Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención en Tecnologías de la Información

Herramienta para aplicar el proceso ETL a datos de la AEMET y prueba de posibles aplicaciones sobre los datos

Autor:
Enrique García Miravalles

01/07/2020

Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática
Mención en Tecnologías de la Información

Herramienta para aplicar el proceso ETL a datos de la AEMET y prueba de posibles aplicaciones sobre los datos

Autor:
Enrique García Miravalles

Tutor:
Quiliano Isaac Moro Sancho

01/07/2020

Agradecimientos

Agradecer a mi familia y especialmente a mis padres todo el esfuerzo que han realizado para poder darme la mejor educación posible y enseñarme que sin esfuerzo y sacrificio no hay recompensa.

A las experiencias vividas y todo lo que me ha ofrecido mi estancia en la universidad.

A D. Quiliano Isaac por el tiempo que me ha dedicado durante la realización de este Trabajo de Fin de Grado.

Resumen

En la actualidad es difícil encontrar un sitio web que nos permita visualizar un histórico de datos meteorológicos y más difícil aún es tener acceso a estos datos para poder trabajar con ellos libremente.

En el presente TFG se ha desarrollado una herramienta para realizar el proceso ETL de datos meteorológicos de la web de la AEMET consiguiendo obtener datos meteorológicos de diversas estaciones para centralizarlos en un único espacio de almacenamiento.

Como ejemplos de aplicaciones que se pueden realizar sobre los datos meteorológicos almacenados, se ha creado una aplicación con carácter didáctico para la visualización de los datos meteorológicos y predicción de temperaturas máximas diarias, además de conectar una herramienta de visualización externa para mostrar otra forma de trabajo con los datos meteorológicos almacenados.

Palabras clave

Proceso ETL, metodologías ágiles, modelo de predicción, máquina virtual, herramienta de visualización, interfaz gráfica.

Abstract

Nowadays it is hard to find a website that allows us to visualize a historical meteorological data and even more difficult to have access to this meteorological data to work freely with them.

In this Bachellor's Thesis a tool has been developed to perform the ETL process of meteorological data obtained from the AEMET website. These meteorological data have been obtained from different stations and centralized in a single storage space.

As an example of the possibilities of this work, we will use the stored meteorological data to present several visualisation types and predict maximum daily temperatures.

As another example of working with the data, an external visualization tool will be connected to show another the flexibility of the tool.

Keywords

ETL process, agile methodology, prediction model, virtual machine, visualization tool, graphic interface.

Índice general

Índice de figuras

Índice de tablas

1. Introducción

El presente documento se trata del Trabajo de Fin de Grado (TFG) necesario para finalizar el plan de estudios del grado de Ingeniería Informática cursado en la Escuela de Ingeniería Informática de Valladolid.

Por definición el TFG consiste en la realización de un proyecto por parte de un alumno bajo la supervisión de un tutor académico, en el cual se demostrarán los conocimientos y competencias adquiridas durante el grado cursado.

Debido a la dificultad para encontrar portales o sitios web que permitan mostrar datos meteorológicos que podríamos considerar antiguos, una herramienta que permita crear un histórico de datos atmosféricos que en un futuro pueda ser usado para, por ejemplo, comprobar la temperatura de un municipio en concreto en una fecha concreta del pasado o para comprobar las tendencias climáticas o variaciones meteorológicas en ciertas fechas, podría ser de gran utilidad.

En el presente documento se explicarán los procedimientos usados para poder desarrollar una herramienta capaz aplicar el proceso ETL[1] (extracción, transformación y cargado) de datos meteorológicos propios de la Agencia Estatal de Meteorología (AEMET)[2], de forma que tras su consecución obtengamos un repositorio con datos meteorológicos de diferentes estaciones meteorológicas centralizados en un mismo espacio de almacenamiento.

Se presentarán también los trabajos realizados para crear una aplicación con el objeto de visualizar los datos meteorológicos que han sido almacenados y realizar operaciones con ellos como puede ser la creación de diferentes modelos para la predicción de temperaturas máximas diarias.

Como demostración de una aplicación a mayores que se puede realizar sobre los datos meteorológicos almacenados se mostrará cómo, mediante la conexión de una herramienta de visualización y análisis externa, se pueden obtener diferentes interpretaciones de los datos almacenados.

Aclarar que, en el proyecto realizado, solo se han recogido datos meteorológicos de la provincia de Valladolid, en concreto de aquellos municipios o lugares que poseen una estación meteorológica y cuyos datos son registrados en el sitio web de la Agencia Estatal de Meteorología. La decisión de limitarse únicamente a la provincia de Valladolid se debe a la consideración de que son datos suficientes para representar perfectamente el funcionamiento de la herramienta a desarrollar, a pesar de que, en un futuro, la zona geográfica de la que se recojan datos sea mucho más extensa.

1.1 Estructura del TFG

El contenido de esta memoria del TFG quedará dividido en lo que podemos considerar tres partes: una primera de desarrollo de una herramienta para desempeñar el proceso ETL de los datos meteorológicos de la AEMET logrando como resultado un almacén de datos de distintas estaciones meteorológicas. Una segunda que implica la creación de una aplicación que permita la visualización de los datos almacenados además de la creación de modelos estadísticos para la predicción de temperaturas máximas

diarias. Y por último, una tercera de representación de datos para la que se usará una herramienta externa de visualización que permita mostrar un análisis más extenso y profesional de los datos que el que se pueda ofrecer con la aplicación creada durante el desarrollo del TFG.

1.2 Estructura de la memoria

La memoria va a seguir una estructura determinada por la metodología elegida para el desarrollo del proyecto. Al tratarse de una metodología ágil la elegida, que se explicará en detalle más adelante, supondrá la división de la memoria en la explicación de las distintas iteraciones o *sprints* realizados de forma individualizada, con lo que para cada *sprint* se explicarán las secciones de análisis, diseño, desarrollo y pruebas realizadas para el desarrollo del mismo.

El resto de secciones de la memoria servirán de apoyo a las secciones existentes por cada sprint ayudando a su comprensión y facilitando datos de interés que completan la información necesaria para entender el desarrollo del proyecto en su conjunto.

2. Objetivos

En este capítulo podemos distinguir dos tipos de objetivos dependiendo de su procedencia. Por una parte, objetivos personales o lo que espero conseguir personalmente de la realización de este proyecto y, por otra parte, objetivos funcionales de la herramienta a desarrollar y del trabajo a realizar.

Los objetivos personales que pretendo conseguir con la realización de este TFG, son demostrar los conocimientos y competencias adquiridos durante mi estancia en la Escuela de Ingeniería Informática de Valladolid, una buena dinámica y estructuración del trabajo basada en la experiencia que he podido obtener con la realización de proyectos anteriores, aunque no tuviesen el mismo grado de complejidad y por último demostrar mi capacidad para resolver los problemas planteados que requieran de adquisición de nuevos conocimientos o de búsqueda de nueva información, lo cual incluye un correcto análisis y síntesis de esta información, para posteriormente poder poner los conocimientos adquiridos en práctica.

En cuanto a lo referido a la herramienta a desarrollar, el principal objetivo es demostrar que la herramienta es capaz de obtener datos meteorológicos útiles, es decir, que tengan una utilidad posterior a la extracción de los mismos y que la aplicación desarrollada es capaz de tratar o trabajar con esta serie de datos con el objetivo de realizar una representación de la información, ya sea mediante la creación de gráficos, de modelos estadísticos, o mediante la visualización de resúmenes de estos que nos ofrezcan diferentes formas de interpretarlos.

Este objetivo principal se puede desglosar en diferentes subobjetivos que marcarán su consecución:

- Lograr almacenar los datos obtenidos vía web, del sitio web de la Agencia Estatal de Meteorología, en una herramienta de almacenamiento acondicionada para poder guardar los datos que se consideren necesarios.

- Lograr procesar los datos adquiridos de la forma más acertada y óptima posible, antes de poder guardarlos en la herramienta de almacenamiento elegida para tal fin.

- Encontrar una forma de automatizar el proceso de obtención y guardado de datos, en caso de ser posible, para evitar la adquisición de estos de forma manual.

Para completar el trabajo, se consideran otros dos objetivos:

- Desarrollo de una aplicación para la visualización de datos y creación de modelos de predicción de temperaturas máximas diarias.

- Obtener una forma de conectar la herramienta de visualización de terceros a la herramienta de almacenamiento, para poder visualizar los datos contenidos en ésta utilizando toda la potencia que estas herramientas de visualización ofrecen hoy en día.

Con respecto a la creación de los modelos de predicción de temperaturas máximas diarias, el objetivo es demostrar el uso de los diferentes tipos de modelos en un ámbito que podríamos considerar a nivel de usuario, sin la necesidad de crear modelos que ofrezcan un ajuste o precisión muy cercanos a los

valores reales, ya que lo que se pretende con la realización de estos modelos es probar que los datos obtenidos mediante la herramienta creada son totalmente aprovechables y una forma de obtener o sintetizar información a partir de estos datos es la creación de modelos de predicción, que, sin buscar ser sobresalientes, sirvan para transmitir una información que pueda considerarse útil.

A pesar de que no se trata de un trabajo de estadística donde lo que primaría en los modelos creados sería un estudio de estos o aproximar los modelos lo máximo posible a los valores reales, un propósito más es la muestra de un histórico de las predicciones realizadas junto con los valores de temperaturas reales para cierto modelo donde se puedan observar las diferencias entre los valores predichos y los reales y ver si existe coherencia entre ambos conjuntos de datos. Al final es otra forma más de trabajar con los datos obtenidos ofreciendo una posibilidad de interacción a mayores con estos datos meteorológicos iniciales.

3. Planificación y costes

El desarrollo del proyecto ha conllevado la elección de una planificación junto con la selección y adquisición de los recursos necesarios para completar su implementación. En esta sección se muestra el plan de proyecto que se ha seguido junto con una estimación de los costes que supondrá el desarrollo del mismo.

3.1 Planificación del proyecto

En un principio el desarrollo de este Trabajo de Fin de Grado tenía como trabajo mínimo la creación de un sistema para ejecutar los procesos de extracción, transformación y almacenamiento de los datos meteorológicos más la habilidad de conectar el sistema con aplicaciones externas, como herramientas de visualización, aunque al final también se ha considerado, oportuno, desde un punto de vista didáctico, incluir una herramienta de creación de modelos de predicción de temperaturas con una interfaz gráfica para este propósito.

En vista de la segunda posibilidad que se incluiría en función del avance del proyecto y la disponibilidad para su realización, se ha optado por una metodología *Scrum*[3] con un desarrollo adaptativo e incremental ya que los objetivos finales del proyecto en un principio no estaban totalmente cerrados pudiendo sumarse algunas funcionalidades más, como las nombradas anteriormente, en caso de encontrar una buena dinámica de avance.

En vistas de la metodología escogida el trabajo se ha dividido en fases o *sprints*, en los que con la finalización de cada uno de ellos obtenemos un incremento en la funcionalidad del proyecto desarrollado. En cada *sprint* podremos encontrar una lista de eventos que nos indicarán los objetivos que se esperan de cada una de estas iteraciones y la forma de conseguirlos:

- Reunión de seguimiento: se trata de una reunión al principio de cada *sprint* donde se fijan los objetivos del *sprint* a realizar. En esta reunión se produce la identificación de tareas con su consiguiente estimación temporal.

- Listado de requisitos para la realización del *sprint.*

- Incremento de funcionalidad obtenido de los requisitos.

- Reunión de revisión: se realiza al final de cada *sprint* una vez se haya concluido el incremento planificado. Estas reuniones de revisión en el caso del presente TFG han coincidido con las de seguimiento, a excepción de la primera de seguimiento y última de revisión, el resto de reuniones realizadas se han aprovechado para cubrir los dos acometidos de revisar el trabajo realizado y posteriormente incluir las nuevas tareas a realizar. En estas reuniones de revisión se ha analizado el incremento de funcionalidad desarrollado en el *sprint* determinando y si se debería realizar algún cambio o cumple con las expectativas.

Durante el desarrollo del proyecto además podemos encontrar dos roles diferenciados:

- Propietario del producto: en el caso del presente proyecto este rol será cubierto por el tutor del TFG y entre las funcionalidades que tiene son las de establecer los objetivos o el trabajo que se requiere realizar.

- Personal de desarrollo: este rol será asumido por el alumno/autor del presente TFG con el cometido de completar las tareas necesarias para la consecución de los objetivos marcados.

La metodología *Scrum* que usaremos no se ajustará totalmente a las pautas de metodología ágil estándar marcadas ya que contará con algunas configuraciones propias, como puede ser la no realización de breves reuniones diarias típicas de una metodología *Scrum* o que solamente estarán presentes los dos roles nombrados con anterioridad, siendo esto poco común además de que los dos roles explicados sean cubiertos por una única persona. A pesar de estas discrepancias, es la mejor adaptación que podemos realizar de esta metodología que en términos de funcionamiento se adapta a la perfección a la realización del proyecto, ya que el desarrollo global de este no estaba completamente fijado en un principio y avanzar de forma incremental ha sido una gran solución para poder ir acoplando el proyecto al avance y a las necesidades existentes.

El proyecto final ha sido conformado por un total de 6 *sprints* con unas duraciones comprendidas entre 2 y 4 semanas cada *sprint*, dependiendo de la carga de trabajo a realizar en base a la estimación concretada en función de las tareas identificadas y objetivos marcados en las diferentes reuniones de seguimiento. La realización de la memoria de proyecto no se ha contemplado como tiempo de desarrollo para cada *sprint* realizándose posteriormente a su finalización. En contraposición y como medida para resolver la inexistencia de documentación sobre el desarrollo del proyecto, durante la realización del mismo se ha procedido a llevar un seguimiento o "tracking" del mismo, registrando los pasos y trabajos realizados además de las correspondientes anotaciones requeridas en cada tarea que luego se han usado a la hora de la realización de esta memoria.

Los *sprints* se han realizado en los siguientes periodos de tiempo:

Sprint	**Descripción**	**Horas estimadas**
1º	El comienzo de este *sprint* se corresponde con el comienzo del proyecto. La fecha de comienzo de este *sprint* ha sido el 03 de febrero de 2020, mientras que la duración del *sprint* ha sido de 2 semanas finalizando el 17 de febrero de 2020.	35 horas
2º	Su duración ha sido de 2 semanas, comprendidas entre el 17 de febrero de 2020 y el 02 de marzo de 2020.	25 horas
3º	Su duración ha sido de 2 semanas, comprendidas entre el 02 de marzo de 2020 y el 16 de marzo de 2020.	35 horas
4º	Ha tenido una duración de 4 semanas entre las fechas del 16 de marzo de 2020 y el 13 de abril de 2020.	70 horas

5º	Su duración ha sido de 4 semanas y se ha realizado entre las fechas 13 de abril de 2020 y 11 de mayo de 2020.	70 horas
6º	Su inicio a datado en el 11 de mayo de 2020 mientras que su finalización ha sido el 01 de junio de 2020 tras 3 semanas de duración y coincidiendo la fecha de finalización de este *sprint* con la de finalización del proyecto completo.	25 horas

Tabla 1: *sprints* del proyecto

3.1.1 Primer *Sprint*

Se trata de la primera toma de contacto con el proyecto y el punto de partida de este. El proyecto tiene como elemento principal de trabajo en todos los ámbitos datos meteorológicos y en base a esa premisa, durante el primer *sprint* se procederá al desarrollo de un programa para la extracción, transformación y almacenamiento de datos en un servidor de bases de datos, siendo la consecución del programa el objetivo de este *sprint*.

Las tareas identificadas para este primer *sprint* han sido:

- Extracción de los datos mediante una petición web al sitio web de la Agencia Estatal de Meteorología.
- Lograr un correcto procesado de los datos obtenidos para poder almacenarlos.
- Elección e instalación de un servidor de bases de datos.
- Almacenamiento de los datos obtenidos de la AEMET en la base de datos.

Con la conclusión de este *sprint* ya tenemos una herramienta para la extracción y almacenamiento de los datos en un servidor de bases de datos.

3.1.2 Segundo *Sprint*

En el periodo de duración del segundo *sprint* los trabajos a realizar son la configuración de la máquina y la programación de la ejecución del programa creado en el primer *sprint* gracias al administrador de procesos. El objetivo de este *sprint* es lograr la automatización del proceso de extracción y almacenamiento.

Las tareas identificadas para este *sprint* han sido:

- Acceso y configuración de la máquina virtual.
- Instalación del servidor de base de datos y configuración de este para permitir el acceso remoto.

- Automatización del proceso de extracción, transformación y almacenamiento de datos.

Al finalizar el *sprint* la extracción de los datos y el almacenamiento de estos ya no requerirá de trabajo manual para ejecutar el programa en un instante determinado.

3.1.3 Tercer *Sprint*

Durante el tercer *sprint* se establece como objetivo el desarrollo de una interfaz gráfica con Python que permita configurar y realizar una conexión con la base de datos instalada en la máquina virtual.

Las tareas identificadas para el *sprint* han sido:

- Elección de una biblioteca que ofrezca soporte para desarrollar un interfaz gráfico con Python.
- Familiarización con el módulo elegido para el desarrollo de la interfaz gráfica.
- Creación de una interfaz sencilla para la conexión con la base de datos probando el acceso remoto al servidor de bases de datos de la máquina virtual.

La finalización del *sprint* cumpliendo los objetivos marcados implica la obtención de la ventana de conexión de la aplicación de visualización y predicción de temperaturas máximas.

3.1.4 Cuarto *Sprint*

Los objetivos del cuarto *sprint* son la realización de la ventana principal de la aplicación y la inclusión de elementos de visualización de los datos meteorológicos almacenados en la base de datos alojada en la máquina virtual.

Las tareas identificadas del *sprint* han sido:

- Diseño de una interfaz para la página principal de la aplicación.
- Inclusión de herramientas para la visualización de los datos meteorológicos.

Con la finalización de este *sprint* tenemos la parte de visualización de datos meteorológicos de la aplicación a desarrollar.

3.1.5 Quinto *Sprint*

Los objetivos del quinto *sprint* son la realización de los modelos de predicción de temperaturas máximas y la inclusión de estos en la aplicación que se está desarrollando como una sección más de esta.

Las tareas identificadas a realizar durante el quinto *sprint* son:

- Estudio de los distintos tipos de algoritmos de regresión.
- Búsqueda y elección de una biblioteca para el desarrollo de los modelos de predicción con diferentes algoritmos.
- Diseño de una interfaz gráfica para la inclusión del apartado de predicción de temperaturas máximas mediante el uso de los modelos creados.

Con la finalización del *sprint* tendríamos la funcionalidad ofrecida por la aplicación completada.

3.1.6 Sexto *Sprint*

Los objetivos del último *sprint* implican finalizar la aplicación completando los últimos detalles y acoplar la herramienta de visualización externa que se usará para lograr una mejor representación y más completa que la que puede ofrecer la aplicación creada de los datos meteorológicos almacenados.

Las tareas identificadas del último *sprint* son:

- Corrección y mejora de algunos aspectos de la aplicación.
- Inclusión de una guía de uso en la aplicación desarrollada.
- Conectar con la herramienta externa de visualización y realizar algunos *dashboards* (paneles de mando) que nos permitan hacer patentes posibles relaciones entre los distintos datos medidos.

Con la finalización del sexto *sprint*, podríamos dar por concluido el proyecto desarrollado como Trabajo de Fin de Grado.

En la siguiente imagen podemos visualizar el diagrama de Gantt del TFG con los *sprint*, las tareas identificadas y su duración:

Trabajo de Fin de Grado	86días	02/03/2020	06/01/2020
Primer sprint	11días	02/03/2020	02/17/2020
Programa de ETL	0día	02/17/2020	02/17/2020
Segundo sprint	10días	02/18/2020	03/02/2020
Plataforma de datos meteorológicos	0día	03/02/2020	03/02/2020
Tercer sprint	10días	03/03/2020	03/16/2020
Interfaz de conexión	0día	03/16/2020	03/16/2020
Cuarto sprint	20días	03/17/2020	04/13/2020
Módulos de visualización	0día	04/13/2020	04/13/2020
Quinto sprint	20días	04/14/2020	05/11/2020
Modulo de predicción	0día	05/11/2020	05/11/2020
Sexto sprint	15días	05/12/2020	06/01/2020
Aplicación y herramienta de visualización externa	0día	06/01/2020	06/01/2020

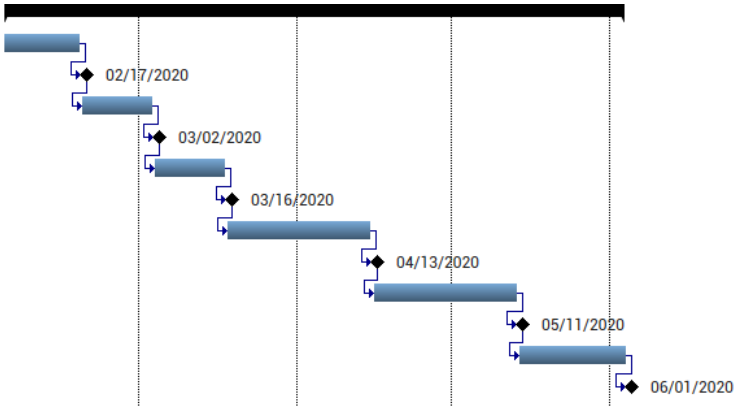


Figura 1: diagrama de Gantt TFG

3.2 Recursos materiales y costes

En esta sección mostraremos los recursos materiales con las características que se consideran mínimas para poder replicar el proyecto realizado por una parte y los recursos que ha usado el alumno para el desarrollo de dicho proyecto por otra parte.

También se representan los costes totales estimados de la realización del proyecto incluyendo tanto los costes del material usado, como los costes de personal.

3.2.1 Recursos materiales mínimos

Los costes básicos para la realización del proyecto básicamente se pueden resumir en tres:

El primero de ellos es la posesión o adquisición de un ordenador con al menos 2GB de RAM, 50Gb de disco para poder asumir el proyecto con garantías, aunque esta cifra podría ser mayor en caso de recoger datos de un ámbito geográfico grande y durante un periodo de tiempo muy amplio, un procesador de al menos 1.6GHz y tarjeta de red para poder realizar una conexión a internet.

El ordenador a usar necesitará los requisitos mínimos nombrados anteriormente para poder albergar programas de edición como Visual Studio Code (usado por el alumno), Eclipse, etc., para la realización de los programas, aunque vale cualquier otra herramienta de edición de código que posiblemente sea menos potente; pero especialmente son necesarias estas cualidades en el equipo a usar si se trata de un equipo con sistema operativo de distribución Linux y se pretende usar como herramienta de visualización de terceros Power BI, debido a la obligación de instalar una herramienta de virtualización donde se pueda instalar un sistema operativo de Windows para poder ejecutar este software de visualización que solamente está disponible en versiones Windows.

El segundo recurso, aunque no imprescindible, ya que se puede realizar el proyecto perfectamente sin él, se trata de una máquina virtual, un espacio en un servidor o incluso un servidor dedicado. En el caso de la adquisición de una máquina virtual alojada en un servidor, al poder estar habilitada en todo momento nos permite programar la ejecución de tareas lo cual será muy útil para lograr automatizar la

extracción y almacenamiento diario de los datos meteorológicos evitando tener que realizar manualmente estas tareas a diario. Con el caso de disponer de un hosting en un servidor será posible instalar una máquina virtual cumpliendo con el mismo propósito, mientras que con el servidor dedicado podremos aprovechar el propio administrador de tareas si se trata de un servidor con sistema operativo Windows o Cron si se trata de un servidor con una distribución de Linux, obteniendo la misma posibilidad de programar la ejecución de ciertos programas o tareas y evitando tener que ejecutarlos manualmente.

El tercer recurso para poder realizar el proyecto se trata de la conexión a internet que será necesaria para instalar cualquier herramienta que vayamos a utilizar durante el desarrollo del proyecto o para la conexión con el segundo recurso nombrado anteriormente entre otros aspectos. Se trata de un recurso esencial puesto que sin la conexión a internet no podremos instalar el entorno de desarrollo mínimo que nos permita comenzar con la implementación del TFG.

3.2.2 Recursos usados por el alumno

Características del ordenador usado para la realización del proyecto:

- Portatil Asus X555LDB.
- 4Gib de RAM.
- Procesador Intel Core i7-5500 - 2.4GHz.
- Sistema operativo distribución Linux, Ubuntu 19.10, kernel Linux 5.4.0-33-generic.
- HDD 500GB.
- SSD 120GB.

Características de la máquina virtual usada:

- Sistema operativo: Ubuntu 18.04.4 LTS, kernel Linux 4.15.0-76-generic
- Procesador de distribución Intel.
- Memoria RAM 2GiB.
- Tamaño de disco: 8 GiB.
- Puertos: listado de los puertos de acceso a la máquina virtual remotamente.
    - Puerto SSH: 20051

- Puerto Web: 20052
- Puerto MySQL: 20053

3.2.3 Costes del proyecto

Podemos diferenciar por una parte entre los costes del material expuesto en el apartado 3.2.2 y los costes de personal que son necesarios para el desarrollo del proyecto basándonos en las horas estimadas que se indican en la Tabla 1 con los *sprints* del proyecto.

El coste total de los recursos materiales utilizados para el desarrollo del proyecto contando los meses de conexión a internet, el precio del portátil y el coste de la máquina virtual es de 825€ que quedan desglosados de la siguiente forma:

- Coste del portátil: 600€
- Coste de internet: 225€ (4 meses x 45€/mes)
- Coste de máquina virtual: 0€ (Costes asumidos por la Escuela de Ingeniería Informática de Valladolid)

El coste de personal estimado basándose en un coste medio de 10€/hora de trabajo y en función de las horas estimadas en la Tabla 1 para la finalización del proyecto y las horas estimadas de documentación del mismo, se establece en una cifra de 3200€

Coste total estimado para la realización del proyecto: 4025€

4. Entorno tecnológico

Para el desarrollo del TFG han sido necesarias numerosas tecnologías que nos permiten poder desarrollar o completar cada una de las tareas a realizar y que conforman el proyecto completo.

Algunas de estas tecnologías por sí solas no ofrecen toda la funcionalidad requerida para el desarrollo del proyecto, por lo que en algunos casos se ha requerido de la inclusión de funcionalidades complementarias con estas tecnologías, que generalmente se han obtenido gracias a las bibliotecas de Python.

A continuación, se van a describir cada una de las tecnologías usadas y las bibliotecas de Python que se han requerido para incluir nuevas funcionalidades:

4.1 Python

El lenguaje de programación usado para la realización de la parte lógica del proyecto, ha sido Python. Python es un lenguaje de programación interpretado que aporta diferentes soluciones y herramientas para un desarrollo rápido de aplicaciones en gran variedad de plataformas.

Para la realización de la parte de extracción y almacenamiento de los datos meteorológicos de la AEMET se ha usado la versión de Python 2.7.17[4], ya que es la que se instaló en la máquina virtual usada para estas funciones, a pesar de que se podía haber realizado con la última versión de Python disponible, la versión 3.8.2[5], aunque se desestimó la opción ya que el funcionamiento que estaba ofreciendo el programa para la extracción y almacenamiento de datos era el buscado.

Con respecto a la aplicación diseñada para la visualización de datos y predicción de temperaturas, si se ha realizado con la versión 3.8.2.

4.2 PIP

PIP es el instalador de paquetes de Python, gracias al cual se pueden gestionar, instalar o desinstalar de una forma muy sencilla paquetes de software escritos en Python.
La instalación de paquetes es muy sencilla basándose en sentencias de la forma:

pip install nombre_paquete.

Para desinstalar los paquetes previamente instalados, simplemente tendremos que sustituir "*install"* por *"uninstall"* en la sentencia descrita anteriormente.

4.3 Bibliotecas Python

No todas las funcionalidades deseadas están disponibles en lo que podríamos denominar "Python de serie" o el lenguaje que nos encontramos después de realizar su instalación. Debido a la necesidad de obtener nuevas herramientas de trabajo que nos ofrezcan nuevas funcionalidades para continuar con el desarrollo del TFG, requerimos del uso de bibliotecas para Python que nos van a dar la posibilidad de tener la funcionalidad deseada.

Gracias a "PIP", el sistema de gestión de paquetes de software escritos en Python visto anteriormente, se han añadido las bibliotecas necesarias a Python, para poder trabajar con ellas a la hora de desarrollar los programas necesarios para implementar la herramienta de extracción, almacenamiento y transformación de datos, además del modelo de predicción para las temperaturas máximas diarias.

4.3.1 Wget

Es un paquete de software creado para poder descargar contenido vía web de una forma sencilla. Wget deriva de "World Wide Web" (w) y de obtener (get), lo cual nos pre indica el uso que vamos a poder dar a este comando, el cual nos permite la descarga de contenido de servidores web.

Generalmente es una herramienta usada por línea de comandos en sistemas UNIX, que puede ser fácilmente llamada en scripts, programas, como en el caso actual, jobs…

GNU Wget soporta descargas de contenido mediante HTTP, HTTPS, FTP y FTPS, que son los protocolos más usados en Internet.

La sintaxis de uso básico de esta herramienta es:

wget.download(url_del_elemento_a_descargar)

4.3.2 os y os.path

De los servicios genéricos del sistema operativo, importamos el módulo "os" (miscellaneous operating system interfaces), el cual nos provee soporte para usar las dependencias funcionales del sistema.

En concreto del módulo de las dependencias funcionales del sistema importamos "remove", la cual nos permite eliminar un archivo del sistema conociendo previamente la ruta o camino hasta dicho archivo. La dependencia "remove", se identifica con el comando "rm" en UNIX, ya que ambos muestran el mismo comportamiento y podemos decir que se trata de la misma dependencia del sistema.

Para el uso del comando simplemente basta con usar la siguiente sentencia:

remove(path)

La biblioteca path (os.path) del módulo nos ofrece funciones para trabajar con nombres o rutas de archivos (path). Existen diferentes posibilidades como *(sentencia a usar)*:

- *dirname(path)*: nos devuelve el nombre del directorio de una ruta dada.
- *isdir(path)*: nos devuelve "true" si la ruta dada es un directorio.
- *isfile(path)*: nos devuelve "true" si la ruta especificada es un archivo.
- *exists(path)*: nos devuelve "true" si la ruta dada existe.

Las funcionalidades presentadas anteriormente son algunas de las más útiles, aunque podemos encontrar muchas más en la biblioteca. Durante el desarrollo del TFG se ha utilizado la función *exists(path)* para comprobar la existencia de los archivos CSV previamente a su borrado en caso de existir.

4.3.3 CSV

El módulo CSV nos permite leer, escribir o realizar cualquier modificación a archivos que están en formato CSV (Comma Separated Values).

Los archivos en formato CSV son especialmente usados al trabajar con bases de datos o herramientas como Excel o similares, ya que permiten realizar operaciones de importación y exportación de datos de una forma bastante sencilla y en este formato que resulta sencillo de procesar a la hora de trabajar con los archivos.

Gracias a este módulo, podemos usar diferentes funciones que nos permiten trabajar con archivos en formato CSV, de una forma muy intuitiva permitiendo múltiples formas de trabajo con estos archivos. Las principales operaciones que se pueden realizar con este módulo son las siguientes:

- csv.reader(csvfile, dialect='excel', fmtparam): nos permite iterar las diferentes líneas de un "archivo.csv" dado (csvfile), con la posibilidad de especificar los delimitadores de los campos del "archivo.csv" (dialect) y la posibilidad de sustituir o sobreescribir algún carácter específico del documento (fmtparam).

- csv.writer(csvfile, dialect, fmtparams): permite escribir datos en un archivo con formato CSV. La función "csv.writer()" tiene los mismos parámetros, que la función "csv.reader()" descrita anteriormente, con la excepción de la forma de apertura del archivo pasado como parámetro, que en el caso anterior será en modo lectura, mientras que en este se realizará en modo escritura.

4.3.4 Datetime

El módulo datetime nos proporciona clases para la manipulación de fechas y horas, en diferentes formatos y con diferentes posibilidades, ya sea representando diferentes modelos de fechas, realizando operaciones aritméticas con fechas o analizando las propias fechas u horas.

Los valores de horas, son representados con la clase “time”, y tiene como posibles atributos, hora, minuto, segundo y microsegundo, mientras que las fechas son representadas por la clase “date”, teniendo como atributos año, mes y día.

La clase “datetime”, permite usar atributos tanto de “date” como de “time”, pudiendo representar fechas completas. Además de la representación de una fecha completa en el formato predeterminado, el ISO-8601 (YYYY-MM-DD HH:MM:SS.mmm), se pueden generar formatos distintos con “strftime()”, que permite crear una cadena con los valores del “datetime” y a mayores se puede elegir entre las diferentes zonas horarias.

4.3.5 Pandas

Pandas es una biblioteca de Python que está escrita como una extensión de la biblioteca Numpy y que nos ofrece herramientas de manipulación de datos de alto nivel. Gracias a Pandas se pueden crear y trabajar con datos del tipo matricial, pudiendo distinguir tres tipos básicos de datos:

- Series: se tratan de listas de una dimensión, que pueden crearse como un diccionario, es decir, con una clave y un valor asociado a dicha clave.

- DataFrame: son tablas de dos dimensiones y al igual que las series pueden ser del tipo clave-valor. Además, en los dataframe, una de las particularidades que tienen, es que podemos especificar tanto el nombre de las filas (index), como el de las columnas (columns).

- Panels: son tablas de 3 dimensiones, con características similares al DataFrame

De los tres tipos de datos expuestos anteriormente, el más destacado y el que podríamos denominar como estructura clave de Pandas, es el DataFrame. Un DataFrame, podemos definirlo como una colección ordenada de columnas que pueden tener un identificador. La estructura de un dataframe, podríamos decir que se asemeja a una tabla de una base de datos relacional, en las cuales tenemos un nombre bien definido para cada columna y los datos que representan dicha columna. En un DataFrame al igual que en las tablas de las bases de datos, cada columna puede albergar diferentes tipos de datos.

Gracias al DataFrame, podemos seleccionar o filtrar el contenido de las tablas de datos de una forma sencilla, fusionar conjuntos de datos, insertar o eliminar columnas de la estructura, manipular series temporales…

La sintaxis básica de un DataFrame es la siguiente:

```
dataframe = DataFrame({
        ‘nombre_lista1’ : [ lista1 ],
        ‘nombre_lista2’ : [ lista2 ],
                “
                “
        ‘nombre_lista_n” : [ lista_n ]
```

});

Además de ofrecernos la posibilidad de crear DataFrames, contiene numerosas opciones para el trabajo con datos como puede ser la creación de series, comparación de elementos, filtrado de estos o incluso la lectura de datos en diferentes formatos como puede ser Excel, CSV, JSON…

4.3.6 Matplotlib

Matplotlib es una biblioteca de Python que permite hacer representaciones gráficas de conjuntos de datos.

Matplotlib se ha usado durante el desarrollo de los modelos de para la predicción de temperaturas, como soporte para la correcta comprensión de algunos modelos realizados y en la aplicación para la visualización de datos creada, donde los gráficos usados para este fin se han realizado mediante esta biblioteca que permite la realización de múltiples tipos de gráficos como pueden ser de dispersión, barras o de sectores entre otros, además de permitir añadir una barra de herramientas, el NavigationToolbar2Tk, gracias al cual se puede interactuar con los gráficos creados siendo posible seleccionar ciertos elementos de estos o ampliar ciertas zonas para mejorar la visualización entre otras funcionalidades.

4.3.7 Scikit-learn

La biblioteca Scikit-learn está expresamente creada para Machine Learning ofreciéndonos soporte para el uso de diversas herramientas creadas para aplicar algoritmos de regresión, clasificación y análisis sobre un conjunto de datos cualesquiera.

Las herramientas que ofrece Scikit-learn usan tanto algoritmos de aprendizaje no supervisado como pueden ser las redes neuronales no supervisadas, como algoritmos de aprendizaje supervisado, siendo un ejemplo de estos los árboles de decisión, la regresión lineal, máquinas de vectores de soporte…

Scikit-learn está ligada a bibliotecas vistas anteriormente como pandas o Matplotlib, además de otras como SciPy (Python científico) o SymPy (competencias en álgebra computacional), necesarias para el desarrollo de algunas de las técnicas proporcionadas por Scikit-learn.

En concreto el módulo Scikit-learn ha sido usado en el proceso de desarrollo de los diferentes modelos de predicción empleando para cada modelo creado un algoritmo diferente. Las herramientas de Scikit-learn que se han usado han sido “linear_model” que permite la realización de modelos de regresión lineal, “MLPRegresor” que permite la creación de redes neuronales de regresión, “KNeighborsRegressor” usado para la regresión mediante el algoritmo de k-vecinos más próximos, y “DecisionTreeRegressor” gracias al cual se puede obtener una regresión mediante árboles de decisión.

4.3.8 Tkinter

Es un módulo que permite la creación de interfaces gráficas de usuario (GUI). Tkinter establece un enlace entre Python y la biblioteca Tk que es la que se encarga de comunicar con el sistema de ventanas del sistema operativo.

Tkinter pone nuestra disposición un amplio repertorio de widgets como pueden ser botones, contenedores de contenido, etiquetas, entradas de texto o selectores de opciones entre otros que facilitan el desarrollo de las interfaces gráficas gracias a las facilidades que ofrece y lo intuitivo que resulta su uso.

Tkinter está basado en una estructura "event driver" o dirigida por eventos que consiste en la creación de la interfaz gráfica de usuario y esperar a la ocurrencia de un evento mientras el programa se encuentra en un bucle denominado "event loop" a la espera de la ocurrencia del evento en cuestión.

En concreto esta biblioteca se ha utilizado para el desarrollo de la parte frontend de la aplicación para la visualización de datos meteorológicos y predicción de temperaturas máximas.

4.3.9 Threading

El threading permite la ejecución de varias operaciones simultáneamente en el mismo espacio de procesos. La biblioteca Threading de Python permite la creación y manejo de hilos para conseguir realizar operaciones de forma simultánea durante la ejecución de un programa.

En Python cada hilo o thread se considera como un subproceso que ejecuta una operación determinada, de esta forma podríamos considerar que ejecutar varios hilos es casi como ejecutar varios procesos, con la diferencia de que los hilos comparten recursos comunes del mismo proceso al trabajar sobre el mismo espacio de datos, siendo más fácil la comunicación entre estos que de tratarse de procesos separados.

La sintaxis básica para la creación de hilos es la siguiente:

hilo = *threading.Thread(target = tarea_a_realizar, args = argumentos_funcion, daemon = True/False)*
hilo.start()

Para finalizar un hilo usaremos la siguiente sentencia.

hilo.join()

4.3.10 Python Imaging Library (PIL)

Se trata de una biblioteca de Python que ofrece soporte para la modificación visualización y guardado de diferentes formatos de archivos del tipo imagen.

Esta biblioteca permite la manipulación de imágenes siendo posible modificarlas por pixeles, por contrastes y brillo, agregar texto a estas o muchas más opciones.

Durante el desarrollo del TFG la biblioteca PIL se ha utilizado para la apertura de imágenes en la interfaz gráfica de la herramienta desarrollada para la visualización de datos meteorológicos y predicción de temperaturas máximas.

4.3.11 Time

El módulo Time de Python se ha usado exclusivamente para el uso de retardos (*delays*) en la aplicación de visualización y predicción de temperaturas, mediante la función "sleep()" propia de este módulo, la cual para la ejecución del hilo actual por un tiempo determinado en segundos especificado mediante un parámetro dado.

La sintaxis de uso de esta operación:

```
time.sleep(tiempo_en_segundos)
```

4.3.12 MySQL Connector

La extensión Mysql Connector nos proporciona un controlador (Driver) de bases de datos de MySQL, que nos permite realizar una conexión con servidor de MySQL usando una API (Application Programming Interface), la cual está escrita íntegramente en Python con lo que no hay dependencias de otras plataformas.

MySQL Connector nos proporciona un conjunto de herramientas completo, con el cual podemos realizar cualquier operación que realizaríamos desde un servidor MySQL.

Mediante Mysql Connector podemos realizar tanto una conexión a un servidor remoto gracias a conexiones TCP/IP, como a un servidor local, especificando el usuario o a la base de datos concreta a la que queremos acceder, con un puerto de acceso definido y otros campos más a mayores. La sentencia de conexión básica es la siguiente:

```
conexion = mysql.connector.connect(
        user = 'nombre_usuario',
        passwd = 'contraseña_usuario',
        database = 'base_datos',
        host = 'dirección_servidor_mysql',
        port = 'puerto_TCP/IP_del_servidor',
);
```

La sentencia anterior es un ejemplo de conexión básica a un servidor de MySQL, en la cual se especifican los campos estrictamente necesarios para poder conectarse, a pesar de existir muchos más campos a mayores.

Una vez establecida la conexión con el servidor MySQL, podremos realizar operaciones como crear una base de datos, crear una tabla, insertar datos en una tabla o seleccionar o borrar elementos de una tabla, entre otros. La sintaxis de uso de las anteriores operaciones es:

cursor = conexion.cursor();

query = "sintaxis_MySQL" ej: "CREATE DATABASE DB"

cursor.execute(query);

4.4 MySQL

MySQL es un sistema gestor de bases de datos relacional, donde una relación se puede considerar como un conjunto de datos, al que llamamos tupla. Si abstraemos la idea de relación a una tabla con el gestor de datos MySQL, tenemos que cada fila de la tabla se representa como una tupla, mientras que cada columna de una tabla son los campos de ésta.

MySQL, es el gestor de bases de datos más usado, en parte debido a:

- Es de código abierto.
- Disponibilidad en múltiples plataformas y sistemas operativos.
- Facilidad de conexión, tanto a un servidor MySQL local, como a uno remoto mediante sockets TCP/IP.
- Verificación y seguridad basada en contraseñas y un sistema de privilegios, para los usuarios del servidor.
- Gran capacidad de almacenamiento y procesado de datos.

Destacar que en el caso del desarrollo del TFG actual, el servidor MySQL usado se ha instalado sobre una máquina virtual y gracias a la facilidad de acceso al servidor desde hosts remotos, la parte de visualización de los datos almacenados se ha podido desarrollar desde un host remoto, independientemente de la localización del servidor.

4.5 Máquina virtual

Una máquina virtual se trata de un contenedor de software aislado que puede albergar un sistema operativo y aplicaciones, siendo de esta forma capaz de emular el comportamiento de un computador real.

Para dar lugar a una máquina virtual, debe existir un proceso de virtualización en el cual se abstraen los recursos hardware de una máquina física, a la que denominamos *Host* y el sistema operativo de la máquina virtual, a la que denominamos *Guest.*

Entre el host y la o las máquinas virtuales o Guests, se encuentra una capa de software denominada hipervisor o monitor de máquina virtual (VMM), que genera una capa de virtualización que separa los recursos físicos del Host de las máquinas virtuales creadas. El hipervisor de esta forma se encarga de emular y administrar estos recursos físicos del Host (CPU, memoria RAM, dispositivos de almacenamiento e interfaz de red) para cada máquina virtual albergada, de forma que cada máquina virtual piense que los recursos físicos reales están a su disposición.

En la siguiente imagen podemos ver un sistema tradicional a la izquierda, donde el sistema operativo se aloja directamente en el hardware por lo que tendrá a su disposición todos los recursos físicos, mientras que a la derecha tenemos una arquitectura virtual, donde podemos ver como el hipervisor, VMWare (tipo de hipervisor), posibilita la creación de varias máquinas virtuales sobre el Host.

Figura 2: hipervisores de máquinas virtuales

Las máquinas virtuales tienen una serie de características que hacen ventajoso el uso de estas herramientas:

- Particionamiento: permite ejecutar varias máquinas virtuales en un mismo host y la distribución de recursos físicos entre estas.

- Aislamiento: cada máquina virtual alojada en un host es independiente del resto de máquinas alojadas en éste, de forma que cualquier fallo en una de éstas solo afectará a la máquina virtual que haya sufrido dicho fallo.

- Encapsulación: garantiza que el estado de una máquina virtual pueda ser guardado como un archivo, lo cual permite la copia de máquinas virtuales con gran facilidad.

- Independencia del hardware: el hardware virtualizado para cada una de las máquinas virtuales, puede ser diferente del hardware real del Host y diferente para cada una de las máquinas alojadas.

A la hora de realizar el proceso de virtualización podemos distinguir dos tipos de hipervisores:

- Hipervisor de tipo 1: en este caso el hipervisor se ejecuta directamente sobre el hardware de la máquina física y se le denomina "nativo" o "unhosted". A este tipo de virtualización se le denomina "Virtualización basada en hipervisor". Algunos ejemplos de hipervisores de este tipo son: Linux KVM, VMWare ESXi, Microsoft Hyper-V Server...

- Hipervisor de tipo 2: también llamado "hosted", a diferencia del hipervisor de tipo 1, este hipervisor se ejecuta sobre un sistema operativo previamente instalado en el Host. Algunos ejemplos de hipervisores de este tipo son: Oracle Virtual Box, VMware, QEMU…

En el caso de la máquina virtual usada para la realización del presente Trabajo de Fin de Grado, el host sobre el que está alojada dicha máquina virtual se trata del servidor bajo el dominio *virtual.lab.inf.uva.es*, un servidor propio de la Escuela de Ingeniería Informática de la Universidad de Valladolid.

El proceso de virtualización se ha realizado mediante un hipervisor de tipo 2, en concreto mediante QEMU, el cual ha permitido emular un sistema informático completo a excepción de la interfaz gráfica ya que se realizan todas las interacciones mediante comandos.

La elección del uso de una máquina virtual, para realizar los procesos de extracción y almacenamiento de datos, además del proceso de predicción, está basada en dos factores determinantes:

- La disponibilidad del servidor: el servidor está activo en todo momento, permitiendo que la máquina virtual esté en ejecución constantemente, facilitando la automatización de procesos gracias al uso del administrador de tareas o procesos en el caso de Ubuntu, Cron. Gracias a la automatización de las tareas se evita la ejecución manual de los programas de extracción y almacenamiento de los datos meteorológicos cada día.

- Posibilidad de crear un entorno de producción: el hecho de poder acceder remotamente a los datos almacenados desde cualquier dispositivo, permite que la herramienta creada, pase a ser un entorno de producción, esto es debido a que cualquier usuario final de la herramienta, únicamente tendrá que adquirir el software de visualización y ya podría acceder a los datos guardados gracias a la herramienta desarrollada y que se encuentra alojada en la máquina virtual.

4.6 Cron

Es un administrador de procesos propio de los sistemas operativos Unix, que se ejecuta como un "daemon". Los daemons pueden considerarse como servicios del sistema que se ejecutan en segundo plano, es decir, no son interactivos, por lo que no son ejecutados directamente por el usuario.

El servicio Cron se encarga de ejecutar otros procesos o tareas que se encuentran especificados en el archivo *crontab* o tabla de tareas, de forma regular y en intervalos definidos en el propio archivo.

En el archivo *crontab* se guarda la lista de comandos que deben ejecutarse en un tiempo especificado por el propio usuario.

En el caso del presente TFG, se ha utilizado este administrador de procesos para la ejecución automática del programa de extracción y almacenamiento de los datos obtenidos de la Agencia Estatal de Meteorología. La programación guardada en el archivo crontab, implica la ejecución del programa tres veces al día durante todos los días automatizando de esta forma la tarea de extracción y almacenamiento de datos sin la necesidad de ejecutar manualmente el programa diariamente para este fin.

En la siguiente imagen podemos ver la sintaxis básica para incluir una tarea en el archivo *crontab*:

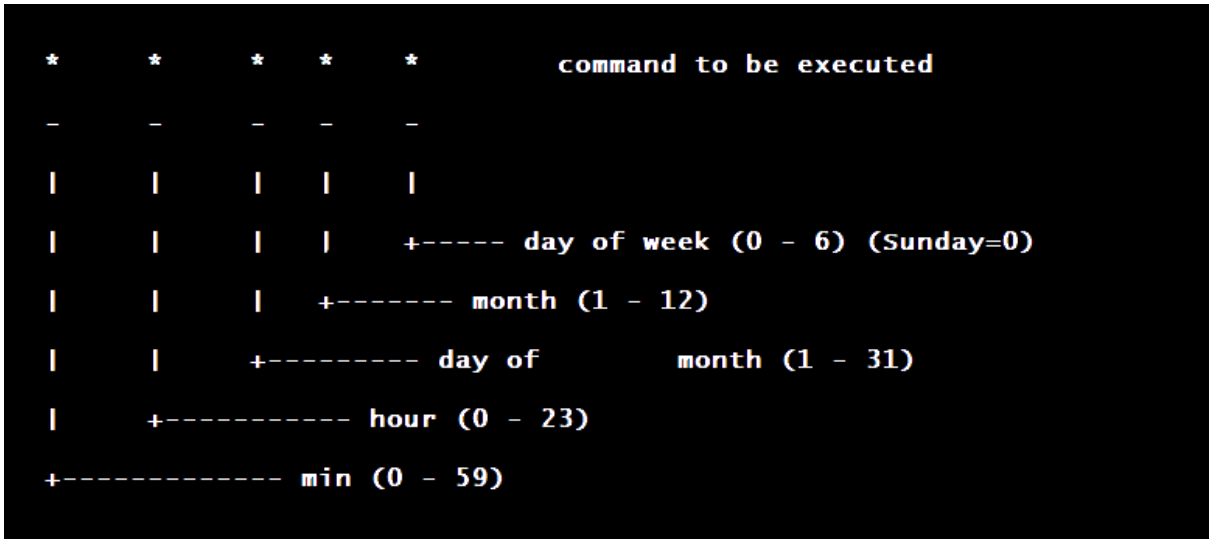


Figura 3: sintaxis del archivo crontab

4.7 Power BI

Es una herramienta para el análisis que permite la visualización de diferentes conjuntos de datos provenientes de una o varias fuentes distintas con el objetivo de que puedan ser consultados, analizados y representados de diferentes modos de una forma fácil y muy intuitiva gracias a la cantidad de herramientas que pone a nuestra disposición para trabajar con los datos seleccionados.

En este TFG, Power BI es la herramienta de visualización externa usada para demostrar que el trabajo realizado con la adquisición de los datos meteorológicos y su almacenamiento en una base de datos puede tener varios usos posteriores que permiten el trabajo con éstos. El uso de Power BI aporta información más detallada de los datos meteorológicos además de permitir la total personalización de su representación, de forma que se puedan visualizar en diferentes formatos o agrupaciones para mejorar la comprensión de estos.

4.8 Consideraciones finales sobre el software usado

En este apartado de entorno tecnológico únicamente se ha procedido a realizar una breve descripción de las herramientas usadas para el desarrollo del Trabajo de Fin de Grado, mientras que la conformación de estas herramientas en caso de requerirlo está especificada en el manual adjunto en los anexos del documento, donde se pueden ver las diferentes configuraciones usadas y los pasos necesarios para lograr dichas configuraciones con el propósito de replicar correctamente los desarrollos realizados durante el TFG.

5. Primer *sprint*: herramienta para el proceso ETL

El principal objetivo del desarrollo de este *sprint* es la consecución de una herramienta para desempeñar el proceso ETL (extracción, transformación y carga) de datos meteorológicos de la web de la AEMET. La consecución de esta herramienta que permita el almacenamiento de datos meteorológicos tratados y transformados a un formato adecuado es la base del proyecto completo del TFG.

El desarrollo de la herramienta para ejecutar el proceso ETL se considera la tarea más crítica del TFG, ya que el resto de procesos a realizar se realizarán con los datos meteorológicos obtenidos gracias a dicha herramienta y de los cuales dependerán todos los trabajos futuros.

La realización del *sprint* ha supuesto el análisis de requisitos y la realización del diseño de la solución previos, la consiguiente implementación de la solución y las pruebas realizadas sobre el trabajo desarrollado.

5.1 Análisis

A continuación, se muestran los requisitos expuestos para la realización de este *sprint*. De los requisitos expuestos se indicará una descripción y su consideración como requisito crítico o no crítico.

Nº de requisito	**Descripción**	**Crítico**
1	El proceso/programa/script (herramienta ETL) a desarrollar permitirá la extracción de los datos meteorológicos y los guardará en una herramienta de almacenamiento.	Sí
2	Los datos meteorológico se obtendrán del sitio web de la Agencia Estatal de Meteorología (AEMET).	Sí
3	El programa será flexible a la incorporación de futuras estaciones para la extracción de sus datos y posterior almacenamiento de estos.	Sí
4	Los datos meteorológicos deberán ser tratados previamente a su almacenamiento para guardarlos con un formato acorde a su tipología.	Sí
5	El programa se encargará del creado y formateado de la base de datos a usar para el almacenamiento de los datos meteorológicos.	Sí
6	La herramienta de almacenamiento se tratará de una base de datos del tipo relacional, a ser posible MySQL	No
7	Los datos meteorológicos a extraer provendrán de las distintas estaciones de la provincia de Valladolid.	No
8	El lenguaje de programación a usar será Python.	No

Tabla 2: requisitos primer sprint

5.1.1 Modelo de dominio

En el siguiente modelo de dominio podemos ver cada una de las entidades identificadas para la resolución del primer *sprint*, junto con sus atributos y relaciones ofreciéndonos una primera vista de cómo será el diseño que debe tomar el sistema a desarrollar y las tareas que se deberán llevar a cabo.

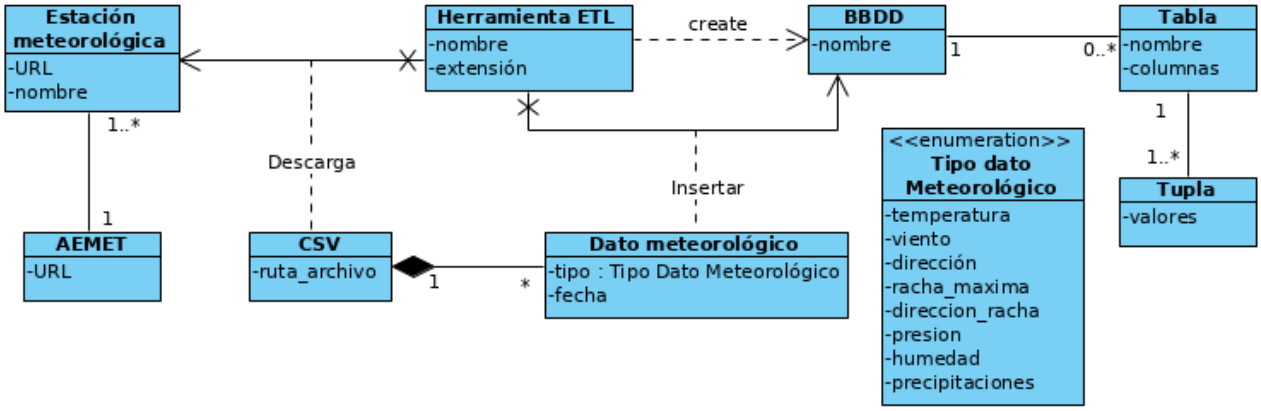


Figura 4: modelo de dominio primer sprint

5.1.2 Casos de uso

En este primer sprint solo podemos considerar un caso de uso muy básico que se detallará en la tabla 3.

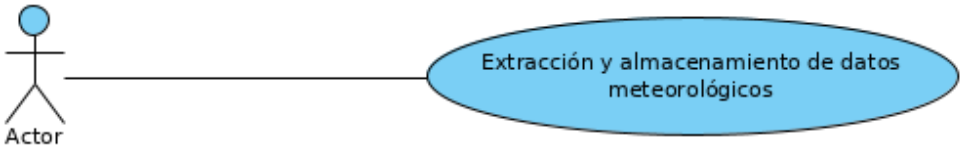


Figura 5: diagrama de caso de uso primer sprint

Caso de uso primer *sprint*	
Título	Extracción y almacenamiento de datos meteorológicos
Actor	Usuario

Descripción	El programa debe ser capaz de descargar los datos almacenados, tratarlos y guardarlos en la base de datos que deberá también crear, tras su ejecución.	
Secuencia normal	**Pasos**	**Acción**
	1	El actor ejecuta el programa creado
	2	El programa descarga cada uno de los archivos CSV asociados a las diferentes estaciones con los datos meteorológicos de ese día de la página web de la AEMET y lo notifica.
	3	El programa almacena los datos meteorológicos e indica su finalización.
Excepciones	**Pasos**	**Acción**
	3	Si no hay ninguna base de datos creada con sus respectivas tablas para cada estación meteorológica el programa las crea y lo notifica.

Tabla 3: caso de uso primer sprint

5.2 Diseño

La representación o visualización de la solución que se pretende conseguir viene dada por el diseño de la misma. En esta sección se especificarán los componentes, arquitectura y otras características del sistema a desarrollar en el primer *sprint* y que vendrán predefinidos por el trabajo de análisis realizado previamente.

5.2.1 Diseño del sistema

En base de los requisitos expuestos, la funcionalidad y el único modo de interacción posible representado anteriormente en el caso de uso, el sistema tendrá una arquitectura del tipo *cliente-servidor*, ya que podemos diferenciar claramente un proveedor, en este caso de los datos meteorológicos que sería "la AEMET", y el cliente que es la base de datos creada para el almacenamiento de los datos obtenidos.

Además, la estructura *cliente-servidor* es el estándar de interacción con internet, la cual se realiza durante el proceso de extracción de los datos meteorológicos.

En el siguiente diagrama de despliegue podemos comprobar la estructura *cliente-servidor* que tiene el sistema a desarrollar en el primer *sprint*:

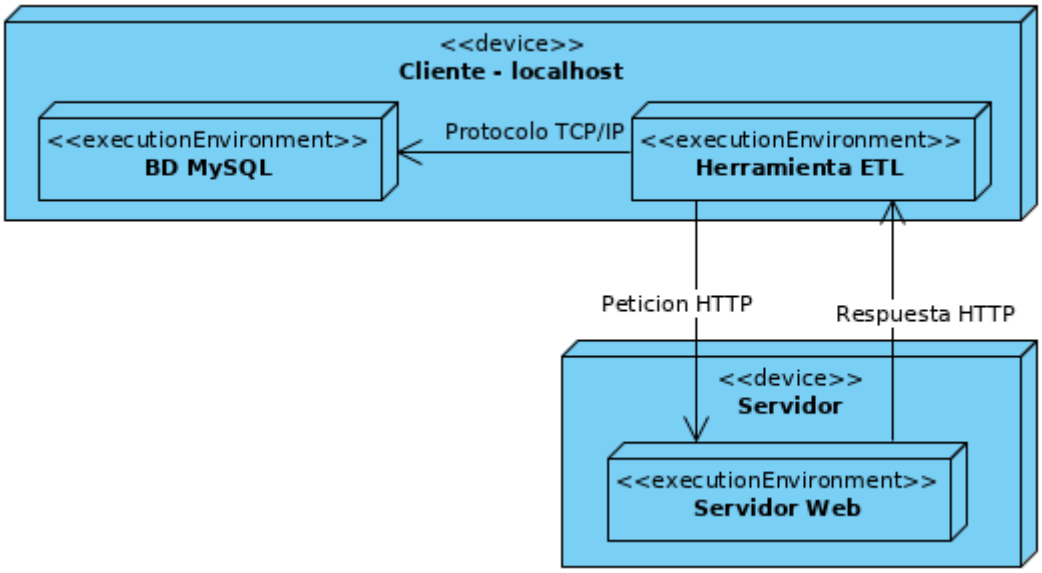


Figura 6: diagrama de despliegue primer sprint

En el diagrama de despliegue se puede observar claramente la disposición física de la arquitectura del sistema a desarrollar en el primer sprint, es decir el *hardware* usado, donde podemos distinguir los siguientes dispositivos físicos:

- Cliente-localhost: encontramos la base de datos instalada y el programa a ejecutar que se encargará tanto de la extracción como del almacenamiento de los datos meteorológicos.

- Servidor: el servidor web atenderá a las peticiones HTTP realizadas mediante el comando *wget.*

5.2.2 Diseño de la base de datos

La base de datos tiene un diseño muy simple formado por una tabla por cada estación meteorológica de la que se obtienen los datos meteorológicos. Las tablas no cuentan con ningún tipo de relación entre ellas y cuentan con un campo identificador o *clave primaria.*

En la siguiente tabla podemos ver el formato de una tabla con cada uno de los campos que la componen:

Nombre de la tabla	*nombre_estacion*	
Campos	**Descripción - tipo de valor**	**Clave**
Fecha (*Primary key*)	Tipo datetime - Not null (fecha - valor no nulo)	Primary key (PK)

Temperatura	Tipo float (valor con decimales)	
Viento	Tipo int (valor entero)	
Direccion	Tipo varchar (hasta 255 caracteres)	
Racha	Tipo int (valor entero)	
DirRacha	Tipo varchar (hasta 255 caracteres)	
Precipitaciones	Tipo float (valor con decimales)	
Presion	Tipo float (valor con decimales)	
Humedad	Tipo int (valor entero)	

Tabla 4: estructura de una tabla de la base de datos

La estructura de la base de datos una vez ejecutado por primera vez el programa creado y en función de las estaciones meteorológicas de la provincia de Valladolid, que son las que se han utilizado para el desarrollo del TFG, es la siguiente:

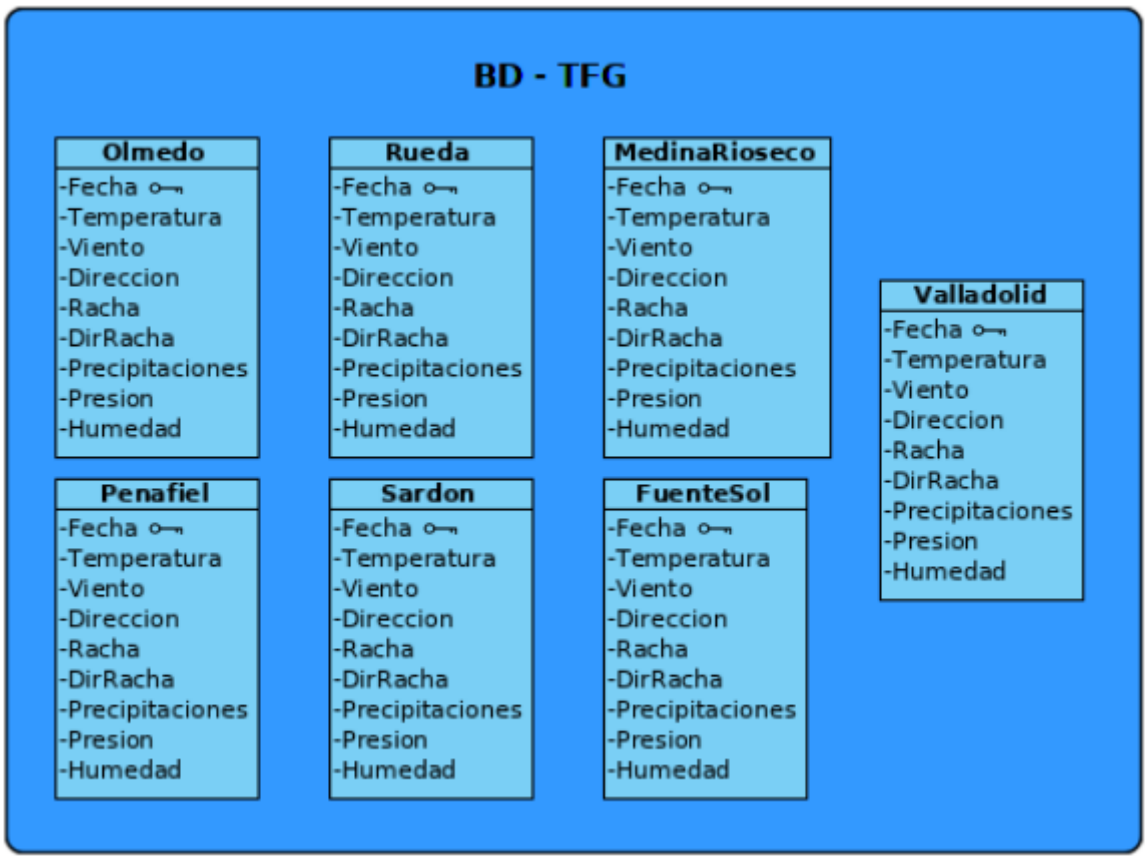


Figura 7: estructura base de datos

5.2.3 Diagrama de clases de diseño

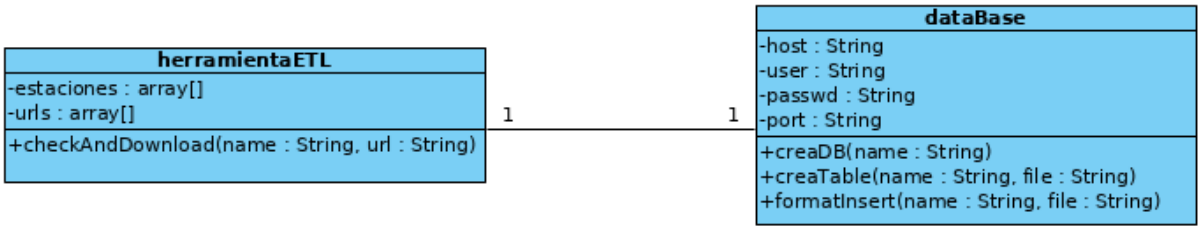


Figura 8: diagrama de clases primer sprint

5.2.4 Diagrama de secuencia del caso de uso

A continuación, se muestra el diagrama de secuencia del caso de uso de extracción, transformación y almacenamiento de datos.

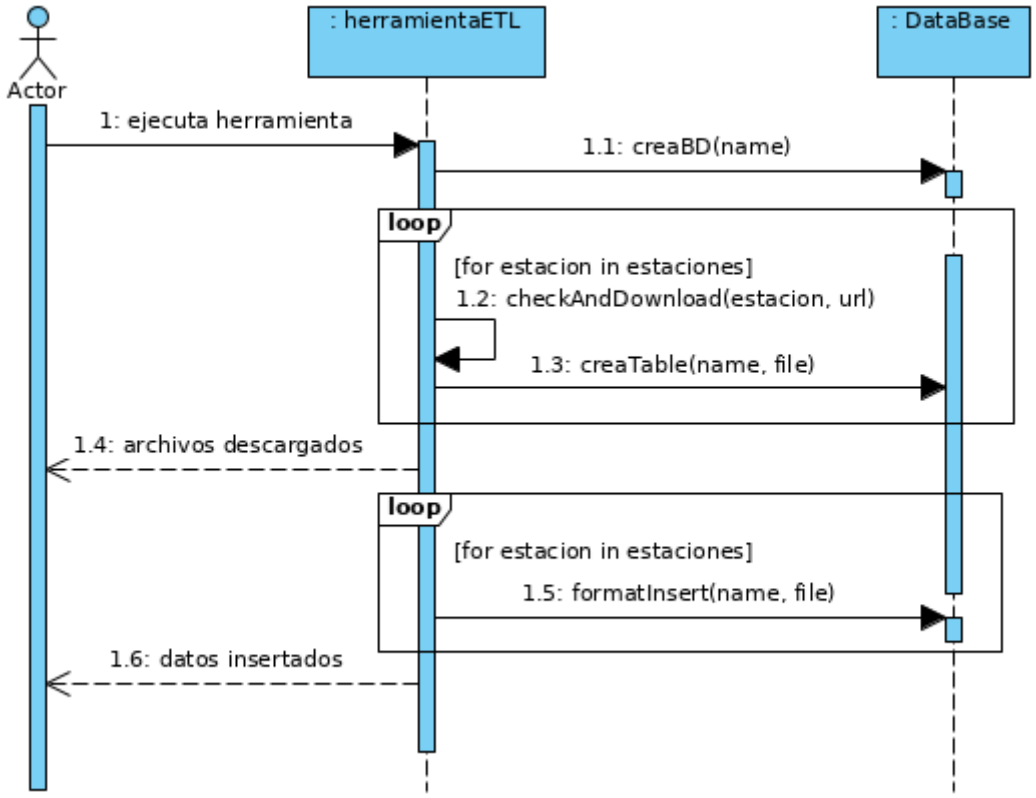


Figura 9: diagrama de secuencia caso de uso ETL de datos meteorológicos

5.3 Desarrollo

Para poder empezar con el desarrollo del TFG el primer trabajo identificado es establecer el entorno de desarrollo gracias al cual podremos ejecutar las tareas necesarias para completar el Trabajo de Fin de Grado.

Este entorno de desarrollo estará formado básicamente por dos elementos:

- Editor de código: para la realización del proyecto se ha utilizado Visual Studio Code como editor de código.

- Servidor de bases de datos: se ha procedido a la instalación de MySQL como gestor de bases de datos a usar durante el TFG.

Una vez instalado el entorno de desarrollo y estando este operativo para su uso, se ha procedido a la instalación de Python, el lenguaje de programación junto con su intérprete con el que se implementarán todos los trabajos.

Para la obtención de los datos meteorológicos es necesario conocer los enlaces disponibles en la web de la AEMET que nos permiten descargarlos.

Una vez identificadas todas las estaciones meteorológicas y los enlaces a los archivos en formato CSV, se ha procedido a la creación de un pequeño *Script* en Python con el uso de la biblioteca *wget* gracias al cual se pueden descargar en una ruta local a elegir cada uno de los archivos CSV de las distintas estaciones meteorológicas.

Hasta este punto podemos decir que disponemos de la parte de extracción de los datos meteorológicos, los cuales se encuentran en archivos CSV siendo ésta una ventaja gracias a las facilidades para trabajar con este tipo de formato que ofrecen bibliotecas como *CSV* de Python, que ha sido la elegida para la lectura y tratamiento de este tipo de archivos.

La biblioteca CSV nos permite procesar los archivos como si estuviésemos usando un programa específico como Excel u otro similar con la capacidad de escoger las variables meteorológicas deseadas según su fecha de medición, de forma que podemos modificar el formato de los valores y agruparlos de la forma adecuada para su posterior inserción en la base de datos.

Para poder acceder al servidor de bases de datos MySQL instalado en local con Python, es necesaria la instalación de un controlador (driver) para la conexión. El driver elegido para realizar esta conexión ha sido MySQL Connector[6] facilitando todas las operaciones realizadas con el servidor de bases de datos.

Con el uso de MySQL Connector se implementa con Python un proceso para la creación de la base de datos del TFG junto con las distintas tablas de las estaciones meteorológicas a usar y sus configuraciones, que han sido previamente mostradas en la parte de diseño de la base de datos. Este proceso se ha desarrollado pensando en la escalabilidad del sistema y al igual que para el proceso de extracción de datos solo se requerirá de la inclusión del URL del archivo CSV de la nueva estación meteorológica, el nombre de la estación y una llamada a la función para crear una nueva tabla en la base de datos.

Con la sección de extracción y tratamiento de datos configurada por un lado y la herramienta de almacenamiento por otro lado, el último trabajo a realizar es la inserción o carga de los datos meteorológicos en sus respectivas tablas.

El proceso de inserción requiere del uso de la sintaxis básica de MySQL para la ejecución de esta operación. Desde Python gracias al driver instalado se ha creado un método para la inserción de los datos meteorológicos previamente tratados y en un formato almacenable por MySQL, de forma que se pueda cumplir especialmente el uso de la clave primaria para el campo Fecha (formato datetime) evitando duplicados en las inserciones. Este método al igual que los de creación de tablas podrá ser llamado por cada una de las estaciones existentes o que se deseen incluir logrando que el programa global para el proceso ETL (Extracción + Transformación + Carga) sea escalable.

Finalmente, para facilitar la inclusión de nuevas estaciones se ha configurado la herramienta ETL creada para obtener el nombre de la base de datos a usar o crear si se trata de la primera vez de uso de la herramienta, los nombres de las estaciones y los URL de sus respectivos archivos CSV de un archivo de texto, de forma que, en caso de querer añadir una nueva estación solo habrá que añadir su nombre y el URL para descargar el archivo CSV.

5.4 Pruebas

Durante el desarrollo del primer sprint las pruebas realizadas han estado enfocadas en comprobar si los resultados obtenidos con el programa creado para ejecutar el proceso ETL son satisfactorios. A continuación, se muestran los ejemplos de pruebas realizadas para asegurar el correcto funcionamiento de los trabajos realizados.

Prueba 1 primer *sprint*	**Extracción de datos meteorológicos**
Descripción	Descarga de los archivos CSV mediante con la herramienta ETL.
Resultado esperado	El programa indica que se han descargado los archivos y estos se guardan en la ruta indicada.
Resultado	Correcto.

Tabla 5: prueba 1 primer sprint

Prueba 2 primer *sprint*	**Creación de la base de datos**
Descripción	Creación de la base de datos y de sus tablas mediante la herramienta ETL.
Resultado esperado	Se crea una tabla para cada estación incluida en la herramienta si no se ha creado con anterioridad.
Resultado	Incorrecto.
Causa	Inserta tablas duplicadas al no estar correctamente implementado el control de tablas.
Solución	Añadir un filtro en la sentencia MySQL, de forma que se crea la tabla solo si no existe. Sentencia: "*CREATE TABLE IF NOT EXISTS nombre_estacion*"

Tabla 6: prueba 2 primer sprint

Prueba 3 primer *sprint*	**Inserción de datos**
Descripción	Inserción de los datos meteorológicos ya tratados en la base de datos mediante la herramienta ETL.

Resultado esperado	Se insertan los datos correctamente en sus respectivas tablas de la base de datos.
Resultado	Correcto.

Tabla 7: prueba 3 primer sprint

6. Segundo *sprint*: automatización del proceso ETL

El segundo *sprint* tiene como objetivo principal la automatización de la ejecución de herramienta que realiza el proceso creado en el primer *sprint* para la extracción, transformación y almacenamiento o carga de datos.

Para la automatización de este proceso ETL ha sido necesario el uso de una máquina virtual que ha sido proporcionada por la Escuela de Ingeniería Informática y gracias a la cual se ha podido programar este proceso ETL, posteriormente a la correcta configuración de la máquina.

Una vez completado el sprint el proceso ETL deberá ejecutarse de forma periódica sin la necesidad de interacción humana.

En las siguientes secciones se muestran los procesos de análisis, diseño, desarrollo y pruebas realizadas durante este *sprint*.

6.1 Análisis

Durante la reunión de seguimiento con el tutor se han concretado los siguientes requisitos a cumplir con la realización de este *sprint:*

Nº de requisito	**Descripción**	**Crítico**
1	La configuración de la máquina virtual permitirá acceder remotamente a ésta	Sí
2	Los desarrollos creados en el primer *sprint* serán replicados en la máquina virtual	Sí
3	La ejecución del programa que desempeña el proceso ETL debe auto programarse.	Sí
4	Los datos meteorológicos deberán ser accesibles desde un dispositivo externo a la máquina virtual.	Sí
5	La periodicidad con la que se ejecuta el programa que desempeña el proceso ETL debe ser de al menos 2 veces diarias para evitar posibles pérdidas de datos.	No
6	Se crearán copias de seguridad de los datos almacenados.	No

Tabla 8: requisitos segundo sprint

6.1.1 Modelo de dominio

En el siguiente modelo de dominio podemos ver cada una de las entidades identificadas para la resolución del *sprint*.

Como se puede comprobar es prácticamente igual al del primer *sprint* a excepción de la inclusión del administrador de procesos Cron que se encargará de la ejecución del programa.

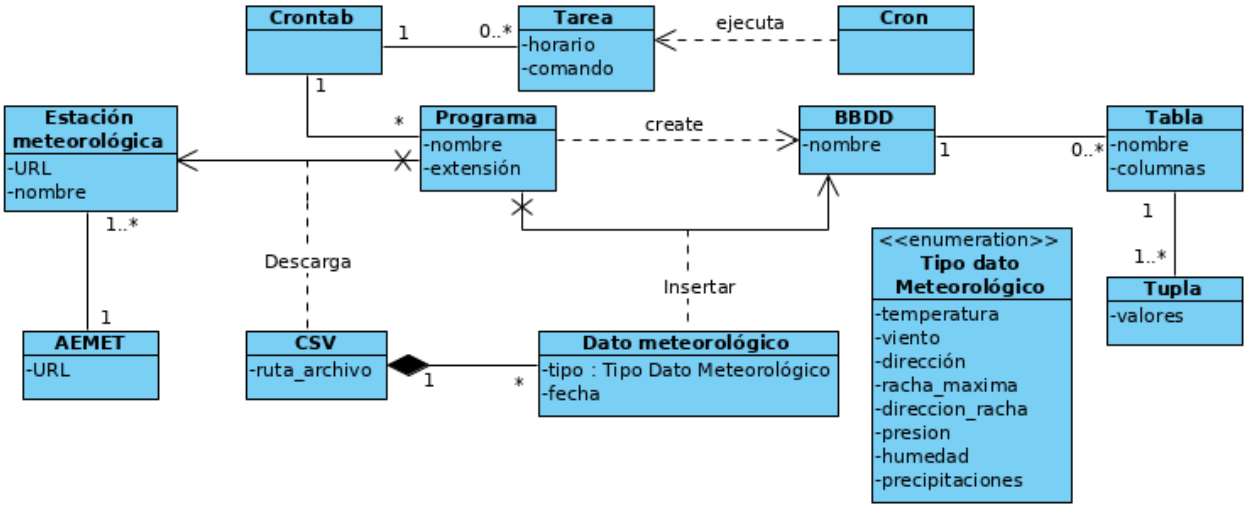


Figura 10: modelo de dominio segundo sprint

En este *sprint* no se identifica ningún caso de uso, ya que el proceso ETL que se ejecutaba anteriormente a mano presentando el único caso de uso, en esta ocasión es ejecutado por el administrador de procesos Cron.

6.2 Diseño

El diseño del sistema creado va a tener la misma arquitectura *cliente-servidor* que en el primer *sprint* con la única diferencia de que el nivel de cliente anteriormente se encontraba en el ordenador local del alumno, mientras que ahora el sistema va a estar instalado sobre la máquina virtual proporcionada por la Escuela de Ingeniería Informática de Valladolid, por lo que el único cambio lo observamos a nivel del cliente.

Al igual que en el primer *sprint* podemos diferenciar claramente un proveedor o servidor, en este caso de los datos meteorológicos que sería “la AEMET”, y el cliente que es la base de datos creada en la máquina virtual.

Diagrama de despliegue:

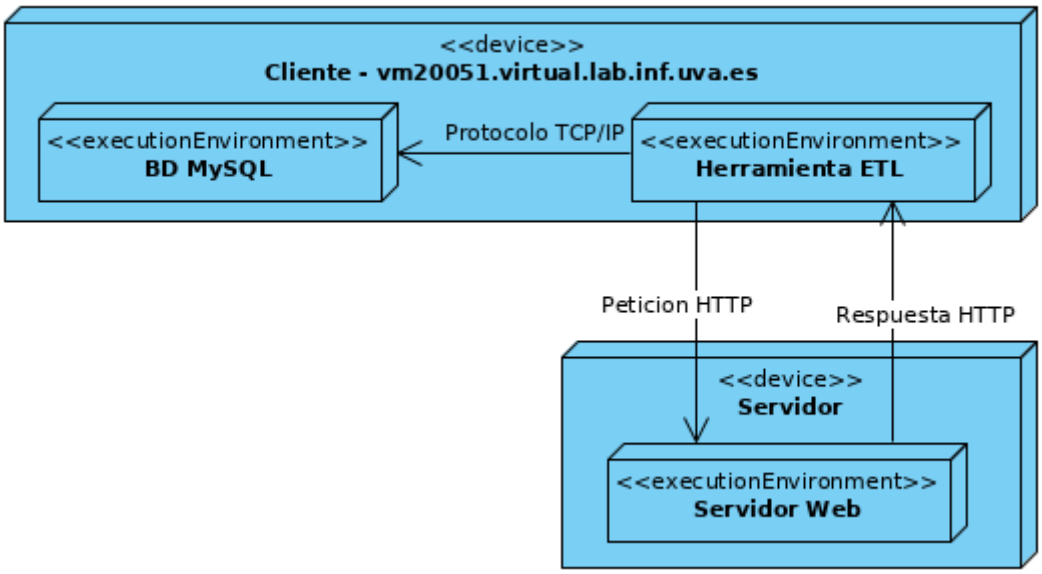


Figura 11: diagrama de despliegue segundo sprint

En la figura anterior podemos ver claramente la diferencia en el nivel de cliente con respecto al primer *sprint*, donde el dispositivo "físico" o parte "hardware" es la máquina virtual usada.

6.3 Desarrollo

La máquina virtual proporcionada por la Escuela de Ingeniería Informática de Valladolid, se halla bajo el dominio virtual.lab.inf.uva.es mientras que los puertos de acceso son 20051 para conexión por SSH, el puerto 20052 para conexión HTTP y el 20053 para conexión MySQL.

Acceso a la máquina por SSH e instalación del servidor de bases de datos y herramienta ETL

El acceso mediante SSH a la máquina virtual al puerto 20051 y con el usuario y contraseña indicados por la Escuela de Ingeniería Informática se realiza mediante el siguiente comando:

ssh -p 20051 usuario@virtual.lab.inf.uva.es

Una vez se ha logrado el acceso a la máquina virtual se procede a la instalación del servidor de bases de datos MySQL como primer paso para replicar los desarrollos del primer *sprint.*

En el directorio */home/usuario* de la máquina virtual se crea el programa Python encargado de ejecutar el proceso ETL (herramienta ETL) con el mismo código que se ha usado en el primer *sprint.* Para lograr que el programa funcione correctamente se ha comprobado la versión de Python instalada en la máquina virtual, se ha instalado el driver MySQL Connector e importado las bibliotecas *wget* y *CSV.*

Los desarrollos del primer *sprint* estarían replicados en este punto con la instalación del servidor de bases de datos MySQL y la creación del programa Python para el proceso ETL. El siguiente paso, el cual es el principal objetivo de este *sprint* es la automatización del proceso.

Automatización del proceso ETL

La ejecución de forma automática de un programa o proceso en una máquina o dispositivo con un sistema operativo tipo Linux, como es en este caso con Ubuntu 18.04.4 LTS, requiere del uso del administrador de procesos Cron[7].

Cron nos permite programar la ejecución de un programa o proceso en una fecha y momento determinados, pero para que esto sea posible la tarea de ejecución del programa en el momento y fecha elegidos debe incluirse en el archivo /etc/*crontab*. El archivo */etc/crontab* es comprobado por Cron cada minuto en busca de tareas para su ejecución.

Para programar la ejecución del programa Python se incluye la siguiente sentencia en el archivo */etc/crontab*:

*0 0,8,16 * * * usuario python /home/usuario/herramientaETL.py estaciones.txt*

En la sentencia anterior estamos indicando que se debe ejecutar el programa */home/usuario/herramientaETL.py* usando las estaciones del archivo *estaciones.txt* a las 00:00, 08:00 y 16:00 cada día de la semana, en todos los días del mes y en cada mes del año:

- El primer 0 nos indica los minutos
- Los números 0,8,16 son las horas a las que debe ejecutarse el programa
- El primer asterisco indica que debe ejecutarse todos los días del mes
- El segundo asterisco que debe ejecutarse todos los meses
- El tercer y último asterisco que debe ejecutarse cada día de la semana

Acceso remoto a la base de datos

Para poder acceder remotamente al servidor de bases de datos MySQL instalado en la máquina virtual hay que realizar dos configuraciones[8]:

1. Modificar los permisos del usuario o usuarios MySQL elegidos: en el caso del presente TFG se ha procedido a modificar los permisos de acceso del usuario *root* de forma que tenga acceso desde cualquier host. Para ello accedemos a la base de datos con el usuario *root* y ejecutamos las siguientes sentencias:

   *GRANT ALL PRIVILEGES ON *.* TO root@'%' IDENTIFIED BY 'contraseña';*

FLUSH PRIVILEGES;

Con la primera sentencia se ha indicado que el usuario root tiene permisos para acceder remotamente desde cualquier IP, indicado con el '%' que significa cualquier valor de IP, y a cualquier base de datos y tablas del servidor, que se indica con "*.*" donde el primer asterisco nos indica que se puede acceder a cualquier base de datos y el segundo a cualquier tabla de las bases de datos. La segunda sentencia indica que los cambios realizados en el usuario *root* se han actualizado

El resultado de estas operaciones si ejecutamos el comando select user,host from mysql.user; es:

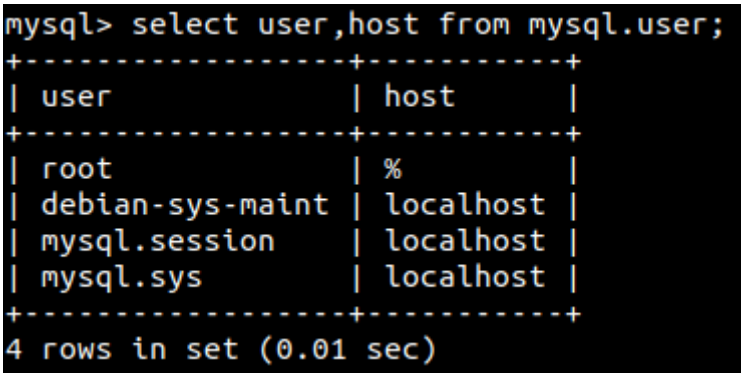


Figura 12: usuarios MySQL y permisos de acceso

En la figura 12 se puede ver cómo el usuario *root* puede acceder desde cualquier host al tener asignado '%' en el campo *host*.

2. Modificar el archivo */etc/mysql/mysql.cnf* y reiniciar el servicio MySQL: se procede a la modificación de la línea donde encontramos la sentencia *bind-address* y se incluye lo siguiente:

   bind-address = 0.0.0.0

   Con la anterior sentencia estamos indicando al servicio MySQL que escuche conexiones en el puerto predefinido de MySQL desde cualquier IP (0.0.0.0 - cualquier dirección es válida).

   Con la siguiente sentencia reiniciamos el servicio MySQL y ya tendríamos completada la configuración para acceder remotamente con el usuario *root* desde cualquier IP y a cualquier base de datos:

   sudo service mysql restart

Una vez realizados los pasos nombrados anteriormente ya podemos probar la conexión remota al servidor MySQL de la máquina virtual. Para realizar la conexión utilizamos el siguiente comando:

mysql -u root -p --host=virtual.lab.inf.uva.es --port=20053

Sistema de BackUps de la base de datos

Como medida de seguridad para evitar la pérdida de los datos meteorológicos durante el trabajo con los mismos, se ha procedido a implementar un sistema de copias de seguridad mediante un pequeño *script bash* que se encarga de mantener copias de seguridad de la base de datos de los últimos 7 días.

Este script se ha programado para su ejecución con Cron cada día a las 23:59 horas.

El script:

```
#!/bin/bash

now=$(date '+%Y-%m-%d') #Fecha actual
clean=$(date --date='-7 days' '+%Y-%m-%d')          #Fecha de hace 7 días

if [ -f /home/usuario/$clean.sql.gz ];  #Comprueba si existe un copia de hace 7 días
then
rm /home/usuario/$clean.sql.gz                  #Si existe la copia, la borra
fi

#Genera la copia de seguridad con el nombre de la fecha de hoy
mysqldump --user=root --password=contraseña tfg | gzip >/home/usuario/$now.sql.gz
```

Este mecanismo de seguridad se ha implementado para cubrir la opción de que durante el trabajo con los datos meteorológicos estos puedan ser alterados o borrados, siendo posible gracias a estas copias de la base de datos restaurar los originales.

6.4 Pruebas

Para mostrar las pruebas realizadas durante el desarrollo de este sprint se ha creado la siguiente tabla donde se pueden visualizar los distintos ensayos y sus resultados:

Prueba 1 segundo *sprint*	**Ejecución de la herramienta ETL**
Descripción	Ejecución de la herramienta que ejerce el proceso ETL en la máquina virtual.
Resultado esperado	Se extraen, tratan y cargan correctamente los datos meteorológicos en el servidor instalado en la máquina virtual.
Resultado	Correcto.

Tabla 9: prueba 1 segundo sprint

Prueba 2 segundo *sprint*	**Automatización del proceso ETL**
Descripción	Cron ejecuta la herramienta de ETL.
Resultado esperado	Se extraen, tratan y cargan correctamente los datos meteorológicos en el servidor instalado en la máquina virtual en el momento indicado en el archivo crontab de forma automática.
Resultado	Incorrecto.
Causa	No se ha indicado correctamente la ruta del programa en el archivo crontab.
Solución	Se modifica la ruta del programa que sirve como herramienta ETL.

Tabla 10: prueba 2 segundo sprint

Prueba 3 segundo *sprint*	**Automatización de las copias de la base de datos**
Descripción	Cron ejecuta el Script para realizar las copias de la base de datos.
Resultado esperado	Se crea una copia diaria y se mantienen las siete últimas copias realizadas.
Resultado	Incorrecto.
Causa	La copia de la base de datos no se guarda con el nombre esperado.
Solución	Establecer el formato de la fecha correctamente.

Tabla 11: prueba 3 segundo sprint

7. Tercer *sprint*: GUI para la conexión con la base de datos

El principal objetivo de este *sprint* es realizar una interfaz gráfica de usuario que permita configurar una conexión con una base de datos MySQL ya sea de forma remota o localmente.

El desarrollo de una interfaz gráfica requerirá de la elección de una biblioteca de Python que dé soporte a la creación de interfaces, además de la familiarización con el uso de la biblioteca elegida.

Como resultado final del *sprint* se espera que una vez se pueda configurar una conexión con la base de datos MySQL elegida, dicha conexión quede preparada para poder desempeñar un trabajo futuro con ella.

7.1 Análisis

Los requisitos o necesidades funcionales que se deben conseguir con la realización de este *sprint* son los siguientes*:*

Nº de requisito	**Descripción**	**Crítico**
1	La biblioteca elegida deberá proporcionar soporte para la creación de interfaces gráficas.	Sí
2	La conexión deberá ser iniciada por el usuario al indicarlo mediante algún tipo de interacción con la interfaz gráfica.	Sí
3	Los datos de conexión deben ser modificables desde la interfaz.	Sí
4	La interfaz gráfica mostrará si se ha podido establecer la conexión.	Sí
5	La interfaz gráfica mostrará el error de conexión producido en caso de no establecerse conexión.	Sí
6	La conexión debe poder realizarse con diferentes base de datos MySQL	Sí
7	El diseño de la interfaz gráfica tiene que ser intuitivo para el usuario	No
8	La contraseña de conexión con la base de datos debe mostrarse oculta si es posible	No

Tabla 12: requisitos tercer sprint

7.1.1 Modelo de dominio

Tras realizar el análisis de requisitos del tercer *sprint*, se ha procedido a la creación del modelo de dominio del mismo:

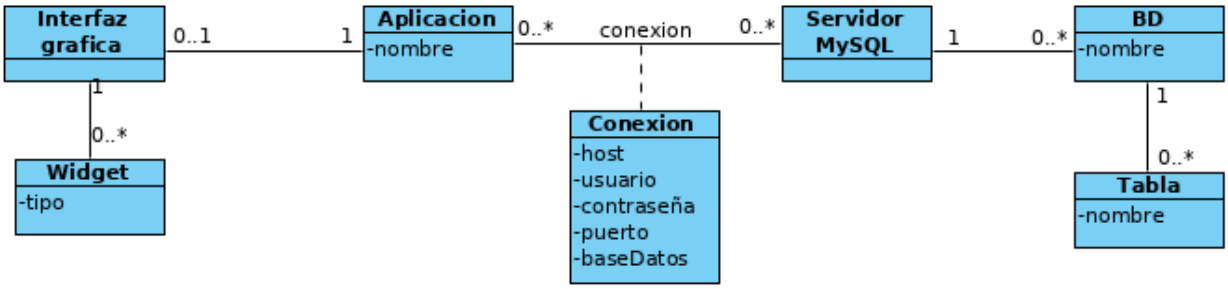


Figura 13: modelo de dominio tercer sprint

7.1.2 Casos de uso

En el tercer sprint solo podemos considerar un único caso de uso, ya que la única actividad que va a poder realizar el usuario es la configuración de la conexión con la base de datos. A continuación, se mostrará el caso de uso junto con la secuencia de interacción:

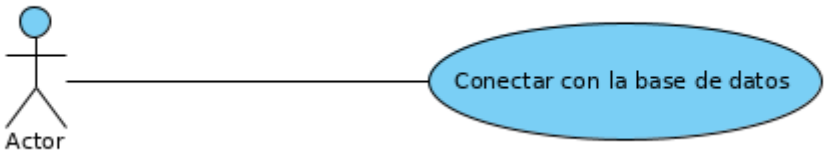


Figura 14: diagrama de caso de uso tercer sprint

Caso de uso tercer *sprint*		
Título	Conectar con la base de datos.	
Actor	Usuario.	
Descripción	La interfaz gráfica creada deberá permitir configurar una conexión con la base datos MySQL elegida por el usuario.	
Secuencia normal	**Pasos**	**Acción**
	1	El actor ejecuta la aplicación.

	2	El sistema solicita completar los campos necesarios para configurar la conexión (host, usuario, contraseña, puerto, base de datos).	
	3	El actor introduce los datos solicitados.	
	4	El actor solicita realizar la conexión.	
	5	El sistema crea la conexión y notifica al actor que se ha conectado con éxito a la base de datos seleccionada.	
Excepciones	**Pasos**	**Acción**	
	3	El actor cierra la ventana y acaba el caso de uso.	
	4.1	El actor cierra la ventana y acaba el caso de uso.	
	4.2	Si el actor ha dejado sin completar algún campo obligatorio para la conexión.	
		4.2.1	El sistema indica al actor que debe completar el campo o campos que están vacíos y se vuelve al paso 3.
	4.3	Si alguno de los campos introducidos por el actor es erróneo.	
		4.3.1	El sistema indica al actor qué tipo de error se ha producido al intentar conectar con la base de datos y se vuelve al paso 3.

Tabla 13: caso de uso tercer sprint

7.2 Diseño

El siguiente diagrama de despliegue nos permite comprobar cómo se distribuyen los diferentes componentes de software que van a forma el sistema, físicamente. Podemos diferenciar dos nodos o elementos físicos, siendo uno el ordenador del cliente donde se va ejecutar la aplicación a desarrollar y otro el nodo de la máquina virtual en nuestro caso, ya que trabajaremos con la base de datos alojada en la propia máquina virtual. En caso de usar una base de datos en local solo encontraríamos un único nodo, el del usuario donde se ejecuta la aplicación.

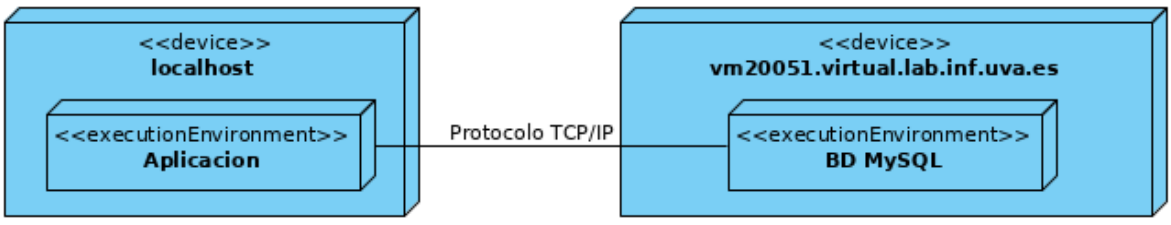


Figura 15: diagrama de despliegue tercer sprint

7.2.1 Modelo-Vista-Controlador

El patrón modelo-vista-controlador (MVC) ha sido el elegido para la arquitectura del software de la aplicación. El uso de este patrón nos permite separar los datos y la lógica de negocio, de la interfaz gráfica de usuario.

Con el uso de este patrón distinguimos tres componentes:

- Modelo: se encarga del acceso a los datos almacenados en la base de datos y de la lógica de negocio o trabajos que se realizarán con los datos para obtener la información necesaria.

- Controlador: es el encargado de procesar los eventos emprendidos por el usuario de la aplicación para poder dar respuesta a sus peticiones.

- Vista: se corresponde con la interfaz gráfica de usuario y muestra el estado del modelo.

El uso de este patrón de arquitectura software ha sido de especial ayuda para probar la funcionalidad deseada que se quería conseguir, ya que el modelo por si solo es completamente funcional, es decir, nos ofrece la posibilidad de hacer las pruebas necesarias para asegurar el correcto funcionamiento antes de tener la interfaz gráfica de usuario implementada.

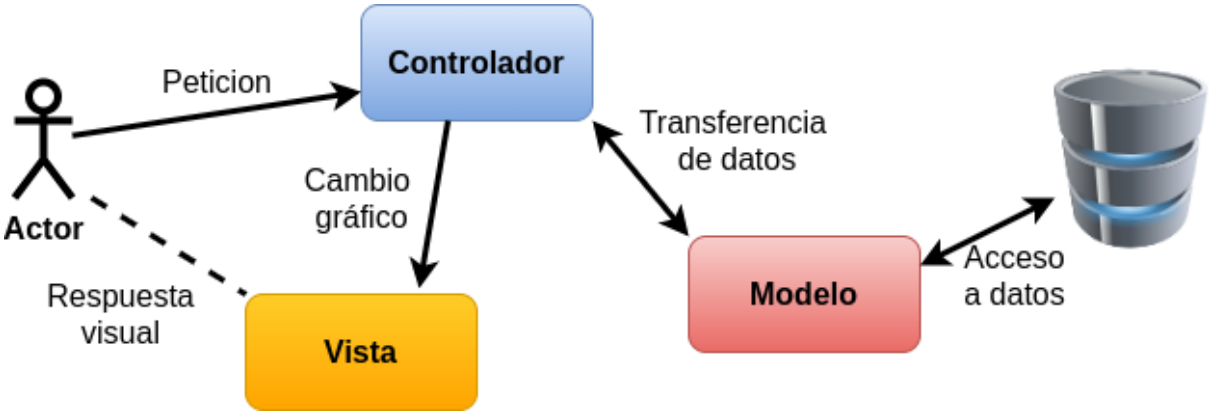


Figura 16: disposición de la lógica del sistema

7.2.2 Diagrama de clases

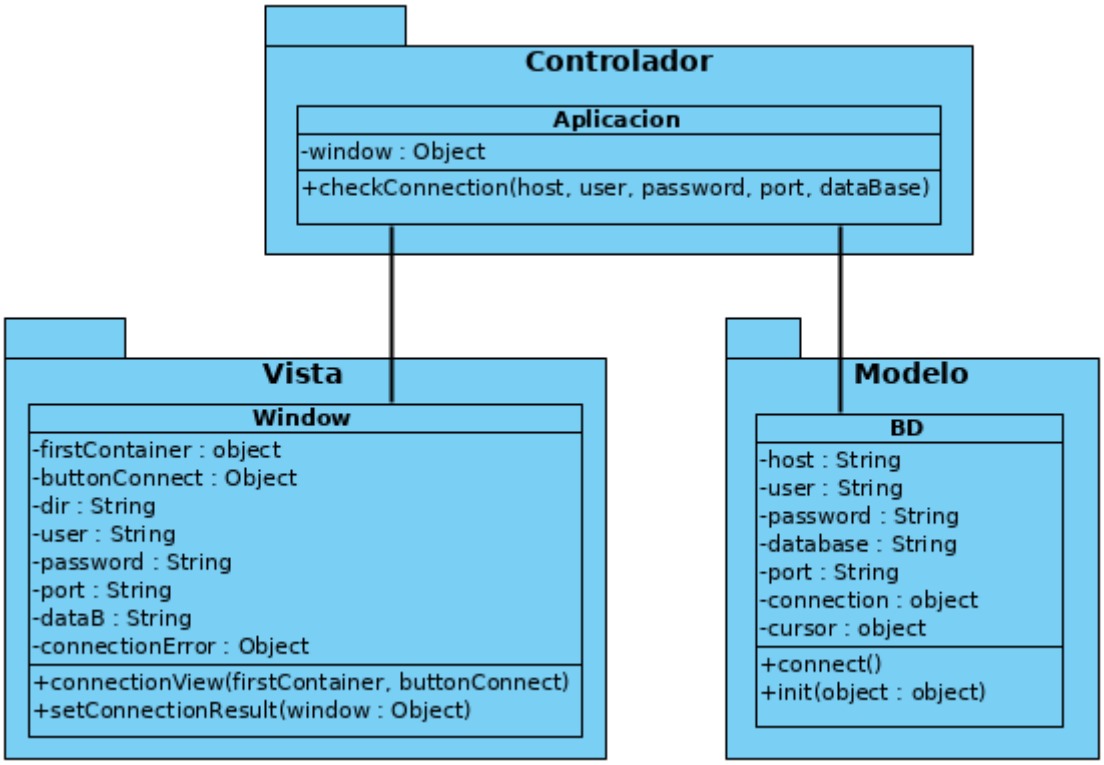


Figura 17: diagrama de clases tercer sprint

7.2.3 Diagrama de secuencia del caso de uso

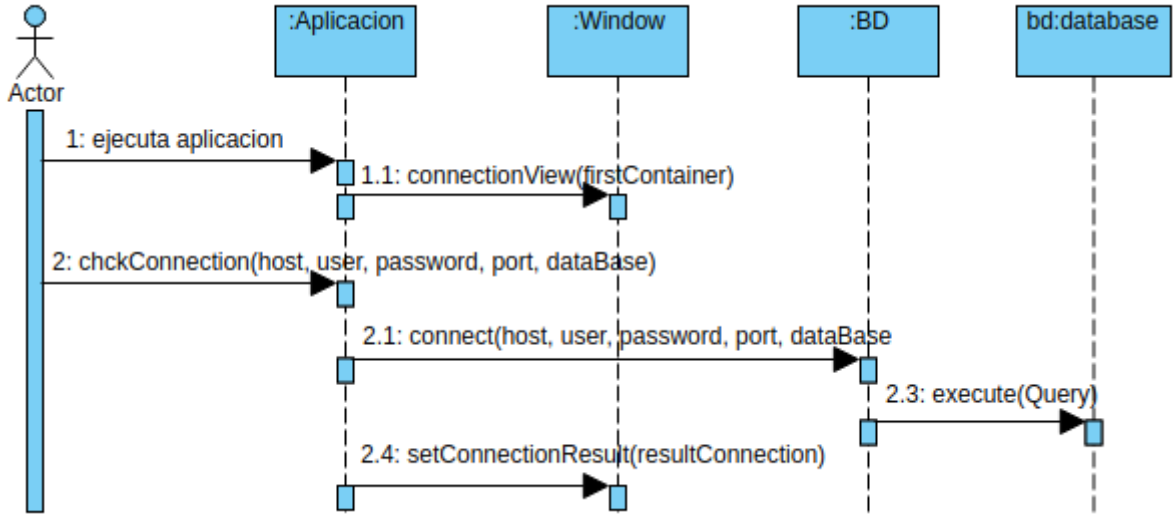


Figura 18: diagrama de secuencia caso de uso conectar con la base de datos

7.3 Desarrollo

Para el desarrollo de la interfaz gráfica se ha escogido como biblioteca gráfica de Python, *Tkinter*. Mediante el uso de esta biblioteca se ha procedido a crear la interfaz gráfica que permite configurar una conexión con una base de datos MySQL cualquiera. La finalización de la interfaz gráfica para configurar la conexión con la base de datos ha requerido de la realización de una serie de trabajos previos.

Diseño de la ventana de conexión

El diseño gráfico de la ventana que se usará para configurar la conexión ha sido el primer trabajo a realizar tras la instalación de *Tkinter*.

Tkinter[9] permite crear ventanas e incluir en éstas widgets como contenedores, botones, entradas de texto, etiquetas… Gracias a las posibilidades que ofrece esta biblioteca se ha podido realizar la interfaz deseada con un diseño personalizado.

La ventana creada con Tkinter tienen unas dimensiones reducidas, ya que únicamente se ha requerido de la inclusión de una serie de etiquetas, los *inputs* (entradas) de texto que permitirán añadir los datos de conexión con el servidor MySQL y un botón para realizar la conexión con los datos incluidos mediante los *inputs.*

El diseño de la interfaz gráfica ha incluido finalmente los siguientes elementos:

- Ventana principal.
- Un *frame* o contenedor principal sobre la ventana principal donde se han incluido el resto de elementos.
- Una *label* o etiqueta de título.
- Un *frame* o contenedor para los datos de conexión.
- Cinco *labels* o etiquetas con información de los campos que se deben completar.
- Cinco *inputs* o entradas de texto, una por cada dato de conexión requerido.
- Una *label* o etiqueta para mostrar los errores que solo es visible una vez se realizar la conexión.
- Un botón para realizar la conexión con la base de datos.

La estructura de la interfaz gráfica tiene el siguiente formato:

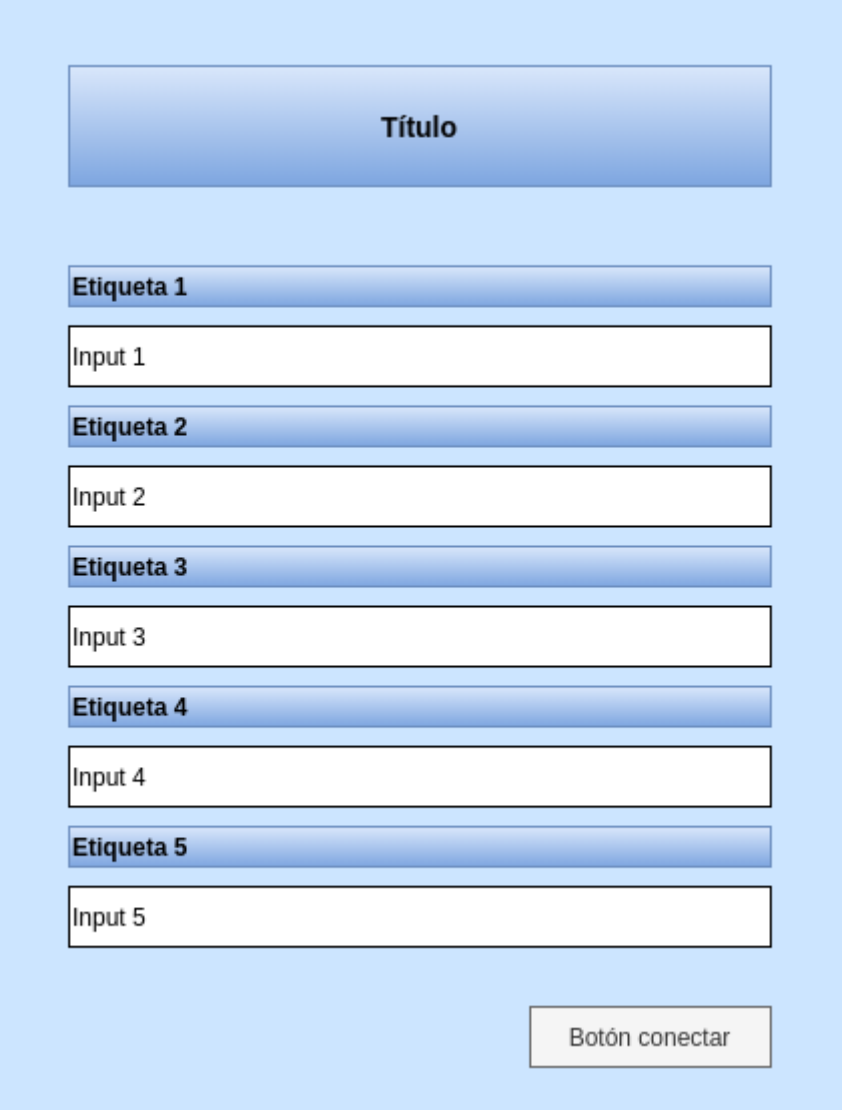


Figura 19: estructura de la interfaz gráfica para la configuración de la conexión

Conexión a la base de datos y mensajes de error

La conexión con la base de datos usando el driver MySQL Connector es el primer paso para poder trabajar con los datos meteorológicos.

Al crear el proceso para la conexión se ha incluido un sistema de notificación del resultado de la conexión usando el sistema de errores propio de MySQL.

La función usada para la conexión:

```
def connect(self):
    try:                                    #Se intenta la conexión
      db = mysql.connect(
        host = 'host',                      #Host de conexión
        user = 'user',                      #Usuario usado
        passwd = 'password',                #Contraseña de usuario
```

```
        port = 'port',                          #Puerto de conexión
        database = 'database',          #Nombre de la base de datos elegida
    );
    self.connection = db;               #Conexión realizada
    self.cursor = db.cursor();          #Cursor para realizar operaciones
    return True;                        #Resultado: conexión exitosa
except mysql.Error as err:              #No se ha podido conectar
    if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
        return 'Usuario o contraseña incorrectos';  #Contraseña o usuario incorrectos
    elif err.errno == errorcode.ER_BAD_DB_ERROR:
        return 'No existe la base de datos especificada';      #No existe base de datos
    else:
        return err;                     #Error predefinido por MySQL
```

Con el código anterior se realiza una conexión con cualquier servidor MySQL. El código nos proporciona la conexión con la base de datos, el resultado de la conexión y la posibilidad de identificar los errores que se hayan producido en la misma, como por ejemplo si el usuario o contraseña son erróneos o si la base de datos seleccionada no existe en el servidor. El resto de errores usados son los predefinidos por MySQL.

Integración de la función de conexión en la interfaz gráfica

Teniendo la función para la conexión e identificación de errores por un lado y la interfaz gráfica por otro, el último paso a realizar es la integración de la función en la propia interfaz.

La función de conexión debe ejecutarse con el suceso de un evento en la interfaz gráfica, siendo en nuestro caso el evento "click" del botón de conexión incluido el elegido para ser asociado a la ejecución de la función para conectar con la base de datos. De esta forma cada vez que se produce el evento de "pulsar" el botón se ejecutará la función. Lo único que falta es sustituir los datos de conexión de la función (*host, user, password, port y database)* por los valores de los *input* o entradas de texto incluidas en la interfaz gráfica.

Por último, se asigna el valor del error o resultado obtenido de la conexión a la etiqueta creada para mostrar este valor y que solo es visible cuando existe un valor para el resultado de conexión.

La última modificación que se incluye es la ocultación de la contraseña evitando que quede expuesta mediante el uso de asteriscos.

7.4 Pruebas

Prueba 1 tercer *sprint*	**Conexión con base de datos**

Descripción	Se configura una conexión con la base de datos alojada en la máquina virtual.
Resultado esperado	La aplicación notifica que se ha conectado correctamente y se obtiene un objeto conexión.
Resultado	Correcto.

Tabla 14: prueba 1 tercer sprint

Prueba 2 tercer *sprint*	**Verificación de formulario**
Descripción	Se indica que se quiere conectar sin haber completado los campos.
Resultado esperado	La aplicación notifica que se deben completar todos los campos del formulario y no realiza la conexión.
Resultado	Correcto.

Tabla 15: prueba 2 tercer sprint

Prueba 3 tercer *sprint*	**Conexión con base de datos alternativa**
Descripción	Se configura una conexión con una base de datos en la máquina local, para comprobar que conecta con diferentes fuentes.
Resultado esperado	La aplicación notifica que se ha conectado correctamente y se obtiene un objeto conexión.
Resultado	Correcto.

Tabla 16*:* prueba 3 tercer sprint

8. Cuarto *sprint*: aplicación principal

En los anteriores *sprints* se ha conseguido desarrollar la herramienta ETL para el almacenamiento de datos meteorológicos de diferentes estaciones en un único espacio y una interfaz para configurar el acceso a estos datos. Ahora el objetivo de este sprint es dar un primer uso a los datos almacenados ya sea mediante la representación de estos o mediante el empleo de cualquier herramienta, de forma que obtengamos diferentes alternativas de interpretar la información que nos ofrecen estos datos meteorológicos.

En concreto durante *sprint* se ha fijado como trabajo a realizar el desarrollo de una aplicación que permita visualizar los datos meteorológicos ofreciendo alguna forma de interactuar con estos y obtener diferentes formas de representarlos.

8.1 Análisis

Los requisitos que se deben cubrir con el trabajo a realizar durante el *sprint* para conseguir los objetivos impuestos en el mismo son los siguientes*:*

Nº de requisito	**Descripción**	**Crítico**
1	La aplicación integrará la ventana de conexión creada en el tercer *sprint*.	Sí
2	Se integrarán herramientas para la visualización de los datos meteorológicos.	Sí
3	La aplicación deberá tener un sistema para seleccionar entre las distintas funcionalidades.	Sí
4	Las herramientas usadas permitirán crear resúmenes de los datos meteorológicos.	Sí
5	Las herramientas usadas permitirán crear gráficos para visualizar los datos meteorológicos.	Sí
6	Se incluirán filtros para que el usuario pueda elegir los datos a visualizar.	No
7	La aplicación deberá tener un diseño *responsive*.	No
8	La aplicación permitirá desconectar de la base de datos usada para configurar una nueva conexión.	No

Tabla 17: requisitos cuarto sprint

8.1.1 Modelo de dominio

A continuación, se muestra el modelo de dominio y clases necesarias identificadas para el desarrollo del cuarto *sprint:*

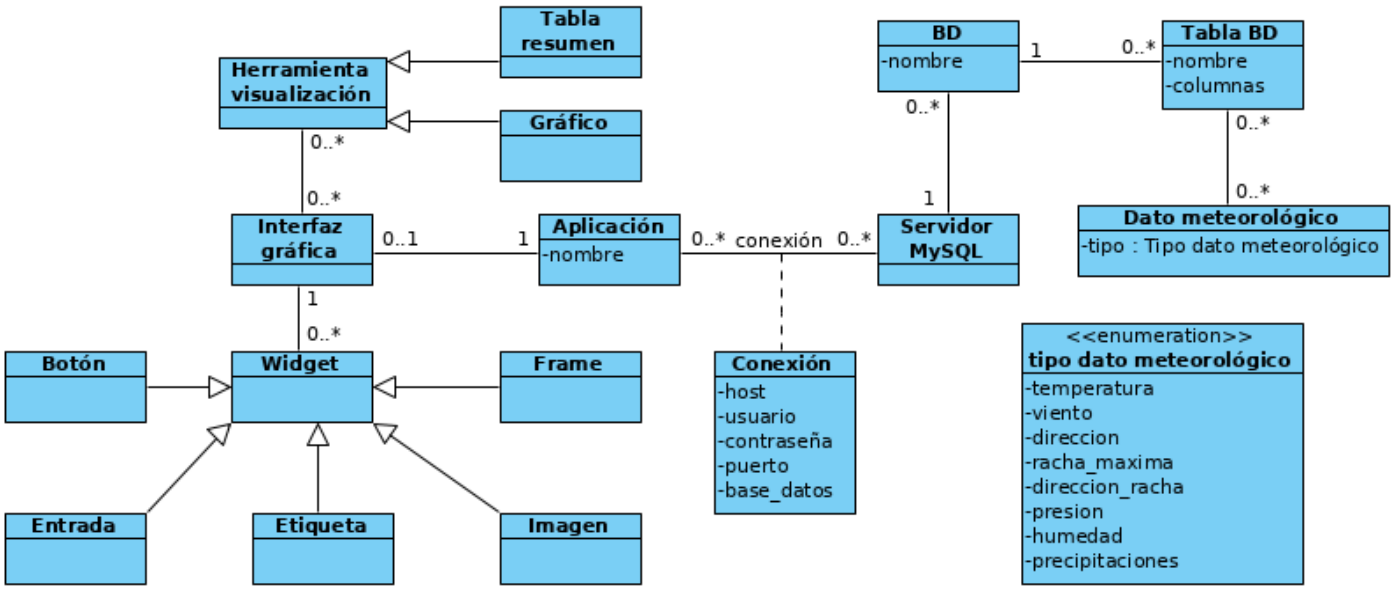


Figura 20: modelo de dominio cuarto sprint

8.1.2 Casos de uso

En el cuarto sprint como usuarios de la aplicación podemos realizar una serie de acciones o actividades interactuando con la aplicación que se detallarán a continuación.

Indicar que todas las posibles acciones que se van a mostrar parten de que la conexión con la base de datos se ha establecido correctamente completando la secuencia normal del caso de uso mostrado en el tercer *sprint*.

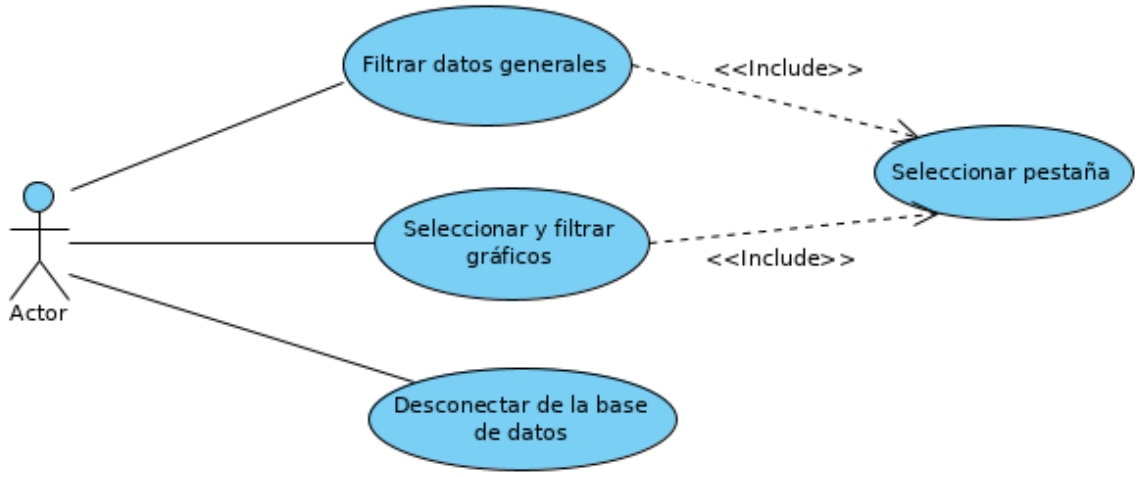


Figura 21: diagrama de casos de uso cuarto sprint

En la anterior figura podemos ver los distintos casos de uso presentes en el sprint 3. El caso de uso "seleccionar pestaña" tiene una relación de "include" (inclusión) con los casos de uso "mostrar gráficos" y "mostrar datos generales" ya que si no hay una pestaña seleccionada no se visualizará ni la sección de "mostrar datos generales" ni de "mostrar gráficos". Para que se puedan dar todos los casos de uso es necesario que se haya realizado la conexión con la base de datos siguiendo el caso de uso presentado en el tercer *sprint*.

1º caso de uso cuarto *sprint*		
Título	Selección de pestaña o funcionalidad.	
Actor	Usuario.	
Descripción	El actor podrá visualizar la pestaña de datos generales o de gráficos según lo elija en el *sidebar* de la aplicación.	
Precondición	Se ha conectado con la base de datos.	
Secuencia normal	**Pasos**	**Acción**
	1	El sistema muestra por defecto (configuración de serie) la pestaña de datos generales y lo indica en el *sidebar*.
	2	El actor selecciona en el *sidebar* la pestaña de gráficos.
	3	El sistema cambia de pestaña, muestra la de gráficos e indica en el *sidebar* que nos hayamos en dicha pestaña.

	4		El actor solicita en el *sidebar* ver la pestaña de datos generales.
	5		El sistema cambia de pestaña, muestra la de datos generales e indica en el *sidebar* que nos hayamos en dicha pestaña.
Excepciones	**Pasos**		**Acción**
	2.1		El actor cierra la ventana y acaba el caso de uso.
	2.2	**2.2.1**	El actor pulsa el botón de desconectar situado en el *sidebar*.
		2.2.2	El sistema vuelve a la vista de conexión con la base de datos y se inicia el caso de uso del tercer *sprint.*
	4.1		El actor cierra la ventana y acaba el caso de uso.
	4.2	**4.2.1**	El actor pulsa el botón de desconectar situado en el *sidebar*.
		4.2.2	El sistema vuelve a la vista de conexión con la base de datos y se inicia el caso de uso del tercer *sprint.*

Tabla 18: 1º caso de uso cuarto sprint

2º caso de uso cuarto *sprint*		
Título	Desconectar de la base de datos.	
Actor	Usuario.	
Descripción	El actor podrá finalizar la conexión actual con la base de datos.	
Precondición	Se ha conectado con la base de datos.	
Secuencia normal	**Pasos**	**Acción**
	1	El sistema muestra la vista principal.
	2	El actor pulsa el botón de desconexión situado en el *sidebar*.
	3	El sistema cierra la vista principal.

	4	El sistema muestra la vista de conexión y se incia el caso de uso del tercer *sprint.*

Tabla 19: 2º caso de uso cuarto sprint

3º caso de uso cuarto *sprint*			
Título	Mostrar datos generales y filtrar por fecha.		
Actor	Usuario.		
Descripción	El actor podrá visualizar una tabla resumen con los datos generales de las distintas estaciones meteorológicas y tendrá la opción de filtrar estos datos por fechas.		
Precondición	Se ha conectado con la base de datos y se ha seleccionado la pestaña de datos generales.		
Secuencia normal	**Pasos**	**Acción**	
	1	El sistema muestra por defecto la tabla con el resumen de todos los datos alojados en la base de datos por estación meteorológica y los filtros de fechas.	
	2	El actor introduce las fechas de inicio y fin para filtrar los datos meteorológicos.	
	3	El actor pulsa el botón de filtrar.	
	4	El sistema muestra la tabla con los datos actualizados.	
Excepciones	**Pasos**	**Acción**	
	2.1	El actor cierra la ventana y acaba el caso de uso.	
	2.2	**2.2.1**	El actor pulsa el botón de desconectar situado en el *sidebar*.
		2.2.2	El sistema vuelve a la vista de conexión con la base de datos y se inicia el caso de uso del tercer *sprint.*
	3.1	Si el actor ha introducido las fechas en un formato incorrecto.	
		3.1.1	El sistema indica al actor que el formato de las fechas es incorrecto y se vuelve al paso 2.

	3.2	Si el actor ha introducido la primera fecha mayor que la segunda.	
		3.2.1	El sistema indica al actor que la primera fecha debe ser menor que la segunda y se vuelve al paso 2.
	3.3	Si para el rango de fechas no hay valores registrados en la base de datos.	
		3.3.1	El sistema indica al actor que no existen datos para las fechas seleccionadas y se vuelve al paso 2.

Tabla 20: 3º caso de uso cuarto sprint

4º caso de uso cuarto *sprint*			
Título	Mostrar gráficos y filtrar sus datos.		
Actor	Usuario.		
Descripción	El actor podrá visualizar una serie de gráficos según sus preferencias y podrá interactuar con estos.		
Precondición	Se ha conectado con la base de datos y se ha seleccionado la pestaña de gráficos.		
Secuencia normal	**Pasos**	**Acción**	
	1	El sistema muestra por defecto el gráfico de temperaturas de la estación de Valladolid para todas las fechas disponibles en la base de datos y los filtros disponibles.	
	2	El actor introduce la estación, la variable meteorológica y las fechas de inicio y fin en el filtro.	
	3	El actor pulsa el botón de filtrar.	
	4	El sistema muestra el gráfico para la estación, la variable y el rango de fechas seleccionado.	
Excepciones	**Pasos**	**Acción**	
	2.1	El actor cierra la ventana y acaba el caso de uso.	
	2.2	**2.2.1**	El actor pulsa el botón de desconectar situado en el *sidebar*.

		2.2.2	El sistema vuelve a la vista de conexión con la base de datos y se inicia el caso de uso del tercer *sprint.*
	3.1	Si el actor no ha introducido o la estación o la variable meteorológica para poder crear el gráfico.	
		3.1.1	El sistema indica al actor que debe seleccionar la estación y la variable meteorológica para poder crear el gráfico y se vuelve al paso 2.
	3.2	Si el actor ha introducido las fechas en un formato incorrecto.	
		3.2.1	El sistema indica al actor que el formato de las fechas es incorrecto y se vuelve al paso 2.
	3.3	Si el actor ha introducido la primera fecha mayor que la segunda.	
		3.3.1	El sistema indica al actor que la primera fecha debe ser menor que la segunda y se vuelve al paso 2.
	3.4	Si para el rango de fechas no hay valores registrados en la base de datos.	
		3.4.1	El sistema indica al actor que no existen datos para las fechas seleccionadas y se vuelve al paso 2.

Tabla 21: 4º caso de uso cuarto sprint

8.2 Diseño

La aplicación principal se ha desarrollado con el patrón de arquitectura de software modelo-vista-controlador (MVC) siguiendo los pasos empezados en el anterior *sprint* para desarrollar la interfaz gráfica para la conexión con la base de datos, la cual es la entrada de esta parte principal de la aplicación. En el apartado 7.2.1 del tercer *sprint* se puede encontrar la explicación del patrón MVC usado.

8.2.1 Diagrama de clases

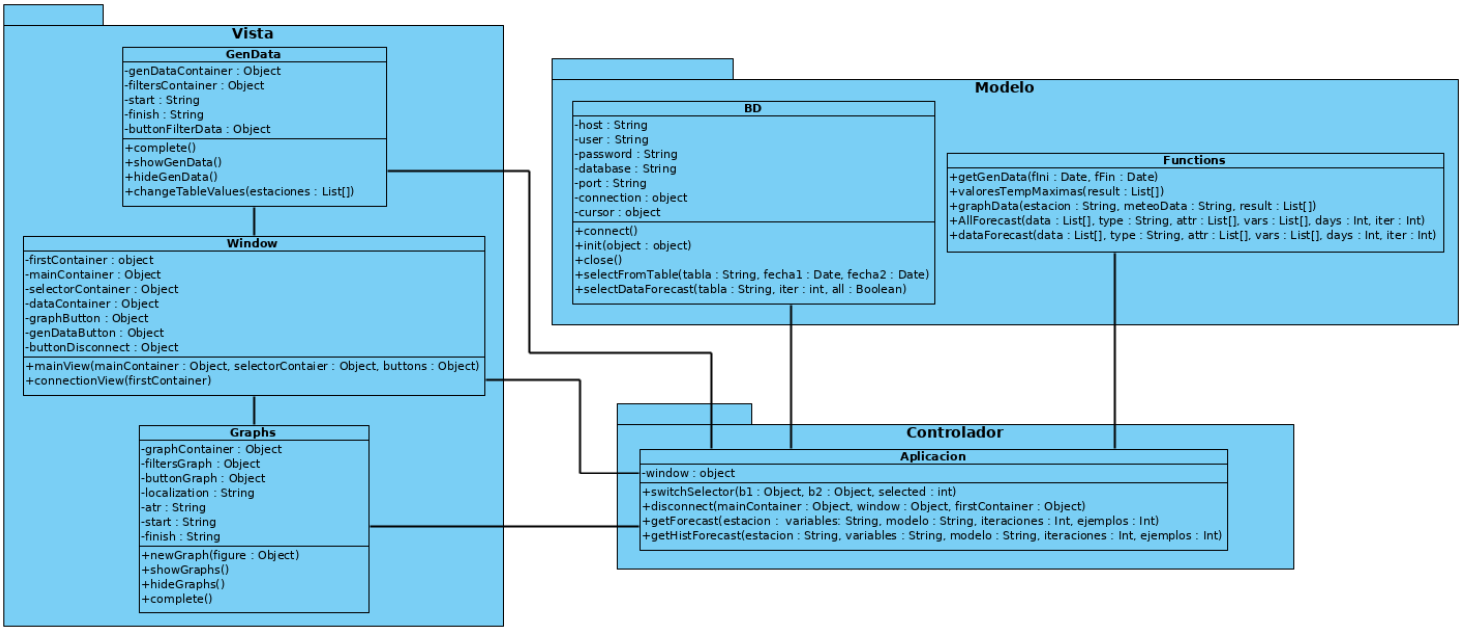


Figura 22: diagrama de clases cuarto sprint

8.2.2 Diagramas de secuencia casos de uso

Caso de uso desconectar de la base de datos

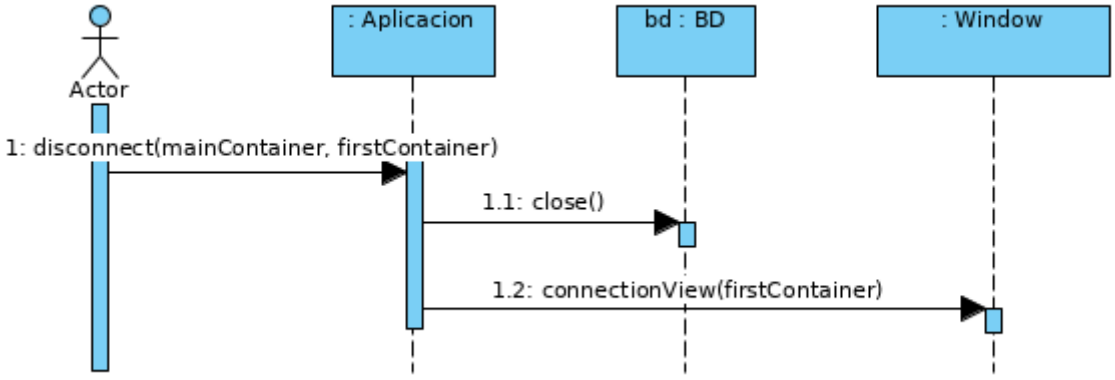


Figura 23: secuencia casos de uso desconectar

Caso de uso seleccionar pestaña

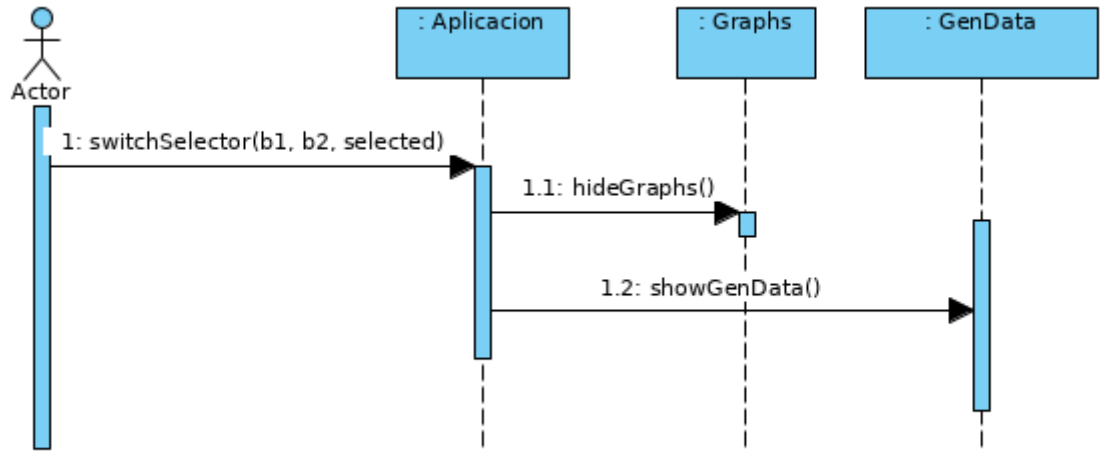


Figura 24: secuencia caso de uso elegir pestaña

Caso de uso mostrar datos generales

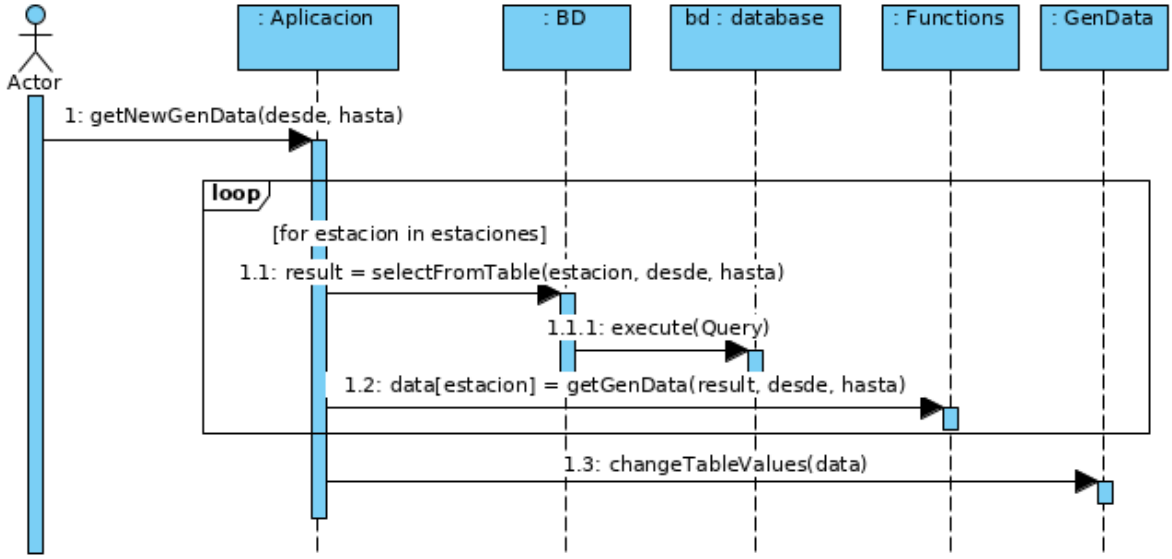


Figura 25: secuencia caso de uso mostrar datos generales

Caso de uso mostrar gráficos

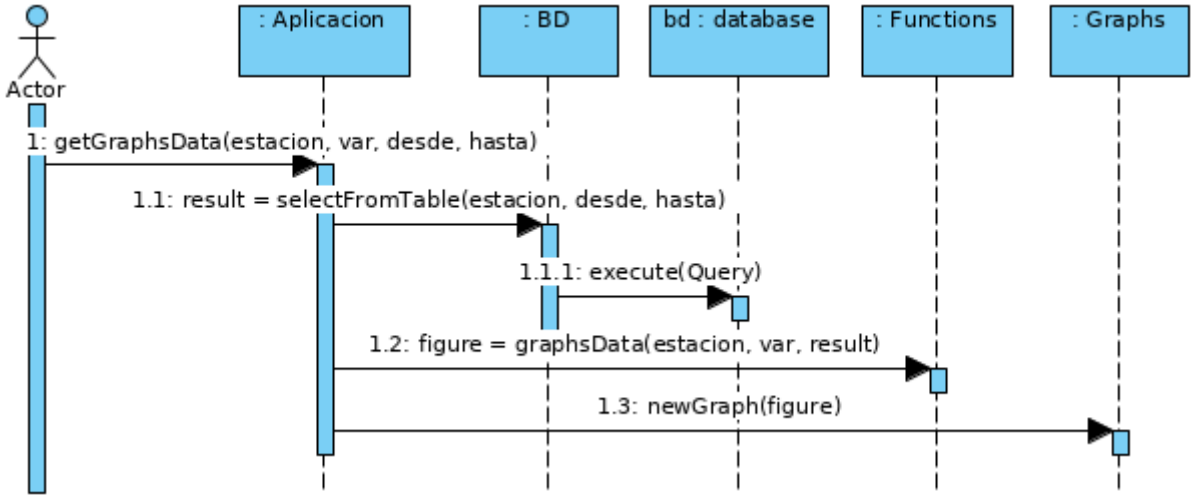


Figura 26: secuencia caso de uso mostrar gráficos

8.3 Desarrollo

El desarrollo del sprint se puede desglosar en tres partes principalmente, el trabajo de diseño de la interfaz gráfica, el trabajo a realizar con los datos meteorológicos para conseguir filtrarlos, agruparlos o seleccionarlos dependiendo de la configuración que se use en las herramientas de visualización, y la integración de las propias herramientas de visualización en la interfaz gráfica.

Diseño de la vista principal de la aplicación

Para ofrecer la mejor visualización de la información, se ha procedido a diseñar una ventana con un diseño *responsive* que se ajuste a los tamaños de las diferentes pantallas en las que se vaya a visualizar la aplicación. El diseño responsive se consigue mediante la asignación adecuada de pesos y porcentajes a los diferentes *frames* (contenedores) que formarán el *grid*[10] (cuadrícula) que compone la vista principal de la aplicación, consiguiendo que se adapten a los diferentes tamaños de las pantallas en las que se vaya a visualizar la aplicación.

Para conseguir el sistema de selección de las diferentes funcionalidades o pestañas, correspondiéndose cada pestaña con un *frame* diferente, se ha creado un *sidebar* o barra lateral desde donde se puede elegir la pestaña a visualizar y donde se encuentra el botón de desconexión con la base de datos. El uso de un *sidebar* a mayores aporta facilidad para incluir nuevos contenidos o vistas en la aplicación, ya que solo se requerirá de añadir una nueva opción en el *sidebar* y el *frame* asociado a dicha opción para tener otra forma de interacción en la aplicación.

En la siguiente imagen se puede observar la estructura de la interfaz gráfica de la vista principal de la aplicación:

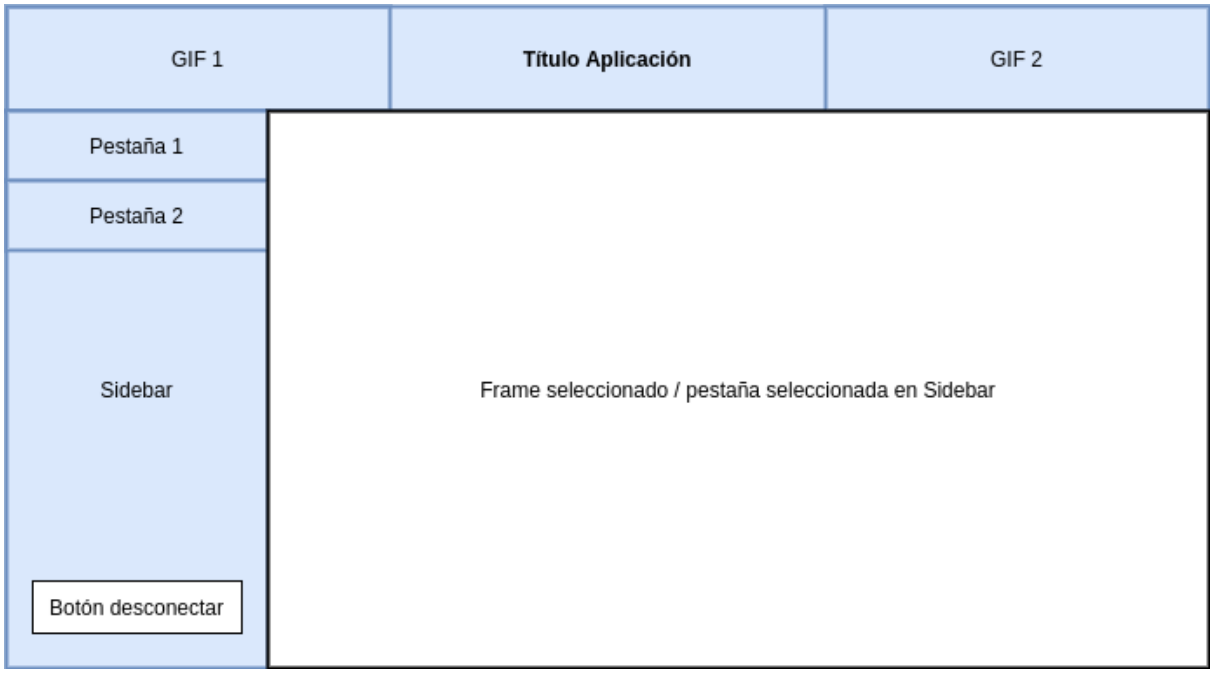


Figura 27: estructura de la vista principal de la aplicación

Operaciones con los datos meteorológicos

Las funcionalidades a incluir en la aplicación son la visualización de una tabla resumen de datos meteorológicos con la posibilidad de filtrar por fecha estos datos y la visualización de gráficos individuales para cada estación meteorológica y variable pudiendo filtrarse también los datos por fechas. Los trabajos y operaciones realizadas con los datos meteorológicos están enfocadas en conseguir las dos funcionalidades descritas, tratándose cada una como un módulo individual.

La tabla resumen de los datos meteorológicos estará constituida por una primera columna con cada una de las estaciones meteorológicas usadas y la primera fila con cada una de las variables a mostrar. Estas variables van a ser la temperatura media, temperatura mínima, temperatura máxima, viento medio, dirección predominante, racha máxima, dirección de racha máxima, precipitaciones totales, precipitaciones máximas, precipitaciones medias, presión media y humedad media.

En base al formato diseñado para la tabla resumen las operaciones a realizar tendrán como finalidad obtener los valores correspondientes a cada una de las variables nombradas anteriormente para cada una de las estaciones meteorológicas y en el rango de fechas indicado en el filtro.

El primer paso ha sido crear un método para obtener los datos meteorológicos de la base de datos alojada en la máquina virtual para cada estación meteorológica incluyendo como variables del método las fechas del rango en el que estarán comprendidos los datos meteorológicos obtenidos. Una vez tenemos los datos con los que trabajar, se ha creado una estructura de datos del tipo diccionario donde se ha incluido cada una de las estaciones y se le ha asignado un valor nulo. La estructura de datos:

```
estaciones = {
    'Valladolid': None,
    'MedinaRioseco': None,
```

```
    'Sardon': None,
    'Penafiel': None,
    'FuenteSol': None,
    'Rueda': None,
    'Olmedo': None,
}
```

Gracias al uso de una función creada para realizar las operaciones aritméticas necesarias para conseguir las variables a incluir en la tabla, se ha podido crear y asignar un array con los datos de las variables a cada estación de la estructura de datos del tipo diccionario creado. La estructura de datos con el array asignado:

```
estaciones = {
    'Valladolid': [tMax, tMedia, tMin, vMedio, dirPredo, rMax, dirRMax, precTot, precMax, precMedia, presMedia, humMedia],
    'MedinaRioseco': [tMax, tMedia, tMin, vMedio, dirPredo, rMax, dirRMax, precTot, precMax, precMedia, presMedia, humMedia],
    'Sardon': [tMax, tMedia, tMin, vMedio, dirPredo, rMax, dirRMax, precTot, precMax, precMedia, presMedia, humMedia],
    'Penafiel': [tMax, tMedia, tMin, vMedio, dirPredo, rMax, dirRMax, precTot, precMax, precMedia, presMedia, humMedia],
    'FuenteSol': [tMax, tMedia, tMin, vMedio, dirPredo, rMax, dirRMax, precTot, precMax, precMedia, presMedia, humMedia],
    'Rueda': [tMax, tMedia, tMin, vMedio, dirPredo, rMax, dirRMax, precTot, precMax, precMedia, presMedia, humMedia],
    'Olmedo': [tMax, tMedia, tMin, vMedio, dirPredo, rMax, dirRMax, precTot, precMax, precMedia, presMedia, humMedia],
}
```

Una vez tenemos la estructura completada con los datos de las variables para cada estación, ya disponemos de todos los datos a mostrar en la tabla resumen de datos generales.

La biblioteca Matplotlib nos permite crear gráficos interactivos que serán usados para cubrir la funcionalidad demandada de representar mediante este tipo de herramientas visuales los datos meteorológicos.

Para obtener los datos necesarios para el gráfico se configura el método creado para obtener los datos meteorológicos que se usa en el proceso de creación de la tabla resumen de datos generales, incluyendo como variable el nombre de la tabla de la cual se quieren obtener los datos meteorológicos además de las fechas del rango para filtrarlos. Esta modificación ha conllevado adaptar el proceso para obtener los datos que se incluirán en la tabla resumen, pero ha permitido usar un único método para cubrir todas las necesidades.

Sobre los datos meteorológicos obtenidos y filtrados por estación y fechas, se ha procedido a usar un método que toma como parámetros los propios datos y la variable que se pretende mostrar en el gráfico de forma que tras incluir las operaciones y selecciones de los datos adecuadas se obtenga el conjunto de datos elegido a representar gráficamente.

Para implementar un gráfico únicamente es necesario indicar al gráfico el conjunto de datos que se quiere representar y configurar la leyenda para distinguir los diferentes datos representados.

En función de la variable meteorológica a representar se han usado distintos tipos de gráficos:

- Temperatura: gráfico de líneas y puntos representando la temperatura media, mínima y máxima diaria.
- Viento: gráfico de líneas y puntos representando el viento medio, mínimo y máximo diario.
- Dirección: gráfico de sectores mostrando los porcentajes de dirección del viento durante el periodo seleccionado en el rango de fechas.
- Precipitaciones: gráfico de barras con las precipitaciones totales diarias.
- Humedad: gráfico de líneas y puntos representando la humedad media, mínima y máxima diaria.
- Presión: gráfico de líneas y puntos representando la presión media, mínima y máxima diaria.

Diseño de la interfaz gráfica de los módulos de datos generales y gráficos

La biblioteca Tkinter no tiene un widget del tipo tabla por lo que para la inclusión del módulo que muestra los datos generales se ha procedido a formar una estructura similar mediante el uso de etiquetas agrupadas a las que se les ha incluido un borde sólido obteniendo una estructura final que ofrece la sensación de un widget del tipo tabla. A las etiquetas se les han asignado los valores de las estaciones, nombres de las variables o valores de la estructura de datos del tipo diccionario creada según correspondiese en función de la posición de las mismas.

El resultado final de la estructura de la interfaz gráfica de la pestaña o módulo de datos generales es el siguiente:

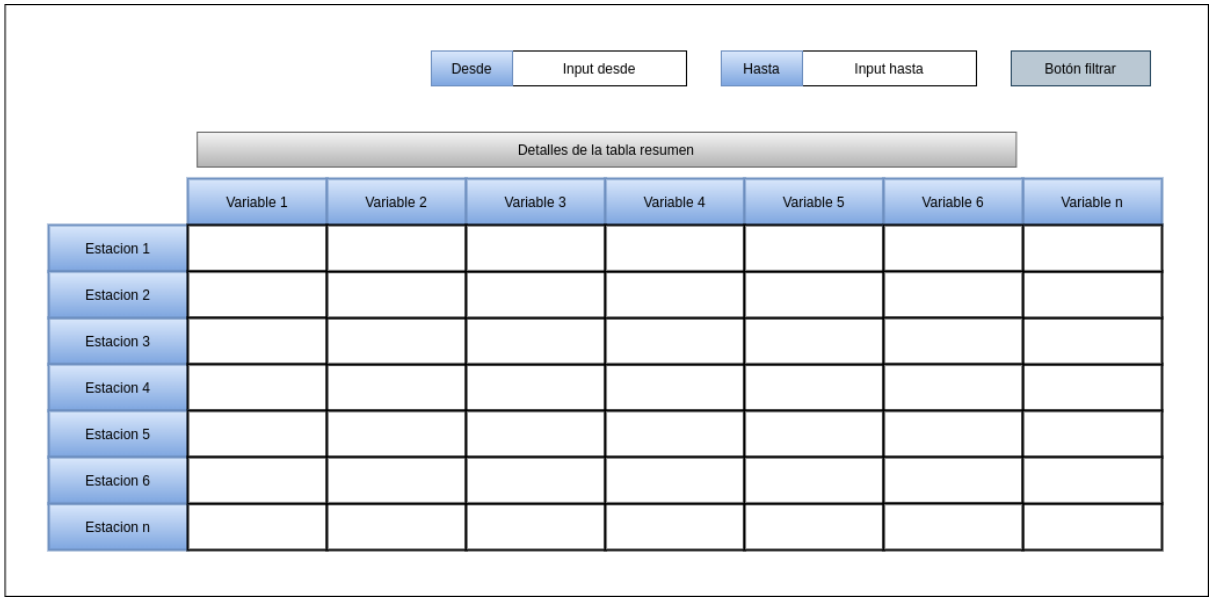


Figura 28: estructura de la interfaz gráfica de la pestaña de datos generales

La integración del módulo de gráficos en la ventana ha requerido únicamente de la creación de un frame y la asignación del gráfico al frame, pero mediante este método el gráfico que se mostraba no era interactivo permitiendo al usuario ampliar o seleccionar ciertos elementos. Para conseguir que el gráfico presentase esas funcionalidades ha sido necesario incluir una barra de herramientas o *toolbar* compatible con los gráficos de la biblioteca Matplotlib. Una extensión de la propia biblioteca Matplotlib[11] ha proporcionado el *toolbar* brindándonos la posibilidad de ejecutar diversas operaciones con los gráficos como puede ser ampliarlos, descargarlos, seleccionar ciertos elementos…

El resultado final de la estructura de la interfaz gráfica de la pestaña de gráficos es el siguiente:

Figura 29: estructura de la interfaz gráfica de la pestaña de gráficos

GIFs

En la parte superior de la vista principal de la aplicación se han incluido GIFs para los que ha sido necesario el uso de hilos. Se ha recurrido a la biblioteca Threading de Python debido a que Tkinter no cuenta con widgets con formato GIF y se ha procedido a realizar el GIF manualmente mediante un bucle infinito de las imágenes que forman cada GIF y el uso de retardos o *delays* para para conseguir reproducir el efecto de animación correctamente.

Tkinter está basado en una metodología "event driver" con un único hilo o *thread* principal que se encuentra en un bucle continuo llamado "event loop" a la espera de un evento que suponga una acción. La inclusión de un GIF que requiere de una ejecución continua supone que, si se usa el hilo principal para este propósito, este nunca se encontrará en el "event loop" esperando la ocurrencia de un evento, por lo que la aplicación habrá perdido toda su funcionalidad.

La solución ha sido incluir hilos o *threads* exclusivamente para ejecutar los procesos de los GIF, de forma que el hilo principal pueda encontrarse en el "event loop" y la aplicación siga manteniendo toda la funcionalidad.

8.4 Pruebas

Prueba 1 cuarto *sprint*	**Visualización de datos generales 1**
Descripción	Se procede a filtrar para visualizar los datos generales filtrando por la fecha de inicio.
Resultado esperado	La aplicación modifica los datos de la tabla resumen.
Resultado	Correcto.

Tabla 22: prueba 1 cuarto sprint

Prueba 2 cuarto *sprint*	**Visualización de datos generales 2**
Descripción	Se procede a filtrar para visualizar los datos generales filtrando por un rango de fechas futuro.
Resultado esperado	La aplicación notifica que no hay datos para las fechas seleccionadas
Resultado	Correcto.

Tabla 23: prueba 2 cuarto sprint

Prueba 3 cuarto *sprint*	**Desconexión de la base de datos**
Descripción	Se desconecta de la base de datos actual y se vuelve a la vista de configuración de la conexión.
Resultado esperado	La aplicación desconecta correctamente y muestra el formulario de conexión.
Resultado	Incorrecto..
Causa	El formulario se muestra completo con los datos de la anterior conexión.
Solución	Se reinicia el formulario al desconectar.

Tabla 24: prueba 3 cuarto sprint

Prueba 4 cuarto *sprint*	**Visualización del gráfico de temperaturas**
Descripción	Se completa el filtro de la pestaña de gráficos para visualizar las temperaturas en la estación de Peñafiel.
Resultado esperado	La aplicación muestra el gráfico de temperaturas.
Resultado	Correcto.

Tabla 25: prueba 4 cuarto sprint

Prueba 5 cuarto *sprint*	**Visualización de gráfico de precipitaciones**
Descripción	Se completa el filtro de la pestaña de gráficos para visualizar las precipitaciones en la estación de Valladolid añadiendo un rango de fechas.
Resultado esperado	La aplicación muestra el gráfico esperado para el rango de fechas seleccionado.
Resultado	Correcto.

Tabla 26: prueba 5 cuarto sprint

Prueba 6 cuarto *sprint*	**Interacción con Toolbar de gráficos**
Descripción	Se prueban todas las funcionalidades que ofrece el toolbar.
Resultado esperado	El toolbar integrado manualmente permite desempeñar todas las funcionalidades que ofrece.
Resultado	Correcto.

Tabla 27: prueba 6 cuarto sprint

9. Quinto *sprint*: módulo de predicción de temperaturas máximas

Se va a proceder a desarrollar un módulo para la predicción de temperaturas máximas diarias que se integrará en la aplicación creada y cuyo funcionamiento se base en la creación de diferentes modelos de predicción mediante el uso de diferentes algoritmos de regresión que usarán como datos de entrada los datos meteorológicos almacenados en la base de datos alojada en la máquina virtual, mientras que como valor de salida obtendremos la predicción de la temperatura máxima del día siguiente.

En el módulo se añadirá la posibilidad de visualizar un histórico de las predicciones para todos los días pasados con los que se ha podido usar el modelo de predicción creado y las temperaturas reales de estos días. Estos datos se representarán mediante un gráfico.

Este módulo se desarrollará con un carácter didáctico, ya que no se pretende que los resultados obtenidos de las predicciones se aproximen con gran exactitud a los valores reales de temperaturas máximas diarias. La finalidad de este módulo es ofrecer una posibilidad más de uso a los datos meteorológicos almacenados gracias a la herramienta ETL.

9.1 Análisis

El módulo a desarrollar que se integrará en la aplicación deberá cumplir los siguientes requisitos de funcionalidad:

Nº de requisito	**Descripción**	**Crítico**
1	El módulo se integrará como una pestaña más de la aplicación creada	Sí
2	El módulo permitirá elegir el algoritmo a usar para crear el modelo.	Sí
3	Las variables propias de cada algoritmo en caso de tener, se podrán modificar desde la interfaz gráfica.	Sí
4	El módulo permitirá configurar el modelo mediante la selección de las variables de entrada.	Sí
5	El módulo permitirá visualizar un histórico de predicciones realizadas con el modelo creado.	Sí
6	Se podrán predecir temperaturas máximas para cada una de las estaciones existentes.	No
7	Los widgets de selección numérica para las variables no serán del tipo *input* o entradas de texto normales	No
8	El módulo tendrá un diseño responsive acorde a la aplicación donde se va a integrar.	No

Tabla 28: requisitos quinto sprint

9.1.1 Modelo de dominio

En el siguiente modelo de dominio se pueden visualizar cada una de las entidades o clases identificadas para la resolución del *sprint*. El dominio es prácticamente igual al del cuarto *sprint*, los únicos cambios son que únicamente se utilizarán gráficos como herramientas de visualización durante el desarrollo de este sprint y la inclusión de la entidad "modelo de predicción". También se ha añadido algún widget a mayores que no se había usado anteriormente.

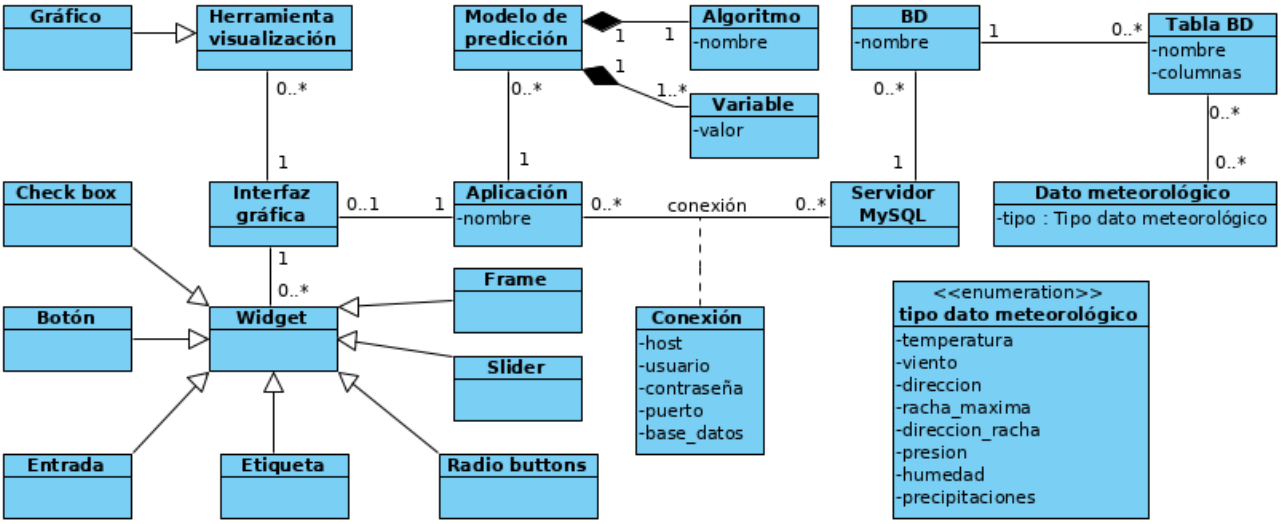


Figura 30: modelo de dominio quinto sprint

9.1.2 Casos de uso

Al igual que con el cuarto *sprint*, todas las posibles acciones a realizar como usuarios parten de que la conexión con la base de datos se ha establecido previamente completando la secuencia normal del caso de uso mostrado en el tercer *sprint*.

A continuación, se muestran cada uno de los casos de uso que se pueden dar con el módulo a desarrollar:

Figura 31: diagrama de casos de uso quinto sprint

El caso de uso “Mostrar histórico de predicciones con el modelo usado para la predicción”, se considera del tipo “extend”, ya que es necesario que se produzca el caso de uso “Predecir temperatura del día siguiente” previamente, considerándose una acción más a realizar del caso de uso principal.

<table>
<tr><th colspan="4">1º caso de uso quinto sprint</th></tr>
<tr><td>Título</td><td colspan="3">Predecir temperatura máxima del día siguiente</td></tr>
<tr><td>Actor</td><td colspan="3">Usuario.</td></tr>
<tr><td>Descripción</td><td colspan="3">El actor podrá crear un modelo y predecir con ese modelo la temperatura del día siguiente.</td></tr>
<tr><td>Precondición</td><td colspan="3">Se ha conectado con la base de datos y se ha seleccionado la pestaña de predicciones.</td></tr>
<tr><td rowspan="5">Secuencia normal</td><td>Pasos</td><td colspan="2">Acción</td></tr>
<tr><td>1</td><td colspan="2">El sistema muestra los campos a seleccionar o completar necesarios para obtener los datos meteorológicos y configurar el modelo de predicción.</td></tr>
<tr><td>2</td><td colspan="2">El actor elige una localización, el algoritmo que se usará para crear el modelo y las variables para crear el modelo.</td></tr>
<tr><td>3</td><td colspan="2">El actor pulsa el botón de predicción.</td></tr>
<tr><td>4</td><td colspan="2">El sistema muestra la predicción de temperatura máxima del día siguiente.</td></tr>
<tr><td rowspan="6">Excepciones</td><td>Pasos</td><td colspan="2">Acción</td></tr>
<tr><td>2.1</td><td colspan="2">El actor cierra la ventana y acaba el caso de uso.</td></tr>
<tr><td rowspan="2">2.2</td><td>2.2.1</td><td>El actor pulsa el botón de desconectar situado en el sidebar.</td></tr>
<tr><td>2.2.2</td><td>El sistema vuelve a la vista de conexión con la base de datos y se inicia el caso de uso del tercer sprint.</td></tr>
<tr><td>3.1</td><td colspan="2">El actor cierra la ventana y acaba el caso de uso.</td></tr>
<tr><td rowspan="2">3.2</td><td>3.2.1</td><td>El actor pulsa el botón de desconectar situado en el sidebar.</td></tr>
<tr><td>3.2.2</td><td>El sistema vuelve a la vista de conexión con la base de datos y se inicia el caso de uso del tercer sprint.</td></tr>
</table>

	3.3	Si el actor ha dejado alguna de las variables sin completar.	
		3.3.1	El sistema indica al actor que deben completar todas las variables y se vuelve al paso 2.
	3.4	Si el actor ha creado un modelo para el cual no hay datos disponibles.	
		3.4.1	El sistema indica al actor que no hay datos disponibles para el modelo usado y se vuelve al paso 2.

Tabla 29: 1º caso de uso quinto sprint

2º caso de uso quinto *sprint*			
Título	Mostrar histórico de predicciones con el modelo usado para la predicción.		
Actor	Usuario.		
Descripción	El actor podrá visualizar un gráfico con el histórico de las predicciones realizadas con el modelo creado para predecir la temperatura del día siguiente.		
Precondición	Se ha conectado con la base de datos, se ha seleccionado la pestaña de predicciones y se ha completado el caso de uso "Predecir temperatura máxima del día siguiente".		
Secuencia normal	**Pasos**	**Acción**	
	1	El sistema muestra un botón para mostrar el histórico de predicciones.	
	2	El actor pulsa el botón para mostrar el histórico de predicciones.	
	3	El sistema muestra un gráfico con las predicciones de temperaturas máximas y las temperaturas reales para todos los días pasados con los que se ha podido usar el modelo de predicción.	
Excepciones	**Pasos**	**Acción**	
	2.1	El actor cierra la ventana y acaba el caso de uso.	
	2.2	**2.2.1**	El actor pulsa el botón de desconectar situado en el *sidebar*.
		2.2.2	El sistema vuelve a la vista de conexión con la base de datos y se inicia el caso de uso del tercer *sprint.*

	2.3	**2.3.1**	El actor modifica alguna de las variables usadas para crear el modelo.
		2.3.2	El sistema oculta el botón de mostrar histórico de predicciones y se vuelve al caso de uso “Predecir temperatura máxima del día siguiente”.

Tabla 30: 2º caso de uso quinto sprint

9.2 Diseño

Al igual que se ha indicado en los anteriores sprints el módulo de predicción se ha desarrollado e incluido en la aplicación siguiendo el patrón MVC. La lógica de negocio para realizar las predicciones y acceder a los datos se encuentra en el modelo, la vista se corresponde con la interfaz gráfica de la pestaña y el controlador es el mismo usado con el resto de la aplicación donde se han incluido los nuevos eventos disponibles que ofrece el módulo creado.

9.2.1 Diagrama de clases

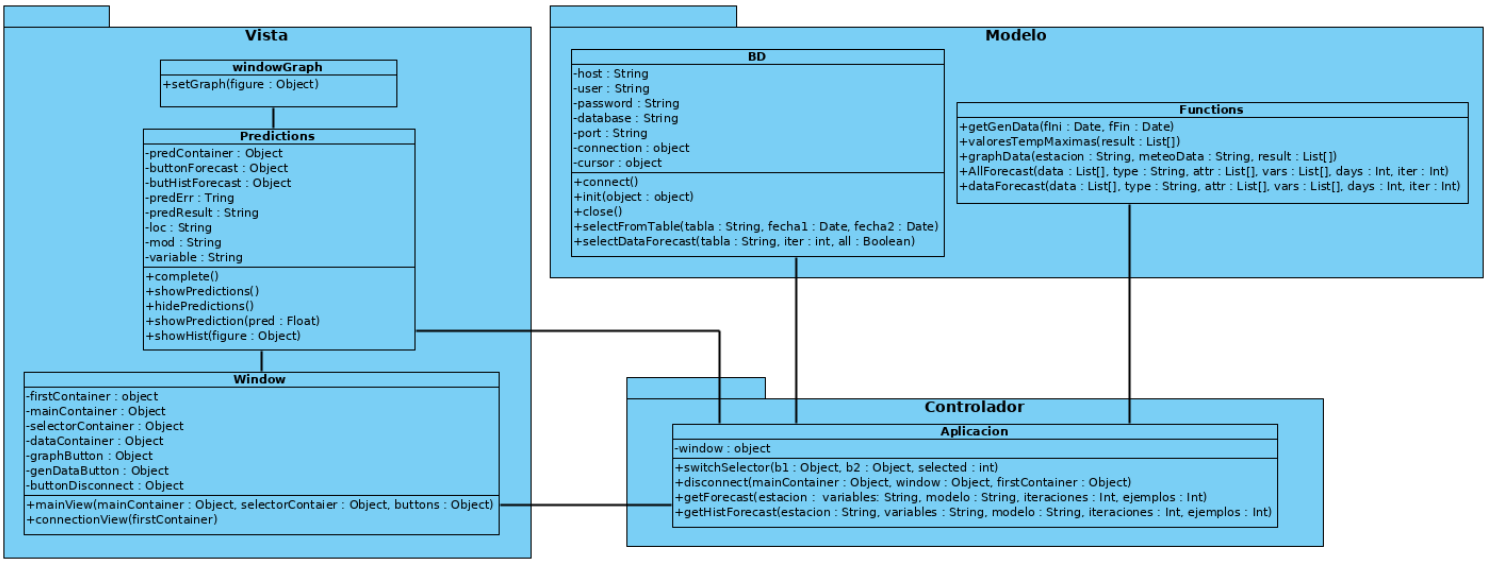


Figura 32: diagrama de clases quinto sprint

9.2.2 Diagramas de secuencia casos de uso

Caso de uso predecir temperatura máxima del día siguiente / mostrar predicción

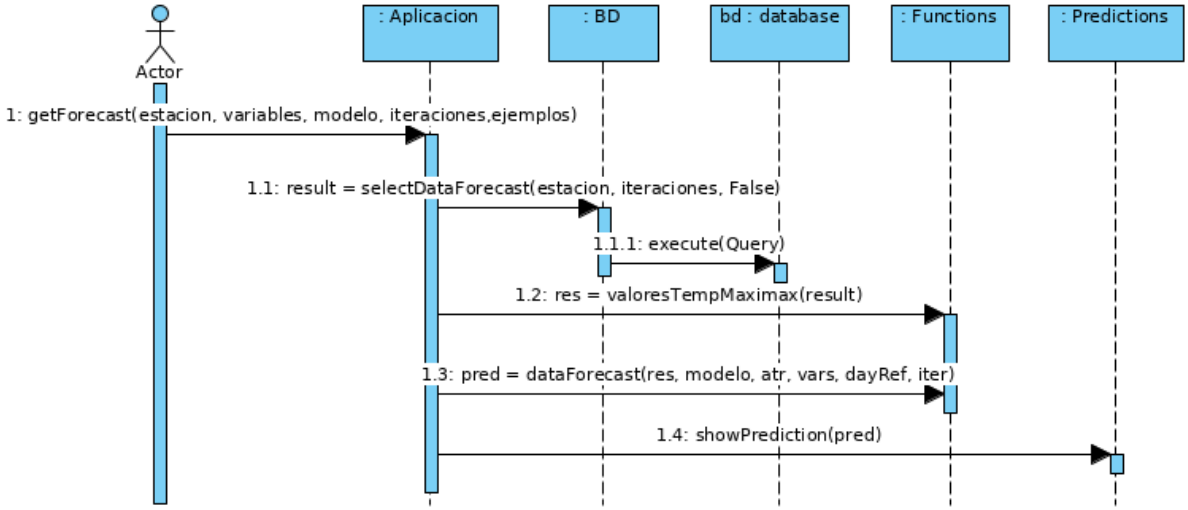


Figura 33: secuencia caso de uso predecir temperatura máxima

Caso de uso mostrar histórico de predicciones con el modelo usado para la predicción

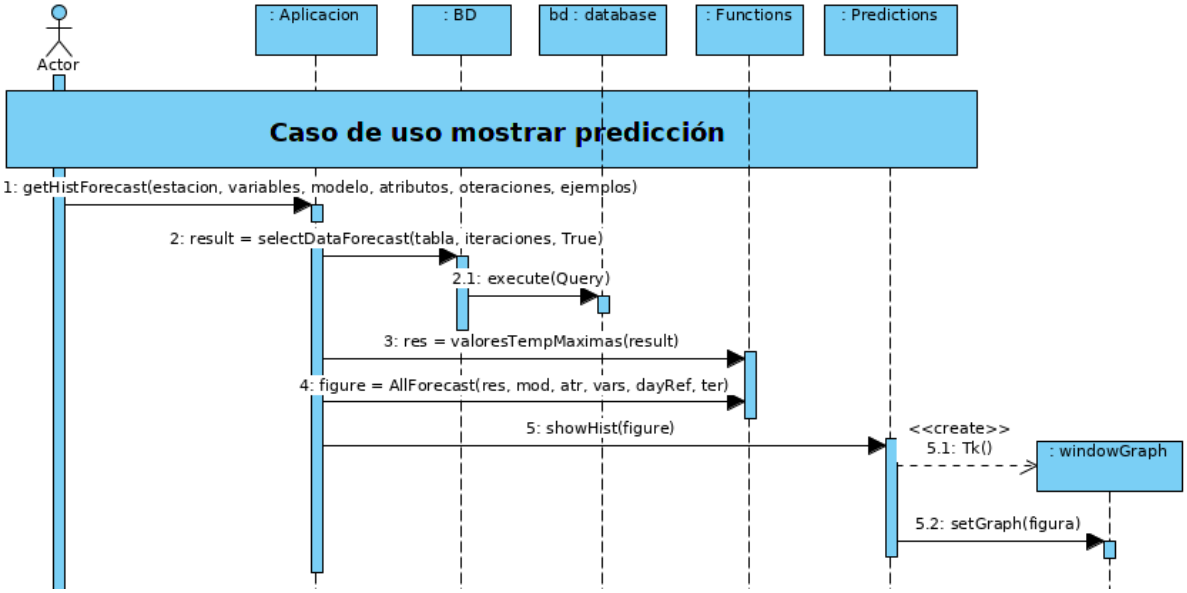


Figura 34: secuencia mostrar histórico de predicciones

9.3 Desarrollo

Los modelos que se desarrollarán van a ser todos modelos de regresión, ya que buscamos obtener la relación que existe entre una variable dependiente Y que en este caso va a ser la temperatura máxima diaria y las variables independientes X, que se detallarán a continuación.

La variable temperatura máxima diaria va a ser un valor que va a estar determinado por los valores de un conjunto de variables que podremos seleccionar a la hora de crear el modelo de predicción.

Los modelos

La biblioteca usada para la creación de los diferentes modelos que se usarán es Scikit-learn. Gracias a Scikit-learn[12] se han podido crear los siguientes tipo de modelos, usando todos ellos algoritmos orientados a la regresión:

- Modelo de regresión lineal.
- Árbol de decisión regresión.
- Red neuronal regresión.
- K-vecinos más próximos regresión.

Para crear un modelo es necesaria la existencia de una serie de parámetros que se corresponderán con un conjunto de variables explicativas y dependientes, gracias a las cuales podremos crear los distintos modelos requeridos para el desarrollo del módulo de predicciones.

Distinguimos tres tipos de parámetros usados para crear el modelo:

- Parámetros meteorológicos: podemos distinguir los siguientes tipos de variables meteorológicas que se podrán usar como variables de un modelo:

    - Temperatura máxima del día: valor numérico (float).

    - Viento en el momento de medición de la temperatura máxima del día: valor numérico (integer).

    - Dirección del viento: es el único caso de valores que tiene un formato diferente. La dirección del viento estará determinada por los valores de un array de 9 elementos. El array tendrá valores del tipo *True* o *False* (1 o 0), de forma que el valor que se halle a *True* (1) se corresponda con dirección de viento en el momento que se midieron los datos, mientras que el resto de valores se encontrarán a *False* (0).

      En el array se representan las siguientes variables:

      [Norte, Noreste, Este, Sudeste, Sur, Sudoeste, Oeste, Noroeste, Calma]

      Por ejemplo, si el viento sopla del Norte, tendremos el siguiente array como entrada:

      [1, 0, 0, 0, 0, 0, 0, 0, 0]

    - Presión en el momento de medición de la temperatura máxima del día: valor numérico (float).

    - Humedad en el momento de medición de la temperatura máxima del día: valor numérico (integer).

- Días como referencia: número de días pasados de los cuales se han usado valores de variables expuestas anteriormente (temperatura, viento, dirección, presión, humedad) como variables independientes para explicar el valor de la variable dependiente que en este caso es la temperatura máxima del día posterior al día más reciente de aquellos días que se han usado para escoger las variables independientes.

$$\text{Temperatura}(t) = \underbrace{A1 \cdot \text{Entrada}(t-1) + A2 \cdot \text{Entrada}(t-2) + \ldots An \cdot \text{Entrada}(t-n)}_{\text{Nº de días usados como referencia para describir un valor}} + An+1$$

- Número de ejemplos para crear el modelo: número de días para los cuales el valor de temperatura máxima del día ha sido usado como variable dependiente descrita en función de una serie de variables independientes. El ejemplo mostrado en el punto "días como referencia", se corresponde con un caso de ejemplo para crear un modelo.

$$\begin{aligned}
\text{Temperatura}(t) &= A1 \cdot \text{Entrada}(t-1) + A2 \cdot \text{Entrada}(t-2) + \ldots An \cdot \text{Entrada}(t-n) + An+1 \\
\text{Temperatura}(t-1) &= A1 \cdot \text{Entrada}(t-2) + A2 \cdot \text{Entrada}(t-3) + \ldots An \cdot \text{Entrada}(t-n+1) + An+1 \\
\text{Temperatura}(t-2) &= A1 \cdot \text{Entrada}(t-3) + A2 \cdot \text{Entrada}(t-4) + \ldots An \cdot \text{Entrada}(t-n-+2) + An+1 \\
&\vdots \\
\text{Temperatura}(t-m) &= A1 \cdot \text{Entrada}(t-1+m) + A2 \cdot \text{Entrada}(t-2+m) + \ldots An \cdot \text{Entrada}(t-n+m) + An+1
\end{aligned}$$

En el ejemplo de arriba podemos ver cómo se habrían usado ***m*** ejemplos para crear el modelo el modelo de regresión.

En base a lo expuesto anteriormente sobre los parámetros usados para la creación de un modelo de regresión se creará cada uno de los modelos elegidos de la librería Scikit-learn. Aunque los algoritmos de estos modelos no funcionen exactamente igual que los ejemplos mostrados durante la descripción de los parámetros necesarios, los cuales se corresponden con el modelo de regresión lineal simple, en lo referente a los parámetros usados los valores van a tener el mismo formato independientemente del modelo a usar.

Predicción

El objetivo de la predicción es hallar el valor en el instante *t+1*, basándonos en los conocimientos adquiridos con los modelos de regresión creados. Estos modelos nos proporcionan una serie de reglas o valores que, ante una serie de datos de entrada dados para las variables independientes, podremos obtener el valor de la variable dependiente.

A continuación, se muestra el que sería el ejemplo para obtener la temperatura en el instante *t+1* con el algoritmo de regresión lineal, pero que servirá como ejemplo para la comprensión del funcionamiento del resto de algoritmos:

Temperatura(t+1) = A1 . Entrada(t) + A2 . Entrada(t-1) + … An . Entrada(t-n) + An+1

Una ecuación para obtener el valor de la temperatura en *t+1* tendrá la siguiente estructura donde los valores de entrada para las variables independientes son conocidos, mientras que los valores o coeficientes que determinan la regresión son proporcionados por el modelo creado previamente posibilitando la obtención de la temperatura en el instante *t+1*.

Modelo de regresión lineal

El ejemplo mostrado durante la descripción de las variables se corresponde con un ejemplo típico de regresión lineal[13]. La ecuación típica de estos modelos es la siguiente:

$$y = a + b1 * x1 + b2 * x2 + b3 * x3 + \ldots bn * xn$$

Un modelo de regresión lineal estará definido por el número de ecuaciones como la representada arriba y el número de variables que usará cada ecuación. Las variables *a, b1, b2…* son los coeficientes de regresión que deben ser hallados para poder establecerse un modelo de forma que, ante una serie de valores para las variables dependientes dados, podamos obtener el valor de la variable dependiente.

Árbol de decisión regresión

Los árboles de decisión para regresión crean un conjunto de reglas basadas en las variables empleadas para crear el modelo, que serán usadas en un futuro para obtener el valor de regresión dadas unas variables como entrada. Las reglas de este tipo de árboles suelen ser del tipo “menor que” o “mayor que” y, dependiendo de los resultados de las reglas, se escogerá un resultado para la variable dependiente.

El algoritmo usado por el modelo del árbol de decisión depende de la variable de profundidad máxima que se le permite al árbol de decisión. En la interfaz gráfica de la aplicación se permite modificar esta variable a la hora de crear modelos con este algoritmo.

Para mostrar el funcionamiento del árbol de decisión regresión se usará un ejemplo propio del módulo creado:

- Datos usados: las variables meteorológicas son temperatura y viento, el número de días como referencia es 2, mientras que el número de ejemplos usados para crear el modelo es 3.

- Las variables usadas para crear el modelo han sido tomadas en la estación de Valladolid:

Ejemplo	TMáx (Value)	TMáx (X[0])	TMáx (X[1])	Viento (X[2])	Viento(X[3])
1	26.6 (05-06-2020)	20.4 (04-06-2020)	27.1 (03-06-2020)	13 (04-06-2020)	20 (03-06-2020)
2	20.4 (04-06-2020)	27.1 (03-06-2020)	30.6 (02-06-2020)	20 (03-06-2020)	6 (02-06-2020)
3	27.1 (03-06-2020)	30.6 (02-06-2020)	29.0 (01-06-2020)	6 (02-06-2020)	3 (01-06-2020)

Tabla 31: variables del modelo de ejemplo de árbol decisión

- Variables de entrada para la predicción en *t+1*, día 06-06-2020:
    - X[0] = 26.6 (05-06-2020)
    - X[1] = 20.4 (04-06-2020)
    - X[2] = 7.0 (05-06-2020)
    - X[3] = 13.0 (04-06-2020)

- Resultado devuelto por el árbol de decisión: 26.6 ºC

- Las reglas que ha usado el árbol de decisión regresión:

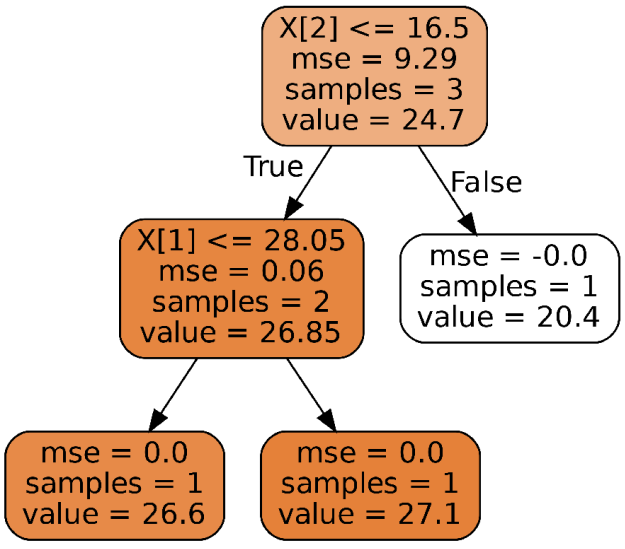


Figura 35: árbol de decisión

K-vecinos más próximos - k-NN

El modelo se encarga proporcionar un valor para la variable dependiente teniendo un conjunto de valores de entrada tomados como variables independientes en función de las similitudes/proximidad de estos con los ejemplos o casos usados para crear el modelo. El valor que se le asignará a la variable dependiente será igual a la media de las variables dependientes de los k-vecinos más próximos, siendo el parámetro k seleccionable. En la interfaz gráfica de la aplicación se permite modificar el valor de la variable k.

Red neuronal perceptrón multicapa para regresión (MLP Regressor)

Los algoritmos de redes neuronales están basados en técnicas de aprendizaje y procesamiento automático y tienen como finalidad "simular el funcionamiento del sistema nervioso humano". Las redes neuronales perceptrón multicapa para regresión usadas en este trabajo están compuestas por un conjunto de neuronas conectadas entre sí mediante enlaces formando una estructura de capas, donde las neuronas de las capas superiores toman como entradas las salidas de las neuronas de las capas inferiores. En cada neurona el valor de la entrada se multiplica por un valor o coeficiente proporcionando un valor nuevo de salida.

Los pesos por los que han sido multiplicadas las diferentes variables al crear el modelo, serán los coeficientes que se usarán para determinar la variable dependiente dado como entrada un conjunto de variables independientes, permitiéndonos de esta forma aplicar el algoritmo para la obtención de temperaturas en el instante *t+1*.

En la interfaz de la aplicación creada se permite la modificación de algunas de las variables propias de este algoritmo como pueden ser el número de iteraciones, el número de capas y neuronas por capa y la constante de aprendizaje del algoritmo.

Trabajo previo con los datos meteorológicos

La creación de los modelos ha implicado especialmente un trabajo previo de transformación de los datos a usar para crearlos, ya que una vez obtenidos los diferentes parámetros usados para formar el modelo los propios algoritmos de la librería Scikit-learn se encargan de crearlo, mientras que para la generación del valor predicho para t+1, sólo es necesario pasar los datos de las variables independientes al propio modelo.

La transformación de los datos ha requerido del uso de bibliotecas como Pandas[14] de la cual ha sido especialmente útil el uso de las estructuras de datos *DataFrames* que permiten la reordenación de los elementos que conforman la propia estructura facilitando la correcta configuración de los parámetros usados para crear los diferentes modelos. Gracias a esta biblioteca también se han podido llevar a cabo tareas de corrección tanto a la hora de crear los modelos como para la predicción de temperaturas máximas evitando valores nulos en los datos de entrada al poder identificarlos fácilmente gracias a las herramientas de Pandas.

Interfaz gráfica del módulo de predicciones

Como se indicó en el cuarto *sprint* se ha procedido a incluir un botón de selección de pestaña más en el *sidebar* correspondiente a la pestaña o módulo de predicciones y un marco donde visualizar las predicciones. De esta forma, para añadir la nueva funcionalidad se ha seguido el plan que se había creado para añadir nuevos módulos en la aplicación de una forma sencilla y rápida.

El frame creado consta principalmente de campos para la selección de las variables necesarias para crear los diferentes modelos, además de los botones necesarios, tanto para predecir la temperatura del día siguiente, como para mostrar el histórico de predicciones que se visualizará en una ventana diferente a la de la aplicación. En el histórico se han añadido campos de error medio y de número de predicciones realizadas, siendo posible de esta forma comparar los valores de las temperaturas reales con los de las temperaturas predichas conociendo el valor de error.

Para la selección de las distintas variables se han usado nuevos tipos de widgets como sliders, checkbox o radio buttons que resultan más intuitivos y proporcionan mayor facilidad para configurar el modelo además de mejorar la estética de la aplicación.

A continuación, se muestra la estructura de la interfaz gráfica del módulo de predicciones:

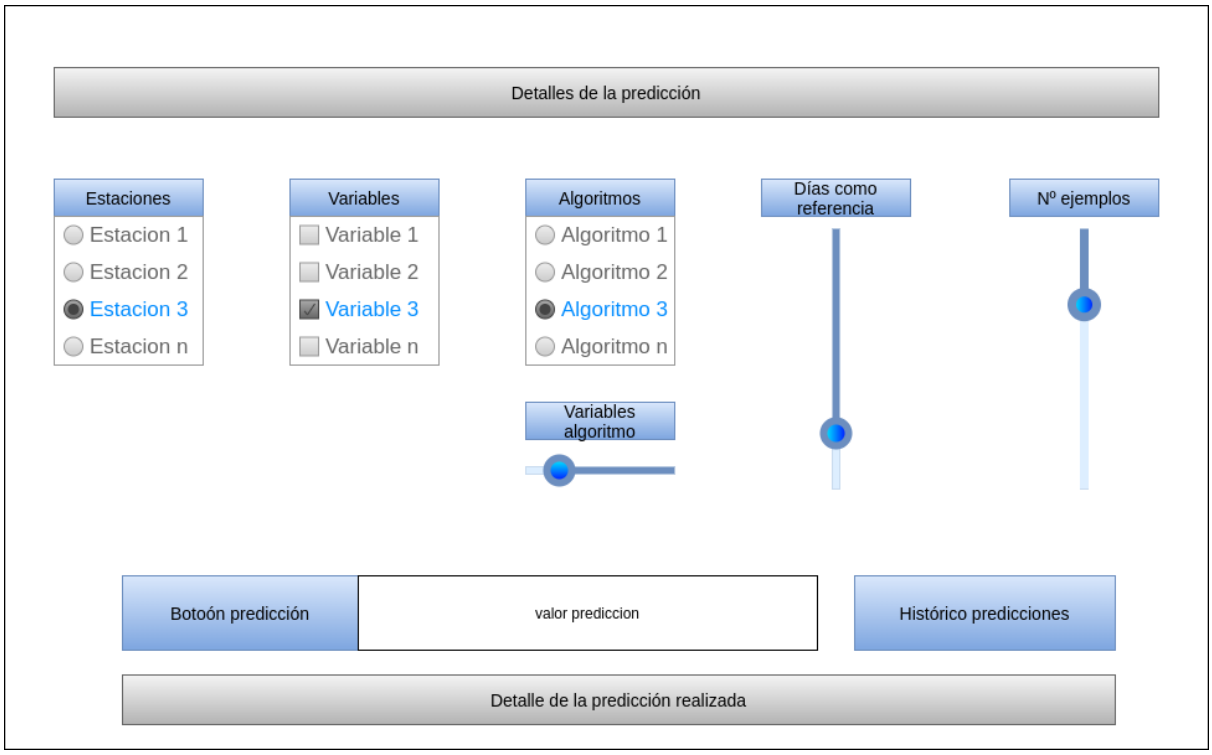


Figura 36: estructura de la interfaz gráfica de la pestaña de predicciones

9.4 Pruebas

Las pruebas realizadas durante el desarrollo del quinto *sprint* son las siguientes:

Prueba 1 quinto	**Predicción de temperatura máxima 1**

sprint	
Descripción	Predicción de temperatura máxima en Valladolid con el algoritmo de regresión lineal y usando como variables viento y dirección, 3 días como referencia y 5 ejemplos para crear el modelo
Resultado esperado	La aplicación muestra el valor de la temperatura máxima para el día siguiente.
Resultado	Correcto.

Tabla 32: prueba 1 quinto sprint

Prueba 2 quinto *sprint*	**Predicción de temperatura máxima 2**
Descripción	Predicción de temperatura máxima en Peñafiel con el algoritmo de árbol de decisión y usando como variables temperatura y humedad, 3 días como referencia y 5 ejemplos para crear el modelo
Resultado esperado	La aplicación muestra el valor de la temperatura máxima para el día siguiente.
Resultado	Correcto.

Tabla 33: prueba 2 quinto sprint

Prueba 3 quinto *sprint*	**Predicción de temperatura máxima 3 y muestra de histórico**
Descripción	Predicción de temperatura máxima en Medina de Rioseco con el algoritmo K-Vecinos más próximos y usando como variables temperatura y dirección, 3 días como referencia y 5 ejemplos para crear el modelo. Muestra del histórico con el mismo modelo.
Resultado esperado	La aplicación muestra el valor de la temperatura máxima para el día siguiente y muestra el gráfico con el histórico de predicciones.
Resultado	Correcto.

Tabla 34: prueba 3 quinto sprint

Prueba 4 quinto *sprint*	**Predicción de temperatura máxima 4 y muestra de histórico**
Descripción	Predicción de temperatura máxima en Olmedo con el algoritmo de Red neuronal y usando como variables temperatura, viento y dirección, 5 días como referencia y 8 ejemplos para crear el modelo.
Resultado esperado	La aplicación muestra el valor de la temperatura máxima para el día siguiente y muestra el gráfico con el histórico de predicciones.
Resultado	Correcto.

Tabla 35: prueba 4 quinto sprint

10. Sexto *sprint*: módulo informativo y visualización con Power BI

Para el desarrollo del sexto y último *sprint* se han marcado dos objetivos:

- Incluir una pestaña o módulo informativo en la aplicación que facilite al usuario el uso de la misma, explicando brevemente cada una de las funcionalidades que se pueden realizar y mostrando un resumen general de la finalidad de la aplicación que ponga en contexto al usuario que vaya a usarla.

- Usar una herramienta externa para la visualización y análisis de datos con el objeto de proporcionar otra forma de uso a los datos meteorológicos almacenados en la base de datos de la máquina virtual y demostrar que con el desarrollo del TFG se ha creado una herramienta que desempeña el proceso ETL de datos meteorológicos de la AEMET y que los datos almacenados son totalmente aprovechables.

En el presente sprint no se han realizado los apartados de análisis de casos de uso y dominio ni el apartado de diseño al considerarse que los trabajos a realizar no necesitaban de estas secciones o no procedía a su uso.

10.1 Análisis

Los requisitos que se deben cubrir con el trabajo a realizar durante este último *sprint* para conseguir los objetivos descritos anteriormente, son los siguientes*:*

Nº de requisito	**Descripción**	**Crítico**
1	El módulo informativo será incluido con el sistema ya creado de pestañas en la aplicación.	Sí
2	El módulo informativo permitirá al usuario conocer las funcionalidades que ofrece la aplicación.	Sí
3	La herramienta de visualización externa deberá conectarse a la base de datos donde se encuentran los datos meteorológicos.	Sí
4	Se usará la herramienta de visualización externa para realizar un ejemplo de representación y análisis de los datos que sirva como demostración de la validez de los datos almacenados gracias a la herramienta que ejerce el proceso ETL.	Sí

Tabla 36: requisitos sexto sprint

10.2 Desarrollo

Como se ha explicado en la introducción del sprint, hay dos objetivos de una índole diferente, por lo que para la parte de desarrollo diferenciaremos entre los trabajos realizados para la consecución de cada uno.

Módulo informativo

Para añadir un nuevo módulo en la aplicación se ha seguido el procedimiento explicado en *sprints* previos. Este proceso consiste en incluir el botón de selección del módulo a incluir en el *sidebar* de la aplicación y posteriormente crear un contenedor o *frame* asociado al botón en cuestión, que se corresponderá con la pestaña donde se incluirá toda la información, de forma que cuando el botón se halle seleccionado podamos visualizar dicha pestaña informativa.

En la pestaña informativa se muestra un resumen del propósito general que tiene la aplicación creada para poner en contexto al usuario. Además del resumen se incluye una breve descripción de los distintos módulos o pestañas incluidas y sus formas de interacción, donde se indican algunos consejos de uso para conseguir los resultados esperados y poder obtener toda la funcionalidad que ofrece la aplicación.

Herramienta de visualización externa: Power BI

La herramienta de visualización externa que se ha utilizado para hacer las demostraciones de representación y análisis de los datos meteorológicos ha sido Power BI.

Power BI ha sido la elegida por el hecho de ser una herramienta cuya versión estándar está disponible sin coste alguno, además de resultar muy intuitiva y práctica y a la hora de trabajar con ella. Power BI cuenta con dos versiones, una versión de escritorio desde la cual se pueden realizar los trabajos con los datos y una versión web donde se pueden publicar los trabajos realizados en la versión de escritorio, de forma que estos permanecerán accesibles vía web para su visualización.

Por contra, la versión de escritorio de Power BI, que es la necesaria para poder realizar los trabajos, sólo está disponible en entornos de Windows, por lo que en mi caso ha sido necesario el uso de VirtualBox para conseguir una máquina virtual con el sistema operativo Windows, logrando de esta forma poder obtener la aplicación Power BI en versión de escritorio.

La aplicación de escritorio se puede descargar tanto de la tienda oficial de Microsoft (Microsoft store), como de la página oficial de Power BI.

Una vez iniciamos la aplicación en el apartado “obtener datos” situado en la barra de herramientas superior de la aplicación, podremos elegir la fuente de datos que usaremos para realizar los posteriores trabajos. En nuestro caso elegimos “MySQL database” como fuente de datos. Al elegir la opción si no tenemos instalado el conector de MySQL, la propia aplicación nos indicará que debemos instalarlo. En el siguiente enlace se pueden observar las dependencias de las diferentes fuentes de datos y conectores necesarios, encontrándose entre estos la información referente a MySQL: Requisitos previos de los orígenes de datos de Power BI

Con el conector de MySQL instalado ya podemos establecer los parámetros de conexión con la base de datos. En las siguientes imágenes se muestran las opciones de conexión a completar con los parámetros introducidos:

Figura 37: configuración de conexión - 1

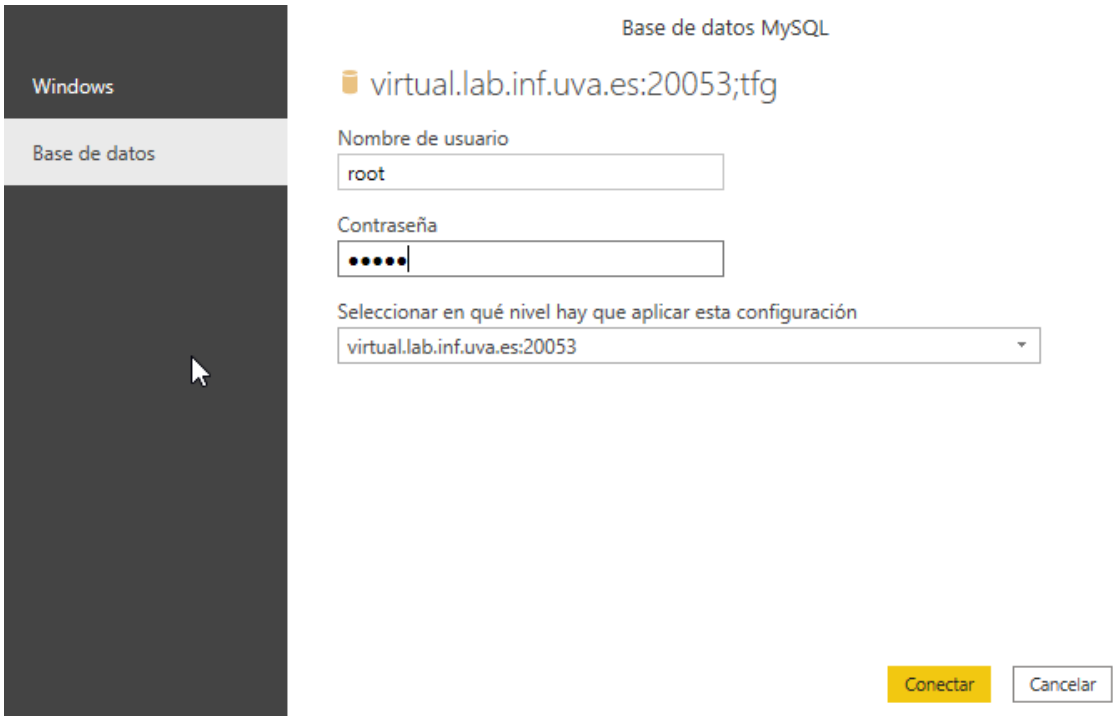


Figura 38: configuración de conexión - 2

Una vez completada la configuración y establecida la conexión tendremos acceso a los datos de cada una de las tablas que se encuentran en la base de datos creada para el Trabajo de Fin de Grado, pudiendo acceder a los datos meteorológicos para realizar multitud de representaciones gráficas permitiendo filtrar, combinar o agrupar los datos de diversos modos y ofreciendo una variedad de formas de trabajo mucho más amplia que la que se ofrece con la aplicación creada durante el desarrollo del TFG.

Como ejemplo de aplicación sobre los datos meteorológicos, se ha creado un *dashboard* o cuadro de mando donde se han incluido diferentes representaciones de los datos de la estación de Valladolid.

Se ha escogido esta estación al ser la única de las diferentes estaciones de la provincia de Valladolid para la que los datos almacenados son completos.

El *dashboard* está compuesto por 3 pestañas, identificándose cada una con el estudio de una variable meteorológica en concreto para la estación. De esta forma tenemos que en la primera pestaña se ha realizado un estudio de la temperatura de Valladolid en función del resto de variables meteorológicas, en la segunda pestaña se ha procedido a estudiar el viento y en la tercera se ha estudiado las precipitaciones de la ciudad en función de otras variables como pueden ser la dirección del viento, la presión…

Las pestañas del dashboard se han estructurado dividiéndolas en dos columnas a la hora de mostrar los diferentes gráficos, reservando el espacio superior de las mismas para mostrar el título identificativo de la pestaña y los filtros por año y mes. La estructura ha quedado de la siguiente forma:

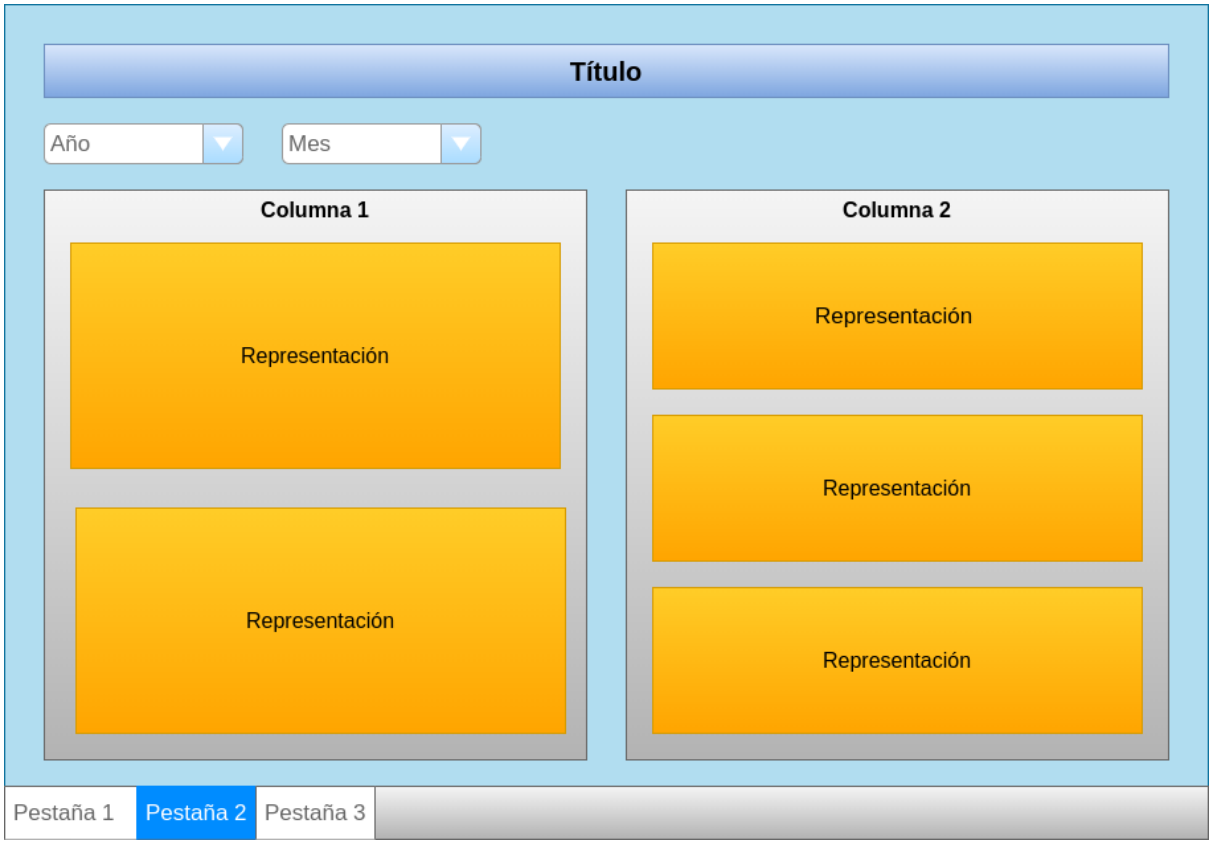


Figura 39: estructura dashboard Power BI

Como se ha comentado antes, una de las ventajas de Power BI, es que una vez realizado el trabajo de análisis y representación de datos requerido, se puede publicar en la versión o servicio web de Power BI en la *nube*, de forma que tendremos acceso a nuestros trabajos vía web. Este servicio ha sido de gran ayuda para poder visualizar el trabajo realizado sin la necesidad de tener que usar la máquina virtual para abrir el *dashboard* con la aplicación de escritorio.

11. Conclusiones y trabajo futuro

11.1 Conclusiones

A nivel personal he podido poner en práctica los conocimientos adquiridos durante mi estancia universitaria y he conseguido ampliarlos debido al gran trabajo de investigación que conlleva realizar un Trabajo de Fin de Grado. Además, el TFG realizado ha supuesto trabajar con diferentes áreas de la informática como pueden ser ingeniería del software, programación, sistemas operativos, administración de bases de datos o incluso con técnicas de aprendizaje automático o *machine learning*, aunque a un nivel muy básico.

Desde el punto de vista del trabajo realizado, se ha cubierto el principal objetivo de desarrollar una herramienta que permite aplicar el proceso ETL a datos meteorológicos de la AEMET, de diferentes estaciones meteorológicas.

Gracias al uso de una máquina virtual proporcionada por la Escuela de Ingeniería Informática de Valladolid, ha sido posible automatizar el proceso ETL de forma que actualmente en la máquina virtual se encuentra un repositorio de datos meteorológicos de diferentes estaciones de la provincia de Valladolid, que es actualizado diariamente, siendo posible además acceder a estos datos para poder realizar cualquier tipo de trabajo con ellos.

La utilidad de los datos del repositorio que se encuentra en la máquina virtual y que han sido almacenados gracias a la automatización de la ejecución de la herramienta que desempeña el proceso ETL, se ha podido demostrar por una parte gracias a la aplicación que se ha desarrollado con carácter didáctico para poder realizar representaciones de los datos almacenados y predicciones de temperaturas máximas diarias y por otra parte mediante la conexión de una herramienta de visualización externa verificando que los datos almacenados son interpretables por diferentes herramientas independientemente de su procedencia y que pueden ser usados para realizar diferentes aplicaciones y con diferentes propósitos obteniendo información relevante.

Un aspecto importante a destacar de los trabajos realizados es la facilidad para incluir nuevas estaciones con las que trabajar para la herramienta ETL y el dinamismo de la aplicación desarrollada para la visualización de datos meteorológicos y predicción de temperaturas máximas que muestra las estaciones que se añaden sin la necesidad de modificar el código de la misma, de forma que son desarrollos que no se han parametrizado completamente para el uso con un número predeterminado de estaciones meteorológicas.

Como conclusión final, valorar la importancia que hasta ahora desconocía de los procesos ETL que permiten recolectar (posiblemente desde distintas fuentes), tratar (limpiar, procesar, adaptar, filtrar…) los datos así obtenidos, y centralizar en un único espacio de almacenamiento los datos así procesados, de forma que estos pueden ser empleados para diversos fines, siendo más fácil su uso debido a las operaciones que realiza previamente el proceso ETL sobre los mismos.

11.2 Líneas futuras

- Aplicación web: actualmente la aplicación desarrollada, sólo puede ejecutarse en entornos con un sistema operativo basado en una distribución de Linux y tras la descarga de la misma. El desarrollo de una aplicación web con las mismas funcionalidades es una solución a las dependencias que tiene la aplicación actual, además de que el desarrollo web permite muchas posibilidades a la hora de diseñar la interfaz gráfica por lo que también supondría una mejora a nivel estético con respecto al diseño actual de la aplicación.

- Inclusión de gráficos multivariables: la aplicación desarrollada actualmente cuenta con una pestaña de gráficos que permite representar las distintas variables meteorológicas de forma individual. Una mejora sería permitir la configuración de los gráficos para mostrar diferentes variables y ofrecer nuevas formas de interpretar la información.

- Generar informes PDF: los gráficos actualmente se pueden exportar en formato imagen, pero de forma individual gracias al toolbar incluido en la pestaña de gráficos. Una posibilidad sería añadir la opción de exportar a PDF los gráficos a petición del usuario, de forma que se pueda crear un informe desde la propia aplicación.

- Uso de Docker: encapsulación de todo el entorno necesario para poder ejecutar la aplicación desarrollada en un contenedor Docker, de forma que cualquier ordenador o máquina que tenga Docker instalado podrá ejecutar la aplicación independientemente de su sistema operativo y de las dependencias de la aplicación que no pueda cubrir sin Docker la máquina en cuestión.

Referencias bibliográficas

[1] Buck, A. et al. (2020). *Extract, transform, and load (ETL).* 2019. Último acceso: 03-06-2020. URL:https://github.com/microsoftdocs/architecture-center/blob/master/docs/data-guide/relational-data/etl.md

[2] Agencia Estatal de Meteorología Española (AEMET). Último acceso: 20-06-2020. URL: http://www.aemet.es

[3] Scrum.org (2018). *What is Scrum?*. Último acceso: 04-06-2020. URL:https://www.scrum.org/resources/what-is-scrum

[4] The Python Software Foundation. *Python 2.7 documentation*. Último acceso: 25-03-2020. URL:https://docs.python.org/2.7/

[5] The Python Software Foundation. *Python 3.8.3 documentation*. Último acceso: 02-06-2020. URL:https://docs.python.org/3.8/

[6] Vishal (2020). *Python MySQL Database Connection Explained with Examples*. Último acceso: 13-03-2020. URL:https://pynative.com/python-mysql-database-connection/

[7] Both, D. (2020). *How to use Cron in Linux.* Último acceso: 27-02-2020. URL:https://opensource.com/article/17/11/how-use-cron-linux

[8] Alves, R. (2017). *Permitir acceso remoto para MySQL*. Último acceso: 27-02-2020. URL:https://support.scriptcase.net/es/article/1058-permitir-acceso-remoto-para-mysql

[9] The Python Software Foundation. *Graphical User Interfaces with Tk*. Último acceso: 02-06-2020. URL: https://docs.python.org/3/library/tk.html

[10] Roseman, M. (2020). *The Grid Geometry Manager*. Último acceso: 08-05-2020. URL:https://tkdocs.com/tutorial/grid.html

[11] Matplotlib. *Interactive navigation*. Último acceso: 03-06-2020. URL:https://matplotlib.org/3.1.1/users/navigation_toolbar.html

[12] Scikit-learn developers. Scikit-learn User guide. Último acceso: 24-05-2020. URL:https://scikit-learn.org/stable/user_guide.html

[13] Singh Chauhan, N. (2019). A beginner's guide to Linear Regression in Python with Scikit-Learn. Último acceso: 20-04.2020. URL:https://towardsdatascience.com/a-beginners-guide-to-linear-regression-in-python-with-scikit-learn-83a8f7ae2b4f

[14] Pandas (2020). *pandas - Python Data Analysis Library - Documentation*. Último acceso: 31-05-2020. URL:https://pandas.pydata.org/docs/user_guide/index.html#user-guide

Anexo I - Manual de instalación

Requisitos para la instalación

Tanto la herramienta para el proceso ETL, como la aplicación para la visualización de datos meteorológicos y predicción de temperaturas máxima diarias, están preparadas para ejecutarse en un entorno Linux.

Para poder realizar la instalación debemos clonar el directorio con todos los archivos necesarios desde GitHub. Usaremos la siguiente sentencia de forma que tendremos una copia del repositorio en nuestro ordenador:

$ git clone https://github.com/enrique97gm/DatosTFGEnriqueGarciaMiravalles.git

Instalación de la herramienta para el proceso ETL de los datos meteorológicos

El primer paso para poder ejecutar la herramienta es la instalación del lenguaje de programación usado para desarrollarla. Instalaremos Python en su versión 2.7 que es la versión que se ha usado en la máquina virtual y con la que podemos asegurar un correcto funcionamiento.

$ sudo apt install python2.7

Instalamos el gestor de paquetes de Python, *pip:*

$ sudo apt install python-pip

Una vez tenemos instalado Python junto con su gestor de paquetes podremos instalar las bibliotecas que se requerirán para su correcto funcionamiento, pero antes es necesario instalar el servidor de base de datos que se va a usar para almacenar los datos meteorológicos que vayamos a extraer. Instalamos el servidor de MySQL que ha sido el usado en este proyecto.

$ *sudo apt install mysql-server*

Instalamos las dependencias o bibliotecas necesarias para poder ejecutar la herramienta:

- Wget:

  $ pip install wget

- CSV:

  $ pip install csv

- Datetime:

$ pip install DatetTime

- MySQL Connector

$ pip install mysql-connector-python

Con las bibliotecas instaladas falta por configurar la conexión con el servidor de bases de datos creado, para ello modificaremos el siguiente fragmento del código del archivo /TFG/*HerramientaETL/dataBase.py* donde indicaremos los detalles de conexión con el servidor MySQL instalado:

```
db = mysql.connect(
        host = "dirección_servidor",
        port = "puerto",
        user = "usuario",
        passwd = "contraseña",
        auth_plugin='mysql_native_password'
);
```

Una vez hayamos completado todos los pasos, ya podremos ejecutar el programa que desempeña el proceso ETL, con el siguiente comando:

$ python /ruta_carpeta/DatosTFGEnriqueGarciaMiravalles/HerramientaETL/herramientaETL.py estaciones.txt

Instalación de la aplicación para la visualización de datos meteorológicos y predicción de temperaturas máximas diarias

La aplicación ha sido desarrollada con Python 3.8, por lo que para ejecutarla será necesario tenerlo instalado, además de tener instalado el gestor de paquetes propio de Python 3.

$ sudo apt install python3.8

$ sudo apt install python3-pip

Una vez que tenemos instalado Python 3.8 y el gestor de paquetes de Python 3, podremos proceder a la instalación de las dependencias necesarias para ejecutar el programa. Las bibliotecas necesarias que permitirán cubrir las dependencias son las siguientes:

- Tkinter.

$ sudo apt install python3-tk

- MySQL Connector:

$ pip3 install mysql-connector-python

- PIL:

  $ pip3 install pillow

- Datetime:

  $ pip3 install DatetTime

- Matplotlib:

  $ pip3 install matplotlib

- Pandas:

  $ pip3 install pandas

- Sklearn:

  $ pip3 install Scikit-learn

Una vez hayamos completado todas las instalaciones de las dependencias o bibliotecas podríamos ejecutar la aplicación, pero para poder visualizar datos necesitamos tenerlos en el servidor de bases de datos MySQL.

En el contenido clonado del repositorio existe un archivo llamado *DatosMeteorológicos.sql* que se trata de una copia de la base de datos de la máquina virtual realizada el día 01/07/2020. Para poder trabajar con los datos de esta copia será necesario importarlos en una base de datos del servidor MySQL. Para la importación deberemos completar dos pasos:

1. Crear una base de datos en el servidor en la que vayamos a importar los datos:

   mysql> create database nombre_base_datos;

2. Importar la copia de la base de datos:

   $ mysql -u usuario -p nombre_base_datos < DatosMeteorologicos.sql

Tras completar los pasos anteriores podremos ejecutar la aplicación para la visualización de datos meteorológicos y predicción de temperaturas máximas diarias con la garantía de que además podremos trabajar con los datos que hemos importado en la base de datos de nuestro servidor. Para ejecutar la aplicación usaremos el siguiente comando:

$ python3.8 /ruta_carpeta/DatosTFGEnriqueGarciaMiravalles/AplicacionTFG/app.py

Anexo II - Manual de usuario

Estaciones presentes en la herramienta ETL

En el contenido clonado del repositorio y en concreto en la carpeta HerramientaETL, podemos encontrar un archivo denominado *estaciones.txt* en el cual se encuentran las diferentes estaciones con las que está trabajando la herramienta ETL y la base de datos en la cual está guardando los datos extraídos.

Este archivo es configurable para añadir cuantas estaciones se considere y tiene la siguiente estructura:

Base_de_datos
Estacion_1
URL_estacion_1
Estacion_2
URL_estacion_2
- .
- .
- .

Estacion_n
URL_estacion_n

Para añadir una estación deberemos ir a la página de la AEMET y desde el menú principal en la sección “el tiempo” acceder a “observación de hoy y últimos días”, la que es el siguiente URL:

http://www.aemet.es/es/eltiempo/observacion/ultimosdatos

Desde la página que encontramos en el URL anterior podremos acceder a las diferentes estaciones de todas las provincias de España. Al acceder a una provincia y escoger una estación podremos visualizar un menú donde hay una entrada que nos permite ver los datos en un formato tabla. Si accedemos a la visualización en formato tabla podremos encontrar un botón en la parte superior de la tabla que nos indica “exportar a csv” y el cual está enlazado con el archivo CSV con los datos de la estación elegida. Copiamos la ruta del enlace al archivo CSV y la añadimos en el archivo *estaciones.txt* siguiendo la estructura mostrada anteriormente, en la cual se añade el nombre de la estación a incluir previamente a el URL del archivo.

Una vez añadida la nueva estación si ejecutamos el siguiente comando tendremos una nueva tabla con el nombre y datos de la nueva estación en la base de datos especificada en el archivo *estaciones.txt:*

$ python /ruta_carpeta/DatosTFGEnriqueGarciaMiravalles/HerramientaETL/herramientaETL.py estaciones.txt

Aplicación para la visualización de datos meteorológicos y predicción de temperaturas máximas diarias

En el presente apartado del manual de usuario se mostrará cómo se usa la aplicación para la visualización de datos meteorológicos y predicción de temperaturas máximas desde el punto de vista del usuario. Para enseñar de una forma más visual el uso de la aplicación se van a adjuntar imágenes con los pasos que se pueden llevar a cabo.

Al ejecutar la aplicación la primera vista que podemos visualizar es una ventana informativa de inicio donde se explica brevemente la finalidad de la aplicación y hay un botón que nos permite avanzar a la siguiente vista:

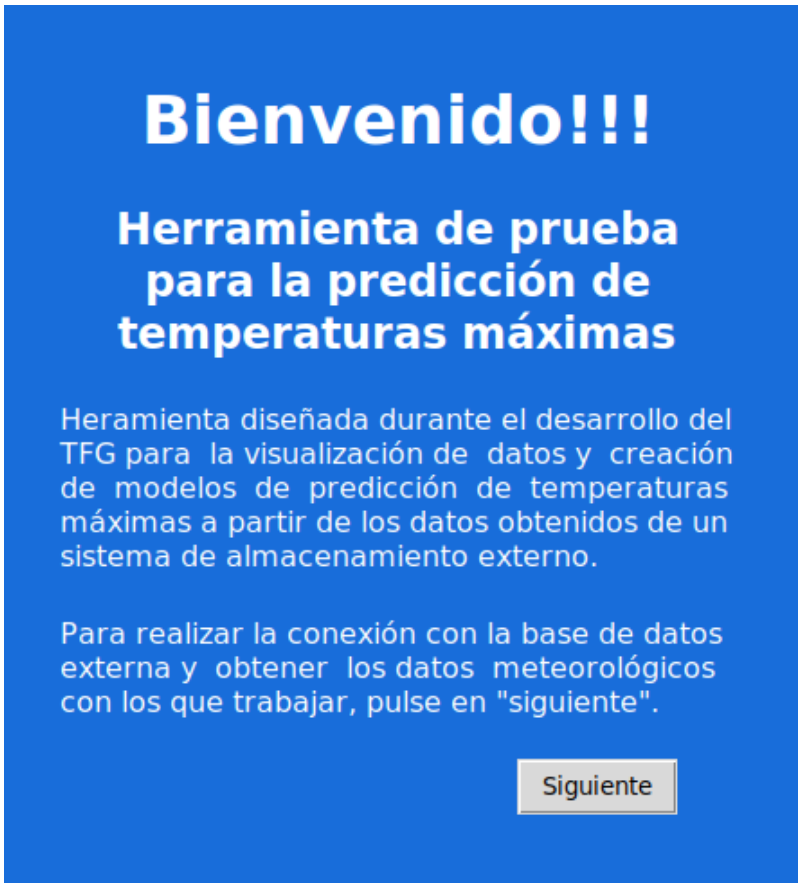


Figura 40: primera vista de la aplicación

La segunda vista de la aplicación se trata de la vista de conexión con la base de datos.

La aplicación nos solicitará completar un formulario de conexión donde se deberá indicar la dirección del servidor de base de datos, el usuario de acceso al servidor, la contraseña de usuario, el puerto de conexión y la base de datos que se pretende usar.

Figura 41: vista de conexión

Una vez se completen todos los campos de la vista y se pulse en el botón conectar, se accederá a la ventana principal de la aplicación.

La pestaña de información de la aplicación es lo primero que vamos a visualizar al acceder a la ventana principal tras haber realizado la conexión con la base de datos. En esta pestaña podemos visibilizar información que nos pondrá en contexto para el uso de la aplicación y que podrá ser consultada en caso de duda ante el uso de las diferentes funcionalidades de la aplicación.

Desde la ventana principal podemos visualizar a la izquierda un *sidebar* donde encontramos las diferentes pestañas de interacción representando cada una funcionalidad y un botón para desconectar de la base de datos actual de forma que si lo pulsamos volveremos a la vista de la figura 41.

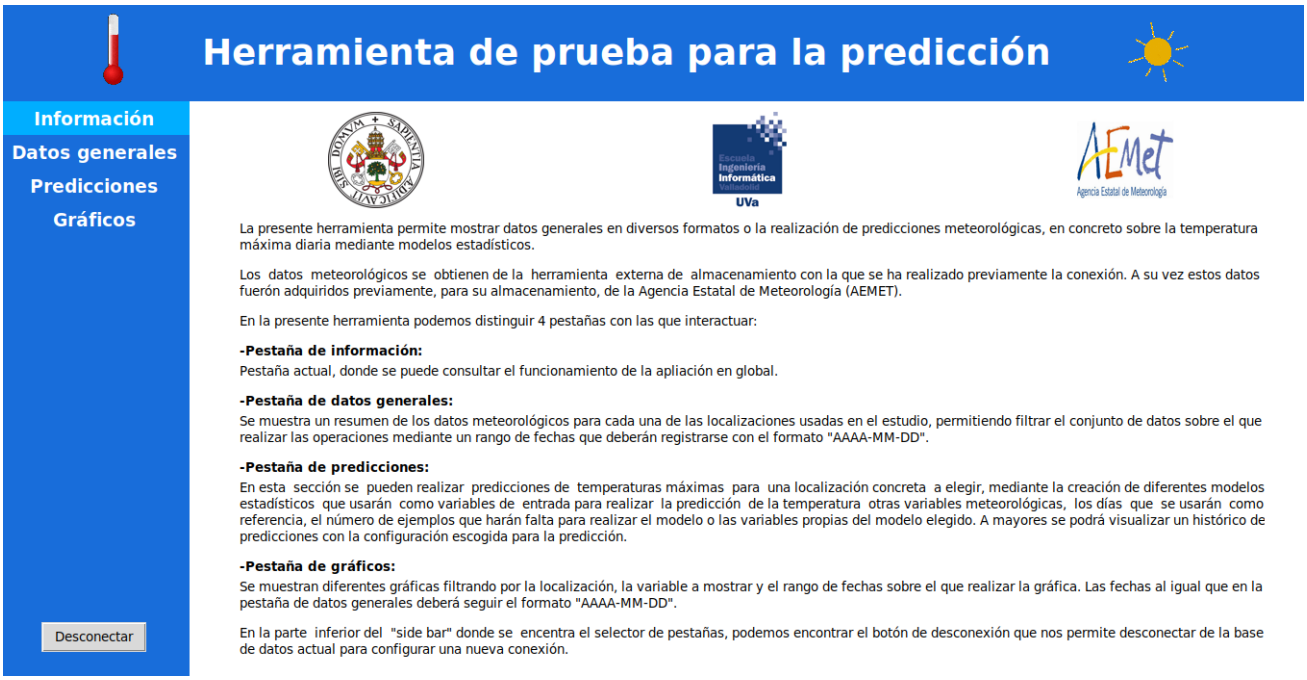
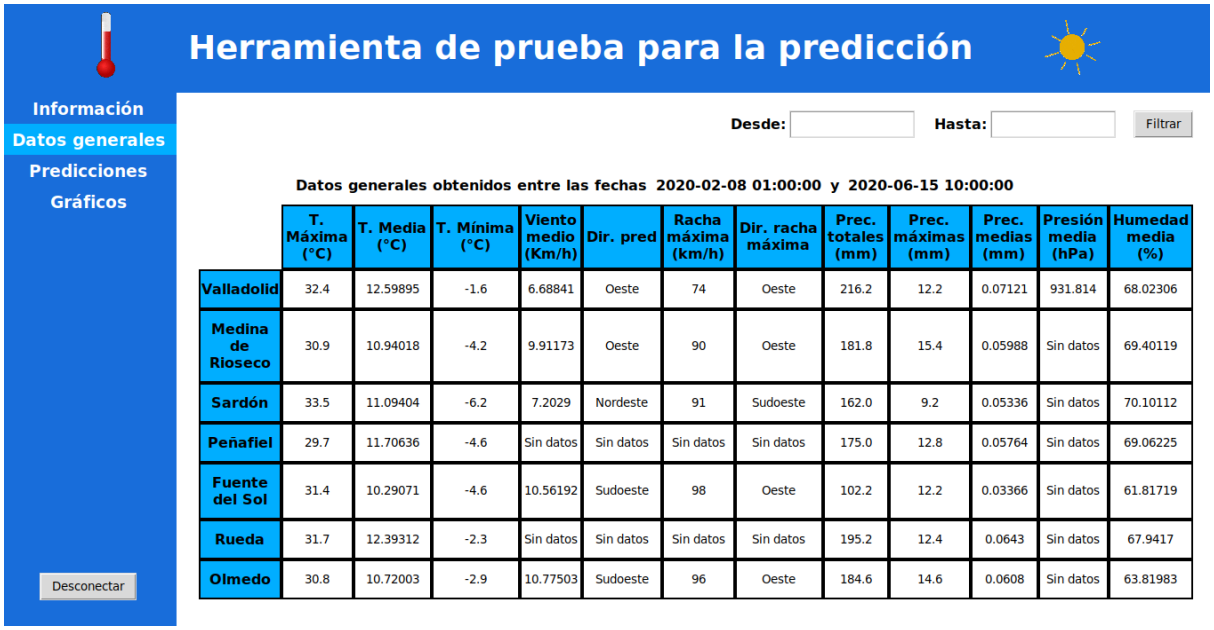


Figura 42: pestaña de información

La pestaña de datos generales nos permite visualizar una tabla resumen con información de las diferentes estaciones de las que se recogen los datos meteorológicos. Esta información general puede ser filtrada por un rango de fechas, mediante el filtro que se incluye en la parte superior de la pestaña. Sobre el filtro hay que tener las siguientes consideraciones:

- Las fechas a introducir deben seguir un formato de 'AAAA-MM-DD', Es decir, año, mes y día en formato numérico, con una separación entre los valores.
- La fecha de inicio debe ser menor que la fecha de fin del rango.
- En caso de indicar sólo la fecha de inicio, se mostrará el resumen de los datos existentes desde esa fecha.
- En caso de indicar sólo la fecha de fin, se mostrará el resumen de los datos existentes hasta esa fecha.
- Para visualizar el nuevo contenido, hay que pulsar sobre el botón "Filtrar"

A continuación, se muestra la pestaña de datos generales:



	T. Máxima (°C)	T. Media (°C)	T. Mínima (°C)	Viento medio (Km/h)	Dir. pred	Racha máxima (km/h)	Dir. racha máxima	Prec. totales (mm)	Prec. máximas (mm)	Prec. medias (mm)	Presión media (hPa)	Humedad media (%)
Valladolid	32.4	12.59895	-1.6	6.68841	Oeste	74	Oeste	216.2	12.2	0.07121	931.814	68.02306
Medina de Rioseco	30.9	10.94018	-4.2	9.91173	Oeste	90	Oeste	181.8	15.4	0.05988	Sin datos	69.40119
Sardón	33.5	11.09404	-6.2	7.2029	Nordeste	91	Sudoeste	162.0	9.2	0.05336	Sin datos	70.10112
Peñafiel	29.7	11.70636	-4.6	Sin datos	Sin datos	Sin datos	Sin datos	175.0	12.8	0.05764	Sin datos	69.06225
Fuente del Sol	31.4	10.29071	-4.6	10.56192	Sudoeste	98	Oeste	102.2	12.2	0.03366	Sin datos	61.81719
Rueda	31.7	12.39312	-2.3	Sin datos	Sin datos	Sin datos	Sin datos	195.2	12.4	0.0643	Sin datos	67.9417
Olmedo	30.8	10.72003	-2.9	10.77503	Sudoeste	96	Oeste	184.6	14.6	0.0608	Sin datos	63.81983

Figura 43: pestaña de datos generales

La pestaña de predicciones está constituida por diferentes campos de selección que nos permitirán crear el modelo de predicción que deseemos. Todos los campos deben ser completados para poder crear el modelo.

El campo de localizaciones nos permitirá elegir la localización de la estación para la que queremos predecir un valor de temperatura máxima. El campo de variables se trata de un *checkbox*, por lo que es posible escoger más de una variable para conformar el modelo. El campo de algoritmos de predicción nos permite elegir el algoritmo que usaremos para el modelo. Para alguno de estos algoritmos se pueden modificar parámetros de configuración propios, que aparecerán dependiendo del algoritmo a usar. Los campos de días como referencia y ejemplos para crear el modelo se modifican mediante un *slider*.

Una vez completados los campos, para visualizar la predicción de temperatura máxima, será necesario pulsar en el botón “Mostrar predicción”.

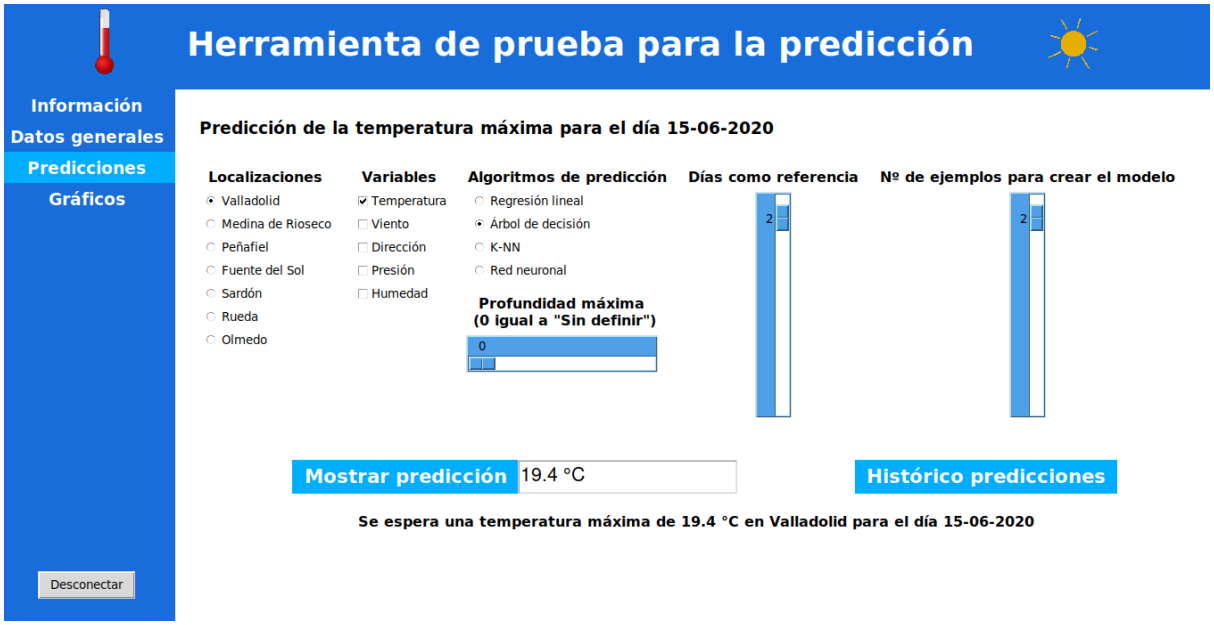


Figura 44: pestaña de predicciones

Una vez que se ha realizado la predicción, aparece la posibilidad de mostrar el histórico de predicciones realizadas con el mismo modelo. Para visualizar el histórico, deberemos pulsar en "Histórico predicciones".

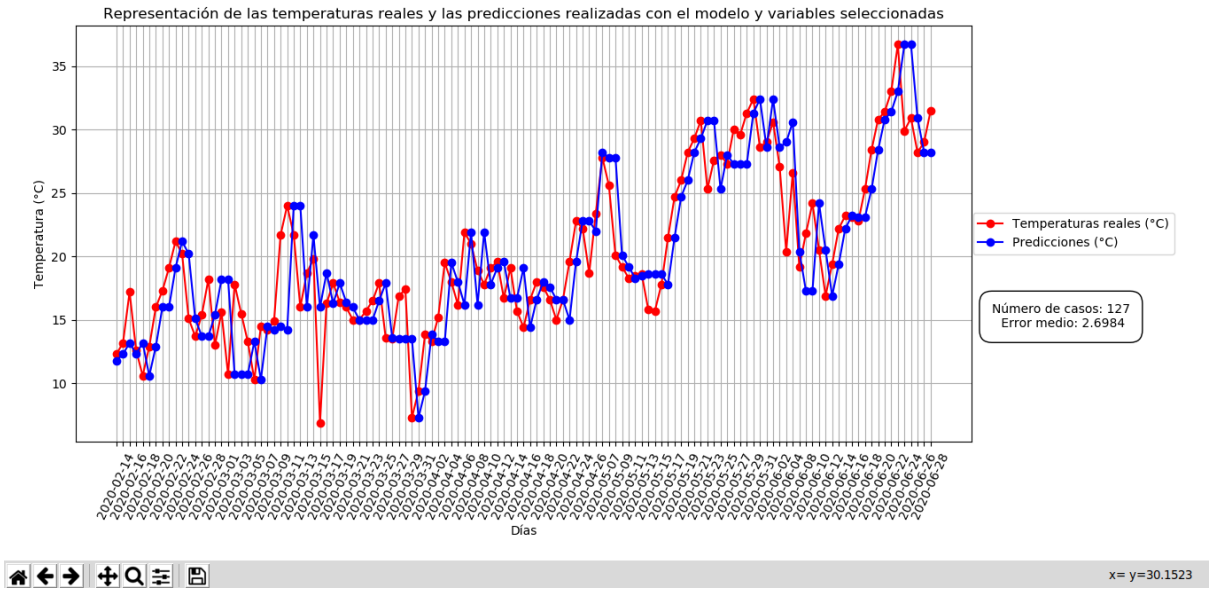


Figura 45: histórico de predicciones

La última pestaña que encontramos en la aplicación es la de gráficos. Para visualizar un gráfico deberemos completar todos los campos que no sean de fechas, es decir la localización y la variable que

queremos visualizar. Estos campos se tratan de selectores desplegables que permiten seleccionar un ítem de la lista desplegada. Las fechas siguen el mismo formato que en la pestaña de datos generales, por lo que todas las consideraciones mostradas sobre las fechas anteriormente son válidas en esta pestaña.

Para visualizar un gráfico, es necesario pulsar en el botón “Filtrar”, una vez estén completados los campos.

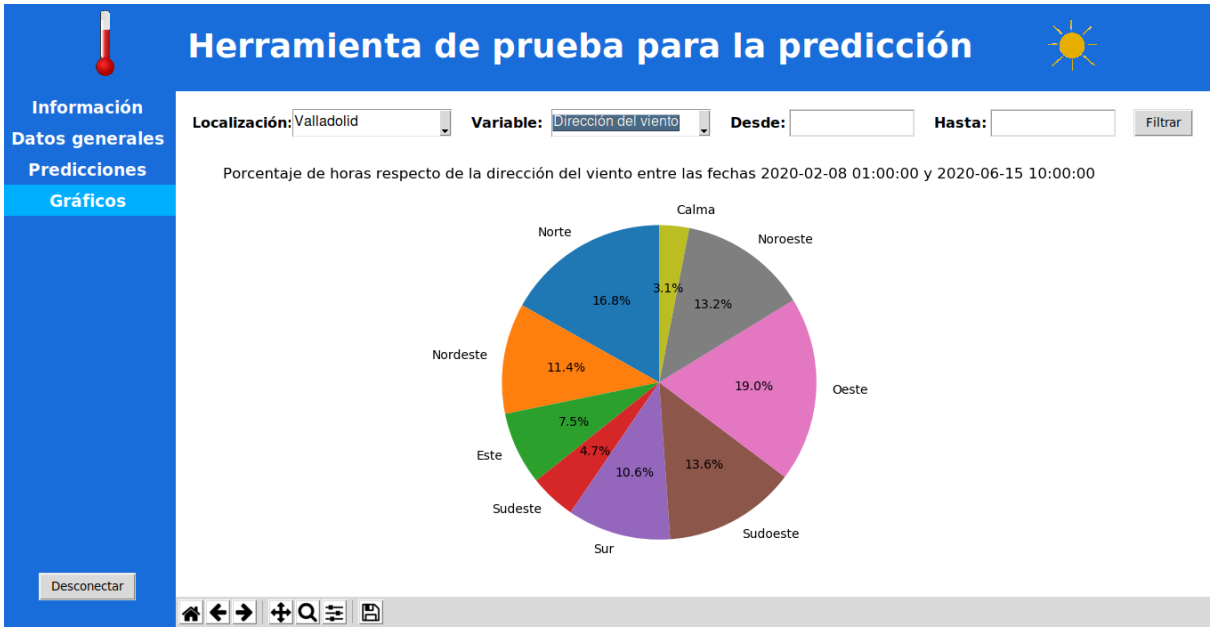


Figura 46: pestaña de gráficos

Anexo III - Dashboard Power BI

Entre los ficheros del repositorio podemos encontrar un archivo denominado *PowerBI.pbix* el cual se trata de un archivo creado con la herramienta Power BI y donde se encuentra el *dashboard* realizado con la herramienta utilizando los datos meteorológicos de la estación de Valladolid.

Con Power BI desktop podremos abrir el archivo *PowerBI.pbix* y visualizar el *dashboard* del cual se mostrará a continuación unas imágenes.

En la siguiente imagen podemos ver un estudio de la temperatura en la estación de Valladolid, donde se muestran las diferentes representaciones interactivas pudiendo filtrar sus datos por año y mes:

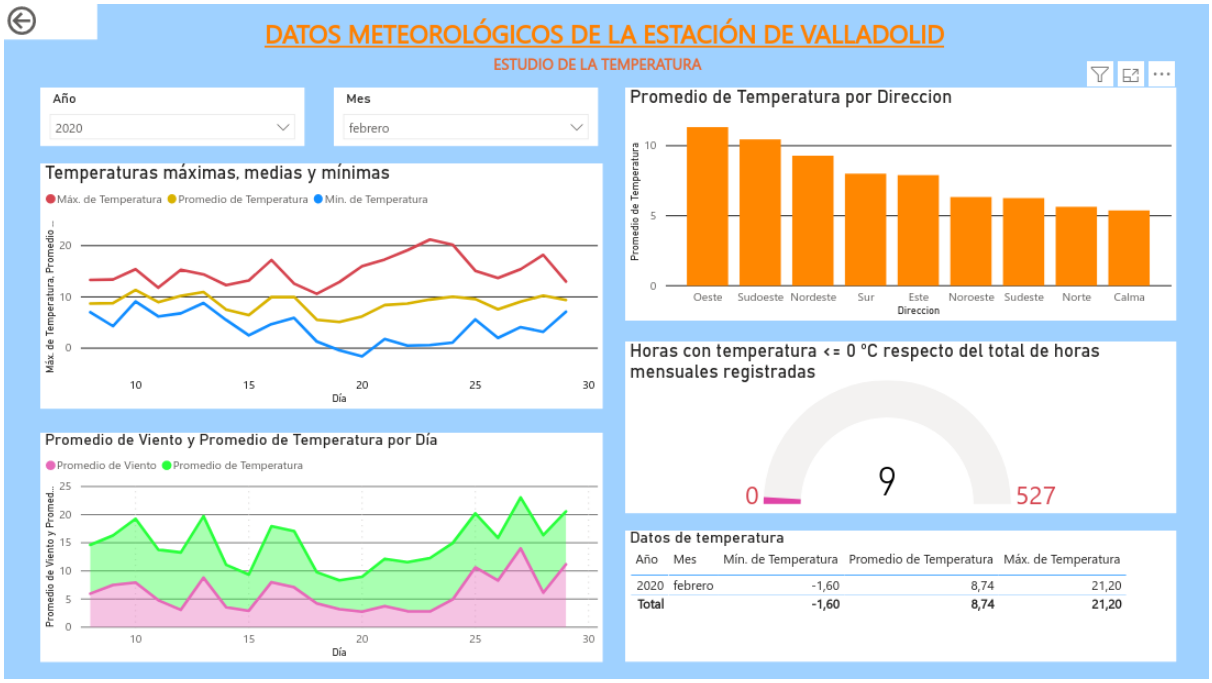


Figura 47: dashboard Power BI - vista 1

En la siguiente imagen se muestra un estudio del viento en la estación de Valladolid donde se pueden ver los valores del viento, dirección y rachas que ha registrado Valladolid filtrando por año y por mes:

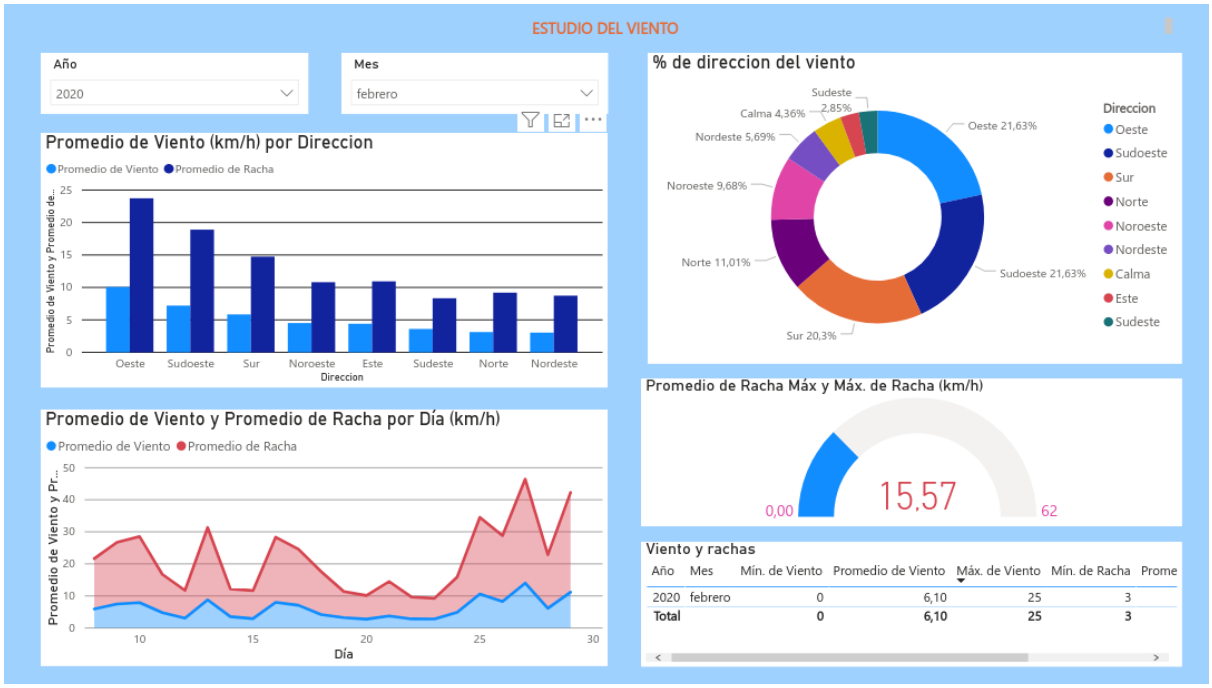


Figura 48: dashboard Power BI - vista 2

La última vista que se ha realizado es un estudio de las precipitaciones en la estación de Valladolid, en el cual se muestran la relación que tiene las precipitaciones con otras variables como puede ser la dirección del viento, humedad, presión…

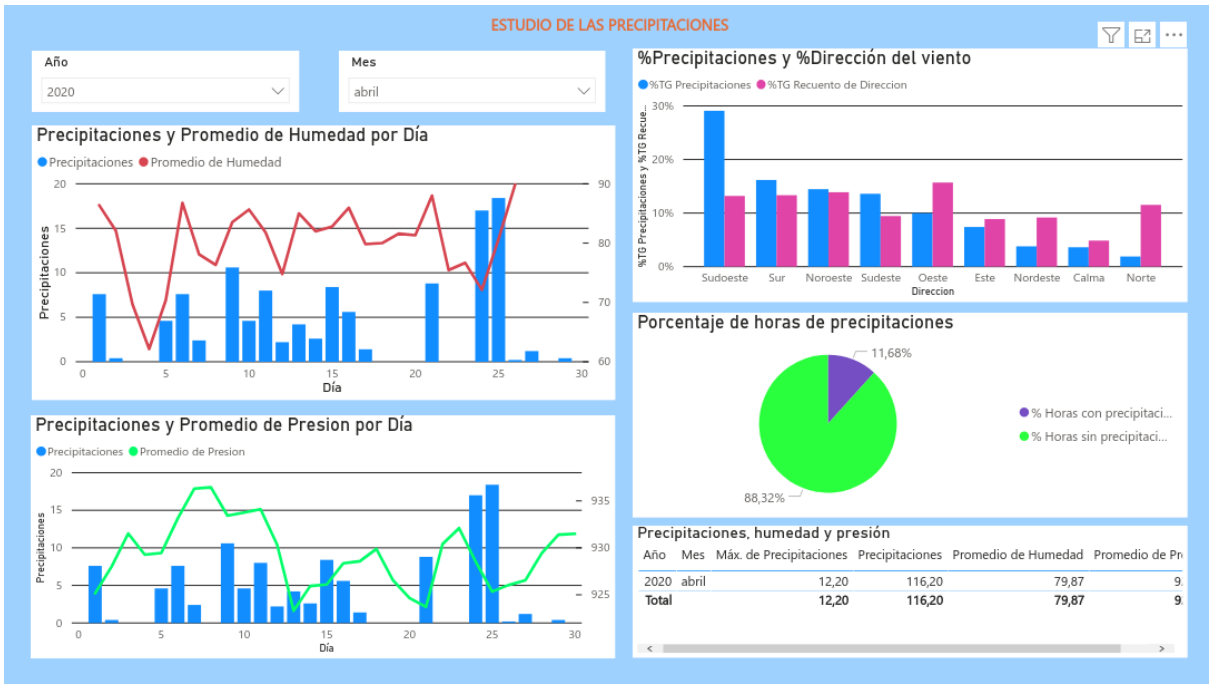


Figura 49: dashboard Power BI - vista 3

Anexo IV - Contenido del repositorio

- Carpeta HerramientaETL:
  - herramientaETL.py
  - dataBase.py
  - estaciones.txt
- Carpeta AplicaciónTFG
  - Carpeta Activos
  - app.py
  - window.py
  - model.py
  - predictions.py
  - info.py
  - graphs.py
  - genData.py
- PowerBI.pbix
- DatosMeteorologicos.sql