UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

TFG del Grado en Ingeniería Informática

Mejora de interfaz para planta piloto y aplicación web (v 2.0)

Presentado por Iván Cortés Aliende
en Universidad de Burgos — 18 de junio de 2022
Tutores: Dr. Alejandro Merino Gómez - Dr. Daniel Sarabia Ortiz

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

Dr. Daniel Sarabia Ortiz, profesor del departamento de Ingeniería Electromecánica, área de Ingeniería de Sistemas y Automática, y Dr. Alejandro Merino Gómez, profesor del departamento de Ingeniería Electromecánica, área de Ingeniería de Sistemas y Automática.

Exponen:

Que el alumno D. Iván Cortés Aliende, con DNI 71706785F, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado "Mejora de interfaz para planta piloto y aplicación web (v 2.0)".

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 18 de junio de 2022

V°. B°. del Tutor:

D. Daniel Sarabia Ortiz

V°. B°. del co-tutor:

D. Alejandro Merino Gómez

Resumen

El objetivo de este proyecto es la mejora de la interfaz desarrollada en 2019 en el TFG "Desarrollo de una interfaz para planta piloto" por Francisco Crespo Diez, la cual permite al usuario manejar una placa Freescale FRDMK64F que, a su vez, controla una planta piloto de laboratorio que los alumnos utilizan en diferentes titulaciones de la Universidad de Burgos.

Las mejoras se centran en el arreglo de errores detectados, la implementación de nuevas características solicitadas por los usuarios y la implementación de una versión de dicha interfaz via web.

Descriptores

Aplicación web, solución de errores, .NET, ASP, C#, interfaz, base de datos

Abstract

The goal of this project is to improve the interface developed in 2019 in the TFG "Desarrollo de una interfaz para planta piloto" by Francisco Crespo Diez, which allows the user to manage a Freescale FRDMK64F board that controls a pilot laboratory plant that students use in differents degrees at the University of Burgos.

The improvements focus on the fixing detected errors, the implementation of new features requested by the useres and the implementation of a web based version.

Keywords

Web application, error fixing, .NET, ASP, C#, interface, data base.

Índice general

Índice general III
Índice de figuras VI
Introducción 1
1.1. Planta piloto ... 1
1.2. Estructura de la memoria ... 4
1.3. Recursos adjuntos ... 5
Objetivos del proyecto 7
2.1. Requisitos técnicos a tener en cuenta ... 7
Conceptos teóricos 10
3.1. .NET Framework ... 10
3.2. ASP.NET ... 12
3.3. C# ... 13
3.4. IIS Express ... 14
3.5. Base de datos ... 14
3.6. Comunicación en Serie ... 16
Técnicas y herramientas 19
4.1. GitHub ... 19
4.2. GitKraken ... 20

4.3. Microsoft Visual Studio 2019 21
4.4. Microsoft Visual Studio Code 21
4.5. Microsoft Word 21
4.6. Microsoft SharePoint 22
4.7. Microsoft Teams 23
4.8. IIS Express 23
4.9. Open Broadcaster Software 24
4.10. Lightshot 24
4.11. Mockplus RP 25
4.12. P&E Driver 25
4.13. Katalon Recorder 26
4.14. HTML Help Workshop 26
4.15. Modelo Vista Controlador (MVC) 27
4.16. SonarQube 27
4.17. Kanban 28
Aspectos relevantes del desarrollo del proyecto 31
5.1. Primeros pasos del proyecto 31
5.2. Comprensión del código y formación 32
5.3. Cambio de la base de datos 33
5.4. Detección de comunicación en serie 34
5.5. Desarrollo de la aplicación web 35
5.6. Sistema de registros 39
5.7. Unión de ambas aplicaciones 41
5.8. Refactorizaciones y pruebas 43
Trabajos relacionados 46
6.1. Primera versión de la planta piloto 46
6.2. Segunda versión de la planta piloto 46
6.3. Primera versión de la interfaz para planta piloto 47
Conclusiones y líneas de trabajo futuras 49
7.1. Conclusiones 49

7.2. Líneas de trabajo futuras 50

Bibliografía 53

Índice de figuras

Figura 1.1: Componentes de la planta. Imagen facilitada por los tutores. 2
Figura 1.2: Partes de la planta piloto. 3
Figura 1.3: Diagrama con las conexiones del sistema. 4
Figura 3.4: Comunicación en serie. 16
Figura 3.5: Comunicación en serie con un gran número de bits. 17
Figura 4.6: Repositorio del proyecto en GitHub. 20
Figura 4.7: Interfaz de GitKraken con el repositorio GitHub del proyecto abierto. 20
Figura 4.8: Microsoft Visual Studio con el proyecto abierto. 21
Figura 4.9: Carpeta del proyecto en el SharePoint de la UBU. 22
Figura 4.10: Pantalla de lanzamiento del servidor web mediante la integración de IIS Express. 23
Figura 4.11: El editor de Mockplus RP con el proyecto de la interfaz principal de la aplicación web. 25
Figura 4.12: Menú de ayuda compilado en formato CHM. 26
Figura 4.13: Panel de SonarQube con las distintas métricas. 28
Figura 4.14: Tablero Trello del proyecto. 29
Figura 5.15: Menú de selección del puerto en serie. 34
Figura 5.16: Mensaje mostrado al no encontrar ningún puerto en uso. 35
Figura 5.17: Mensaje mostrado al usuario al encontrar un puerto en uso y seleccionarlo en el menú correspondiente. 35

Figura 5.18: Página principal de la aplicación web. 37
Figura 5.19: Página con la información de la base de datos 38
Figura 5.20: Página con la información relativa a la autoría de la aplicación web. 39
Figura 5.21: Directorio "Logs" con registros de anteriores sesiones. 41
Figura 5.22: Ejemplo de un fichero de registro de una sesión previa. 41
Figura 5.23: Formulario de lanzamiento del servidor web. 42
Figura 5.24: Servidor web lanzado desde la aplicación original. 43
Figura 5.25: Gráfico de actividad de la aplicación local en SonarQube. 44
Figura 5.26: Gráfico de actividad de la aplicación web en SonarQube. 44

Introducción

En 2019 el alumno de la Universidad de Burgos Francisco Crespo Diez desarrolló como Trabajo Fin de Grado una interfaz que permitía a los alumnos comunicarse con las placas NXP FRDM-K64F de una forma mucho más cómoda e intuitiva, permitiendo que estos se pudieran centran en aprender en vez de gastar tiempo con métodos de comunicación mucho más complicados [1].

Dicha interfaz está siendo utilizada hoy en día por alumnos de distintas titulaciones para poder comunicarse con las placas, cambiando las variables que estas manejan y estudiando cómo estos cambios afectan al caudal de aire que llega al sensor de la planta piloto. Su uso durante estos años ha permitido tanto a alumnos como a profesores detectar posibles mejoras que la interfaz podría implementar y fallos que se podrían arreglar, todo con el fin de mejorar la experiencia de usuario.

La principal mejora de todas las implementadas ha sido el desarrollo de una aplicación web la cual permite a los usuarios comunicarse con la placa con una interfaz similar a la local que ya se encontraba presente en el proyecto original, pero con todas las ventajas que supone desplegarlo en un entorno web.

1.1. Planta piloto

Este proyecto parte de mejorar la interfaz que permite interactuar con las plantas piloto situadas en el laboratorio de Ingeniería de Sistemas y Automática

y empleadas en asignaturas de control de los grados de Ingeniería Electrónica Industrial y Automática y el máster de Ingeniería Industrial. Estas plantas piloto fueron diseñadas en un principio por Rubén Zambrana Rodríguez en su Trabajo Fin de Grado [2] y mejoradas en una segunda versión por María Isabel Revilla Izquierdo, también en su Trabajo Fin de Grado [3].

La planta piloto se basa en un sistema multivariable que permite controlar el caudal de aire, mediante el uso de un ventilador, y la temperatura, mediante una resistencia capaz de calentarse. Ambos componentes pueden ser ajustados mediante la asignación de valores a sus variables. También se encuentran en la planta piloto un caudalímetro de aire como sensor de aire y un sensor de temperatura. Todos estos componentes se pueden apreciar en la Figura 1.1. Todo este sistema es controlado por un microcontrolador en una placa FRDM-K64F.

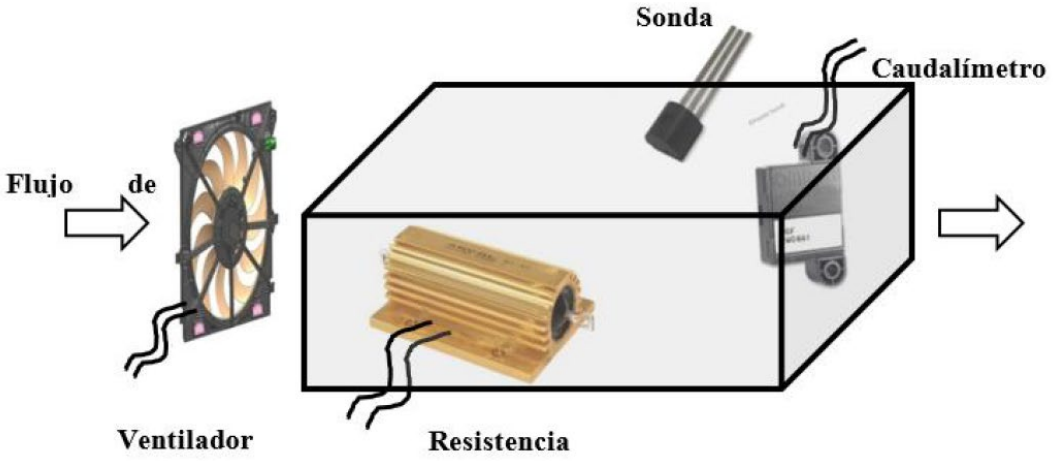

Figura 1.1: Componentes de la planta. Imagen facilitada por los tutores.

La planta piloto (Figura 1.2), está conformada por los siguientes componentes:

- Una planta, formada a su vez por un ventilador, una resistencia, una sonda de temperatura y un caudalímetro.
- Una fuente de alimentación, encargada de alimentar a todo el sistema.
- Un conjunto de etapas de acondicionamiento de señales.
- Una placa FRDM-K64F que implementa el sistema de control y adquisición de variables del proceso y se encarga de servir de intermediario entre la interfaz y la planta mediante intercambio de

mensajes.

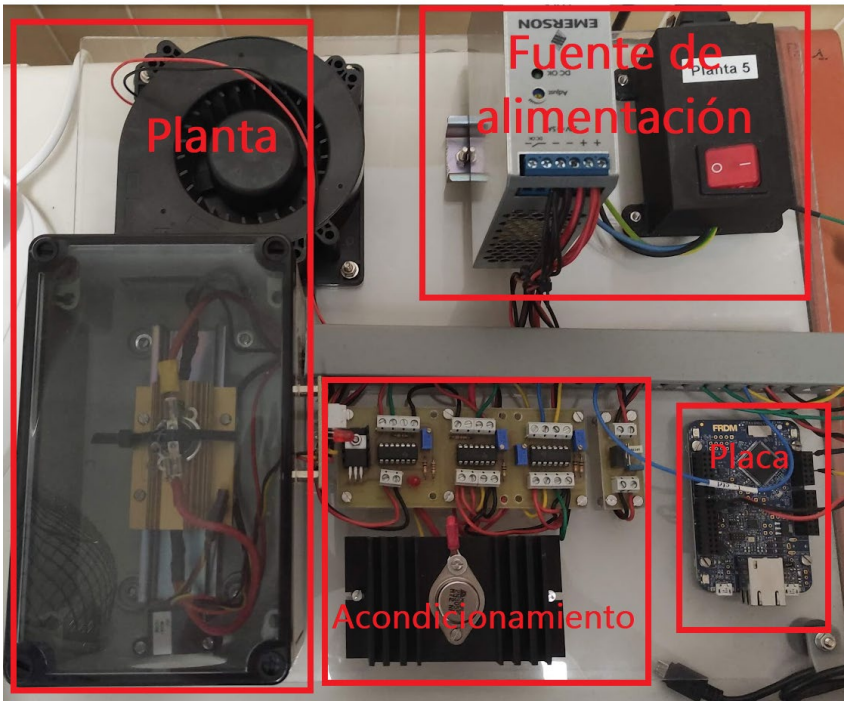

Figura 1.2: Partes de la planta piloto.

Lo más interesante de la planta piloto en cuanto a este proyecto se refiere es la placa FRDM-K64F, ya que es la que interactúa con la aplicación. La placa se conecta por USB a un equipo con el sistema operativo Windows y con la aplicación ejecutándose. Una vez se ha establecido la conexión, el programa es el encargado de ir recogiendo periódicamente los datos que envía la placa mediante el puerto USB y la conexión en serie para almacenarlos en una tabla de una base de datos. Dicha base de datos será empleada por la aplicación web desarrollada en este proyecto para comunicarse con la interfaz local. Además, desde la interfaz también se le pueden mandar mensajes a la placa para que cambie los valores de sus variables internas, de forma que podamos no solo monitorizar el proceso sino actuar sobre él. En la figura 1.3 se pueden apreciar las comunicaciones que se acaban de explicar.

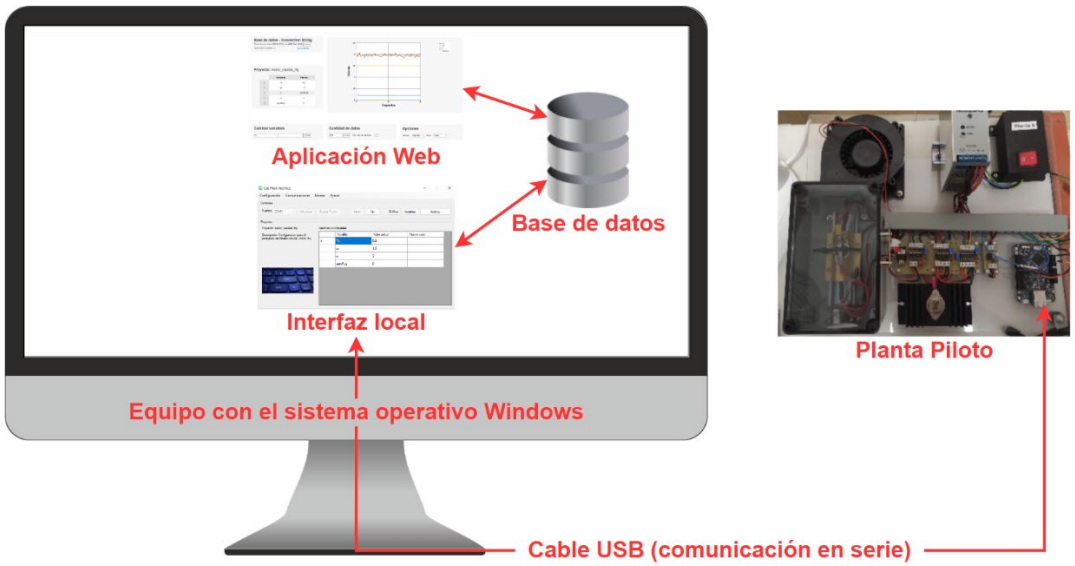

Figura 1.3: Diagrama con las conexiones del sistema.

1.2. Estructura de la memoria

La memoria se organiza de la siguiente manera:

- **Introducción.** Descripción y contexto del proyecto realizado, estructura de los documentos presentados y qué recursos han sido adjuntados.

- **Objetivos del proyecto.** Objetivos que se buscaban cumplir con el desarrollo de este proyecto

- **Conceptos teóricos**. Marco y contexto teórico del proyecto.

- **Técnicas y herramientas**. Utilidades y procedimientos que se han seguido para la realización del proyecto.

- **Aspectos relevantes del desarrollo del proyecto**. Cómo ha sido llevado a cabo el proyecto.

- **Trabajos relacionados**. Proyectos similares al presentado y el proyecto de partida.

- **Conclusiones y Líneas de trabajo futuras.** Deducciones a las que se han llegado tras la realización del proyecto y posibles futuras mejoras y/o funcionalidades a implementar.

- **Bibliografía.** Procedencia de los materiales utilizados en este documento.

1.3. Recursos adjuntos

Objetivos del proyecto

En este apartado se detallarán, como ya hemos indicado, los objetivos que se planeaban cumplir con el desarrollo de este proyecto:

1. Depuración de errores de código.
2. Implementación de nuevas funcionalidades sugeridas por usuarios de la aplicación.
3. Desarrollo de una nueva interfaz en forma de aplicación web mediante la cual los usuarios también puedan comunicarse con la placa.
4. Cambio del sistema gestor de bases de datos que utilizaba la aplicación original basado en Microsoft SQL Server 2017 Express por uno más ligero que se crea y destruye cada vez que se inicia la aplicación, dado que los datos que se almacenan en la base de datos no son demasiados ni presentan una estructura compleja.
5. Añadir a la sección de ayuda de la aplicación y al manual de usuario las nuevas funcionalidades implementadas.

2.1. Requisitos técnicos a tener en cuenta

Hay una serie de requisitos técnicos que se han debido tener en cuenta a la

hora de cumplir los objetivos ya definidos:

1. Se debe partir de la aplicación inicial ya desarrollada reutilizando en mayor medida las clases y métodos ya implementados.
2. Se deben mantener las compatibilidades con máquinas Windows y con el método de conexión en serie con la placa.
3. Se debe de utilizar un sistema de control de versiones.
4. Se debe mantener .NET Framework como base de la aplicación.
5. Las interfaces deben mantenerse lo más intuitivas posible, manteniendo así el objetivo principal del proyecto original, facilitar a los alumnos su uso y aprendizaje.

Conceptos teóricos

3.1. .NET Framework

Es un entorno de ejecución de aplicaciones administrado para el sistema operativo Windows que proporciona servicios a las aplicaciones que se están ejecutando [4]. Se basa en dos componentes principales:

- **Common Language Runtime (CLR).** Es el motor de ejecución de las aplicaciones. Se encarga de controlar las aplicaciones que se encuentran en ejecución en el sistema.
- **Biblioteca de clases.** Se encarga de dotar a los desarrolladores de una biblioteca compuesta por código reusable y altamente probado para que estos la puedan utilizar a la hora de desarrollar sus proyectos en .NET Framework.

.NET Framework ofrece los siguientes servicios a las aplicaciones en ejecución:

- **Administración de la memoria.** CLR se encarga de asignar y liberar la memoria, así como de gestionar la vida útil de los objetos instanciados en la aplicación, de manera que los desarrolladores no tengan que ocuparse de ello.
- **Sistema de tipos comunes.** .NET Framework se encarga de definir los tipos básicos. Esto nos permite que todos los lenguajes destinados a ser

utilizados junto con .NET Framework tengan estos tipos comunes, haciéndolos interoperables.

- **Biblioteca de clases extensa.** Los desarrolladores pueden utilizar la biblioteca de clases de .NET Framework para ahorrarse el tener que programar grandes cantidades de código de bajo nivel.
- **Marcos y tecnologías de desarrollo.** .NET Framework dispone de una serie de bibliotecas especializadas en el desarrollo de determinados tipos de aplicaciones. Por ejemplo, ASP.NET para el desarrollo de aplicaciones web.
- **Interoperabilidad de lenguajes.** Continuando con lo explicado en el apartado de Sistemas de tipos comunes, los compiladores de los lenguajes de programación destinados a .NET Framework producen el denominado Lenguaje Intermedio Común (CIL). Este lenguaje es compilado en pleno tiempo de ejecución por el CLR, de manera que las rutinas programadas en un determinado lenguaje son accesibles para el resto de los lenguajes.
- **Compatibilidad de versiones.** Las aplicaciones desarrolladas para una determinada versión de .NET Framework serán compatibles para versiones posteriores. Esto se cumple en la mayoría de las situaciones, pero se pueden dar extrañas excepciones.
- **Ejecución en paralelo.** .NET Framework permite que coexistan varias versiones de CLR en el mismo equipo, de manera que también puedan coexistir varias versiones de las mismas aplicaciones, permitiendo que cada una se ejecute en la versión de .NET Framework en la que se compiló.
- **Compatibilidad con múltiples versiones.** Si los desarrolladores de bibliotecas establecían como destino .NET Stardard (que es la especificación formal de las API de .NET [5]) sus bibliotecas podían ser utilizadas en varias plataformas de .NET Framework.

Dichos servicios son proporcionados por .NET Framework con la intención de cumplir los siguientes objetivos [6].

- Dotar al desarrollador de un entorno de programación orientada a objetos (POO) coherente en el cual el código fuente de los objetos se pueda:
    - Almacenar y ejecutarse de forma remota.

  - Almacenar de forma distribuida en Internet, pero ejecutarse de forma local
  - Almacenar y ejecutar localmente.
- Dotar al usuario de un entorno de ejecución de código el cual:
  - No cuente con los problemas de rendimiento presentes en otros entornos, como los entornos con scripts o interpretados.
  - Cuente con los menores conflictos de software y de control de versiones posibles.
  - Promueva la ejecución segura de código
- Proporcionar al desarrollador de una experiencia muy similar entre todos los tipos de aplicaciones que pueden ser desarrollados con .NET Framework.
- Basar todos los protocolos de comunicaciones en estándares ya definidos, de tal forma que sea sencillo integrar el código desarrollado con .NET Framework con otro distinto.

.NET Framework fue el entorno de ejecución para el que Francisco Crespo Diez desarrolló la primera versión de la aplicación en su día. Por lo tanto, ha sido el empleado también durante el desarrollo de este proyecto de mejora e implementación de nuevas funcionalidades.

3.2. ASP.NET

Es un marco web pensado para la creación de páginas y aplicaciones web mediante la utilización de los lenguajes HTML, CSS y JavaScript. También es utilizado para la utilización de tecnologías de tiempo real y la creación de APIs web.

ASP.NET nos ofrece tres marcos distintos para la creación de aplicaciones web. Todos ellos cuentan con la totalidad de las ventajas y las características de ASP.NET:

- **Web Forms (ASP.NET Web Forms)**: Permite la creación de aplicaciones web dinámicas mediante un modelo controlado por eventos.

Está pensado para la creación de aplicaciones web centradas en la utilización de datos y en el manejo mediante una interfaz de usuario.

- **MVC (ASP.NET MVC)**: Como su propio nombre indica, se basa en el patrón de arquitectura Modelo-Vista-Controlador. Permite compilar espacios web dinámicos basándose en modelos, de forma que se consigue una separación muy marcada de intereses y proporciona un control absoluto del marcado.
- **Web Pages (ASP.NET Web Pages)**: Permite la combinación del código backend del servidor con el lenguaje de marcado HTML para la creación de espacios web dinámicos utilizando la sintaxis de Razor.

Estos tres marcos web que proporciona ASP.NET se basan en .NET Framework, compartiendo la funcionalidad principal de .NET y ASP.NET. Estos marcos funcionan de manera independiente, pero pueden existir conjuntamente dentro de la misma aplicación, siendo frecuente encontrar su uso combinado en bastantes proyectos. [7]

ASP.NET ha sido el marco de desarrollo web empleado para la creación de la aplicación web de este proyecto, en concreto, el marco de ASP.NET Web Forms.

3.3. C#

Es un lenguaje de programación proveniente del lenguaje C y basado en la utilización de objetos y cuenta con seguridad de tipos. Está pensado para el desarrollo de aplicaciones pensadas para ser ejecutadas en .NET [8].

Se trata de un lenguaje orientado a objetos dirigidos a componentes. Permite a los desarrolladores crear y usar componentes software mediante las construcciones de lenguaje que el propio lenguaje nos proporciona.

Características principales del lenguaje:

- Recolección de elementos no utilizados. Se encarga de demandar de forma automática la memoria ocupada por objetos no utilizados o inaccesibles.
- Tipos que aceptan valores NULL.
- Control de excepciones.

- Admisión de expresiones lambda.
- Language Integrated Query (LINQ). Patrón común que permite al lenguaje operar con datos independientemente de cuál sea su origen.
- Compatible con operaciones asíncronas.
- Implementa un sistema de tipos unificado
- Admite tipos de referencia que el usuario haya definido y tipos de valor.

C# ha sido el lenguaje utilizado para programar la parte lógica tanto de la aplicación local pensada para entornos Windows con .NET Framework como para la versión web en ASP.NET.

3.4. IIS Express

IIS Express es una versión ligera y portable del servidor web de Microsoft Internet Information Server (IIS). Cuenta con todas las capacidades de IIS 7 y posteriores, así como las siguientes características propias:

- No requiere de permisos de administrador para realizar sus tareas.
- Es capaz de trabajar con aplicaciones ASP.NET y PHP sin necesidad de instalar ningún complemento del servidor ni realizar ninguna configuración.
- Soporta a varios usuarios trabajando en el mismo equipo de forma simultánea.

Además, se trata del servidor web que viene incluido con el entorno de desarrollo de Microsoft Visual Studio, y es el que incorpora para probar aplicaciones web antes de su lanzamiento a producción [9].

IIS Express ha sido el servidor web para el que se ha ideado el panel de lanzamiento de la aplicación web.

3.5. Base de datos

Una base de datos es un conjunto de información estructurada que es almacenada en un sistema informático y controlada por un Sistema de Gestión

de Bases de Datos (DBMS).

Lo más habitual en los sistemas de bases de datos actuales es que la información se estructure y se almacene en forma de tablas, mediante la utilización de filas y columnas (bases de datos relacionales), y que se lea, se escriba, se edite y se elimine dicha información mediante la utilización de un Lenguaje de Consulta Estructurada (SQL)

Existe un inmenso número de tipos de bases de datos. Algunos de ellos son los siguientes:

- **Bases de datos relacionales:** Sus elementos se organizan en un conjunto de tablas, mediante filas y columnas. Su tecnología aporta la manera más eficiente y flexible de poder acceder a la información estructurada. Este tipo de base de datos ha sido el empleado en el proyecto para almacenar los datos de la placa y de la aplicación web.
- **Bases de datos orientadas a objetos:** Como su propio nombre indica, la información es estructurada en forma de objetos.
- **Bases de datos distribuidas:** Las que se encuentran en sistemas distribuidos, es decir, divididas entre distintos equipos.
- **Almacenes de datos:** Diseñados para consultas y análisis rápidos.
- **Bases de datos no relacional:** Permite almacenar y manejar información que no ha sido estructurada o información semiestructurada.
- **Bases de datos orientadas a grafos:** Almacena y estructura la información en nodos y relaciones entre nodos.
- **Bases de datos OLTP:** Pensada para que un gran número de usuarios realicen un gran número de transacciones, debido a ser una base de datos veloz y analítica.

Sistema de Gestión de Bases de Datos (DBMS)

Software que sirve como interfaz entre la base de datos y los usuarios o aplicaciones finales, de forma que estos pueden gestionar cómo se organiza la información [10].

También permite un gran número de operaciones de índole administrativa, ya que nos facilita el control y la monitorización de las bases de datos.

En la antigua versión de la aplicación se empleaba Microsoft SQL Server 2017 Express como sistema gestor de bases de datos. En este proyecto este gestor ha sido cambiado por un sistema de bases de datos basado en servicios, una herramienta de Microsoft Visual Studio y .NET Framework que incorpora una base de datos funcional en un único fichero.

3.6. Comunicación en Serie

En la comunicación en serie los datos se envían del dispositivo emisor al dispositivo receptor de forma secuencial bit a bit [11]. Los pasos que se siguen, como se puede apreciar en la Figura 3.4, son los siguientes:

1. Los datos se transforman de formato paralelo a formato secuencial.
2. Los bits se organizan uno tras otro, formando una serie.
3. Se envían los bits al dispositivo receptor, siendo el primero en ser enviado el bit menos significativo.
4. El dispositivo receptor recibe los datos y los vuelve a transformar a formato paralelo.

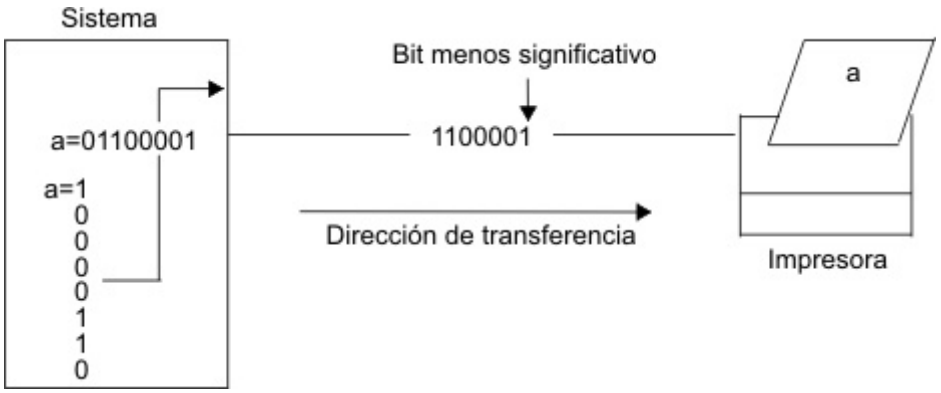

Figura 3.4: Comunicación en serie.

Pese a la simplicidad de este protocolo de comunicaciones, podemos encontrarnos con un importante problema, que el dispositivo receptor no sepa dónde finaliza un carácter y empieza el siguiente. Este problema surge cuando el mensaje está conformado por un gran número de caracteres. Para poder poner

solución a este problema, tanto el dispositivo emisor como el dispositivo receptor deben estar temporizados o sincronizados. En la Figura 3.5 se puede observar este problema.

Figura 3.5: Comunicación en serie con un gran número de bits.

La comunicación en serie se aplica en este proyecto para la comunicación entre la placa controladora de la planta piloto y la interfaz que se ejecuta en el equipo Windows que recibe y le envía datos. En concreto, se ha utilizado el tipo de comunicación en serie UART (Universal Asynchronous Receiver/Transmiter), el cual incorpora un transmisor, un receptor, generadores de reloj y una serie de registros que permiten configurar ciertos parámetros (el formato de los paquetes, las velocidades de la comunicación, etc.) [12]. La comunicación es bidireccional.

Técnicas y herramientas

A continuación, se describen las técnicas y herramientas que se han empleado a la hora del desarrollo del proyecto, su documentación, y la comunicación con los tutores del proyecto.

4.1. GitHub

GitHub es la plataforma más conocida y utilizada para el control de versiones y el desarrollo de aplicaciones de forma colaborativa a día de hoy. Emplea el sistema Git para permitir a sus usuarios la creación de repositorios donde ellos y sus compañeros pueden participar.

Ha sido la herramienta empleada en cuanto a control de versiones se refiere. Como Francisco Crespo Diez ya creó un repositorio para el desarrollo de su primera versión en 2019, se ha partido de un fork de dicho repositorio, el cual se puede apreciar en la Figura 4.6.

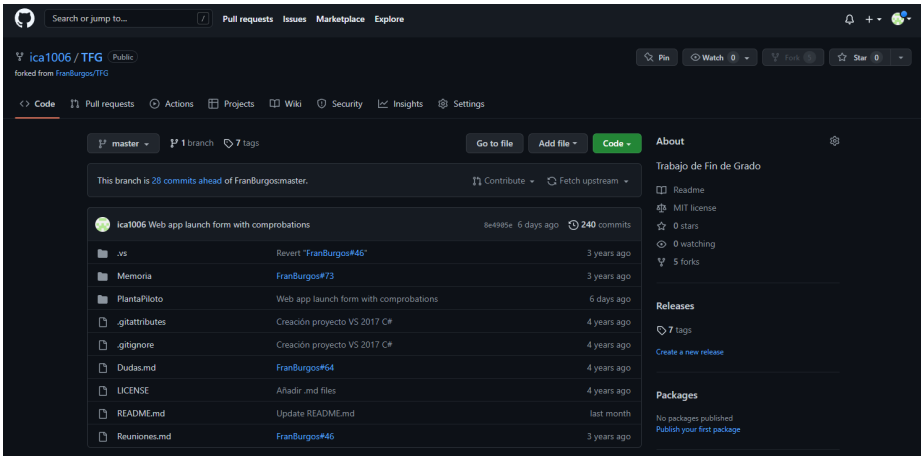

Figura 4.6: Repositorio del proyecto en GitHub.

4.2. GitKraken

GitKraken es un cliente multiplataforma que nos permite un manejo y una gestión de los repositorios de GitHub mediante el uso de una interfaz muy visual y sencilla.

Ha sido el cliente empleado para el control del repositorio en GitHub del proyecto debido a que, mediante su intuitiva y personalizable interfaz (Figura 4.7), la comunicación con GitHub y el registro de cambios han sido mucho más rápidos que mediante el uso de la consola de comandos.

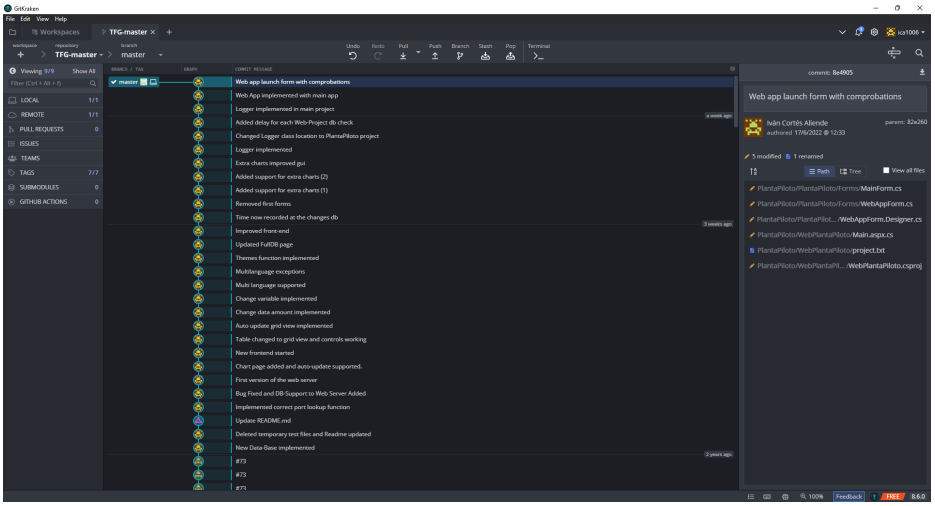

Figura 4.7: Interfaz de GitKraken con el repositorio GitHub del proyecto abierto.

4.3. Microsoft Visual Studio 2019

Microsoft Visual Studio 2019 se trata de un entorno de desarrollo integrado multiplataforma desarrollado por Microsoft y enfocado en el desarrollo de aplicaciones .NET y C++, dotado de multitud de herramientas y documentación para ello.

Se trata, como se puede observar en la Figura 4.8, del entorno de desarrollo integrado principal de todo el proyecto debido a que, al ser las principales bases del proyecto .NET Framework, ASP.NET y C++, era la opción más segura y recomendable.

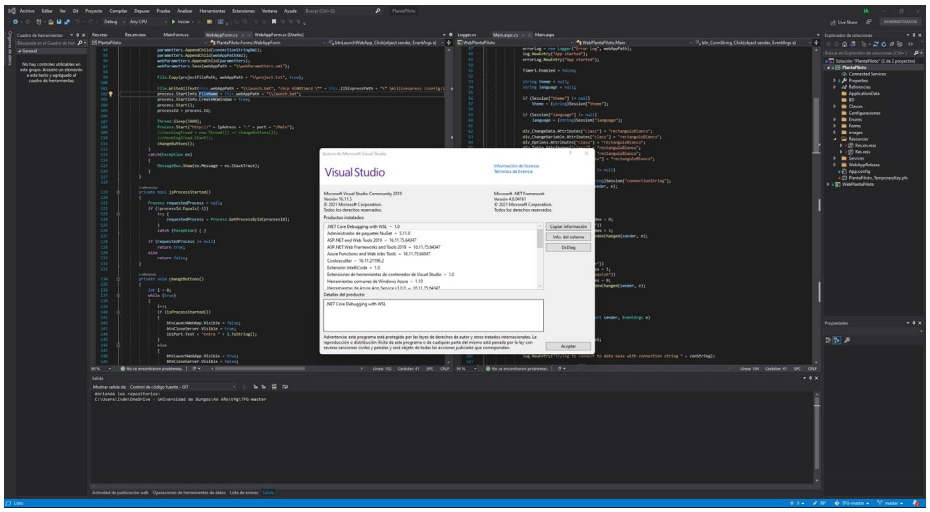

Figura 4.8: Microsoft Visual Studio con el proyecto abierto.

4.4. Microsoft Visual Studio Code

Microsoft Visual Studio Code es un editor de código fuente multiplataforma desarrollado por Microsoft muy versátil. Esto es debido a que mediante la instalación de distintas extensiones es posible trabajar con la mayoría de lenguajes de programación de una forma intuitiva, cómoda y rápida.

Ha sido empleado en momentos puntuales para edición de ficheros HTML, XML y TXT.

4.5. Microsoft Word

Microsoft Word es un software de procesamiento de textos desarrollado por Microsoft. Por defecto viene incluido en el paquete de software Microsoft Office.

Se trata de la herramienta empleada para la redacción de la documentación del proyecto. Al principio se pensó en emplear el sistema de composición de textos LATEX, pero se acabó descartando esa opción por la poca experiencia que tengo en esa tecnología, lo cual iba a acabar afectando a la velocidad del desarrollo. La documentación ha sido desarrollada empleando la licencia de Microsoft 365 (la cual incluye Microsoft Office y, por tanto, Microsoft Word) otorgada por la Universidad de Burgos a sus alumnos.

4.6. Microsoft SharePoint

Microsoft SharePoint es una plataforma virtual de colaboración empresarial e institucional desarrollada por Microsoft e incluida en la licencia de Microsoft 365 proporcionada por la Universidad de Burgos.

Ha sido empleada como una plataforma de compartición de ficheros entre los tutores de este proyecto y yo mismo, de tal forma que todos pudiéramos estar al día de la situación del proyecto (Figura 4.9).

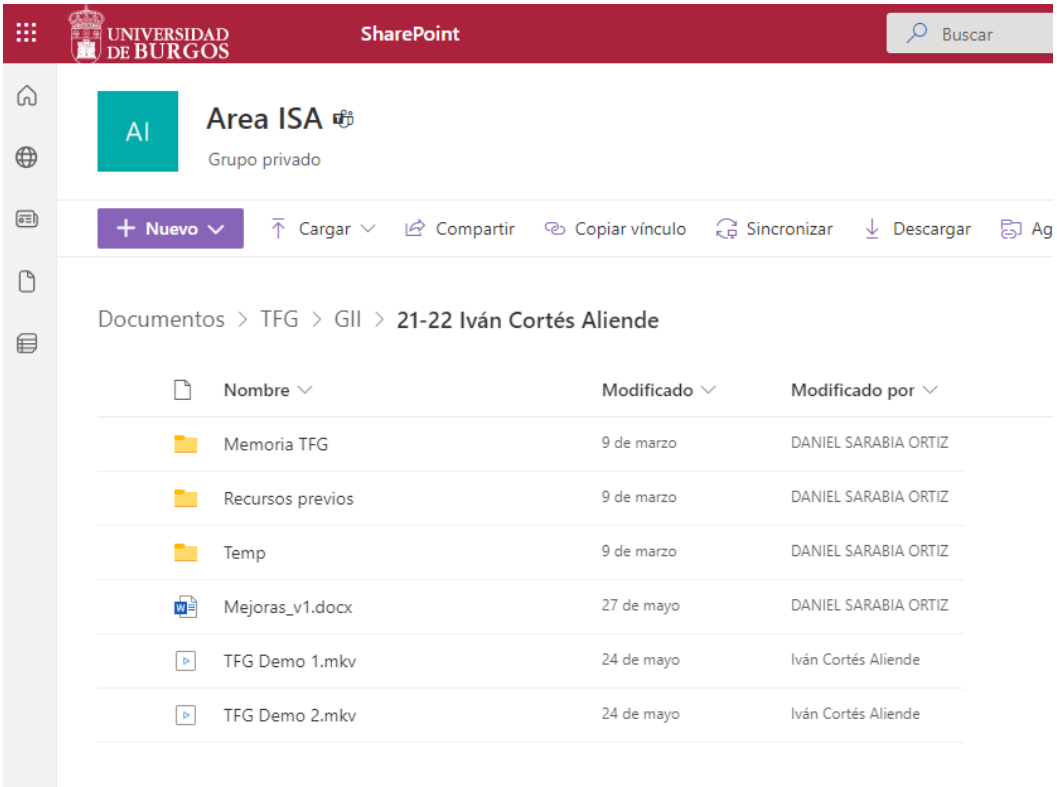

Figura 4.9: Carpeta del proyecto en el SharePoint de la UBU.

4.7. Microsoft Teams

Microsoft Teams es una plataforma virtual desarrollada por Microsoft y de índole empresarial e institucional con el objetivo de facilitar la colaboración del trabajo incorporando características como videollamadas, compartición de ficheros, chat de texto, etc.

Se trata de la plataforma a través de la cual han tenido lugar algunas reuniones con los tutores del proyecto.

4.8. IIS Express

Como ya se explicó en el punto de Conceptos Teóricos, IIS Express es un servidor web desarrollado por Microsoft pensado para mantener en línea páginas y aplicaciones web desarrolladas en ASP.NET o PHP. Se caracteriza por ser portable, ligero y consumir muy pocos recursos.

Ha sido empleado como servidor de pruebas durante el desarrollo de la aplicación web en ASP.NET, ya que tiene integración directa con el entorno de desarrollo Microsoft Visual Studio 2019.

Además, también ha sido el servidor web para el que se ha desarrollado la integración y automatización del lanzamiento de la aplicación web en la interfaz original (Figura 4.10).

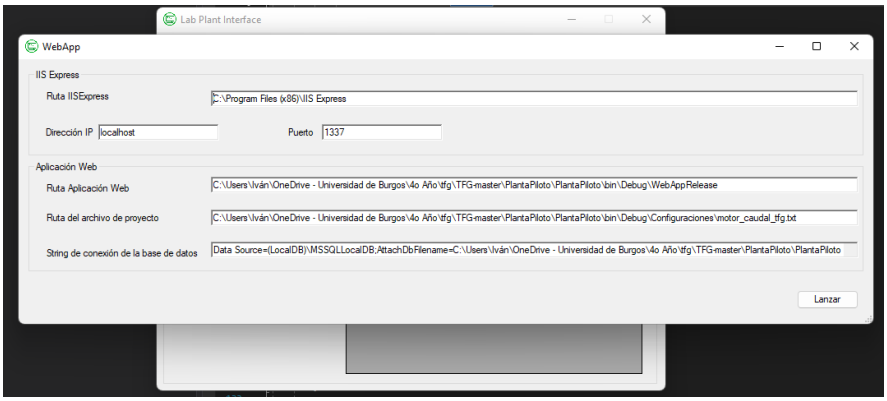

Figura 4.10: Pantalla de lanzamiento del servidor web mediante la integración de IIS Express.

En un principio se pensó en lanzar la aplicación web en un contenedor de Docker una vez su desarrollo hubiera finalizado. El problema fue que la imagen oficial de Microsoft para aplicaciones web ASP.NET tenía un tamaño excesivo, de unos 8GB, lo cual ralentizaría enormemente el flujo del trabajo en los

ordenadores de los laboratorios de la universidad, por lo que las prácticas de los estudiantes con la planta piloto se verían afectadas. Además, realizando pruebas se tuvieron problemas a la hora de instalar Docker en entornos Windows y automatizar el lanzamiento del contenedor (problemas de permisos, tiempos de espera muy elevados, alto consumo de recursos, etc.). Por todo ello, está opción se descartó y se optó por IIS Express por ser ligero, rápido y consumir muy pocos recursos.

4.9. Open Broadcaster Software

Open Broadcaster Software (OBS Studio) es un software de grabación y retransmisión en directo de código abierto y multiplataforma. Se caracteriza por consumir pocos recursos, permitir la gestión de multitud de dispositivos de entrada de vídeo y audio, creación y manejo de varias escenas durante la grabación y por ser altamente personalizable a la hora de establecer cómo procesar y codificar el vídeo.

Este software ha sido empleado para la grabación de distintos vídeos mostrando el estado del desarrollo del proyecto, los problemas solucionados y las nuevas funcionalidades implementadas, con el objetivo de informar a los tutores del estado del proyecto. También ha sido utilizado para la grabación de los vídeos de muestra de la aplicación adjuntos.

4.10. Lightshot

Lightshot es un sencillo software multiplataforma de realización de capturas de pantalla desarrollado por Skillbrains. Una vez presionado un determinado atajo de teclado, sobrepone en el sistema una interfaz mediante la cual se puede seleccionar un área de la pantalla donde realizar sencillas modificaciones (agregar textos, recuadros, flechas, etc.) y, al finalizar, guardar el área seleccionada y sus modificaciones como archivo de imagen.

Este software ha sido empleado a la hora de realizar capturas de la pantalla para el desarrollo de la documentación del proyecto, para la redacción del manual de usuario y para los archivos de ayuda de la aplicación.

4.11. Mockplus RP

Mockplus RP es un editor visual que permite, mediante un sistema muy sencillo de arrastrar y soltar (Figura 4.11), diseñar interfaces para aplicaciones de cualquier ámbito (escritorio, web, móvil, etc.). Se trata de una aplicación web basada en la computación en la nube con un plan gratuito y otro de pago.

Ha sido utilizado para el diseño de las distintas interfaces de la aplicación web antes incluso de empezar su desarrollo, de tal forma que al empezar a desarrollarla ya estaban claros los distintos campos que la conformaban y su estructura, lo cual hizo más ágil este proceso. Pese a que también soporte exportar las interfaces diseñadas directamente a HTML y CSS para su maquetado, este fue realizado a mano en Microsoft Visual Studio 2019, por limitaciones del plan gratuito y para poder asegurarme de que el código fuera lo más limpio y óptimo posible.

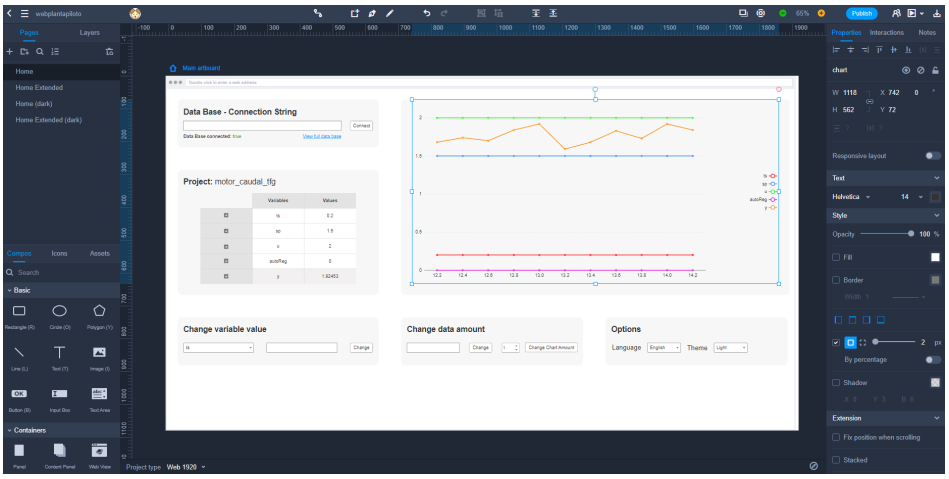

Figura 4.11: El editor de Mockplus RP con el proyecto de la interfaz principal de la aplicación web.

4.12. P&E Driver

P&E Driver es un software desarrollado por P&E Microcomputer Systems que otorga a un entorno Windows de la capacidad de emular una comunicación en serie UART en un puerto USB.

Este software es el que otorga la capacidad de conectar la placa FRDM-K64F de la planta piloto con los entornos Windows en los que los estudiantes realizan las prácticas mediante un cable USB. Mediante este software, el sistema operativo es capaz de leer los valores que le llegan de la placa y de mandar

mensajes él mismo. Por lo tanto, ha sido esencial a la hora del desarrollo del proyecto, pues sin él la comunicación con la placa no hubiera sido posible. Fue proporcionado por los tutores del proyecto al comenzar con el desarrollo de este.

4.13. Katalon Recorder

Katalon Recorder es una extensión de navegador pensada para la automatización de pruebas en aplicaciones web. Se compone de una interfaz muy simple que permite la grabación de acciones del usuario para más tarde replicarlas de forma automática. Es compatible con Selenium IDE.

Ha sido la herramienta empleada para la realización de pruebas automáticas de entrada de datos en la aplicación web.

4.14. HTML Help Workshop

HTML Help Workshop es una aplicación pensada para compilar archivos HTML en proyectos CHM (Archivo de Ayuda de HTML Compilado), que son los utilizados por aplicaciones desarrolladas para entornos Windows para mostrar los menús de ayuda.

Como es de suponer y como se puede observar en la Figura 4.12, ha sido la herramienta utilizada para la actualización de los menús de ayuda con las nuevas funcionalidades implementadas.

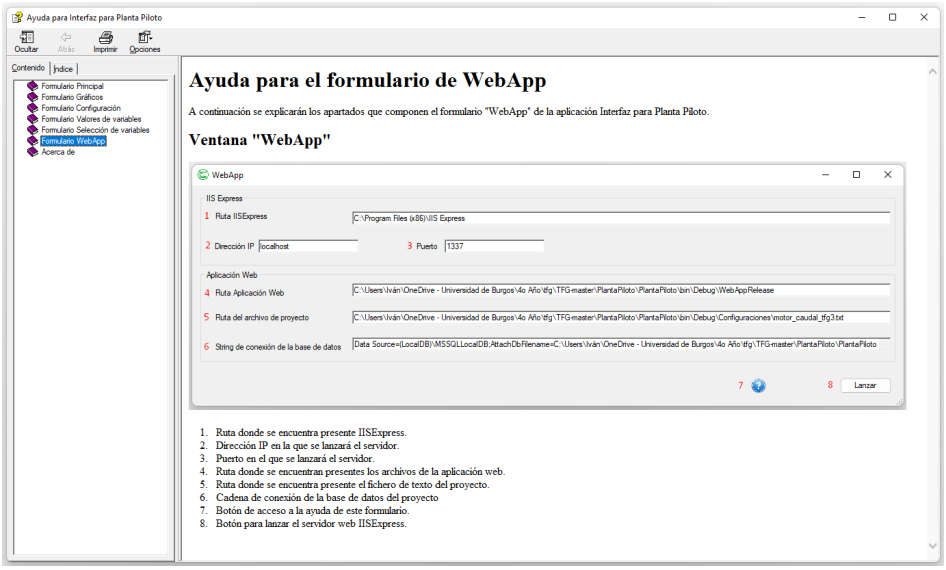

Figura 4.12: Menú de ayuda compilado en formato CHM.

4.15. Modelo Vista Controlador (MVC)

Se trata de un modelo de arquitectura software que separa en tres capas los distintos componentes de una aplicación [13]:

- Modelo: la capa relativa al tratamiento de los datos.
- Vista: hace referencia a la interfaz del usuario.
- Controlador: intermediario entre la vista y el modelo.

Francisco Crespo Diez siguió este modelo arquitectónico durante el desarrollo de la primera versión de la interfaz para la planta piloto, por lo cual se ha seguido empleando durante esta revisión a fin de seguir un procedimiento coherente y óptimo.

4.16. SonarQube

SonarQube es un software libre y multiplataforma que incorpora una serie de herramientas para el análisis de código fuente, con el objetivo de detectar bugs, vulnerabilidades, potenciales fallos de seguridad, fallos de diseño, complejidad ciclomática, código duplicado, etc. Se divide en tres planes de pago y uno gratuito, cada uno con una serie de características diferentes. También tiene una alternativa basada en cloud, *SonarCloud*.

Se ha empleado el plan gratuito de SonarQube para la detección de fallos y mejora del código, tanto de la nueva aplicación web como para la interfaz local (Figura 4.13).

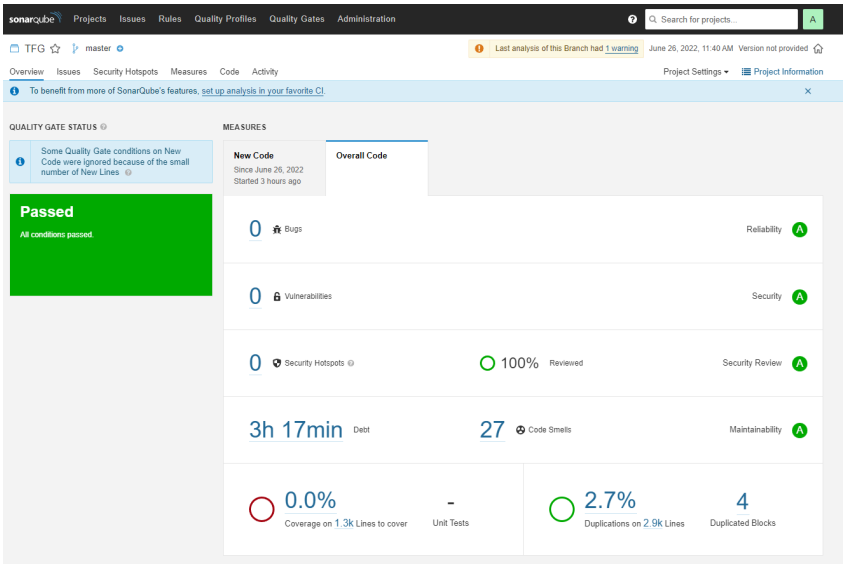

Figura 4.13: Panel de SonarQube con las distintas métricas.

4.17. Kanban

El sistema *Kanban* es una metodología de trabajo ágil basado en la división del trabajo en una serie de tarjetas que colocadas en un tablero. La posición que estas tarjetas tomen en dicho tablero representará la etapa del desarrollo en la que se encuentran. Las tareas dispondrán de miembros asignados a las mismas, así como de plazos de entrega, si así se desea. De esta manera, *Kanban* permite que de un único vistazo podamos saber en estado del desarrollo nos encontramos, qué trabajadores son los que más trabajo tienen, qué tareas se han completado y cuales están por empezar, etc. A continuación se mencionan las cinco propiedades primordiales de Kanban [14].

1. Visualizar el flujo de trabajo
2. Limitar el número de tareas en desarrollo
3. Medir y administrar el flujo de trabajo
4. Hacer explícitas las políticas de procesos
5. Utilizar modelos para identificar las oportunidades de mejora

Kanban ha sido la metodología de trabajo ágil utilizada durante todo el desarrollo del proyecto. Ha servido como método de planificación temporal mediante la utilización de plazos de tiempo en las tareas. En concreto, se ha utilizado Trello como plataforma virtual para el empleo de esta metodología.

Trello

Trello es una herramienta web que permite la creación de tableros y tareas colaborativas basándose en la metodología Kanban, caracterizándose por su sencilla e intuitiva interfaz. Sirve, por tanto, como herramienta para la planificación y gestión de tareas de un proyecto.

Se ha utilizado un tablero Trello en su plan gratuito como herramienta mediante la cual aplicar la metodología Kanban al desarrollo del proyecto. En la Figura 4.14 puede visualizarse dicho tablero.

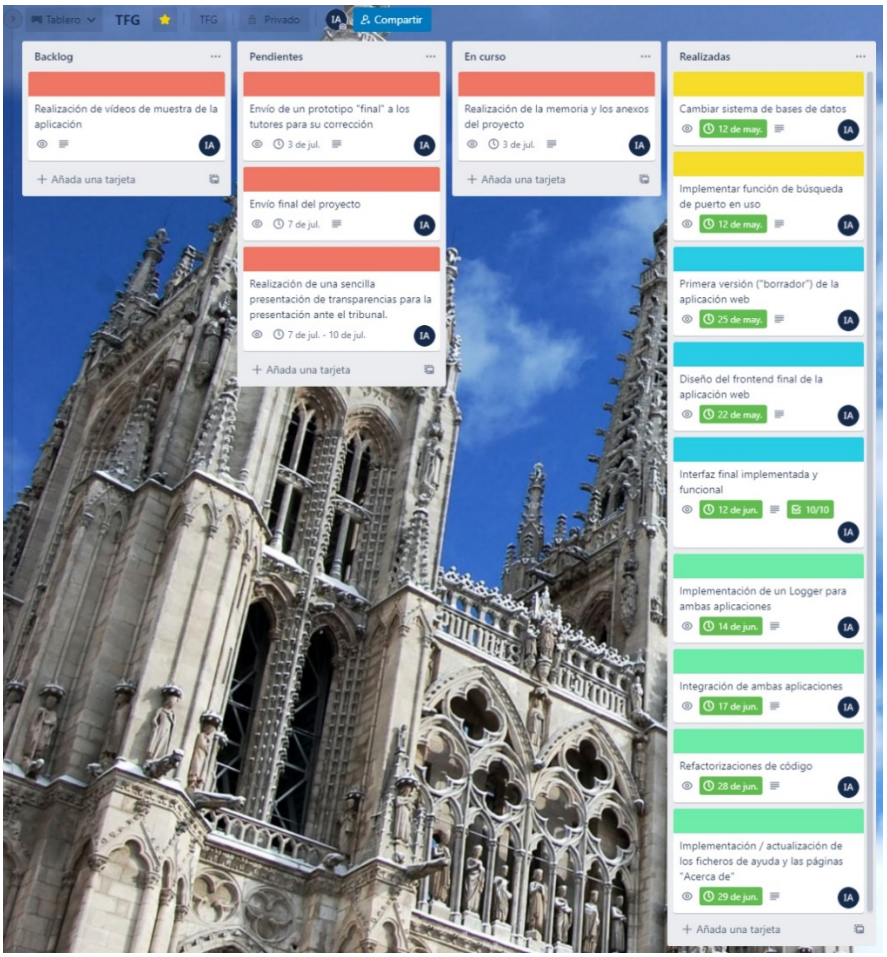

Figura 4.14: Tablero Trello del proyecto.

Aspectos relevantes del desarrollo del proyecto

En este apartado se explican los aspectos más relevantes del desarrollo del proyecto, haciendo hincapié en otras alternativas posibles, problemas que surgieron durante el desarrollo, cómo se solucionaron, etc.

5.1. Primeros pasos del proyecto

Me decidí por la realización de este proyecto porque el desarrollo de una aplicación tradicional era algo que me daba la oportunidad de poner en práctica de manera conjunta todos los conocimientos adquiridos en la carrera, cubriendo todas las fases de desarrollo de la aplicación, algo que un proyecto con un enfoque más teórico no me podía aportar. Además, veía muy interesante para mi formación personal adaptarme y partir de un código ya escrito por un compañero, ya que es una situación de lo más habitual en el mercado laboral.

Por todo ello, decidí reunirme con los tutores para informarme más al respecto. Al finalizar dicha reunión decidí aceptar la propuesta del proyecto, ya que era exactamente lo que estaba buscando.

5.2. Comprensión del código y formación

Al partir de un código ya desarrollado en una primera versión por un compañero, las primeras semanas me centré exclusivamente en leer y entender cada uno de los apartados del código, con la intención de saber manejarme con soltura por la estructura del código y poder implementar mis cambios de una manera eficiente. Para ello me fue de gran ayuda el repositorio de GitHub de que en su día utilizó Francisco Crespo Diez, ya que estaba muy bien estructurado y con una documentación muy bien redactada. Eso sí, me hubiera gustado que los commits tuvieran una explicación más desarrollada de qué era lo que se implementaba o se corregía, ya que muchos de ellos no tenían ninguna clase de explicación, y eso me resultó un poco molesto a la hora de entender el procedimiento que había seguido en el desarrollo.

Ya tenía unos conocimientos previos básicos de **C#** y **SQL**, adquiridos durante la carrera, pero al comienzo del proyecto profundicé en ambos con la intención de poder desarrollar el TFG de una manera rápida y eficiente. Para ello, empecé con C#. La sintaxis del lenguaje era algo que no me suponía mayor problema, ya que es muy similar a Java, un lenguaje con el que tengo bastante más experiencia. Por ello, me centré más en informarme de cuáles eran las bibliotecas más utilizadas e importantes a la hora de desarrollar en C# y .NET Framework, tanto aplicaciones de escritorio para entornos Windows como aplicaciones web ASP.NET.

En cuanto a SQL estuve investigando acerca de la variante utilizada en **Microsoft SQL Server**, el sistema gestor de base de datos que se pensó para ser utilizado junto con esta aplicación. Dicho lenguaje variante es **Transact SQL** (T-SQL), y lo encontré bastante similar a PostgreSQL, el lenguaje que más he utilizado durante la carrera. Por ello, la sintaxis nuevamente volvió a no ser un problema, así que me centré en formarme acerca de cómo acceder a una base de datos Microsoft SQL Server de la manera óptima desde una aplicación .NET Framework. Por suerte no tuve mucho al problema al respecto, ya que al ser tanto .NET Framework como Microsoft SQL Server desarrollados por Microsoft, esto es algo muy facilitado y para lo que existe una gran cantidad de documentación.

5.3. Cambio de la base de datos

Uno de los cambios en el que más insistieron los tutores a la hora de la descripción del proyecto fue en el cambio del sistema gestor de bases de datos. Como ya he comentado, en un principio la aplicación utilizaba Microsoft SQL Server. El problema es que se debía tener en ejecución el gestor mientras se utilizaba la aplicación, lo cual consumía muchos recursos de los equipos del laboratorio. Además, se necesitaba realizar una configuración previa poco intuitiva y algo complicada del gestor de base de datos para que la aplicación pudiera acceder a él, lo cual complicaba todavía más las cosas.

Se barajaron varias alternativas, tales como MariaDB Server o MongoDB Community Server. Ambos son gestores de bases de datos más livianos que Microsoft SQL Server, por lo que la ejecución de cualquiera de los dos consumiría menos recursos. El problema es que seguía existiendo el inconveniente de tener que ejecutar cualquiera de estos gestores previamente y realizar una configuración apropiada para que la aplicación se pudiera conectar, por lo que ninguna de esas opciones era óptima.

Buscando alternativas, me encontré con las Bases de datos basadas en servicio que proporciona .NET Framework. Estas bases de datos se componen de un único fichero en formato MDF que incorpora una base de datos funcional que se lanza de forma automática al ejecutar la aplicación, utilizando una variante de Microsoft SQL Server basada en librerías dll, por lo que no hace falta instalar ningún SGBD ni configurarlo. De esta forma, la base de datos solo está disponible cuando la aplicación se está ejecutando. Esto solventaba el consumo de recursos, ya que es reducido enormemente; el problema de lanzar el sistema gestor de base de datos antes de ejecutar la aplicación, ya que se realiza de forma automática; y la configuración previa, ya que incorpora una configuración por defecto especificada durante la etapa de programación. Por lo tanto, esta fue la opción finalmente elegida, ya que cumplía con todos los requisitos buscados.

Además, esta base de datos también se basa en el lenguaje T-SQL, por lo que apenas hubo que hacer cambios en los comandos especificados en la aplicación.

5.4. Detección de comunicación en serie

Otra de las mejoras que los tutores veían más importantes era la de incorporar la funcionalidad que permitiera que los estudiantes no tuvieran que lidiar con buscar qué puerto en serie se estaba empleando y pudieran automatizar este apartado (Figura 5.15).

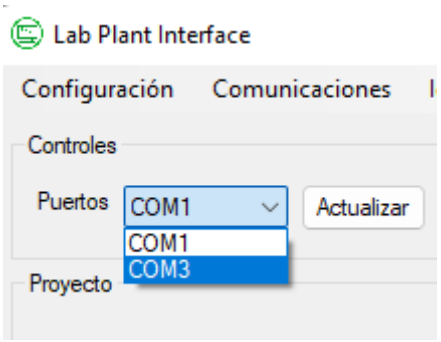

Figura 5.15: Menú de selección del puerto en serie.

A la hora de comenzar con la comunicación en serie entre la interfaz y la placa FRDM-K64F es necesario especificar en qué puerto del equipo es por el que se está produciendo la comunicación. Los alumnos no tenían ninguna manera de saber esta información, por lo que tenían que ir probando hasta dar con el puerto en uso.

Para solucionar ese problema se implementó un botón "Buscar Puerto" que ejecuta una función que sigue los siguientes pasos:

1. Actualiza el listado de puertos, por si la placa se ha conectado después de lanzar la aplicación.

2. En bucle, lee el buffer de entrada de todos los puertos. Si no encuentra información en uno de los puertos, no hace nada (pasa al siguiente), si la encuentra queda marcado como el puerto en uso.

3. Si pasan unos 20 segundos y no se ha dado con el puerto en uso, el bucle se para y se devuelve un mensaje al usuario notificándole el suceso (Figura 5.16).

4. Si se detecta un puerto en uso, el bucle se detiene, se muestra el nombre de dicho puerto al usuario en un mensaje y se le selecciona en el menú automáticamente (Figura 5.17).

Como se puede observar, esta solución es muy simple, pero se espera que se

traduzca en tiempo perdido ahorrado a los alumnos de tal forma que las clases sean más dinámicas y productivas.

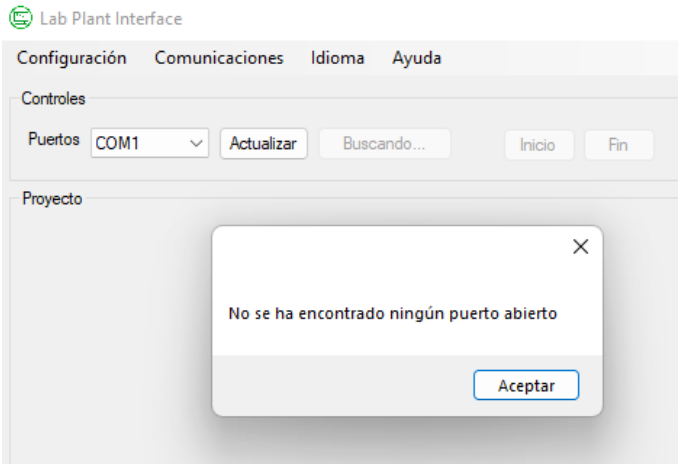

Figura 5.16: Mensaje mostrado al no encontrar ningún puerto en uso.

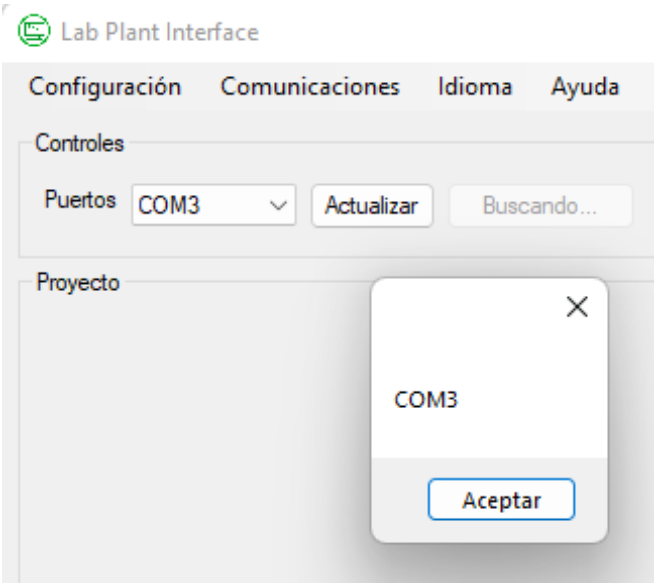

Figura 5.17: Mensaje mostrado al usuario al encontrar un puerto en uso y seleccionarlo en el menú correspondiente.

5.5. Desarrollo de la aplicación web

La tarea principal de este proyecto ha sido el desarrollo de una aplicación web muy similar a la interfaz ya desarrollada por Francisco Crespo Diez y que se integre a esta como una funcionalidad nueva.

Estando desarrollada la interfaz para .NET Framework, tuve claro desde el principio desarrollar la aplicación web en ASP.NET, debido a que la integración

es muy sencilla al basarse en la misma tecnología. Por ello, creé un nuevo proyecto en Microsoft Visual Studio 2019 dentro de la misma solución que la interfaz original y creé una referencia desde la aplicación web al proyecto original, de manera que pudiera reutilizar el código de sus clases y evitar código duplicado.

Como dos aplicaciones no pueden acceder al mismo puerto en serie de forma simultánea tuve que buscar una alternativa para poder recibir y enviar datos desde la aplicación web. Para ello, utilicé la base de datos como intermediaria para la comunicación entre la aplicación web y la aplicación original, y esta a su vez sería la encargada de enviar y recibir información por el puerto en serie a la placa (Figura 1.3).

De esta forma ya podía recibir e interpretar todos los datos recibidos y representar datos en función a estos, pero el problema vino a la hora de enviar información desde la aplicación web a la placa. La forma que me pareció más sencilla era la de crear una nueva tabla en la base de datos. En un uso habitual, se crea una tabla por cada proyecto en la que se van almacenando los datos recibidos por el puerto en serie provenientes de la placa, y luego esta tabla se consulta para la representación de los datos en la interfaz en tablas y gráficas. Pues bien, para poder comunicar la aplicación web con la placa se crea una tabla suplementaria cada vez que se lanza la aplicación web. En esa tabla se almacenan los datos del mensaje que se quiere enviar a la placa (Nombre de la variable a cambiar + nuevo valor que va a tomar) y la aplicación original consultará esta tabla una vez cada segundo (para evitar sobrecargar de peticiones a la base de datos) y si detecta una nueva entrada transforma el contenido a un mensaje en serie que siga el formato especificado en la placa y se lo envía.

Para la representación de los datos en forma de gráficos se barajó la posibilidad de utilizar la API de gráficos de Google (utilizada en la suite de Google Apps) ya que proporcionaba un mayor número de controles, herramientas e información. Esta opción fue descartada porque requería que los equipos tuvieran conexión a Internet y, teniendo en cuenta que este proyecto está pensado para ser utilizado en los laboratorios de la Escuela Politécnica Superior, llegamos a la conclusión de que esto podía limitarnos más que ayudarnos. Actualmente se utiliza la biblioteca de gráficos por defecto de .NET Framework.

Las páginas que componen la aplicación web se muestran a continuación.

Página principal

Se trata del panel de inicio que los usuarios se van a encontrar nada más conectarse a la aplicación. Con él podrán realizar la mayoría de las funciones que permite la interfaz local, tales como visualizar los valores de las variables, asignar nuevos valores a dichas variables, representación de los datos en gráficas, etc.

En la Figura 5.18 podemos apreciar la página principal de la aplicación web, con los siguientes apartados señalados en rojo:

Figura 5.18: Página principal de la aplicación web.

1. Campo donde figura el Connection String para la conexión con la base de datos. Si la conexión se realiza correctamente, se mostrarán todos los campos y aparecerá un "true" en verde justo debajo.

2. Enlace para visualizar la página donde figuran todas las tablas de la base de datos en relación con el proyecto.

3. Tabla con los valores en tiempo real de las variables que componen el proyecto.

4. Enlace para visualizar la página "Acerca de", con información acerca de la autoría de la aplicación web.

5. Botón para visualizar la ayuda de la página principal.

6. Campo relativo al cambio de valor de una variable de escritura del proyecto. Mediante este campo, podemos seleccionar un nuevo valor de una variable, que será guardado en la tabla de cambios que la aplicación tradicional detectará y enviará a la planta piloto.

7. Campo para seleccionar la cantidad de datos que se representarán en los gráficos y la cantidad de gráficos que se mostrarán.

8. Desplegable para seleccionar el idioma deseado de la aplicación.

9. Desplegable para seleccionar el tema de visualización de la aplicación (claro u oscuro).

10. Gráfico principal de la aplicación. En él se representarán las variables marcadas en la tabla de la izquierda (punto 3).

Página con la información de la base de datos

En esta página los usuarios podrán ver toda la información almacenada en la base de datos del proyecto, de tal manera que puedan estudiar cómo cambian las variables en el tiempo.

Siguiendo con la Figura 5.19, destacamos los siguientes apartados en rojo:

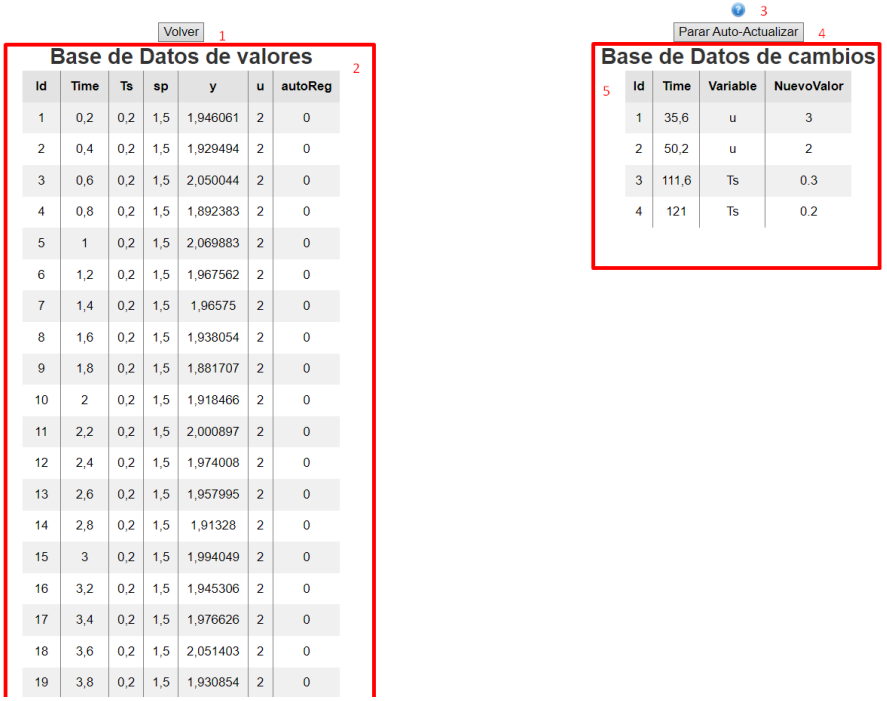

Figura 5.19: Página con la información de la base de datos

1. Botón para volver a la página principal.
2. Tabla del proyecto con todos los datos recibidos hasta el momento de las distintas variables por el puerto en serie.
3. Botón para mostrar la ayuda de esta página.
4. Botón para parar o reanudar la actualización automática de los datos mostrados en las tablas.
5. Tabla del proyecto relativa a los cambios de variables enviados desde la página principal de la aplicación web.

Página "Acerca de"

Como se puede apreciar en la Figura 5.20, en esta página se muestra la autoría de la aplicación web y el enlace de vuelta a la página principal.

Figura 5.20: Página con la información relativa a la autoría de la aplicación web.

5.6. Sistema de registros

Otro de los requisitos que debía estar presente en este proyecto es un sistema de registro en forma de fichero de texto (Log). Esto es debido a que se detectó que en la anterior versión de la interfaz se producían errores puntuales y al usuario le era complicado averiguar el por qué.

El sistema de logs implementado es un sistema muy simple que registra las acciones del usuario y sus consecuencias en dos ficheros de texto distintos, dependiendo de su naturaleza:

- En un fichero se almacenan las acciones del usuario y las acciones que el programa lleva a cabo en función de éstas. También se almacena el resultado obtenido en caso de que no genere una excepción. En caso de que se genere una excepción, se agrega una nueva entrada a este fichero indicando que una excepción se ha producido y que para más información se consulte el log de errores de esa sesión.
- En otro log se almacenan exclusivamente las excepciones producidas. Para ello, en cada entrada se registra una pequeña descripción de donde se ha producido, el mensaje que esa excepción ha devuelto y la ubicación de la instrucción que ha generado dicha excepción.

Para evitar la creación de una gran cantidad de ficheros que puedan sobrecargar el almacenamiento del sistema, cada vez que se crea un log se consulta el número de archivos presentes en el directorio “Logs” provenientes de otras sesiones previas. Si existen 12 logs o más (6 normales y 6 de errores) se borran los más antiguos hasta llegar a 10 y se crean los nuevos, de manera que en una situación habitual haya siempre 12 logs (Figura 5.21). El directorio “Logs” de la aplicación web es distinto del directorio “Logs” de la aplicación tradicional.

El nombre del fichero de registro se compone de una pequeña descripción y la fecha y hora en el que fue creado. Por cada entrada también se guarda la fecha y hora exacta en el que fue registrada, de manera que se facilite el seguimiento de las acciones del usuario y las posibles generaciones de excepciones (Figura 5.22).

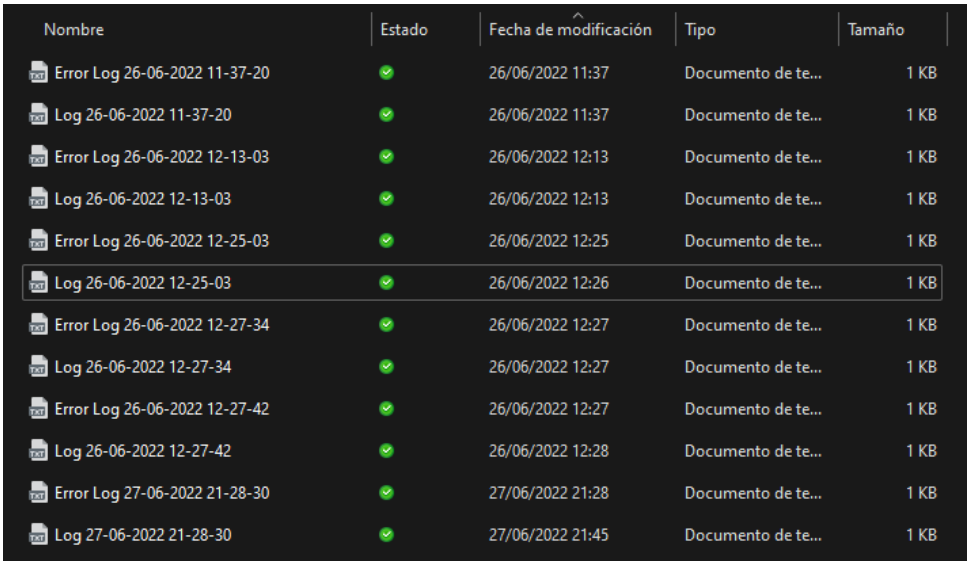

Figura 5.21: Directorio "Logs" con registros de anteriores sesiones.

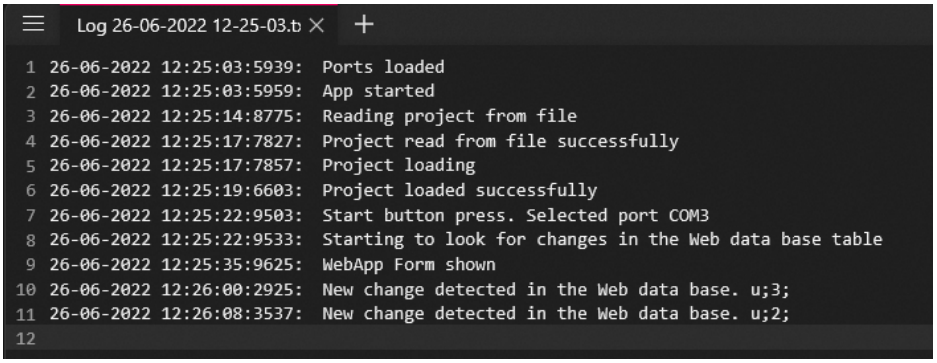

Figura 5.22: Ejemplo de un fichero de registro de una sesión previa.

5.7. Unión de ambas aplicaciones

Una vez desarrollada la aplicación web quedaba cómo integrar su lanzamiento y uso en la aplicación original. Para ello se implementó una opción que, una vez cargado un proyecto e iniciada la comunicación en serie, desplegaba un pequeño formulario con información básica acerca del lanzamiento del servidor web (Figura 5.23).

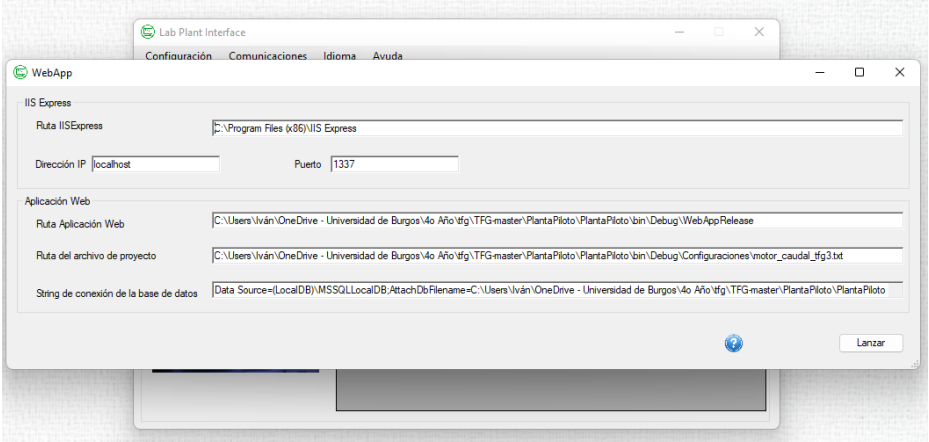

Figura 5.23: Formulario de lanzamiento del servidor web.

Como se puede apreciar en la figura superior y como ya se comentó en el apartado 4.8 de este documento, el servidor web para el que se diseñó la integración de ambas aplicaciones fue IIS Express, debido a sus bajos requisitos de uso, su bajo peso y su portabilidad. Se barajó utilizar Docker y ejecutar la aplicación en forma de contenedor, pero se descartó esta opción por el alto peso de la imagen oficial de Microsoft de ASP.NET para Docker y su alto consumo de recursos.

La forma en la que se lanza la aplicación web es muy sencilla. Esta está presente dentro de un directorio de la raíz de la aplicación. Cuando se presiona el botón “Lanzar” en el formulario de la Figura 5.23 se ejecuta una serie de comandos que realizan las siguientes acciones:

1. Navega hasta el directorio de IIS Express.

2. Lanza IIS Express con un fichero de configuración pasado por parámetro. Dicho fichero se encuentra presente en la carpeta de la aplicación web y, antes de lanzarse los comandos se editan los campos relativos a la IP, al puerto y a la ruta de la aplicación web con los parámetros especificados por el usuario en el formulario de la Figura 5.23.

3. Se pausa la consola hasta que el usuario presione una tecla del teclado. Esto es así para que, en caso de un fallo de IIS Express que conlleve una parada de este, el usuario pueda ver la información mostrada en la consola y esta no se cierre de forma repentina.

Durante la ejecución de IIS Express se pueden monitorear mediante la consola mostrada las distintas peticiones HTTP que los usuarios realizan al

servidor (Figura 5.24).

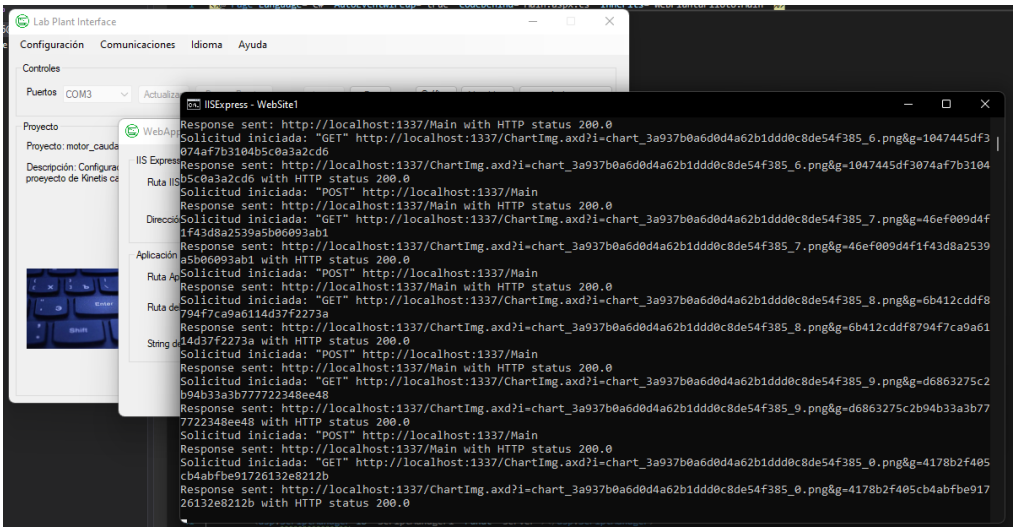

Figura 5.24: Servidor web lanzado desde la aplicación original.

5.8. Refactorizaciones y pruebas

Durante el desarrollo de la aplicación web se llevaron a cabo una serie de pruebas automatizadas mediante Katalon Recorder. Estas pruebas se encargaban de probar una serie de datos de entrada para los distintos apartados del formulario y comprobaban que la respuesta recibida fuera la apropiada. Esto permitió la detección de fallos de una manera muy rápida y efectiva, ahorrando posibles fallos de la funcionalidad de la aplicación.

Tanto para la aplicación web como para la aplicación tradicional se utilizó SonarQube en las últimas fases del desarrollo para la detección de código duplicado, bad smells, bugs, vulnerabilidades y posibles fallos de seguridad. Esto acabó resultando en refactorizaciones en una parte del código fuente del proyecto para evitar todas estas malas prácticas a la hora de programar, con el objetivo de tener un código fácilmente comprensible y de calidad. Para ambos análisis el resultado obtenido fue de A, y se pueden ver el registro de actividades en las Figuras 5.25 y 5.26.

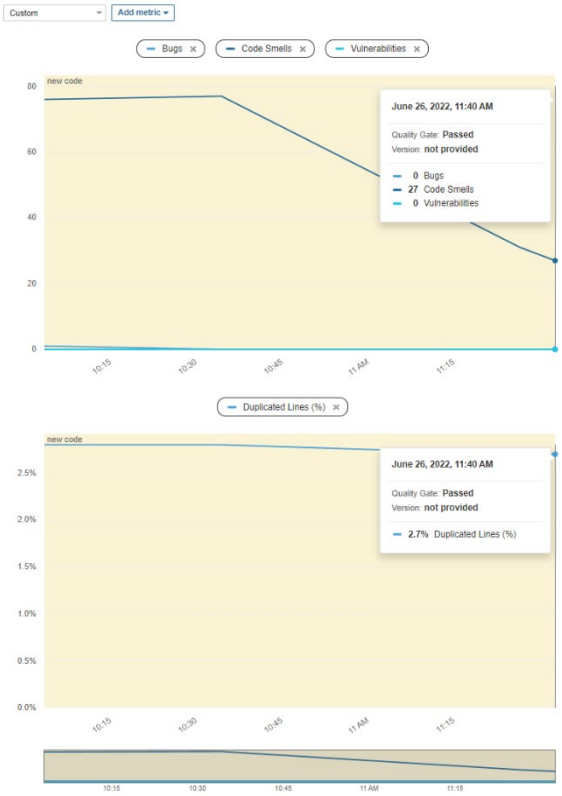

Figura 5.25: Gráfico de actividad de la aplicación local en SonarQube.

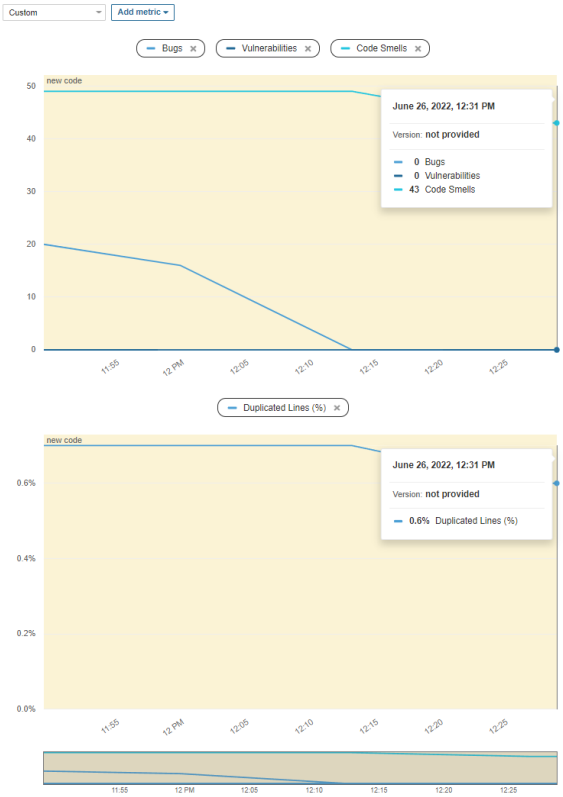

Figura 5.26: Gráfico de actividad de la aplicación web en SonarQube.

Trabajos relacionados

En este apartado se tratarán otros trabajos relacionados con el proyecto. En este caso particular este apartado tiene cierto valor, pues este ya es el cuarto trabajo fin de grado relacionado con la planta piloto

6.1. Primera versión de la planta piloto

Rubén Zambrana Rodríguez, egresado en Ingeniería Electrónica Industrial y Automática por la Universidad de Burgos, en 2017 realizó el proyecto "Diseño, construcción y control de una planta de laboratorio para la regulación de temperatura y caudal de aire" [2]. Dicho proyecto culminó en la construcción de una primera versión de la planta piloto, la cual contaba con una resistencia, un ventilador, un caudalímetro y una sonda.

6.2. Segunda versión de la planta piloto

María Isabel Revilla Izquierdo, egresada en Ingeniería Electrónica Industrial y Automática por la Universidad de Burgos, en 2018 realizó el proyecto "Implementación de reguladores PIDs industriales y reguladores avanzados en microcontroladores" [3]. Durante dicho proyecto se modificó la primera versión de la planta piloto, culminando en la versión utilizada actualmente en los laboratorios del área de Ingeniería de Sistemas y Automática de la Universidad

de Burgos.

6.3. Primera versión de la interfaz para planta piloto

Francisco Crespo Diez, egresado en Ingeniería Informática por la Universidad de Burgos, en 2019 realizó el proyecto "Desarrollo de una interfaz para planta piloto" [1]. Durante dicho proyecto simplificó y mejoró la comunicación entre el usuario y la planta piloto, desarrollando una interfaz intuitiva que solucionaba el problema de los alumnos de tener que lidiar con el uso de una pantalla de comandos para poder interactuar con la placa. Además, la interfaz permite la representación de datos en forma de tablas y gráficos, lo cual ayuda y simplifica el aprendizaje de los alumnos.

Este ha sido el proyecto del que se ha partido directamente para la elaboración de esta segunda revisión, corrigiendo problemas e implementando nuevas funcionalidades.

Conclusiones y líneas de trabajo futuras

Este apartado está destinado a contener las conclusiones obtenidas tras el desarrollo del proyecto y las posibles líneas de trabajo futuras de cara a continuar con el desarrollo del proyecto.

7.1. Conclusiones

A nivel personal termino este proyecto con un muy buen sabor de boca, me ha dado la oportunidad de utilizar todos los conocimientos adquiridos durante estos cuatro años de carrera en un contexto práctico. Ha sido la primera ocasión en la que he podido desarrollar un proyecto partiendo de otro ya desarrollado por un compañero, Francisco Crespo Diez, una situación que se da habitualmente en el mundo laboral, para el que siento que ahora estoy más preparado.

Además, tampoco había tenido la oportunidad de trabajar en todas las etapas de un proyecto, desde su diseño hasta su finalización, ya que durante la carrera normalmente afrontamos cada una de estas fases de manera individual, en asignaturas distintas. El tener que lidiar con todas estas etapas en el mismo contexto te permite valorar por qué es importante cada una de ellas, ver que aportan al proyecto, algo que en algunos casos no era capaz de ver tan

claramente al no disponer de un contexto tan amplio como este.

Hablando de las tecnologías empleadas, .NET Framework era una de las que más me llamaron la atención durante la carrera, ya que simplifica el trabajo a los desarrolladores mediante un entorno de desarrollo y de ejecución muy integrado, permitiéndoles centrarse en los aspectos más relevantes del proyecto. Este proyecto me ha dado la oportunidad de formarme más en esta tecnología y en C#, lenguaje que no había utilizado demasiado durante la carrera. Por lo tanto, me han aportado una experiencia muy valiosa para mi futuro desempeño en el mercado laborar.

Concluyendo este apartado, siento que este proyecto me ha nutrido mucho a nivel personal y profesional, ha conseguido que me sienta más seguro de mis capacidades como futuro ingeniero informático y me ha otorgado una experiencia muy valiosa. Además, la propia naturaleza del proyecto (dar más herramientas a los alumnos de la Universidad de Burgos para su formación) me hace sentir orgulloso de poder ayudar a mis compañeros y ser una pequeña parte de su paso por la universidad.

7.2. Líneas de trabajo futuras

El desarrollo de este proyecto y la comunicación con los tutores me ha permitido detectar una serie de funcionalidades que se podrían implementar en un futuro y que mejorarían su experiencia de uso.

Implementar el estándar OPC

OPC es un estándar de comunicaciones muy utilizado en el mundo industrial. Se basa en la implementación de una interfaz de comunicaciones común que permite a los componentes interactuar entre sí y compartir información [15].

Implementar este estándar en la comunicación entre la placa y la interfaz sería una muy buena idea, pues simplificaría las futuras revisiones del código, además de facilitar la experiencia del usuario.

Comunicación a través de la red

Actualmente la placa se conecta al equipo mediante un conector USB por el

que se produce la comunicación en serie.

Sería muy interesante para futuras revisiones el soporte de un mayor número de tipos de conexiones, y la conexión en red en un modelo cliente – servidor podría ser muy útil, ya que podría permitir a varios alumnos estudiar los datos de la misma planta piloto de forma simultánea desde equipos diferentes.

Soporte de un mayor número de idiomas

Actualmente tanto la interfaz como la aplicación web están disponibles en inglés y en español. Sería muy interesante, sobre todo para alumnos del programa Erasmus, soportar un mayor número de idiomas, pues les facilitaría el uso de la aplicación y mejoraría su experiencia de usuario.

Bibliografía

[1] F.C. Diez, "*Desarrollo de una interfaz para planta piloto*", tech.rep., Grado en Ingeniería Informática - Universidad de Burgos, 2019.

[2] R. Z. Rodríguez, "*Diseño, construcción y control de una planta de laboratorio para la regulación de temperatura y caudal de aire*", tech.rep., Grado de Electrónica Industrial y Automática - Universidad de Burgos, 2017.

[3] M.I.R. Izquierdo, "*Implementación de reguladores pids industriales y reguladores avanzados en microcontroladores*", tech.rep., Grado de Electrónica Industrial y Automática - Universidad de Burgos, 2018.

[4] Microsoft, "*Introducción a .NET Framework*", 2022. [Internet; descargado 19-junio-2022].

[5] Microsoft, *".NET Standard*", 2022. [Internet; descargado 19-junio-2022].

[6] Microsoft, "*Información general acerca de .NET Framework*", 2022. [Internet; descargado 19-junio-2022].

[7] Microsoft, "*Información general de ASP.NET*", 2022. [Internet;

descargado 19-junio-2022].

[8] Microsoft, "*Paseo por el lenguaje C#*", 2022. [Internet; descargado 19-junio-2022].

[9] Microsoft, "*IIS Express Overview*", 2022. [Internet; descargado 19-junio-2022].

[10] Oracle, "*¿Qué es una base de datos?*", 2019. [Internet; descargado 19-junio-2022].

[11] IBM, "*Comunicación serie*", 2021. [Internet; descargado 20-junio-2022].

[12] Antonio Díaz Estrella. *Teoría y diseño con microcontroladores de Freescale: familia Flexis de 32 bits MCF51QE*. McGraw-Hill Interamericana de España, 2008.

[13] Universitat d'Alacant, "*Modelo Vista Controlador (MVC)*", 2013. [Internet; descargado 24-junio-2022].

[14] David J. Anderson. *Kanban: Successful Evolutionary Change for Your Technology Business*. Blue Hole Press, 2010.

[15] Wikipedia, "*OPC*", 2019. [Internet; descargado 28-junio-2022].