DESARROLLO DE UN SISTEMA PARA EL PROCESAMIENTO Y VISUALIZACIÓN DE DATOS DE EMISIONES DE VEHÍCULOS DE COMBUSTIÓN

DEVELOPMENT OF A SYSTEM FOR THE PROCESSING AND VISUALIZATION OF COMBUSTION VEHICLE EMISSIONS DATA

Miguel Marzal García

GRADO EN INGENIERÍA INFORMÁTICA

FACULTAD DE INFORMÁTICA

UNIVERSIDAD COMPLUTENSE DE MADRID

TRABAJO DE FIN DE GRADO

CURSO 2021-2022

DIRECTOR

Antonio Sarasa Cabezuelo

AGRADECIMIENTOS

A mi familia y mi novia por aguantarme y apoyarme durante todo este curso, y hacer posible que acabe la carrera después de tanto tiempo.

RESUMEN

En este trabajo se recoge la especificación, diseño e implementación de un sistema de procesamiento y visualización de datos de emisiones de vehículos de combustión. Se recogen todos los componentes tanto del servicio encargado del procesamiento de datos como de la aplicación web para la visualización de los mismos. Se incluye todo lo necesario para la compresión de este trabajo.

El sistema se compone de dos aplicaciones independientes, pero que necesitan una de la otra para el perfecto funcionamiento. Consiste en una aplicación web que ofrece al usuario un panel de administración en el que se podrán seleccionar y visualizar los datos de una sesión de medición de gases contaminantes, y de un servicio de Windows, que se encarga del procesado de los datos de dicha sesión.

Lo datos son generados por un sensor denominado “L6” que pertenece a la empresa OPUS RSE, el cual genera los archivos binarios con datos en “crudo”, que deben procesarse para extraer la información de utilidad y mostrarla al usuario final con la mayor claridad posible.

PALABRAS CLAVE

Emisiones, web, servicio Windows, sql server, vehículo

ABSTRACT

This work includes the specification, design and implementation of a system for processing and visualization of emissions data from combustion vehicles. All the components of both the service in charge of data processing and the web application for data visualization are included. Everything necessary for the compression of this work is included.

The system is composed of two independent applications, but they need each other for perfect operation. It consists of a web application that offers the user an administration panel where the data of a measurement session of pollutant gases can be selected and displayed, and a Windows service, which is responsible for processing the data from that session.

The data are generated by a sensor called "L6" belonging to the company OPUS RSE, which generates binary files with "raw" data, which must be processed to extract the useful information and display it to the end user as clearly as possible.

KEYWORDS

Emissions, web, windows service, sql server, vehicle

INDICE

INDICE FIGURAS

INDICE TABLAS

Capítulo 1. Introducción.

En este capítulo de la memoria se va explicar el motivo justifica a la realización de este trabajo y el objetivo que pretende alcanzar. Además, se describe la estructura que se ha seguido durante la escritura de esta memoria.

1.1 Motivación.

OPUS RS Europe es una empresa privada de España, responsable del negocio y uso de la tecnología OPUS Remote Sensing en toda Europa. OPUS RS Europe es el único laboratorio de detección remota ISO-17025 en Europa, acreditado para la medición remota de emisiones de escape de contaminantes atmosféricos de vehículos de motor.

La emisión de gases contaminantes del tráfico rodado ha ido decreciendo en los últimos años debido al endurecimiento de las normativas internacionales en el sector de la automoción (Favre, 2003)[1]. (Kozina, 2022)[2], además, el incumplimiento deliberado de la normativa por parte de algunos fabricantes, ha hecho que el interés en la medida de las emisiones de los vehículos en condiciones reales de funcionamiento se haya visto incrementado en los últimos años (Grange, 2020)[3].

La empresa Opus RSE dispone de un sistema certificado para la monitorización remota de las emisiones del tráfico rodado. Este sistema permite la realización de campañas de medición a demanda de los clientes. La tecnología en la que se basa el sistema, se desarrolló al final del siglo pasado (Stedman, 1991)[4] y permite la medida de los distintos contaminantes con una precisión insuficiente para la correcta evaluación de los vehículos de nuevas generaciones, y es esta la motivación principal para el desarrollo de una nueva tecnología que permita la medida de vehículos modernos.

El nuevo sistema desarrollado por la empresa basa su funcionamiento en la técnica de espectroscopía de absorción con láseres de diodo sintonizables (TDLAS por sus siglas en inglés), y permite incrementar la precisión en la medida de bajas concentraciones de gases contaminantes en un factor x100 con respecto al sistema que está actualmente en uso. Para poder medir las emisiones de cada vehículo, el sistema adquiere 1.5 millones de muestras en 0.5 s, y es en el procesado de esas muestras en el que se centra la primera parte de este trabajo de fin de grado. Además, la complejidad de los datos obtenidos durante la fase de procesado, hace necesario el desarrollo de una interfaz que permita, por un lado, almacenar los datos obtenidos tanto en un servidor local como en un servidor online, y por otro lado visualizar de manera inteligible los datos obtenidos para permitir una evaluación adecuada por parte del usuario final. La interfaz desarrollada se muestra en la segunda parte de la memoria, y funciona, además como interfaz con el usuario final.

1.2 Objetivos.

El objetivo principal de este trabajo consiste en la implementación de un sistema que permita procesar los datos obtenidos por el sensor desarrollado tras el paso de un vehículo en menos de un segundo y su posterior visualización en tiempo real en una aplicación web.

El objetivo principal se puede desglosar en los siguientes dos objetivos específicos:

- Desarrollo de un servicio de Windows que, a partir de una serie de ficheros generados por el sensor, sea capaz de traducirlo a ratios de emisiones de los diferentes gases que es capaz de medir.
- Desarrollo de una aplicación web que permita la administración de inicio y fin de sesiones de medición del sensor, permitiendo crear clientes y proyectos para los que se realizan las mediciones, así como, las localizaciones en las cuales se van a realizar dichas mediciones.

1.3 Planificación.

1.3.1 Diagrama de Gantt.

En la Figura 1 se muestra la planificación del proyecto.

Figura 1 - Diagrama de Gantt

1.3.2 Toma y análisis de requisitos de la aplicación web.

En esta fase se realizó la toma de requisitos. Esto se realizó en conjunto con los compañeros de campo, que son los que van a utilizar el sistema en el día a día realizando campañas de medición.

Los compañeros explicaron sus necesidades de cara a la web, y realizaron una serie de peticiones de las que les gustaría disponer en la aplicación, en comparación con el software de medición que ellos están acostumbrados a utilizar.

Una vez tomados los requisitos, se analizaron y se pasó a la fase de diseño e implementación de la web.

1.3.3 Diseño e implementación de la aplicación web.

En esta fase, una vez analizados los requisitos de la aplicación, se empezó a realizar el diseño de la misma, así como, la forma en la que se iba a estructurar y las tecnologías que se iban a utilizar para la implementación.

Una vez realizado el diseño y definidas las tecnologías necesarias se inició el desarrollo de la aplicación.

1.3.4 Pruebas de la aplicación web sin servicio de Windows.

Esta fase se realizó una vez terminada la implementación de la web. Aunque aún no se había implementado el servicio Windows, se pudieron probar el resto de módulos de la aplicación.

1.3.5 Toma y análisis de requisitos del Servicio Windows.

En esta fase se realizó la toma de requisitos del sistema encargado de procesar los ficheros en "crudo" que genera el sensor. Esto se realizó en conjunto con el compañero especialista en óptica, encargado del diseño del sensor, y quien realizado pruebas de laboratorio con dichos ficheros.

El compañero explicó la forma en la que daríamos sentido a estos datos, describiendo las acciones que se deberían realizar para darle sentido a los datos generados por el sensor.

Una vez tomados los requisitos, se analizaron y se pasó a la fase de desarrollo e implementación del servicio.

1.3.6 Diseño e implementación del Servicio Windows.

Una vez se analizaron los requisitos, lo que mejor se adaptaba a nuestra necesidad era un servicio de Windows, para el cual se definieron una estructura y unas tecnologías a utilizar.

Una vez terminado el diseño, se pasó a realizar la implementación de dicho servicio.

1.3.7 Pruebas de la aplicación web con el Servicio Windows.

Una vez terminada la implementación del servicio Windows, se pudieron realizar pruebas conjuntas de la aplicación web y el servicio Windows, de manera que ya se podía comprobar en tiempo real, como se procesaba cada fichero.

1.4 Estructura de la memoria.

A continuación, se describe brevemente la estructura de la memoria.

- Capítulo 1: En este capítulo se describe la motivación del trabajo, los objetivos y la estructura de la memoria.

- Capítulo 2: En este capítulo se estudia una herramienta similar a la que se ha realizado en el trabajo.

- Capítulo 3: En este capítulo se describe la tecnología utilizada.

- Capítulo 4: En este capítulo se definen los actores y se describen casos de uso mediante tablas junto a sus requisitos.

- Capítulo 5: En este capítulo se explicará la implementación de la base de datos.

- Capítulo 6: En este capítulo se explicará la arquitectura de la aplicación.

- Capítulo 7: En este capítulo se realizará un estudio sobre el diseño e implementación de las funcionalidades de la web y el servicio windows.

- Capítulo 8: En este capítulo se expondrá las estadísticas aportadas por usuarios que han trabajado con el sistema.

- Capítulo 9: En este capítulo se describen algunas funcionalidades que se añadirán en el futuro.

- Anexo I: Preguntas de la evaluación de la web.

Chapter 1. Introduction.

This chapter of the report explains the reason for this work and the objective it aims to achieve. In addition, the structure followed during the writing of this report is described.

1.1 Motivation.

OPUS RS Europe is a privately owned company in Spain, responsible for the business and use of OPUS Remote Sensing technology throughout Europe. OPUS RS Europe is the only ISO-17025 remote sensing laboratory in Europe, accredited for the remote measurement of exhaust emissions of air pollutants from motor vehicles.

The emission of pollutant gases from road traffic has been decreasing in recent years due to the tightening of international regulations in the automotive sector (Favre, 2003)[1] (Kozina, 2022)[2], in addition, the deliberate failure of some manufacturers to comply with the regulations has led to an increased interest in the measurement of vehicle emissions in real operating conditions in recent years (Grange, 2020)[3].

Opus RSE has a certified system for remote monitoring of road traffic emissions. This system allows measurement campaigns to be carried out on demand by customers. The technology on which the system is based was developed at the end of the last century (Stedman, 1991)[4] and allows the measurement of different pollutants with an insufficient precision for the correct evaluation of new generation vehicles, and this is the main motivation for the development of a new technology that allows the measurement of modern vehicles.

The new system developed by the company is based on the technique of absorption spectroscopy with tunable diode lasers (TDLAS), and allows to increase the accuracy in the measurement of low concentrations of pollutant gases by a factor x100 with respect to the system currently in use. In order to measure the emissions of each vehicle, the system acquires 1.5 million samples in 0.5 s, and it is the processing of these samples that is the focus of the first part of this thesis. In addition, the complexity of the data obtained during the processing phase, makes necessary the development of an interface that allows, on the one hand, to store the data obtained both in a local server and in an online server, and on the other hand, to visualize in an intelligible way the data obtained to allow a proper evaluation by the end user. The developed interface is shown in the second part of the report, and also functions as an interface with the end user.

1.2 Objectives.

The main objective of this work consists of the implementation of a system that allows processing the data obtained by the sensor developed after the passage of a vehicle in less than a second and its subsequent display in real time in a web application.

The main objective can be broken down into the following two specific objectives:

- Development of a Windows service that, from a series of files generated by the sensor, is able to translate it into emission ratios of the different gases it is able to measure.
- Development of a web application that allows the administration of the start and end of measurement sessions of the sensor, allowing the creation of clients and projects for which the measurements are performed, as well as the locations in which these measurements are going to be performed.

1.3 Planification.

1.3.1 Gantt diagram.

Figura 2 shows the project schedule

Figura 2 – Gantt diagram

1.3.2 Requirements gathering and analysis of the web application..

In this phase the requirements gathering was carried out. This was done in conjunction with field colleagues, who are the ones who will use the system on a day-to-day basis to carry out measurement campaigns.

The colleagues explained their needs for the web, and made a series of requests that they would like to have in the application, compared to the measurement software they are used to using.

Once the requirements were taken, they were analyzed and moved on to the design and implementation phase of the website.

1.3.3 Design and implementation of the web application.

In this phase, once the requirements of the application were analyzed, the design of the application was started, as well as the way it was going to be structured and the technologies that were going to be used for the implementation.

Once the design was completed and the necessary technologies were defined, the development of the application began.

1.3.4 Testing of the web application without Windows service.

This phase was carried out once the web implementation had been completed. Although the Windows service had not yet been implemented, the rest of the application modules could be tested.

1.3.5 Windows Service requirements gathering and analysis.

In this phase the requirements of the system in charge of processing the "raw" files generated by the sensor were taken. This was done in conjunction with the optics specialist colleague, who was in charge of the sensor design, and who performed laboratory tests with these files.

The colleague explained how we would make sense of this data, describing the actions that should be performed to make sense of the data generated by the sensor.

Once the requirements were taken, they were analyzed and we moved on to the development and implementation phase of the service.

1.3.6 Design and implementation of the Windows Service.

Once the requirements were analyzed, what best suited our needs was a Windows service, for which a structure and technologies to be used were defined.

Once the design was finished, the implementation of the service was started.

1.3.7 Testing of the web application with the Windows Service.

Once the implementation of the Windows service was finished, joint tests of the web application and the Windows service could be carried out, so that it was possible to check in real time how each file was processed.

1.4 Memory structure.

The structure of the report is briefly described below.

- Chapter 1: This chapter describes the motivation for the work, the objectives and the structure of the report.
- Chapter 2: In this chapter a tool similar to the one used in the paper is studied.
- Chapter 3: This chapter describes the technology used.
- Chapter 4: In this chapter the actors are defined and use cases are described by means of tables together with their requirements.
- Chapter 5: This chapter explains the implementation of the database.
- Chapter 6: This chapter explains the application architecture.
- Chapter 7: This chapter will study the design and implementation of the web functionalities and the windows service.
- Chapter 8: This chapter will present the statistics provided by users who have worked with the system.
- Chapter 9: This chapter describes some functionalities that will be added in the future.
- Annex I: Web evaluation questions..

Capítulo 2. Estado del arte.

En este capítulo se va a describir un sistema similar al que se va a desarrollar en este proyecto.

El sistema NextGen [5], al igual que el que se va a desarrollar en este proyecto, está desarrollado para el procesamiento de datos de emisiones de vehículos de combustión y su visualización en tiempo real, pero para sensores de tecnología basada en infrarrojos y ultravioleta. En la Figura 3 se puede ver la pantalla principal de dicho sistema.

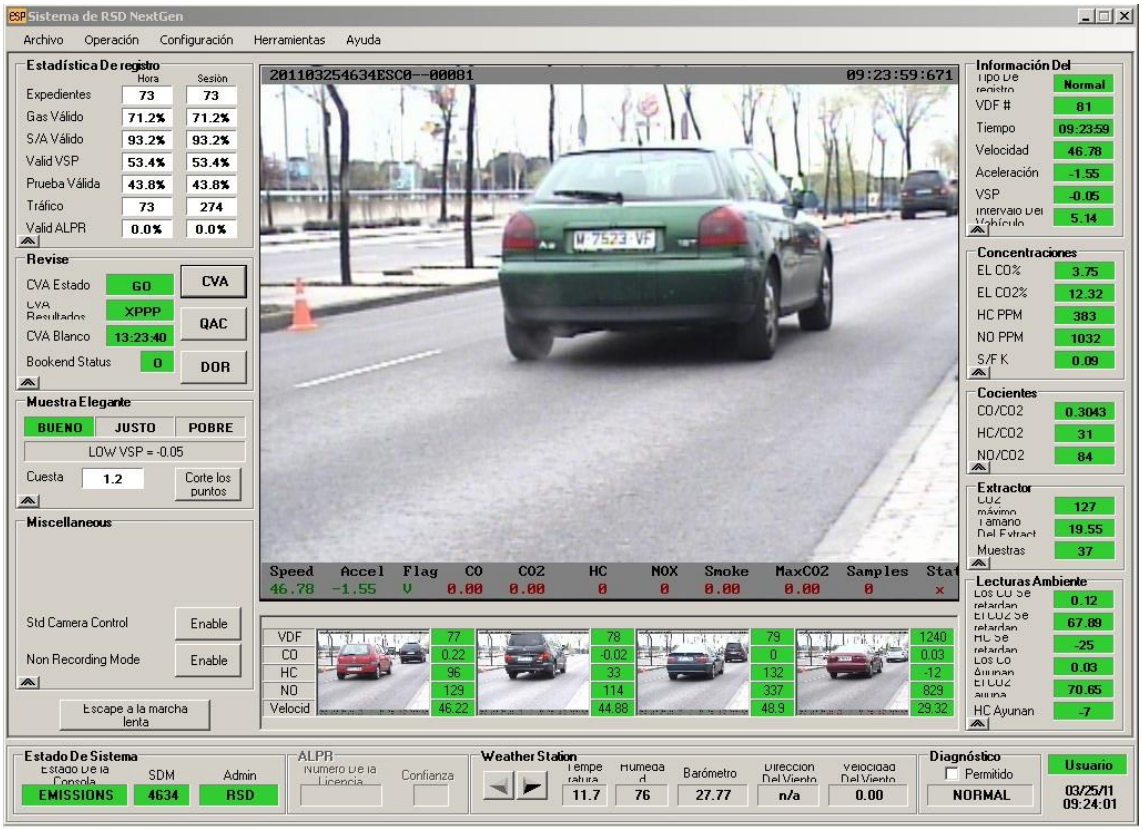


Figura 3 - Aplicación NextGen propiedad de Opus RS Europe

Capítulo 3. Tecnología empleada

En este capítulo se detallarán las tecnologías usadas para el desarrollo de este proyecto.

3.1 Microsoft Visual Studio

Microsoft Visual Studio [6] es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para Windows y macOS. Es compatible con múltiples lenguajes de programación, tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP, al igual que entornos de desarrollo web, como ASP.NET MVC, Django, etc., a lo cual hay que sumarle las nuevas capacidades en línea bajo Windows Azure en forma del editor Mónaco.

Visual Studio permite a los desarrolladores crear sitios y aplicaciones web, así como servicios web en cualquier entorno compatible con la plataforma .NET (a partir de la versión .NET 2002). Así, se pueden crear aplicaciones que se comuniquen entre estaciones de trabajo, páginas web, dispositivos móviles, dispositivos embebidos y videoconsolas, entre otros.

3.2 ASP.NET MVC Framework

ASP.NET MVC Framework [7] es un framework de aplicaciones web que implementa el patrón modelo-vista-controlador (MVC).

Basado en ASP.NET, permite a los desarrolladores de software construir una aplicación web como una composición de tres funciones: modelo, vista y controlador.

En marzo de 2009 se hizo pública la primera versión de ASP.NET MVC. El patrón de arquitectura MVC (model-view-controller) no es nuevo (data de 1979) ni es algo que haya inventado Microsoft. Existen muchos frameworks de desarrollo web populares que utilizan MVC, como por ejemplo Ruby on Rails, Spring o Apache Struts. MVC es un patrón de arquitectura que ayuda a crear una separación lógica entre el modelo (información y lógica de negocio), la vista (la lógica de presentación) y el controlador (intermediario entre la vista y el modelo).

Uno de los pilares básicos de ASP.NET MVC es el concepto de enrutamiento (routing), lo que permite a las aplicaciones aceptar peticiones a URL que no se corresponden con ficheros físicos en el servidor.

3.3 Microsoft SQL Server

Microsoft SQL Server [8] es un sistema de gestión de base de datos relacional, desarrollado por la empresa Microsoft.

El lenguaje de desarrollo utilizado (por línea de comandos o mediante la interfaz gráfica de Management Studio [9]) es Transact-SQL [10] (TSQL), una implementación del estándar ANSI del lenguaje SQL, utilizado para manipular y recuperar datos (DML), crear tablas y definir relaciones entre ellas (DDL).

3.4 Azure Devops Server

Herramientas de desarrollo de software de colaboración para todo el equipo

Anteriormente conocido como Team Foundation Server (TFS), Azure DevOps Server [11] es un conjunto de herramientas de desarrollo de software de colaboración, hospedadas en el entorno local. Azure DevOps Server se integra con el IDE o editor existente, lo que permite a su equipo de funciones cruzadas trabajar de forma eficaz en proyectos de todos los tamaños.

3.5 Internet Information Services

Internet Information Services o IIS [12] es un servidor web y un conjunto de servicios para el sistema operativo Microsoft Windows. Originalmente era parte del Option Pack para Windows NT. Luego fue integrado en otros sistemas operativos de Microsoft destinados a ofrecer servicios, como Windows 2000 o Windows Server 2003, 2016 y 2019. Windows XP Profesional incluye una versión limitada de IIS. Los servicios que ofrece son: FTP, SMTP, NNTP y HTTP/HTTPS.

Este servicio convierte a un PC en un servidor web para Internet o una intranet, es decir que en los ordenadores que tienen este servicio instalado se pueden publicar páginas web tanto local como remotamente.

Se basa en varios módulos que le dan capacidad para procesar distintos tipos de páginas. Por ejemplo, Microsoft incluye los de Active Server Pages (ASP) y ASP.NET. También pueden ser incluidos los de otros fabricantes, como PHP o Perl.

3.6 GitMind

GitMind [13] es una aplicación gratuita de mapas mentales escrita en JavaScript.1 GitMind fue lanzado en 2019 por WANGXU TECHNOLOGY (HK) CO., LIMITED.

Es compatible con mapas mentales, diagramas de flujo, Diagrama de Ishikawa, diagramas de árbol, Organigrama, Lenguaje unificado de modelado, Modelo entidad-relación, etc. Normalmente, se utiliza para la gestión de conocimientos, actas de reuniones, gestión de proyectos y otras tareas creativas. Además, GitMind puede leer e importar archivos XMind.

3.7 Otras tecnologías

- **C#**

C#[14] es un lenguaje de programación multiparadigma desarrollado y estandarizado por la empresa Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA (ECMA-334) e ISO (ISO/IEC 23270). C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común.

- **SignalR**

ASP.NET SignalR[15] es una biblioteca para desarrolladores de ASP.NET que simplifica el proceso de agregar funcionalidad web en tiempo real a las aplicaciones. La funcionalidad web en tiempo real es la capacidad de hacer que el código de servidor inste contenido a los clientes conectados al instante a medida que esté disponible, en lugar de hacer que el servidor espere a que un cliente solicite nuevos datos.

- **JavaScript**

JavaScript[16] (abreviado comúnmente JS) es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos,2 basado en prototipos, imperativo, débilmente tipado y dinámico.

Se utiliza principalmente del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas3 y JavaScript del lado del servidor (Server-side JavaScript o SSJS).

- **JQuery**

jQuery[17] es una biblioteca multiplataforma de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.1 Fue presentada el 14 de enero de 2006 en el BarCamp NYC. De acuerdo a un análisis de la Web (realizado en 2017) JQuery es la biblioteca de JavaScript más utilizada, por un amplio margen

- **AJAX**

AJAX[18], acrónimo de Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones web asíncronas. Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible interactuar con el servidor sin necesidad de recargar la página web, mejorando la interactividad, velocidad y usabilidad en las aplicaciones.

- **Bootstrap**

Bootstrap[19] es una biblioteca multiplataforma o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como extensiones de JavaScript adicionales. A diferencia de muchos frameworks web, solo se ocupa del desarrollo front-end.

- **HTML**

HTML[20], siglas en inglés de HyperText Markup Language (‘lenguaje de marcado de hipertexto’), hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia del software que conecta con la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición

de contenido de una página web, como texto, imágenes, videos, juegos, entre otros.

- **CSS**

CSS[21] (siglas en inglés de Cascading Style Sheets), en español «Hojas de estilo en cascada», es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado.

- **LINQ**

Language Integrated Query (LINQ[22], pronunciado "link") o Consulta Integrada en el Lenguaje es un componente de la plataforma Microsoft .NET que agrega capacidades de consulta a datos de manera nativa a los lenguajes .NET, si bien existen implementaciones para Java, PHP, JavaScript y ActionScript

LINQ extiende el lenguaje a través de las llamadas expresiones de consulta, que son parecidas a las sentencias SQL y pueden ser usadas para extraer y procesar convenientemente datos de arrays, clases enumerables, documentos XML, bases de datos relacionales y fuentes de terceros. Otros usos, que utilizan expresiones de consulta como plataforma general para la composición de expresiones más legibles, incluyen la construcción de manejadores de eventos.

- **Entity Framework**

Es un conjunto de API de acceso a datos para el Microsoft .NET Framework.

Una entidad del Entity Framework[23] es un objeto que tiene una clave representando la clave primaria de una entidad lógica de datastore. Un modelo conceptual Entity Data Model (modelo Entidad-Relación) es mapeado a un modelo de esquema de datastore. Usando el Entity Data Model, el Framework permite que los datos sean tratados como entidades independientemente de sus representaciones del datastore subyacente.

El Entity SQL es un lenguaje similar al SQL para consultar el Entity Data Model (en vez del datastore subyacente). Similarmente, las extensiones del Linq, Linq-to-Entities, proporcionan consultas tipeadas en el Entity Data Model. Las consultas Entity SQL y Linq-to-Entities son convertidas internamente en un Canonical Query Tree que entonces es convertido en una consulta comprensible al datastore subyacente (ej. en SQL en el caso de una base de datos relacional). Las entidades pueden utilizar sus relaciones, y sus cambios enviados de regreso al datastore.

Capítulo 4. Casos de uso.

En este capítulo se describirán tanto los casos de uso como los actores que se han definido para el desarrollo del sistema. En la primera parte se presentarán los actores y en la segunda, los casos de uso de cada uno de ellos.

4.1 Actores del sistema.

En este apartado se describen los diferentes actores que se han definido para el uso del sistema.

- **Operador.**

  Representa un usuario registrado, que previamente ha sido dado de alta por el administrador, y el cual tiene disponibles las acciones de inicia y parar una sesión, así como ver la historia de la misma.

- **Administrador.**

  Representa el administrador del sistema y tiene control total en la aplicación.

4.2 Módulo operador.

En esta sección, se detallan los casos de uso del operador. En la Figura 4 se puede ver el diagrama de casos de uso de dicho usuario.

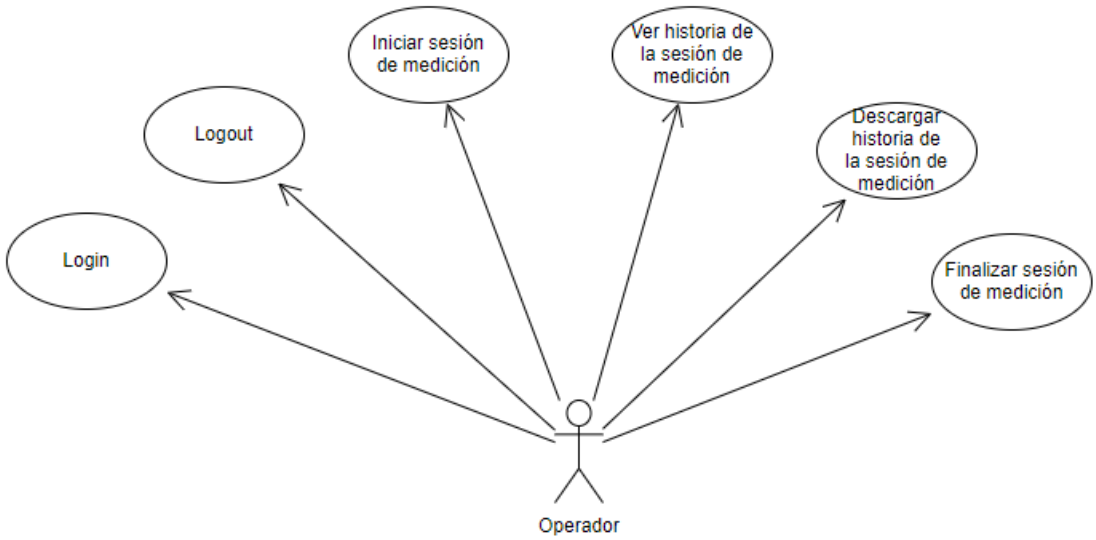


Figura 4 - Diagrama casos de uso operador.

CU-O-1	Login	
Versión	1.0.0	
Prioridad	Alta	
Estabilidad	Alta	
Precondición	El usuario ha sido dado de alta por el administrador	
Descripción	Lo usuarios que ya han sido dados de alta, pueden iniciar sesión para acceder a la pantalla de administración de sesiones.	
Entrada	Usuario y contraseña.	
Salida	NA	
Secuencia normal	Paso	Acción
	1	El usuario accede a la aplicación y se le muestra la pantalla de inicio de sesión.
	2	El usuario introduce sus credenciales y pulsa el botón “Sign In”.
	3	La aplicación comprueba las credenciales y permite al usuario entrar a la pantalla principal de la aplicación.
Postcondición	NA	
Excepciones	Paso	Acción
	3	La aplicación comprueba que las credenciales son incorrectas y devuelve a la pantalla de inicio de sesión.

Tabla 1 – Login.

CU-O-2	Logout	
Versión	1.0.0	
Prioridad	Alta	
Estabilidad	Alta	
Precondición	El usuario ha sido dado de alta por el administrador y haber iniciado sesión.	
Descripción	El usuario podrá cerrar su sesión.	
Entrada	NA	
Salida	NA	
Secuencia normal	Paso	Acción
	1	El usuario accede a la aplicación y se le muestra la pantalla principal.
	2	El usuario pulsa en “Sign Out” situado en la parte superior.
	3	El sistema confirma el cierre de sesión y vuelve a la pantalla de inicio de sesión.
Postcondición	NA	
Excepciones	Paso	Acción

Tabla 2 – Logout.

CU-O-3	Iniciar sesión de medición	
Versión	1.0.0	
Prioridad	Alta	
Estabilidad	Alta	
Precondición	El usuario debe haber iniciado sesión en la aplicación y no debe haber ninguna sesión iniciada.	
Descripción	El usuario podrá iniciar una sesión de medición.	
Entrada	El usuario debe seleccionar un “customer”, un “project” y un “site”.	
Salida	NA	
Secuencia normal	Paso	Acción
	1	El usuario accede a la aplicación y selecciona un cliente, un proyecto y una localización.
	2	El usuario pulsa en “Start”.
	3	El sistema crea una nueva sesión de medición, mostrando los datos asociados a esta.
Postcondición	NA	
Excepciones	Paso	Acción
	3	Se produce un error al crear la sesión y vuelve a la página principal para volver a seleccionar los datos necesarios e iniciar de nuevo.

Tabla 3 – Iniciar sesión de medición.

CU-O-4	Ver historia de una sesión de medición	
Versión	1.0.0	
Prioridad	Alta	
Estabilidad	Alta	
Precondición	El usuario debe haber iniciado sesión en la aplicación y debe haber una sesión iniciada.	
Descripción	El usuario podrá visualizar el historial de vehículos que ha monitorizado una sesión de medición.	
Entrada	NA	
Salida	NA	
Secuencia normal	Paso	Acción
	1	El usuario accede a la aplicación.
	2	El usuario pulsa en “Show VDR History”.
	3	El sistema abre una nueva pestaña en la que se muestran los datos de emisiones de los vehículos registrados en esa sesión.
Postcondición	NA	
Excepciones	Paso	Acción

Tabla 4 – Ver historia de una sesión de medición.

CU-O-5	Descargar historia de una sesión de medición	
Versión	1.0.0	
Prioridad	Alta	
Estabilidad	Alta	
Precondición	El usuario debe haber iniciado sesión en la aplicación y debe haber una sesión iniciada.	
Descripción	El usuario podrá visualizar el historial de vehículos que ha monitorizado una sesión de medición.	
Entrada	NA	
Salida	NA	
Secuencia normal	Paso	Acción
	1	El usuario accede a la aplicación.
	2	El usuario pulsa en “Download VDR History”.
	3	El sistema devuelve un fichero con los datos de la sesión de medición.
Postcondición	NA	
Excepciones	Paso	Acción

Tabla 5 – Descargar historia de una sesión de medición.

CU-O-6	Finalizar sesión de medición	
Versión	1.0.0	
Prioridad	Alta	
Estabilidad	Alta	
Precondición	El usuario debe haber iniciado sesión en la aplicación y debe haber una sesión iniciada.	
Descripción	El usuario podrá finalizar una sesión de medición.	
Entrada	NA	
Salida	NA	
Secuencia normal	Paso	Acción
	1	El usuario accede a la aplicación.
	2	El usuario pulsa en “Stop”.
	3	El sistema finaliza la sesión de medición, y vuelve a la pantalla principal para iniciar una nueva sesión.
Postcondición	NA	
Excepciones	Paso	Acción
	3	Se produce un error al finalizar la sesión y la aplicación vuelve a la página principal con la sesión iniciada.

Tabla 6 – Iniciar sesión de medición.

4.3 Módulo administrador.

En esta sección, se detallan los casos de uso del administrador. En la Figura 5 se puede ver el diagrama de casos de uso de dicho usuario.

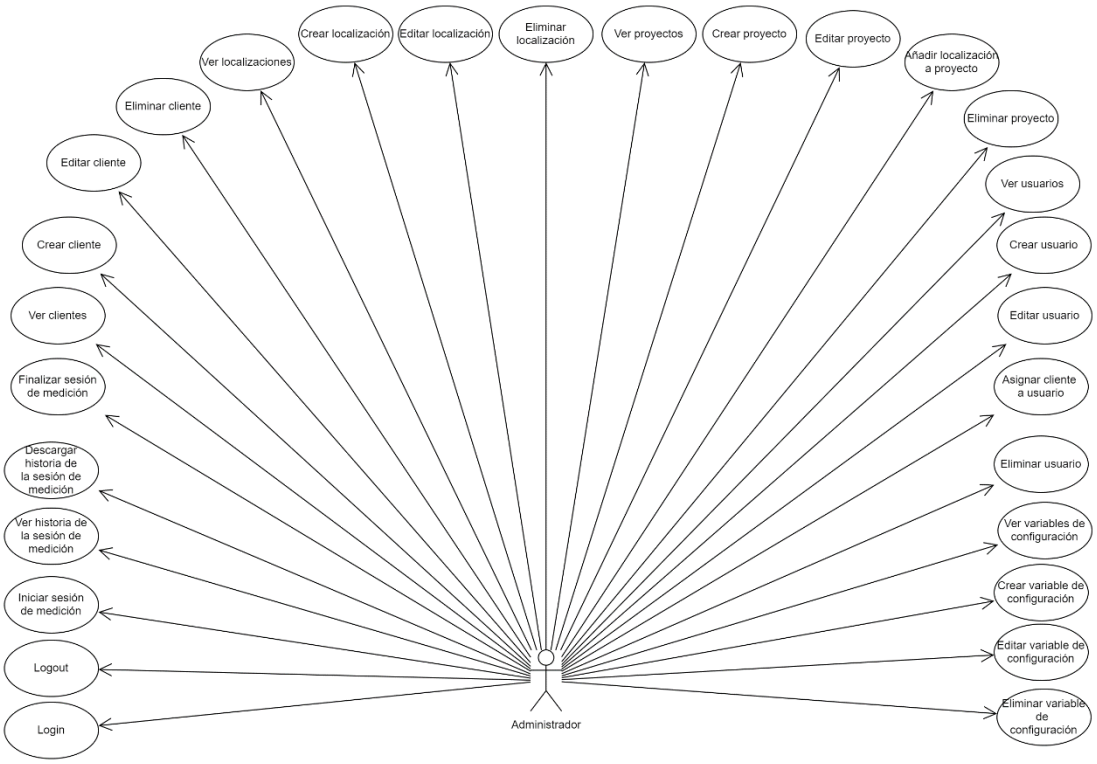


Figura 5 - Diagrama casos de uso administrador.

CU-A-1	Ver clientes	
Versión	1.0.0	
Prioridad	Alta	
Estabilidad	Alta	
Precondición	El usuario debe haber iniciado sesión en la aplicación.	
Descripción	El usuario podrá ver la lista de clientes registrados en la aplicación.	
Entrada	NA	
Salida	NA	
Secuencia normal	Paso	Acción
	1	En la parte superior de la pantalla principal el usuario debe pulsar en “Customers”.
	2	La aplicación muestra una página con la lista de clientes.
Postcondición	NA	
Excepciones	Paso	Acción

Tabla 7 – Ver clientes.

CU-A-2	Crear cliente	
Versión	1.0.0	
Prioridad	Alta	
Estabilidad	Alta	
Precondición	El usuario debe haber iniciado sesión en la aplicación.	
Descripción	El usuario podrá añadir un nuevo cliente.	
Entrada	El usuario deberá introducir la descripción del cliente.	
Salida	NA	
Secuencia normal	Paso	Acción
	1	En la parte superior de la pantalla principal el usuario debe pulsar en “Customers”.
	2	En la siguiente ventana el usuario debe pulsar en “New customer”
	3	En la siguiente ventana el usuario introducirá la descripción y pulsará en “Create”.
	4	El sistema valida que no existe un cliente con la misma descripción, registra el nuevo cliente y redirige a la página de visualización de clientes.
Postcondición	NA	
Excepciones	Paso	Acción
	4	El sistema detecta que ya existe un cliente con esa descripción, no registra el cliente y redirige a la página de visualización de clientes.

Tabla 8 – Crear cliente.

CU-A-3	Editar cliente	
Versión	1.0.0	
Prioridad	Alta	
Estabilidad	Alta	
Precondición	El usuario debe haber iniciado sesión en la aplicación y el cliente debe estar registrado en ella.	
Descripción	El usuario podrá editar un cliente existente.	
Entrada	El usuario deberá introducir la descripción del cliente.	
Salida	NA	
Secuencia normal	Paso	Acción
	1	En la parte superior de la pantalla principal el usuario debe pulsar en “Customers”.
	2	En la siguiente ventana el usuario debe pulsar en el botón “Edit” de la fila en la que se encuentra el cliente que se desea editar.
	3	En la siguiente ventana el usuario introducirá la descripción y pulsará en “Save”.
	4	El sistema valida que no existe un cliente con la misma descripción, actualiza la descripción del cliente y redirige a la página de visualización de clientes.
Postcondición	NA	
Excepciones	Paso	Acción
	4	El sistema detecta que ya existe un cliente con esa descripción, no actualiza el cliente y redirige a la página de visualización de clientes.

Tabla 9 – Editar cliente.

CU-A-4	Eliminar cliente	
Versión	1.0.0	
Prioridad	Alta	
Estabilidad	Alta	
Precondición	El usuario debe haber iniciado sesión en la aplicación y el cliente debe estar registrado en ella.	
Descripción	El usuario podrá eliminar un cliente existente.	
Entrada	NA	
Salida	NA	
Secuencia normal	Paso	Acción
	1	En la parte superior de la pantalla principal el usuario debe pulsar en “Customers”.
	2	En la siguiente ventana el usuario debe pulsar en el botón “Delete” de la fila en la que se encuentra el cliente que se desea eliminar.
	3	En la siguiente ventana, el sistema pregunta si se desea eliminar el cliente y el usuario pulsará en “Delete”.
	4	El sistema elimina el cliente y redirige a la página de visualización de clientes.
Postcondición	NA	
Excepciones	Paso	Acción

Tabla 10 – Eliminar cliente.

CU-A-5	Ver localizaciones	
Versión	1.0.0	
Prioridad	Alta	
Estabilidad	Alta	
Precondición	El usuario debe haber iniciado sesión en la aplicación.	
Descripción	El usuario podrá ver la lista de localizaciones registradas en la aplicación.	
Entrada	NA	
Salida	NA	
Secuencia normal	Paso	Acción
	1	En la parte superior de la pantalla principal el usuario debe pulsar en “Sites”.
	2	La aplicación muestra una página con la lista de localizaciones.
Postcondición	NA	
Excepciones	Paso	Acción

Tabla 11 – Ver localizaciones.

CU-A-6	Crear localización	
Versión	1.0.0	
Prioridad	Alta	
Estabilidad	Alta	
Precondición	El usuario debe haber iniciado sesión en la aplicación.	
Descripción	El usuario podrá añadir una nueva localización.	
Entrada	El usuario deberá introducir la descripción, un código de identificación de la localización, el país, la provincia, la localidad, la latitud y la longitud del lugar donde se vaya a realizar la medición.	
Salida	NA	
Secuencia normal	Paso	Acción
	1	En la parte superior de la pantalla principal el usuario debe pulsar en “Sites”.
	2	En la siguiente ventana el usuario debe pulsar en “New site”.
	3	En la siguiente ventana el usuario introducirá los datos solicitados y pulsará en “Create”.
	4	El sistema valida que no existe una localización con estos datos, la registra y redirige a la página de visualización de localizaciones.
Postcondición	NA	
Excepciones	Paso	Acción
	4	El sistema detecta que ya existe una localización con estos datos, no la registra y redirige a la página de visualización de localizaciones.

Tabla 12 – Crear localización.

CU-A-7	Editar localización	
Versión	1.0.0	
Prioridad	Alta	
Estabilidad	Alta	
Precondición	El usuario debe haber iniciado sesión en la aplicación y la localización debe estar registrada en ella.	
Descripción	El usuario podrá editar una localización existente.	
Entrada	El usuario deberá editar los campos que desee de la localización.	
Salida	NA	
Secuencia normal	Paso	Acción
	1	En la parte superior de la pantalla principal el usuario debe pulsar en “Sites”.
	2	En la siguiente ventana el usuario debe pulsar en el botón “Edit” de la fila en la que se encuentra la localización que se desea editar.
	3	En la siguiente ventana el usuario modificará los datos que desea y pulsará en “Save”.
	4	El sistema valida que no existe una localización con la misma descripción y código de identificación, actualiza los datos y redirige a la página de visualización de localizaciones.
Postcondición	NA	
Excepciones	Paso	Acción
	4	El sistema detecta que ya existe una localización con esa descripción y/o código de identificación, no actualiza la localización y redirige a la página de visualización de localizaciones.

Tabla 13 – Editar localización.

CU-A-8	Eliminar localización	
Versión	1.0.0	
Prioridad	Alta	
Estabilidad	Alta	
Precondición	El usuario debe haber iniciado sesión en la aplicación y la localización debe estar registrada en ella.	
Descripción	El usuario podrá eliminar una localización existente.	
Entrada	NA	
Salida	NA	
Secuencia normal	Paso	Acción
	1	En la parte superior de la pantalla principal el usuario debe pulsar en “Sites”.
	2	En la siguiente ventana el usuario debe pulsar en el botón “Delete” de la fila en la que se encuentra la localización que se desea eliminar.
	3	En la siguiente ventana, el sistema pregunta si se desea eliminar la localización y el usuario pulsará en “Delete”.
	4	El sistema elimina la localización y redirige a la página de visualización de localizaciones.
Postcondición	NA	
Excepciones	Paso	Acción

Tabla 14 – Eliminar localización.

CU-A-9	Ver proyectos	
Versión	1.0.0	
Prioridad	Alta	
Estabilidad	Alta	
Precondición	El usuario debe haber iniciado sesión en la aplicación.	
Descripción	El usuario podrá ver la lista de proyectos registrados en la aplicación.	
Entrada	NA	
Salida	NA	
Secuencia normal	Paso	Acción
	1	En la parte superior de la pantalla principal el usuario debe pulsar en “Projects”.
	2	La aplicación muestra una página con la lista de proyectos.
Postcondición	NA	
Excepciones	Paso	Acción

Tabla 15 – Ver proyectos.

CU-A-10	Crear proyecto	
Versión	1.0.0	
Prioridad	Alta	
Estabilidad	Alta	
Precondición	El usuario debe haber iniciado sesión en la aplicación.	
Descripción	El usuario podrá añadir un nuevo proyecto	
Entrada	El usuario deberá introducir una descripción y seleccionar el cliente al que pertenece el proyecto.	
Salida	NA	
Secuencia normal	Paso	Acción
	1	En la parte superior de la pantalla principal el usuario debe pulsar en “Projects”.
	2	En la siguiente ventana el usuario debe pulsar en “New project”
	3	En la siguiente ventana el usuario introducirá y seleccionará los datos solicitados y pulsará en “Create”.
	4	El sistema valida que no existe una un proyecto con estos datos, lo registra y redirige a la página de visualización de proyectos
Postcondición	NA	
Excepciones	Paso	Acción
	4	El sistema detecta que ya existe un proyecto con estos datos, no lo registra y redirige a la página de visualización de proyectos.

Tabla 16 – Crear proyecto.

CU-A-11	Editar proyecto	
Versión	1.0.0	
Prioridad	Alta	
Estabilidad	Alta	
Precondición	El usuario debe haber iniciado sesión en la aplicación y el proyecto debe estar registrado en ella.	
Descripción	El usuario podrá editar la descripción un proyecto existente.	
Entrada	El usuario deberá editar la descripción y seleccionar el cliente del proyecto.	
Salida	NA	
Secuencia normal	Paso	Acción
	1	En la parte superior de la pantalla principal el usuario debe pulsar en “Projects”.
	2	En la siguiente ventana el usuario debe pulsar en el botón “Edit” de la fila en la que se encuentra el proyecto que se desea editar.
	3	En la siguiente ventana el usuario modificará la descripción, seleccionará el cliente y pulsará en “Save”.
	4	El sistema valida que no existe un proyecto con la misma descripción para el mismo cliente, actualiza los datos y redirige a la página de visualización de proyectos.
Postcondición	NA	
Excepciones	Paso	Acción
	4	El sistema detecta que ya existe un proyecto con esa descripción para ese cliente, no actualiza el proyecto y redirige a la página de visualización de proyectos.

Tabla 17 – Editar proyecto.

CU-A-12	Añadir localización a proyecto	
Versión	1.0.0	
Prioridad	Alta	
Estabilidad	Alta	
Precondición	El usuario debe haber iniciado sesión en la aplicación y el proyecto debe estar registrado en ella.	
Descripción	El usuario podrá añadir localizaciones en las que se harán mediciones en un proyecto existente.	
Entrada	El usuario deberá seleccionar la localización.	
Salida	NA	
Secuencia normal	Paso	Acción
	1	En la parte superior de la pantalla principal el usuario debe pulsar en “Projects”.
	2	En la siguiente ventana el usuario debe pulsar en el botón “Edit” de la fila en la que se encuentra el proyecto al que se desea añadir la localización.
	3	En la siguiente ventana el usuario seleccionará la localización deseada, seleccionará el cliente y pulsará en “Add”.
	4	El sistema añade el site a ese proyecto, actualiza la ventana de datos del proyecto mostrando la nueva localización añadida.
Postcondición	NA	
Excepciones	Paso	Acción

Tabla 17 – Añadir localización a proyecto.

CU-A-13	Eliminar proyecto	
Versión	1.0.0	
Prioridad	Alta	
Estabilidad	Alta	
Precondición	El usuario debe haber iniciado sesión en la aplicación y el proyecto debe estar registrado en ella.	
Descripción	El usuario podrá eliminar un proyecto existente.	
Entrada	NA	
Salida	NA	
Secuencia normal	Paso	Acción
	1	En la parte superior de la pantalla principal el usuario debe pulsar en “Projects”.
	2	En la siguiente ventana el usuario debe pulsar en el botón “Delete” de la fila en la que se encuentra el proyecto que se desea eliminar.
	3	En la siguiente ventana, el sistema pregunta si se desea eliminar el proyecto y el usuario pulsará en “Delete”.
	4	El sistema elimina el proyecto y redirige a la página de visualización de proyectos.
Postcondición	NA	
Excepciones	Paso	Acción

Tabla 18 – Eliminar proyecto.

CU-A-14	Ver Usuarios	
Versión	1.0.0	
Prioridad	Alta	
Estabilidad	Alta	
Precondición	El usuario debe haber iniciado sesión en la aplicación.	
Descripción	El usuario podrá ver la lista de usuarios registrados en la aplicación.	
Entrada	NA	
Salida	NA	
Secuencia normal	Paso	Acción
	1	En la parte superior de la pantalla principal el usuario debe pulsar en “Users”.
	2	La aplicación muestra una página con la lista de usuarios.
Postcondición	NA	
Excepciones	Paso	Acción

Tabla 19 – Ver usuarios.

CU-A-15	Crear usuario	
Versión	1.0.0	
Prioridad	Alta	
Estabilidad	Alta	
Precondición	El usuario debe haber iniciado sesión en la aplicación.	
Descripción	El usuario podrá añadir un nuevo usuario.	
Entrada	El usuario deberá introducir un email, un nombre de usuario, una contraseña y seleccionar el rol que se desea asignar.	
Salida	NA	
Secuencia normal	Paso	Acción
	1	En la parte superior de la pantalla principal el usuario debe pulsar en “Users”.
	2	En la siguiente ventana el usuario debe pulsar en “New user”.
	3	En la siguiente ventana el usuario introducirá los datos solicitados y pulsará en “Create”.
	4	El sistema valida que no existe un usuario con el mismo nombre de usuario, la registra y redirige a la página de visualización de usuarios.
Postcondición	NA	
Excepciones	Paso	Acción
	4	El sistema detecta que ya existe un usuario con ese nombre de usuario, no la registra y redirige a la página de visualización de usuarios.

Tabla 20 – Crear usuario.

CU-A-16	Editar usuario	
Versión	1.0.0	
Prioridad	Alta	
Estabilidad	Alta	
Precondición	El usuario debe haber iniciado sesión en la aplicación y el usuario debe estar registrado en ella.	
Descripción	El usuario podrá editar un usuario existente.	
Entrada	El usuario deberá editar los campos que desee del usuario.	
Salida	NA	
Secuencia normal	Paso	Acción
	1	En la parte superior de la pantalla principal el usuario debe pulsar en “Users”.
	2	En la siguiente ventana el usuario debe pulsar en el botón “Edit” de la fila en la que se encuentra el usuario que se desea editar.
	3	En la siguiente ventana el usuario modificará los datos que desea y pulsará en “Save”.
	4	El sistema valida que no existe un usuario con el mismo nombre de usuario, actualiza los datos y redirige a la página de visualización de localizaciones.
Postcondición	NA	
Excepciones	Paso	Acción
	4	El sistema detecta que ya existe un usuario con ese nombre de usuario, no actualiza la localización y redirige a la página de visualización de localizaciones.

Tabla 21 – Editar usuario.

CU-A-17	Asignar cliente a usuario	
Versión	1.0.0	
Prioridad	Alta	
Estabilidad	Alta	
Precondición	El usuario debe haber iniciado sesión en la aplicación y el usuario y cliente deben estar registrado en ella.	
Descripción	El usuario podrá asignar un cliente para el cual hará realizará las sesiones de medición.	
Entrada	El usuario deberá seleccionar el cliente.	
Salida	NA	
Secuencia normal	Paso	Acción
	1	En la parte superior de la pantalla principal el usuario debe pulsar en “Users”.
	2	En la siguiente ventana el usuario debe pulsar en el botón “Edit” de la fila en la que se encuentra el usuario al que se desea asignar el cliente.
	3	En la siguiente ventana el usuario seleccionará el cliente deseado, y pulsará en “Add”.
	4	El sistema asigna el cliente a ese usuario, actualiza la ventana de datos del usuario mostrando el nuevo cliente asignado.
Postcondición	NA	
Excepciones	Paso	Acción
Comentarios	Este caso de uso solo aplica a usuarios con rol operador, ya que el administrador puede ver todos los clientes.	

Tabla 22 – Asignar cliente a usuario.

CU-A-18	Eliminar usuario	
Versión	1.0.0	
Prioridad	Alta	
Estabilidad	Alta	
Precondición	El usuario debe haber iniciado sesión en la aplicación, el usuario debe estar registrado en ella y realizado ninguna sesión de medición.	
Descripción	El usuario podrá eliminar un usuario existente.	
Entrada	NA	
Salida	NA	
Secuencia normal	Paso	Acción
	1	En la parte superior de la pantalla principal el usuario debe pulsar en “Users”.
	2	En la siguiente ventana el usuario debe pulsar en el botón “Delete” de la fila en la que se encuentra el usuario que se desea eliminar.
	3	En la siguiente ventana, el sistema pregunta si se desea eliminar el usuario y el usuario pulsará en “Delete”.
	4	El sistema comprueba que haya sesiones realizadas por ese usuario, elimina el usuario y redirige a la página de visualización de usuarios.
Postcondición	NA	
Excepciones	Paso	Acción

Tabla 23 – Eliminar usuario.

CU-A-19	Ver variables de configuración	
Versión	1.0.0	
Prioridad	Alta	
Estabilidad	Alta	
Precondición	El usuario debe haber iniciado sesión en la aplicación.	
Descripción	El usuario podrá ver la lista de variables de configuración registradas en la aplicación.	
Entrada	NA	
Salida	NA	
Secuencia normal	Paso	Acción
	1	En la parte superior de la pantalla principal el usuario debe pulsar en “Configuration”.
	2	La aplicación muestra una página con la lista de variables.
Postcondición	NA	
Excepciones	Paso	Acción

Tabla 24 – Ver variables de configuración.

CU-A-20	Crear variable de configuración	
Versión	1.0.0	
Prioridad	Alta	
Estabilidad	Alta	
Precondición	El usuario debe haber iniciado sesión en la aplicación.	
Descripción	El usuario podrá añadir una nueva variable de configuración.	
Entrada	El usuario deberá introducir el nombre y el valor de la variable.	
Salida	NA	
Secuencia normal	Paso	Acción
	1	En la parte superior de la pantalla principal el usuario debe pulsar en “Configuration”.
	2	En la siguiente ventana el usuario debe pulsar en “New param”.
	3	En la siguiente ventana el usuario introducirá los datos solicitados y pulsará en “Create”.
	4	El sistema valida que no existe una variable con este nombre, la registra y redirige a la página de visualización de variables de configuración.
Postcondición	NA	
Excepciones	Paso	Acción
	4	El sistema detecta que ya existe una variable con este nombre, no la registra y redirige a la página de visualización de localizaciones.

Tabla 25 – Crear variable de configuración.

CU-A-21	Editar variable de configuración	
Versión	1.0.0	
Prioridad	Alta	
Estabilidad	Alta	
Precondición	El usuario debe haber iniciado sesión en la aplicación y la variable debe estar registrada en ella.	
Descripción	El usuario podrá editar una variable de configuración existente.	
Entrada	El usuario deberá editar los campos que desee de la variable.	
Salida	NA	
Secuencia normal	Paso	Acción
	1	En la parte superior de la pantalla principal el usuario debe pulsar en “Configuration”.
	2	En la siguiente ventana el usuario debe pulsar en el botón “Edit” de la fila en la que se encuentra la variable que se desea editar.
	3	En la siguiente ventana el usuario modificará los datos que desea y pulsará en “Save”.
	4	El sistema valida que no existe variable con el mismo nombre, actualiza los datos y redirige a la página de visualización de variables.
Postcondición	NA	
Excepciones	Paso	Acción
	4	El sistema detecta que ya existe una variable con el mismo nombre, no actualiza la variable y redirige a la página de visualización de variables.

Tabla 26 – Editar variable de configuración

CU-A-22	Eliminar variable de configuración	
Versión	1.0.0	
Prioridad	Alta	
Estabilidad	Alta	
Precondición	El usuario debe haber iniciado sesión en la aplicación y la variable debe estar registrada en ella.	
Descripción	El usuario podrá eliminar una variable existente.	
Entrada	NA	
Salida	NA	
Secuencia normal	Paso	Acción
	1	En la parte superior de la pantalla principal el usuario debe pulsar en “Configuration”.
	2	En la siguiente ventana el usuario debe pulsar en el botón “Delete” de la fila en la que se encuentra la variable que se desea eliminar.
	3	En la siguiente ventana, el sistema pregunta si se desea eliminar la variable y el usuario pulsará en “Delete”.
	4	El sistema elimina la variable y redirige a la página de visualización de localizaciones.
Postcondición	NA	
Excepciones	Paso	Acción

Tabla 27 – Eliminar variable de configuración.

Capítulo 5. Arquitectura.

En este capítulo se describe la arquitectura del sistema desarrollado.

La arquitectura del sistema se basa en el modelo cliente(s) – servidor.

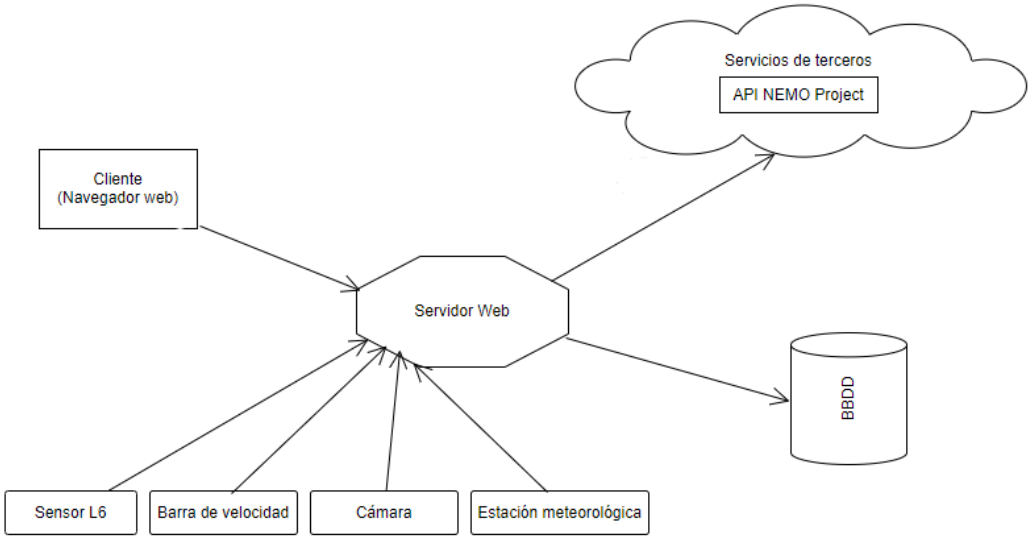


Figura 6 - Arquitectura del sistema.

Como se observa en la Figura 6, el navegador web interactúa con el servidor a través de la aplicación web.

En el lado del servidor, se procesa cada petición del cliente aplicando la lógica de negocio de cada acción.

Los periféricos interactúan con el servidor mediante protocolo TCP, a través de una red interna.

La comunicación con la API del proyecto NEMO se realiza mediante envíos a una cola de Apache ActiveMQ.

A continuación, se va a detallar el Framework de MVC que se ha utilizado para el desarrollo de la aplicación.

Es un conjunto de plantillas de Visual Studio y bibliotecas que permiten crear aplicaciones ASP.Net basadas en el patrón de diseño Modelo-Vista-Controlador (MVC) (Figura 7).

El modelo Modelo-Vista-Controlador (MVC) separa una aplicación en tres componentes: el modelo, la vista y el controlador.

ASP.NET MVC proporciona una alternativa al modelo de formularios Web Forms de ASP.NET para crear aplicaciones web, y un marco de presentación de poca complejidad y fácil de comprobar que se integra con las características de ASP.NET existentes, tales como páginas maestras y la autenticación basada en pertenencia.

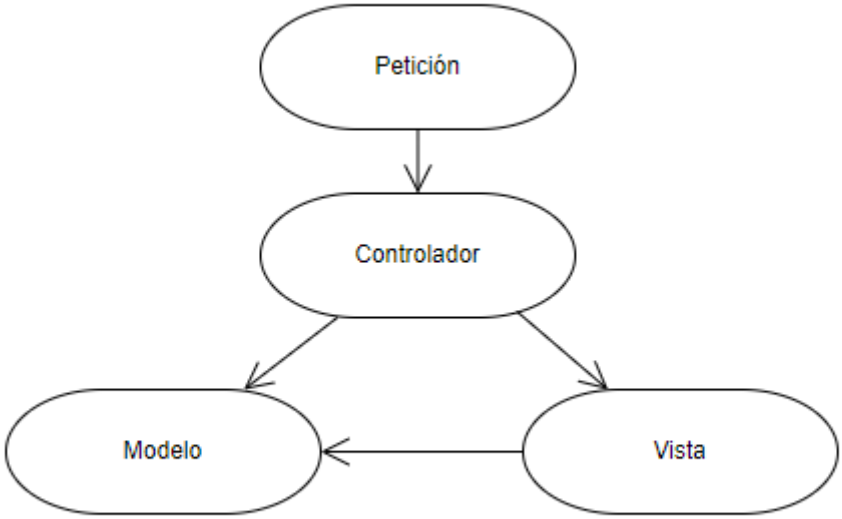


Figura 7 - Esquema MVC.

Los proyectos basados en MVC parten de la plantilla “Aplicación web ASP.NET (.NET Framework)” que ofrece Visual Studio, de manera que al crear el proyecto se crean automáticamente las carpetas que almacenaran las diferentes clases de los modelos, las vistas y los controladores.

A continuación, se describen cada una de las capas:

5.1 Modelo.

Los objetos de modelo son las partes de la aplicación que implementan la lógica para el dominio de datos de la aplicación. A menudo, los objetos de modelo recuperan y almacenan el estado del modelo en una base de datos. Por ejemplo, un objeto Site podría recuperar información de la base de datos, operar en ella y, a continuación, volver a escribir información actualizada en una tabla Site en SQL Server.

Para la creación de modelo se ha utilizado Entity Framework Database First, que permite aplicar ingeniería inversa a un modelo de datos de una base de datos existente.

De esta forma, cada una de las tablas de la base de datos se corresponde con una clase.

Se muestra un ejemplo de la entidad y la clase Customer en las Figura 8 y Figura 9.

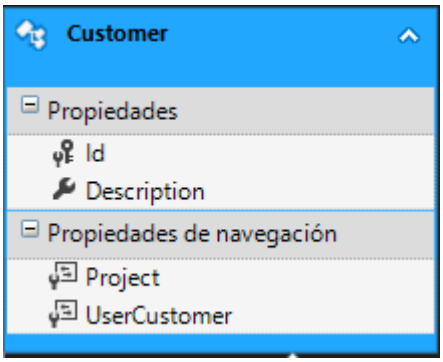


Figura 8 - Entidad Customer.

```
namespace L6WebApp.Models
{
    using System;
    using System.Collections.Generic;

    10 referencias
    public partial class Customer
    {
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]
        0 referencias
        public Customer()
        {
            this.Project = new HashSet<Project>();
            this.UserCustomer = new HashSet<UserCustomer>();
        }

        7 referencias
        public int Id { get; set; }
        7 referencias
        public string Description { get; set; }

        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]
        1 referencia
        public virtual ICollection<Project> Project { get; set; }
        [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]
        1 referencia
        public virtual ICollection<UserCustomer> UserCustomer { get; set; }
    }
}
```

Figura 9 - Clase Customer.

5.2 Vista.

Las vistas son los componentes que muestran la interfaz de usuario (UI) de la aplicación.

Normalmente, esta interfaz de usuario se crea a partir de los datos de modelo.

Para la creación de estas vistas, Visual Studio utiliza Bootstrap como framework de interfaz de usuario.

5.3 Controlador.

Los controladores son los componentes que controlan la interacción del usuario, trabajan con el modelo y por último seleccionan una vista para representar la interfaz de usuario. En una aplicación de MVC, la vista solo muestra información; el controlador controla y responde a la interacción y los datos que introducen los usuarios.

Este framework permite configurar el mapeo de URL's a distintas clases del controlador. Por defecto las URL's tipo /x/ se asocian con la clase xController.

Su funcionamiento se describe en la Figura 10:

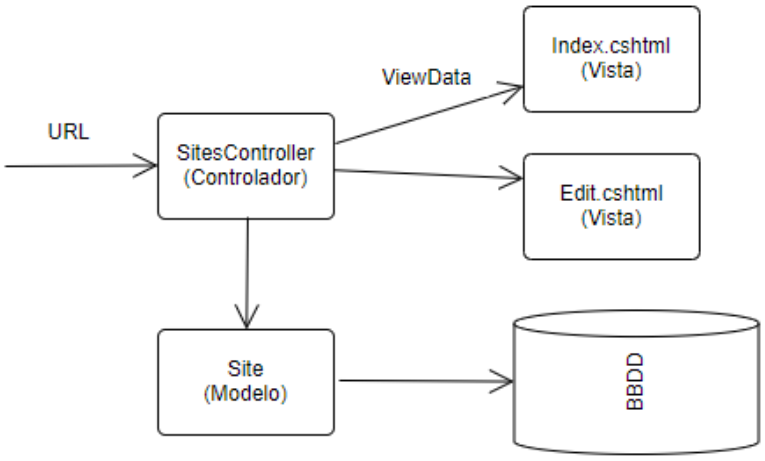


Figura 10 - Petición URL.

La clase del controlador es la que decide que lógica ha de ejecutarse en función de la petición realizada en la URL, transformando dicha URL en llamadas a los métodos de la clase.

Siguiendo lo anterior, la URL "/Sites/Index" invoca al método Index del controlador Sites, que una vez "controlada" la acción de la aplicación deberá entregar u obtener del dato al Modelo y visualizarlo en la Vista.

Capítulo 6. Modelo de datos.

En este capítulo se describe el modelo de datos desarrollado para el sistema. Este, ha sido diseñado para almacenar los datos de las emisiones de un vehículo a lo largo de una sesión de medición, y además permite almacenar los datos relativos a dicha sesión, como son el usuario que la realiza, y el proyecto y la localización del cliente para el que se esta llevando a cabo esa sesión.

6.1 Implementación de la base de datos.

En la Figura 11 se muestra el diagrama de la base de datos.

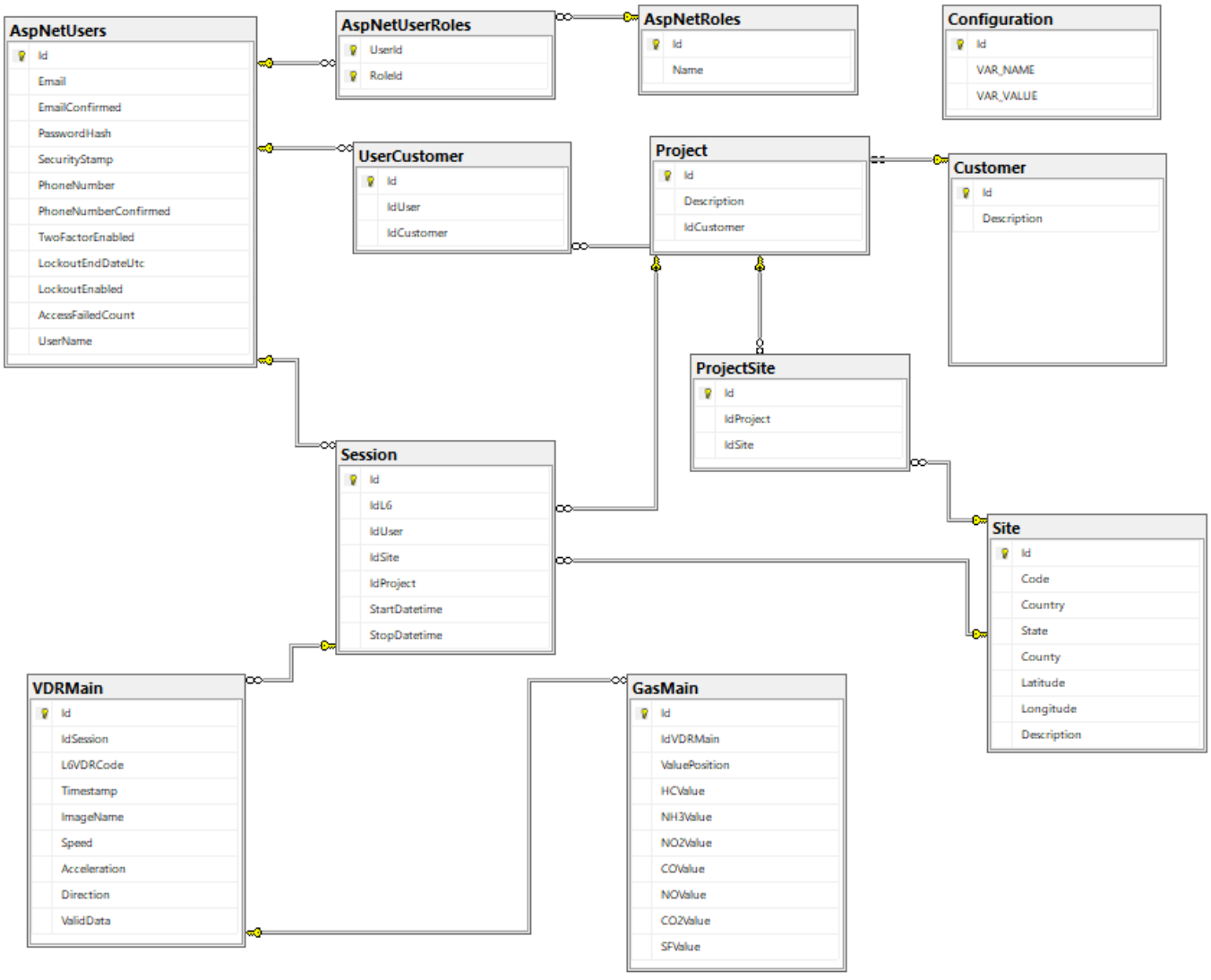


Figura 11 - Diagrama de base de datos.

A continuación, se van a describir cada una de las tablas de la base de datos.

6.2 Descripción de tablas de base de datos.

1) Tabla AspNetUsers: representa un usuario en el modelo Identity de ASP.NET de Microsoft. Cuenta con gran cantidad de campos, aunque para nuestro sistema utilizaremos los que se describen a continuación:
   - Id: Identificador del usuario.
   - Email: Correo electrónico del usuario.
   - PasswordHash: Hash de la contraseña del usuario.
   - UserName: Nombre de usuario.

2) Tabla AspNetRoles: representa un rol en el modelo Identity de ASP.NET de Microsoft.
   - Id: Identificador del rol.
   - Name: Nombre del rol.

3) Tabla AspNetUserRoles: entidad de combinación que asocia usuarios y roles.
   - UserId: Identificador del usuario.
   - RoleId: Identificador del rol.

4) Tabla Configuration: guarda la información de las variables de configuración del sistema.
   - Id: Identificador de la variable.
   - VAR_NAME: Nombre de la variable.
   - VAR_VALUE: Valor de la variable.

5) Tabla Configuration: guarda la información de las variables de configuración del sistema.
   - Id: Identificador de la variable.
   - VAR_NAME: Nombre de la variable.
   - VAR_VALUE: Valor de la variable.

6) Tabla Customer: representa un cliente.
   - Id: Identificador de la variable.
   - Description: Descripción del cliente.

7) Tabla UserCustomer: entidad de combinación que asocia usuarios y clientes.
   - Id: Identificador de la relación.
   - IdUser: Identificador del usuario.
   - IdCustomer: Identificador del cliente.

8) Tabla Project: guarda información de un proyecto.
   - Id: Identificador del proyecto.
   - Description: Descripción del proyecto.
   - IdCustomer: Identificador del cliente al que pertenece el proyecto.

9) Tabla Site: guarda información de las localizaciones.
    - Id: Identificador de la localización.
    - Code: Código de identificación de la localización.
    - Country: País de la localización.
    - State: Provincia de la localización.
    - County: Localidad de la localización.
    - Latitude: Latitud de la localización.
    - Longitude: Longitud de la localización.
    - Description: Descripción larga de la localización.

10) Tabla ProjectSite: entidad de combinación que asocia proyectos y localizaciones.
    - Id: Identificador de la relación.
    - IdProject: Identificador del proyecto.
    - IdSite: Identificador del site.

11) Tabla Session: guarda la información de una sesión de medición.
    - Id: Identificador de la sesión.
    - IdL6: Identificador del sistema en el que se realiza la sesión, será el valor de una variable de configuración.
    - IdUser: Identificador del usuario que inicia la sesión.
    - IdSite: Identificador de la localización en la que se realiza la sesión.
    - IdProject: Identificador del proyecto al que pertenece la sesión.
    - StartDatetime: Marca de tiempo de inicio de la sesión.
    - StopDatetime: Marca de tiempo de finalización de la sesión.

12) Tabla VDRMain: guarda la información de cada vehículo leído por el sensor.
    - Id: Identificador del vehículo para la base de datos.
    - L6VDRCode: Identificador del vehículo establecido por el sensor.
    - Timestamp: Marca de tiempo de la lectura del vehículo.
    - ImageName: Nombre de la imagen asociada a la lectura del vehículo
    - Speed: Velocidad del vehículo en la lectura.
    - Direction: Sentido del desplazamiento del vehículo con respecto del sensor.
    - ValidData: Flag que indica si la lectura del vehículo es válida o no.

13) Tabla GasMain: guarda la información de los valores de la densidad de columna de cada gas a lo largo de 32 puntos (ValuePosition) por cada lectura de vehículo. Además, se añade un punto 33 que contiene el ratio de cada gas.
    - Id: Identificador de cada punto.
    - IdVDRMain: Identificador de la lectura del vehículo.
    - ValuePosition: Indica la posición del 1 al 33 de cada uno de los puntos.
    - HCValue: Valor de la densidad de columna de HC en el punto.
    - NH3Value: Valor de la densidad de columna de NH3 en el punto.
    - NO2Value: Valor de la densidad de columna de NO2 en el punto.
    - COValue: Valor de la densidad de columna de CO en el punto.
    - NOValue: Valor de la densidad de columna de NO en el punto.
    - CO2Value: Valor de la densidad de columna de CO2 en el punto.
    - SFValue: Valor de la densidad de columna de SF (Smoke Factor u opacidad) en el punto.

Capítulo 7. Diseño.

En este capítulo se detallan los aspectos principales en cuanto a la funcionalidad y el diseño realizado para el proyecto.

7.1 Estilo.

Para esta aplicación se decidió crear un estilo sencillo y una interfaz simple e intuitiva.

En cuanto a los colores de la interfaz, se decidió establecer un mínimo de colores personalizados para las zonas principales, los cuales coinciden en su mayoría con los colores corporativos de la compañía. Estos colores son el blanco y una pequeña gama de verdes y azules.

Para otras partes de la aplicación se decidió utilizar los colores que se establecen en por defecto en Bootstrap, que es el Framework de estilos utilizado.

7.2 Funcionalidad de la Web(L6WebApp).

En este apartado se describen las funcionalidades desarrolladas en la aplicación web, de la cual se muestran las vistas más importantes.

7.2.1 Dashboard

Es la pantalla principal y la más importante de la aplicación por la cantidad de información que contiene y la lógica que se ha implementado para ella.

En la parte superior se encuentra una cabecera, que es compartida por todas las páginas de la aplicación, que muestra las diferentes secciones a las que podemos acceder.

Las secciones que se muestran dependerán del rol del usuario con el que se inicie la sesión, de manera que para el administrador se mostrarán todas las secciones disponibles (Figura 12) y en caso de que no tenga este rol, únicamente se mostrará la sección Home.

Figura 12 - Imagen de la cabecera de la web.

Cada uno de los botones de la cabecera llama a la vista Index (Figura 13) de cada uno de los controladores.

```
@if (Request.IsAuthenticated)
{
    <nav class="navbar navbar-expand-lg navbar-custom bg-custom fixed-top">
        <div class="container-fluid">
            <img class="navbar-brand navbar-custom-profile-image-pic" src="~/Content/img/opurse.png">
            <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarSupportedContent">
                <ul class="navbar-nav me-auto mb-2 mb-lg-0">
                    <li>@Html.ActionLink("Home", "Index", "Home", routeValues: null, htmlAttributes: new { @class = "nav-link" })</li>
                    @if (User.IsInRole("Admin"))
                    {
                        <li>@Html.ActionLink("Customers", "Index", "Customers", routeValues: null, htmlAttributes: new { @class = "nav-link" })</li>
                        <li>@Html.ActionLink("Projects", "Index", "Projects", routeValues: null, htmlAttributes: new { @class = "nav-link" })</li>
                        <li>@Html.ActionLink("Sites", "Index", "Sites", routeValues: null, htmlAttributes: new { @class = "nav-link" })</li>
                        <li>@Html.ActionLink("Users", "Index", "Account", routeValues: null, htmlAttributes: new { @class = "nav-link" })</li>
                        <li>@Html.ActionLink("Configuration", "Index", "Configurations", routeValues: null, htmlAttributes: new { @class = "nav-link" })</li>
                    }
                </ul>
                @Html.Partial("_LoginPartial")
            </div>
        </div>
    </nav>
}
```

Figura 13 - Código HTML de la cabecera.

Esta página viene como resultado de solicitar la vista Index al controlador HomeController (Figura 14).

```
public class HomeController : Controller
{
    private L6_DBEntities db = new L6_DBEntities();
    0 referencias
    public ActionResult Index()
    {
        string idL6 = db.Configuration.Where(x => x.VAR_NAME == "L6_IDENTIFIER").Select(x => x.VAR_VALUE).FirstOrDefault();
        ViewBag.IdL6 = idL6;

        Session session = db.Session.Where(s => s.StopDatetime == null).FirstOrDefault();

        if (session != null)
        {
            ViewBag.IdSession = session.Id;
            ViewBag.IdSiteSession = session.Site.Id;
            ViewBag.IdProjectSession = session.Project.Id;
            ViewBag.IdCustomerSession = session.Project.Customer.Id;
            ViewBag.StartDatetimeSession = session.StartDatetime.Value.ToString("yyyy-MM-dd HH:mm:ss");
        }

        return View();
    }
```

Figura 14 - Método Index del controlador HomeController.

En el caso de no haber una sesión de medición iniciada, como se muestra en la Figura 15, se permite iniciar una nueva sesión de medición, previo a la selección de una serie de valores, los cuales se han ido precargando en base a diferentes eventos.

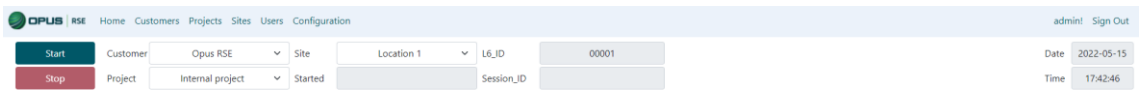


Figura 15 - Imagen de la web sin sesión de medición iniciada.

Los clientes se cargan dentro del evento $(document).ready(function () {…} (Figura 16) de Jquery de la página.

```
$.ajax({
    url: '/Customers/GetMyCustomersOrAdmin',
    contentType: 'application/json ; charset:utf-8',
    type: 'GET',
    dataType: 'json',
    success: function (result) {
        $.each(result, function (key, value) {
            customers.append($("<option></option>")
                .attr("value", value.Id)
                .text(value.Description));
        });
        if (session != '') {
            $('#customer option[value="@ViewBag.IdCustomerSession"]').attr("selected", true)
        }
        dselect(document.querySelector('#customer'), {
            search: true
        });
        if (session != '')...
    },
    error: function () {
        console.log('error');
    }
});
```

Figura 16 - Código de carga de clientes.

Esta llamada realiza una petición al método GetMyCustomersOrAdmin (Figura 17) del controlador CustomersController, el cual devolverá todos los clientes si el usuario con el que se ha iniciado la aplicación es administrador o por el contrario solo mostrara los clientes que tenga asignado ese usuario.

```
public JsonResult GetMyCustomersOrAdmin()
{
    bool IsAdmin = User.IsInRole("Admin");
    if (!IsAdmin)
    {
        string IdUser = User.Identity.GetUserId();
        var customersAssigned = db.UserCustomer.Where(ps => ps.IdUser == IdUser).Select(ps => ps.IdCustomer).ToList();
        return Json(db.UserCustomer.Where(s => customersAssigned.Contains(s.IdCustomer)).Select(s => new { Id = s.Customer.Id, Description = s.Customer.Description }), JsonRequestBehavior.AllowGet);
    }
    else
    {
        return Json(db.Customer.Select(s => new { s.Id, s.Description }), JsonRequestBehavior.AllowGet);
    }
}
```

Figura 17 - Método GetsMyCustomersOrAdmin del controlador CustomersController.

Una vez seleccionado el cliente, en el evento $('#customer').change(function () {…}, se cargarán los proyectos de este cliente, siguiendo la misma metodología.

De igual manera, una vez se seleccione un proyecto, se cargarán los sites de dicho proyecto siguiendo la misma metodología en el evento correspondiente.

Una vez seleccionados se utiliza el evento onClick de Jquery del botón de Start (Figura 18). Este evento realiza una llamada asíncrona utilizando Ajax al método NewSession del controlador SessionsController (Figura 19), que será el encargado de crear una nueva sesión y devolver los datos correspondientes de la misma, como se ve en la Figura 20.

```
$('#btnStart').on('click', function (e) {
    var valueSite = $("#site").val();
    var valueProject = $("#project").val();
    var idSession;
    $.ajax({
        url: '/Sessions/NewSession',
        contentType: 'application/text ; charset:utf-8',
        type: 'GET',
        data: { site: valueSite, project: valueProject },
        dataType: 'text',
        success: function (result) {
            $('#btnStart').prop("disabled", true);
            $('#btnStop').prop("disabled", false);
            $('#startdatetime').val(result.split(';')[1]);
            idSession = result.split(';')[0];
            $('#sessionId').val(result.split(';')[0]);
            $('button.form-select').prop('disabled', true);

        },
        error: function () {
            console.log('error');
        }
    }).done(function () {

        getAllMessages();
        $("#container").show();
        $("#showHistory").attr('href', '/VDRMains?IdSession=' + idSession);
    });
});
```

Figura 18 - Evento OnClick del botón Start.

```
0 referencias
public string NewSession(string site, string project) {
    int _site = int.Parse(site);
    int _project = int.Parse(project);
    Session s = new Session();
    s.IdL6 = db.Configuration.Where(c => c.VAR_NAME == "L6_IDENTIFIER").Select(c => c.VAR_VALUE).FirstOrDefault();
    s.IdUser = db.AspNetUsers.Where(u => u.UserName == User.Identity.Name).Select(u => u.Id).FirstOrDefault();
    s.IdSite = _site;
    s.IdProject = _project;
    s.StartDatetime = DateTime.Now;

    db.Session.Add(s);
    db.SaveChanges();

    return s.Id.ToString() + ";" + s.StartDatetime.Value.ToString("yyyy-MM-dd HH:mm:ss");
}
```

Figura 19 - Método NewSession del controlador SessionsController.

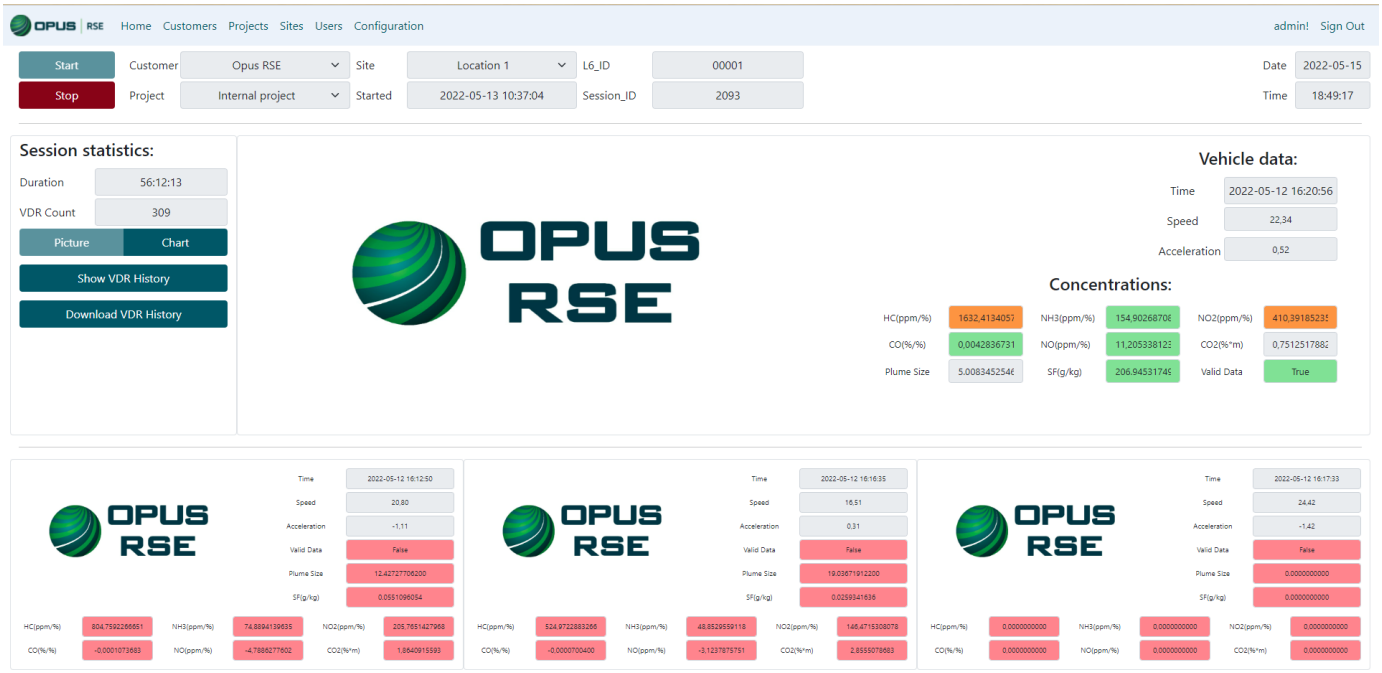


Figura 20 - Imagen de la web con sesión de medición iniciada.

Cuando se carga la página principal, SignalR establece una conexión el servidor mediante un hub (Figura 21), que será el encargado de notificar los cambios que se produzcan en la base de datos.

```
$(function () {

    var notifications = $.connection.myHub;

    notifications.client.updateMessages = function () {
        getAllMessages()
    };
    // Start the connection.
    $.connection.hub.start().done(function () {
        getAllMessages()
    }).fail(function (e) {
        alert(e);
    });
    $.connection.hub.disconnected(function () {
        setTimeout(function () {
            $.connection.hub.start();
        }, 2000); // Restart connection after 5 seconds.
    });
});
```

Figura 21 - Código establecimiento de conexión hub de SignalR.

La conexión a través de hubs con SignalR ofrece una mayor abstracción, permitiendo la comunicación entre cliente y servidor de forma casi mágica (Figura 22). De esta forma se consigue que cada vez que el servidor detecte que hay cambios en la base de datos, se envíe un mensaje de actualización a los clientes conectados.

```
[HubName("myHub")]
3 referencias
public class MyHub : Hub
{
    [HubMethodName("sendMessages")]
    2 referencias
    public static void SendMessages()
    {
        IHubContext context = GlobalHost.ConnectionManager.GetHubContext<MyHub>();
        context.Clients.All.updateMessages();
    }
}
```

Figura 22 - Clase MyHub.

Cuando el servidor notifica un cambio, se lo comunica a todos los clientes a través de la función updateMessages(), y seguidamente todos los clientes ejecutan la función getAllMessages() que es la encargada de traer los nuevos datos al cliente y actualizar todos los componentes de esta pantalla.

La función getAllMessages() (Figura 23) se encarga de traer los datos de los cuatro últimos registros que haya en la tabla VDRMain ordenados por el campo Timestamp de forma descendente y de devolver una serie de vistas parciales componen la pantalla principal.

```
function getAllMessages() {

    var tbl = $('#tbCustomers');
    var tbl1 = $('#tbCustomers1');
    var tbl2 = $('#tbCustomers2');
    var tbl3 = $('#tbCustomers3');
    var sessionId = $('#sessionId').val();
    /*alert(sessionId);*/
    if (sessionId == '') {
        sessionId = '-1';
    }
    if (sessionId != '') {
    $.ajax({
        url: '/VDRMains/GetAllMessages',
        contentType: 'application/html ; charset:utf-8',
        type: 'GET',
        data: { IdSession: parseInt($('#sessionId').val()) },
        dataType: 'json',
        success: function (result) {
            var res = JSON.stringify(result);
            //if (result[0] != null) {
                $.ajax({
                    url: '/VDRMains/GetMessages',
                    contentType: 'application/html ; charset:utf-8',
                    type: 'GET',
                    data: { i: result[0], ichart: '0' },
                    dataType: 'html',
                    success: function (data) {
                        tbl.empty().append(data);

                        //if (result[1] != null) {
                            $.ajax({
                                url: '/VDRMains/GetMessagesMini',
                                contentType: 'application/html ; charset:utf-8',
                                type: 'GET',
                                data: { i: result[1], ichart: '1' },
                                dataType: 'html',
                                success: function (data) {
                                    tbl1.empty().append(data);
                                    //if (result[2] != null) {
                                        $.ajax({
```

Figura 23 - Parte del código de la función getAllMessages().

Dentro de esta función se llama a diferentes métodos del controlador VDRMainsController.

La primera llamada se realiza al método GetAllMessages (Figura 24), que recibiendo el Id de la sesión de medición establece la dependencia con el servidor de base de datos para las notificaciones de los cambios en esta tabla y devuelve un Json con los Id de los cuatro últimos registros de la tabla VDRMain.

```
public JsonResult GetAllMessages(int IdSession)
{
    string connectionString = db.Database.Connection.ConnectionString;
    List<int> l;
    string commandText = null;
    var parameters = new SqlParameter[0];

    var query = db.VDRMain.Where(x => x.IdSession == IdSession) as DbQuery<VDRMain>;

    //var query = db.GasMain.Where(x => x.IdVDRMain == vdr) as DbQuery<GasMain>;

    commandText = query.ToString();

    var internalQuery = query.GetType()
        .GetFields(BindingFlags.NonPublic | BindingFlags.Instance)
        .Where(field => field.Name == "_internalQuery")
        .Select(field => field.GetValue(query))
        .First();

    var objectQuery = internalQuery.GetType()
        .GetFields(BindingFlags.NonPublic | BindingFlags.Instance)
        .Where(field => field.Name == "_objectQuery")
        .Select(field => field.GetValue(internalQuery))
        .Cast<ObjectQuery<VDRMain>>()
        .First();

    parameters = objectQuery.Parameters.Select(x => new SqlParameter(x.Name, x.Value)).ToArray();


    using (SqlConnection connection = new SqlConnection(connectionString))
    {
        using (SqlCommand command = new SqlCommand(commandText, connection))
        {
            connection.Open();

            command.Parameters.AddRange(parameters);

            var sqlDependency = new SqlDependency(command);

            sqlDependency.OnChange += new OnChangeEventHandler(dependency_OnChange);

            // NOTE: You have to execute the command, or the notification will never fire.
            using (SqlDataReader reader = command.ExecuteReader())
            {
                l = query.OrderByDescending(s => s.Id).Take(4).Select(s => s.Id).ToList();

            }
        }
    }
    return Json(l, JsonRequestBehavior.AllowGet);
}
```

Figura 24 - Método GetAllMessages del controlador VDRMainsController.

Una vez recibida la respuesta de este método se procede de la siguiente manera.

La pantalla principal muestra información de los cuatro últimos vehículos que han sido registrados por el sistema. Siendo el último vehículo registrado el que ocupa la mayor parte de la pantalla.

Para mostrar los datos de este, se realiza una llamada al método GetMessages (Figura 25) del controlador VDRMainsController, que devolverá una vista parcial con todos los datos del registro.

```
public ActionResult GetMessages(string i, string ichart)
{

    if (i == null)
    {
        i = "-1";
    }
    int _id = int.Parse(i);

    VDRMain v = db.VDRMain.Include(g => g.GasMain).Where(g => g.Id == _id).FirstOrDefault();
    List<Data> HCList = new List<Data>();
    if (v != null)
    {
        foreach (var HC in v.GasMain.Where(g => g.ValuePosition >= 1 && g.ValuePosition <= 32).OrderBy(s => s.ValuePosition).Select(s => new { s.HCValue, s.ValuePosition }).ToList())
        {
            HCList.Add(new Data(HC.ValuePosition, HC.HCValue ?? 0));
        }
    }
    else
    {
        HCList = new List<Data> { };
    }
```

…

```
        ViewBag.HCList = JsonConvert.SerializeObject(HCList);
        ViewBag.NH3List = JsonConvert.SerializeObject(NH3List);
        ViewBag.NO2List = JsonConvert.SerializeObject(NO2List);
        ViewBag.COList = JsonConvert.SerializeObject(COList);
        ViewBag.NOList = JsonConvert.SerializeObject(NOList);
        ViewBag.CO2List = JsonConvert.SerializeObject(CO2List);
        ViewBag.SFList = JsonConvert.SerializeObject(SFList);
        ViewBag.ChartName = "MyChart" + ichart;


        if (v != null)
        {
            List<Configuration> configurations = db.Configuration.Where(x => x.VAR_NAME.Contains("Limit")).ToList();
            ViewBag.HC = v.GasMain.Where(x => x.ValuePosition == 33).FirstOrDefault().HCValue.ToString().Replace(",", ".");
            ViewBag.NH3 = v.GasMain.Where(x => x.ValuePosition == 33).FirstOrDefault().NH3Value.ToString().Replace(",", ".");
            ViewBag.NO2 = v.GasMain.Where(x => x.ValuePosition == 33).FirstOrDefault().NO2Value.ToString().Replace(",", ".");
            ViewBag.CO = v.GasMain.Where(x => x.ValuePosition == 33).FirstOrDefault().COValue.ToString().Replace(",", ".");
            ViewBag.NO = v.GasMain.Where(x => x.ValuePosition == 33).FirstOrDefault().NOValue.ToString().Replace(",", ".");
            ViewBag.CO2 = v.GasMain.Where(x => x.ValuePosition == 33).FirstOrDefault().NO2Value.ToString().Replace(",", ".");

            ViewBag.HCLimit = configurations.Where(x => x.VAR_NAME == "HCLimit").FirstOrDefault().VAR_VALUE.ToString().Replace(",", ".");
            ViewBag.NH3Limit = configurations.Where(x => x.VAR_NAME == "NH3Limit").FirstOrDefault().VAR_VALUE.ToString().Replace(",", ".");
            ViewBag.NO2Limit = configurations.Where(x => x.VAR_NAME == "NO2Limit").FirstOrDefault().VAR_VALUE.ToString().Replace(",", ".");
            ViewBag.COLimit = configurations.Where(x => x.VAR_NAME == "COLimit").FirstOrDefault().VAR_VALUE.ToString().Replace(",", ".");
            ViewBag.NOLimit = configurations.Where(x => x.VAR_NAME == "NOLimit").FirstOrDefault().VAR_VALUE.ToString().Replace(",", ".");
            ViewBag.SFLimit = configurations.Where(x => x.VAR_NAME == "PMLimit").FirstOrDefault().VAR_VALUE.ToString().Replace(",", ".");
            ViewBag.PlumeLimit = configurations.Where(x => x.VAR_NAME == "PlumeLimit").FirstOrDefault().VAR_VALUE.ToString();

            ViewBag.PlumeSize = (v.GasMain.Where(x => x.ValuePosition == 33).FirstOrDefault().CO2Value / 15 * 100).ToString().Replace(",", ".");
            ViewBag.SmokeFactor = v.GasMain.Where(x => x.ValuePosition == 33).FirstOrDefault().SFValue.ToString().Replace(",", ".");
            ViewBag.Speed = v.Speed.ToString();
            ViewBag.Acceleration = v.Acceleration.ToString();
            ViewBag.ValidData = v.ValidData.ToString();
        }
        else
        {
            ViewBag.HC = "";
            ViewBag.NH3 = "";
            ViewBag.NO2 = "";
            ViewBag.CO = "";
            ViewBag.NO = "";
            ViewBag.CO2 = "";

            ViewBag.PlumeSize = "";
            ViewBag.SmokeFactor = "";
            ViewBag.Speed = "";
            ViewBag.Acceleration = "";
            ViewBag.ValidData = "";
        }
        return PartialView("IndexPartial", v);
    }
    0 referencias
```

Figura 25 - Método GetMessages del controlador VDRMainsController.

El resultado de esta llamada se puede visualizar en las siguientes imágenes, de manera que en una se muestra la imagen del vehículo y en la otra una gráfica de dispersión de cada gas, en función de la selección que se haga con los botones Picture (Figura 26) o Chart (Figura 27).

Figura 26 - Datos de emisión del vehículo principal con imagen del vehículo.

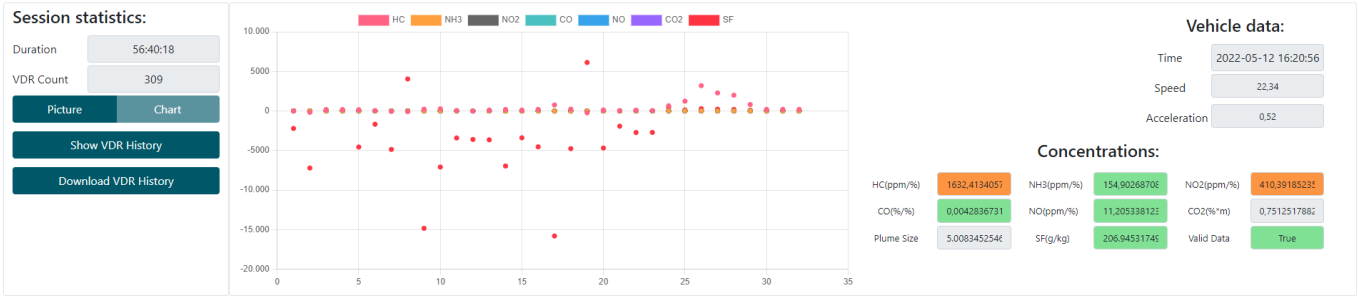


Figura 27 - Datos de emisión del vehículo principal con gráfica.

Las gráficas se han desarrollado utilizando el plugin Chart.js [24], que facilita mucho la creación de estas a partir de datos json.

En la parte inferior podemos ver los datos de los tres vehículos anteriores al que se muestra en la parte principal, ordenados de derecha a izquierda en orden de registro.

Estos datos, se cargan en tres vistas parciales y la metodología que se utiliza para ello es similar a la que se utiliza para cargar el vehículo de la parte principal, y al igual se pueden visualizar con imagen (Figura 28) o con gráfica (Figura 29).

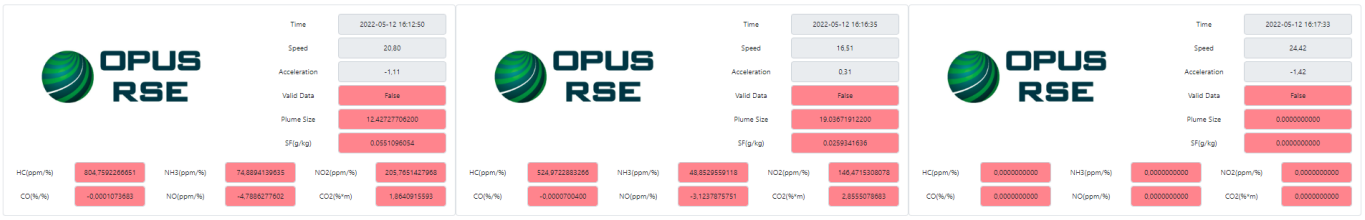


Figura 28 - Datos de emisión de los vehículos anteriores al principal con imagen del vehículo.

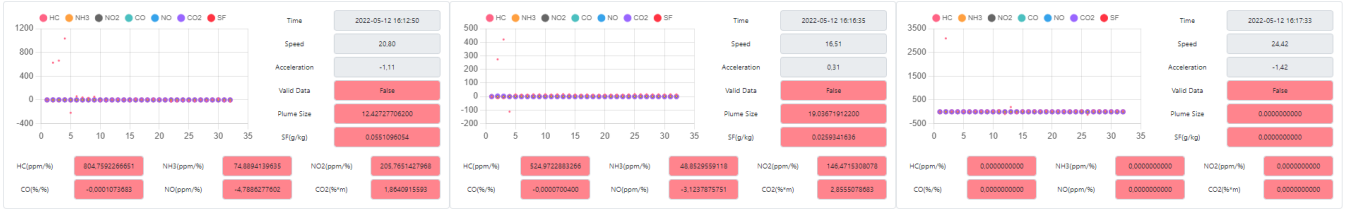


Figura 29 - Datos de emisión de los vehículos anteriores al principal con gráfica.

Otra de las funcionalidades de la pantalla principal es la que permite la visualización de un resumen de todos los vehículos que se han registrado en la sesión de medición.

Para ello se ha implementado el botón “Show VDR History” (Figura 30) como un action link, que, al pulsarse, hace una petición al método Index (Figura 31) del controlador VDRMainsController y nos abre una nueva pestaña que contiene una lista con todos estos registros.

```
<div class="row my-1">
    <div class="col-md-12">
        @Html.ActionLink("Show VDR History", "Index", "VDRMains", new { ViewBag.IdSession }, new { @class = "btn login-btn-color w-100", target = "_blank", id = "showHistory" })
    </div>
</div>
```

Figura 30 - Código html del botón “Show VDR History”.

```
public ActionResult Index(int IdSession)
{
    ViewBag.IdSession = IdSession;
    return View();
}
```

Figura 31 - Método Index del controlador VDRMainsController.

Este método guarda en la variable ViewBag.IdSession el valor del Id de la sesión que está en proceso y devuelve una vista, dentro de la cual se cargarán todos los registros a través de una petición Ajax (Figura 32).

```
$.ajax({
    url: '/VDRMains/GetVDRMains',
    contentType: 'application/json ; charset:utf-8',
    type: 'GET',
    dataType: 'json',
    data: { session: '@ViewBag.IdSession'},
    success: function (result) {
        params.success(result);
    },
    error: function () {
        console.log('error');
    }
});
```

Figura 32 - Código Ajax de carga de la historia de vehículos registrados.

Esta petición consiste en llamar al método GetVDRMains (Figura 33) del controlador VDRMainsController, el cual, nos devuelve una lista con los valores que mostraremos en la vista.

```
public JsonResult GetVDRMains(int session)
{
    ViewBag.IdSession = session;
    var res = db.VDRMain.Where(vdr => vdr.IdSession == session).OrderByDescending(s => s.Timestamp)
        .Select(s => new { s.Id, s.L6VDRCode, Concentrations = s.GasMain.Where(x => x.ValuePosition == 33)
        .Select(x => new { x.HCValue, x.NH3Value, x.NO2Value, x.COValue, x.NOValue, x.CO2Value }).FirstOrDefault(), s.Timestamp, s.ImageName }).ToList();
    return Json(res, JsonRequestBehavior.AllowGet);
}
```

Figura 33 - Método GetVDRMains del controlador VDRMainsController.

La última funcionalidad implementada para esta pantalla principal ha sido el botón que permite descargar un resumen de los datos de la sesión “Download VDR History” (Figura 34).

```
@using (Html.BeginForm("Export", "VDRMains", FormMethod.Post))
{
    <input type="submit" id="btnDownload" class="btn login-btn-color w-100" value="Download VDR History" />
}
```

Figura 34 - Código html del botón "Download VDR History".

Esta funcionalidad se ha implementado de manera que este botón lo que hace es hacer una petición al método Export (Figura 35) del controlador VDRMainsController, el cual, devuelve un fichero con los datos solicitados.

```
public FileResult Export()
{
    Session idSession = db.Session.Where(s => s.StopDatetime == null).FirstOrDefault();
    List<object> customers = (from GasMain in db.GasMain.ToList()
                              where GasMain.VDRMain.IdSession == idSession.Id && GasMain.ValuePosition == 33
                              orderby GasMain.VDRMain.Timestamp
                              select new[] { "",
                                             GasMain.VDRMain.L6VDRCode.ToString(),
                                             GasMain.VDRMain.Timestamp.ToString(),
                                             @"C:\L6Images\" + GasMain.VDRMain.ImageName,
                                             GasMain.VDRMain.Speed.ToString(),
                                             GasMain.VDRMain.Acceleration.ToString(),
                                             GasMain.HCValue.ToString(),
                                             GasMain.NH3Value.ToString(),
                                             GasMain.NO2Value.ToString(),
                                             GasMain.COValue.ToString(),
                                             GasMain.NOValue.ToString(),
                                             GasMain.CO2Value.ToString(),
                                             GasMain.SFValue.ToString(),
                                             (GasMain.CO2Value / 15 * 100).ToString() ,
                    }).ToList<object>();

    //Insert the Column Names.
    customers.Insert(0, new string[14] { "IdVDRMain", "L6VDRCode", "Image", "Timestamp", "Speed", "Acceleration", "HCValue", "NH3Value", "NO2Value", "COValue", "NOValue", "CO2Value", "SFValue", "PlumeSize" });
    StringBuilder sb = new StringBuilder();
    for (int i = 0; i < customers.Count; i++)
    {
        string[] customer = (string[])customers[i];
        for (int j = 0; j < customer.Length; j++)
        {
            //Append data with separator.
            if (i != 0)
            {
                if (j == 0)
                {
                    sb.Append((i).ToString() + ';');
                }
                else
                {
                    sb.Append(customer[j] + ';');
                }
            }
            else
            {
                sb.Append(customer[j] + ';');
            }
        }

        //Append new line character.
        sb.Append("\r\n");

    }

    return File(Encoding.UTF8.GetBytes(sb.ToString()), "text/csv", "SessionId_" + idSession.Id + "_" + idSession.StartDatetime.ToString() + "_" + Guid.NewGuid().ToString() + ".csv");
}
```

Figura 35 - Método Export del controlador VDRMainsController.

7.2.2 Customers

Para implementar esta sección se ha utilizado Scaffolding (generación de código), hace referencia a los elementos de datos dinámicos que generan automáticamente páginas web para cada tabla de una base de datos, aunque posteriormente se han editado para que mantengan el estilo de la aplicación.

Las páginas que se crean automáticamente son las que permiten realizar las acciones de listar todos los registros, creación, visualización, edición y eliminación (CRUD) de cada tabla.

- **Visualización de clientes**

  Pulsando en el apartado Customers de la cabecera, se realiza una petición al método Index del controlador CustomersController, que devuelve la vista con el mismo nombre.

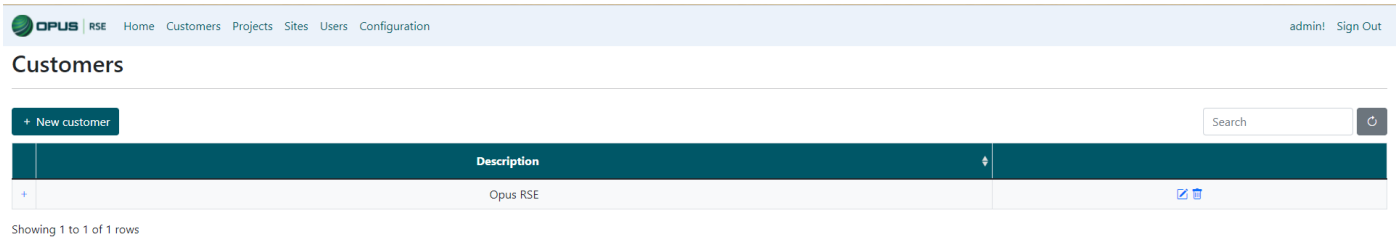


Figura 36 - Visualización de clientes.

La tabla mostrada en la Figura 36 pertenece al plugin Bootstrap-Table [25], el cual implementa tablas con funcionalidades ampliadas y que se integra sin problema con Bootstrap, que es el Framework de diseño utilizado. Este plugin facilita mucho la creación de este tipo de tablas al venir con funciones definidas que simplifican la carga de datos en la tabla, la visualización de detalles de cada una de las filas de la tabla y las operaciones que se pueden realizar en cada una de las filas de la tabla.

El código html de la tabla es el que se puede ver en la Figura 37.

```
<div id="toolbar">
    <a id="new" class="btn login-btn-color" href="/Customers/Create">
        <i class="bi bi-plus"></i> New customer
    </a>
</div>
<table id="table"
       data-toolbar="#toolbar"
       data-search="true"
       data-show-refresh="true"
       data-detail-view="true"
       data-click-to-select="true"
       data-detail-formatter="detailFormatter"
       data-minimum-count-columns="1"
       data-pagination="true"
       data-id-field="Id"
       data-page-list="[10, 25, 50, 100, all]"
       data-ajax="ajaxRequest"
       data-sortable="true"
       data-response-handler="responseHandler">
    <thead class="login-btn-color">
    </thead>
    <tbody class="bg-light">
    </tbody>
</table>
```

Figura 37 - Código html de la tabla generada en la página de visualización de clientes.

En el código podemos ver cómo podemos añadir fácilmente una toolbar en la parte superior, con el botón “New customer” para crear nuevos clientes (data-toolbar), asignarle el formato del detalle a cada fila (data-detail-formater) (Figura 38) e indicar la función que cargara los datos en la tabla (data-ajax) (Figura 39).

```
function detailFormatter(index, row) {
    var html = []
    html.push('<p><b>Projects:</b></p>')
    $.each(row.Projects, function (key, value) {
        html.push('<p>' + value + '</p>')
    })
    return html.join('')
}
```

Figura 38 - Función detailFormatter.

```
function ajaxRequest(params) {
    $.ajax({
        url: '/Customers/GetCustomers',
        contentType: 'application/json ; charset:utf-8',
        type: 'GET',
        dataType: 'json',
        success: function (result) {
            params.success(result);
        },
        error: function () {
            console.log('error');
        }
    });

}
```

Figura 39 - Función ajaxRequest.

Dentro de la función ajaxRequest se hace una petición al método GetCustomers (Figura 40) del controlador CustomersController, que nos devolverá la lista de clientes registrados.

```
public JsonResult GetCustomers()
{
    var res = db.Customer.Select(s => new { s.Id, s.Description, Projects = db.Project.Where(p => p.IdCustomer == s.Id).Select(p => p.Description).ToList() }).ToList();
    return Json(res, JsonRequestBehavior.AllowGet);
}
```

Figura 40 - Método GetCustomers.

Para poder personalizar más la tabla existe la función initTable (Figura 41), que se ejecutara cuando se cargue la página, y en la que se establece el idioma y los campos de las columnas.

```
function initTable() {
    $table.bootstrapTable('destroy').bootstrapTable({
        locale: 'en-US',
        columns: [
            [{
                field: 'Description',
                title: 'Description',
                sortable: true,
                align: 'center'
            }, {
                field: 'operate',
                title: '',
                align: 'center',
                clickToSelect: false,
                events: window.operateEvents,
                formatter: operateFormatter
            }]
        ]
    })
}

$(function () {
    initTable()
    $(".bi-plus").addClass()
})
```

Figura 41 - Función initTable.

Para establecer las operaciones que se pueden realizar con cada fila de la tabla, utilizamos la función operateFormatter (Figura 42), en la cual definimos las operaciones de editar y eliminar un registro.

```
function operateFormatter(value, row, index) {
    return [
        '<a class="edit" href="/Customers/Edit/ ' + row.Id + '" title="Edit">',
        '<i class="bi bi-pencil-square"></i>',
        '</a>',
        '  ',
        , '<a class="remove" href="/Customers/Delete/ ' + row.Id + '" title="Delete">',
        '<i class="bi bi-trash"></i>',
        '</a>'
    ].join('')
}
```

Figura 42 - Función operateFormatter.

- **Creación de un cliente**

  Desde la página de visualización de clientes pulsamos en el botón “New customer”. Este botón realiza una petición al método Create (Figura 43) del controlador CustomerController. Este devuelve la vista que tiene el mismo nombre del método (Figura 44).

```
public ActionResult Create()
{
    return View();
}
```

Figura 43 - Método Create del controlador CustomersController.

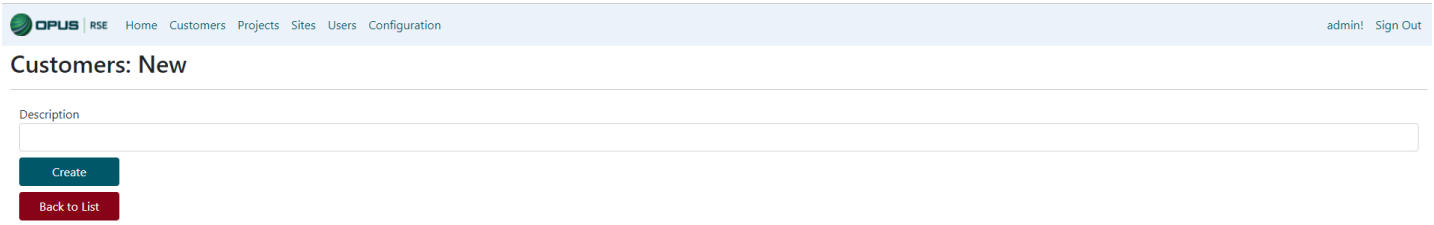


Figura 44 - Visualización nuevo cliente.

En esta vista se ha desarrollado incluyendo sintaxis de Razor [26], como se ve en la Figura 45, que permite insertar código basado en servidor en la página html.

En esta página se carga un formulario con el que posteriormente se creará un nuevo cliente.

```
@model L6WebApp.Models.Customer

@{
    ViewBag.Title = "Customers";
}
<h2>Customers: New</h2>
<hr />
@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()
    <div class="container-fluid">
        <div class="col-md-12">
            <div class="row my-1">
                <div class="col-md-12">
                    <div class="row my-1">
                        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
                        <div class="form-group">
                            @Html.LabelFor(model => model.Description, htmlAttributes: new { @class = "control-label col-md-2" })
                            <div class="col-md-12">
                                @Html.EditorFor(model => model.Description, new { htmlAttributes = new { @class = "form-control" } })
                                @Html.ValidationMessageFor(model => model.Description, "", new { @class = "text-danger" })
                            </div>
                        </div>
                    </div>
                </div>
                <div class="col-md-12">
                    <div class="row my-1">
                        <div class="col-md-1">
                            <input type="submit" value="Create" class="btn login-btn-color w-100" />
                        </div>
                    </div>
                </div>
                <div class="col-md-12">
                    <div class="row my-1">
                        <div class="col-md-1">
                            @Html.ActionLink("Back to List", "Index", null, new { @class = "btn login-btn-color-stop w-100" })
                        </div>
                    </div>
                </div>

            </div>
        </div>
    </div>
}
```

Figura 45 - Código html de la vista Create (Customer).

Este tiene el botón Create, que es el encargado de enviar el formulario al servidor, creando un nuevo cliente y redirigiendo a la pantalla de visualización de clientes (Figura 46).

```
public ActionResult Create([Bind(Include = "Id,Description")] Customer customer)
{
    if (ModelState.IsValid)
    {
        db.Customer.Add(customer);
        db.SaveChanges();
        return RedirectToAction("Index");
    }

    return View(customer);
}
```

Figura 46 - Método Create del controlador CustomersController.

- **Edición de un cliente**
  Desde la página de visualización de clientes se puede acceder a editar un cliente pulsando en el botón Edit de la fila en la que se encuentra el cliente que se desea editar. Este botón realiza una petición al método Edit (Figura 47) del controlador CustomerController. Este devuelve la vista que tiene el mismo nombre del método (Figura 48).

```
public ActionResult Edit(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Customer customer = db.Customer.Find(id);
    if (customer == null)
    {
        return HttpNotFound();
    }
    return View(customer);
}
```

Figura 47 - Método Edit del controlador CustomersController.

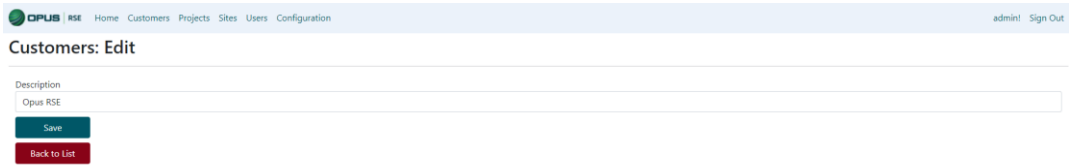


Figura 48 - Visualización editar cliente.

De la misma forma que para la creación de un cliente, en esta página se carga un formulario con el que posteriormente se modificará el cliente (Figura 49).

```
@model L6WebApp.Models.Customer

@{
    ViewBag.Title = "Customers";
}

<h2>Customers: Edit</h2>
<hr />
@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()
    <div class="container-fluid">
        <div class="col-md-12">
            <div class="row my-1">
                <div class="col-md-12">
                    <div class="row my-1">
                        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
                        <div class="form-group">
                            @Html.LabelFor(model => model.Description, htmlAttributes: new { @class = "control-label col-md-2" })
                            <div class="col-md-12">
                                @Html.EditorFor(model => model.Description, new { htmlAttributes = new { @class = "form-control" } })
                                @Html.ValidationMessageFor(model => model.Description, "", new { @class = "text-danger" })
                            </div>
                        </div>
                    </div>
                </div>
                <div class="col-md-12">
                    <div class="row my-1">
                        <div class="col-md-1">
                            <input type="submit" value="Save" class="btn login-btn-color w-100" />
                        </div>
                    </div>
                </div>
                <div class="col-md-12">
                    <div class="row my-1">
                        <div class="col-md-1">
                            @Html.ActionLink("Back to List", "Index", null, new { @class = "btn login-btn-color-stop w-100" })
                        </div>
                    </div>
                </div>

            </div>
        </div>
    </div>
}
```

Figura 49 - Código html de la vista Edit (Customer).

Este tiene el botón Save, que es el encargado de enviar el formulario al servidor, editando los datos del cliente y redirigiendo a la pantalla de visualización de clientes (Figura 50).

```
public ActionResult Edit([Bind(Include = "Id,Description")] Customer customer)
{
    if (ModelState.IsValid)
    {
        db.Entry(customer).State = EntityState.Modified;
        db.SaveChanges();
        return RedirectToAction("Index");
    }
    return View(customer);
}
```

Figura 50 - Método Edit del controlador CustomersController.

- **Eliminación de un cliente**
  Desde la página de visualización de clientes se puede acceder a eliminar un cliente pulsando en el botón Delete de la fila en la que se encuentra el cliente que se desea eliminar. Este botón realiza una petición al método Delete del controlador CustomerController (Figura 51). Este devuelve la vista que tiene el mismo nombre del método (Figura 52).

```
public ActionResult Delete(int? id)
{
    if (id == null)
    {
        return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
    }
    Customer customer = db.Customer.Find(id);
    if (customer == null)
    {
        return HttpNotFound();
    }
    return View(customer);
}
```

Figura 51 - Método Delete del controlador CustomersController.

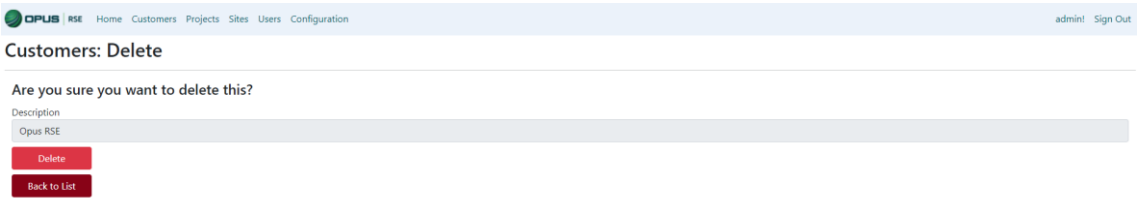
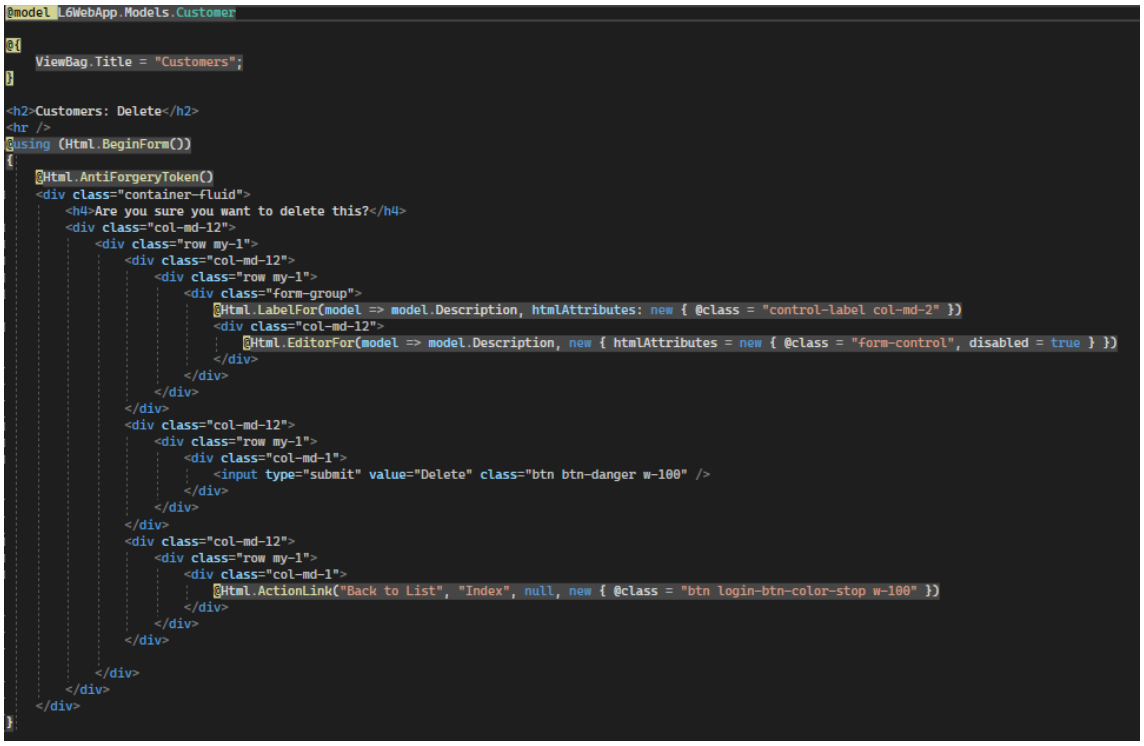


Figura 52 - Visualización eliminar cliente.

De la misma forma que para la creación de un cliente, en esta página se carga un formulario con el que posteriormente se eliminará el cliente (Figura 53).



```
@model L6WebApp.Models.Customer

@{
    ViewBag.Title = "Customers";
}

<h2>Customers: Delete</h2>
<hr />
@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()
    <div class="container-fluid">
        <h4>Are you sure you want to delete this?</h4>
        <div class="col-md-12">
            <div class="row my-1">
                <div class="col-md-12">
                    <div class="row my-1">
                        <div class="form-group">
                            @Html.LabelFor(model => model.Description, htmlAttributes: new { @class = "control-label col-md-2" })
                            <div class="col-md-12">
                                @Html.EditorFor(model => model.Description, new { htmlAttributes = new { @class = "form-control", disabled = true } })
                            </div>
                        </div>
                    </div>
                </div>
                <div class="col-md-12">
                    <div class="row my-1">
                        <div class="col-md-1">
                            <input type="submit" value="Delete" class="btn btn-danger w-100" />
                        </div>
                    </div>
                </div>
                <div class="col-md-12">
                    <div class="row my-1">
                        <div class="col-md-1">
                            @Html.ActionLink("Back to List", "Index", null, new { @class = "btn login-btn-color-stop w-100" })
                        </div>
                    </div>
                </div>

            </div>
        </div>
    </div>
}
```

Figura 53 - Código html de la vista Delete (Customer).

Este tiene el botón Delete, que es el encargado de enviar el formulario al servidor, eliminando el cliente y redirigiendo a la pantalla de visualización de clientes (Figura 54).

```
0 referencias
public ActionResult DeleteConfirmed(int id)
{
    Customer customer = db.Customer.Find(id);
    List<Project> projects = db.Project.Where(p => p.IdCustomer == id).ToList();

    foreach (var p in projects) {
        db.Project.Remove(p);
    }

    db.Customer.Remove(customer);
    db.SaveChanges();
    return RedirectToAction("Index");
}
```

Figura 54 - Método DeleteConfirmed del controlador CustomersController.

7.2.3 Projects, Sites, Users, Configuration

El resto de secciones de la aplicación comparten la estructura de la sección Customer.

Todas ellas tienen una vista de visualización de registros, de creación de un nuevo, edición de un registro y de eliminación de un registro existente.

Para cada una de ellas se han modificado los métodos para adaptarlos a sus modelos correspondientes.

7.3 Servicio de Windows (L6FileSystemWatcher)

Esta parte del sistema se trata de un Servicio de Windows, creado con la plantilla que ofrece Visual Studio para crear Servicios de Windows de .Net Framework con lenguaje de programación C#.

El algoritmo desarrollado se divide en dos procesos, un proceso que genera una LUT (Tablas de consulta) para cada uno de los gases medidos en base a la información meteorológica (P,T) registrada mediante un sensor externo (que se utilizará como referencia para el cálculo de la concentración de cada uno de los gases) y un segundo proceso, que, tomando como argumento de entrada las LUT calculadas en el primer proceso y los datos en crudo que salen del módulo HW de control e interfaz realiza el cálculo de las concentraciones correspondientes a cada uno de los gases a lo largo del 0.5 s que dura la medida. Éste último módulo realiza 32 cálculos de concentración para cada uno de los 6 gases que se analizan en menos de un segundo.

Como parte de la propiedad intelectual de la empresa, no se puede mostrar el código del algoritmo, pero en la Figura 55 se muestra el funcionamiento del servicio:

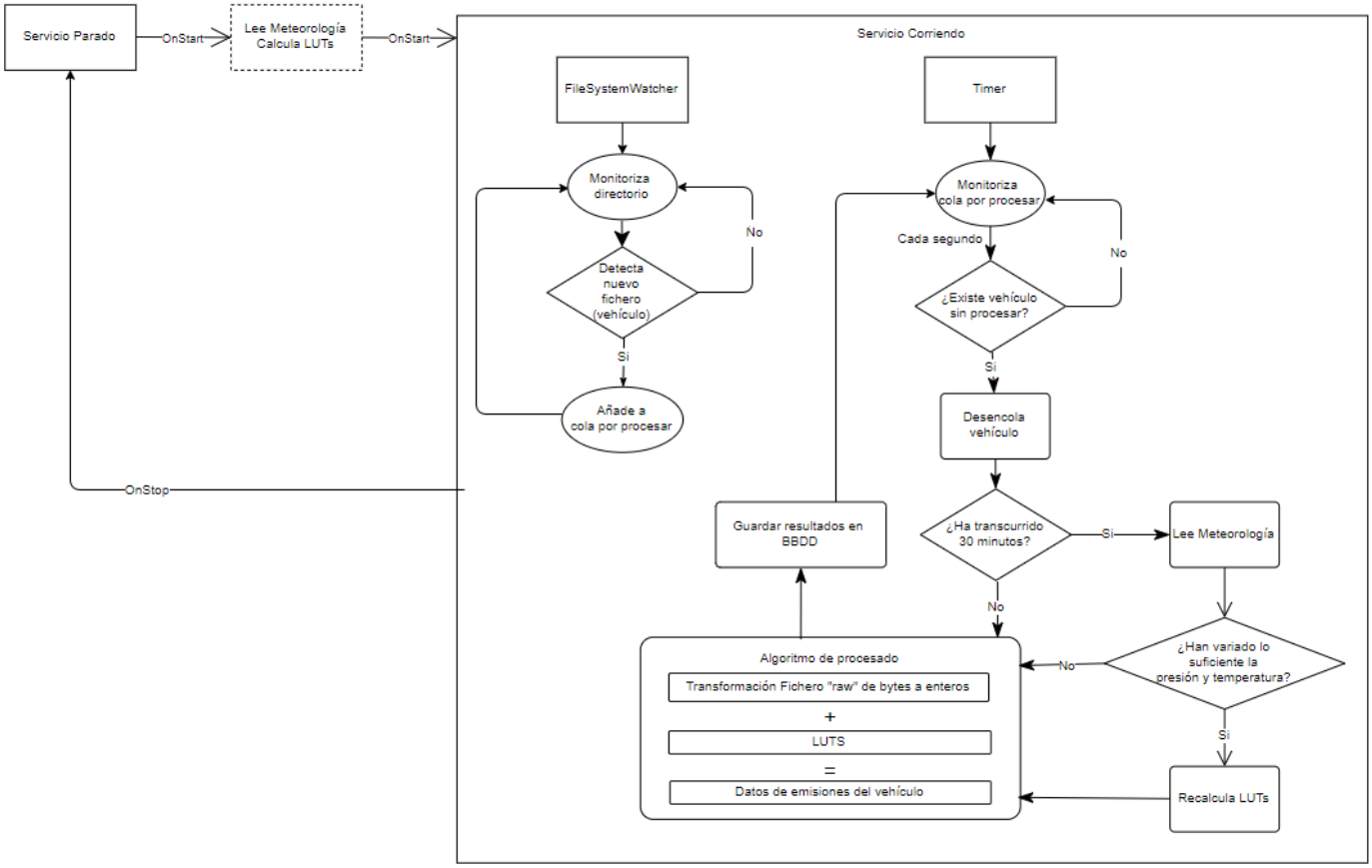


Figura 55 - Esquema del servicio Windows para el procesado de ficheros.

Capítulo 8. Evaluación.

En este capítulo se analizarán los resultados obtenidos de una evaluación de la usabilidad de la web, y además se explicará cómo se ha verificado la bondad del algoritmo para el cálculo de emisiones.

8.1 Evaluación de la usabilidad de la web

8.1.1 Metodología

Para realizar la evaluación, se ha utilizado la herramienta de Google Forms[27]. La evaluación ha sido realizada por los tres compañeros que realizan el trabajo de campo en las campañas de medición. Estos usuarios estaban muy familiarizados con el mundo de la tecnología.

El tipo de preguntas realizadas han sido de respuesta sí/no o preguntas subjetivas acerca de las diferentes características principales de la aplicación, cambios que realizarían o funciones que desearían tener en el futuro.

Las preguntas propuestas se pueden consultar en el ANEXO I. Preguntas de la evaluación de la web.

Para realizar la evaluación, los usuarios estuvieron trabajando con la herramienta durante un mes y se les facilitó un enlace al cuestionario.

8.1.2 Resultados

Los resultados obtenidos en el formulario fueron muy favorables, ya que ninguno de los usuarios que realizó la evaluación, ha encontrado alguna dificultad para realizar las labores para la cual está desarrollada la aplicación.

Los resultados de algunas de las preguntas se muestran en las Figura 56, Figura 57 y Figura 58.

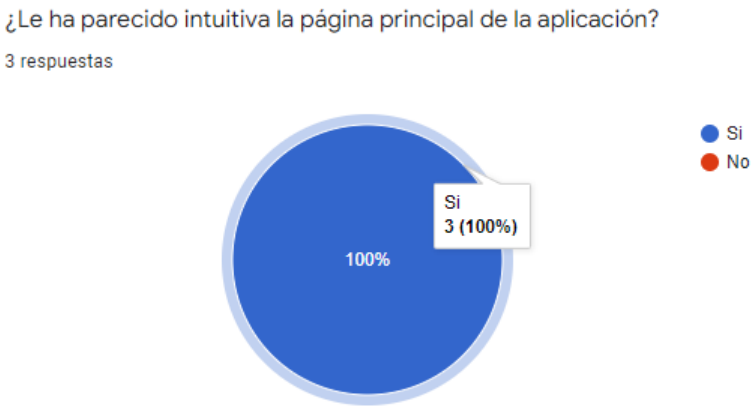


Figura 56 - Respuestas evaluación web 1.

Figura 57 - Respuestas evaluación web 2.

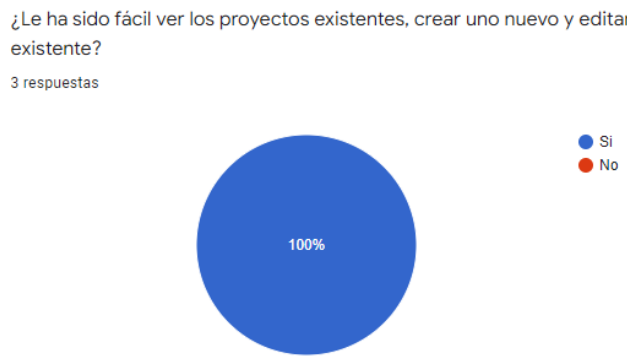


Figura 58 - Respuestas evaluación web 3.

Al final del formulario, se les pedía a los usuarios, que indicasen que cambios realizarían ellos en la aplicación (Figura 59).

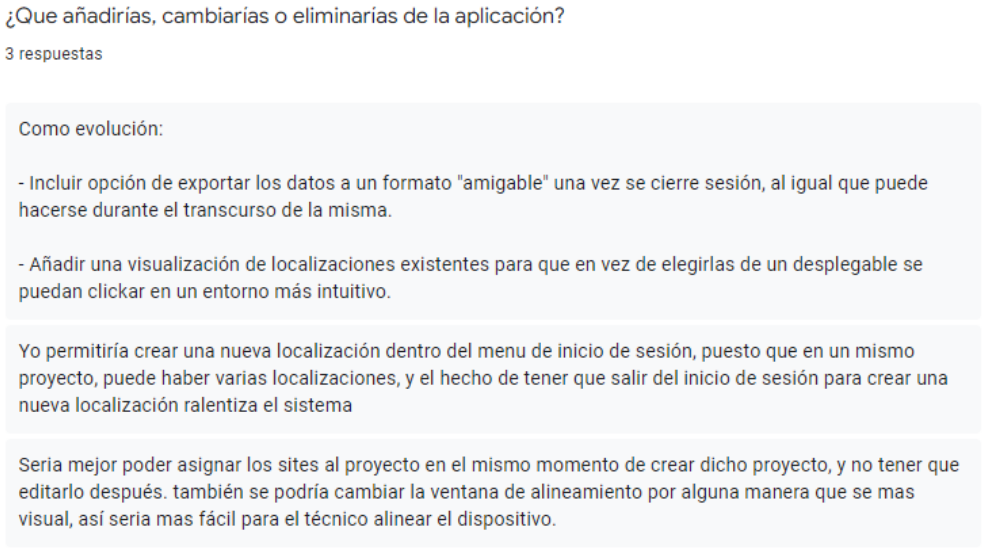


Figura 59 - Respuestas evaluación web 4.

Estos cambios se tendrán en cuenta para el trabajo futuro.

8.2 Evaluación del algoritmo de procesado de datos

8.2.1 Metodología

Las pruebas se centran en la evaluación de la capacidad del sistema para determinar las concentraciones relativas de los distintos contaminantes presentes en el tubo de escape de un vehículo de carretera. En este caso se realizaron en el laboratorio, siendo un entorno interior controlado.

Con el fin de validar el algoritmo matemático que calcula la concentración de gas, se difundirán diferentes mezclas calibradas de gases secos en la trayectoria óptica del sistema utilizando el método "Puff".

El método “Puff” consiste en generar diferentes densidades dinámicas de columna utilizando una válvula de gas controlada electrónicamente para difundir mezclas de gas desde cilindros calibrados (Figura 60) en la trayectoria óptica del sistema.

Esto se repite con 3 cilindros de gas con diferentes concentraciones de contaminantes, los cilindros 2,7 y 10 se utilizaron en esta prueba.

Botellas de gas calibradas utilizadas durante los test.

	C_3H_8 (ppm)	NH_3 (ppm)	NO_2 (ppm)	CO (ppm)	CO_2 (ppm)	NO (ppm)
Cyl_1	651,9 ± 0,6%	0	0	4818 ± 0,5%	144600 ± 0,5 %	0
Cyl_2	0	0	0	0	150000± 0,5%	7199,2 ± 0,7%
Cyl_3	9110 ± 0,5%	0	0	20310 ± 0,5%	1311000 ± 0,5 %	0
Cyl_4	0	0	0	0	150300 ± 0,5 %	1696,6 ± 0,7 %
Cyl_5	4186,6 ± 0,5%	0	0	38920 ± 0,5%	120600 ± 0,5 %	0
Cyl_6	0	0	0	0	149900 ± 0,5 %	2197,8 ± 0,7 %
Cyl_7	11190 ± 0,5%	0	0	49370 ± 0,5%	111800 ± 0,5%	0
Cyl_8	0	0	0	0	150000 ± 0,5%	429,5 ± 0,8%
Cyl_9	0	0	498,1 ± 0,8%	0	149800 ± 0,5%	0
Cyl_10	0	0	999,7 ± 0,9%	0	150000 ± 0,5%	0
Cyl_11	5000 ± 0,5%	1000 ± 0,5%	0	0	0	0

Figura 60 - Botellas de gas calibradas utilizadas durante los test.

8.2.2 Resultados

La prueba de “Puff” muestra (Figura 61) una diferencia relativa máxima del 13,20%, lo que subestima el valor real en la medición de la proporción de hidrocarburos. Aunque la diferencia relativa en todos los contaminantes está por debajo del 15%, estos resultados indican que todavía hay margen de mejora. La relación de NH3 no pudo medirse porque no hay CO2 en el cilindro 11.

	RELATIVE DIFFERENCE BETWEEN EXPECTED AND MEASURED VALUE (%)				
	Type of release	NO/CO_2	NO_2/CO_2	CO/CO_2	HC/CO_2
Cyl_2	Short	7.00%	-	-	-
	Medium	2.80%	-	-	-
	Long	-4.70%	-	-	-
Cyl_7	Short	-	-	0.00%	-7.80%
	Medium	-	-	4.55%	-13.20%
	Long	-	-	-2.27%	-7.80%
Cyl_10	Short	-	-7.31%	-	-
	Medium	-	9.20%	-	-
	Long	-	-4.68%	-	-

Figura 61 - Diferencia relativa entre la proporción medida y la esperada.

Capítulo 9. Conclusiones y trabajo futuro.

9.1 Conclusiones

Se ha desarrollado un servicio Windows que permite procesar los datos obtenidos por un sensor externo, utilizando un algoritmo propio que permite alcanzar los objetivos de velocidad de procesado establecidos al principio del proyecto. Además, el servicio permite que el sistema sea autoadaptable a cambios en las condiciones meteorológicas, integrando en el procesado los datos obtenidos de sensores externos al sistema.

Se ha desarrollado una aplicación web, que cumple con las necesidades de la compañía, permitiendo tanto el almacenamiento local de los datos como su subida a la nube. La aplicación cumple las funciones de interfaz con el usuario.

La usabilidad de la aplicación cumple con las expectativas del usuario final, tal como muestran los resultados de la encuesta de satisfacción reportada en esta memoria.

Se puede descargar el código fuente de la aplicación web en el enlace https://drive.google.com/file/d/17ULjO1XBSISYcVH8JRhIV1owoxJj7zxV/view?usp=sharing

9.2 Trabajo futuro

La funcionalidad del proyecto se mejorará de la siguiente manera:

- **Añadir comunicación con el módulo HW del sensor:** esto permitirá realizar la configuración del sensor sin necesidad de aplicaciones externas.
- **Añadir comunicación con periféricos:** esto permitirá realizar la configuración de cada uno de los periféricos, como son la barra de velocidad, la cámara o la estación meteorológica.
- **Envío de datos a BBDD en la nube**: actualmente los datos se quedan en una base de datos local dentro del servidor físico, pero la idea es implementar el envío de estos a la nube en tiempo real, de manera que cuando exista más de un sensor (ahora solo existe el prototipo) puedan sincronizarse los datos de todos los sensores centralizados en una única base de datos.
- **Integración con clientes:** cuando se comercialice el sensor, es probable que aparezcan clientes que quieran tener los datos también en sus sistemas, con lo que a futuro también se plantea el desarrollo de integraciones con los clientes.

Chapter 9. Conclusions and future work.

9.1 Conclusions

A Windows service has been developed to process the data obtained from an external sensor, using a proprietary algorithm that allows achieving the processing speed objectives established at the beginning of the project. In addition, the service allows the system to be self-adaptable to changes in weather conditions, integrating in the processing the data obtained from external sensors to the system.

A web application has been developed, which meets the needs of the company, allowing both local storage of data and its upload to the cloud. The application fulfills the functions of user interface.

The usability of the application meets the expectations of the end user, as shown by the results of the satisfaction survey reported in this report.

You can download the source code of the web application at the link https://drive.google.com/file/d/17ULjO1XBSISYcVH8JRhIV1owoxJj7zxV/view?usp=sharing

9.2 Future work

The functionality of the project will be improved as follows:

- **Adding communication with the sensor HW module**: this will allow to perform the sensor configuration without the need of external applications.
- **Adding communication with peripherals:** this will allow the configuration of each of the peripherals, such as the speed bar, the camera or the weather station.
- **Sending data to a cloud database:** currently the data remains in a local database within the physical server, but the idea is to implement the sending of data to the cloud in real time, so that when there is more than one sensor (now there is only the prototype) the data from all sensors can be synchronized and centralized in a single database.
- **Integration with customers:** when the sensor is marketed, it is likely that customers will appear who also want to have the data in their systems, so that in the future the development of integrations with customers will also be considered.

Bibliografía

[1]. Favre, C. M. (2003). Emissions control technologies to meet current and future European vehicle emissions legislation. Association for Emissions Control by Catalyst (AECC).

[2]. Kozina, A. R. (2022). Emission Analysis of Diesel Vehicles in Circumstances of Emission Regulation System Failure: A Case Study. Journal of Energy Resources Technology, 144(8).

[3]. Grange, S. K. (2020). Post-dieselgate: evidence of NOx emission reductions using on-road remote sensing. 7(6). Environmental science & technology letters, 382-387.

[4]. Stedman, D. H. (1991). Evaluation of a remote sensor for mobile source CO emissions. Denver Univ., CO (USA). Dept. of Chemistry., (No. PB-91-148320/XAB).

[5]. Opus RSE Next Gen. https://www.opusrse.com/technology/data-analytics-iot/. Recuperado el día 11 de Mayo de 2022.

[6]. Visual Studio. https://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado. Recuperado el día 11 de Mayo de 2022.

[7]. ASP.NET MVC Framework. https://es.wikipedia.org/wiki/ASP.NET_MVC_Framework . Recuperado el día 11 de Mayo de 2022.

[8]. Microsoft SQL Server https://es.wikipedia.org/wiki/Microsoft_SQL_Server. Recuperado el día 11 de Mayo de 2022.

[9]. SQL Server Management Studio. https://en.wikipedia.org/wiki/SQL_Server_Management_Studio .Recuperado el día 11 de Mayo de 2022

[10]. Transact-SQL. https://es.wikipedia.org/wiki/Transact-SQL . Recuperado el día 11 de Mayo de 2022.

[11]. Azure Devops Server. https://azure.microsoft.com/es-es/services/devops/server/ . Recuperado el día 11 de Mayo de 2022.

[12]. Internet Information Services. https://es.wikipedia.org/wiki/Internet_Information_Services . Recuperado el día 11 de Mayo de 2022.

[13]. GitMind. https://es.wikipedia.org/wiki/GitMind . Recuperado el día 11 de Mayo de 2022.

[14]. C# https://es.wikipedia.org/wiki/C_Sharp .Recuperado . Recuperado el día 15 de Mayo de 2022.

[15]. SignalR. https://docs.microsoft.com/es-es/aspnet/signalr/overview/getting-started/introduction-to-signalr . Recuperado el día 15 de Mayo de 2022.

[16]. JavaScript. https://es.wikipedia.org/wiki/JavaScript . Recuperado el día 15 de Mayo de 2022.

[17]. JQuery. https://es.wikipedia.org/wiki/JQuery . Recuperado el día 15 de Mayo de 2022.

[18]. AJAX. https://es.wikipedia.org/wiki/AJAX . Recuperado el día 15 de Mayo de 2022.

[19]. Bootstrap. https://es.wikipedia.org/wiki/Bootstrap_(framework) . Recuperado el día 15 de Mayo de 2022.

[20]. Html. https://es.wikipedia.org/wiki/HTML . Recuperado el día 15 de Mayo de 2022.

[21]. CSS. https://es.wikipedia.org/wiki/CSS . Recuperado el día 15 de Mayo de 2022.

[22]. Linq. https://es.wikipedia.org/wiki/Language_Integrated_Query . Recuperado el día 15 de Mayo de 2022.

[23]. Entity Framework. https://es.wikipedia.org/wiki/ADO.NET_Entity_Framework . Recuperado el día 15 de Mayo de 2022.

[24]. Chart.js. https://www.chartjs.org/ .Recuperado el 16 de Mayo de 2022.
[25]. Bootstrap-Table. https://bootstrap-table.com/ .Recuperado el 16 de Mayo de 2022.
[26]. Sintaxis Razor. https://docs.microsoft.com/es-es/aspnet/web-pages/overview/getting-started/introducing-razor-syntax-c .Recuperado el 16 de Mayo de 2022.
[27]. Google Forms. https://es.wikipedia.org/wiki/Formularios_de_Google . Recuperado el 16 de Mayo de 2022.

Anexo I. Guía del usuario.

Una vez se haya desplegado la aplicación y se haya creado la base de datos en el servidor, se podrá acceder a la aplicación web desde un navegador, introduciendo la url configurada para ello.

El servicio Windows estará configurado para que se inicie al encenderse el servidor.

Una vez hemos accedido a la aplicación en el navegador, podremos hacer Login siempre que se disponga de una cuenta. En caso contrario, el administrador deberá suministrarnos un usuario y una contraseña.

Introducimos el nombre de usuario y contraseña, pulsamos en “Sign In” (Figura 62).

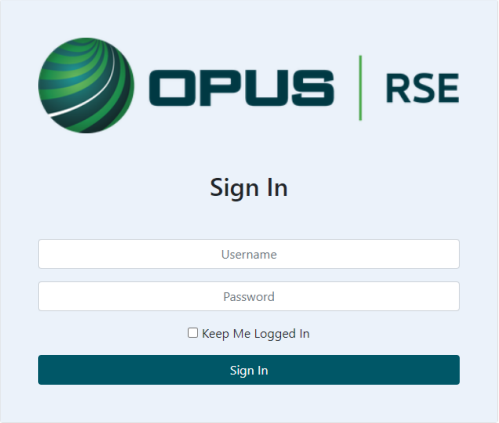


Figura 62 - Login de la aplicación web.

La aplicación nos redirige a la página principal de manera que podremos visualizar todas las secciones si tenemos el rol de administrador (Figura 63) o únicamente la sección de la pantalla principal (Figura 64).

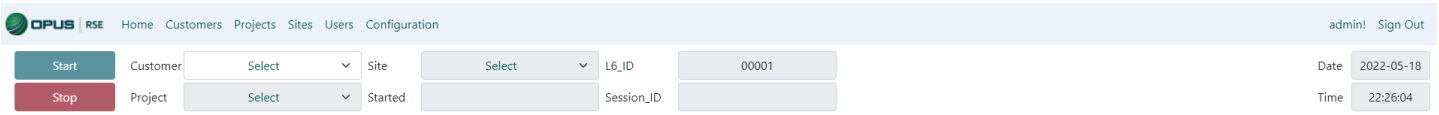


Figura 63 – Pantalla principal con rol administrador.

Figura 64 – Pantalla principal con rol operador.

Para iniciar una sesión de medición se seleccionará un cliente, un proyecto y una localización, y de esta manera se nos habilitará el botón Start (Figura 65) para poder iniciar la sesión.

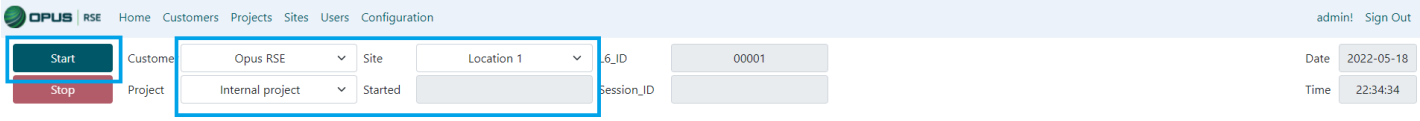


Figura 65 – Cliente, proyecto y localización seleccionados.

Pulsamos en el botón Start, y veremos como la página muestra los datos de la sesión. Inicialmente se mostrará sin datos de vehículos (Figura 66), pero a medida que se registren se irán mostrando en el orden correspondiente, por defecto, mostrándose la foto del vehículo (Figura 67).

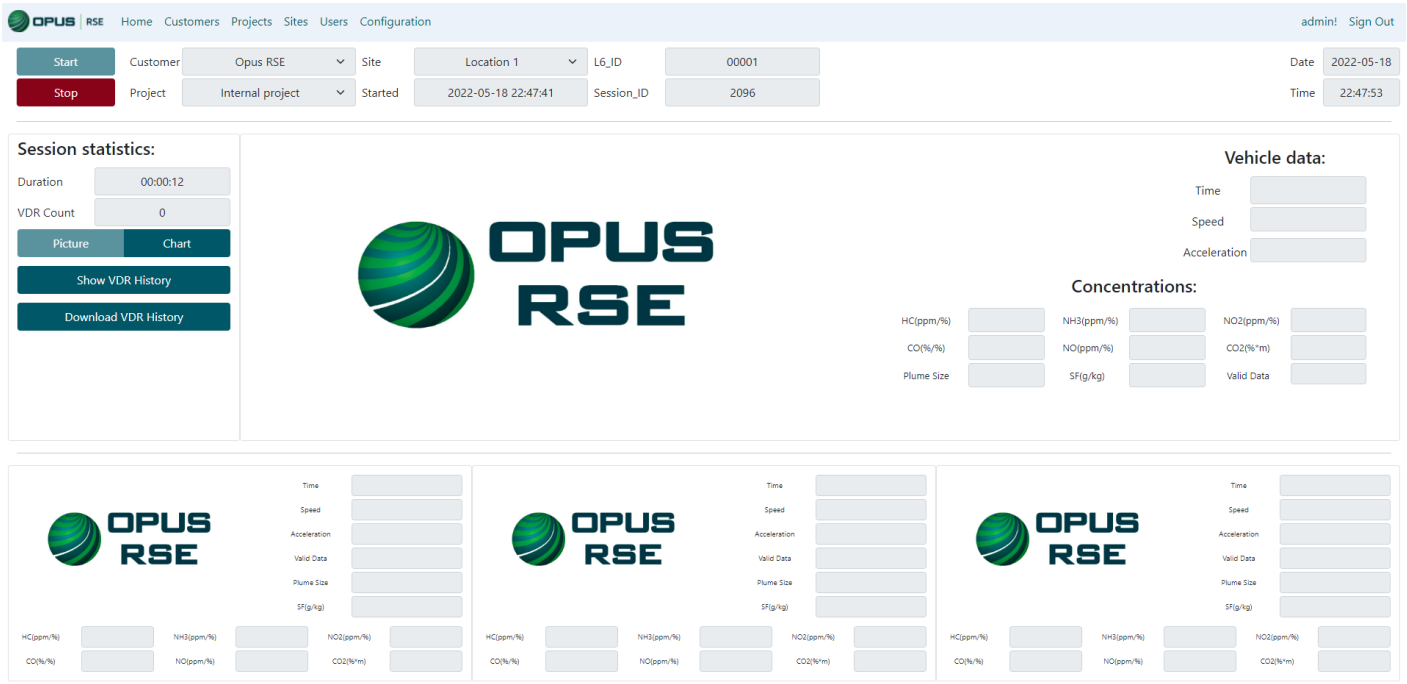


Figura 66 – Pantalla principal sesión iniciada vacía.

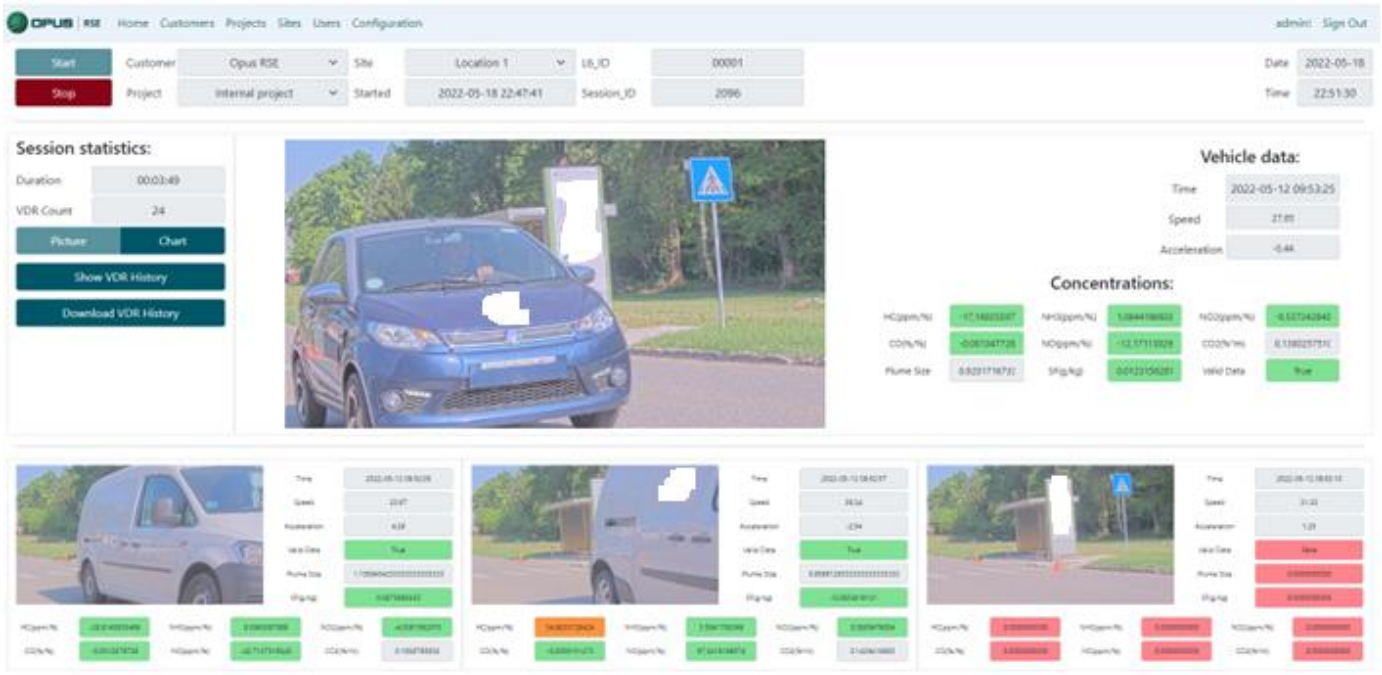


Figura 67 – Pantalla principal sesión iniciada datos vehículos con foto.

El pulsar sobre el botón Chart, podremos visualizar la gráfica de dispersión de los gases emitidos por el vehículo, pudiendo quitar o añadir gases a la gráfica pulsando sobre el color de cada gas (Figura 68).

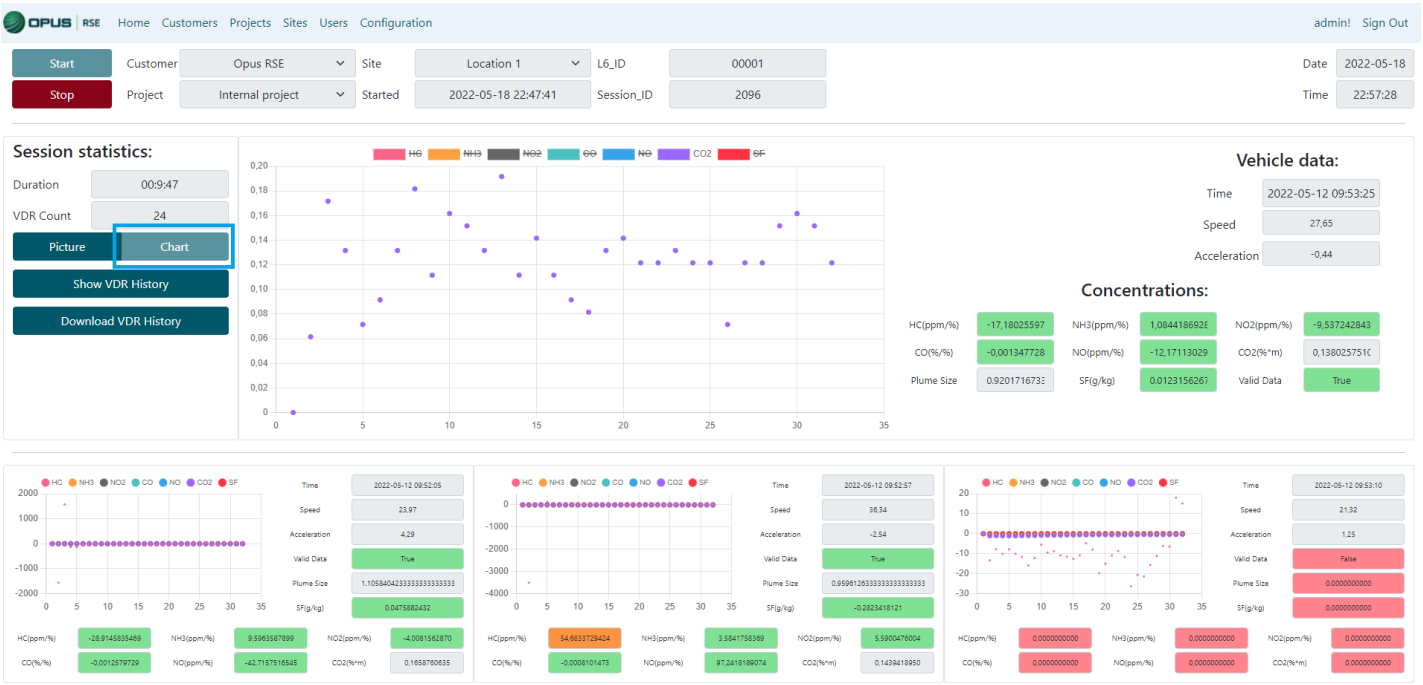


Figura 68 – Pantalla principal sesión iniciada datos vehículos con gráfica y gas seleccionado.

Podemos ver una tabla resumen de los vehículos registrados durante la sesión pulsando en el botón Show VDR History (Figura 69).

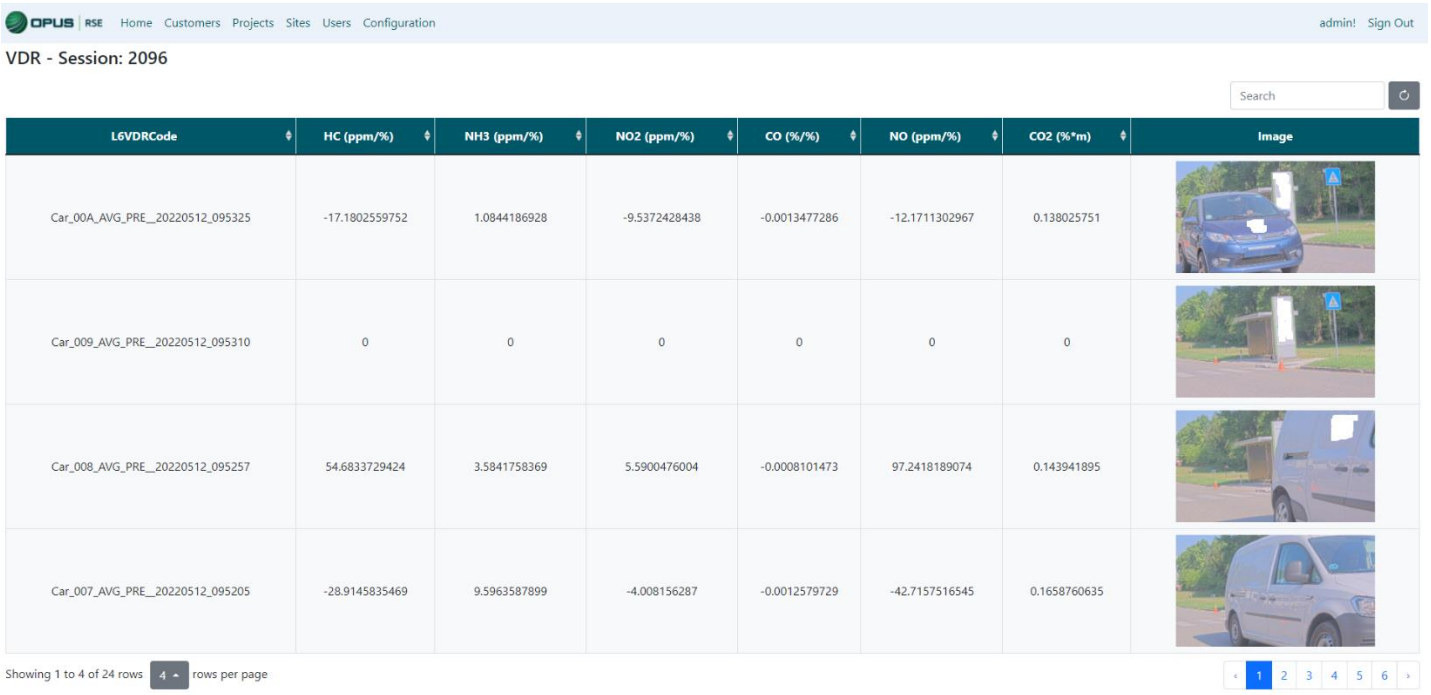


Figura 69 – Resumen vehículos registrados en la sesión.

También podemos descargarnos esta información pulsando sobre el botón Download VDR History.

Para terminar una sesión pulsaremos en el botón Stop de la pantalla principal (Figura 70).

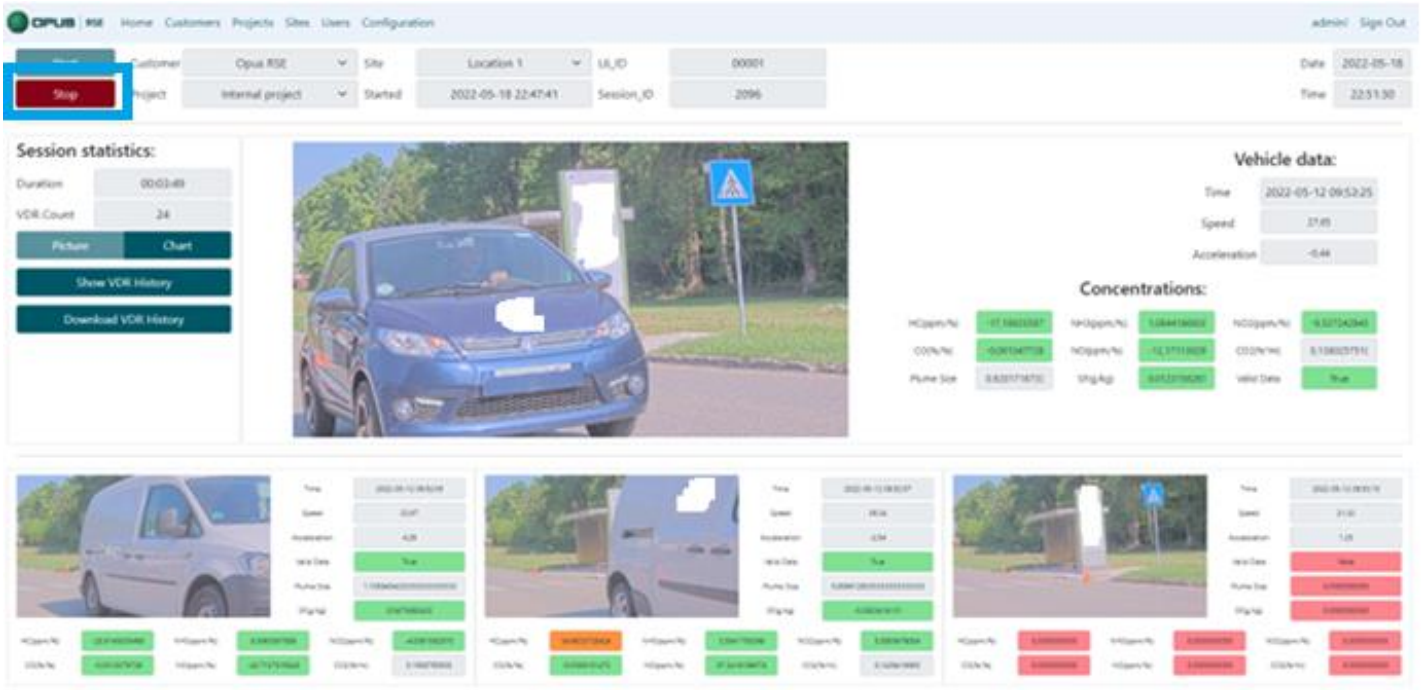


Figura 70 – Terminar sesión.

Para ver los usuarios registrados en el sistema pulsaremos sobre Users, en la barra de la parte superior. Desde esta ventana podremos tanto ver los usuarios existentes, como añadir, editar o eliminar un usuario (Figura 71).

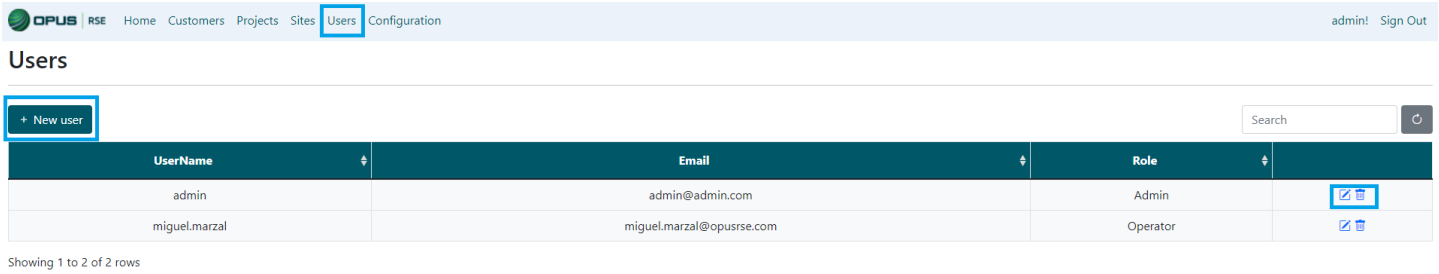


Figura 71 – Sección usuarios.

Para añadir un usuario pulsaremos sobre New user de la Figura 71. Esto nos redirigirá a otra página en la que se nos solicitarán los campos necesarios para dar de alta al usuario, que habrá que rellenar.

Después de rellenar los campos pulsaremos sobre Create, se creará el usuario, y se nos redirigirá a la ventana de visualización de usuarios (Figura 72).

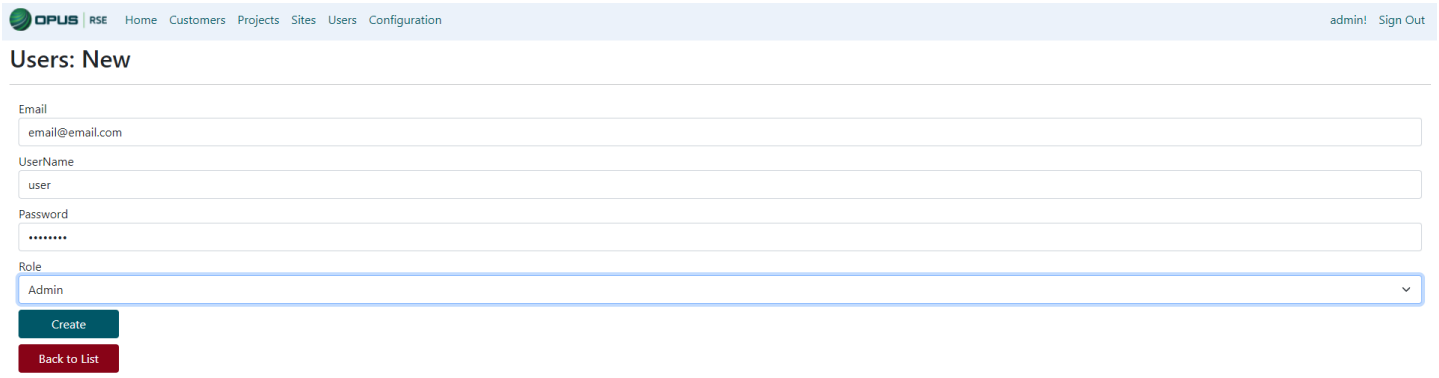


Figura 72 – Crear usuario.

Para editar los datos de un usuario existente, pulsaremos sobre el botón Edit de la fila de la tabla de la Figura 71 en la que se encuentra el usuario que se desea modificar.

El sistema nos redirige a la página de edición del usuario, donde editaremos los campos deseados y pulsaremos sobre Save, actualizando los datos de dicho usuario y redirigiéndonos a la ventana de visualización de usuarios.

Además, en esta ventana se podrá añadir o eliminar clientes de un usuario, que solo será necesario cuando el usuario tenga rol operador (Figura 73).

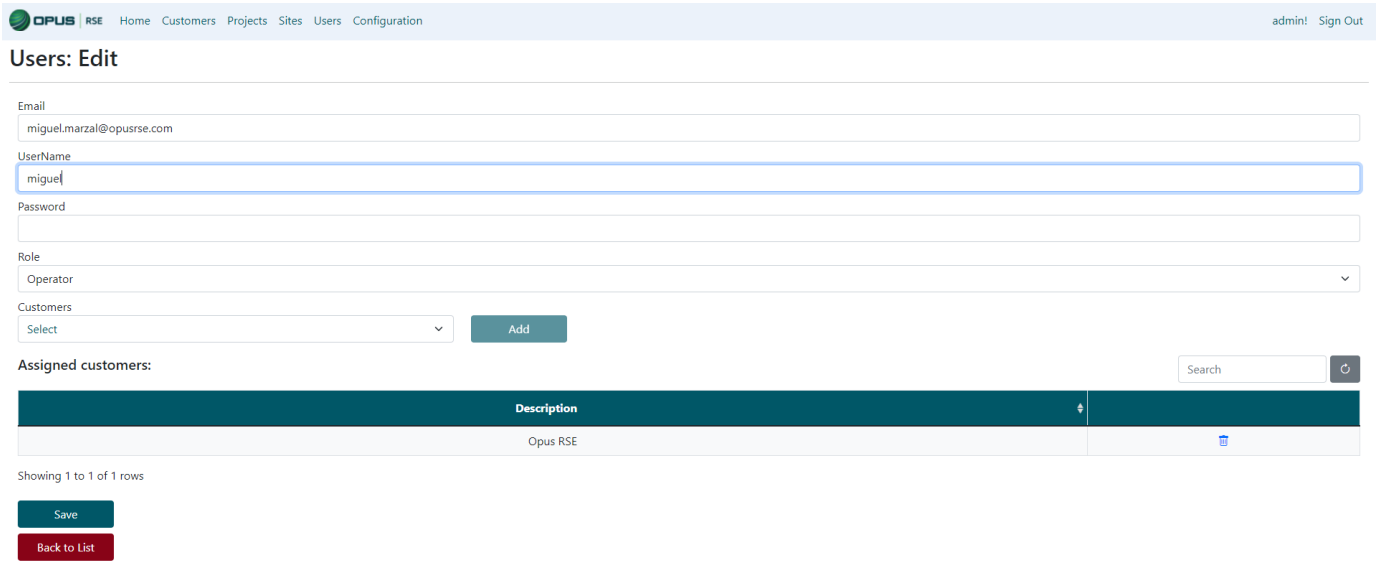


Figura 73 – Editar usuario.

Para eliminar un usuario pulsaremos sobre el botón Delete de la fila de la tabla de la Figura 71 en la que se encuentra el usuario que se desea eliminar. Seguidamente el sistema nos pedirá confirmar que queremos eliminar el usuario. Pulsaremos en Delete, y el sistema eliminará el usuario y nos redirigirá a la ventana de visualización de usuarios (Figura 74).

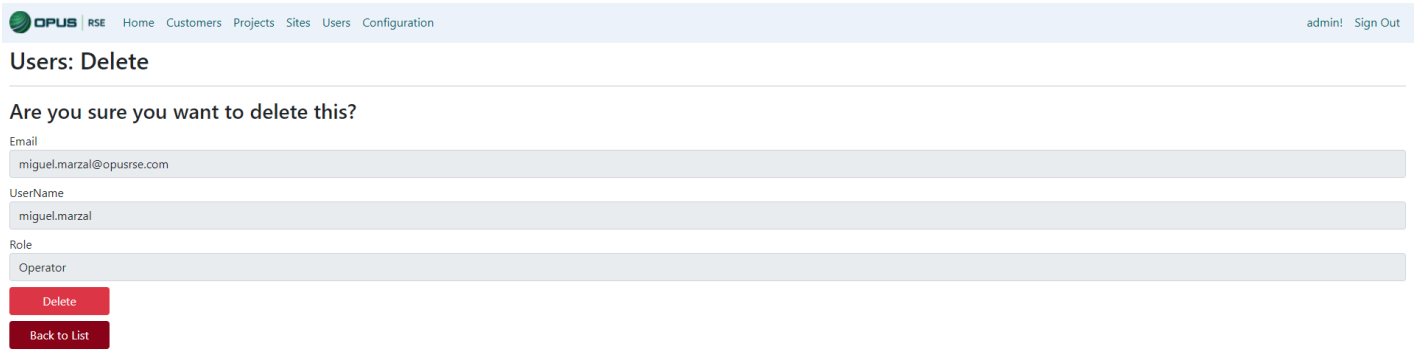


Figura 74 – Eliminar usuario.

Las secciones Clientes, Proyectos, Localizaciones y Configuración, siguen la misma estructura de páginas que la sección de Usuarios, explicada anteriormente, de manera que, para visualizar, añadir, editar y eliminar registros de dichas secciones, se realizar de forma prácticamente igual, a diferencia de los campos a rellenar, que será específico de cada una de ellas.

Anexo II. Preguntas de la evaluación de la web.

A continuación, se muestran las preguntas propuestas para realizar la evaluación de usabilidad de la aplicación web.

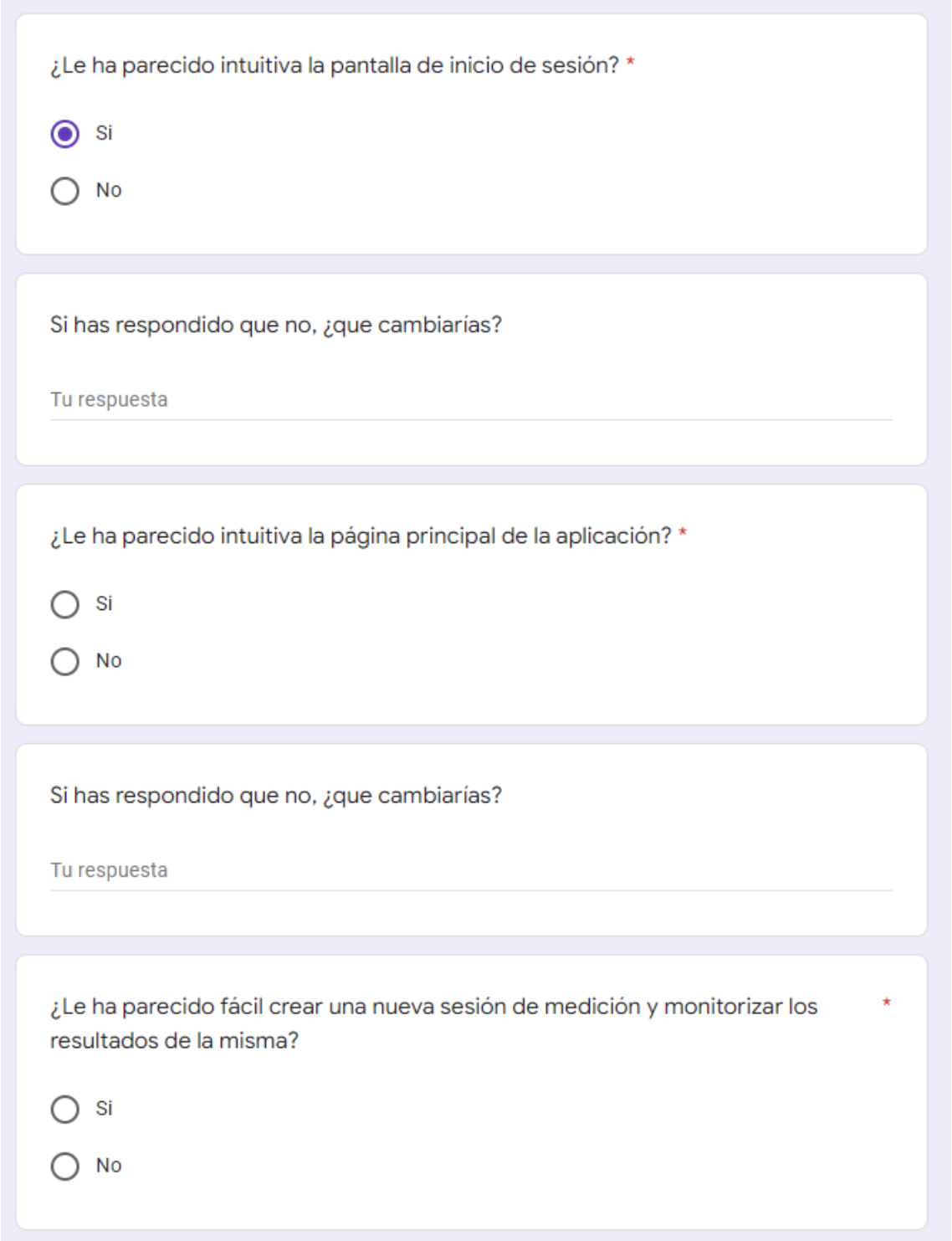


Figura 75 - Preguntas evaluación web 1.

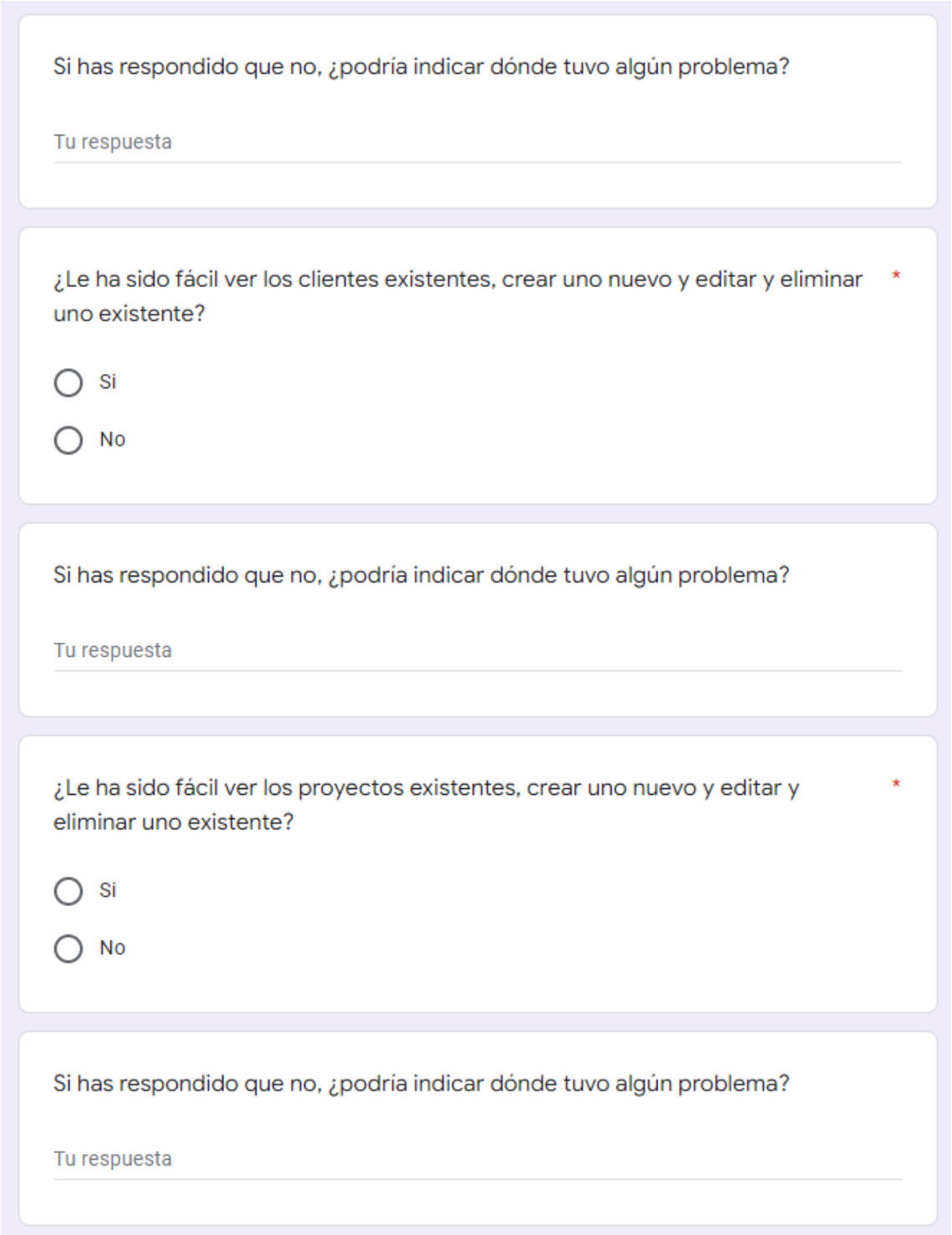


Figura 76 - Preguntas evaluación web 2.

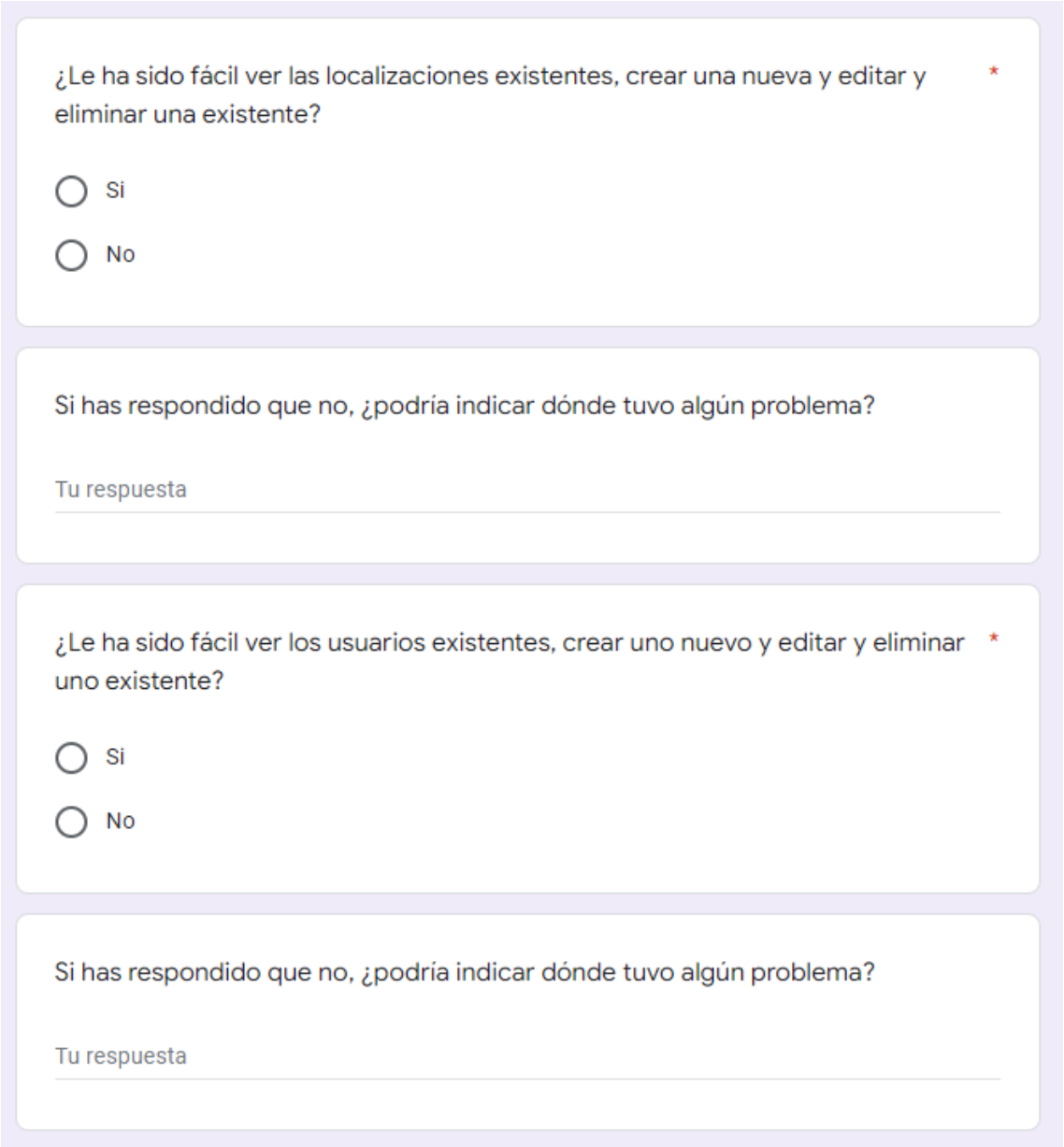


Figura 77 - Preguntas evaluación web 3.

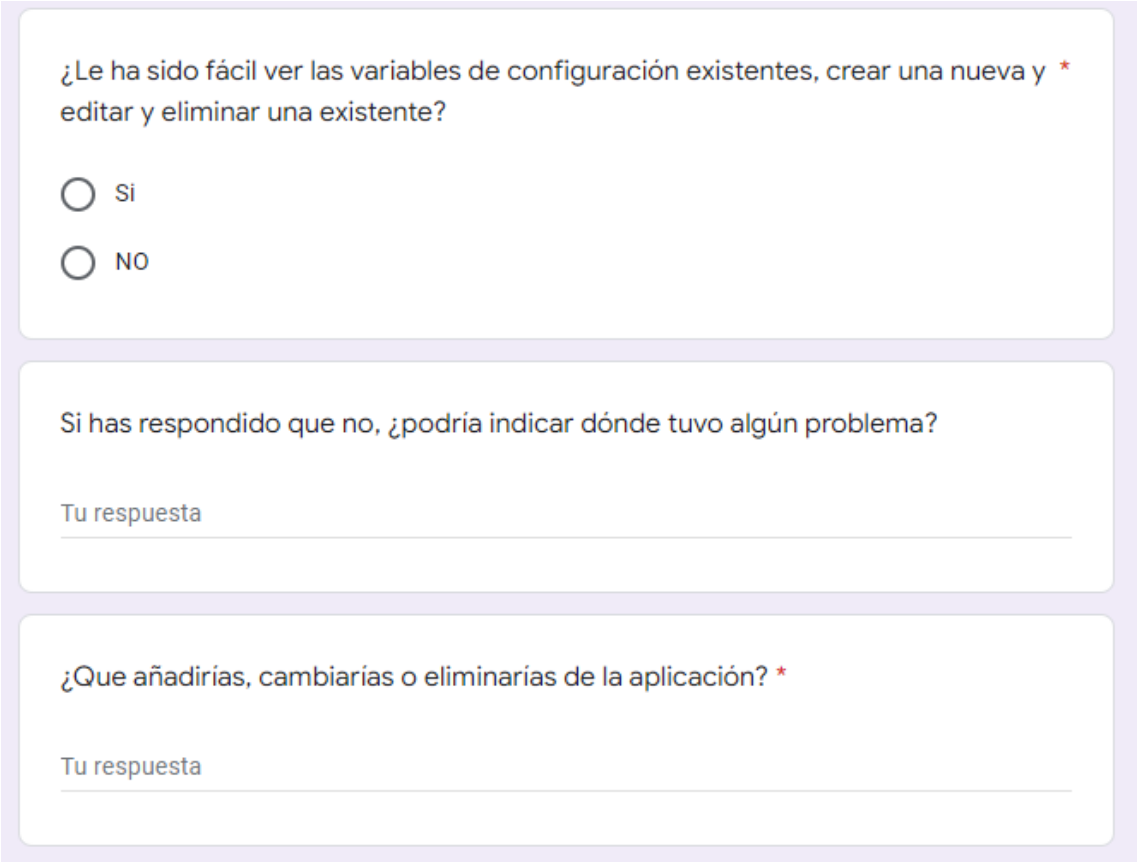


Figura 78 - Preguntas evaluación web 4.