



24 DE MAYO DE 2023


DESARROLLO DE UN COMPLEMENTO DE SOFTWARE PARA EL VIDEOJUEGO IRACING

[SUBTÍTULO DEL DOCUMENTO]

VICENTE IZQUIERDO FORMENT

[NOMBRE DE LA EMPRESA]

[Dirección de la compañía]



Resumen

Castellano

El objetivo de este proyecto es el desarrollo de una aplicación auxiliar al videojuego de simulación de carreras online iRacing, con la que obtendremos información adicional sobre la carrera en la que participamos (clasificación en tiempo real, tiempo de vuelta del resto de participantes, consumo de combustible, cálculo de repostaje de combustible...), información que facilitará al jugador la toma de decisiones durante su carrera. Desarrollado utilizando el lenguaje de programación Python 3.11.2, SQLite 3.41.2, PyQt 5.15.4 y Python iRacing SDK (pyirsdk).

English

The objective of this project is the development of an auxiliary application to the online racing simulation video game iRacing, with which we will obtain additional information about the race in which we participate (real-time classification, lap time of the other participants, fuel consumption, refueling calculation...), information that will help the player to make decisions during the race. Developed using the programming language Python 3.11.2, SQLite 3.41.2, PyQt 5.15.4 and Python iRacing SDK (pyirsdk).

Valencià

L'objectiu d'aquest projecte és el desenvolupament d'una aplicació auxiliar al videojoc de simulació de carreres en línia iRacing, amb la qual obtindrem informació addicional sobre la cursa en què participem (classificació en temps real, temps de tornada de la resta de participants, consum de combustible, càlcul de proveïment de combustible...), informació que facilitarà al jugador la presa de decisions durant la seva carrera. Desenvolupat utilitzant el llenguatge de programació Python 3.11.2, SQLite 3.41.2, PyQt 5.15.4 i Python iRacing SDK (pyirsdk).

Índice

1. Introducción.
2. Estudio de viabilidad. Método DAFO.
 1. Estudio de mercado.
 1. Viabilidad técnica/económica del proyecto
 2. Recursos HW
 3. Recursos SW
 4. Recursos humanos
 5. Viabilidad temporal
 2. Planificación temporal o agenda de trabajo.
3. Análisis de requisitos
 1. Descripción de requisitos.
 1. Texto explicativo
 2. Diagramas de caso de uso de los más relevantes. Realizando un caso de uso general y si es necesario otros diagramas más específicos.
4. Diseño
 1. Diseño Conceptual Entidad Relación
 2. Diseño Lógico Relacional o Paso a tablas.
 3. Diseño Físico (paso a tablas, optimizaciones)
 4. Descripción de las tablas y campos.
 5. Orientación a objetos:
 1. Diagramas de clases. Descripción de clases y atributos.
 2. Diagrama de secuencias. De lo más relevante.
 3. Diagrama de actividad. De lo más relevante.
 6. Diseño UX.
 7. Mockups.
5. Codificación.
 1. Tecnologías elegidas y su justificación (lenguajes, frameworks, librerías, etc.)
 1. Desarrollo de servicios.
 1. Descripción general.
 2. Seguridad.
 2. Desarrollo multiplataforma.
 1. Descripción general.
 2. Asegurar la funcionalidad en distintos dispositivos.
 2. Documentación interna de código (puede ir en un anexo).
 1. Descripción de cada fichero. Autor, función y fecha de creación.
 2. Descripción de cada función. Autor, función y fecha de creación.
 3. Documentación externa (puede ir en un anexo).
 4. Manual del usuario.
6. Despliegue
 1. Diagramas de despliegue
 2. Descripción de la instalación o despliegue
7. Herramientas de apoyo
8. Control de versiones.
9. Sistemas de integración continua.
10. Gestión de pruebas
11. Conclusiones.
 1. Conclusiones sobre el trabajo realizado
 2. Conclusiones personales
 3. Posibles ampliaciones y mejoras
12. Bibliografía (comentada)
13. Libros, artículos y apuntes
14. Direcciones web

1. Introducción.

El mundo de las carreras de simulación ha experimentado un gran auge en los últimos años, y una de las plataformas más populares es iRacing. Con la intención de mejorar la experiencia de los usuarios en este juego, se ha desarrollado un programa auxiliar con el que se pretende ayudar al jugador con información adicional que le ayude a la toma de decisiones durante la carrera. Esta app tiene dos funcionalidades principales: la ventana Clasificaciones y la ventana Calculadora de combustible.

La aplicación arranca en un sencillo menú principal que nos permite arrancar cualquiera de las dos ventanas principales mediante unos botones. Además, se nos indica si iRacing está en marcha y nos deja arrancar el programa, en caso de no estar disponible, se mantiene a la espera hasta que arranque el juego.

La ventana Clasificaciones muestra una tabla con las posiciones de carrera actualizadas en tiempo real, así como también información útil sobre los pilotos. Como su mejor tiempo de vuelta, su última vuelta, iRating y licencia de piloto... Junto con esto disponemos de información relacionada con la sesión en la que participamos, se mostrará el nombre del vehículo del jugador, nombre de la sesión en la que se participa, SOF (Strenght Of Field, la media de iRating de los participantes) de la carrera, temperatura de pista, número de vueltas restantes y el tiempo restante de carrera.

Por otro lado, la ventana Calculadora de combustible, es una herramienta esencial para cualquier conductor serio que quiera maximizar su rendimiento. Calcula la cantidad de combustible que el usuario necesita para completar la carrera y proporciona información importante sobre los pit stops, incluyendo el momento adecuado para entrar en boxes y la cantidad de combustible a repostar, así como la cantidad de vueltas restantes con el combustible actual.

En resumen, el programa auxiliar que se ha desarrollado para iRacing tiene como objetivo proporcionar información clave y ayudar a los usuarios a tomar decisiones estratégicas durante la carrera. Esto mejorará la experiencia de los jugadores y los ayudará a competir con mayor éxito.

2. CheatSheet

A continuación, voy a explicar unos conceptos que me he percatado mientras realizaba la memoria, que son términos específicos dentro del mundo del simracing o del propio iRacing y que puede llevar a confusión la lectura de la siguiente memoria si no se conocen dichos términos. Los que considero más importantes son:

Simracing: o automovilismo virtual, consiste en competir en carreras virtuales utilizando software de simulación de carreras realistas. Los jugadores utilizan diferentes periféricos (volantes, pedales, cambios de marcha...) anclados a chasis (cockpit) para controlar los vehículos en el juego y competir contra otros jugadores en línea. Es una forma popular de entrenamiento para pilotos de carreras reales, ya que permite con una inversión mínima, el poder aprender circuitos a miles de kilómetros de distancia sin moverte de casa.

Cockpit: o chasis, es la base sobre la que se montan los simuladores, son generalmente de aluminio, aunque pueden ser de madera, hierro, PVC... Suelen estar compuestos por un

asiento, un puente donde se coloca el volante, y un soporte para los pedales. Opcionalmente pueden llevar integrados un soporte para uno o varios monitos, soportes para teclado y ratón, soportes para un PC. Suelen ser modulares y ajustables a las necesidades de cada jugador.



iRacing: es la plataforma online de simulación de carreras por excelencia, permite a los jugadores competir en un sistema basado en campeonatos de diferentes categorías y tipos de vehículos (**Road, Dirt, Oval**), donde los jugadores participan en carreras programadas. Los campeonatos se dividen en temporadas y se clasifican por **licencias** (A, B, C, D, Rookie), en los que solamente podrán participar los jugadores que tengan una licencia igual o superior a la del campeonato. Además, los jugadores serán separados en diferentes **splits** (servidores) según su nivel de **iRating** para que estos corran con gente de nivel similar. Todo esto se sustenta gracias a un sistema de reportes que permite denunciar a los jugadores que incumplan las normas, llegando incluso a suspender las cuentas de los jugadores reincidentes.

Licencias: son los “carnets” virtuales del juego, dan acceso a los diferentes campeonatos y dependen enteramente del Safety Rating. Se puede tanto subir como bajar de licencia según nuestro comportamiento en pista, a excepción de Rookie, una vez que hayamos salido no podremos volver a descender a Rookie. Las licencias se miden según el Safety rating, todas van de 0 a 4.99, este se reinicia cada temporada y determina que pasa con tu licencia al final de dicha temporada. Si al final de la temporada tenemos menos de 2.0 de Safety Rating, descenderemos de licencia (Ej. B > C), y si tenemos más de 3.0 ascenderemos (Ej. B > A). Pero en caso de bajar de 1.0 o subir de 4.0 descenderemos o ascenderemos automáticamente independientemente del momento de la temporada en el que estemos. Cada categoría tiene su propia licencia, por lo que podemos tener clase B en Road y ser Rookies en Oval.

Rookie: es la licencia con la que todo jugador empieza al crear la cuenta por primera vez, se empieza con 2.5 de Safety Rating. Para poder subir a clase D hay que participar en mínimo 4 carreras y subir a 3.0 de Safety Rating. En esta categoría solamente podremos participar en un número limitado de campeonatos, siendo el más icónico el Global Mazda MX5 Cup.

Temporadas: es la manera en la que se dividen los campeonatos dentro de iRacing. Cada año se divide en 4 temporadas de 12 + 1 semanas, empezando en febrero y terminando en enero del año siguiente. Las 12 primeras semanas de cada temporada se corre un calendario previamente estipulado, en el que se indica los horarios y los circuitos de cada semana y cada martes se avanza a la siguiente semana del calendario. Al terminar la semana 12, se entra en la denominada “Week 13”, que es una semana de transición entre temporadas donde se eliminan la mayoría de campeonatos y a excepción de 2, ninguno cuenta como carrera oficial, por lo que ni ganamos ni perdemos SR o iR. Durante esta semana se publican los calendarios de la temporada siguiente y se realizan las actualizaciones del juego que tienen previstas, estas se prueban durante esta semana para evitar fallos durante la temporada.

Safety Rating: es la puntuación con la que el juego mide la seguridad de los pilotos, durante las carreras, se suma un pequeño porcentaje por cada curva completada sin incidentes y se resta por cada incidente cometido, por lo que, para aumentar al máximo deberemos correr en circuitos con el mayor número de curvas posible y durante el mayor tiempo sin cometer ningún incidente. Por lo que, si tenemos 10 incidentes en una carrera de 45 minutos en un circuito de 18 curvas, ganaremos más Safety Rating que si tenemos 5 incidentes en una carrera de 15 minutos en un circuito de 8 curvas.

Incidentes: son los incidentes que ocurren durante la carrera y suelen tener un límite de 17 para carreras inferiores a 45 minutos, 25 para carreras de más de 45 minutos y para carreras de resistencia (6h, 12h 24h) pueden llegar hasta los 100 incidentes. Si superamos dicho límite de incidentes, seremos desclasificados y expulsados de la carrera. Los incidentes se pueden clasificar en:

- x0 Contact: Ligero roce con otro piloto o con un muro sin llegar a tocarse, son los únicos que no restan Safety Rating.
- x1 Off track: Se producen al sobrepasar los límites de pista por parte de los pilotos, generalmente sacar el neumático exterior completo fuera de dichos límites.
- x2 Loss Of Control: Se producen al perder el control del vehículo, generalmente al realizar un trompo.
- x4: se producen al impactar dos vehículos de manera más fuerte que en el X0, aunque no se lleguen a dañarse ni a salirse de pista en ningún momento, el x4 lo reciben ambos pilotos, tengan o no la culpa de recibir el golpe.

iRating: es la puntuación con la que el sistema mide la habilidad de los pilotos, se obtiene generalmente según la posición de carrera en la que terminemos, pero depende también de

nuestro iRating y del SOF del servidor. Si tenemos 1000 de iR y quedamos en 5º posición con un SOF de 4000, ganaremos mas iR que si quedamos en 5º posición en un SOF de 850. Éste determina en que Split correremos, ya que el sistema nos asignara el servidor con el SOF mas parecido a nuestro iRating.

Split: son las divisiones que crea el sistema para repartir los jugadores lo mas igualados posibles. Depende de la cantidad de jugadores apuntados en la carrera y de su iRating, si hay 100 jugadores, creara 5 splits con 20 jugadores, que se ordenaran según su iR, colocando a los 20 primeros en el primer Split, los siguientes 20 en el segundo y así sucesivamente.

Categorías: Actualmente existen 4 categorías totalmente separadas entre ellas en iRacing y son las siguientes:

- Road: son las carreras tal y como las conocemos en Europa, se corren con diferentes tipos de vehículos en circuitos con curvas entrelazadas entre sí, tanto a derechas como a izquierdas. Suelen requerir una mayor habilidad de conducción y conocimiento del circuito, en las que suele ganar el más rápido. Por ejemplo, la Formula 1, entraría en esta categoría.
- Oval: son las carreras que se corren en circuitos Ovals, mayoritariamente en los Estados Unidos. Son siempre en circuitos con forma de Óvalo con curvas a izquierdas. No requieren de tanta habilidad de conducción, pero suelen tener una igualdad máxima entre todos los participantes, determinando la victoria a la estrategia de paradas y adelantamientos, suelen ser mas emocionantes y no decirse la victoria hasta el último momento.
- Dirt Road: son las carreras que se corren en circuitos con características similares a Road, pero sobre terrenos no asfaltados (tierra) a los que se les añaden espectaculares saltos en la pista, por lo que los vehículos pasan gran parte del tiempo con las ruedas en el aire.
- Dirt Oval: son las carreras que ocurren en circuitos ovals de tierra, se caracterizan por que los vehículos pasan grandes cantidades de tiempo derrapando, pero como en Oval, únicamente tienen curvas a izquierdas.

2. Estudio de viabilidad. Método DAFO.

1. Estudio de mercado.

1. Viabilidad técnica/económica del proyecto
2. Recursos HW
3. Recursos SW
4. Recursos humanos
5. Viabilidad temporal

2. Planificación temporal o agenda de trabajo.

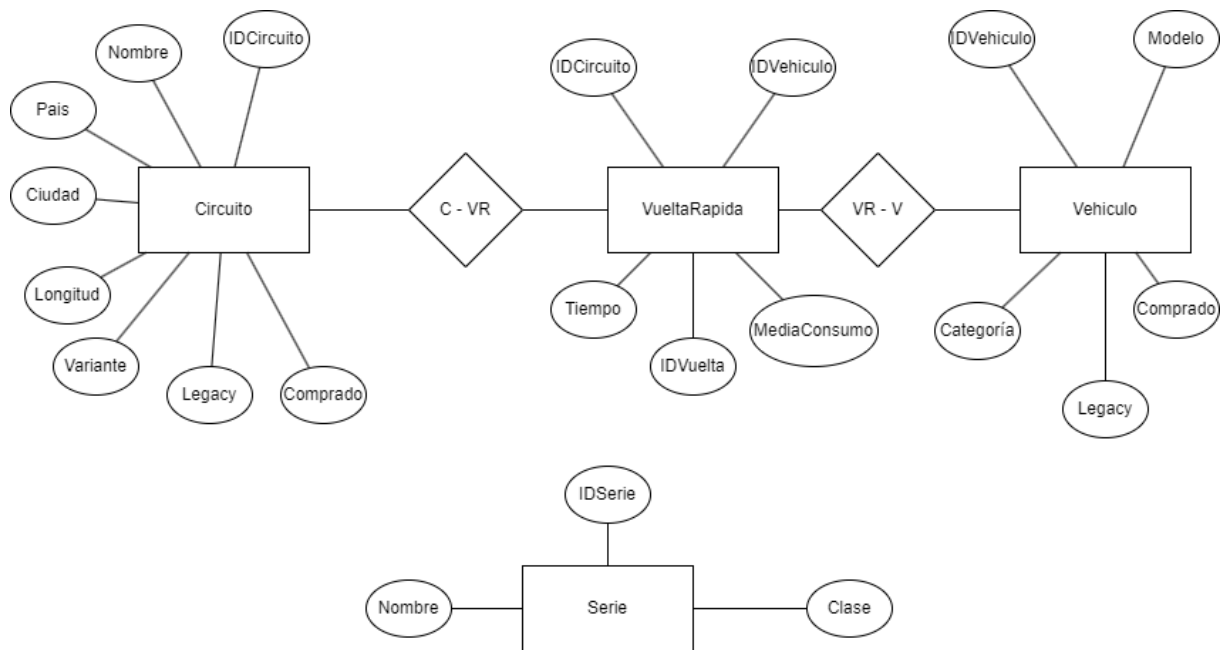
3. Análisis de requisitos

3. Descripción de requisitos.

1. Texto explicativo
2. Diagramas de caso de uso de los más relevantes. Realizando un caso de uso general y si es necesario otros diagramas más específicos.

4. Diseño

1. Diseño Conceptual Entidad Relación



Como se puede observar en el esquema, la base de datos estará formada por las siguientes entidades:

Circuito:

- IDCircuito (identificador único del circuito)
- Nombre
- País
- Ciudad
- Longitud
- Variante
- Legacy
- Comprado

VueltaRapida:

- IDVuelta (identificador único de la vuelta rápida)
- IDCircuito (identificador del circuito al que pertenece la vuelta rápida, clave externa hacia la tabla Circuito)
- IDVehiculo (identificador del vehículo que realizó la vuelta rápida, clave externa hacia la tabla Vehículo)
- Tiempo
- MediaConsumo

Vehículo:

- IDVehiculo (identificador único del vehículo)
- Modelo
- Categoría
- Legacy
- comprado

Serie:

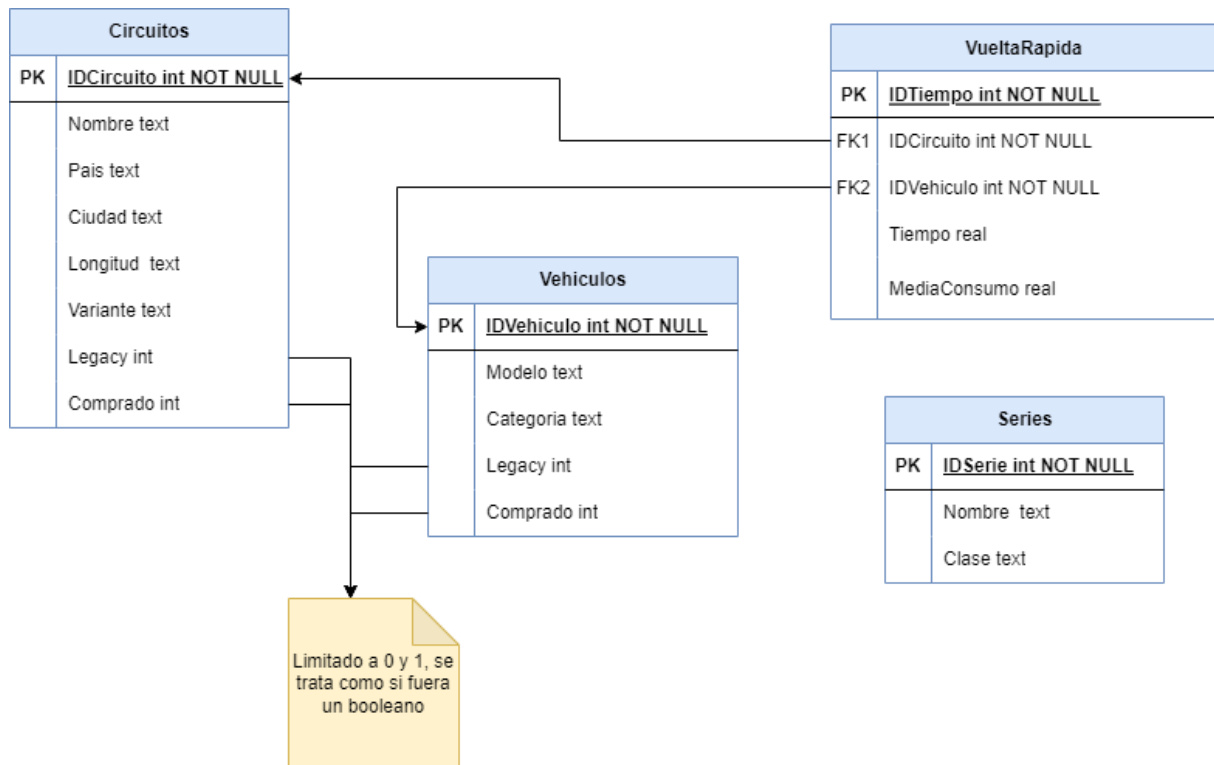
- IDSerie (identificador único de la serie)
- Nombre
- Clase

Éstas se entrelazan mediante las siguientes relaciones:

- La tabla VueltaRapida se relaciona con la tabla Circuito mediante el atributo IDCircuito. Esta relación indica que una vuelta rápida pertenece a un circuito, y un circuito puede tener varias vueltas rápidas.
- La tabla VueltaRapida se relaciona con la tabla Vehículos mediante el atributo IDVehiculo. Esta relación indica que una vuelta rápida es realizada por un vehículo, y un vehículo puede tener varias vueltas rápidas.

2. Diseño Lógico Relacional o Paso a tablas.

La estructura final de las tablas será la siguiente:



3. Diseño Físico (paso a tablas, optimizaciones)

Con el siguiente script sql crearemos las tablas de nuestra bbdd y le añadiremos algo de información a dichas tablas. Como en la parte de programación se va a controlar dicha información de la bbdd (añadiendo nuevos datos o actualizando los existentes), no se va a añadir excesiva información a la bbdd.

Utilizaremos el SGDB SQLite3, que nos permite gestionar de manera ligera y sencilla nuestra bbdd.

```
1. CREATE TABLE Vehiculos (
2.     IDVehiculo INTEGER NOT NULL PRIMARY KEY,
3.     Modelo TEXT,
4.     Categoria TEXT,
5.     Legacy INTEGER,
6.     Comprado INTEGER
7. )
```

```

8.      CREATE TABLE Circuitos (
9.      IDCircuito INTEGER NOT NULL PRIMARY KEY,
10.     Nombre TEXT,
11.     Pais TEXT,
12.     Ciudad TEXT,
13.     Longitud TEXT,
14.     Variante TEXT,
15.     Legacy INTEGER,
16.     Comprado INTEGER
17.     )
18.      CREATE TABLE VueltaRapida (
19.      IDTiempo INTEGER NOT NULL PRIMARY KEY,
20.      IDCircuito INTEGER,
21.      IDVehiculo INTEGER,
22.      Tiempo REAL,
23.      MediaConsumo REAL,
24.      FOREIGN KEY(IDCircuito) REFERENCES Circuitos(IDCircuito
25.      ),
26.      FOREIGN KEY(IDVehiculo) REFERENCES Vehiculos(IDVehiculo
27.      )
28.      INSERT INTO Circuitos VALUES
29.      (95, 'Sebring International Raceway', 'USA', 'Sebring,
30.      Florida', '5.95 km', 'International', 0, 1),
31.      (96, 'Sebring International Raceway', 'USA', 'Sebring,
32.      Florida', '3.17 km', 'Modified', 0, 1),
33.      (97, 'Sebring International Raceway', 'USA', 'Sebring,
34.      Florida', '2.81 km', 'Club', 0, 1),
35.      (47, 'Weathertech Raceway at Laguna
36.      Seca', 'USA', 'Monterey, California', '3.60 km', 'Full
37.      Course', 0, 1),
38.      (432, 'Watkins Glen International', 'USA', 'Watkins
39.      Glen, New York', '3.89 km', 'Classic', 0, 1),
40.
41.      INSERT INTO Vehiculos VALUES
42.      (1, 'Skip Barber Formula 2000', 'Skip Barber', 0, 1),
43.      (67, 'Mazda MX-5 Cup', 'MX-5 Cup', 0, 1),
44.      (112, 'Audi RS3 LMS', 'TCR', 0, 1),
45.      (119, 'Porsche 718 Cayman GT4 Clubsport
46.      MR', 'GT4', 0, 1),
47.      (144, 'Ferrari 488 GT3 Evo 2020', 'GT3', 0, 1),
48.      (148, 'iRacing iR-04', 'F4', 0, 1),
49.      (160, 'Toyota GR86', 'GR86', 0, 1)
50.
51.      INSERT INTO VueltaRapida VALUES
52.      (1, 47, 144, 86.375, 2.19),
53.      (2, 95, 144, 126.242, 3.39),
54.      (3, 403, 148, 95.500, 0.95),
55.      (4, 445, 144, 103.663, 2.51)

```

4. Descripción de las tablas y campos.

Tabla Vehículos: Tabla donde almacenaremos la información de los vehículos con los que se correrá en el juego. Está compuesta por los siguientes campos:

- IDVehiculo (Integer): N.º identificador del vehículo, utilizaremos los mismos id que utiliza iRacing para cada vehículo, para así poder buscarlos en la bbdd utilizando los datos que nos proporciona el sdk sin tener que hacer búsquedas por marca o modelo dentro de la bbdd. Es la clave primaria de la tabla.
- Marca (Text): Nombre de la marca del vehículo, lo utilizaremos para mostrar información al usuario junto con el modelo.
- Modelo (Text): Modelo del vehículo lo utilizaremos para mostrar información al usuario junto con la marca.
- Categoría (Text): Nombre de la categoría a la que pertenece el vehículo.
- Legacy (Integer): Se tratará como un booleano (0 y 1), indica si el vehículo en cuestión se sigue usando en carreras oficiales dentro de iRacing o si ha sido sustituido por otro vehículo diferente.
- Comprado (Integer): Se tratará como un booleano (0 y 1), indica si dicho vehículo pertenece a el jugador o no.

Tabla Circuitos: Tabla donde almacenaremos la información de los circuitos donde se correrá en el juego. Está compuesta por los siguientes campos:

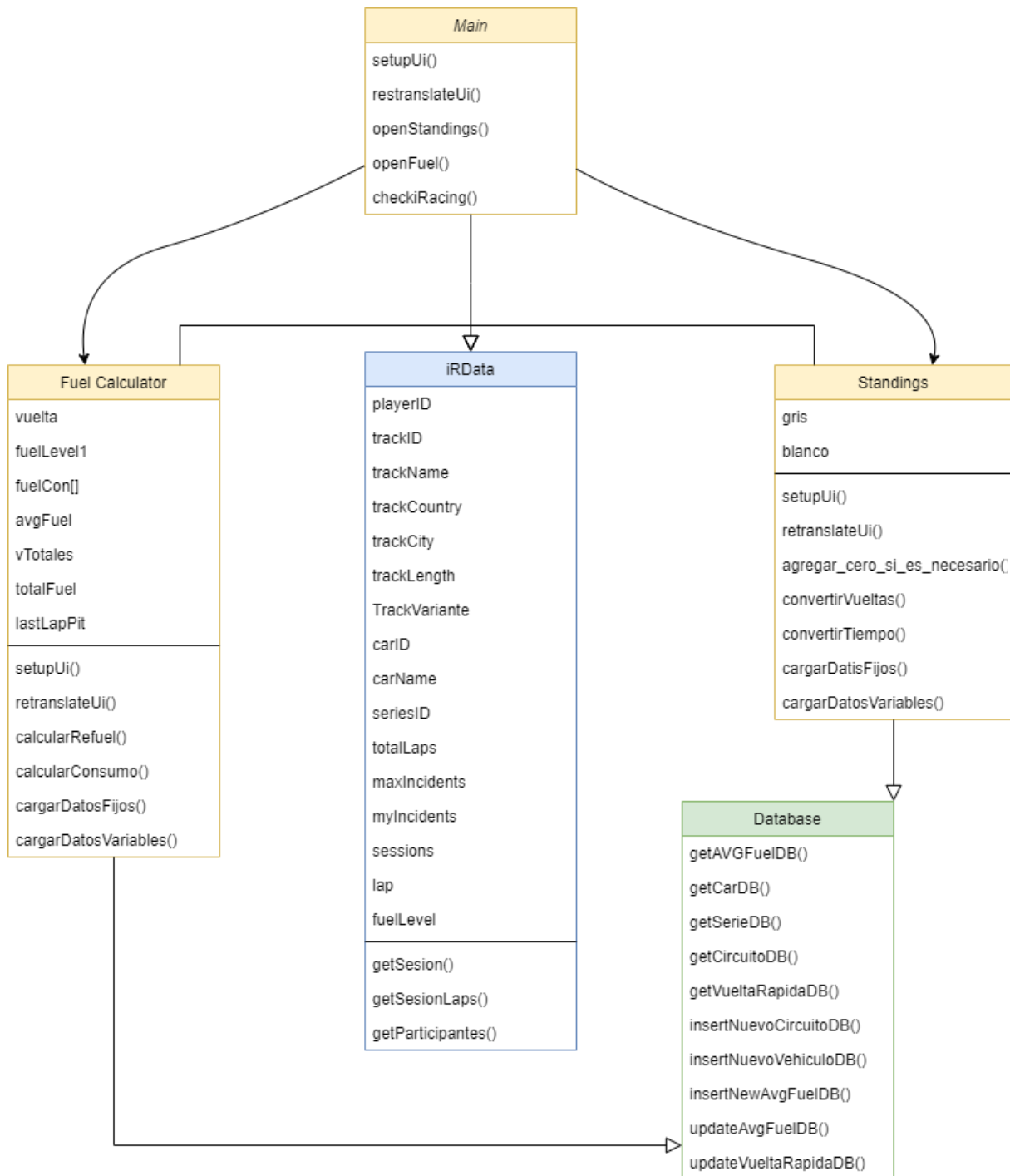
- IDCircuito (Integer): N.º identificador del circuito utilizaremos los mismos id que utiliza iRacing para cada circuito, para así poder buscarlos en la bbdd utilizando los datos que nos proporciona el sdk. Es la clave primaria de la tabla
- Nombre (Text): Nombre por el que se conoce el circuito. Lo utilizaremos para mostrar información al jugador
- País (Text): País donde se encuentra el circuito.
- Ciudad (Text): Ciudad donde se encuentra el circuito.
- Longitud (Text): Longitud en km del circuito
- Variante (Text): Nombre de la variante del circuito, puesto que normalmente los circuitos tienen varios trazados.
- Se tratará como un booleano (0 y 1), indica si el circuito en cuestión se sigue usando en carreras oficiales dentro de iRacing o si ha sido sustituido (en ocasiones reescanean el circuito y lo vuelven a añadir a la tienda como si fuera un nuevo circuito, dejando el antiguo como obsoleto).
- Comprado (Integer): Se tratará como un booleano (0 y 1), indica si dicho circuito pertenece a el jugador o no.

Tabla VueltaRapida: Tabla donde almacenaremos información sobre las vueltas de un vehículo en un circuito concreto. Está compuesta por los siguientes campos:

- IDTiempo (Integer): N.º identificador del tiempo, se crea consecutivamente según se vaya añadiendo datos a la tabla, es la clave primaria de la tabla.
- IDCircuito (Integer): ID del circuito donde se ha marcado el tiempo, es una clave ajena relacionada con el IDCircuito de la tabla Circuitos.
- IDVehiculo (Integer): ID del vehículo con el que se ha marcado el tiempo, es una clave ajena relacionada con el IDVehiculo de la tabla Vehículos.
- Tiempo (Real): El mejor tiempo conseguido en el IDCircuito con el IDVehiculo, cuando termine una sesión de carrera, se comprobará si existen datos en la bbdd, en caso negativo añadirá la nueva información a la bbdd, en caso afirmativo se comparará el mejor tiempo de carrera con el tiempo almacenado en la bbdd y si es menor el de carrera, se actualizará con el nuevo tiempo. Estará almacenado en segundos con hasta 4 decimales.
- MediaConsumo (Real): Será la media de consumo por vuelta en el IDCircuito con el IDVehiculo, cuando termine una sesión de carrera, se comprobará si existen datos en la bbdd, en caso negativo añadirá la nueva información a la bbdd, en caso afirmativo se actualizará los datos.

5. Orientación a objetos:

1. Diagramas de clases. Descripción de clases y atributos.



Main: Clase de la ventana principal de la aplicación, es la primera que ve el usuario al abrir el programa. Permite lanzar las otras dos ventanas de las que se compone la aplicación. Está compuesta por los siguientes *métodos*:

- setupUi(): Método automático de pyQT5, permite la creación de la UI.
- retranslateUi(): Método automático de pyQT5, permite la traducción de la UI.
- openStandings(): Método que permite la creación de la ventana y la llamada a la clase Standings.

- openFuel(): Método que permite la creación de la ventana y la llamada a la clase Fuel.
- checkiRacing(): Método que comprueba la conexión del programa con iRacing, limitando las llamadas al resto de clases y evitando así posibles errores.

Fuel: clase de la ventana FuelCalculator, en esta se mostrará varios datos relacionados con el combustible del vehículo, como el nivel de combustible, la autonomía del vehículo, la cantidad de combustible a repostar... La clase se compone de los siguientes atributos y métodos:

Atributos:

- vuelta (int): Variable donde almacenaremos la vuelta actual del piloto en un momento concreto para realizar diferentes operaciones.
- fuelLevel1 (double): Variable donde almacenaremos la cantidad combustible en litros en un momento concreto para realizar diferentes cálculos.
- fuelCon [double]: Array de doubles donde almacenaremos la cantidad de combustible consumido en cada vuelta para poder realizar el calculo de la media de combustible.
- avgFuel (double): Variable donde se almacenará la media de combustible consumido en carrera, según se vayan avanzando las vueltas, este se irá recalculando y ajustando.
- vTotales (double): Variable donde se almacenará las vueltas totales de carrera, esto será necesario en caso de que la carrera tenga una duración por tiempo y no por nº de vueltas, teniendo el programa que calcular las vueltas aproximadas que durará la carrera.
- totalFuel (double): Variable donde se añadirá el total del combustible usado durante la carrera. Dato que necesitaremos para el calculo de la media de combustible.
- lastLapPit (boolean): Variable que indica si en la vuelta anterior el jugador ha entrado a Boxes, dato que necesitaremos para el calculo de media de combustible.

Métodos:

- setupUi(): Método automático de PyQt5, permite la creación de la UI.
- retranslateUi(): Método automático de PyQt5, permite la traducción de la UI.
- calcularRefuel(): Método dónde se calcula la cantidad de combustible a repostar en caso de necesitarlo.
- calcularConsumo(): Método dónde se calcula la media del gasto de combustible por vuelta.
- cargarDatosFijos(): Método dónde cargamos los datos fijos o que solo necesitamos cargar una vez, como la búsqueda del avgFuel en la bbdd o el calculo del total de vueltas de carrera.
- cargarDatosVariables(): Método dónde cargamos los datos variables de la aplicación, como el nivel de combustible, las llamadas a calcularRefuel() y calcularConsumo()...

Standings: clase de la ventana Standings, en esta se mostrará la clasificación de pilotos de la carrera ordenada por posición, además de información adicional sobre la carrera en la cabecera de la ventana. La clase se compone de los siguientes atributos y métodos:

Atributos:

- gris (QColor): variable de la librería QColor de PyQt5 con el color gris, que utilizaremos para pintar el fondo de la línea del piloto.

- blanco (QColor): variable de la librería QColor de pyQT5 con el color blanco, que utilizaremos para pintar el fondo de la línea del piloto.

Métodos:

- setupUi(): Método automático de pyQT5, permite la creación de la UI.
- retranslateUi(): Método automático de pyQT5, permite la traducción de la UI.
- Agregar_cero_si_es_necesario(): Método que recibe un string de números y le añade un cero delante para formatear el numero en 2 dígitos en caso de ser necesario. Ejemplo: recibe el numero 2.322 y devuelve el 02.322.
- convertirVueltas(): Método que recibe un tiempo de vuelta en segundos y lo convierte al formato MM:SS.MSMS.
- convertirTiempo(): Método que recibe el tiempo de carrera en segundos y lo convierte al formato MM:SS.
- cargarDatosFijos(): Método dónde cargamos los datos fijos o que solo necesitamos cargar una vez, como la búsqueda del nombre vehículo, del nombre de la serie o del nombre del circuito en la bbdd o el cálculo del SOF (Strenght Of Field) de la sesión.
- cargarDatosVariables(): Método dónde cargamos los datos variables de la aplicación, como el tiempo de carrera, la vuelta actual, las posiciones de los pilotos...

iRData: clase donde se encapsula toda la información del sdk (irsdk), donde se realiza la conexión con iRacing. Se compone de los siguientes atributos y métodos:

Atributos:

- ir: Objeto IRSDK que permite la conexión al juego.
- playerId: ID del jugador dentro de la sesión de juego.
- trackID: ID del circuito dentro del juego.
- trackName: Nombre del circuito.
- trackCountry: Nombre del país donde se encuentra el circuito.
- trackCity: Nombre de la ciudad donde está ubicada el circuito
- trackLength: Longitud del circuito
- trackVariante: Nombre de la variante del circuito que se corre en la sesión.
- carID: ID del vehículo dentro del juego.
- carName: Marca y modelo del vehículo del jugador.
- seriesID: ID de la serie(campeonato) dentro del juego:
- totalLaps: n.º de vueltas totales de carrera.
- maxIncidents: n.º máximo de incidentes de carrera por jugador.
- myIncidents: n.º de incidentes del jugador.
- Sessions: Array de sesiones de carrera.
- Lap: Vuelta actual del piloto.
- fuelLevel: nivel de combustible del vehículo del jugador.

Métodos:

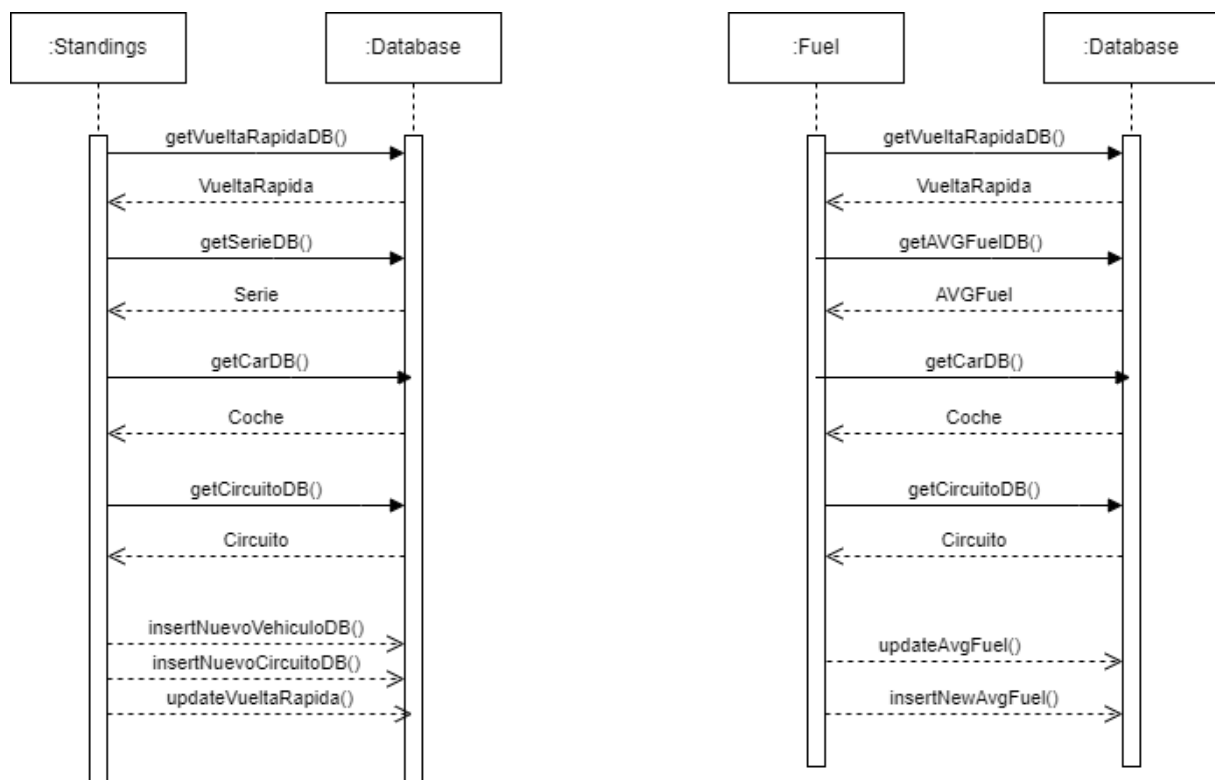
- getSession(): Método que devuelve el identificador de la sesión de juego actual. 2 = carrera, 1 = clasificación, 0 = práctica.
- getSessionLaps(): Método que devuelve el numero de vueltas de la sesión actual.
- getParticipantes() Método que devuelve un array con la información de todos los participantes de la sesión actual.

Database: clase donde se encapsula la conexión a la base de datos. Se compone de los siguientes *métodos*:

- getAVGFuelDB(): Método para obtener la media de consumo de un coche en un circuito de la bbdd.
- getCarDB(): Método para obtener la marca y el modelo de un vehículo de la bbdd.
- getSerieDB(): Método para obtener el nombre de la serie de la bbdd.
- getCircuitoDB(): Método para obtener el nombre del circuito de la bbdd.
- getVueltaRapidaDB(): Método para obtener la vuelta rápida de la bbdd.
- insertNuevoCircuitoDB(): Método para insertar un nuevo circuito en la bbdd.
- insertNewAvgFuelDB(): Método para insertar una nueva media de combustible en la bbdd.
- updateAvgFuelDB(): Método para actualizar la media de consumo en la bbdd.
- updateVueltaRapidaDB(): Método para actualizar la vuelta rápida en la bbdd.

2. Diagrama de secuencias. De lo más relevante.

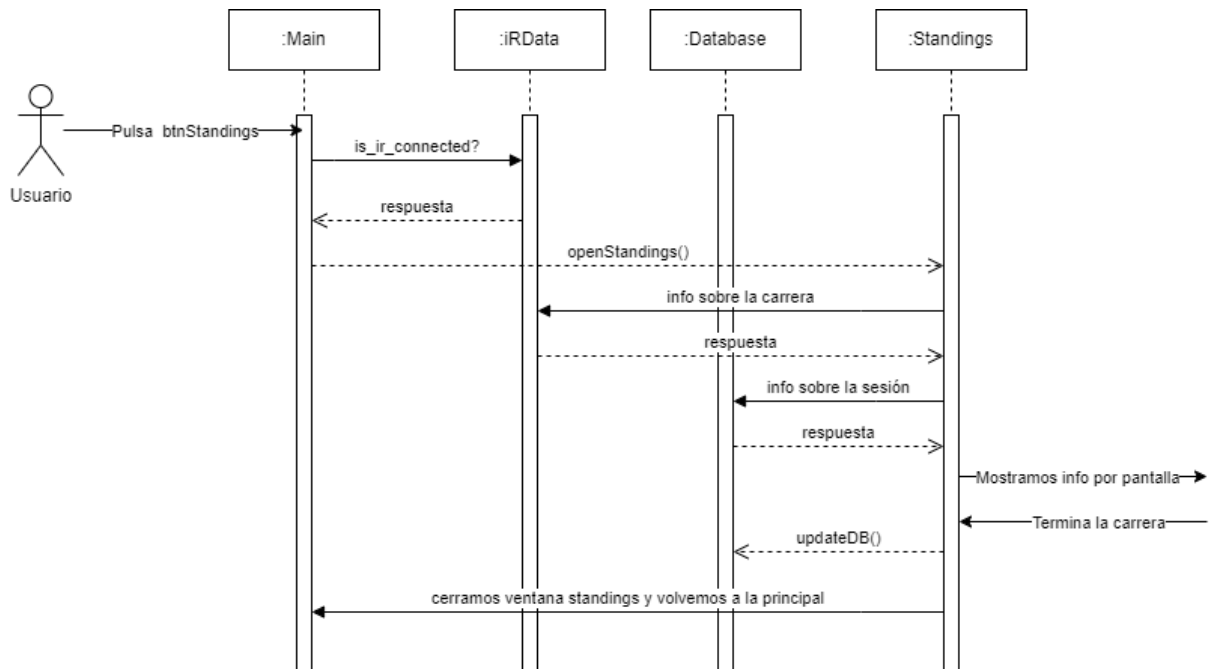
A continuación, vemos los diagramas de secuencia de las interacciones más destacables dentro de la aplicación.



En primer lugar nos encontramos los diagramas de secuencia de las interacciones con la clase `Database()`, a dicha clase únicamente se puede acceder desde las clases `Standings()` y `Fuel()`. Dentro de estas relaciones, solamente existen 2 tipos de comunicación:

- Las síncronas, son las que esperan respuesta por parte del receptor, como `getSerieDB()`, `getCiruitoDB()`, `getVueltaRapidaDB()`...
- Las asíncronas son las que no se espera ninguna respuesta por parte del receptor, como `insertNuevoCircuito()`, `insertNewAvgFuel()`, `updateVueltaRapida()`...

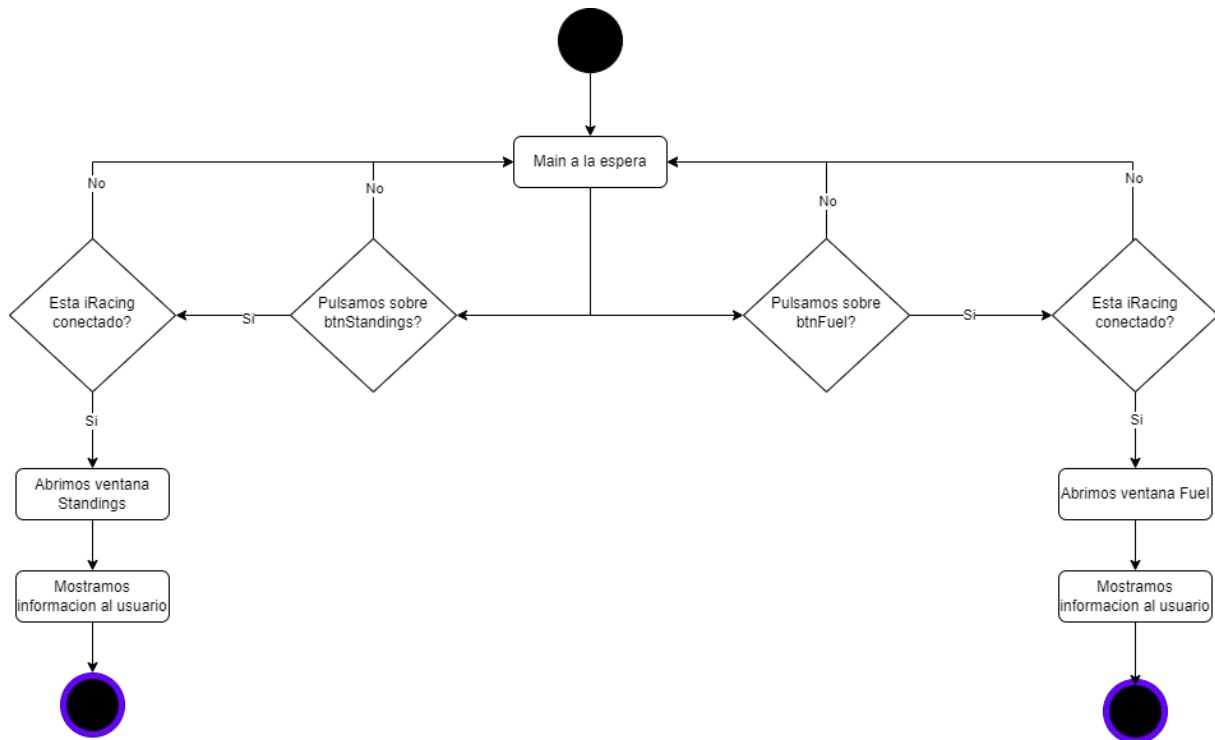
Por otro lado, encontramos el diagrama de secuencia de las interacciones que se realizan entre la clase Main y Standings, pasando por iRData y Database. Este ejemplo es aplicable a la relación entre las clases Main y Fuel.



El btnStandings de la clase Main está en constante comunicación con la clase iRData() gracias a un evento QTimer() que itera el método checkiRacing(), que habilita el botón cuando detecta que iRacing está conectado. Cuando el usuario pulsa el botón, se crea una nueva ventana y se accede a la clase Standings().

Ésta clase se comunica tanto con iRData() como con Database() mediante diferentes métodos con los que consigue información sobre la carrera, para después plasmarla en pantalla y así, transmitirla al usuario. Una vez termina la carrera o el usuario cierra el simulador, la clase Standings realizara una serie de actualizaciones sobre los datos de la bbdd si lo considera necesario, y procede a cerrar la ventana y matar su proceso. Volviendo a la ventana principal.

3. Diagrama de actividad. De lo más relevante.



6. Mockups.

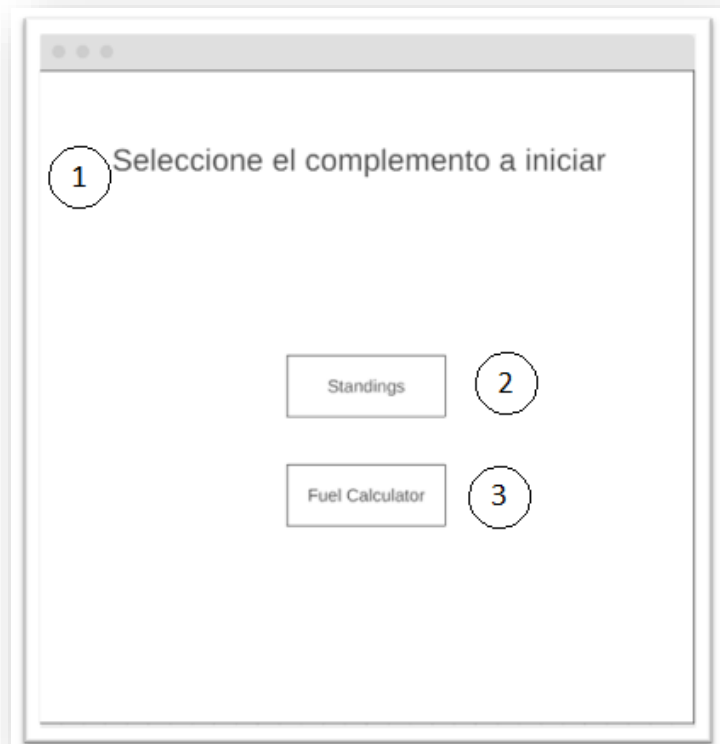
A continuación, se mostrarán los prototipos de diseño UI que han servido como base para la creación de la UI final de la aplicación. Con una explicación de cada uno de los elementos que lo componen.

Ventana Main:

1.- Label explicativo, indica al usuario a pulsar sobre los botones para iniciar la aplicación.

2.- Botón Standings, permite abrir la ventana Standings, si iRacing está conectado estará en color verde, sino en color rojo.

3.- Botón Fuel Calculator, permite abrir la ventana Fuel, si iRacing está conectado estará en color verde, sino en color rojo.



Ventana Standings:

1 Campeonato en el que participamos			2 Coche con el que corremos		
3 Circuito Ricardo Tormo		SOF: 999	Vuelta: 4 / 15	Carrera 06:20 / 18:00	Incidentes 3 / 17
4			5	6	7
1	Jugador 1	A 4.99	9999	01:59.999	01:59.999
2	Jugador 2	A 4.99	9999	01:59.999	01:59.999
3	Jugador 3	A 4.99	9999	01:59.999	01:59.999
4	Jugador 4	A 4.99	9999	01:59.999	01:59.999
5	Jugador 5	A 4.99	9999	01:59.999	01:59.999
6	Jugador 6	A 4.99	9999	01:59.999	01:59.999
7	Jugador 7	A 4.99	9999	01:59.999	01:59.999
8	Jugador 8	A 4.99	9999	01:59.999	01:59.999
9	Jugador 9	A 4.99	9999	01:59.999	01:59.999
10	Jugador 10	A 4.99	9999	01:59.999	01:59.999
11	Jugador 11	A 4.99	9999	01:59.999	01:59.999
12	Jugador 12	A 4.99	9999	01:59.999	01:59.999
13	Jugador 13	A 4.99	9999	01:59.999	01:59.999
14	Jugador 14	A 4.99	9999	01:59.999	01:59.999
15	Jugador 15	A 4.99	9999	01:59.999	01:59.999
16	Jugador 16	A 4.99	9999	01:59.999	01:59.999
8	9	10	11	12	13

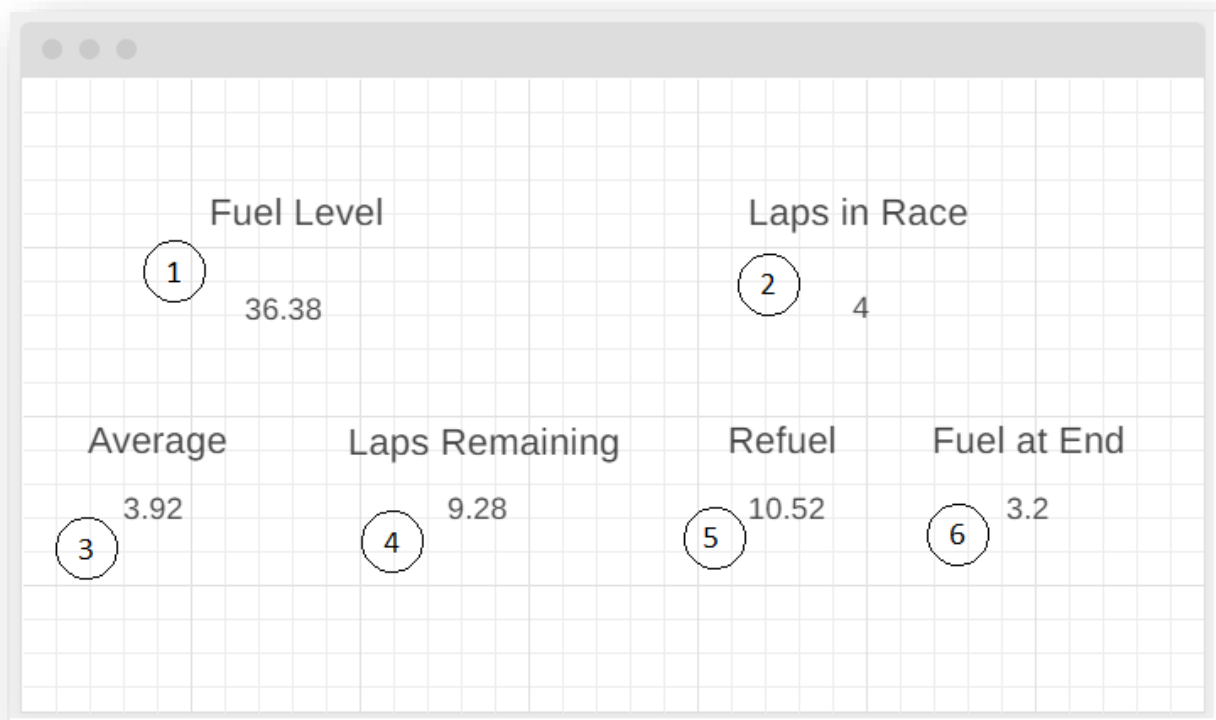
- 1.- lblSerie: donde se indicará la serie (campeonato) donde participa el jugador.
- 2.- lblCar: donde se indicará el vehículo con el que participa el jugador.
- 3.- lblCircuito: donde se indicará el circuito donde corre el jugador.
- 4.- lblSOF: donde se indicará el SOF (Strenght Of Field) de la partida. El SOF es la media de iRating del servidor y únicamente se calcula en las carreras, no en las practicas.
- 5.- lblVueltas: donde se indicará la vuelta actual y el total de vueltas, si es una carrera por limite de tiempo, el programa calculara las vueltas aproximadas, si es una carrera a un numero determinado de vueltas, simplemente indicará cuántas.
- 6.- lblTiempo: donde se indicará el tiempo de carrera, en caso de ser una carrera por limite de tiempo, se mostrará una cuenta atrás con el tiempo restante y el tiempo total, si es una carrera por vueltas, se mostrará una cuenta progresiva del tiempo transcurrido en la carrera y una estimación aproximada del tiempo total de carrera.
- 7.- lblIncidentes: donde se indicará la cantidad de incidentes cometidos por el jugador y el número máximo de incidentes permitidos en el servidor donde se encuentra.

Los siguientes datos se mostrarán en un listado de corredores donde todos tendrán la siguiente estructura:

- Posición del jugador, ya sea clasificación por vuelta rápida en practica y en clasificatoria o por posición de carrera.
- Nombre del jugador, con formato inicial nombre, apellido. Ej. V. Izquierdo
- Licencia + Safety Rating del jugador.

- iRating del jugador.
- Tiempo de la mejor vuelta del jugador, en formato MM:SS.mss
- Tiempo de la última vuelta del jugador, en formato MM:SS.mss

Ventana Fuel Calculator:

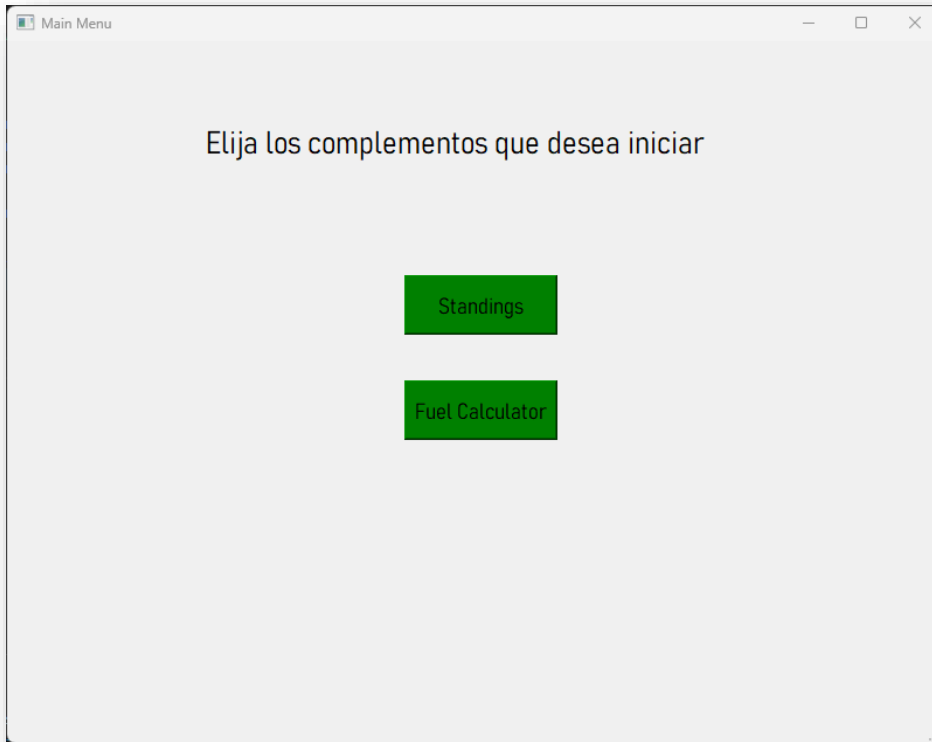


- 1.- lblFuel: donde se indicará el nivel de combustible del vehículo en litros.
- 2.- lblLaps: donde se indicará las vueltas completadas por el jugador.
- 3.- lblAverage: donde se mostrará la media de consumo de combustible por vuelta (l/vuelta). Esta se ira actualizando según vayan pasando las vueltas.
- 4.- lblRemainingLaps: donde se mostrará una estimación de la cantidad de vueltas restantes con el nivel de combustible actual. Éste se irá actualizando según baje el nivel de combustible y se vaya actualizando la media de consumo.
- 5.- lblRefuel: donde se le indicará al jugador la cantidad de combustible a repostar, este se ira adaptando a la conducción del jugador y permitirá llegar al final de carrera con el combustible justo y necesario sin cargar el depósito de más, pero siempre con un margen de seguridad de 1.5 litros extras para evitar que el jugador se quede sin combustible por algún imprevisto.
- 6.- lblFuelAtEnd: donde se indicará la cantidad de combustible sobrante al final de carrera, para ofrecer al jugador la posibilidad de ser más conservador y alargar el repostaje o de arriesgar y recortar unos segundos al tiempo de boxes.

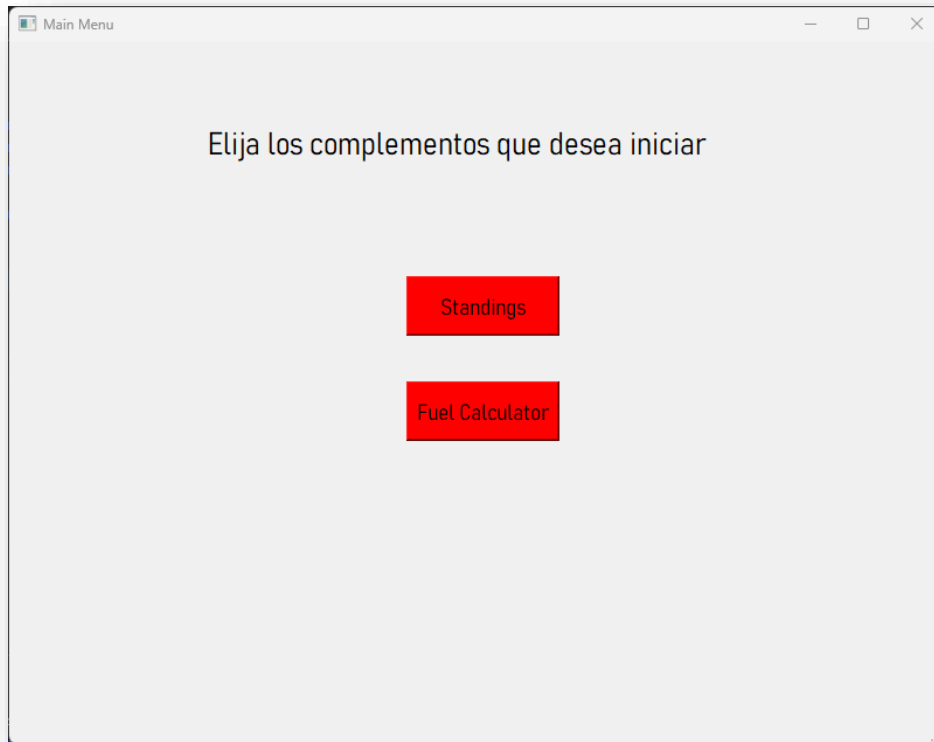
7. UI final.

La interfaz de usuario final se compone de las siguientes ventanas:

Main Menu: muy similar al prototipo, dispone de un label explicativo que nos invita a pulsar en uno de sus botones. Si iRacing está conectado, los botones aparecen en verde, como se ve en la siguiente imagen.



Si iRacing no esta conectado, visualizaremos los botones de color rojo y estos no tendrán ninguna funcionalidad, en cuanto este disponible iRacing de nuevo, cambiarán a color verde.



- Standings

En esta primera imagen, podemos observar los siguientes datos:

- Se trata de una carrera del campeonato Ferrari GT3 Challenge,
- EL jugador utiliza el Ferrari 488 GT3.
- La carrera transcurre en el circuito Oulton Park
- Hay un SOF de 971
- Es una carrera a 15 minutos, por lo que se muestra la vuelta actual y una estimación de las vueltas totales, indicado con el símbolo ~.
- El temporizador muestra una cuenta atrás con el tiempo restante y el tiempo total de carrera.
- Para terminar con la cabecera, vemos que llevamos 3 incidentes de 17.
- Como se observa hay algunos pilotos que no tienen nada en el apartado vuelta rápida, esto se debe a que iRacing no cuenta el tiempo de vuelta si el piloto a cometido algún incidente durante dicha vuelta, si que guarda el tiempo en ultima vuelta, pero no lo guarda en vuelta rápida.

Ferrari GT3 Challenge - Fixed				Ferrari 488 GT3 Evo 2020	
Oulton Park Circuit		SOF: 971	Lap: 4/10.04~	15:00 / 30:00	X 3/17
1	Florido, A	C 3.44	969	01:35.796	01:35.796
2	Park, M	C 4.94	974	01:35.904	01:35.904
3	Farias, G	C 2.64	1083	01:36.006	01:36.006
4	Souza, B	D 3.15	949	01:36.251	01:36.251
5	Blafard, C	D 3.67	934	01:36.327	01:36.327
6	Cooper11, M	D 3.43	839	01:37.242	01:37.446
7	Dülgerbaki, B	D 2.60	1102		01:37.640
8	Pot, A	D 3.46	1033	01:36.165	01:36.165
9	Al-Hasna2, M	D 2.55	998	01:37.662	01:37.662
10	Smith8, A	C 1.26	844		01:38.163
11	Bauer3, T	C 3.45	1067	01:37.652	01:37.652
12	Colleselli, A	D 3.54	1025	01:39.013	01:42.149
13	Izquierdo, V	B 2.05	997	01:41.932	01:41.932
14	Przystawko, K	D 2.42	848	01:41.354	02:07.211
15	Amnitzboell, M	D 2.20	945		
16	Coppola, C	C 2.63	998		

En la siguiente imagen podemos ver los datos de una carrera diferente, en este caso contra la IA, podemos observar lo siguiente:

- Como es una carrera contra bots, no hay campeonato, así que se indica con AI Racing, que se traduciría por carrera contra la Inteligencia Artificial.
- Corremos con el Audi RS3 en el circuito de Daytona.
- Al ser contra bots, no hay SOF, por lo que se deja el hueco vacío.
- En esta ocasión, es una carrera a 15 vueltas, por lo que se indica con seguridad que llevamos 1 de 15 vueltas, y no como estimación.
- Al ser una carrera a vueltas, mostramos un contador con el tiempo transcurrido desde el inicio de la sesión y una estimación del tiempo de carrera total aproximado, indicado con el símbolo ~.
- La primera vuelta no cuenta como vuelta rápida porque al salir desde parado no se contabiliza como una vuelta válida, por eso tenemos datos en última vuelta pero no en vuelta rápida.

AI Racing				Audi RS3 LMS
Daytona International Speedway				06:08 / 28:58~ X 0
Lap: 1/15				
1	David LoVecchio	R 0.00	0	02:09.661
2	Sean Ambrose	R 0.00	0	02:09.722
3	Greg Hill	R 0.00	0	02:11.501
4	Brent Foster	R 0.00	0	02:11.663
5	David Carrillo	R 0.00	0	02:11.879
6	Scottie Nash	R 0.00	0	02:11.933
7	Nim Cross	R 0.00	0	02:15.067
8	Cory Vervynckt	R 0.00	0	02:15.134
9	Jennifer Young	R 0.00	0	02:15.247
10	Drew Adamson	R 0.00	0	02:15.520
11	Steve Myers	R 0.00	0	02:16.187
12	Aussie Greg Hill	R 0.00	0	02:16.196
13	Chris Weidner	R 0.00	0	02:16.328
14	Nick Leep	R 0.00	0	02:16.335
15	Jose Diaz Castillo	R 0.00	0	02:16.455
16	Thomas Leyva	R 0.00	0	02:17.591
17	Robert Plumley	R 0.00	0	02:18.706
18	Greg West	R 0.00	0	02:18.868
19	Chris Leone	R 0.00	0	02:19.175
20	Vicente Izquierdo	R 0.01	1	

- Fuel Calculator

Fuel Level		Laps in Race	
14.94		1	
Average	Laps Remaining	Refuel	Fuel at End
1.52	9.83	--:--	--:--

En este caso nos encontramos en una práctica, por lo que nos muestra el nivel de combustible, las vueltas completadas, la media de consumo y las vueltas restantes con el combustible actual.

15. Codificación.

1. Tecnologías elegidas y su justificación (lenguajes, frameworks, librerías, etc.)

Para el desarrollo de esta aplicación se va a utilizar las siguientes tecnologías:

Lenguaje de programación:

- Python 3.11.2: El principal motivo para utilizar este lenguaje de programación es poder utilizar la librería irsdk, que será el elemento clave de nuestra aplicación, ya que gracias a ella podremos conectarnos a iRacing y obtener la información necesaria para plasmarla al usuario.

Librerías:

- PyQt5 5.15.4: Es una librería de Python que proporciona una interfaz para el desarrollo de aplicaciones de escritorio. Con ella trae la aplicación QT Designer que permite la creación de interfaces de usuario de manera gráfica, este Designer además genera un archivo .ui que mediante un comando permite la generación automática del archivo .py donde ira la lógica.
- irsdk: Como hemos comentado anteriormente, con esta librería seremos capaces de conectarnos a iRacing y obtener información de la sesión de juego y telemetría en directo, permitiendo el desarrollo de la aplicación. Se puede encontrar en el siguiente enlace: <https://github.com/kutu/pyirsdsk>

IDE

- Visual Studio Code 1.78.2

1. Desarrollo de servicios.

1. Descripción general.
2. Seguridad.

2. Desarrollo multiplataforma.

1. Descripción general.
2. Asegurar la funcionalidad en distintos dispositivos.

2. Documentación interna de código (puede ir en un anexo).

1. Descripción de cada fichero. Autor, función y fecha de creación.
2. Descripción de cada función. Autor, función y fecha de creación.

3. Documentación externa (puede ir en un anexo).

4. Manual del usuario.

16. Despliegue

1. Diagramas de despliegue
2. Descripción de la instalación o despliegue

17. Herramientas de apoyo

18. Control de versiones.

19. Sistemas de integración continua.

20. Gestión de pruebas

21. Conclusiones.

1. Conclusiones sobre el trabajo realizado
2. Conclusiones personales
3. Posibles ampliaciones y mejoras

22. Bibliografía (comentada)

23. Libros, artículos y apuntes

24. Direcciones web