24 de mayo de 2023

Vicente Izquierdo Forment

[Nombre de la empresa]

[Dirección de la compañía]

Desarrollo de un complemento de software para el videojuego iRACING

[Subtítulo del documento]

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

Introducción

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 esta 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, asi 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.Estudio de viabilidad. Método DAFO.

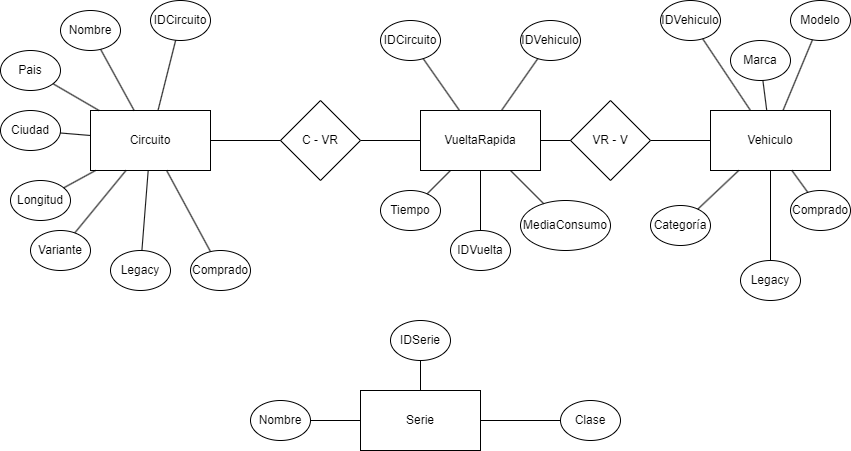
* 1. Estudio de mercado.
     1. Viabilidad técnica/económica del proyecto
     2. Recursos HW
     3. Recursos SW
     4. Recursos humanos
     5. Viabilidad temporal
  2. Planificación temporal o agenda de trabajo.

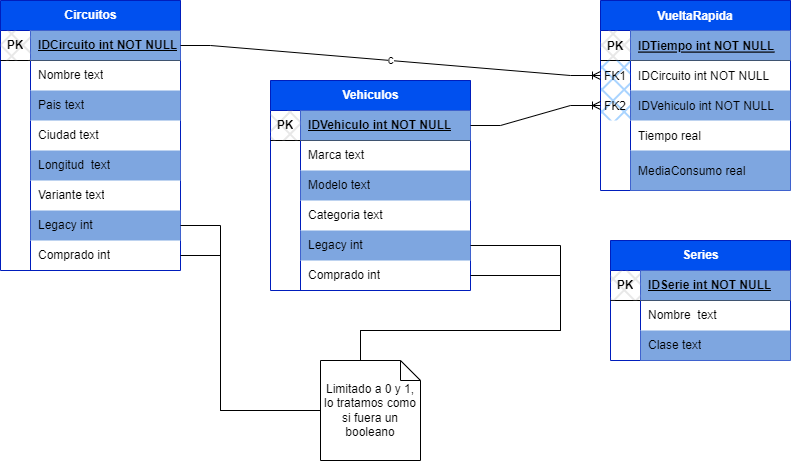
3.Análisis de requisitos

* 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)

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. **FOREIGN** **KEY**(IDVehiculo) **REFERENCES** Vehiculos(IDVehiculo)
26. )
27. **INSERT** **INTO**  Circuitos **VALUES**
28. (95, 'Sebring International Raceway', 'USA', 'Sebring, Florida', '5.95 km', 'International', 0, 1),
29. (96, 'Sebring International Raceway', 'USA', 'Sebring, Florida', '3.17 km', 'Modified', 0, 1),
30. (97, 'Sebring International Raceway', 'USA', 'Sebring, Florida', '2.81 km', 'Club', 0, 1),
31. (47, 'Weathertech Raceway at Laguna Seca', 'USA', 'Monterey, California', '3.60 km', 'Full Course', 0, 1),
32. (432, 'Watkins Glen International', 'USA', 'Watkins Glen, New York', '3.89 km', 'Classic', 0, 1),
34. **INSERT** **INTO** Vehiculos **VALUES**
35. (1, 'Skip Barber Formula 2000', 'Skip Barber', 0, 1),
36. (67, 'Mazda MX-5 Cup', 'MX-5 Cup', 0, 1),
37. (112, 'Audi RS3 LMS', 'TCR', 0, 1),
38. (119, 'Porsche 718 Cayman GT4 Clubsport MR', 'GT4', 0, 1),
39. (144, 'Ferrari 488 GT3 Evo 2020', 'GT3', 0, 1),
40. (148, 'iRacing iR-04', 'F4', 0, 1),
41. (160, 'Toyota GR86', 'GR86', 0, 1)
43. **INSERT** **INTO** VueltaRapida **VALUES**
44. (1, 47, 144, 86.375, 2.19),
45. (2, 95, 144, 126.242, 3.39),
46. (3, 403, 148, 95.500, 0.95),
47. (4, 445, 144, 103.663, 2.51)
    1. 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:

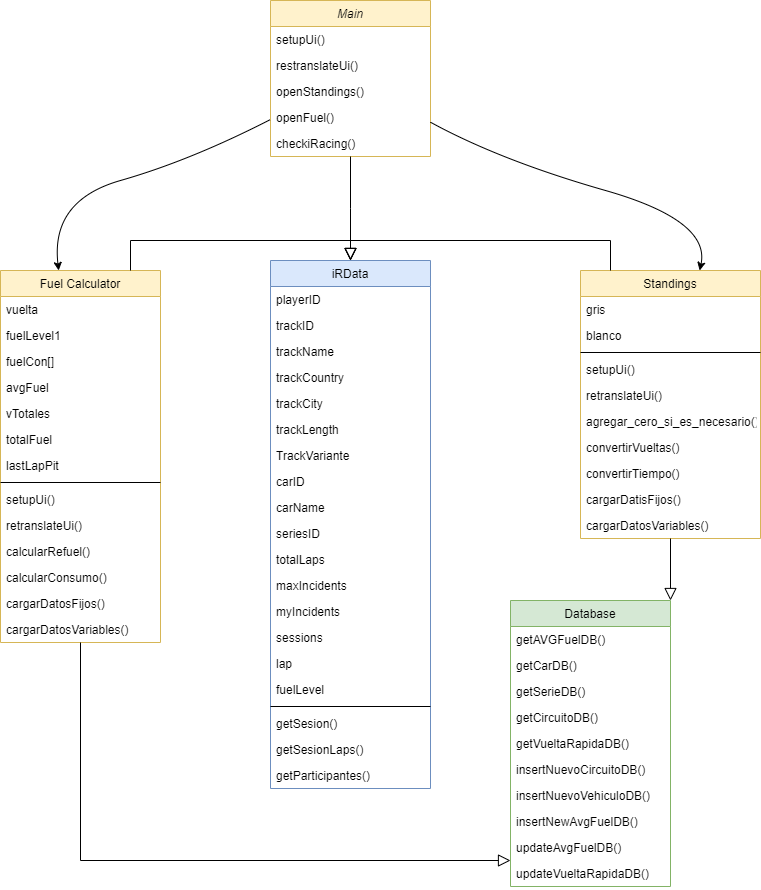
* IDVehiculo (Integer): N.º identificador del vehículo, utilizaremos los mismos id que utiliza iRacing para cada vehículo, para así poder buscarlos en la bbdd utilizando los datos que nos proporciona el sdk sin tener que hacer búsquedas por marca o modelo dentro de la bbdd. Es la clave primaria de la 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
* Pais (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.
  1. 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.

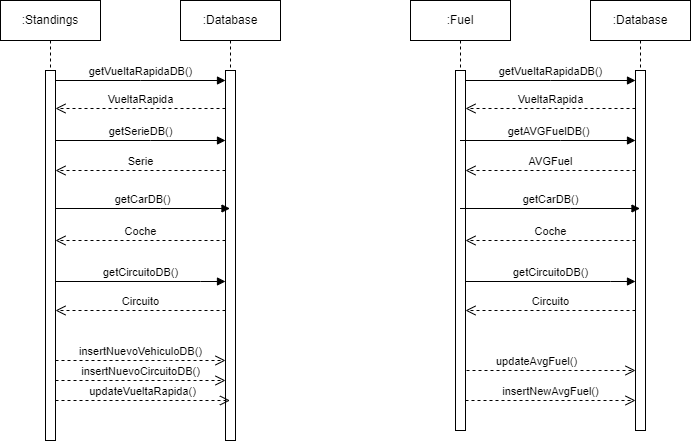
*Métodos:*

* getSesion(): Método que devuelve el identificador de la sesión de juego actual. 2 = carrera, 1 = clasificación, 0 = práctica.
* getSesionLaps(): 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.
  + 1. Diagrama de secuencias. De lo más relevante.

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

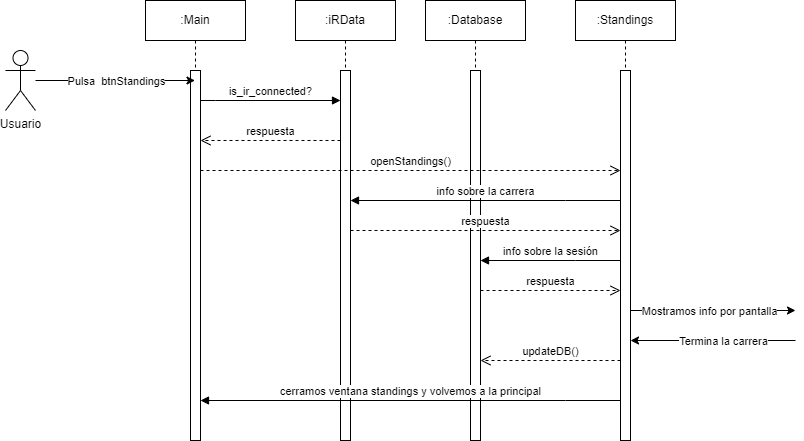


En primer lugar nos encontramos los diagramas de secuencia de las interacciones con la clase Database(), a dicha clase únicamente se puede acceder desde las clases Standings() y Fuel().

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()…

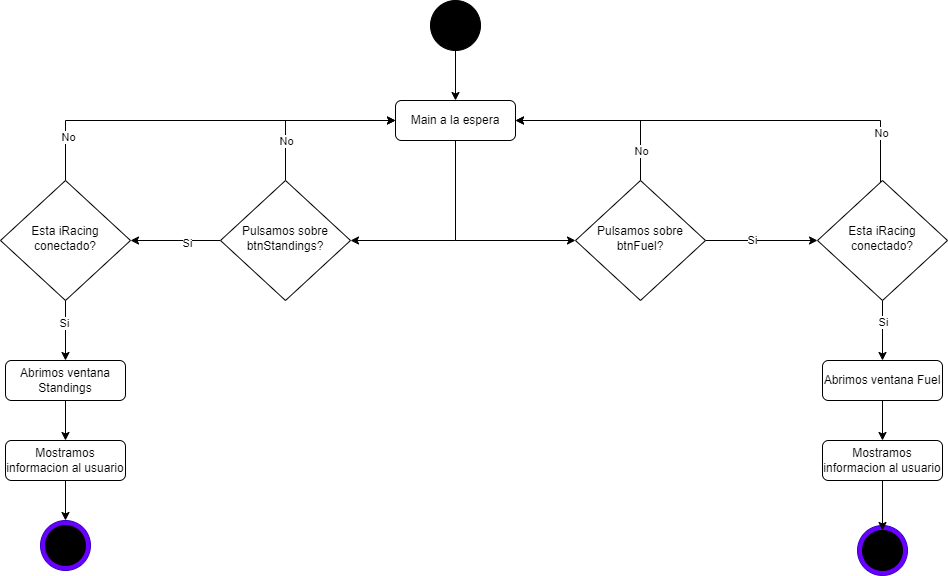
Por otro lado, encontramos el diagrama de secuencia de las interacciones que se realizan entre la clase Main y Standings, pasando por iRData y Database. Este ejemplo es aplicable a la relación entre las clases Main y Fuel.



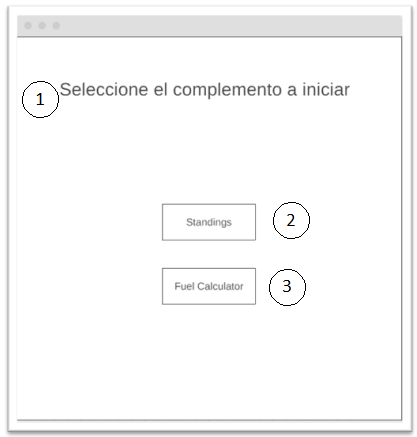
El btnStandings de la clase Main está en constante comunicación con la clase iRData() gracias a un evento QTimer() que itera el método checkiRacing(), que habilita el botón cuando detecta que iRacing está conectado. Cuando el usuario pulsa el botón, se crea una nueva ventana y se accede a la clase Standings().

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.

* + 1. Diagrama de actividad. De lo más relevante.



* 1. 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.

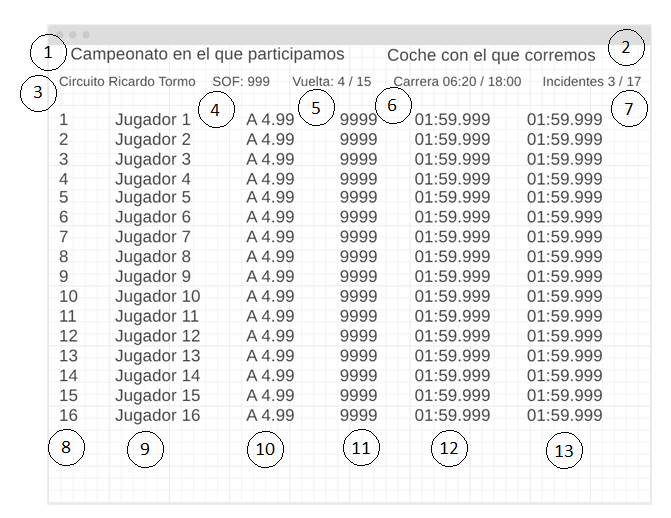
**Ventana Main:**

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.- 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.

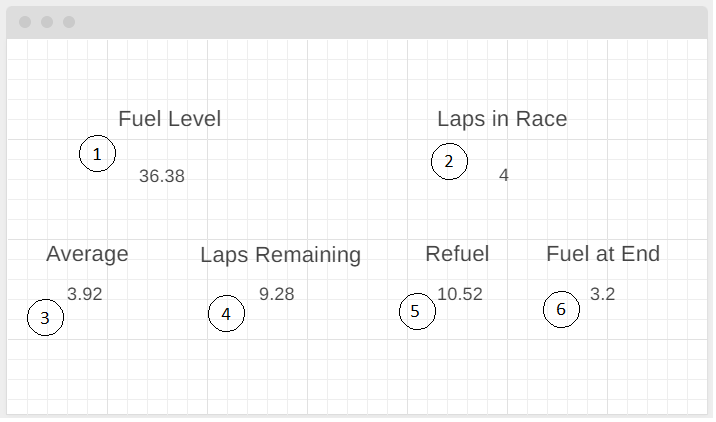
6.- lblTiempo: donde se indicará el tiempo de carrera, en caso de ser una carrera por limite de tiempo, se mostrará una cuenta atrás con el tiempo restante y el tiempo total, si es una carrera por vueltas, se mostrará una cuenta progresiva del tiempo transcurrido en la carrera y una estimación aproximada 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.

* 1. UI final.

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

1. 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.
2. Despliegue
   1. Diagramas de despliegue
   2. Descripción de la instalación o despliegue
3. Herramientas de apoyo
4. Control de versiones.
5. Sistemas de integración continua.
6. Gestión de pruebas
7. Conclusiones.
   1. Conclusiones sobre el trabajo realizado
   2. Conclusiones personales
   3. Posibles ampliaciones y mejoras
8. Bibliografía (comentada)
9. Libros, artículos y apuntes
10. Direcciones web