenes necesarias del icono y logo del proyecto.

Servidor

En lo que a la parte del servidor se refiere, tras realizar la instalación del programa CavAll.exe se crearán los archivos de backup de la base de datos para restaurarlo en SQL Server y una carpeta con el servidor publicado y listo para usarse en Internet (IIS) Information Service publicándolo en la carpeta correspondiente para su uso y seleccionando el puerto a usar el cual deberá tener permisos del firewall. También al realizar las entradas como las salidas se generarán en la carpeta de CavAll los logs de las entradas y salidas tanto permitidas como denegadas según sea el caso.

El cliente esta preparado para ejecutarse en el puerto 9696 y en localhost pudiéndose usar de esta forma tanto cliente como servidor desde el mismo equipo, si se desea que el cliente y el servidor se encuentren en equipos diferentes se deberá de conocer la ip del servidor ya que cada equipo tiene una ip diferente, y en caso de que se quiera emplear otro puerto o una dirección ip en concreto deberá de ponerse en contacto con el creador del proyecto para configurar un puerto e ip determinados y proporcionarle un ejecutable con el puerto e ip escogidos.

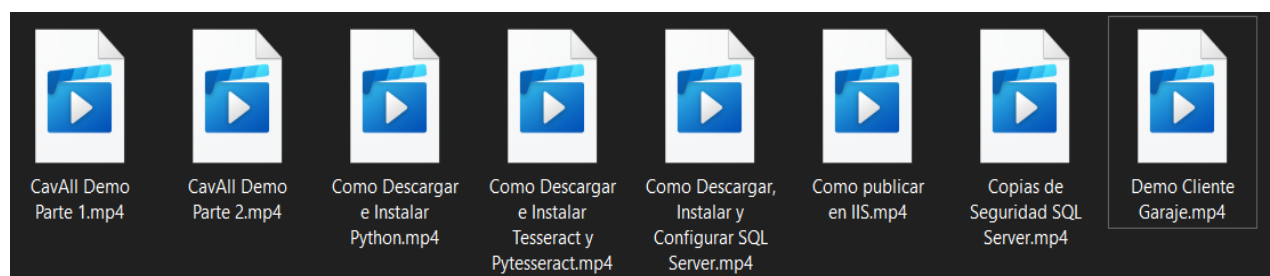
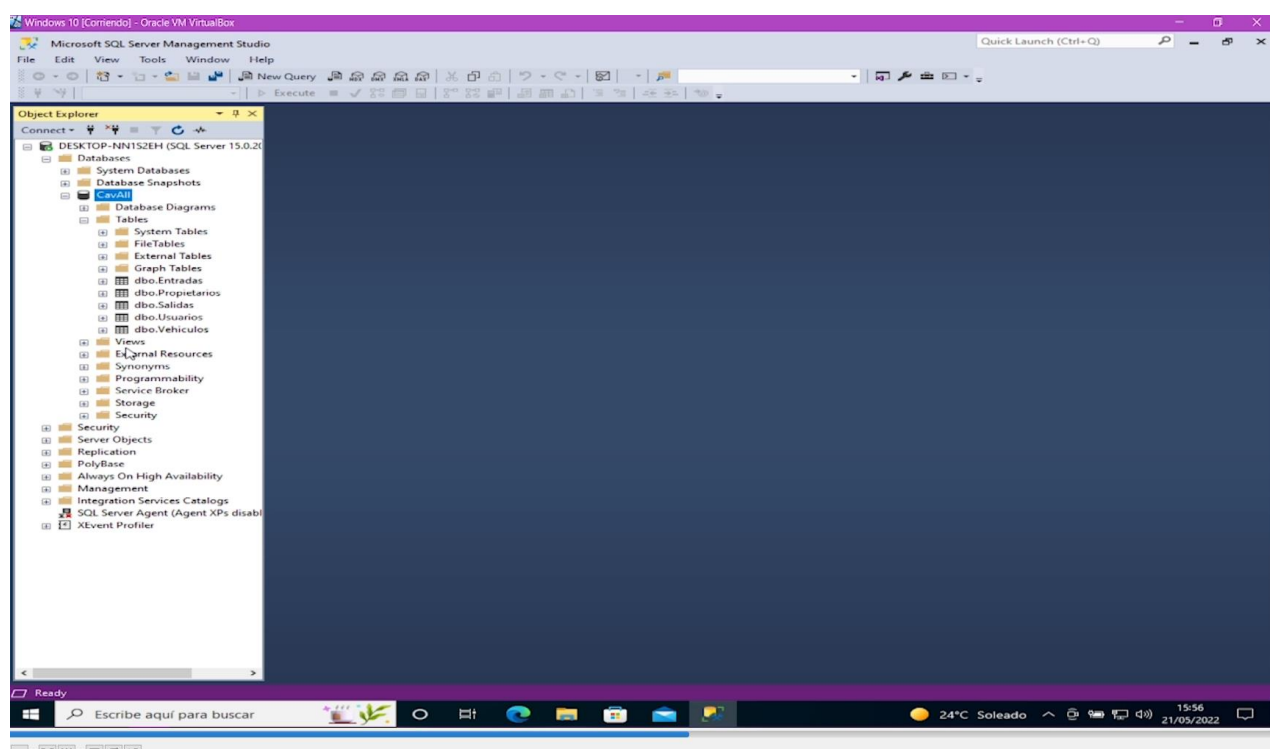
La dirección ip deberá de ser la misma del equipo donde se encuentre publicado el servidor, y el puerto que se le asigne deberá de tener tanto una regla de entrada como de salida en el firewall del equipo servidor. Las pruebas y desarrollo se han hecho usando el puerto 9696 y la ip 192.168.1.50 o localhost.



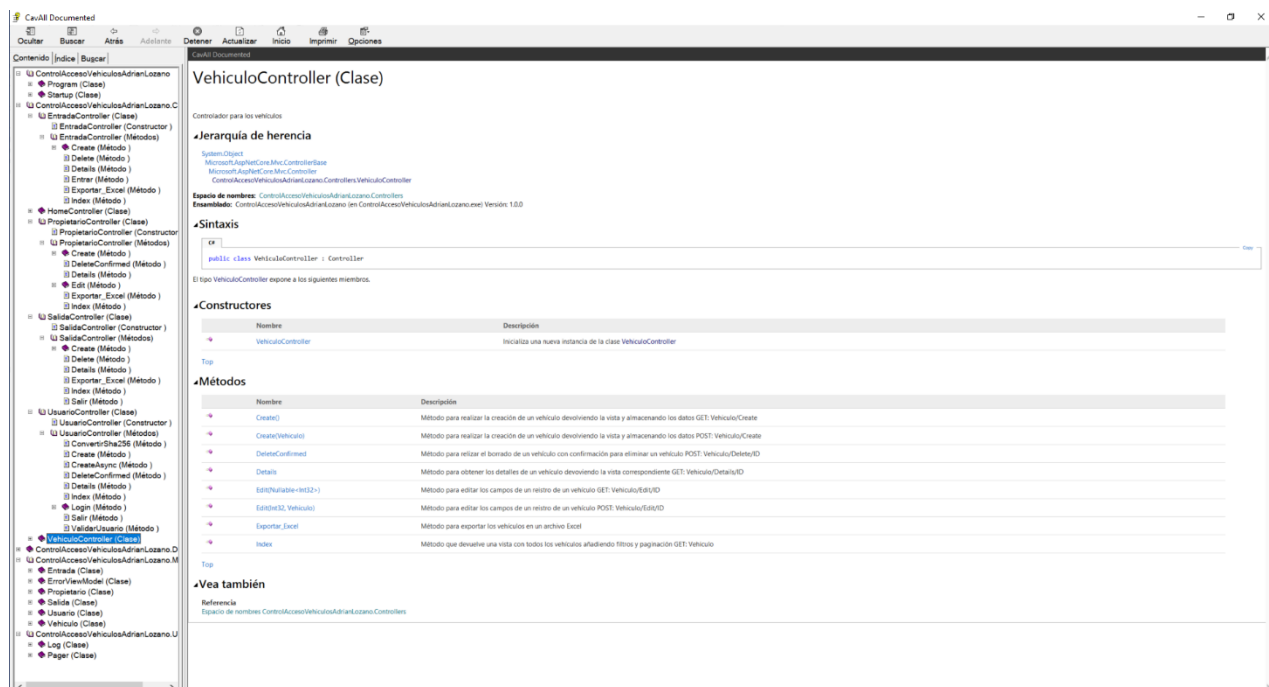
2.5 DOCUMENTACIÓN

En cuanto a la documentación se refiere se han realizado varios vídeos explicando como descargar, instalar y configurar todos los programas y paquetes necesarios para la implementación tanto del cliente como del servidor, como puede ser la descarga, instalación y configuración de Python, tesseract, SQL Server con backup y la publicación del servidor en Internet Information Service.

También se ha realizado diferentes vídeos explicando el funcionamiento tanto del cliente como del servidor y realizándose diferentes pruebas con vehículos reales y matrículas impresas.



También se ha generado un archivo con extensión .chm el cual contiene información documentada de las diferentes clases del servidor.



Para terminar en el script de Python del cliente se ha generado el pydoc correspondiente.



2.6 PROCESO DE DESARROLLO Y PLANIFICACIÓN

Lo primero que se realizó fue buscar información sobre que era y como funciona la detección de matrículas, el reconocimiento de los caracteres de estas y los controles de acceso para vehículos. Una vez sabido todo lo necesario se empezó con la elección de las herramientas y lenguajes más óptimos para la realización del proyecto empezando por las tecnologías que se iban a emplear en el cliente ya que sería el primero en desarrollarse y posteriormente centrarse en el servidor y las conexiones necesarias.

Tras haber seleccionado las tecnologías a emplear se comenzó con el desarrollo de la parte del cliente, para ello se realizaron diversos programas para realizar la detección de objetos y el reconocimiento de caracteres en textos para posteriormente realizarlo con las matrículas de los vehículos. Se tuvo que investigar sobre los diferentes tipos de filtrado y procesamiento de imágenes, a parte de las diferentes formas de reconocimiento de matrículas realizando así diferentes programas para probar cual daba mejores resultados y una tasa mayor de aciertos. Para ello con los distintos scripts desarrollados se realizaron diferentes pruebas, primero se probaron con imágenes de vehículos captadas en la calle, luego se realizaron con algunos vídeos de vehículos estacionados y en movimiento para comprobar que detectase y leyese correctamente las matrículas y para terminar una vez descartados los programas que daban más fallos se usaron dos, uno con binarización al cual se le realizaban algunos cálculos como el área de las matrículas y otro programa que usaba un clasificador en cascada y no era necesario realizar a penas cálculos. Con ellos se procedió a realizar las pruebas ya en un ambiente real como fue en la calle y en un garaje captando los diferentes vehículos tanto con una cámara web como con la cámara del portátil, obteniendo un correcto funcionamiento del sistema excepto por agentes externo como pueden ser la luz ambiente pero aun así se consiguieron buenos resultados, obteniendo una mayor tasa de aciertos el programa con el clasificador por lo cual es el que se ha decidido usar como cliente final.

Una vez realizando la detección de las matrículas y el reconocimiento de los caracteres se procedió con el desarrollo del servidor para ello se comenzó creando las entidades necesarias las cuales fueron cambiando y se fueron añadiendo nuevas según se iban necesitando a medida que se iba avanzando con el desarrollo, cambiando varias veces la base de datos. Al mismo tiempo también se iban realizando los diferentes controladores con sus vistas asociadas empezando por un diseño simple y a medida que avanzaba el desarrollo se fueron añadiendo nuevas funcionalidades, como es un inicio de sesión, diferentes filtrados, paginación y las diferentes funciones básicas en base de datos como son crear, editar, mostrar información y eliminar los campos necesarios a parte de los diferentes métodos con funcionalidades extras, así hasta tener el servidor bastante completo y con todas las funcionalidades necesarias. Una vez tenidos todos los modelos con sus controladores y vistas correspondientes con toda la funcionalidad necesaria se procedió a dar un buen diseño a las vistas, dejándolas con un diseño más intuitivo y visualmente más atractivo para los usuarios con una mejor colocación de los diferentes elementos, añadiéndole colores y estilos para hacerlo mas intuitivo y más agradable a la vista. También se realizaron las conexiones necesarias con la base de datos comprobando de esta forma su funcionamiento.

Cuando ya se tenía una versión bastante completa del servidor, mientras se terminaba se fueron realizando pruebas con el cliente y el servidor comprobando que funcionase todo correctamente, realizando el reconocimiento de las matrículas y observando que aquellas con acceso permitido se registraban correctamente en la base de datos mostrándose en el servidor.

Para finalizar con el proyecto se comenzó con la realización de la documentación explicando en que consiste, que se ha empleado, como se ha desarrollado y cada apartado necesario sobre el desarrollo del proyecto.

También se realizó tanto la implantación como el despliegue del cliente y del servidor, realizando dos archivos con extensión .exe del cliente (uno para registrar las entradas y otro para registrar las salidas) y realizando la publicación del servidor en una red interna para poder acceder desde cualquier dispositivo conectado a dicha red y así poder hacer uso del servidor sin tener que compilarlo.

Tras tener los ejecutables de cliente y la carpeta de publicación del servidor se procedió a crear un entorno de pruebas en una máquina virtual, en la que se instalaron todas las herramientas necesarias para el funcionamiento del cliente y el servidor, para posteriormente probar su funcionamiento, el cual fue bastante bueno probándose primero con matrículas fotocopias y posteriormente probándolo con vehículos reales en un garaje.

Para concluir los últimos días se dedicaron a la preparación del proyecto creando para ello un PowerPoint y diferentes vídeos explicativos sobre como instalar los diferentes programas necesarios y configurarlos.

A continuación se muestra el diagrama de Gantt empleado para seguir la planificación del proyecto.



3 RESULTADOS Y DISCUSIÓN

Con el presente proyecto se ha conseguido desarrollar un sistema de control de acceso completo para vehículos detectando las matrículas de estos para verificarlas y validarlas, empleando para ello tecnologías actuales como puede ser el framework de .NET, el uso de visión artificial junto con reconocimiento de caracteres y leguajes como son Python y C#.

Se han encontrado algunas limitaciones en cuanto a la parte del cliente al realizar la implantación del proyecto en un entorno real, ya que las pruebas se han realizado con unas cámaras con baja resolución y en el caso de emplear cámaras con una mayor resolución el rendimiento será más óptimo realizando el reconocimiento de las matrículas y de los caracteres de una forma más rápida y con menor tasa de errores, también hay que tener en cuenta el ambiente de pruebas y la iluminación los cuales han dificultado un poco a la hora de realizar el reconocimiento, pero al final se ha conseguido realizar correctamente la detección y el reconocimiento de las matrículas, obteniendo los caracteres de estas con una gran tasa de aciertos realizando así de esta forma el reconocimiento de las matrículas satisfactoriamente. Para ello se han realizado diversas pruebas como pueden ser imágenes, vídeos, matrículas impresas e incluso pruebas en la calle y en garaje para comprobar el correcto funcionamiento del cliente haciendo que detecte las matrículas y obtenga los caracteres de las mismas.

En cuanto al servidor no se han tenido tantos problemas ya que ha sido más el aprendizaje para saber incorporar todas las funcionalidades necesarias y realizar el desarrollo de la base de datos, de las conexiones, la configuración y creación de controladores y vistas, para el correcto funcionamiento, en donde también han habido diferentes problemas que se han ido solucionando en el transcurso del proyecto y algunas funcionalidades que se han tenido que dejar de lado ya que no se han podido realizar correctamente o por otros motivos como puede ser la publicación del proyecto en Docker.

4 TRABAJO FUTURO

Como trabajo futuro, se propone realizar el cliente implementándolo en un sistema hardware portátil para la realización de la identificación de las matrículas de los vehículos, como pueda ser una Raspberry Pi que es una placa desarrollada por la organización Raspberry Pi Foundation fundada en Inglaterra en el año 2012 con el objetivo de promover y facilitar el aprendizaje de técnicas de programación en la población. La Raspberry Pi es un hardware de bajo coste que suele tener bajos recursos y capacidades, ya que se suele utilizar para el desarrollo de funciones parciales, para ejecutar sistemas embebidos y adaptados para cada plataforma.

Para ello se han realizado pruebas con una Raspberry Pi Modelo B de 2Gb de RAM, probando el reconocimiento de matrículas en ella pero se han encontrado algunas limitaciones, como pueden ser algunos problemas con el script de Python como ha sido el rendimiento con expresiones regulares a la hora de realizar el análisis de las matrículas en tiempo real y se ha observado un rendimiento inferior al esperado, tardando más en realizar el reconocimiento y procesamiento de las matrículas y obteniendo una tasa mayor de errores que el cliente en Windows. Estos problemas podrían solventarse con una placa con un mayor rendimiento como pueda ser 4 o incluso 8Gb de RAM para que de esta forma funcione más fluido el reconocimiento de las matrículas.

Por ello se ha decidido no implementar la parte del cliente en un hardware portátil como puede ser Raspberry Pi y se ha dejado como trabajo futuro.

En caso de que se decida implementar el cliente en una Raspberry Pi se deberán de seguir los siguientes pasos:

1º Instalación de la Raspberry Pi

Lo primero será instalar el sistema operativo Raspberry OS (basado en debian) en una microSD con Raspberry Pi Imager.

2º Instalación de OpenCV

Se amplía la SWAP a 4096MB:

Se edita la configuración del swap:

\$ sudo nano /sbin/dphys-swapfile

\$ sudo nano /etc/dphys-swapfile

Se reinicia: \$ sudo reboot

Se utilizó el script de Q-Engineering para instalarlo. La versión instalada es la 4.5.0

\$ wget <https://github.com/Qengineering/Install-OpenCV-Raspberry-Pi-32bits/raw/main/OpenCV-4-5-5.sh>

\$ sudo chmod 755 ./OpenCV-4-5-5.sh

\$./OpenCV-4-5-5.sh

3º Instalación de Tesseract

sudo apt-get install tesseract-ocr

pip install pytesseract

Se añade Tesseract al PATH

PATH=\$PATH:/home/informatica/.local/bin

5 CONCLUSIONES

Para finalizar con la documentación del presente proyecto y como conclusión final, se puede afirmar que el alcance del proyecto ha sido el esperado, es decir, se han cumplido con los objetivos establecidos.

A modo personal, la realización del presente Trabajo de Fin de Grado ha supuesto todo un reto y se ha conseguido mejorar los conocimientos obtenidos de las diferentes asignaturas impartidas a lo largo del grado y se han aprendido nuevos conceptos, tecnologías, lenguajes y frameworks a parte se ha fomentado la utilización de herramientas open source, como pueden ser la librería de visión artificial OpenCv y la librería de reconocimiento de caracteres Tesseract, los lenguajes de programación empleados como son Python y C#, bases de datos y frameworks como .NET 5.

Gracias a todo esto se han conseguido los conocimientos necesarios para la realización del proyecto pudiendo así completar todos los objetivos que se habían planteado en el desarrollo principal del proyecto realizándose este correctamente y obtenido unos resultados satisfactorios consiguiendo que funcione correctamente tanto el cliente, el servidor como la conexión entre ambos.

Pero también han surgido diferentes problemas durante el desarrollo del proyecto que han echo ponerme a prueba, enfrentándome a un problema propio de la vida real, teniendo que estar realizando diferentes pruebas, comprobaciones y buscando mucha información para solventar los problemas que se han ido encontrando para así solucionarlos y seguir hacia delante.

6 BIBLIOGRAFÍA

6.1 CLIENTE

Code.visualstudio.com. 2022. *Visual Studio Code - Code Editing. Redefined*. [online] Available at: <<https://code.visualstudio.com/>> [Accessed 3 June 2022].

Docs.python.org. 2022. *re — Regular expression operations — Python 3.10.4 documentation*. [online] Available at: <<https://docs.python.org/3/library/re.html>> [Accessed 3 June 2022].

Docs.python.org. 2022. *time — Time access and conversions — Python 3.10.4 documentation*. [online] Available at: <<https://docs.python.org/3/library/time.html>> [Accessed 3 June 2022].

Es.wikipedia.org. 2022. *Reconocimiento automático de matrículas - Wikipedia, la enciclopedia libre*. [online] Available at: <https://es.wikipedia.org/wiki/Reconocimiento_autom%C3%A1tico_de_matr%C3%ADculas> [Accessed 3 June 2022].

Es.wikipedia.org. 2022. *Reconocimiento óptico de caracteres - Wikipedia, la enciclopedia libre*. [online] Available at: <https://es.wikipedia.org/wiki/Reconocimiento_%C3%B3ptico_de_caracteres> [Accessed 3 June 2022].

GitHub. 2022. *GitHub - opencv/opencv: Open Source Computer Vision Library*. [online] Available at: <<https://github.com/opencv/opencv>> [Accessed 3 June 2022].

GitHub. 2022. *tesseract-ocr*. [online] Available at: <<https://github.com/tesseract-ocr>> [Accessed 3 June 2022].

Ltd, R., 2022. *Buy a Raspberry Pi 4 Model B – Raspberry Pi*. [online] Raspberry Pi. Available at: <<https://www.raspberrypi.com/products/raspberry-pi-4-model-b/>> [Accessed 3 June 2022].

OpenCV. 2022. *Home - OpenCV*. [online] Available at: <<https://opencv.org/>> [Accessed 3 June 2022].

Pi, R., 2022. *Teach, Learn, and Make with Raspberry Pi*. [online] Raspberry Pi. Available at: <<https://www.raspberrypi.org/>> [Accessed 3 June 2022].

PyPI. 2022. *opencv-contrib-python*. [online] Available at: <<https://pypi.org/project/opencv-contrib-python/>> [Accessed 3 June 2022].

PyPI. 2022. *pyinotify*. [online] Available at: <<https://pypi.org/project/pyinotify/>> [Accessed 3 June 2022].

PyPI. 2022. *pytesseract*. [online] Available at: <<https://pypi.org/project/pytesseract/>> [Accessed 3 June 2022].

PyPI. 2022. *requests*. [online] Available at: <<https://pypi.org/project/requests/>> [Accessed 3 June 2022].

Python.org. 2022. *Download Python*. [online] Available at: <<https://www.python.org/downloads/>> [Accessed 3 June 2022].

6.2 SERVIDOR

Docs.microsoft.com. 2022. *Descarga de SQL Server Management Studio (SSMS) - SQL Server Management Studio (SSMS)*. [online] Available at: <<https://docs.microsoft.com/es-es/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16>> [Accessed 3 June 2022].

Docs.microsoft.com. 2022. *Documentación de .NET*. [online] Available at: <<https://docs.microsoft.com/es-es/dotnet/fundamentals/>> [Accessed 3 June 2022].

Docs.microsoft.com. 2022. *Documentos de C#: inicio, tutoriales y referencias.* [online] Available at: <<https://docs.microsoft.com/es-es/dotnet/csharp/>> [Accessed 3 June 2022].

Docs.microsoft.com. 2022. *Novedades de .NET 5*. [online] Available at: <<https://docs.microsoft.com/es-es/dotnet/core/whats-new/dotnet-5>> [Accessed 3 June 2022].

Docs.microsoft.com. 2022. *Referencia sobre la sintaxis de Razor para ASP.NET Core*. [online] Available at: <<https://docs.microsoft.com/es-es/aspnet/core/mvc/views/razor?view=aspnetcore-5.0>> [Accessed 3 June 2022].

En.wikipedia.org. 2022. *SQL Server Management Studio - Wikipedia*. [online] Available at: <https://en.wikipedia.org/wiki/SQL_Server_Management_Studio> [Accessed 3 June 2022].

Es.wikipedia.org. 2022. *C Sharp - Wikipedia, la enciclopedia libre*. [online] Available at: <https://es.wikipedia.org/wiki/C_Sharp> [Accessed 3 June 2022].

Es.wikipedia.org. 2022. *Internet Information Services - Wikipedia, la enciclopedia libre*. [online] Available at: <https://es.wikipedia.org/wiki/Internet_Information_Services> [Accessed 3 June 2022].

Es.wikipedia.org. 2022. *Microsoft SQL Server - Wikipedia, la enciclopedia libre*. [online] Available at: <https://es.wikipedia.org/wiki/Microsoft_SQL_Server> [Accessed 3 June 2022].

Iis.net. 2022. *Home : The Official Microsoft IIS Site*. [online] Available at: <<https://www.iis.net/>> [Accessed 3 June 2022].

Mark Otto, a., 2022. *Buttons*. [online] Getbootstrap.com. Available at: <<https://getbootstrap.com/docs/4.6/components/buttons/>> [Accessed 3 June 2022].

Mark Otto, a., 2022. *Forms*. [online] Getbootstrap.com. Available at: <<https://getbootstrap.com/docs/4.6/components/forms/>> [Accessed 3 June 2022].

Mark Otto, a., 2022. *Introduction*. [online] Getbootstrap.com. Available at: <<https://getbootstrap.com/docs/4.6/getting-started/introduction/>> [Accessed 3 June 2022].

Microsoft.com. 2022. *Download Internet Information Services (IIS) 10.0 Express from Official Microsoft Download Center*. [online] Available at: <<https://www.microsoft.com/es-es/download/details.aspx?id=48264>> [Accessed 3 June 2022].

Visual Studio. 2022. *Visual Studio: IDE y Editor de código para desarrolladores de software y Teams*. [online] Available at: <<https://visualstudio.microsoft.com/es/>> [Accessed 3 June 2022].

ANEXOS

I. PROCESAMIENTO Y TRATAMIENTO DE IMÁGENES

Pirámide de imágenes

La pirámide de imágenes es una técnica que se utiliza para detectar objetos cuando no se conoce con certeza a que distancia se encuentran estos, por lo que las dimensiones del objeto pudieran ser superiores a las dimensiones de la ventana de búsqueda.

Para ello, se produce una serie de copias de la misma imagen, ajustadas a menores resoluciones. Los objetos más cercanos a la cámara se detectan en las imágenes de menor resolución, mientras que los objetos más lejanos se detectan en las imágenes de mayor resolución. La resolución de cada nivel se define como la resolución del nivel inferior entre un factor de reducción, el cual es constante.

Escala de grises

La escala de grises es donde cada elemento de la matriz puede tomar un valor entero dentro del rango [0:255]. Los elementos de la matriz representan diferentes niveles de gris, donde el valor cero suele ser asignado al negro, mientras que el valor 255 se reserva para el blanco. Cada elemento de la matriz se almacena en un byte, que puede almacenar un valor entre 0 y 255.

La operación que nos permitiría pasar de una imagen RGB, en color, a una imagen en escala de grises consistiría en asignar, a cada píxel de cada matriz de cada color básico, el mismo valor.

Este valor pudiera ser la media aritmética de los tres colores de cada píxel o una media ponderada de los mismos. Con esto obtendríamos la misma imagen en escala de grises.

Canny

Este filtro se ha utilizado para reducir la cantidad de información a procesar facilitando el reconocimiento OCR, pudiendo realizar la detección de los bordes para así obtener el contorno que representan los puntos de frontera entre un objeto y el entorno en contacto con él. De esta forma recorriendo los contornos se obtendrá el área para comprobar que se trata de un rectángulo.

Dilatación:

Es el proceso de incorporación al objeto de los puntos del entorno en contacto con él. Produce un incremento de su área en píxeles según el tamaño y la forma de la máscara utilizada.

Binarización

Para realizar la binarización de las imágenes, estas deberán estar ya filtradas en escala de grises o en este caso por canny. La binarización permite establecer un umbral que actúe como frontera a partir de una imagen. Este umbral se puede seleccionar a partir del histograma de la imagen (distribución de los niveles de intensidad de una imagen), o seleccionarlo automáticamente mediante la binarización Otsu siendo esta la más recomendable para obtener un umbral óptimo, haciendo que la dispersión de niveles de grises sea la mínima.

La binarización es el proceso por el que se transforma una imagen de escala de grises a binaria mediante la transformación de los valores de intensidad lumínica de la matriz de la imagen, según la regla del valor límite de intensidad lumínica. Por lo tanto, a todos los píxeles que sean mayores se les asigna el valor 1, mientras que a los menores se les asigna el valor 0. Así, se consigue una imagen binaria Blanco/Negro.

Contornos

Representan los puntos de frontera entre un objeto y el entorno en contacto con él.

También se puede usar un algoritmo de clasificación en vez de usar los filtros y tener que calcular el área, contornos u otros parámetros, pudiendo realizar el reconocimiento de matriculas de una forma más rápida, eficaz y sencilla mediante un clasificador automático hardcascade de Machine Learning.

Reconocimiento de objetos mediante Clasificadores.

Para poder detectar cualquier tipo de objeto en una imagen, se requiere utilizar un algoritmo descriptor, con el cual se define una serie de características que describen el objeto por detectar, y un clasificador o máquina de aprendizaje supervisado, el cual debe ser entrenado con este conjunto de datos para el reconocimiento del objeto en una escena.

Se recomienda el uso del clasificador proporcionado por OpenCv para el reconocimiento de matrículas el cual esta accesible desde su GitHub.

Clasificador en Cascada

El clasificador en cascada es un tipo de máquina de aprendizaje supervisado, el cual recibe un conjunto de datos de entrenamiento, y utilizando algoritmos de aprendizaje definirá una función o una serie de condiciones que le permitirá predecir a que conjunto pertenece un determinado dato de entrada.

Se dice que el clasificador es en cascada, ya que está conformado por una serie de clasificadores más débiles, los cuales forman distintos niveles o etapas de clasificación. En cada etapa de clasificación, se realiza un proceso de evaluación del conjunto de muestras, y aquellas que cumplen con los criterios establecidos en la etapa, pasan al siguiente nivel de clasificación.

Una vez finalizado el análisis multietapa, la cantidad de muestras han sido reducidas drásticamente, por lo que se puede realizar un análisis más complejo sobre un limitado conjunto de muestras, después de desechar las muestras negativas obtenidas.

Una imagen puede contener cientos de miles de características, o variables, que describen el objeto de interés. Para poder realizar el entrenamiento del clasificador, se deben limitar el número de características a estudiar y definir las variables más importantes que describen el objeto. Para ello, se utiliza un algoritmo de optimización conocido como AdaBoost (Adaptive Boosting).

Para cada etapa de entrenamiento, se selecciona un conjunto de muestras positivas y negativas, y se comparan las distintas muestras para determinar cuáles son las variables que diferencian mejor los conjuntos, y en base a esto, se definen las características de la clasificadora correspondiente a la etapa. Sin embargo, la clasificadora de la etapa no es perfecta, es posible que algunas muestras hayan sido clasificadas erróneamente, por lo que, en el entrenamiento de la siguiente etapa, se hace más énfasis en estas muestras, y así sucesivamente hasta que se alcance el numero de etapas deseado, o se alcance un porcentaje de error mínimo.

II. CÁMARAS

Requisitos necesarios de las cámaras

Para poder conseguir una mayor precisión en el reconocimiento de las matrículas, habrá que tener en cuenta los siguientes puntos:

- La condición lumínica y climática, es decir, la iluminación la cual dependerá según si es de día o de noche, y las circunstancias climáticas como pueden ser, la lluvia, nubes o niebla.
- Los píxeles de la cámara, contra mayor sean los píxeles captados por la cámara mejor será el reconocimiento de las matrículas, evitando confundir los diferentes caracteres.
- La altura será fundamental para captar correctamente el vehículo y su matrícula correctamente, teniendo en cuenta también la distancia entre la cámara y el vehículo.
- El ángulo, contra mayor ángulo tenga la cámara, esta podrá captar un escenario mayor facilitando el captar la matrícula del vehículo deseado.

Es recomendable que la instalación de la cámara se realice a una altura entre un metro a metro y medio, teniendo entre ella y el vehículo a captar una distancia recomendada de metro y medio, y en cuanto al ángulo vertical y horizontal no exceda de treinta y cinco grados. Para tener un buen reconocimiento de los caracteres se recomienda una cámara de veinticinco megapíxeles, de ser menor de veinte podrá seguir funcionando correctamente pero posiblemente cueste más realizar el reconocimiento de las matrículas de los vehículos. También se recomienda que la cámara este bien enfocada y situada en una zona con buena visibilidad, evitando la luz de frente de los vehículos y el sol.

Contra mayor sea su resolución, su lente y su ángulo mejor será el reconocimiento de la matrícula. Para terminar, habría que tener en cuenta la tecnología de la cámara como pueden ser los filtros infrarrojos, ultravioletas u otros que permitan la detección del vehículo y el reconocimiento de la matrícula en diferentes condiciones y circunstancias.

Selección de las Cámaras

En el presente proyecto se utilizó una cámara webcam portátil de (0.3MP) y la cámara incorporada en el ordenador portátil donde se ha realizado el desarrollo del proyecto (1MP), que aun siendo unas cámaras muy básicas se consigue realizar correctamente el reconocimiento de las matrículas aunque sea un proceso mayor al que se obtendría mediante una cámara más avanzada.

Otros tipos de cámaras que se recomendarían son:

Cámaras CCTV: las cámaras de Circuito cerrado de televisión, son un sistema de videovigilancia con la finalidad de supervisar áreas y densidad de ambientes, controlar actividades, personal y controles de seguridad.

Cámaras con tecnología IP: este tipo de cámaras suelen tener una buena resolución de imagen, son configurables y asequibles con cualquier ordenador con acceso a internet. A parte son cámaras inteligentes y pueden trabajar por si solas de forma independiente sin un grabador. Los componentes principales que integran este tipo de cámaras de red incluyen una lente, un sensor de imagen, uno o más procesadores y memoria.

Pi NoIR

En caso de emplear una raspberry pi se recomienda el uso de la cámara Pi Noir ya que cuenta con el mismo sensor de 5 megapíxeles que su predecesora, pero sin el filtro infrarrojo, lo que le convierte en la herramienta perfecta para la captura de imágenes en la oscuridad. Es ideal para visión nocturna (como en el caso de este trabajo para permitir el acceso de los vehículos no solo durante el día). Su sensor es capaz de captar imágenes fijas con una resolución de 2592 x 1944, además de grabar vídeos en alta definición de hasta 1080p a 30 fotogramas por segundo. Tiene unas dimensiones de 20 x 25 x 9 mm y se conecta a la Raspberry Pi a través de los pines CSI libres, utilizando la interfaz de control I2C. La cámara Pi NoIR es sensible a la radiación de IR de longitud de onda corta (alrededor de 880 nm) e incorpora una iluminación de IR para ver en la oscuridad.

III. DESCARGAS

Python 3.10.4

<https://www.python.org/ftp/python/3.10.4/python-3.10.4-amd64.exe>

Tesseract OCR

<https://digi.bib.uni-mannheim.de/tesseract/tesseract-ocr-w64-setup-v5.1.0.20220510.exe>

PyTesseract

<https://pypi.org/project/pytesseract/>

SQL Server

<https://download.microsoft.com/download/d/a/2/da259851-b941-459d-989c-54a18a5d44dd/SQL2019-SSEI-Dev.exe>

SSMS

<https://download.microsoft.com/download/c/7/c/c7ca93fc-3770-4e4a-8a13-1868cb309166/SSMS-Setup-ENU.exe>

Paquetes aspnet y donet necesarios

<https://download.visualstudio.microsoft.com/download/pr/3789ec90-2717-424f-8b9c-3adbcbcea6c16/2085cc5ff077b8789ff938015392e406/aspnetcore-runtime-5.0.17-win-x64.exe>

<https://download.visualstudio.microsoft.com/download/pr/14ccbee3-e812-4068-af47-1631444310d1/3b8da657b99d28f1ae754294c9a8f426/dotnet-sdk-5.0.408-win-x64.exe>

<https://download.visualstudio.microsoft.com/download/pr/68b7e1d6-8d11-4d49-926a-23fadb7d1948/e754199aff44f4bb6740f2f75c550724/dotnet-runtime-5.0.16-win-x64.exe>

<https://download.visualstudio.microsoft.com/download/pr/158dce74-251c-4af3-b8cc-4608621341c8/9c1e178a11f55478e2112714a3897c1a/ndp472-devpack-enu.exe>

<https://download.visualstudio.microsoft.com/download/pr/deb4711b-7bbc-4afa-8884-9f2b964797f2/fb603c451b2a6e0a2cb5372d33ed68b9/dotnet-sdk-6.0.300-win-x64.exe>

IIS Express

https://download.microsoft.com/download/C/E/8/CE8D18F5-D4C0-45B5-B531-ADECD637A1AA/iisexpress_amd64_es-ES.msi

Instalación del Sistema Operativo de Raspberry Pi

<https://www.raspberrypi.com/software/>

Instalación de OpenCv en Raspberry Pi

<https://qengineering.eu/install-opencv-4.5-on-raspberry-pi-4.html>

Instalación de Tesseract en Raspberry Pi

<https://www.macheronte.com/en/how-to-install-tesseract-ocr-on-raspberry-pi/>