UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Desarrollo de una interfaz para planta piloto Documentación Técnica

Presentado por Francisco Crespo Diez
en Universidad de Burgos — 13 de febrero de 2019
Tutores: Dr. Daniel Sarabia Ortiz - Dr. Alejandro Merino Gómez

Índice general

Índice de figuras

Índice de tablas

Apéndice A

Plan de Proyecto Software

A.1. Introducción

A lo largo de este apéndice se tratará todo aquello relacionado con la planificación del proyecto, considerándose esta un punto clave para cualquier desarrollo de *software*. En esta fase se estima tanto el dinero, como el trabajo y el tiempo que se va a emplear en completar el proyecto a través de un análisis minucioso de los recursos necesarios.

La fase de planificación se encuentra dividida en:

- Planificación temporal.
- Estudio de viabilidad.
  - Viabilidad económica.
  - Viabilidad legal.

A.2. Planificación temporal

En esta sección se elaborará un programa de tiempos en los que se estima la duración de cada una de las partes del proyecto. Partiendo del establecimiento de una fecha de inicio y una fecha de finalización estimada, a través tanto del peso de cada una de las tareas como de los requisitos necesarios para poder empezar cada una.

Al inicio del proyecto se decidió utilizar *Scrum* como metodología ágil para la gestión del proyecto. Debido a que el equipo formado no contaba

Story points	Estimación temporal
1	30’
2	2h
3	3h
5	5h
8	10h
13	18h
21	26h
40	50h

Tabla A.1: Equivalencia entre *story points* y tiempo.

con más de cuatro personas, no se ha podido seguir a rajatabla, pero sí que se han seguido las líneas generales de esta filosofía:

- Estrategia de desarrollo incremental a través de *sprints* (o iteraciones) y revisiones.
- La duración media de cada *sprint* era de dos semanas.
- Al finalizar cada *sprint* se realizaban reuniones en las que se revisaba el incremento en el proyecto y se planificaba el siguiente *sprint*.
- Tras la planificación del *sprint* se creaban una serie de tareas a realizar, las cuales eran estimadas y priorizadas en un tablero *canvas*.
- La monitorización del progreso del proyecto se llevó a cabo a través de gráficos *burndown*.

Cabe comentar que la estimación se realizó a través de *story points* (incluidos por ZenHub) que tienen una traducción temporal mostrada en la Tabla A.1.

Sprints llevados a cabo:

Sprint #0 (01/10/2018 - 14/10/2018)

A lo largo de este *sprint* pre-inicial se realizó una lectura e investigación sobre los diferentes Trabajos de Fin de Grado disponibles y se tuvo la primera toma de contacto con la plantilla LaTeXque se está utilizando para realizar

esta documentación. Aún no se controlaba bien la asignación de tareas ni la estimación de tiempos, como se puede ver en la Figura A.1 o en enlace a el *sprint* #0.

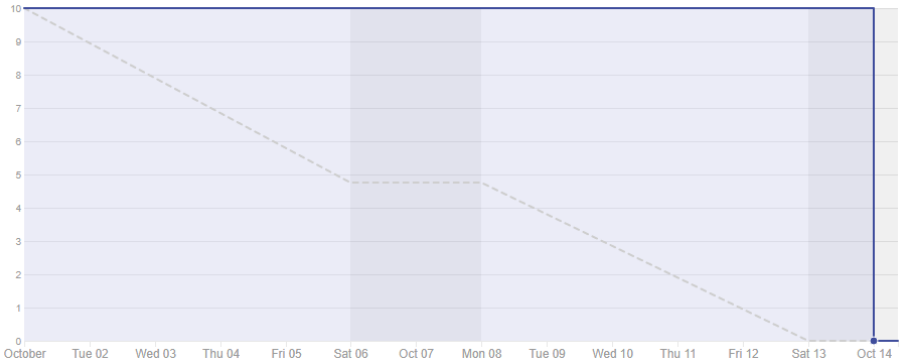

Figura A.1: *Sprint* #0.

Sprint #1 (14/10/2018 - 27/10/2018)

En este *sprint* acontece el primer contacto con los doctores Merino y Sarabia. Se comenzó con una presentación de los materiales que se iban a utilizar en el proyecto, expresando las primeras ideas y conjeturas que cada uno tenía sobre el mismo. También se definieron el tipo de repositorio a utilizar, así como la metodología y el formato de la documentación. Tras la reunión se decidió que la próxima semana se dedicaría a la investigación sobre el lenguaje a utilizar para el desarrollo del proyecto, buscando facilitar la comunicación serie con la placa NXP FRDM K64F. En la Figura A.2 se puede observar el avance sobre el *sprint* #1. A este *sprint* se dedicaron 13 *story points*, lo que se traduciría a 18 horas reales.

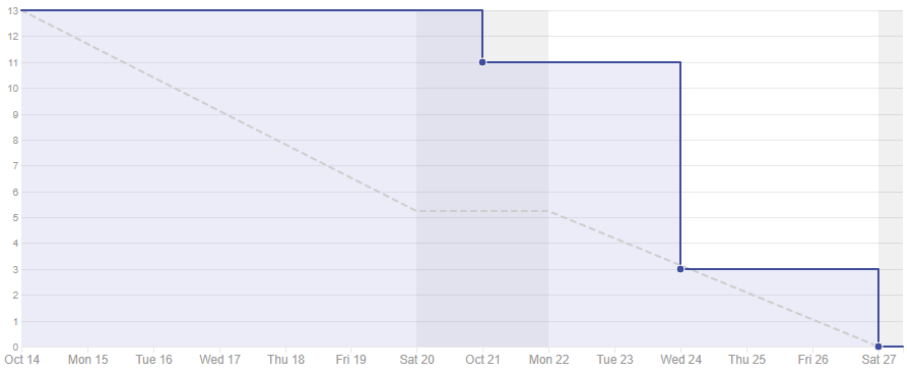

Figura A.2: *Sprint* #1.

Debido a las dudas que se presentaban al principio de este proyecto tuvo lugar una segunda reunión en la que se abordaron los siguientes temas:

- Préstamo de la placa FRDM K64F al estudiante.
- Conexión de la placa mediante puerto serie y visualización de los datos a través del programa Termite.
- Explicación del documento "Especificaciones Interfaz planta piloto.docx".
- Pasos siguientes:
  - Crear una aplicación sencilla que se comunique con la placa a través del puerto serie en C++.
  - Definir un formato del archivo de configuración, donde se guardan los valores de las variables que se van a utilizar en la placa. La aplicación no tiene porqué trabajar siempre con las mismas variables, depende de cómo se configure la placa, por lo que deberá permitir introducir las variables que el usuario considere y trabajar con ellas.
- Se decide utilizar Visual Studio 2015 como IDE por mejor usabilidad con C++ y Windows Form.

Sprint #2 (28/10/2018 - 10/11/2018)

Durante estas dos semanas, y debido a que la elección primera del lenguaje de desarrollo iba a ser C++, se realizaron una serie de tutoriales

sobre dicho lenguaje y sobre su uso en Visual Studio 2017. En la Figura A.3 se puede observar el avance sobre el *sprint* #2. A este *sprint* se dedicaron 82 *story points*, lo que se traduciría a 100 horas reales.

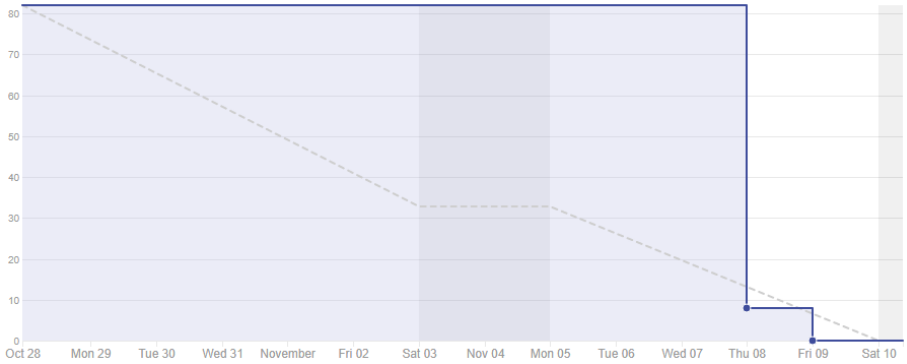

Figura A.3: *Sprint* #2.

Tras realizar varias investigaciones sobre la comunicación serie en otros lenguajes se tomó la decisión de cambiar de C++ a C# en el lenguaje en el que se iba a desarrollar la herramienta, puesto que ofrecía un gran abanico de facilidades y se contaba con mayor experiencia de uso en este segundo lenguaje.

Sprint #3 (10/11/2018 - 24/11/2018)

En este periodo se comenzó a programar de una manera más eficiente, consiguiendo notables avances sobre el código y a su vez, comunicando la aplicación con la placa NXP. A su vez, se diseñó un icono para personalizar la aplicación y el formato en el que las configuraciones que iban a ser cargadas en la aplicación se guardaban en un archivo de texto. En la Figura A.4 se puede observar el avance sobre el *sprint* #3. A este *sprint* se dedicaron 29 *story points*, lo que se traduciría a 36 horas reales.

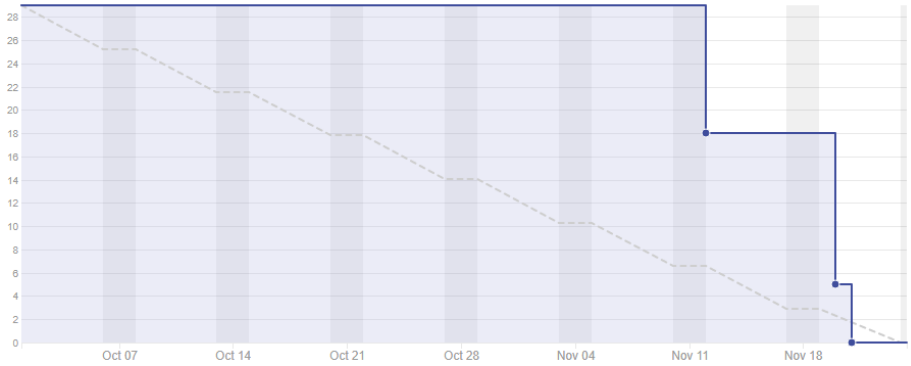

Figura A.4: *Sprint* #3.

Cabe añadir que a partir de este momento se tuvo presente la adición de varios idiomas a la aplicación para dotarla de mayor apertura cultural.

Sprint #4 (24/11/2018 - 08/12/2018)

A lo largo de este *sprint* se llevaron acabo diversas mejoras en la interfaz de la aplicación y en el tratamiento de datos recibidos a través del puerto serie. No obstante, se podría definir como la principal mejora la implementación de una base de datos, con su correspondiente investigación previa, y la conexión de la misma a Visual Studio y al proyecto.

El método utilizado para la conexión fue creado a partir de un asistente del IDE mencionado, el cual facilitó esta implementación, pero a su vez creó el mayor quebradero de cabeza que ha aparecido durante todo el desarrollo. En el momento de querer ejecutar la aplicación en otro ordenador, y, debido a haber creado la conexión de manera local en mi equipo, la aplicación no conseguía funcionar correctamente. Este problema continuó hasta el último *sprint*. En la Figura A.5 se puede observar el avance sobre el *sprint* #4. A este *sprint* se dedicaron 44 *story points*, lo que se traduciría a 55 horas reales.

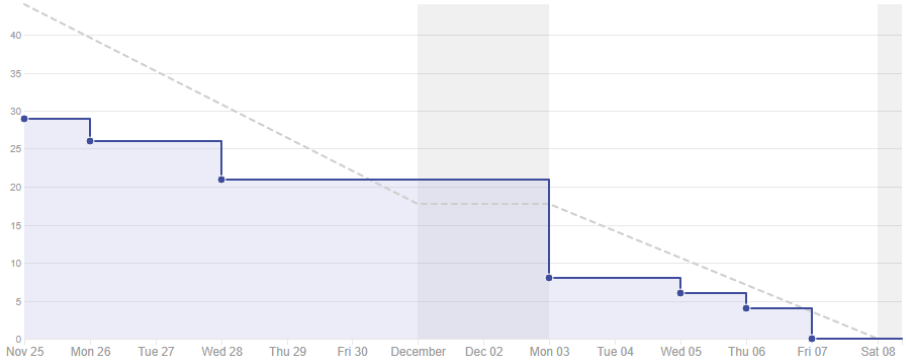

Figura A.5: *Sprint* #4.

Sprint #5 (08/12/2018 - 22/12/2018)

Durante este *sprint* se mostró a los tutores una primera versión de la interfaz de la aplicación, quienes reportaron una serie de mejoras que fueron añadidas al *sprint* que estamos tratando, como podrían ser el archivo de ayuda en la aplicación, etiquetas faltantes o funcionalidades no tenidas en cuenta (por ejemplo, que el usuario pudiera definir la cantidad de datos mostrados en la gráfica en tiempo real). A lo largo de este periodo se libera la primera *pre-release V0.0* de la aplicación. En la Figura A.6 se puede observar el avance sobre el *sprint* #5. A este *sprint* se dedicaron 103 *story points*, lo que se traduciría a 128 horas reales.

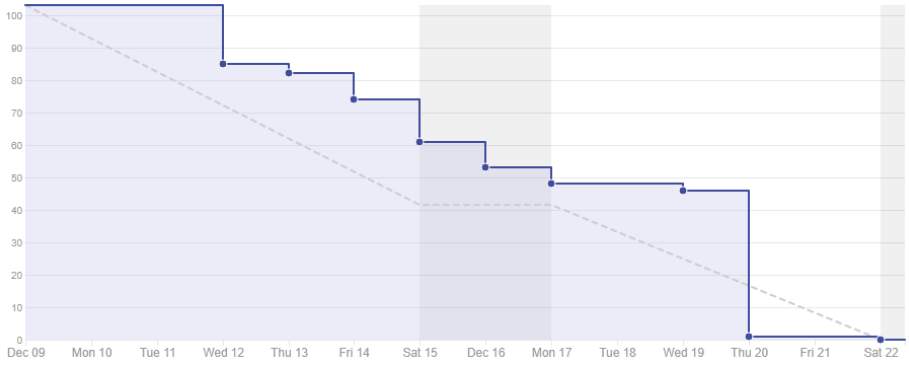

Figura A.6: *Sprint* #5.

Sprint #6 (22/12/2018 - 05/01/2019)

En este *sprint* se lleva a cabo una reunión en la que se incluye al Dr. López Nozal, quien aporta nuevas ideas para mejorar tanto el código como la gestión de proyecto, como por ejemplo, la inclusión de la herramienta Codacy en el mismo. También se plantea la posibilidad de añadir un archivo de log a la aplicación, haciendo más fácil la detección de errores. En la Figura A.7 se puede observar el avance sobre el *sprint* #6. A este *sprint* se dedicaron 54 *story points*, lo que se traduciría a 68 horas reales.

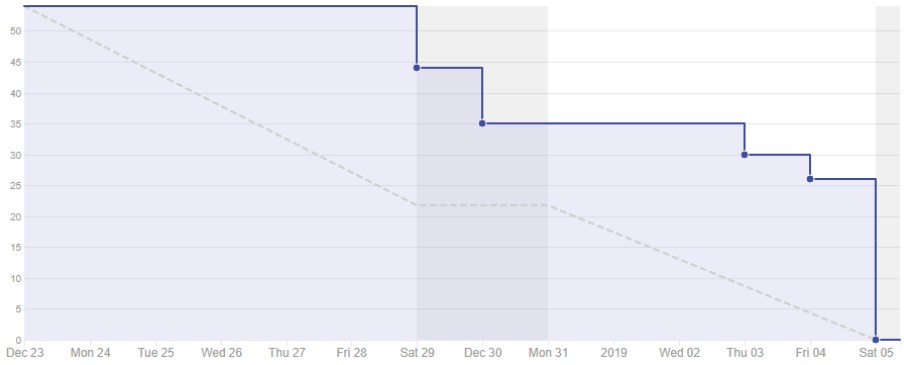

Figura A.7: *Sprint* #6.

En este momento se vaticina cuáles van a ser las funciones que se va a poder cubrir a lo largo de este proyecto y comienzan a barajarse ideas de trabajo futuras.

Sprint #7 (05/01/2019 - 19/01/2019)

A lo largo de este *sprint* y tras reunirme con los tutores y evaluar la *pre-release V0.1* se definen los siguientes pasos a seguir en el proyecto. La mayor parte del tiempo invertido en este *sprint* está dedicado a resolver las puntualizaciones que los tutores comentaron sobre dicha *pre-release*, y que se puede encontrar en el *issue* 57. En la Figura A.8 se puede observar el avance sobre el *sprint* #7. A este *sprint* se dedicaron 28 *story points*, lo que se traduciría a 34 horas reales.

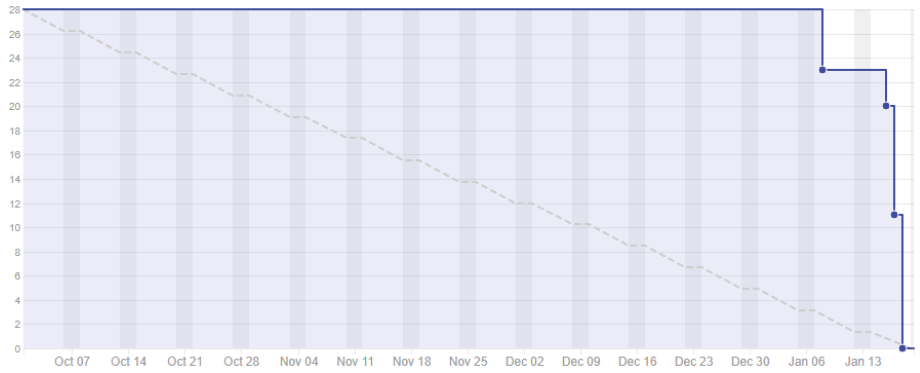

Figura A.8: *Sprint* #7.

Sprint #8 (19/01/2019 - 02/02/2019)

Durante este *sprint* se trabaja sobre la mejora de pequeños detalles de la aplicación, así como el problema que se comentó en el *sprint #4* sobre los problemas que se encontraban al ejecutar la aplicación en otro ordenador. Tras una ardua investigación se consiguió disipar la duda y solventar dicho error, pudiendo así testear la aplicación sin problema. En consecuencia a esta investigación se liberaron diferentes *releases* en las que se añadió la creación de la base de datos desde código y a partir de las cuales se pudieran hacer pruebas. En la Figura A.9 se puede observar el avance sobre el *sprint* #8. A este *sprint* se dedicaron 38 *story points*, lo que se traduciría a 48 horas reales.

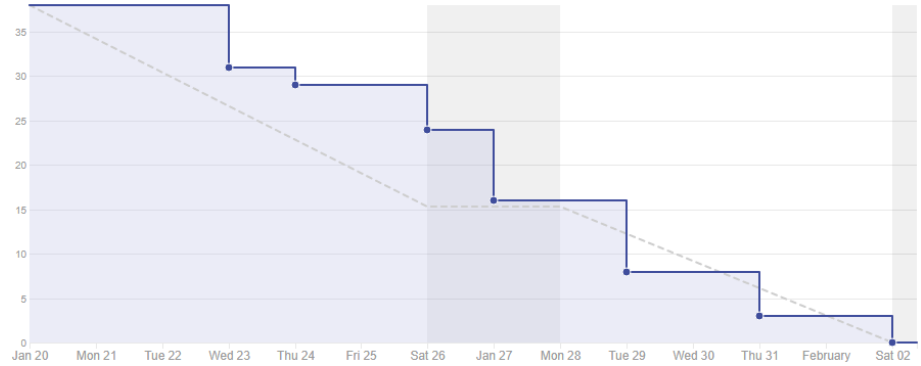

Figura A.9: *Sprint* #8.

Sprint #9 (02/02/2019 - 14/02/2019)

Este *sprint* se ha dedicado en su totalidad a la documentación del proyecto y a preparar los archivos y ejecutables necesarios para concluir el desarrollo del proyecto. En la Figura A.10 se puede observar el avance sobre el *sprint* #9. A este *sprint* se dedicaron 80 *story points*, lo que se traduciría a 100 horas reales.

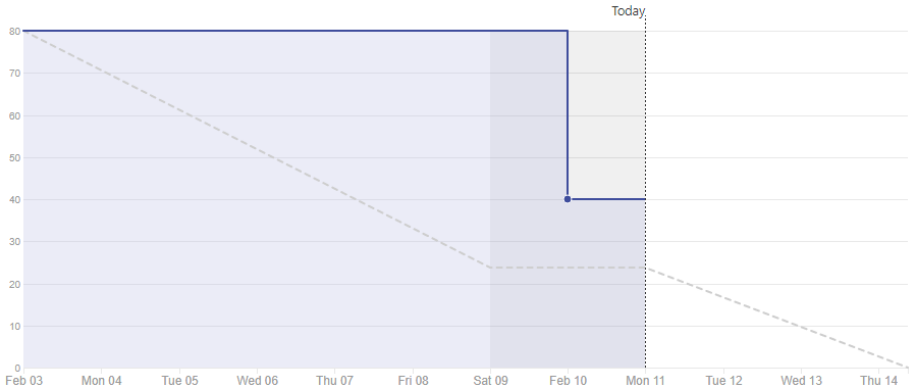

Figura A.10: *Sprint* #9.

Seguimiento de velocidad

Gracias a la herramienta ZenHub se puede obtener un gráfico con el que se informa de la velocidad que se ha tenido en cada *sprint* del proyecto, aportándonos una media de trabajo y una imagen global de nuestros avances, como podemos ver en la Figura A.11.

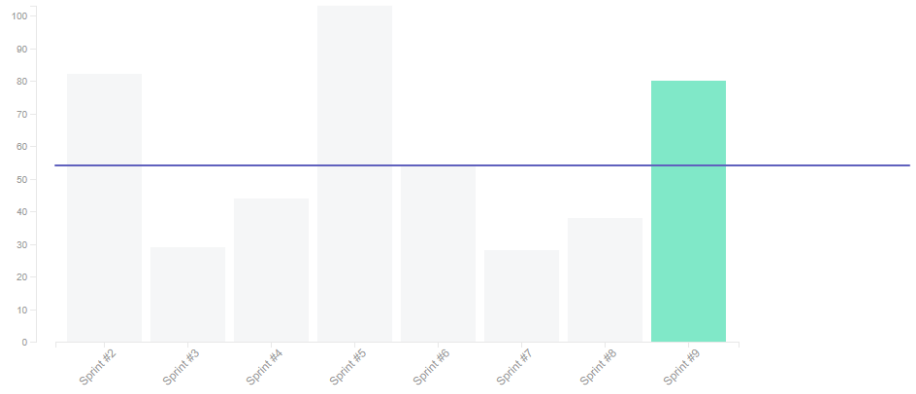

Figura A.11: Seguimiento de velocidad.

A.3. Estudio de viabilidad

Viabilidad económica

En el siguiente apartado se analizarán los costes y beneficios que podría haber supuesto el proyecto en el caso de que se hubiese realizado en un entorno empresarial real.

Coste de personal

El proyecto se lleva a cabo por un desarrollador junior empleado a tiempo parcial (30 horas semanales) durante cuatro meses, ergo se considera el siguiente salario [**?**]:

Concepto	Coste
Salario neto	1000.00€
Retención IRPF (19 %)	360.53€
Seguridad social (28,30 %)	537.00€
Salario bruto	1897.53€
Total	**5692.59€**

Tabla A.2: Coste de personal.

Costes de material

El material utilizado para el proyecto ha sido un ordenador valorado en 700€, del cual se pondrá el coste amortizado a cuatro años, y una placa NXP FRDM K64F valorada en 50€:

Concepto	Coste (€)
Ordenador	58.33€
NXP FRDM K64F	50.00€
Total	**108.33€**

Tabla A.3: Coste de material.

Costes de *software*

El software utilizado para el proyecto, aplicando la amortización por el tiempo usado, ha sido el siguiente:

Concepto	Coste (€)
Microsoft Windows 10 Pro	21.58€
Microsoft Visual Studio 2017	160.00€
Microsoft SQL Server	75.00€
Total	**256.58€**

Tabla A.4: Coste de *software*.

Costes totales

El sumatorio de todos los costes ha sido el siguiente:

Concepto	Coste (€)
Personal	5692.59€
Materiales	108.33€
Software	256.98€
Total	**6057.09€**

Tabla A.5: Coste totales.

Beneficios

Debido a que la finalidad de esta aplicación es la utilización por parte de los alumnos de Grado en Ingeniería Electrónica y Automática de la Universidad de Burgos, no se considera en ningún momento obtener beneficio alguno, siempre y cuando la institución no quiera otorgar cierta cuantía por los servicios prestado, en cuyo caso se aceptarán gustosos.

Viabilidad legal

En este apartado se realiza un estudio sobre las leyes vigentes que puedan afectar al proyecto desarrollado y cómo se solucionarán los posibles problemas que surjan.

Gracias a haber desarrollado toda la aplicación con software de Microsoft, del cual ya se han pagado las licencias pertinentes, no es necesario atender a la utilización de más licencias de terceros.

Añadiendo a lo anterior, cabe mencionar que al tratarse de una aplicación con finalidad no comercial, las restricciones sobre el software creado son mucho menores.

Concluyendo, la licencia que asignaré a esta aplicación será una licencia Creative Commons "Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional", como se puede ver en la Figura A.12.

Figura A.12: Licencia Creative Commons.

Apéndice B

Especificación de Requisitos

B.1. Introducción

Este anexo recoge la especificación de requisitos que define el comportamiento del sistema desarrollado. De esta manera, se consigue tener la documentación correspondiente al análisis de la aplicación y un documento contractual entre el cliente y el equipo de desarrollo.

Para llevar a cabo esta tarea se han seguido las recomendaciones del estándar IEEE 830-1998 sobre una buena especificación de requisitos [**?**]:

- Consistente.
- Completa.
- inequívoca.
- Trazable.
- Correcta.
- Modificable.
- Priorizable.
- Verificable.

B.2. Objetivos generales

- Crear una interfaz hombre-máquina amigable que permita la monitorización y modificación de los datos gestionados por una placa NXP.
- Ayudar a los nuevos estudiantes de la Universidad de Burgos en su comprensión del funcionamiento de una una planta piloto y los sistemas de control necesarios para controlarla. Fundamentalmente orientado para las prácticas de las asignaturas de control en el grado de Electrónica Industrial y Automática, el grado de ingeniería Informática y el máster de Ingeniería Industrial.
- Aportar información extra a los datos recibidos por la placa que ayude a la comprensión de los cambios experimentados en la planta piloto.
- Dar la posibilidad de almacenar los datos recibidos de la planta piloto de una manera concisa, estructurada y de fácil acceso.

B.3. Catalogo de requisitos

Requisitos funcionales

- **RF-1** El usuario debe ser capaz de trabajar con distintos proyectos en la aplicación.
  - **RF-1.1** El usuario debe ser capaz de crear un nuevo proyecto desde la aplicación.
  - **RF-1.2** El usuario debe ser capaz de cargar un proyecto existente desde la aplicación.
  - **RF-1.3** El usuario debe ser capaz de modificar un proyecto cargado en la aplicación.
- **RF-2** El usuario debe ser capaz de modificar el estado de la conexión con el puerto serie.
  - **RF-2.1** El usuario debe ser capaz de abrir una comunicación con el puerto serie desde la aplicación.
  - **RF-2.2** El usuario debe ser capaz de cerrar una comunicación con el puerto serie desde la aplicación.
- **RF-3** El usuario debe ser capaz de modificar los valores de las variables de escritura que tenga el proyecto.

- **RF-4** El usuario debe ser capaz de acceder a los datos devueltos por la placa NXP y almacenados en la base de datos.
    - **RF-4.1** El usuario debe ser capaz de visualizar los datos almacenados en la base de datos en una gráfica.
    - **RF-4.2** El usuario debe ser capaz de visualizar los datos almacenados en la base de datos en una nueva ventana.
    - **RF-4.3** El usuario debe ser capaz de guardar los datos almacenados en la base de datos en un archivo de texto.

Requisitos no funcionales

- **RNF-1** La interfaz debe ser sencilla e intuitiva.
- **RNF-2** La aplicación debe gestionar las posibles excepciones que acontezcan durante la ejecución de la misma, mostrando al usuario las que sean pertinentes.
- **RNF-3** La aplicación debe ser mantenible, facilitando la escalabilidad a la hora de añadir características.
- **RNF-4** La aplicación debe ser estar preparada para soportar varios idiomas y que el usuario pueda cambiar entre ellos.
- **RNF-5** La aplicación debe poder ser desplegable con facilidad y en poco tiempo.
- **RNF-6** El usuario debe ser capaz de acceder a la ayuda de la aplicación desde cualquier ventana.

B.4. Especificación de requisitos

En esta sección se desarrollarán los distintos casos de uso, los cuales cubren los requisitos funcionales presentados en la sección anterior, como se puede ver en la Figura B.1.

Figura B.1: Diagrama Casos de Uso.

CU-01	**Crear proyecto**
Autor	Francisco Crespo Diez
Requisitos asociados	RF-1.1
Descripción	Permite al usuario crear un proyecto y cargarlo

continúa en la página siguiente

continúa desde la página anterior

CU-01	**Crear proyecto**
	en la aplicación.
Precondiciones	Ninguna
Acciones	El usuario rellena el formulario con los datos del proyecto.
Postcondición	El proyecto se carga en la aplicación. Si no existe, se crea la base de datos con el nombre TFG_DB. Si no existe, se crea la tabla cuyo nombre será el nombre del proyecto. En caso de existir dicha tabla se sobrescribe.
Excepciones	Fallo al crear la base de datos o la tabla por falta de archivos de configuración de la base de datos.
Importancia	Alta.

Tabla B.1: CU-01 Crear proyecto

CU-02	**Cargar proyecto**
Autor	Francisco Crespo Diez
Requisitos asociados	RF-1.2
Descripción	Permite al usuario cargar un proyecto ya creado en la aplicación
Precondiciones	Ninguna
Acciones	El usuario selecciona el archivo .txt con la configuración del proyecto.
Postcondición	El proyecto se carga en la aplicación. Si no existe, se crea la base de datos con el nombre TFG_DB. Si no existe, se crea la tabla cuyo nombre será el nombre del proyecto. En caso de existir dicha tabla

continúa en la página siguiente

continúa desde la página anterior

CU-02	**Cargar proyecto**
	se sobrescribe.
Excepciones	Fallo al crear la base de datos o la tabla por falta de archivos de configuración de la base de datos.
Importancia	Alta

Tabla B.2: CU-02 Cargar proyecto

CU-03	**Modificar proyecto cargado**
Autor	Francisco Crespo Diez
Requisitos asociados	RF-1.3
Descripción	Permite al usuario modificar un proyecto cargado en la aplicación.
Precondiciones	Debe haber un proyecto cargado en la aplicación.
Acciones	El usuario selecciona el archivo .txt con la configuración del proyecto.
Postcondición	El proyecto se carga en la aplicación. Si no existe, se crea la base de datos con el nombre TFG_DB. Si no existe, se crea la tabla cuyo nombre será el nombre del proyecto. En caso de existir dicha tabla se sobrescribe.
Excepciones	Fallo al crear la base de datos o la tabla por falta de archivos de configuración de la base de datos.
Importancia	Alta

Tabla B.3: CU-03 Modificación de proyecto cargado

CU-04	**Iniciar la comunicación con el puerto serie**
Autor	Francisco Crespo Diez

continúa en la página siguiente

continúa desde la página anterior

CU-04	**Iniciar la comunicación con el puerto serie**
Requisitos asociados	RF-2.1
Descripción	Permite al usuario iniciar la comunicación con el puerto serie a través de la aplicación.
Precondiciones	Debe haber un proyecto cargado en la aplicación cuyas variables existan en la placa NXP y una conexión con un puerto serie disponible.
Acciones	El usuario inicia la comunicación a través del botón “Inicio".
Postcondición	La aplicación comienza la comunicación con el puerto serie. Se comienzan a guardar datos en la tabla creada. Las funcionalidades “Gráfica", “Variablesz “Archivo"pasan a estar disponibles. CU-03 deja de estar disponible.
Excepciones	Fallo al crear la tabla, requiere reinicio de la comunicación.
Importancia	Alta

Tabla B.4: CU-04 Iniciar la comunicación con el puerto serie

CU-05	**Detener la comunicación con el puerto serie**
Autor	Francisco Crespo Diez
Requisitos asociados	RF-2.2
Descripción	Permite al usuario detener la comunicación con el puerto serie a través de la aplicación.
Precondiciones	Debe haber un proyecto cargado en la aplicación y una conexión con un puerto serie disponible con el que la aplicación tenga la comunicación abierta.
Acciones	El usuario detiene la comunicación a través del botón “Fin".

continúa en la página siguiente

continúa desde la página anterior

CU-05	**Detener la comunicación con el puerto serie**
Postcondición	La aplicación detiene la comunicación con el puerto serie. Se dejan de guardar datos en la tabla creada. Las funcionalidades “Gráfica", “Variablesz “Archivo"dejan de estar disponibles. CU-03 pasa a estar disponible.
Excepciones	Fallo al crear la tabla, requiere reinicio de la comunicación.
Importancia	Alta

Tabla B.5: CU-05 Detener la comunicación con el puerto serie

CU-06	**Modificar el valor de las variables cargadas**
Autor	Francisco Crespo Diez
Requisitos asociados	RF-3
Descripción	Permite al usuario modificar el valor de la variable en la placa desde la aplicación.
Precondiciones	Debe haber un proyecto cargado en la aplicación. La comunicación a través del puerto serie debe estar abierta.
Acciones	El usuario introduce el valor que quiere asignar a la variable.
Postcondición	Si el valor introducido es validado, en caso de que así sea, se envía a la placa. La placa actualiza su valor.
Excepciones	El envío no llega a su destino.
Importancia	Alta

Tabla B.6: CU-06 Modificar el valor de las variables cargadas

CU-07	**Mostrar valores graficados**

continúa en la página siguiente

continúa desde la página anterior

CU-07	**Mostrar valores graficados**
Autor	Francisco Crespo Diez
Requisitos asociados	RF-4.1
Descripción	Permite al usuario ver los valores de las variables seleccionadas en una gráfica.
Precondiciones	Debe haber un proyecto cargado en la aplicación. La comunicación a través del puerto serie debe estar abierta.
Acciones	El usuario selecciona las variables que quiere graficar en la ventana “Selección de variables" y pulsa el botón “Crear gráfica".
Postcondición	Se abrirá una nueva ventana en la que irán apareciendo los últimos valores guardados en BD. La cantidad de valores mostrados se puede modificar.
Excepciones	La tabla no tiene valores.
Importancia	Media

Tabla B.7: CU-07 Mostrar valores graficados

CU-08	**Mostrar valores**
Autor	Francisco Crespo Diez
Requisitos asociados	RF-4.2
Descripción	Permite al usuario ver los valores de las variables seleccionadas en una ventana.
Precondiciones	Debe haber un proyecto cargado en la aplicación. La comunicación a través del puerto serie debe estar abierta.

continúa en la página siguiente

continúa desde la página anterior

CU-08	**Mostrar valores**
Acciones	El usuario selecciona las variables que quiere mostrar en la ventana “Selección de variables" y pulsa el botón “Mostrar valores".
Postcondición	Se abrirá una nueva ventana en la que aparecen los últimos valores de las variables seleccionadas guardados en BD.
Excepciones	La tabla no tiene valores.
Importancia	Media

Tabla B.8: CU-08 Mostrar valores

CU-09	**Guardar valores en archivo**
Autor	Francisco Crespo Diez
Requisitos asociados	RF-4.3
Descripción	Permite al usuario guardar los valores de las variables seleccionadas en un archivo de texto.
Precondiciones	Debe haber un proyecto cargado en la aplicación. La comunicación a través del puerto serie debe estar abierta.
Acciones	El usuario selecciona las variables que quiere mostrar en la ventana “Selección de variables" y pulsa el botón “Crear archivo".
Postcondición	Se volverá a la ventana principal y botón “Archivo"habrá cambiado a “Detener guardado".
Excepciones	La tabla no tiene valores.
Importancia	Media

Tabla B.9: CU-09 Guardar valores en archivo

Apéndice C

Especificación de diseño

C.1. Introducción

A lo largo de este apéndice se resuelven los objetivos y especificaciones expuestos con anterioridad. Se definen los datos que va a utilizar la aplicación, el diseño de interfaces, detalles procedimentales, arquitectura, etc.

C.2. Diseño de datos

La aplicación no cuenta con una entidad fija en la que se guarden los datos, sino que según las variables que contenga el proyecto cargado se crearán los campos correspondientes en la tabla. En la Figura C.1 se puede ver un ejemplo.

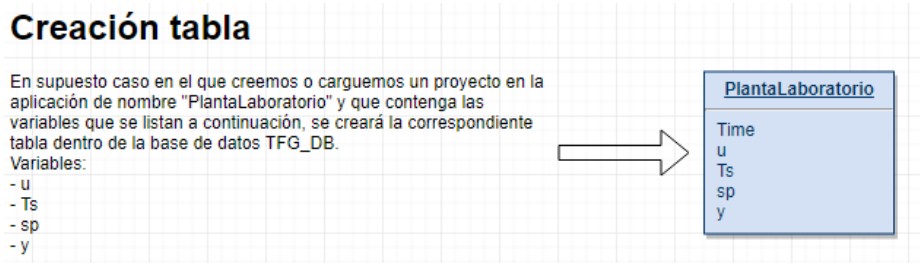

Figura C.1: Ejemplo construcción de una tabla.

Lo que se consigue de esta manera es un almacenamiento temporal de los datos transmitidos durante la comunicación, como se indicaba en el documento funcional.

C.3. Diseño procedimental

En este apartado se muestran los detalles más relevantes de la ejecución de la aplicación conectada a la placa NXP FRDM K64F y a SQL Server.

En el diagrama de secuencia que se muestra en la Figura C.2 se puede apreciar las diferentes interaciones entre los distintos objetos que componen la aplicación para la monitorización y guardado de los datos recibidos desde la placa NXP.

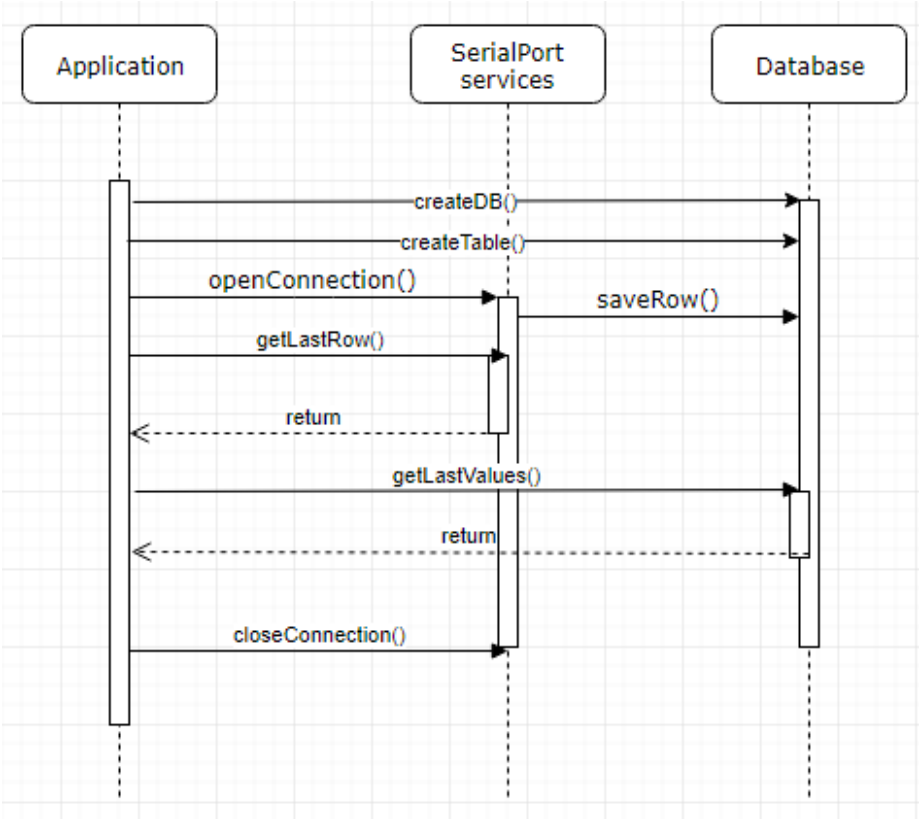

Figura C.2: Diagrama de secuencia de la aplicación.

C.4. Diseño arquitectónico

En esta sección se expondrá todo aquello relacionado con el diseño de la arquitectura que sigue la aplicación.

Modelo - Vista - Controlador (MVC)

En la búsqueda de facilitar la escalabilidad de la herramienta se ha optado por utilizar la arquitectura Modelo Vista Controlador o MVC. A través de esta arquitectura se pretende separar la vista con la que interacionará el usuario, con la lógica de la aplicación, como se muestra en la Figura C.3 [**?**].

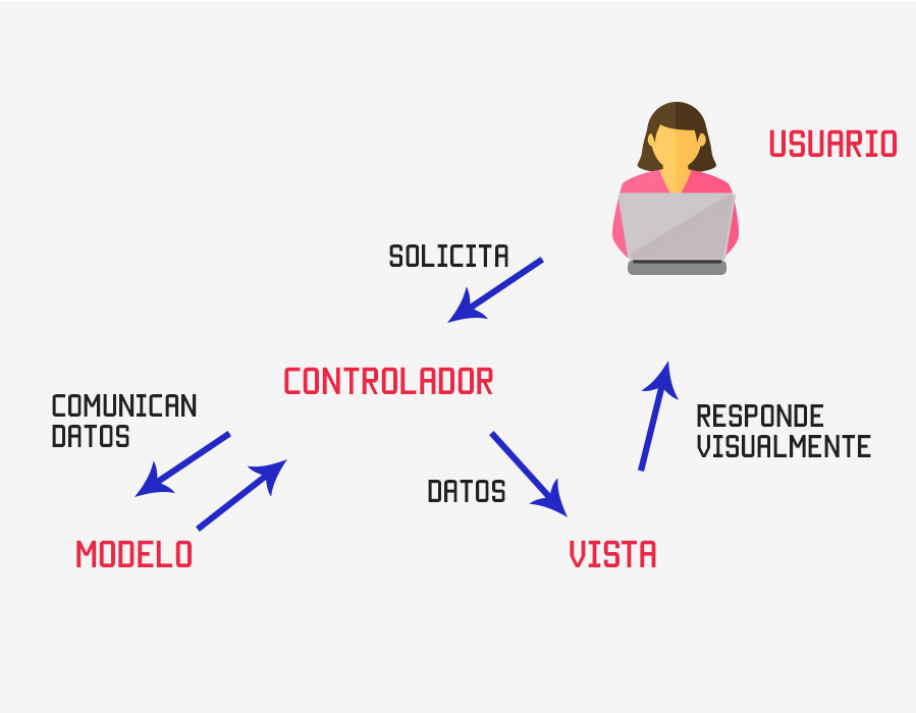

Figura C.3: Modelo - Vista - Controlador.

Traducido a este proyecto:

- El controlador correspondería a la aplicación, la cual se encarga de abrir las comunicaciones e invocar los métodos necesarios.
- La vista correspondería con las ventanas con las que el usuario interactua.
- El modelo correspondería a los datos guardados en la base de datos que se van obteniendo a lo largo del tiempo.

C.5. Diseño de paquetes

A la hora de organizar los archivos que componen la aplicación no se ha seguido la estrategia convencional de paquete por capa (*package by layer approach*), sino que se usó una estrategia de paquete por característica (*package per feature approach*).

En la Figura C.4 se puede apreciar la organización de los archivos según sus distintas funcionalidades. La toma de esta decisión se vio motivada por facilitar la legibilidad y comprensión del proyecto.

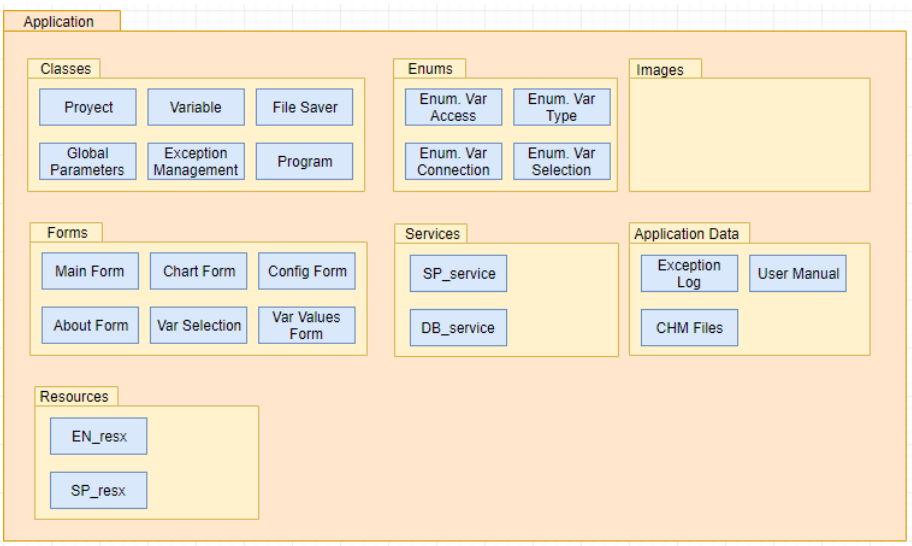

Figura C.4: Diagrama de paquetes.

C.6. Diseño de clases

Para conseguir la funcionalidad deseada en la aplicación se han construido una serie de clases cuya relación entre ellas puede verse en la Figura C.5.

- **ExceptionManagement:** clase encargada de controlar las excepciones que acontecen en la ejecución de la aplicación y guardarlas en un archivo de log.
- **FileSaver:** clase que contiene el método para crear la cabecera de un archivo de texto en el que se guarde tanto el proyecto como los valores de las variables seleccionadas.

- **Program:** clase que arranca la ejecución del programa.
- **Proyect:** clase que define las propiedades que va a tener un proyecto.
- **Variable:** clase que define las propiedades que va a tener una variable.

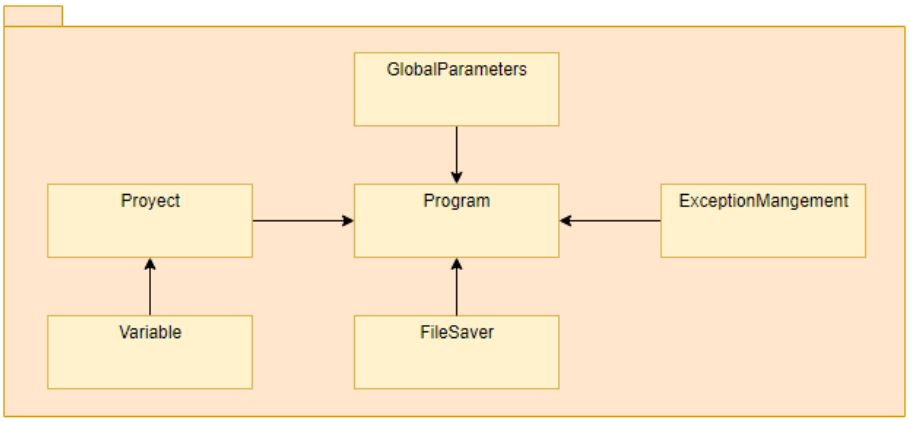

Figura C.5: Diagrama de clases.

Apéndice D

Documentación técnica de programación

D.1. Introducción

En este apéndice se describirá toda aquella documentación técnica relacionada con la programación de la aplicación, incluyendo la estructura de directorios, la instalación en el entorno de desarrollo y un manual para el programador.

D.2. Estructura de directorios

El contenido del repositorio del proyecto es el siguiente:

- `/`: es el directorio raíz, contiene el archivo de licencia, el archivo README.md, el archivo que ha ido guardando un resumen de las reuniones que se han tenido con los tutores y el archivo de dudas en el que se anotaban las dudas para preguntar a los tutores.
- `/Memoria`: este directorio contiene todo lo referente a la documentación que está leyendo. Todos los archivos LaTeX, imágenes, etc. están aquí recogidos.
- `/PlantaPiloto`: es el directorio que recoge la aplicación. Dentro está el archivo .sln (solución del proyecto) y la carpeta con todos los archivos.
- `/PlantaPiloto/PlantaPiloto`: contiene todas las carpetas en las que el proyecto está dividido.

- `/PlantaPiloto/PlantaPiloto/ApplicationData`: directorio que contiene todos los archivos necesarios durante la ejecución de la aplicación, como por ejemplo el archivo de log o el manual de usuario.
- `/PlantaPiloto/PlantaPiloto/Classes`: contiene las clases explicadas en el apéndice C-Diseño de clases.
- `/PlantaPiloto/PlantaPiloto/Enums`: contiene los enumerables que van a ser usados por la aplicación.
- `/PlantaPiloto/PlantaPiloto/Forms`: contiene todos los formularios que aparecen en la aplicación.
- `/PlantaPiloto/PlantaPiloto/Properties`: contiene las propiedades del proyecto.
- `/PlantaPiloto/PlantaPiloto/Resources`: contiene las fuentes del proyecto (diferentes idiomas).
- `/PlantaPiloto/PlantaPiloto/Services`: contiene los servicios que actúan contra la base de datos y contra la comunicación por el puerto serie.
- `/PlantaPiloto/PlantaPiloto/images`: directorio que contiene las imágenes que se utilizan en la aplicación.

D.3. Manual del programador

En esta sección se explicará el contenido del proyecto de una manera en la que el lector sea un programador que quiera trabajar sobre la aplicación.

Entorno de desarrollo

A continuación se expondrán el *software* necesario con los enlaces para su descarga:

- .NET Framework 4.6.1. Será necesario para tener todas las librerías necesarias para la ejecución de la aplicación. Si la versión es superior también valdría.
- SQL Server 17. Será utilizado para guardar los datos durante la ejecución de la aplicación. Es importante configurar el nombre de la conexión como *"localhost/sqlexpress"*, sino habría que cambiar las cadenas de conexión dentro de la clase DB_services.

- Git. Será necesario para la descarga de la última versión del proyecto.
- Visual Studio 2017. Es el IDE recomendado para trabajar con el proyecto.

Descarga del código fuente

Para descargar el código fuente se disponen de varios métodos. El primero de ellos desde Git Bash:

1. Abrir Git Bash.
2. Desplazarse al directorio donde se desee copiar el repositorio con el comando `cd`.
3. Clonar el repositorio con el comando:
   `git clone https://github.com/FranBurgos/TFG.git`

El segundo es directamente desde GitHub donde se aloja el repositorio como se muestra en la Figura D.1.

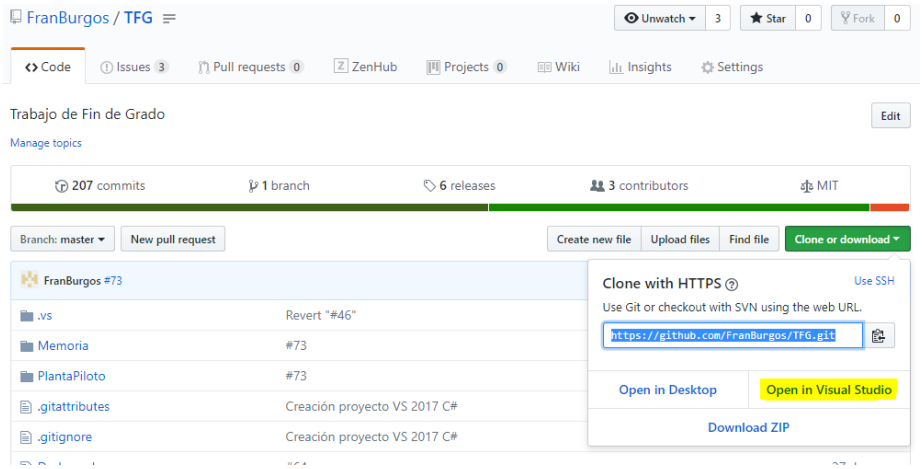

Figura D.1: Descarga de repositorio desde GitHub.

Esto nos cargaría directamente el proyecto en Visual Studio y crearía los directorios necesarios una vez elijamos el destino del proyecto.

Y en tercer lugar, desde el IDE de Visual Studio, en la ventana de Team Explorer podemos clonar directamente el repositorio indicando el directorio donde queremos que se descargue el proyecto, como se muestra en la imagen D.2.

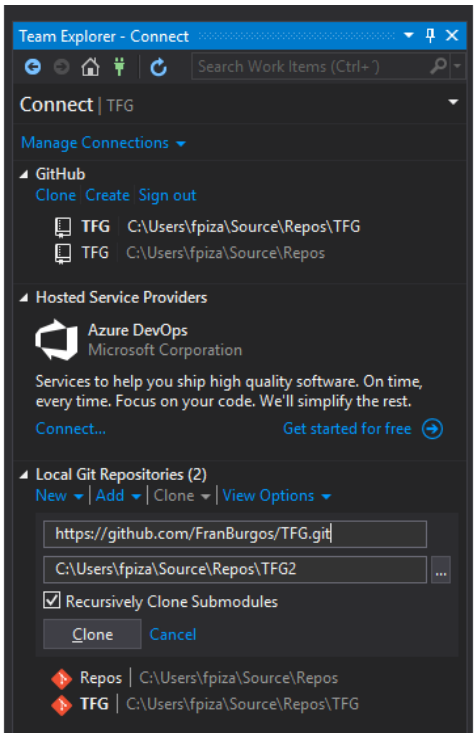

Figura D.2: Descarga de repositorio desde Visual Studio 2017.

Se ruega que se cree a través de Git una nueva rama para los cambios que se quieran llevar a cabo sobre el proyecto.

D.4. Compilación, instalación y ejecución del proyecto

Una vez que se tenga cargado el proyecto en Visual Studio 2017 la compilación y ejecución del proyecto se realizará a través del botón “Start"que se muestra en la Figura D.3.

Figura D.3: Botón de inicio de la compilación del proyecto.

En el caso de que se quiera descargar la última versión de la aplicación

para ejecutarla, se deberá ir al apartado de "Release.[en] el repositorio GitHub y descargar la más reciente. Al descomprimir los archivos descargados se tendrá lo suficiente para ejecutar la aplicación.

D.5. Pruebas del sistema

Las pruebas sobre la aplicación se han realizado de manera manual por lo que futuros desarrolladores deberán seguir este método o incluir algún otro más eficiente.

En cuanto a las pruebas sobre el código y su calidad, se utiliza la herramienta Codacy, la cual en cada *commit* comprueba el estado del proyecto.

Apéndice E

Documentación de usuario

E.1. Introducción

E.2. Requisitos de usuarios

E.3. Instalación

E.4. Manual del usuario