



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFM del Máster Universitario en
Ingeniería Informática**

**Modelo predictivo de
resultados de carreras de
Fórmula 1
Documentación Técnica**



Presentado por Francisco Martín Vargas
en Universidad de Burgos — 5 de julio de 2023
Tutor: Daniel Urda Muñoz

Índice general

Índice general	i
Índice de figuras	iii
Índice de tablas	v
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	1
A.3. Estudio de viabilidad	5
Apéndice B Especificación de Requisitos	11
B.1. Introducción	11
B.2. Objetivos generales	12
B.3. Catálogo de requisitos	12
B.4. Especificación de requisitos	14
Apéndice C Especificación de diseño	25
C.1. Introducción	25
C.2. Diseño de datos	25
C.3. Diseño procedimental	28
C.4. Diseño arquitectónico	31
Apéndice D Documentación técnica de programación	35
D.1. Introducción	35
D.2. Estructura de directorios	35
D.3. Manual del programador	37

D.4. Compilación, instalación y ejecución del proyecto	46
Apéndice E Documentación de usuario	51
E.1. Introducción	51
E.2. Requisitos de usuarios	51
E.3. Instalación	51
E.4. Manual del usuario	51
Bibliografía	61

Índice de figuras

B.1. Caso de uso principal de la especificación de requisitos	14
C.1. Diagramas de secuencia relativo al RF-2.1.1	29
C.2. Diagramas de secuencia relativo al RF-2.1.2	30
C.3. Diagramas de secuencia relativo al RF-2.1.3	31
C.4. Explicación del patrón MVC	32
C.5. Diagrama de clases de la aplicación	33
D.1. Página oficial de descargas de Python	38
D.2. Instalador de Python para Windows	38
D.3. Instalador de Python para Windows	39
D.4. Instalador de Python para Windows	39
D.5. Página oficial de Visual Studio Code	40
D.6. Instalador de Visual Studio Code para Windows	41
D.7. Instalador de Visual Studio Code para Windows	41
D.8. Instalador de Visual Studio Code para Windows	42
D.9. Instalador de Visual Studio Code para Windows	42
D.10. Instalador de Visual Studio Code para Windows	43
D.11. Página oficial de GitHub Desktop	43
D.12. Instalador de GitHub Desktop para Windows	44
D.13. Instalador de GitHub Desktop para Windows	45
D.14. Instalador de GitHub Desktop para Windows	45
D.15. Clonación del repositorio con GitHub Desktop	47
D.16. Importar proyecto en Visual Studio Code	48
D.17. Proyecto importado en Visual Studio Code	49
D.18. Crear una nueva rama desde GitHub Desktop	49
D.19. Ejecución de la app desde Visual Studio Code	50

E.1. Pantalla de inicio de la aplicación.	52
E.2. Pantalla para escoger entre todos los circuitos de la temporada.	53
E.3. Pantalla para escoger entre clima seco o mojado.	54
E.4. Pantalla para escoger entre las opciones de predicción.	54
E.5. Pantalla para escoger la predicción de las posiciones de carrera.	55
E.6. Posiciones predichas por el modelo.	55
E.7. Pantalla para escoger la predicción de las posiciones de carrera.	56
E.8. Ganador predicho por el modelo.	56
E.9. Pantalla para escoger la predicción de las posiciones de carrera.	57
E.10. Ganador de la pole predicho por el modelo.	57
E.11. Pantalla para escoger la posición de salida del piloto Albon.	58
E.12. Pantalla para escoger la posición de salida del piloto Leclerc.	58
E.13. Botón para volver a la pantalla de inicio de la aplicación.	59

Índice de tablas

A.1. Costes de personal	6
A.2. Costes de hardware	7
A.3. Costes de totales del proyecto	8
A.4. Dependencias del proyecto	9
A.5. Características de <i>GPL v3.0</i>	9
A.6. Características de <i>GPL v3.0</i>	9
A.7. Características de <i>Creative Commons Attribution 4.0</i>	10
A.8. Características de <i>Creative Commons Attribution 4.0</i>	10
A.9. Resumen de las licencias escogidas para del proyecto	10
B.1. CU-1 Gestión del modelo entrenado	15
B.2. CU-2 Predicción del resultado de los pilotos en carrera:	16
B.3. CU-3 Predicción sobre el ganador	17
B.4. CU-4 Predicción sobre el poleman	18
B.5. CU-4.1 Predicción sobre el ganador o el resultado sin datos de la clasificación	20
B.6. CU-5 Gestión de la Interfaz de usuario o aplicación	20
B.7. CU-6 Predicción del resultado de los pilotos en carrera en la aplicación	21
B.8. CU-7 Predicción sobre el ganador en la aplicación	22
B.9. CU-8 Predicción sobre el poleman en la aplicación	23
B.10. CU-9 Predicción sobre el ganador o el resultado en la aplicación sin datos de la clasificación	24

Apéndice A

Plan de Proyecto Software

A.1. Introducción

En el presente anexo se describirá la planificación llevada a cabo en el proyecto. La planificación es una etapa crucial en la cual se estiman los costos del proyecto, tanto en términos de trabajo, tiempo y recursos financieros. Analizar la planificación resulta sumamente beneficioso, puesto que permite determinar la viabilidad del proyecto, el tiempo necesario para su culminación, los recursos requeridos y su cronograma de empleo, entre otros aspectos relevantes. Para ello, se ha dividido la planificación en dos fases fundamentales: la planificación temporal y el estudio de viabilidad.

A.2. Planificación temporal

Se va a seguir una planificación por *sprints*, la cual se explica a continuación.

Sprint 0: Planificación y Objetivos del Proyecto

La planificación se llevó a cabo en 3 fases.

- Planificar el desarrollo en sprints del proyecto. La planificación del desarrollo del proyecto comienza con la definición de los objetivos del proyecto y los resultados esperados. Además en este punto vamos a dividir el trabajo en tareas más pequeñas.

- Establecer los objetivos principales del proyecto. Inicialmente se estableció como objetivo principal del proyecto comparar varios algoritmos para encontrar el más óptimo, pero posteriormente se decidió comparar para diferentes algoritmo su desempeño en función de si la selección de datos fue manual o miente un algoritmo, además de desarrollar uno de los modelos para maximizar su desempeño.
- Escoger los entornos de desarrollo, de gestión y de comunicación. Como entorno de desarrollo vamos a utilizar Visual Studio Code por la versatilidad que nos brinda, podremos desarrollar tanto la aplicación como el modelo en esta herramienta.

Este sprint se previó en una semana de duración pero realmente llevó a cabo en dos, para realizar una buena planificación y saber a dónde nos lleva el desarrollo del proyecto.

Sprint 1: Comprensión del problema y buscar fuentes de datos

Este sprint se dividió en varias tareas:

- Comprender el dominio de la F1. El objetivo principal de esta sección es obtener una comprensión profunda del mundo de las carreras de Fórmula 1 y familiarizarse con sus principales características. En este punto examinaré a fondo el dominio de este deporte de élite utilizando mi conocimiento y entusiasmo como aficionado a la Fórmula 1. A través de una investigación profunda de fuentes confiables y un análisis de datos históricos, además de mi experiencia personal como entusiasta de la Fórmula 1.
- Familiarizarse con las fuentes de datos disponibles y escoger una. Para este apartado cuento con mi experiencia como aficionado, ya que conozco muchas de las páginas que hablan de fórmula 1 y varias de ellas que albergan datos históricos. Inicialmente se han escogido para su estudio las siguientes webs: *statsf1*, *racing-reference*, *f1-fansite*, *motorsports-tats*, *Fórmula 1* y *Ergast Developer API*. Una vez identificadas, se realizó un estudio de las fuentes de datos. Esto implicó examinar la precisión de los datos, la metodología aplicada para su recopilación, su consistencia y cualquier posible sesgo o limitación en el conjuntos total de datos. También, hay que conocer los formatos y estructuras de los datos disponibles. Algunas webs no permiten descargar los datos y hay

que usar técnicas de webscrapping y es por eso que me he decantado por la API de Ergast, porque permite descargar los datos en formato json para un tratado de estos más fácil.

Este sprint se llevó a cabo en el tiempo esperado.

Sprint 2: Descarga de datos

Para llevar a cabo el sprint se realizaron las siguientes tareas

- Creación de medios para la obtención de los datos. Una vez escogida la fuente se deben crear los medios necesarios para descargar esos datos. El objetivo de esta tarea fue crear clientes python para obtener los datos de la API de Ergast. Además, se obtuvieron datos de la página web de la página oficial de la Fórmula 1, que al final se descartaron por falta de información, lo que retrasó el proyecto algunos días.
- Recopilación de los datos necesarios. En este momento se descargaron todos los datos suministrados por la API y además se decidió obtener algunos datos más para el entrenamiento, como puede ser la altitud del circuito con respecto al nivel del mar, el clima de las carreras, etc.

Este sprint se planificó para una semana pero realmente se tardaron 3 semanas debido a mi situación personal, estoy trabajando a tiempo completo y no dispuse de mucho tiempo para este proyecto, además del tiempo perdido tras la recopilación de datos en la página de la Fórmula 1.

Sprint 3: Preparación de los datos

Para llevar a cabo la preparación de los datos se realizaron los siguientes pasos

- Preprocesar los datos para poder ser utilizados en la construcción del modelo. El objetivo de este paso fue preparar los datos descargados, entre esta preparación entra mapear los cambios de nombre de las escuderías para que no se tome a una escudería como dos diferentes, cambiar la nacionalidad de los pilotos por su país de procedencia y que coincida con el país de procedencia del equipo y dónde se encuentra el circuito entre otras cosas.

- Creación de nuevos datos a partir de los ya presentes,. Este paso trató de crear nuevas características para los entrenamientos o testeo de los modelos. Se crearon las siguientes nuevas características: fiabilidad de los coches, consistencia del piloto, edad del piloto, ganador de la carrera, ganador de la *pole*, y se generó la clasificación del mundial de pilotos y equipos de cada año.
- Limpieza de los datos. El objetivo de esta tarea fue limpiar los datos existentes, esto requiere eliminar o transformar los datos nulos. En este punto se decidió eliminar los datos nulos para que no afecten al entrenamiento.

Este sprint se llevo a cabo en el tiempo esperado, una semana.

Sprint 4: Selección de características y codificación de los datos

Este sprint se dividió en dos tareas:

- Codificación de los datos. Inicialmente se buscó codificar los datos, para ello se utilizaron varios algoritmos de codificación como One-Hot o codificación ordinal dependiendo del tipo de cada uno. Los datos que ya eran numéricos no se modificaron. Además, las fechas se codificaron en segundos en función de la fecha mínima de dicha características.
- Selección de características. Esta tarea trató de generar dos selecciones, una manual y otra con un algoritmo de selección. Para ello se buscaron algoritmos y se decantó por el algoritmo Recursive Feature Elimination (RFE). Cómo tenemos tres variables objetivo se crearon cuatro conjuntos de datos, uno manual, y tres más diferentes según la variable objetivo.

La duración del sprint se estimó en una semana y se cumplió lo esperado.

Sprint 5: Selección de los modelos, entrenamiento y evaluación de estos

Para llevar a cabo este sprint se realizaron dos tareas

- Selección de modelos El objetivo fue elegir qué modelos se usarán para el entrenamiento. Se escogieron varios de ellos para que la comparación posterior sea más veraz.

- Entrenamiento y evaluación Esta tarea sirvió para entrenar los modelos iterativamente con cada uno de los datos escogidos y para cada una de las variables objetivo. Además se evalúa cada uno tras su entrenamiento guardando los datos en un diccionario para su posterior visionado.

Sprint 6: Ajuste del modelo

Sprint dividido en 2 tareas:

- Seleccionar y recopilar los hiperparámetros de los modelos. La función de esta tarea fue escoger que modelos se pretendían ajustar y recopilar sus hiperparámetros más importantes. Se decidió ajustar el modelo Random Forest Regressor y el Decision Tree Regressor de la predicción del resultado de posición en carrera.
- Escoger los valores e iterar los entrenamientos. El objetivo fue escoger uno o varios valores para cada parámetro e iterar su entrenamiento con el fin de lograr el mayor ajuste posible.

Sprint 7: Implementación y despliegue en una aplicación de interacción

Esta tarea se realizó en 3 tareas

- Diseñar la arquitectura de la aplicación. El objetivo de esta tarea fue diseñar la arquitectura que implementará la aplicación para interactuar con el modelo.
- Desarrollar la interfaz de usuario de la aplicación. Se pretendió desarrollar la interfaz del usuario, y tras varios intentos con un resultado no esperado, se realizó con la librería PyQt.
- Desarrollar la lógica de la aplicación para la interacción con el modelo. Esta tarea debía desarrollar la lógica que cargara el modelo, descargara los datos necesarios, los codificara y posteriormente los pasara por el modelo para lograr la predicción.

A.3. Estudio de viabilidad

Se va a separa este estudio en dos, el estudio de viabilidad legal y el estudio de viabilidad económica.

Viabilidad económica

Con el fin de una posible implementación del proyecto en un entorno real, desarrollaremos los costes y beneficios económicos estimados del proyecto en esta sección.

Costes del proyecto

Podemos dividir los costos del proyecto en los siguientes apartados: costos de personal, costos de hardware, costos de software, costos generales y beneficios potenciales.

Costes personales

Un ingeniero informático a tiempo completo trabajó en este proyecto durante dos meses. Se considera como el salario mínimo de un ingeniero [1]:

Concepto	Coste
Salario mensual neto	1.215,90€
Retención IRPF (24 %)	633,01€
Seguridad Social (29,9 %)	788,62€
Salario mensual bruto	2.637,53€
Total 2 meses	5.275,06 €

Tabla A.1: Costes de personal.

Como estamos hablando del estudio de viabilidad económica del proyecto, para la cotización a la Seguridad Social se tienen en cuenta los tipos de cotización de la empresa. Estos valores han sido calculados utilizando los estándares establecidos por el gobierno, que podemos consultar en [1]. Estos son los diferentes formatos de cotización que se han utilizado:

- Un 23,60 % por contingencias comunes.
- Un 5,50 % por desempleo de tipo general.
- Un 0,20 % para el Fondo de Garantía Salarial o FOGASA.
- Y un 0,60 % para formación profesional.
- Un total del 29,90 % del salario bruto total.

Para calcular la cotización del IRPF se ha consultado [5].

Costes de hardware

Hardware que será necesario para el desarrollo de este proyecto:

Hardware	Precio
Teclado y Ratón	20€
Ordenador para el desarrollo	1000€
Total	1020€

Tabla A.2: Costes de hardware.

En el cálculo del precio se han seguido los siguientes criterios:

- Precio del ordenador y periféricos usados en el desempeño de este proyecto.

Costes de software

Software utilizado en el desarrollo del proyecto:

- Python 3.
- Visual Studio Code.
- Git.

Las licencias de software necesarias son de código abierto o gratuitas, y dado que se pueden ejecutar en Linux, un sistema operativo de código abierto, no hay ningún costo por el software utilizado.

Costes de software

Costes totales del proyecto:

Coste	Valor
Coste personal	5.275,06€
Coste de hardware	1.020€

Coste	Valor
Coste software	0€
Total	6.295,06€

Tabla A.3: Costes de totales del proyecto.

Beneficios

En cuanto a los beneficios que podría aportar el desarrollo del proyecto, podríamos vender la solución a un equipo del campeonato para que puedan usarlo, o a una casa de apuestas para establecer los multiplicadores de acierto.

Viabilidad legal

En este punto investigaremos las leyes que pueden aplicarse al proyecto y las licencias que pueden ser necesarias para implementarlo.

Leyes

De acuerdo con la ley, se debe proteger la privacidad del usuario, lo que significa que cualquier información recopilada sobre él solo se puede utilizar para ejecutar la aplicación.

El usuario debe aceptar activamente y ser informado de las cookies y de la información antes de utilizarlas.

Licencias

A continuación se deben comprobar las licencias que controlan el software utilizado en el proyecto.

Por lo tanto, revisemos las licencias que usan las bibliotecas (tabla A.4).

Dependencia	Versión	Descripción	Licencia
Scikit-learn	1.2.2	Biblioteca uso de los modelos de predicción.	BSD
Pandas	2.0.0	Biblioteca uso de los datos.	BSD
Requests	2.28.2	Biblioteca para peticiones HTTP.	Apache v2.0

Dependencia	Versión	Descripción	Licencia
BeautifulSoup	4.12.2	Biblioteca para webscrapping.	Apache v2.0
Matplotlib	3.7.1	Biblioteca para crear gráficos.	BSD
PyQt	4.2.1	Biblioteca para crear la aplicación.	GPL

Tabla A.4: Dependencias del proyecto.

Una vez conocemos las licencias que usan el software de terceros, buscamos una licencia compatible con las mencionadas en la tabla A.4, Apache v2.0, BSD y GNUGPL - GNU General Public License.

La licencia más ajustada a este proyecto es la licencia *GNU General Public License v3.0*, ya que es compatible con todas las licencias del software de terceros utilizadas en el proyecto. Si quieres ver las licencias compatibles con *GNU General Public License v3.0*, visita [2].

A continuación se muestran dos tablas A.5 y A.6, con los aspectos más relevantes de la licencia *GNU General Public License v3.0*.

Licencia	Enlace	Distribución	Conceder Marca registrada
GPL v3.0	sólo con GPLv 3.0.	Licencias de GNU copyleft.	Copyleft.

Tabla A.5: Características de *GPL v3.0*.

Licencia	Concesión de patente	Uso privado	Sublicenciar	Modificación
GPL v3.0	Sí.	Sí.	Copyleft.	Sí.

Tabla A.6: Características de *GPL v3.0*.

Esta licencia está orientada al código en lugar de a la documentación, para la documentación del proyecto se ha seleccionado la licencia *Creative Commons Attribution 4.0 o CC BY 4.0..*

A continuación podemos ver en las tablas A.7 y A.8, los aspectos más relevantes de la licencia *Creative Commons Attribution 4.0.*

Licencia	Enlace	Distribución	Conceder Marca registrada
CC BY 4.0	Permisivo.	Permisivo.	Permisivo.

Tabla A.7: Características de *Creative Commons Attribution 4.0*.

Licencia	Concesión de patente	Uso privado	Sublicenciar	Modificación
CC BY 4.0	No.	Sí.	Permisivo.	No.

Tabla A.8: Características de *Creative Commons Attribution 4.0*.

Resumen

Resumen de las licencias del proyecto [A.9](#).

Recurso	Licencia
Código fuente del proyecto	GPL v3.0
Documentación	CC BY 4.0

Tabla A.9: Resumen de las licencias escogidas para del proyecto.

Apéndice B

Especificación de Requisitos

B.1. Introducción

Una descripción detallada del comportamiento del sistema a desarrollar es lo que se pretende lograr con la Especificación de requisitos de software (ERS). Además de delinear los requisitos del sistema, también enumera las necesidades del producto (tanto para el usuario como para el cliente). Cubre todos los casos posibles que un usuario final puede realizar con el software.

Este documento sirve de medio de comunicación entre todas las partes (desarrolladores, clientes y usuarios finales).

Según el estándar IEEE-STD-830-1998 [3] una ERS debe presentar las siguientes características:

- **Correcto:** es correcto si, y sólo si, el software satisface todos los requisitos especificados.
- **Consistente:** debe ser coherente con los requerimientos de la ERS, así como con los documentos de distinto nivel.
- **Inequívoco:** es inequívoco si, y sólo si, solo hay una interpretación de cada requisito establecido.
- **Completo:** para ello debe incluir:
 - Requisitos relacionados a desarrollo, funcionalidad, restricciones de diseño, atributos e interfaces externas.

- Definiciones de respuestas de software a todos los datos de entrada y a todas las circunstancias posibles.
- Tener las etiquetas llenas y bien referencias.
- **Delinear que tiene estabilidad y/o importancia:** La importancia y/o estabilidad de cada requisito debe describirse en una ERS. Cada requisito es poseedor de un identificador único que indica su estabilidad o importancia.
- **Comprobable:** cada requisito declarado debe ser verificable. Si existe una forma concreta de verificar que el producto de software cumple con el requisito, entonces ese requisito es verificable.
- **Modifiable:** si su estructura y estilo permiten que los cambios en los requisitos se realicen de manera rápida, completa y consistente mientras se conserva la estructura y estilo.
- **Identifiable:** una ERS es identifiable si la fuente de los requisitos es obvia y facilita la referencia en un posible desarrollo posterior o la documentación del mismo.

B.2. Objetivos generales

Se pretende cumplir los objetivos siguientes:

- Desarrollar un modelo para la predicción de resultados en las carreras de Fórmula 1.
- Comparar una selección mediante algoritmos contra una selección manual.
- Ajustar un modelo para mejorar sus predicciones.
- Desarrollar una aplicación para la interacción con el modelo final.

B.3. Catálogo de requisitos

En esta sección se presentan los requisitos del sistema tanto funcionales como no funcionales.

Requisitos funcionales

- **RF-1 Gestión del modelo entrenado:** El sistema debe de ser capaz de realizar predicciones.

- **RF-1.1 Predicciones de resultados:** El modelo debe poder predecir los resultados.
 - **RF-1.1.1 Predicciones de resultados de pilotos:** El modelo debe poder predecir los resultados de los pilotos en una carrera determinada.
 - **RF-1.1.2 Predicciones del ganador:** El modelo debe ser capaz de predecir si un piloto ganará o no una carrera.
 - **RF-1.1.3 Predicciones del poleman:** El modelo debe ser capaz de predecir si un piloto hará o no la pole.
- **RF-2 Gestión de la Interfaz de usuario:** La interfaz de usuario debe ser capaz de realizar las funciones necesarias para interactuar con el modelo de forma simple y fácil para el usuario.
 - **RF-2.1 Predicciones de resultados:** La aplicación debe poder predecir los resultados.
 - **RF-2.1.1 Predicciones de resultados de pilotos:** La aplicación debe poder predecir los resultados de los pilotos en una carrera determinada.
 - **RF-2.1.2 Predicciones del ganador:** La aplicación debe ser capaz de predecir si un piloto ganará o no una carrera.
 - **RF-2.1.3 Predicciones del poleman:** La aplicación debe ser capaz de predecir si un piloto hará o no la pole.

Requisitos no funcionales

- **RNF-1 Usabilidad:** la aplicación debe mantener una interfaz de usuario intuitiva y ser lo suficientemente similar a otras aplicaciones del mismo entorno para que no sea necesario un tutorial.
- **RNF-2 Rendimiento:** independientemente del dispositivo utilizado, tanto el modelo como la aplicación deben permanecer fluidos, sin fallos ni paradas en el sistema.
- **RNF-3 Disponibilidad:** el modelo debe ser accesible en todo momento y además, la aplicación debe estar disponible para su uso independientemente del momento.
- **RNF-4 Estabilidad:** el sistema debe ser estable, manteniendo un bajo nivel de errores y poder ocultarlos al usuario.

- **RNF-5 Robustez:** los eventos inesperados no deben causar un bloqueo o falla del sistema.
- **RNF-6 Fiabilidad (validez e integridad):** todas las acciones realizadas dentro del sistema deben ser válidas, preservando la integridad de los datos.
- **RNF-7 Mantenibilidad:** el sistema debe admitir la mejora del rendimiento (escalabilidad), la corrección de fallas y la adaptación de una manera simple.
- **RNF-8 Portabilidad:** el modelo debe poder ser ejecutado en varios entornos o plataformas, lo que le permita funcionar en diferentes sistemas operativos o configuraciones de hardware.

B.4. Especificación de requisitos

En este apartado se muestran los casos de uso derivados de los requisitos funcionales del proyecto.

Caso de uso principal

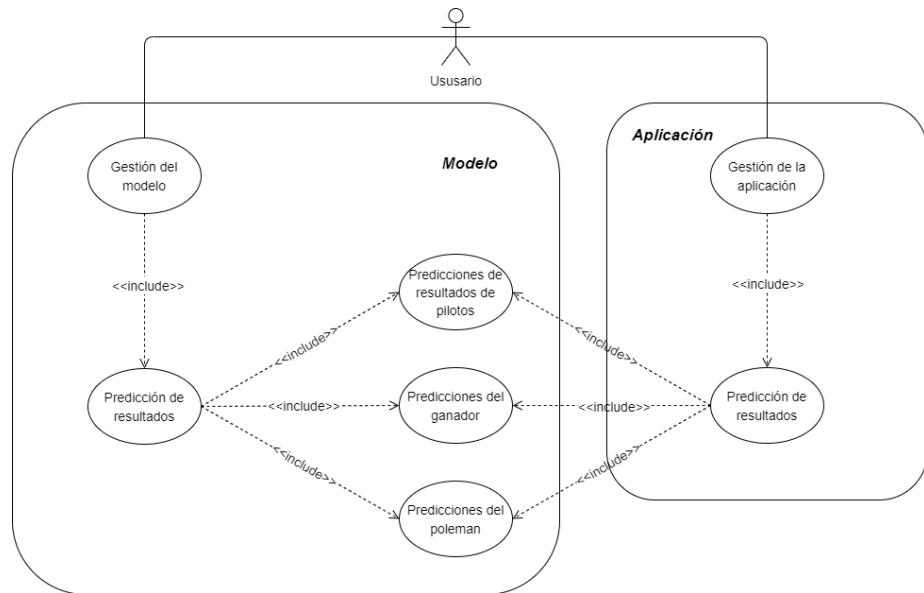


Figura B.1: Caso de uso principal de la especificación de requisitos.

Actores

Un único actor, usuario final que actúa con el sistema.

Casos de uso

CU-1	Gestión del modelo entrenado
Versión	1.0
Autor	Francisco Martín Vargas
Requisitos asociados	RF-1, RF-1.1, RF-1.1.1, RF-1.1.2, RF-1.1.3
Descripción	Permite al usuario gestionar el modelo.
Precondición	El usuario debe encontrarse en la carpeta del proyecto donde se encuentra el modelo entrenado y los codificadores.
Acciones	<ol style="list-style-type: none"> 1. Se ejecuta el programa de predicción. 2. Se muestran las opciones de predicción disponibles.
Postcondición	Se detendrá la ejecución del programa.
Excepciones	<ul style="list-style-type: none"> ■ Error al cargar los datos disponibles.
Importancia	Alta

Tabla B.1: CU-1 Gestión del modelo entrenado.

CU-2	Predicción del resultado de los pilotos en carrera
Versión	1.0
Autor	Francisco Martín Vargas
Requisitos asociados	RF-1.1, RF-1.1.1

CU-2	Predicción del resultado de los pilotos en carrera
Descripción	Permite al usuario predecir los resultados de los pilotos en una carrera.
Precondición	El usuario debe haber ejecutado el programa de predicción
Acciones	<ol style="list-style-type: none"> 1. Se muestra el mensaje de inicio 2. El usuario deberá presionar una tecla para continuar. 3. Se muestran una lista con el circuito a elegir. 4. El usuario deberá escoger entre las opciones disponibles. 5. Se muestran una lista con los climas a elegir. 6. El usuario deberá escoger entre las opciones mostradas. 7. Se muestran una lista con las posibles opciones de predicción. 8. Se escoge la opción "<i>Predecir resultado de pilotos.</i>". 9. Se muestran la predicción del resultado de la carrera.
Postcondición	Se detendrá la ejecución del programa.
Excepciones	<ul style="list-style-type: none"> ■ Error al cargar los datos disponibles.
Importancia	Alta
Tabla B.2: CU-2 Predicción del resultado de los pilotos en carrera:..	
CU-3	Predicción sobre el ganador
Versión	1.0

CU-3	Predicción sobre el ganador
Autor	Francisco Martín Vargas
Requisitos asociados	RF-1.1, RF-1.1.2
Descripción	Permite al usuario predecir el piloto que ganará la carrera.
Precondición	El usuario debe encontrarse en la carpeta del proyecto donde se encuentra el modelo entrenado y los codificadores.
Acciones	<ol style="list-style-type: none"> 1. Se muestra el mensaje de inicio 2. El usuario deberá presionar una tecla para continuar. 3. Se muestran una lista con el circuito a elegir. 4. El usuario deberá escoger entre las opciones disponibles. 5. Se muestran una lista con los climas a elegir. 6. El usuario deberá escoger entre las opciones mostradas. 7. Se muestran una lista con las posibles opciones de predicción. 8. Se escoge la opción "<i>Predecir ganador.</i>" 9. Se muestran la predicción del ganador.
Postcondición	Se detendrá la ejecución del programa.
Excepciones	<ul style="list-style-type: none"> ■ Error al cargar los datos disponibles.
Importancia	Alta

Tabla B.3: CU-3 Predicción sobre el ganador.

CU-4	Predicción sobre el poleman
Versión	1.0

CU-4	Predicción sobre el poleman
Autor	Francisco Martín Vargas
Requisitos asociados	RF-1.1, RF-1.1.3
Descripción	Permite al usuario predecir el piloto que hará la pole.
Precondición	El usuario debe encontrarse en la carpeta del proyecto donde se encuentra el modelo entrenado y los codificadores.
Acciones	<ol style="list-style-type: none"> 1. Se muestra el mensaje de inicio 2. El usuario deberá presionar una tecla para continuar. 3. Se muestran una lista con el circuito a elegir. 4. El usuario deberá escoger entre las opciones disponibles. 5. Se muestran una lista con los climas a elegir. 6. El usuario deberá escoger entre las opciones mostradas. 7. Se muestran una lista con las posibles opciones de predicción. 8. Se escoge la opción "<i>Predecir poleman.</i>". 9. Se muestran la predicción del poleman.
Postcondición	Se detendrá la ejecución del programa.
Excepciones	<ul style="list-style-type: none"> ■ Error al cargar los datos disponibles.
Importancia	Alta

Tabla B.4: CU-4 Predicción sobre el poleman

CU-4.1	Predicción sobre el ganador o el resultado sin datos de la clasificación
Versión	1.0

CU-4.1	Predicción sobre el ganador o el resultado sin datos de la clasificación
Autor	Francisco Martín Vargas
Requisitos asociados	RF-1.1, RF-1.1.1, RF-1.1.2
Descripción	Permite al usuario la predicción del ganador o del resultado de la carrera cuando no ha tenido lugar la clasificación.
Precondición	El usuario debe haber ejecutado la aplicación
Acciones	<ol style="list-style-type: none"> 1. Se muestra el mensaje de inicio 2. El usuario deberá presionar una tecla para continuar. 3. Se muestran una lista con el circuito a elegir. 4. El usuario deberá escoger entre las opciones disponibles. 5. Se muestran una lista con los climas a elegir. 6. El usuario deberá escoger entre las opciones mostradas. 7. Se muestran una lista con las posibles opciones de predicción. 8. Se escoge una opción ("Predecir posiciones." "Predecir ganador"). 9. Se muestra uno por uno el piloto con las posiciones disponibles de salida. 10. Se indica una por una la posición de salida del piloto. 11. Se muestra la predicción producida por el modelo.
Postcondición	Se muestran las opciones del programa.
Excepciones	<ul style="list-style-type: none"> ■ Error al cargar los datos disponibles.
Importancia	Alta

CU-4.1	Predicción sobre el ganador o el resultado sin datos de la clasificación
---------------	---

Tabla B.5: CU-4.1 Predicción sobre el ganador o el resultado sin datos de la clasificación.

CU-5	Gestión de la Interfaz de usuario o aplicación
Versión	1.0
Autor	Francisco Martín Vargas
Requisitos asociados	RF-2, RF-2.1, RF-2.1.1, RF-2.1.2, RF-2.1.3
Descripción	Permite al usuario gestionar la interfaz.
Precondición	El usuario debe encontrarse en la carpeta dónde está el archivo de ejecución de la aplicación.
Acciones	<ol style="list-style-type: none"> 1. Se ejecuta la aplicación. 2. Se muestran la página de inicio.
Postcondición	Se muestran las opciones del programa.
Excepciones	<ul style="list-style-type: none"> ■ Error al cargar los datos disponibles.

Importancia	Alta
--------------------	------

Tabla B.6: CU-5 Gestión de la Interfaz de usuario o aplicación.

CU-6	Predicción del resultado de los pilotos en carrera en la aplicación
Versión	1.0
Autor	Francisco Martín Vargas
Requisitos asociados	RF-2.1, RF-2.1.1

CU-6	Predicción del resultado de los pilotos en carrera en la aplicación
Descripción	Permite al usuario la predicción del resultado de los pilotos en carrera en la aplicación.
Precondición	El usuario debe haber ejecutado la aplicación
Acciones	<ol style="list-style-type: none"> 1. Se muestra la página de inicio 2. El usuario deberá hacer clic en cualquier parte de la app. 3. Se marca el circuito deseado. 4. Se indica el clima. 5. Se muestran las opciones de predicción disponibles. 6. Se pulsa el botón de la predicción. 7. Se muestra la predicción producida por el modelo.
Postcondición	Se muestran las opciones del programa.
Excepciones	<ul style="list-style-type: none"> ■ Error al cargar los datos disponibles.
Importancia	Alta

Tabla B.7: CU-6 Predicción del resultado de los pilotos en carrera en la aplicación.

CU-7	Predicción sobre el ganador en la aplicación
Versión	1.0
Autor	Francisco Martín Vargas
Requisitos asociados	RF-2.1, RF-2.1.2
Descripción	Permite al usuario la predicción del ganador.
Precondición	El usuario debe haber ejecutado la aplicación

CU-7	Predicción sobre el ganador en la aplicación
Acciones	<ol style="list-style-type: none">1. Se muestra la página de inicio2. El usuario deberá hacer clic en cualquier parte de la app.3. Se marca el circuito deseado.4. Se indica el clima.5. Se muestran las opciones de predicción disponibles.6. Se escoge la predicción deseada.7. Se muestra la predicción producida por el modelo.
Postcondición	Se muestran las opciones del programa.
Excepciones	<ul style="list-style-type: none">■ Error al cargar los datos disponibles.

Importancia Alta

Tabla B.8: CU-7 Predicción sobre el ganador en la aplicación.

CU-8	Predicción sobre el poleman en la aplicación
Versión	1.0
Autor	Francisco Martín Vargas
Requisitos asociados	RF-2.1, RF-2.1.3
Descripción	Permite al usuario la predicción del poleman.
Precondición	El usuario debe haber ejecutado la aplicación

CU-8 Predicción sobre el poleman en la aplicación

Acciones

1. Se muestra la página de inicio
2. El usuario deberá hacer clic en cualquier parte de la app.
3. Se marca el circuito deseado.
4. Se indica el clima.
5. Se muestran las opciones de predicción disponibles.
6. Se pulsa el botón de la predicción.
7. Se muestra la predicción producida por el modelo.

Postcondición Se muestran las opciones del programa.

Excepciones

- Error al cargar los datos disponibles.

Importancia Alta

Tabla B.9: CU-8 Predicción sobre el poleman en la aplicación.

CU-9 Predicción sobre el ganador o el resultado en la aplicación sin datos de la clasificación

Versión 1.0

Autor Francisco Martín Vargas

Requisitos asociados RF-2.1, RF-2.1.1, RF-2.1.2

Descripción Permite al usuario la predicción del ganador o del resultado de la carrera cuando no ha tenido lugar la clasificación.

Precondición El usuario debe haber ejecutado la aplicación

CU-9	Predicción sobre el ganador o el resultado en la aplicación sin datos de la clasificación
Acciones	<ol style="list-style-type: none"> 1. Se muestra la página de inicio 2. El usuario deberá hacer clic en cualquier parte de la app. 3. Se marca el circuito deseado. 4. Se indica el clima. 5. Se muestran las opciones de predicción disponibles. 6. Se escoge la predicción deseada ("Predecir posiciones." o "Predecir ganador"). 7. Se muestra uno por uno el piloto con las posiciones disponibles de salida. 8. Se escoge una por una la posición de salida del piloto. 9. Se muestra la predicción producida por el modelo.
Postcondición	Se muestran las opciones del programa.
Excepciones	<ul style="list-style-type: none"> ■ Error al cargar los datos disponibles.
Importancia	Alta
Tabla B.10: CU-9 Predicción sobre el ganador o el resultado en la aplicación sin datos de la clasificación.	

Apéndice C

Especificación de diseño

C.1. Introducción

Este anexo detalla los datos que maneja el sistema, la arquitectura y su diseño procedural para explicar cómo se han cumplido los requisitos y objetivos del proyecto.

C.2. Diseño de datos

En esta sección, se describe cómo se han recopilado, procesado y estructurado los datos relacionados con las carreras de Fórmula 1.

Recopilación

La mayor parte de los datos están recopilados de una API llamada *Ergast Developer API*, que es un servicio web experimental que proporciona un registro histórico de datos de carreras de coches para fines no comerciales. Se han obtenido de la API los siguientes datos: pilotos, circuitos, equipos y resultados de las carreras. Para ello se han ejecutado y obtenido los siguientes archivos:

1. Pilotos: debemos ejecutar el archivo `1_GetDrivers` para obtener los datos de los pilotos. Una vez ejecutados se guardan en la carpeta `code/data` con el nombre `Drivers.csv`.

2. Circuitos: debemos ejecutar el archivo `2_GetCircuits` para obtener los datos de los circuitos. Una vez ejecutados se guardan en la carpeta `code/data` con el nombre `Circuits.csv`.
3. Equipos: debemos ejecutar el archivo `3_GetConstructors` para obtener los datos de los equipos. Una vez ejecutados se guardan en la carpeta `code/data` con el nombre `Constructors.csv`.
4. Resultados de las carreras: debemos ejecutar el archivo `4_GetResults` para obtener los datos de las carreras. Una vez ejecutados se guardan en la carpeta `code/data` con el nombre `Results.csv`.
5. Clasificaciones: también se pueden obtener datos de las clasificaciones de esta API, pero como no se han usado, no se han guardado. En caso de necesitarlo se debe ejecutar el archivo `6_GetQualyDataErgast`. Una vez ejecutados se guardarían en la carpeta `code/data` con el nombre `qualifying.csv`.

Además, se han obtenido datos del clima de las carreras de la página de la Wikipedia a través de webscrapping, para ello ejecutaremos el archivo `5_GetWeatherData` para obtener los datos meteorológicos. Una vez ejecutado se añade en los datos de los resultados y se guardan en la carpeta `code/data` con el nombre `final_results.csv`.

Procesado

Para procesar los datos se ha seguido la siguiente estrategia:

1. Combinación de los datos: se han combinado los datos de los archivos: `Drivers.csv`, `Circuits.csv`, `Constructors.csv` y `Results.csv`.
2. Preparación de los datos: a partir de los datos combinados se han creado datos de finalización de carrera, consistencia de pilotos, fiabilidad de los coches, país de procedencia de equipos y pilotos, edad de los pilotos durante las carreras, ganadores y polemans, además de corregir algunos datos como pueden ser los cambios de nombre del mismo equipo en determinados años. Todo ello ejecutando los archivos `7_A_PreparesAndMergeData` y `7_B_DriverConstructorData`, y guardado en el archivo `merged_data.csv` en la carpeta `code/data`.
3. Limpieza de datos: a partir de los datos obtenidos en `merged_data.csv` se han eliminado las características que contenían datos nulos a través

del archivo `8_A_CleanData`, y guardado en el archivo `all_data.csv` en la carpeta `code/data`.

4. Obtención de las clasificaciones de pilotos y equipos: a partir de los datos obtenidos en `all_data.csv` se han generado los archivos `driver_standings.csv` y `constructor_standings.csv` guardados en la carpeta `code/data`.
5. Codificación y selección de los datos: se han codificado los datos para entrenarlos en los modelos a través de los archivos: `9_A_ManualSelection`, `9_B_CodificationManualData`, `10_A_CodificationData`, `10_B_AutoSelection`, y resultando en los siguientes archivos de datos guardados en la carpeta `code/data`:
 - `coded_auto_selection_data.csv`: datos codificados para la selección de características con el algoritmo RFE.
 - `coded_auto_selection_data_race_winner.csv`: datos con las características ya seleccionadas para la variable objetivo ganador de la carrera.
 - `coded_auto_selection_data_race_final_position.csv`: datos con las características ya seleccionadas para la variable objetivo posiciones finales de carrera.
 - `coded_auto_selection_data_qualy_pole.csv`: datos con las características ya seleccionadas para la variable objetivo ganador de la pole.
 - `coded_manual_data.csv`: datos seleccionados manualmente.
 - `codificadores.pkl`: se han guardado los codificadores utilizados para cada característica en la carpeta `code/coders`.
6. Entrenamiento de los modelos: los modelos han sido entrenados en el archivo `Training`, el cual ha generado los archivos con las características más relevantes para el ajuste posterior en la carpeta `code/best_split_data`. Además, se han guardado los modelos entrenados en la carpeta `code/train_models`.
7. Ajuste de los modelos: encontramos los modelos con ajuste en el archivo `TrainingParams`, que ha guardado el modelo con mejor ajuste en la carpeta `code/train_models`.

Aplicación

Cómo la aplicación es de interacción con el modelo entrenado, no guarda una persistencia de los datos como tal. Contamos con 3 archivos de datos en los que se guardan circuitos constructores y pilotos de la temporada 2023 de Fórmula 1 para hacer las predicciones:

- Pilotos: `app_drivers`.
- Constructores: `app_constructors`.
- Circuitos: `app_circuits`.

C.3. Diseño procedimental

Podemos entender las operaciones internas del sistema gracias a su diseño procedimental.

La interacción entre los diversos componentes del sistema se representará mediante diagramas de secuencia teniendo en cuenta los requisitos del sistema.

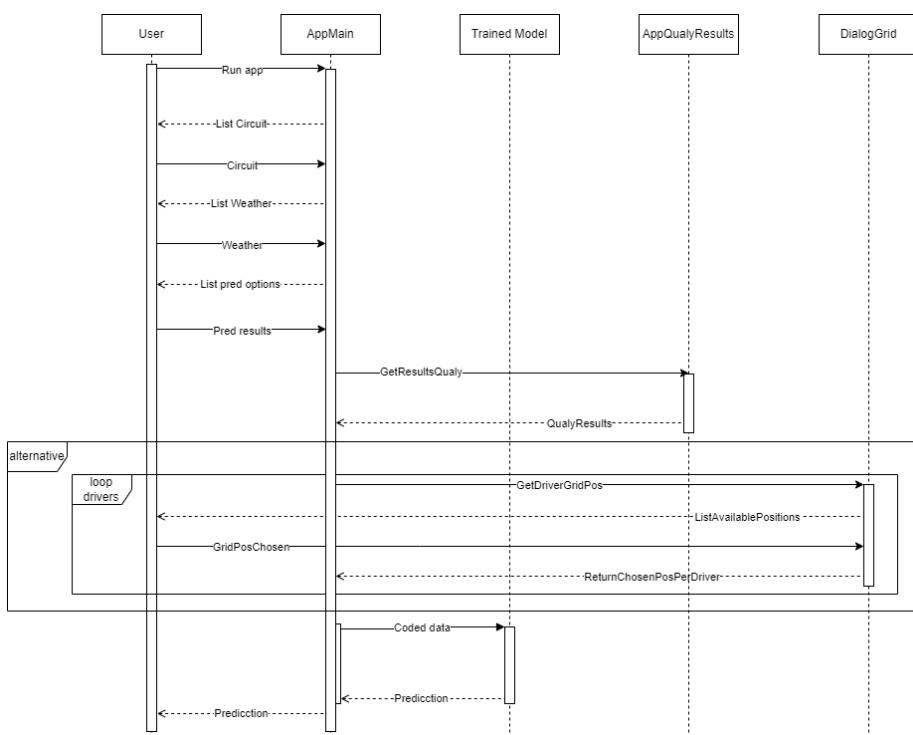


Figura C.1: Diagramas de secuencia relativo al RF-2.1.1.

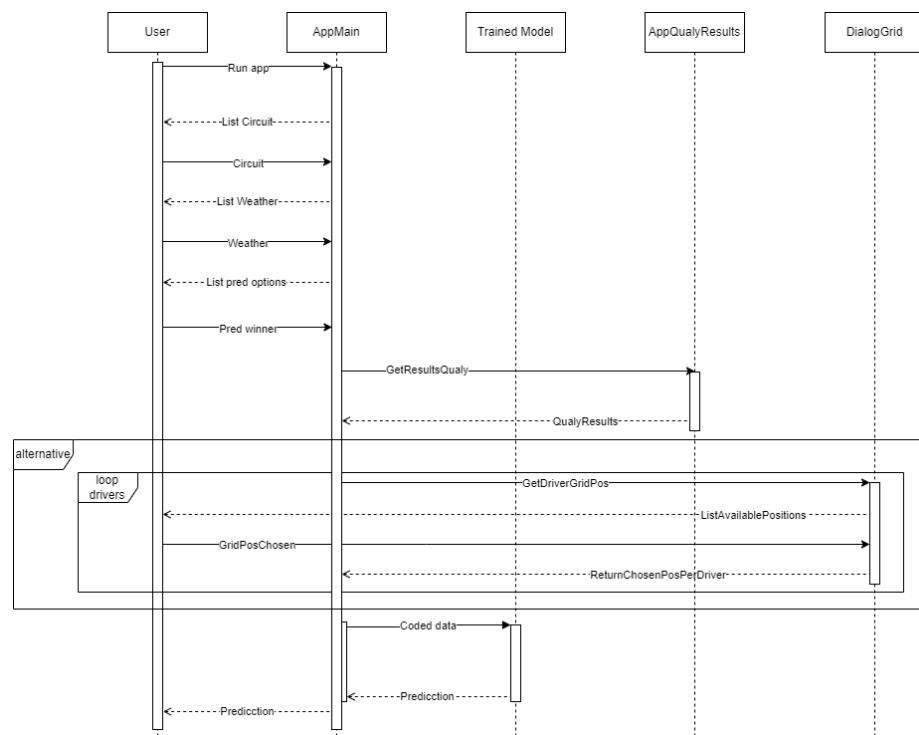


Figura C.2: Diagramas de secuencia relativo al RF-2.1.2.

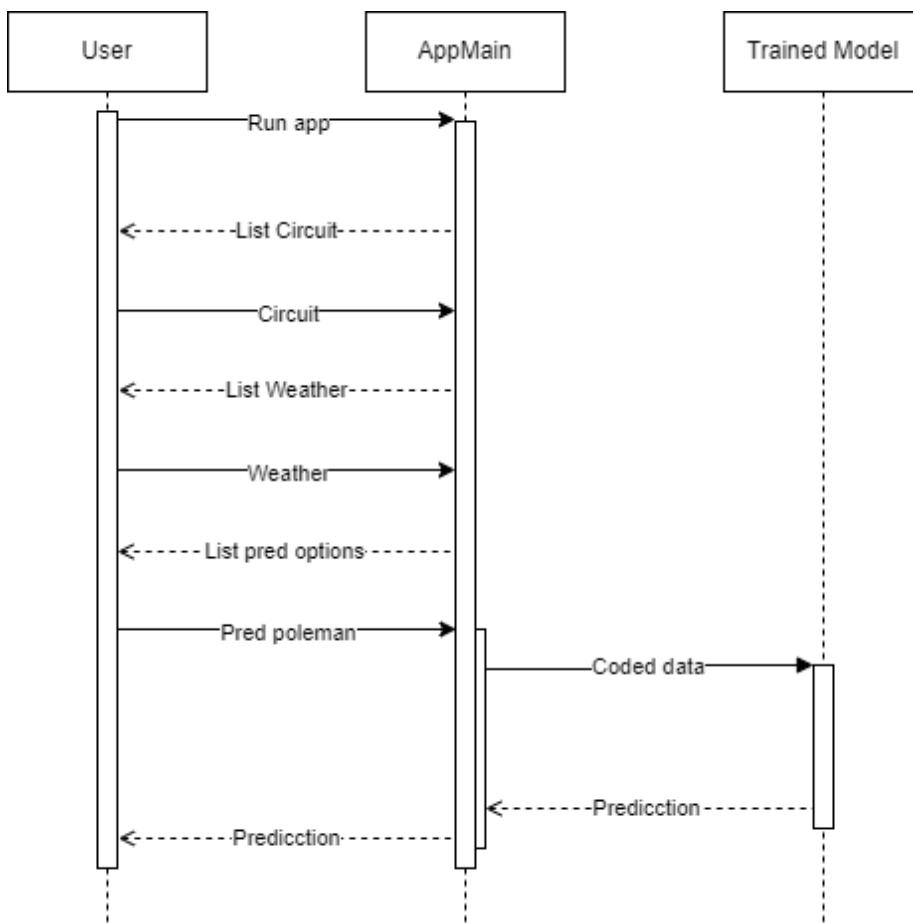


Figura C.3: Diagramas de secuencia relativo al RF-2.1.3.

C.4. Diseño arquitectónico

Arquitectura del sistema

La arquitectura modelo-vista-controlador (MVC) es el patrón de diseño que se ha seguido en la aplicación.

Patrón modelo-vista-controlador (MVC)

El patrón MVC (figura C.4) es un patrón de diseño arquitectónico que divide el sistema en tres componentes: la interfaz de usuario, la lógica de la aplicación y los datos. La capa de datos está representada por el modelo, la

interfaz de usuario está representada por la vista y la lógica de la aplicación está representada por el controlador.

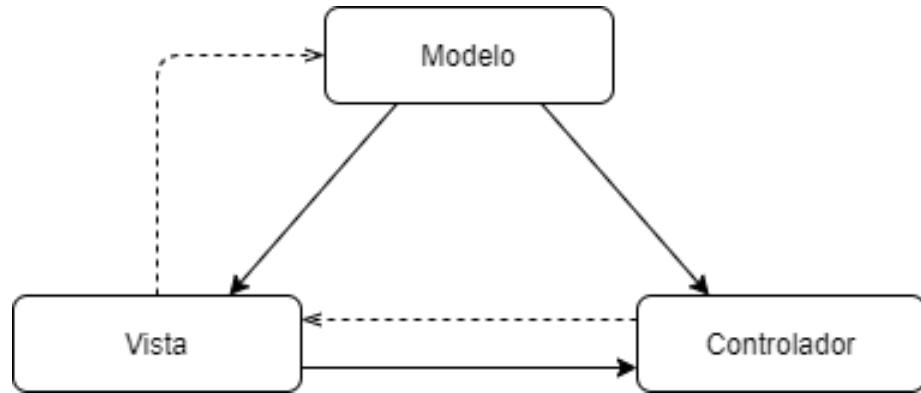


Figura C.4: Explicación del patrón MVC.

En este proyecto se ha utilizado de la siguiente manera:

- Una primera carpeta de datos (*data*) que contiene los datos necesarios para hacer las predicciones.
- Una carpeta llamada *logic* que contiene la lógica de la aplicación.
- Una última carpeta que implementa la capa de presentación (*presentation*).

Estructura global

Por último podemos ver el diagrama que componen la aplicación de interacción con el modelo en la figura C.5.

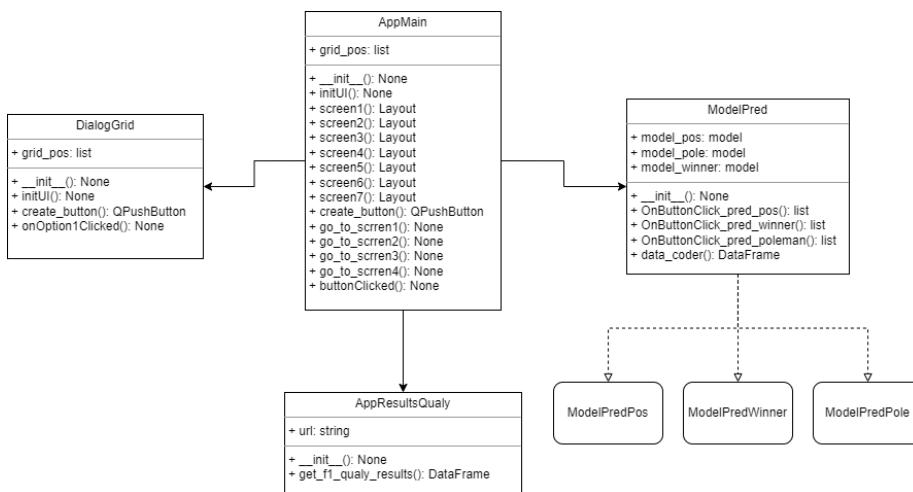


Figura C.5: Diagrama de clases de la aplicación.

Apéndice D

Documentación técnica de programación

D.1. Introducción

La documentación técnica de programación se describe en este anexo. Para facilitar que cualquiera pueda trabajar en este proyecto o continuararlo, se incluye la instalación de IDEs, la estructura de archivos y de la aplicación, su compilación o la configuración de varios servicios utilizados.

D.2. Estructura de directorios

Podemos encontrar la estructura del proyecto en el [repositorio de GitHub](#), Descrita a continuación:

- **/**: Es el directorio raíz del proyecto, en el podemos encontrar el README del repositorio, el documento de licencia y el archivo .gitignore (el cual nos permite ignorar documentos en el rastreo de control de versiones), además de los archivos de requerimientos y runtime.
- **/code/**: Contiene los ficheros para la recopilación, preparación, limpieza y codificación de los datos, además del entrenamiento y ajuste de los modelos. También contiene el fichero con el que se han generado los gráficos necesarios.
- **/code/data/**: datos descargados y codificados.

- **/code/data/data_trained_models/**: datos utilizados en los entrenamientos que obtuvieron los mejores datos de la cross-validation.
- **/code/trained_models/**: contiene los modelos entrenados.
- **/code/best_split_data/**: contiene la mejor división de los datos para los modelos a ajustar.
- **/code/coders/**: codificadores utilizados.
- **/docs/**: encontraremos toda la documentación relativa al proyecto.
- **/docs/img/**: alberga las imágenes empleadas en la documentación relativa al proyecto.
- **/docs/text/**: En la carpeta “text” veremos los diversos documentos que se combinan para formar los documentos maestros.
- **/app/**: contiene los archivos relativos a la aplicación.
- **/app/release/**: contiene el ejecutable de la aplicación.
- **/app/presentation/**: contiene la capa de presentación de la aplicación.
- **/app/logic/**: contiene lo necesarios para la funcionalidad de la aplicación.
- **/app/data/**: datos necesarios usados en la aplicación.
- **/app/data/circuits**: imágenes de los circuitos usados en la aplicación.
- **/app/data/datascsv**: datos en formato csv de los pilotos, circuitos y constructores.
- **/app/data/drivers**: imágenes de los pilotos usados en la aplicación.
- **/app/data/img**: podremos descubrir las imágenes utilizadas en la aplicación.
- **/app/data/models**: encontraremos en esta carpeta los modelos entrenados y los codificadores usados por la app.
- **/app/data/pos**: encontraremos en esta carpeta las imágenes de las posiciones usadas para mostrar la predicción.
- **/app/data/weather**: encontraremos en esta carpeta las imágenes del clima a escoger en la aplicación.

D.3. Manual del programador

Los futuros desarrolladores que trabajen en el proyecto pueden consultar este manual para obtener orientación. Se detallará cómo configurar el entorno de desarrollo, dónde obtener el código fuente y qué se requiere para que funcione.

Requisitos

- Python 3.
- Visual Studio Code.
- GitHub Desktop o Git y Git LFS.

En la siguiente sección se muestra como instalar y configurar correctamente todos los componente.

Entorno de desarrollo

Python 3

Instalación de Python 3 en Windows:

1. Primero nos dirigirnos a la página oficial del Python en el apartado de **descargas**.
2. En segundo lugar descargaremos el archivo de instalación pinchando en “*Download Python 3.X.X*”.



Figura D.1: Página oficial de descargas de Python.

3. Después ejecutamos el archivo descargado y dejamos seleccionada la opción de “*Add Python to path*”, y más tarde hacemos clic en “*Install Now*”

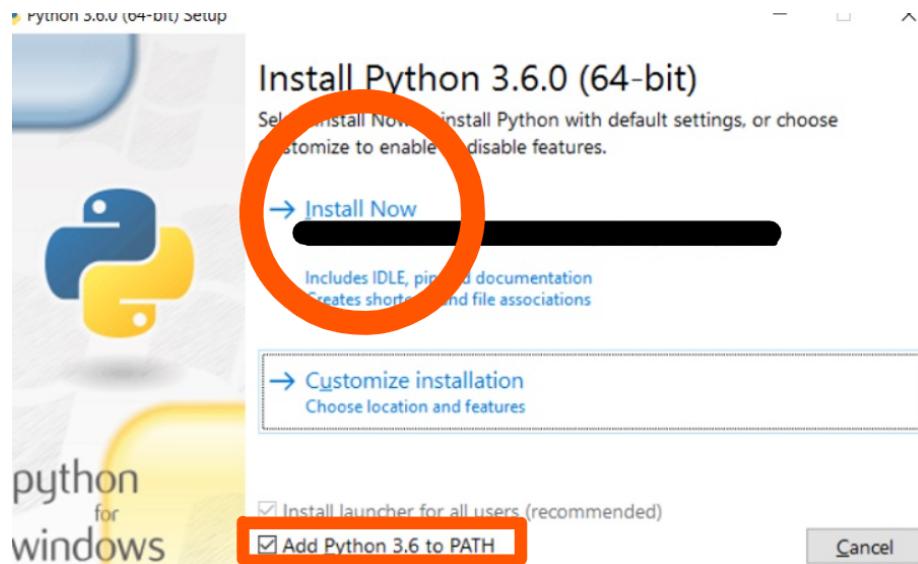


Figura D.2: Instalador de Python para Windows.

4. Dejamos terminar la instalación.

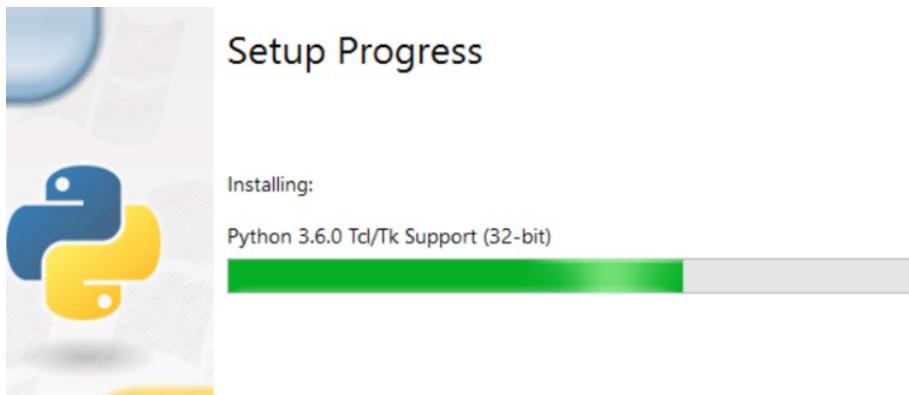


Figura D.3: Instalador de Python para Windows.

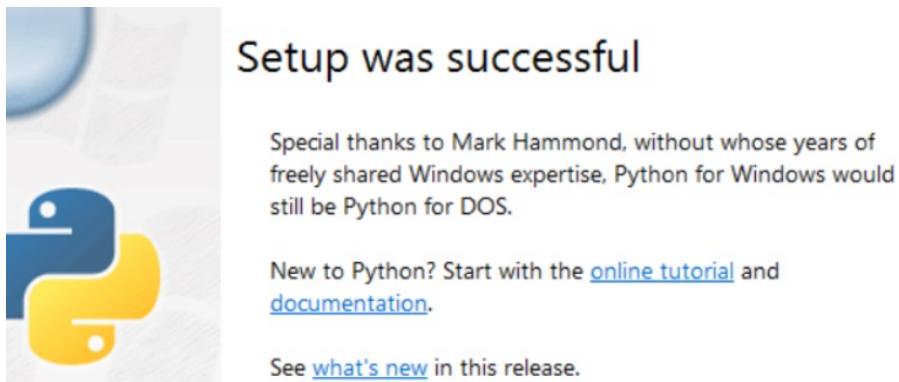


Figura D.4: Instalador de Python para Windows.

Instalación en Linux de Python 3:

1. Primero vamos a dirigirnos a la terminal de Linux.
2. En segundo lugar comprobamos si ya lo tenemos instalado gracias al comando “`python3 -version`”.
3. Si no lo tenemos instalado, ejecutaremos el comando “`sudo apt-get install python3.6`”
4. Esperamos a que termine la instalación y ya tendremos Python 3 instalado.

Visual Studio Code

Instalación de Visual Studio Code:

1. Primero debemos dirigirnos a la página oficial de [Visual Studio Code](#).
2. En segundo lugar descargaremos el archivo de instalación en función del sistema operativo que tengamos.

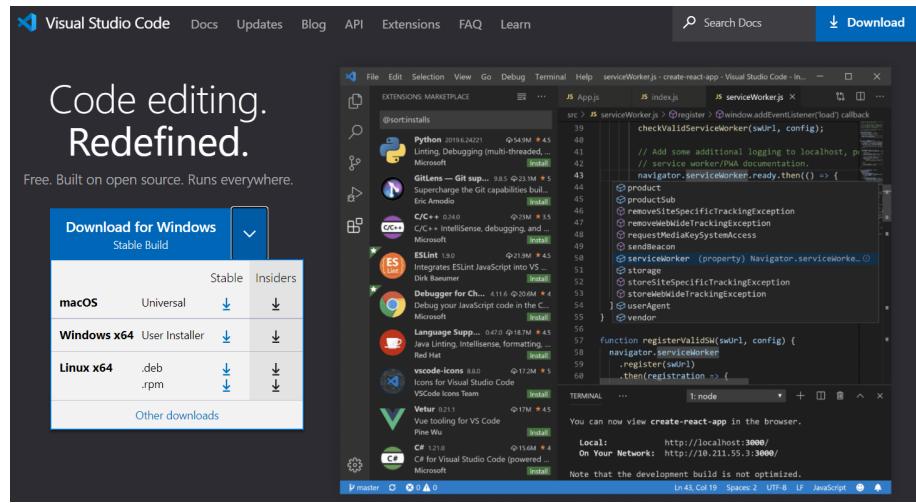


Figura D.5: Página oficial de Visual Studio Code.

3. Después únicamente debemos llevar a cabo los siguientes pasos del instalador.

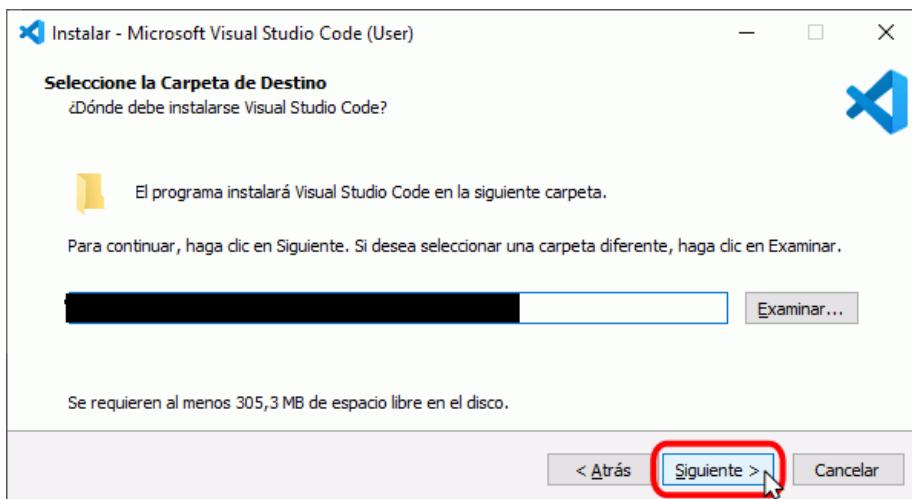


Figura D.6: Instalador de Visual Studio Code para Windows.

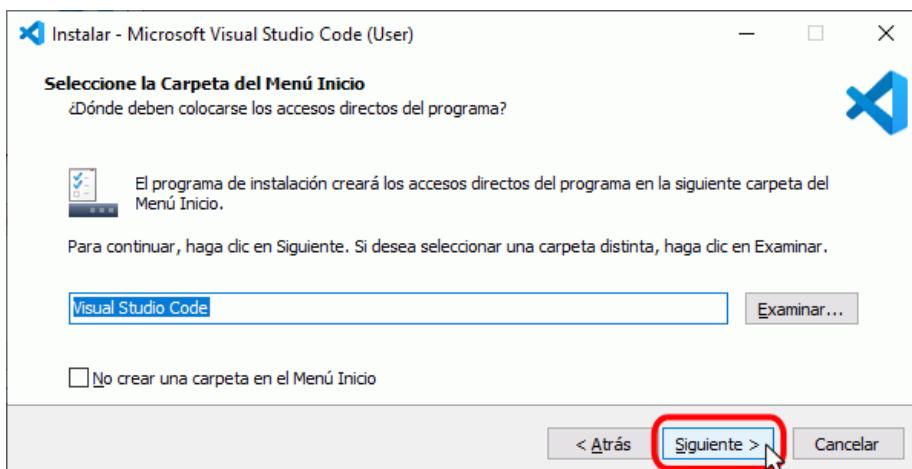


Figura D.7: Instalador de Visual Studio Code para Windows.

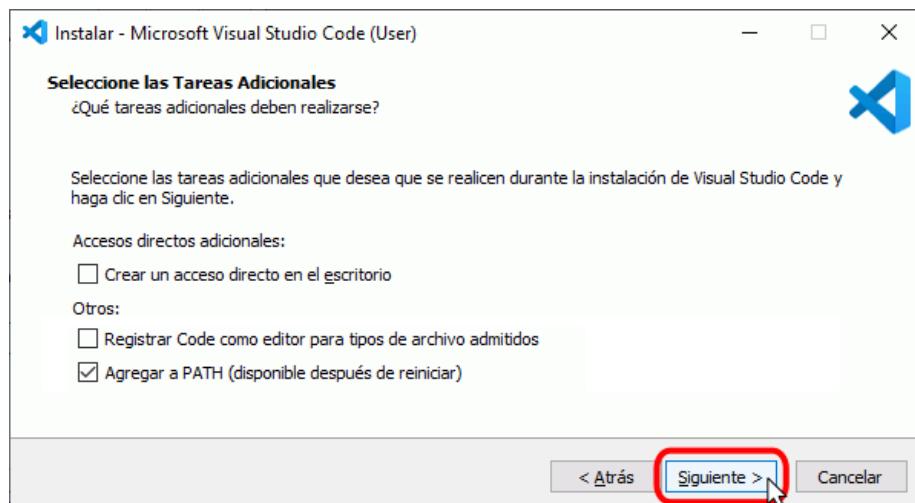


Figura D.8: Instalador de Visual Studio Code para Windows.

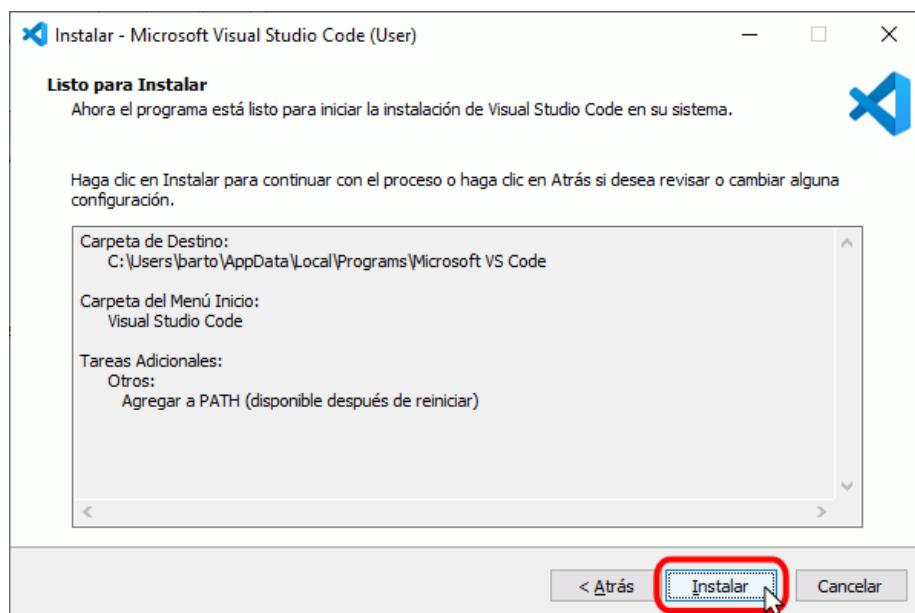


Figura D.9: Instalador de Visual Studio Code para Windows.

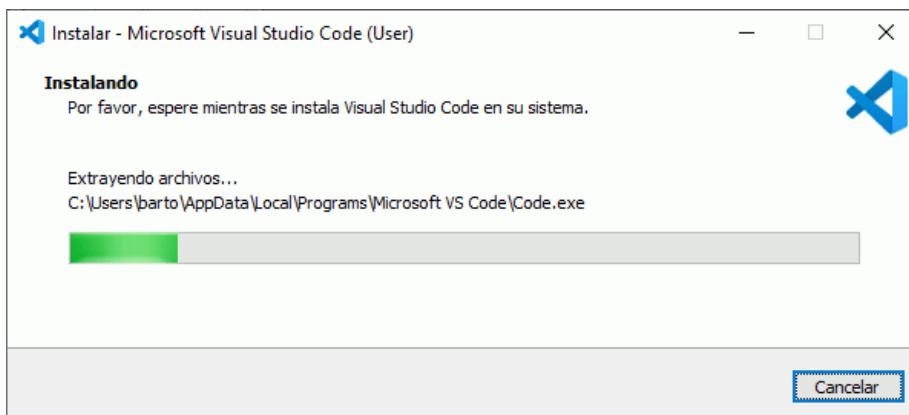


Figura D.10: Instalador de Visual Studio Code para Windows.

4. En el sistema operativo linux podremos ejecutar el comando *sudo apt install code* [4].

GitHub Desktop

Instalación de GitHub Desktop en windows:

1. Primero debemos dirigirnos a la página oficial de [GitHub Desktop](#).
2. En segundo lugar descargaremos el archivo de instalación en función del sistema operativo que tengamos (sólo disponible en Windows y MacOS).



Figura D.11: Página oficial de GitHub Desktop.

3. Despues únicamente debemos llevar a cabo los siguientes pasos del instalador (es posible que el instalador no redirija a un navegador web para iniciar sesión).

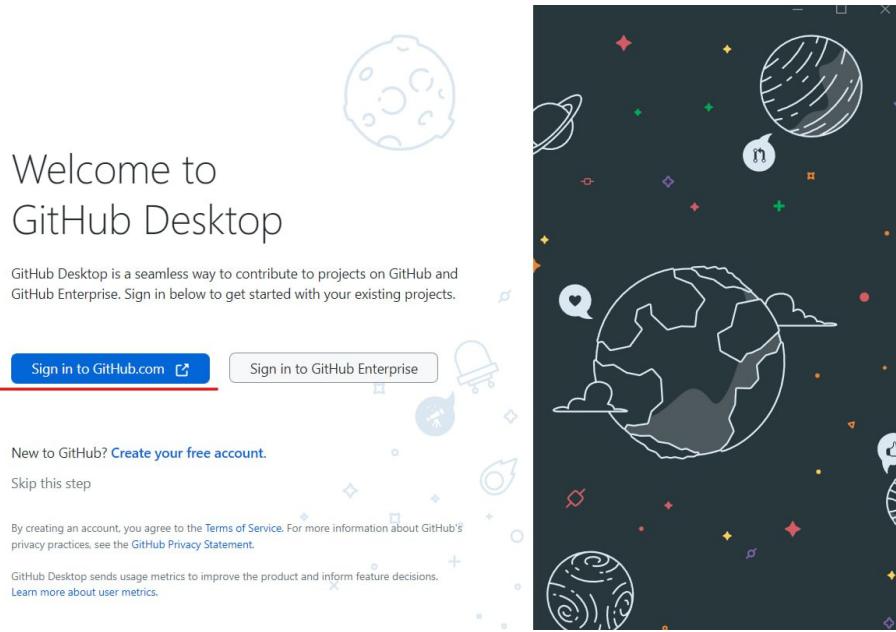


Figura D.12: Instalador de GitHub Desktop para Windows.

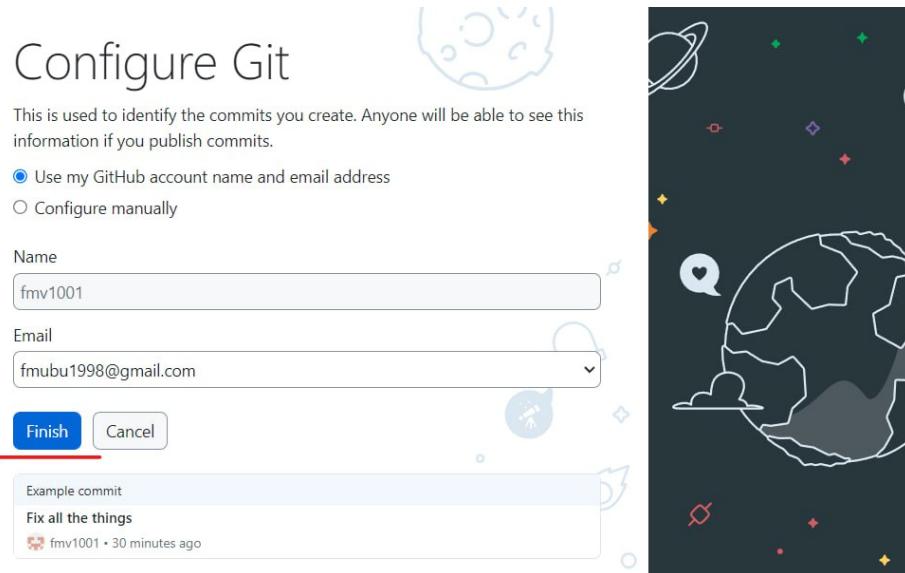


Figura D.13: Instalador de GitHub Desktop para Windows.

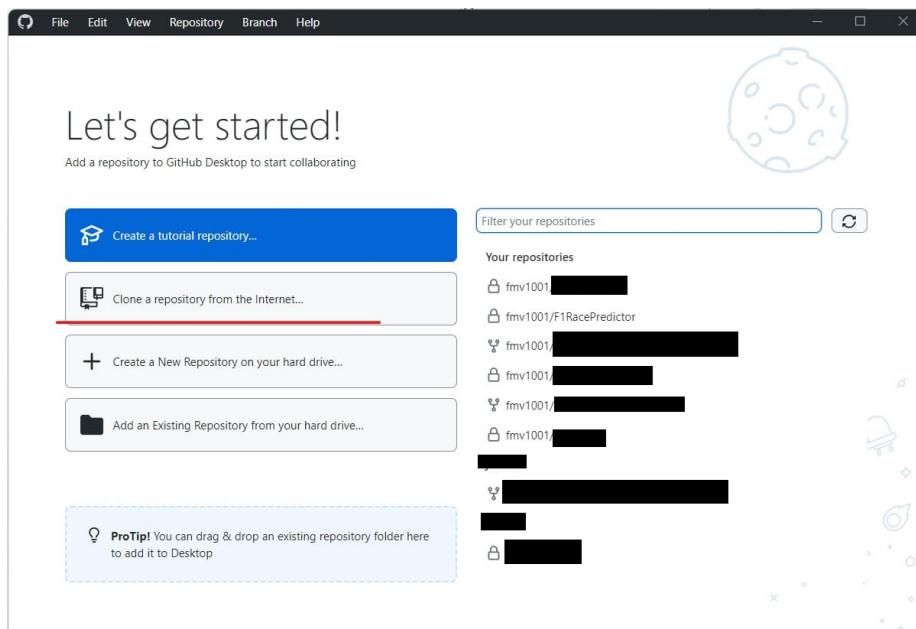


Figura D.14: Instalador de GitHub Desktop para Windows.

Esta aplicación sólo está disponible en Windows y MacOS pero en Linux contamos con la herramienta Git, tendremos las mismas funcionalidades pero mediante comandos.

Git

Instalación de Git:

1. Primero vamos a dirigirnos a la terminal de Linux.
2. En segundo lugar comprobamos si ya lo tenemos instalado gracias al comando “*git -version*”.
3. Si no lo tenemos instalado, ejecutaremos el comando “*sudo apt-get install git*”
4. Esperamos a que termine la instalación y ya tendremos Git instalado.

Git LFS

Instalación de Git LFS:

1. Primero vamos a dirigirnos a la terminal de Linux.
2. En segundo lugar comprobamos si ya lo tenemos instalado gracias al comando “*git lfs -version*”.
3. Si no lo tenemos instalado, ejecutaremos el comando “*sudo apt-get install git lfs*”
4. Esperamos a que termine la instalación y ya tendremos Git LFS instalado.

D.4. Compilación, instalación y ejecución del proyecto

Antes de continuar con la compilación, instalación y ejecución del proyecto, debemos descargar el [repositorio de GitHub](#). Para ello copiaremos la dirección del repositorio (<https://github.com/fmv1001/F1RacePredictor>) y en la herramienta GitHub desktop elegimos la opción *Clone a repository from the Internet...* (figura D.14) que clonara el repositorio en una carpeta local de nuestro dispositivo al seleccionar el apartado URL y pegar la del repositorio tal como vemos en la imagen D.15.

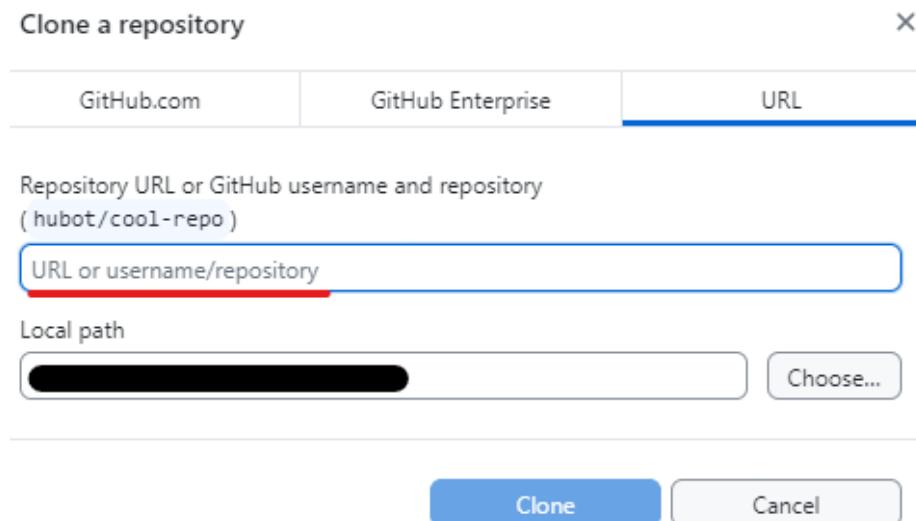


Figura D.15: Clonación del repositorio con GitHub Desktop.

En Linux únicamente debemos ejecutar este comando en una consola cuando estemos en la carpeta que deseemos alojarlo: “*git lfs clone https://github.com/fmv1001/F1RacePredictor*”

Importar el proyecto en Visual Studio Code

Una vez instalada la herramienta Visual Studio Code:

1. Abriremos Visual Studio Code.
2. Nos desplazaremos hasta la esquina superior izquierda y pinchamos en *File > Open Folder...*

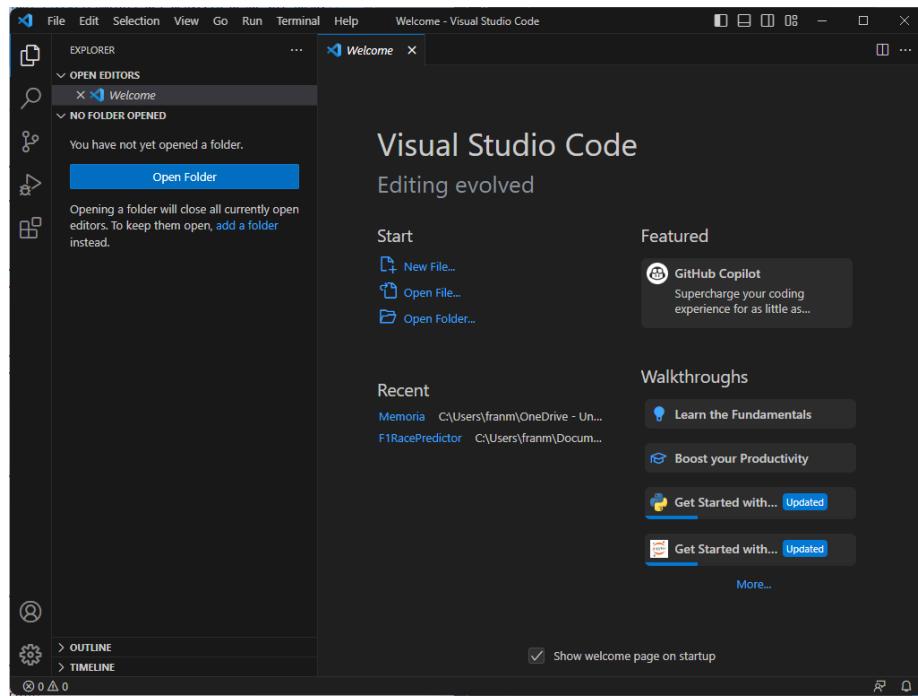


Figura D.16: Importar proyecto en Visual Studio Code.

3. Una vez lleguemos a este momento, debemos navegar hasta la carpeta en la que hemos clonado el proyecto, luego hacemos clic en "*Seleccionar carpeta*". En este punto sólo debemos esperar a que Visual Studio Code cargue los archivos del proyecto.

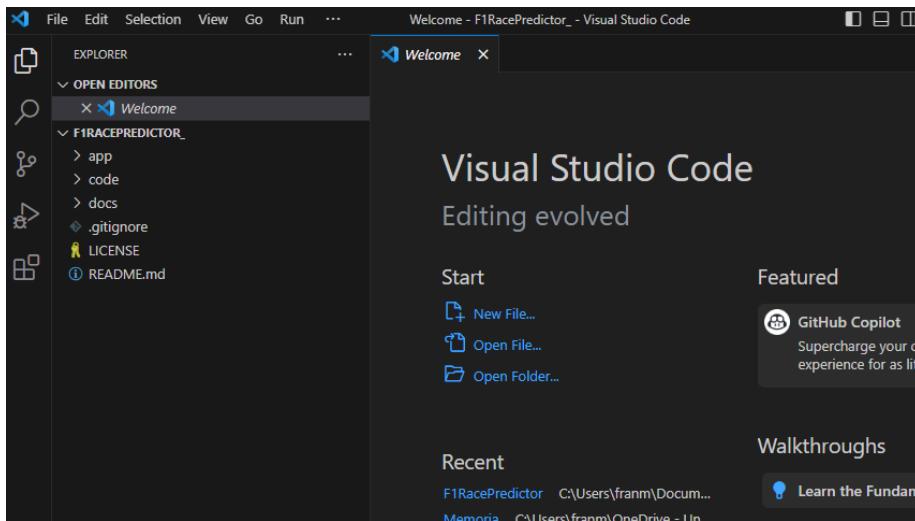


Figura D.17: Proyecto importado en Visual Studio Code

Añadir funcionalidad al sistema

Una vez importado el proyecto, podremos realizar las modificaciones deseadas. Si queremos añadir alguna funcionalidad, debemos realizar como mínimo el siguiente paso:

- Crear una nueva rama en desde la rama principal (*main*) del proyecto desde GitHub Desktop, imagen D.18.

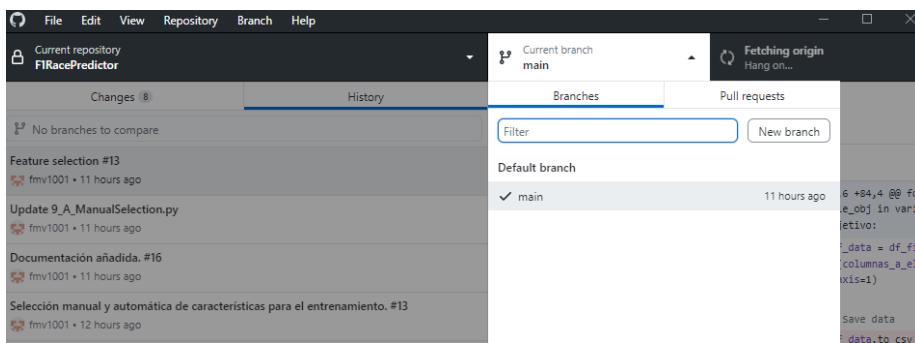


Figura D.18: Crear una nueva rama desde GitHub Desktop

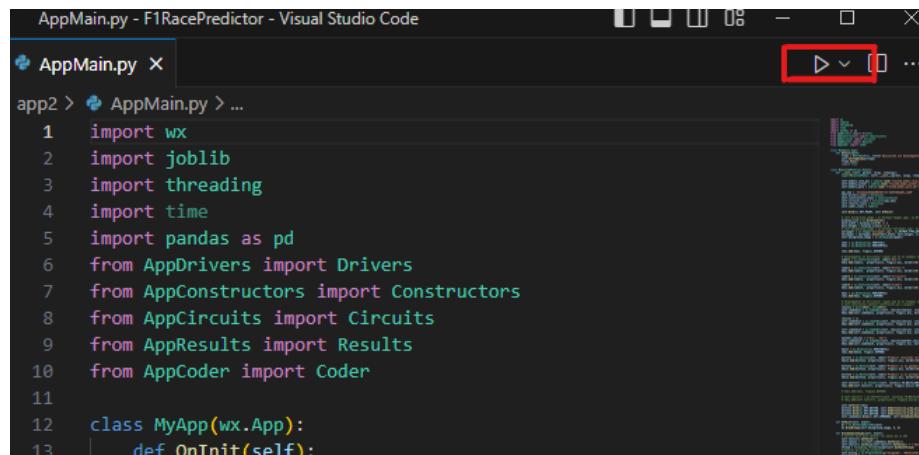
O bien en Linux ejecutar el siguiente comando en una terminal: *git checkout -b export-data main*

Compilación del código del proyecto

Para compilar tanto la aplicación como los demás archivos, únicamente debemos ejecutarlos con python.

Ejecución del sistema

Para la ejecución de la aplicación deberemos haber instalado previamente Visual Studio Code y Python 3 en el dispositivo. Tras la instalación de los componentes, nos dirigimos a Visual Studio Code, importamos el proyecto (tal como se explica en manual D.4). Después abrimos el archivo AppMain.py y en la esquina superior derecha ejecutamos pinchando el el ícono play/run.



```
AppMain.py - F1RacePredictor - Visual Studio Code
AppMain.py X
app2 > AppMain.py > ...
1 import wx
2 import joblib
3 import threading
4 import time
5 import pandas as pd
6 from AppDrivers import Drivers
7 from AppConstructors import Constructors
8 from AppCircuits import Circuits
9 from AppResults import Results
10 from AppCoder import Coder
11
12 class MyApp(wx.App):
13     def OnInit(self):
```

Figura D.19: Ejecución de la app desde Visual Studio Code

También podremos ejecutar la app desde comandos situándonos en la carpeta /app y desde el terminal ejecutamos el comando: *python3 presentation/AppMain.py*

Apéndice E

Documentación de usuario

E.1. Introducción

En esta sección, repasaremos los requisitos previos para configurar y usar el sistema, así como las instrucciones para realizar esa configuración. También se ha creado un manual de uso. Todo ello centrado en el usuario final.

E.2. Requisitos de usuarios

- Dispositivo con sistema operativo Windows, Linux, MacOS o incluso Android, con Python instalado
- Es necesario haber descargado los archivos necesarios (carpeta `/app/release`) disponibles desde el [repositorio](#).

E.3. Instalación

Para realizar la instalación de la aplicación para poder interactuar con el modelo únicamente es necesario hacer clic en el archivo F1Predictor.exe.

E.4. Manual del usuario

En este manual se explica cómo hacer uso de la aplicación para poder realizar predicciones. En este enlace ([Vídeo manual de usuario](#)), podremos ver este manual en formato vídeo.

Inicio de la aplicación

Cuando ejecutamos la aplicación lo primero que vamos a ver es la pantalla de inicio (figura E.1)



Figura E.1: Pantalla de inicio de la aplicación.

Selección del circuito

Procederemos dando clic en cualquier parte de la pantalla de inicio (figura E.1) y se nos muestran todos los circuitos de la temporada, visto en la figura E.2. Debemos escoger uno de ellos haciendo clic sobre él.



Figura E.2: Pantalla para escoger entre todos los circuitos de la temporada.

Elección de clima

Una vez hayamos escogido el circuito (imagen E.2), se nos muestran dos neumáticos, uno de seco y otro de lluvia como podemos ver en la figura E.3. Debemos escoger uno de ellos haciendo clic sobre él.



Figura E.3: Pantalla para escoger entre clima seco o mojado.

Predecir resultados

Con todas las opciones ya marcadas veremos la pantalla de predicciones como vemos en la figura E.4 y procedemos a seleccionar la predicción deseada.



Figura E.4: Pantalla para escoger entre las opciones de predicción.

Predecir resultados de carrera

Si hacemos clic en la predicción de las posiciones de carrera (figura E.5), se nos mostrará uno a uno las posiciones predichas por el modelo, lo cual podemos apreciar en la imagen E.6.

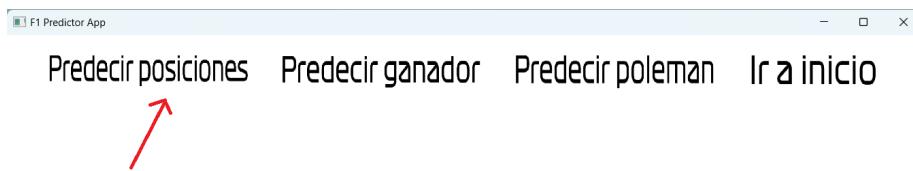


Figura E.5: Pantalla para escoger la predicción de las posiciones de carrera.

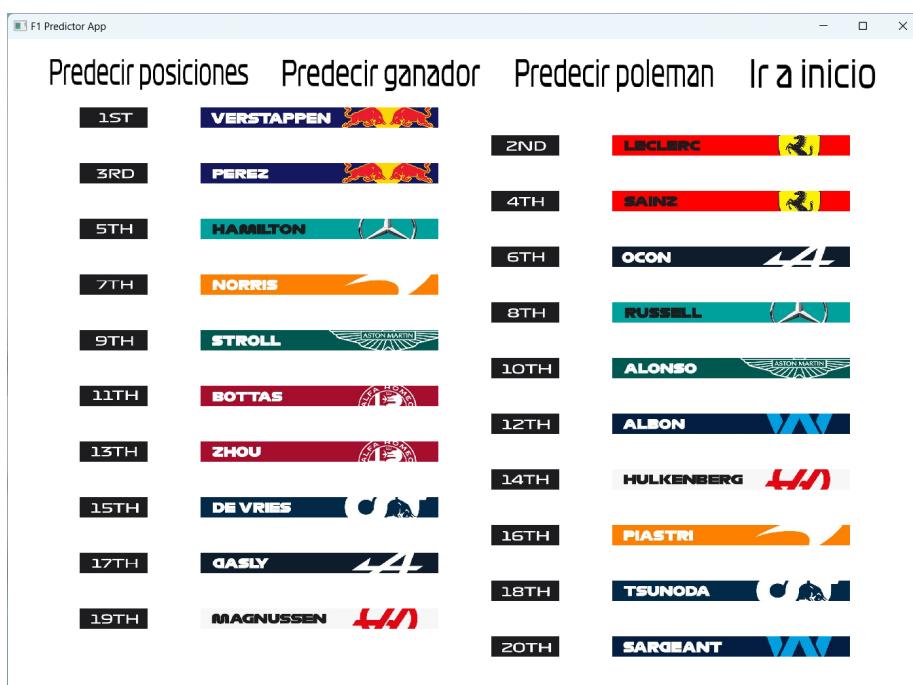


Figura E.6: Posiciones predichas por el modelo.

Predecir el ganador de la carrera

Si hacemos clic en la predicción del ganador de la carrera (figura E.7), se nos mostrará el ganador predicho por el modelo, lo cual podemos apreciar en la imagen E.6.

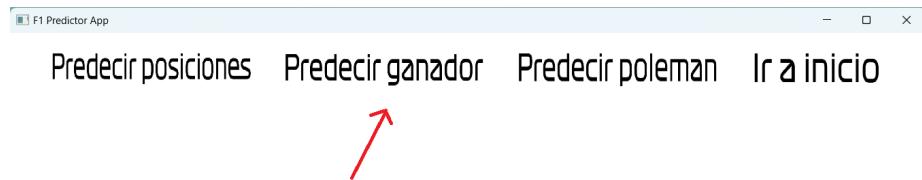


Figura E.7: Pantalla para escoger la predicción de las posiciones de carrera.

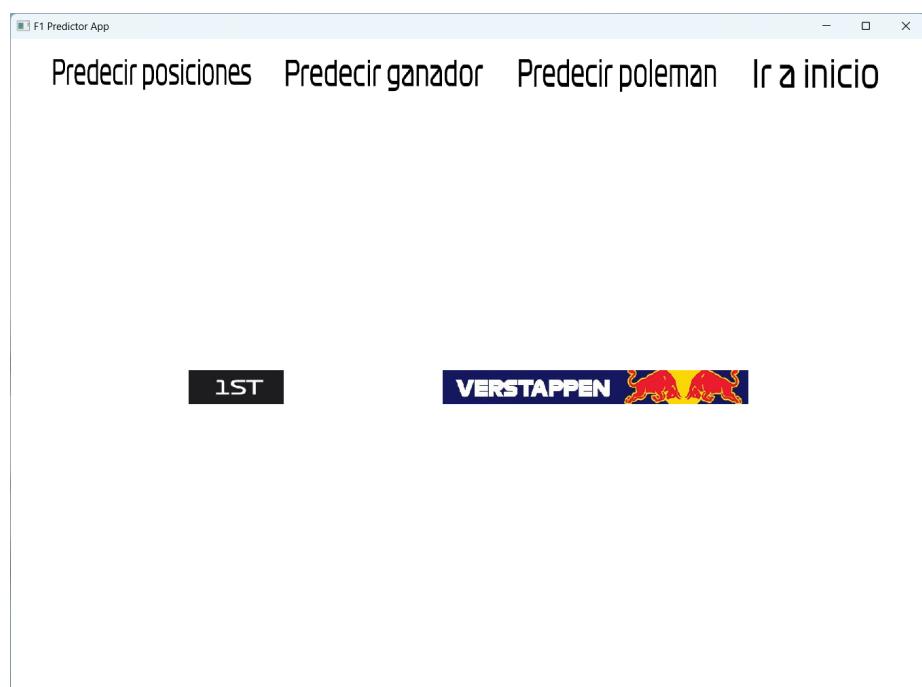


Figura E.8: Ganador predicho por el modelo.

Predecir el poleman

Si hacemos click en la predicción del poleman (figura E.9), se nos mostrará el poleman predicho por el modelo, lo cual podemos apreciar en la imagen E.10.

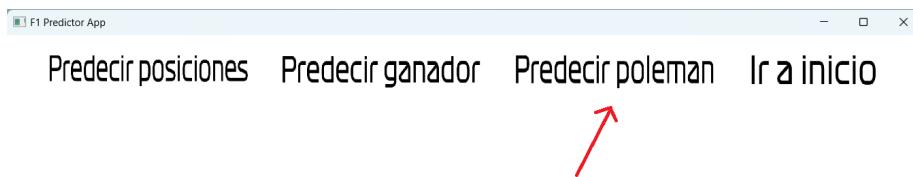


Figura E.9: Pantalla para escoger la predicción de las posiciones de carrera.

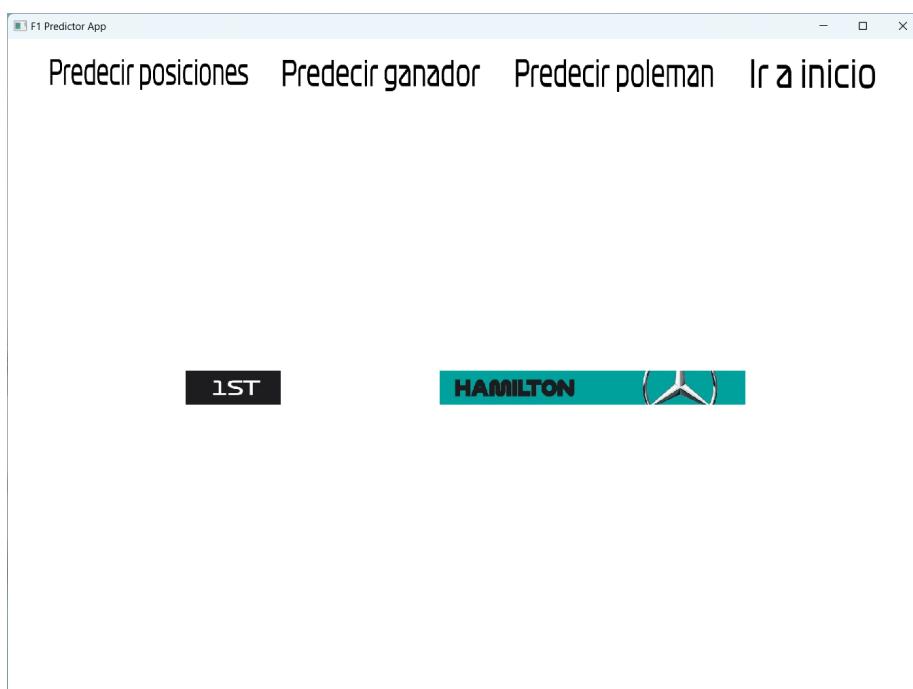


Figura E.10: Ganador de la pole predicho por el modelo.

Predicciones de resultados de carrera o ganador de circuitos en los que no se ha llevado a cabo la clasificación

Puede darse el caso de que no se haya realizado la clasificación en el momento de hacer la predicción. En estos casos la aplicación nos va a pedir uno a uno la posición de salida del piloto para poder predecir el resultado. Esto puede ser interesante porque podemos ver las diferencias entre salir en una posición u otra. En la figura E.11 podemos ver cómo se nos pide la posición de salida del piloto Albon, y más adelante las posiciones disponibles

para asignar al piloto Leclerc en la figura E.12. Para asignar la posición, basta con hacer clic en la imagen de la posición que decidamos.

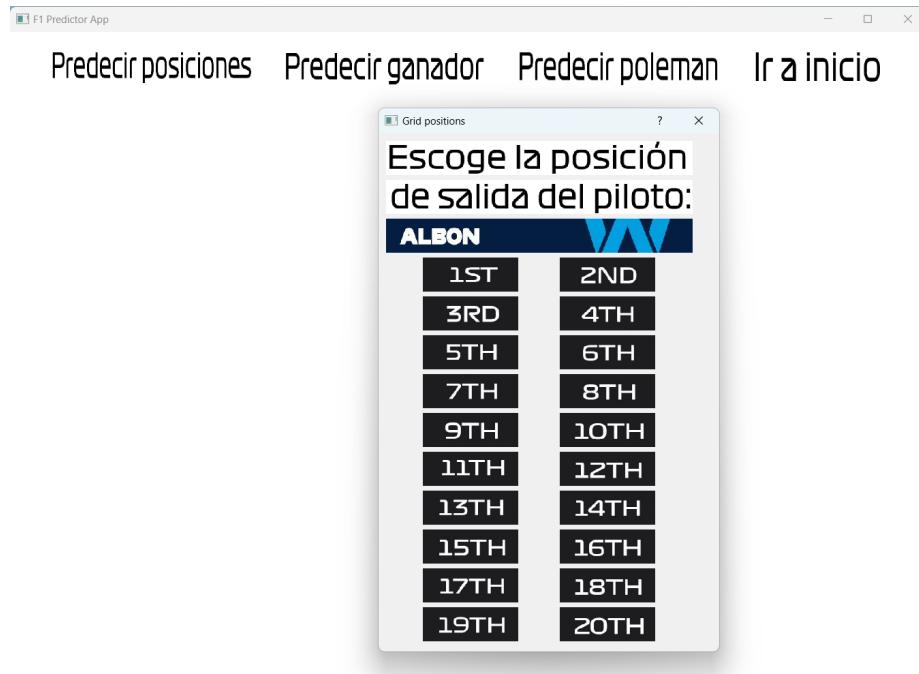


Figura E.11: Pantalla para escoger la posición de salida del piloto Albon.



Figura E.12: Pantalla para escoger la posición de salida del piloto Leclerc.

Volver a inicio

Para concluir este manual, en el caso de querer hacer una nueva predicción podemos volver al inicio desde el botón oportuno que se nos muestra en la imagen E.13.

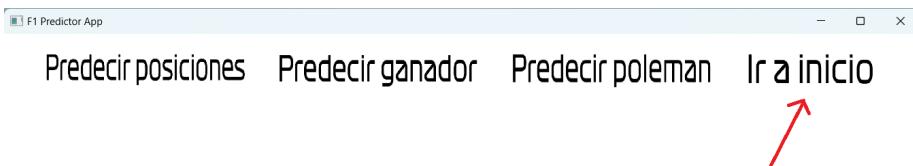


Figura E.13: Botón para volver a la pantalla de inicio de la aplicación.

Bibliografía

- [1] Gobierno de España. Bases y tipos de cotización 2023, 2023.
- [2] GNU. Lista de licencias con comentarios, 2023.
- [3] IEEE. Ieee-std-830-1998 : Especificaciones de los requisitos del software, 2010.
- [4] Microsoft. Visual studio code on raspberry pi, 2023.
- [5] Agencia Tributaria. Principales novedades tributarias introducidas por la ley 11/2020, de presupuestos generales del estado para 2021 (boe de 31 de diciembre), 2021.