Sistema de gamificación del deporte.

INDICE

Enlace al repositorio GitHub del proyecto.

Introducción. Breve descripción de la aplicación a desarrollar indicando su finalidad y objetivos, requisitos y restricciones.

Es un sistema distribuido donde a través de un juego que detecta la realización de un deporte se obtienen recompensas que son usadas en dicho juego, además se crea un ranking con los usuarios que han obtenido más puntos.

La interacción con el usuario se lleva a cabo a través de una aplicación móvil, basada en un juego formato clicker, el usuario obtiene puntos (croquetas) y los puede canjear por objetos del juego.
Con un detector de movimiento, en este caso, el movimiento de la rueda de una bicicleta también puede obtener dichos puntos y además poder visualizarse en un ranking con otros jugadores, asi mismo fomentando la realización de deporte e incentivando el mismo.

El sistema puede ser modificado según el ejercicio que se quiera realizar, ya que la aplicación funciona por si sola y puede a adaptarse a cualquier otro sistema de detención del ejercicio.

La finalidad y objetivo del sistema es el fomento del deporte a través de la gamificación ya que la idea del proyecto surgió desde la necesidad de un usuario de motivar su rutina de bicicleta estática diaria.

Los requisitos / restricciones del sistema son:

- Un teléfono móvil Android.
- Conexión a internet.
- Conexión bluetooth.
- Una versión de Android superior a Android 5 (Lollipop)

Análisis del sistema actual. Qué aplicaciones similares existen actualmente en el mercado, qué problemas/inconvenientes presentan. Qué aporta la herramienta desarrollada por el alumno, qué problemas resuelve.

La Gamificación es una técnica de aprendizaje muy utilizada actualmente en muchos ámbitos cotidianos, especialmente enfocados en el ámbito educativo usando la mecánica de los juegos para conseguir mejores resultados, conocimientos, mejorar habilidades o recompensar acciones como en este caso la acción de la realización del deporte de manera periódica, es decir la creación de hábitos.

En concreto la gamificación del deporte es una tendencia que combina elementos de juegos y deportes para mejorar la experiencia de los participantes y aumentar su motivación. Se trata de aplicar principios y mecánicas de juego en contextos deportivos con el objetivo de hacerlos más atractivos, divertidos y participativos.

Hay aplicaciones en el mercado que detectan el ejercicio de varias formas:

- Detectan el ejercicio usando sensores integrados en el móvil, como por ejemplo usando el podómetro detectan los pasos y se registran en aplicaciones como Google Fit, una de las más reconocidas, aunque hay varias en el mercado.
- Detectan el ejercicio usando los sensores de relojes inteligentes, como por ejemplo Samsung Wear, Apple Watch …

Hay aplicaciones como Samsung Health que combina la información obtenida de los sensores del móvil y de la información obtenida del Samsung Wear.

Hay aplicaciones que incorporan elementos de gamificación como, por ejemplo:

- Nike Training Club:
  Esta aplicación ofrece entrenamientos personalizados con diversos niveles de dificultad.
  Incorpora elementos de gamificación, como recompensas por completar entrenamientos, desafíos y la capacidad de competir con amigos.

- Zombies, Run!:
  Diseñado para corredores, este juego de realidad aumentada presenta una historia apocalíptica de zombis.
  Los jugadores deben correr en el mundo real para escapar de los zombis y cumplir misiones, lo que añade un componente de narrativa y desafío al ejercicio.

- Pokémon GO:
  Aunque no está directamente relacionado con un deporte específico, Pokémon GO fomenta la actividad física al requerir que los jugadores se muevan en el mundo real para capturar Pokémon y participar en eventos especiales.

Para que una buena aplicación pueda incluir este sistema hay algunos aspectos clave de la gamificación a tener en cuenta:

- Puntos y Recompensas:
  Se pueden otorgar puntos por lograr objetivos específicos, como completar un entrenamiento, superar una marca personal o participar en actividades adicionales.
  Las recompensas pueden incluir medallas virtuales, insignias o incluso premios tangibles, como descuentos en equipos deportivos.

  En mi caso, la aplicación al detectar el deporte generará croquetas que tendrán un valor mucho más superior que generados por otros productores y además tendrá logros y estadísticas conforme a tu actividad, cantidad de croquetas generadas, días conectados consecutivos...

- Competencia y Desafíos:
  La gamificación fomenta la competencia amistosa mediante desafíos entre individuos o equipos.
  Los participantes pueden establecer metas personales o competir contra otros para alcanzar ciertos logros.

  El sistema contara con una pagina web para ver el ranking de los jugadores por croquetas generadas y el ranking de jugadores por croquetas obtenidas por la detención de un ejercicio.

- Aplicaciones y Plataformas Digitales:
  Muchas aplicaciones y plataformas digitales utilizan la gamificación para mejorar la experiencia de los usuarios en actividades deportivas.
  Estos sistemas pueden rastrear el progreso, establecer desafíos y proporcionar retroalimentación en tiempo real.

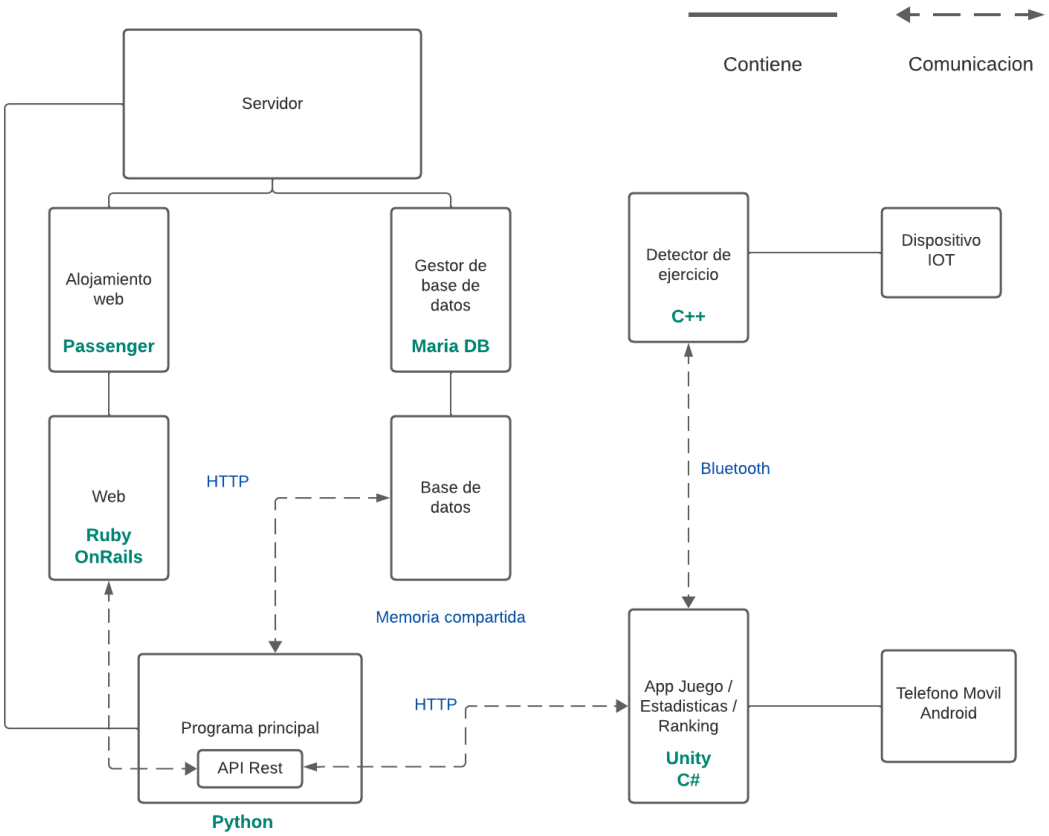
  La plataforma principal será el móvil donde instalaremos el juego, aunque para acceder al ranking podremos hacerlo tanto desde el juego móvil como desde una página web, y el detector del movimiento es un dispositivo IOT externo.

- Elementos de Juego:
  Se pueden incorporar elementos de juego como tableros de líderes, niveles y logros para hacer que la experiencia sea más atractiva.
  Estos elementos ofrecen una estructura clara y objetivos alcanzables, lo que motiva a los participantes a seguir adelante.

- Entrenamiento Divertido:
  La gamificación se utiliza para hacer que el entrenamiento sea más divertido y menos monótono.
  Los juegos y desafíos pueden convertir el ejercicio en una experiencia entretenida y atractiva.

En resumen, la aplicación contiene los aspectos básicos para una buena gamificación, además comparte muchos aspectos con otros juegos que la implementan, aunque es cierto que la aplicación es completamente funcional cuando no detecta el ejercicio, aunque la generación de croquetas es mucho más lenta.

Solución propuesta. Diferentes tecnologías existentes para el desarrollo y puesta en funcionamiento de la aplicación que se desarrollará. Elección y justificación de las tecnologías seleccionadas (plataforma de desarrollo, SGBD, servidor, etc.).



Mi sistema constará de:

- Un servidor, en este caso yo tengo uno propio en mi casa, pero actualmente es fácil obtener un servidor, en este caso Amazon Web Services, Microsoft Azuer, etc.
  Mi servidor, es un ordenador antiguo, con un procesador i3-3100 y 8GB DDR3, tiene un sistema operativo Debian 12 con interfaz de consola.

- Un gestor de base de datos que te brinda el propio servidor o lo escoges según la tecnología que utilices.
  En este caso la idea principal era que el gestor de base de datos fuera un MySQL pero debido a los problemas con el sistema operativo del servidor y versiones, decidí usar mariaDB ya que era una opción que soporte base de datos tipo sql y compatible con mi servidor.

- Una base de datos para guardar información de los usuarios, guardar los datos del juego según el usuario, etc.

Una base de datos tipo SQL, ya que necesitamos una base de datos relacional por el modelo de datos que vamos a implementar, al trabajar con diferentes plataformas y lenguajes el tipo SQL esta más extendido y tiene más librerías y facilita la integración con las plataformas.

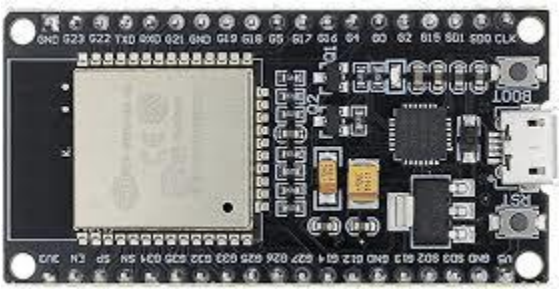
- Un alojamiento web que te brinda el propio servidor.
  La primera idea del proyecto era usar un alojamiento web llamado Passenger ya que es el más optimo para alojar paginas web generadas por RubyOnRails y por compatibilidad con el sistema operativo del servidor.
  Aunque también se consideró una solución menos desarrollada, donde el alojamiento web fuera con Apache por compatibilidad con el sistema operativo y por facilidad de configuración del mismo.

- Una página web para poder visualizar el ranking de los jugadores sin necesidad de loggearte en la aplicación.
  Como hemos mencionado anteriormente, se usaría Ruby como lenguaje de programación y Rails como framework para desarrollar la página web, ya que es una tecnología muy usada para dicha finalidad.
  Pero también se planteó como hemos comentado anteriormente una solución menos desarrollada donde si el alojamiento web es Apache, usar el lenguaje de marcas HTML y JavaScript para desarrollar la web.

- Un programa principal para controlar el funcionamiento general del sistema, el acceso a la base de datos y contener la API Rest, que se encargará de la comunicación de la información requerida por los demás componentes del sistema.
  El programa principal esta desarrollado en Python, ya que es una tecnología que por su potencia puede generar API's de una manera muy sencilla y con una cantidad de código muy reducida. También la manera de acceder a la base de datos con Python es también muy potente y sencilla, aunque su principal punto a favor es la facilidad para trabajar con JSON.
  Además, es Python tiene una cantidad de librerías propias que facilitarán en gran medida el trabajo, a diferencia de otras tecnologías investigadas como, por ejemplo, Java.

- Una aplicación móvil que es un juego basado en un clicker.
  El juego se desarrollará con el motor de videojuegos llamado Unity, con el lenguaje de programación C#. Ya que te ofrece la potencia de desarrollo de videojuegos junto con la posibilidad de generarlos para ordenador o movil. Por lo que para nuestro proyecto es necesario esa herramienta multiplataforma.

- Un detector de ejercicio, el cual integrando un sensor detectará el ejercicio.
  El detector consta de un ESP-32 y el sensor de un cuentakilómetros de una bicicleta.
  He decido elegir un microprocesador por su pequeño tamaño, precio y componentes integrados como Wi-Fi y Bluetooth, es programable en lenguajes como C++, y compatible con herramientas de desarrollo integrado, en este caso he usado PlatformIO que es una extensión a Visual Studio Code.



Planificación temporal del desarrollo del proyecto. Sobre los siguientes puntos (si procede): Diseño de la solución a adoptar. Adquisición del software y hardware. Desarrollo de la aplicación. Puesta a punto de la aplicación. Fase de pruebas. Lista de modificaciones o ampliaciones futuras que se podrían aplicar a la aplicación proyectada. Plan de mantenimiento de la aplicación.

Al iniciar el proyecto, parto de un prototipo de juego que realicé anteriormente, el cual he reciclado algunas cosas, arreglado bug's e implementado algunas funciones nuevas. Por lo que me dió información de los datos que quería guardar, necesitar…

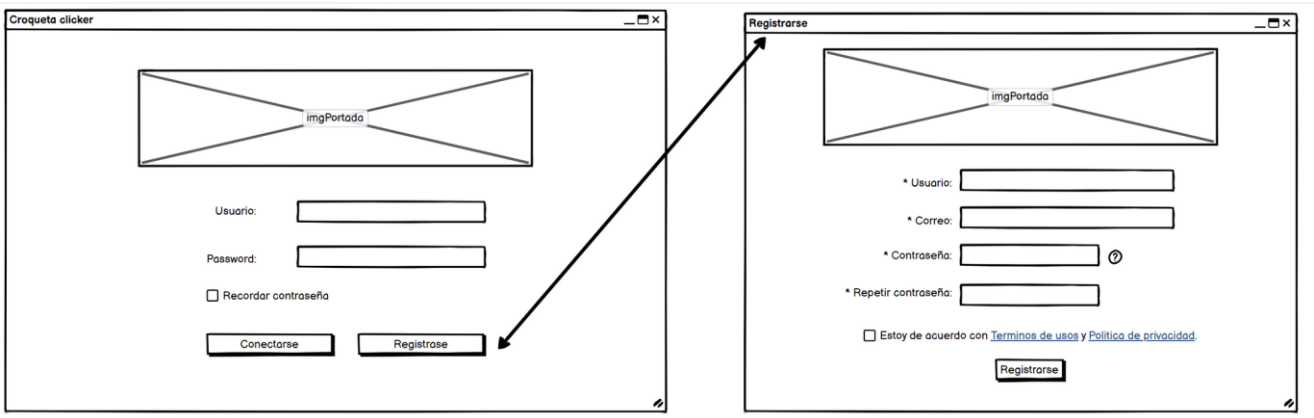
1. Configurar el servidor, instalando Debian 12 e instalando el gestor de base de datos MariaDB.
2. Diseñar la base de datos e implementarla.
3. Configurar Github para ir subiendo el proyecto, tanto en el servidor como en el ordenador.
4. Organizar la estructura del código del Programa principal necesario para obtener peticiones y acceder a la base de datos.
5. Programar la API con Python y la conexión de Python con MariaDB.
6. Arreglar problemas de la aplicación móvil.
7. Agregar efecto clics en la croqueta de la aplicación móvil.
8. Conectar API con la aplicación móvil.
9. Descarga y configuración de Apache en el servidor
10. Programación de la web y conexión con la API.
11. Configuración del rúter y alojamiento web para acceder desde fuera de la red local.
12. Programación del ESP-32.
13. Añadir la electrónica del cuentakilómetros con el ESP-32.
14. Conectar la aplicación móvil con el ESP-32.
15. Instalación de la aplicación en el móvil.
16. Fase de pruebas.

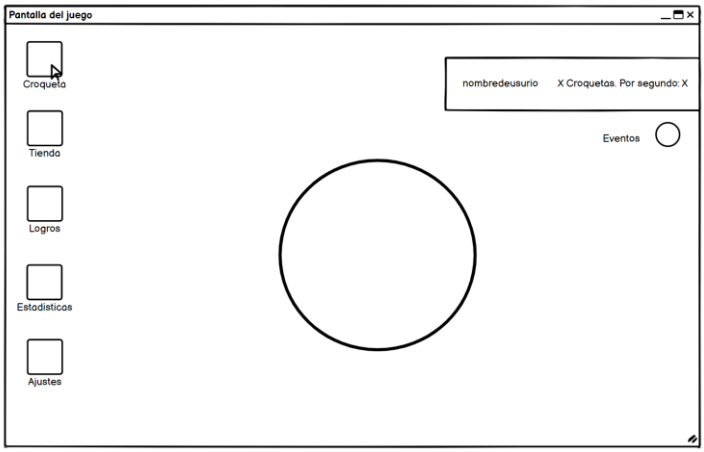
Según el tiempo disponible para la presentación del proyecto se han limitado funciones del sistema, con el tiempo suficiente se mejorará:

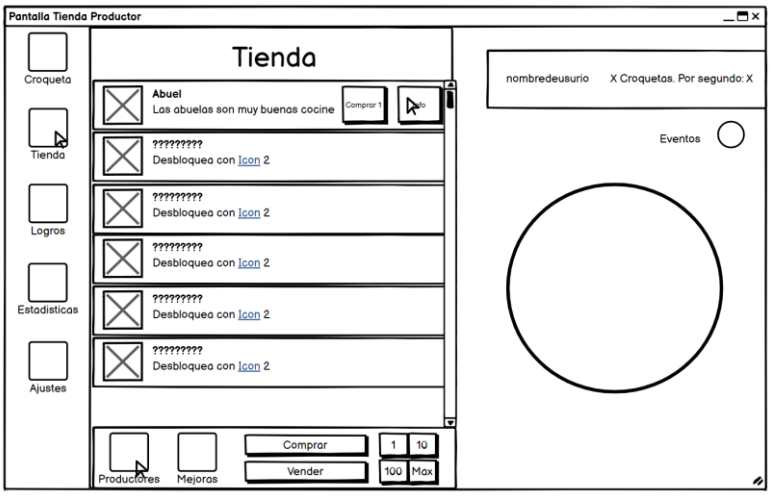
1. Seguridad en contraseñas.
2. Añadir tokens para la API.
3. Migrar la página web a RubyonRails, mejorar la apariencia y muestra de ranking por número de croquetas generadas y otro por número de croquetas generadas por ejercicio.
4. Continuar con la certificación SSL, el cual es un certificado de clave simétrica para obtener HTTPS en la API.
5. Añadir mejoras en la interfaz del juego móvil.
6. Añadir estadísticas en la aplicación, logros y bonus temporales.
7. Poder acceder desde el juego móvil a la web para mostrar el ranking.
8. Añadir información de los productores.

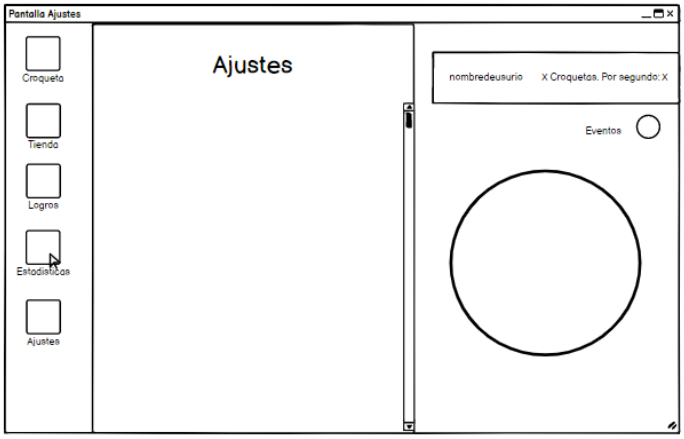
Documentación del diseño e implementación. Incluir, entre otros, los siguientes puntos: Diseño de interfaces (Prototipo de la aplicación). Diseño lógico de la aplicación a desarrollar, diagramación UML o de Flujo de datos. Descripción modular del software a desarrollar, si procede. Diseño de la/s bases de datos, con sus diagramas correspondientes. Otras estructuras de datos que se utilizan en la aplicación. Código fuente documentado (alguna parte importante o interesante de comentar). Otros aspectos.

Diseño de interfaces (Prototipo de la aplicación):



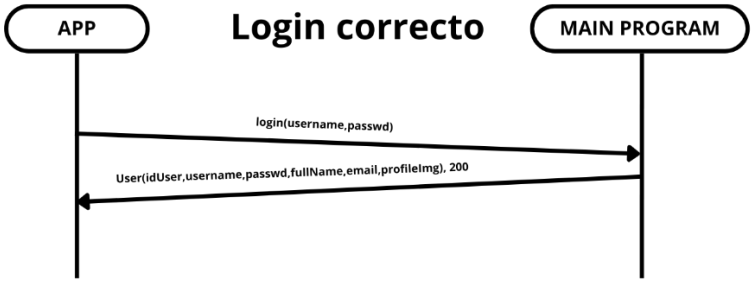


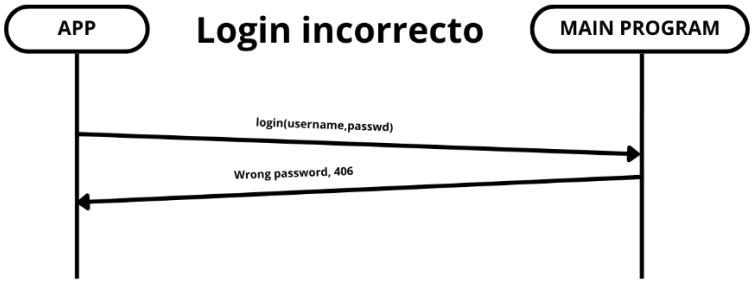




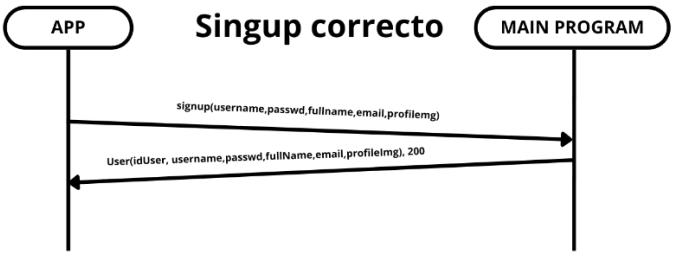
Diseño lógico de la aplicación:

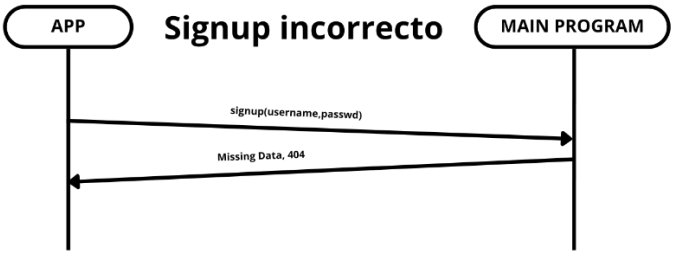
C.D.U Login



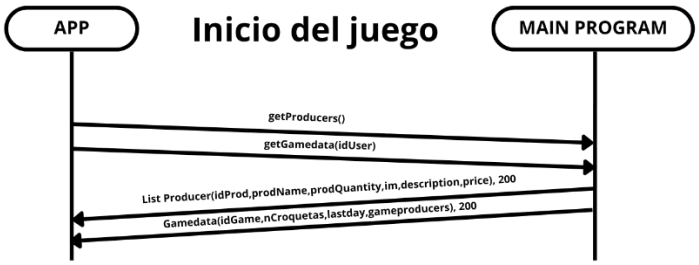


C.D.U Signup

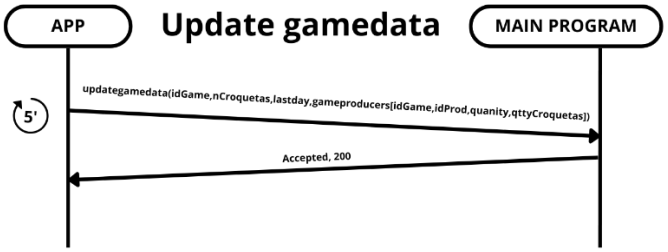




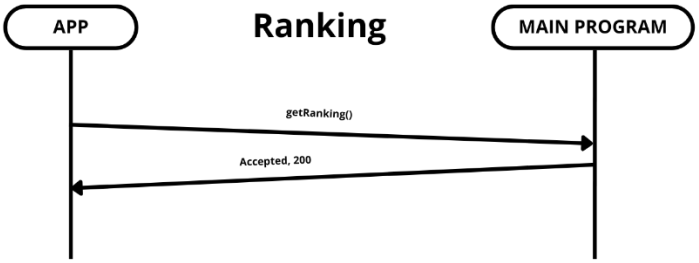
C.D.U Inicio del juego



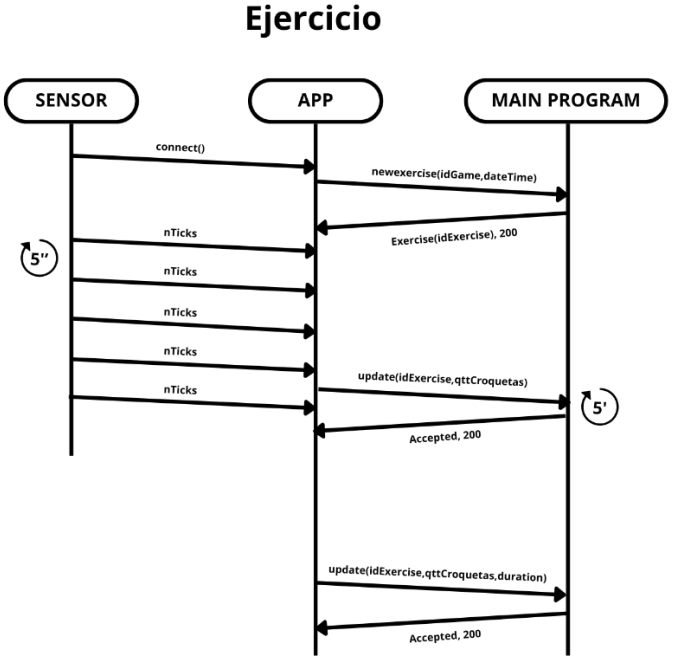
C.D.U Update gamedata



C.D.U Ranking



C.D.U Ejercicio



Descripción modular del sistema:

Main Program

- Controllers
  - ApiManager.py
  - ExerciseManager.py
  - GameManager.py
  - ProducerManager.py
  - RankingManager.py
  - UserManager.py
- Interfaces:
  - API
    - Api.py
  - MariaDB
    - MariaDBConnector.py
    - MariaDBDataReader.py
    - MariaDBDataWriter.py
- Models:
  - Contants.py
  - Exercise.py
  - GameData.py
  - GameProducer.py
  - Producer.py
  - Ranking.py
  - User.py

Croqueta Clicker

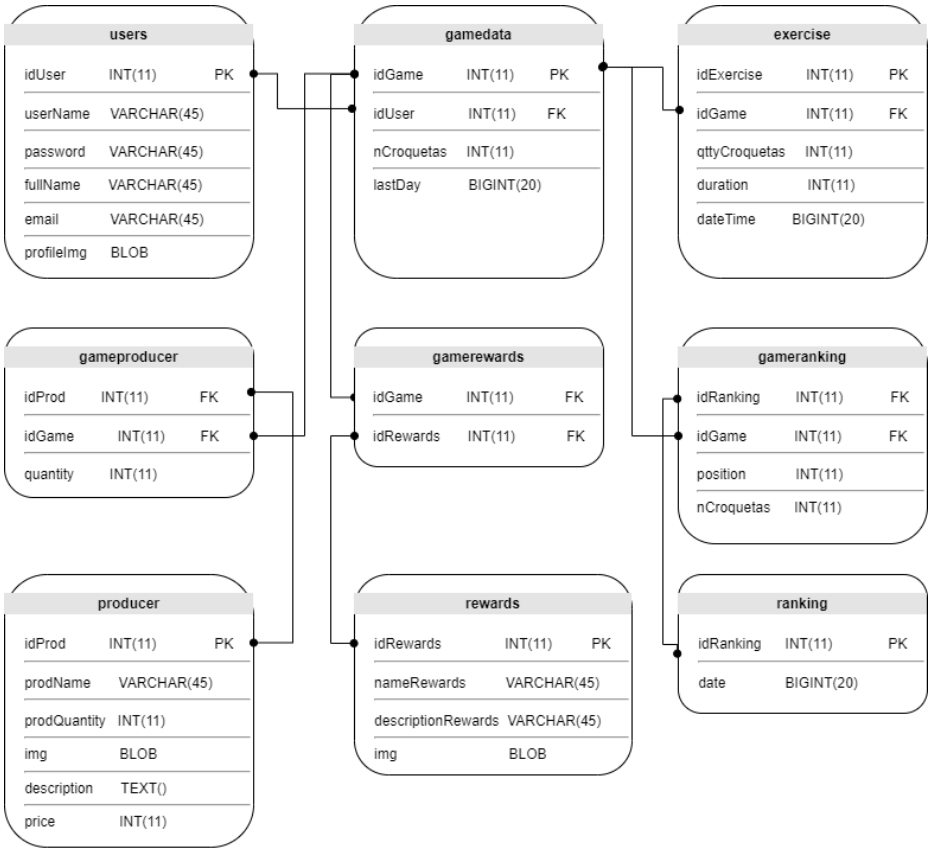
- Assets
  - Animations
  - Images
  - InterfaceElements
  - Scenes
  - Script
    - Bonus
      - Bonus.cs
    - InterfaceScripts
      - CursorsPanelController.cs
      - DontDestroy.cs
      - LoginPnlController.cs
      - ProducerPanelController.cs
      - RankingPnlController.cs
      - SoundBar.c
    - Managers
      - ApiManager.cs
      - BonusManager.cs
      - ProducerManager.cs
      - RankingManager.cs
      - SettingsManager.cs
      - SoundManager.cs
      - UserManager.cs
    - Models
      - Producers
        - InGameProducer.cs
        - Producer.cs
        - GameProducer.cs
      - API.cs

        - User.cs
        - RankingItem.cs
        - JsonHelper.cs
        - GameData.cs
    - TimeStamps
        - BaseTimeStamp.cs
        - ProducerTimeStamp.cs
        - ProductionTimeStamp.cs
    - GameGlobals.cs
    - GameManager.cs
    - MainMenuManager.cs
- Sounds
- TextMeshPro

Sensor

- Src
    - Main.cpp

Diseño de la base de datos:



Partes importantes del código:

1. **Gestión de archivos.**
   El guardado de datos del volumen de los ajustes se guarda en un archivo .dat. Además, lo guardamos de manera recursiva que modifica su dirección de guardado dependiendo del dispositivo en el que se está ejecutando se guarda en una dirección u otra.

   - Linux: Application.persistentDataPath points to $XDG_CONFIG_HOME/unity3d or $HOME/.config/unity3d.

- Android: Application.persistentDataPath points to /storage/emulated/<userid>/Android/data/<packagename>/files on most devices (some older phones might point to location on SD card if present), the path is resolved using android.content.Context.getExternalFilesDir.
- Windows Store Apps: Application.persistentDataPath points to C:\Users\<user>\AppData\LocalLow\<company name>.

```
0 references
public void Start() {
    FileInfo fileSettings = new FileInfo(Application.persistentDataPath + "/Settings/settings.dat");
    using (StreamReader sr = fileSettings.OpenText()) {
        SoundManager.instance.changeVolume(float.Parse(sr.ReadLine()));
        SoundManager.instance.changeEffectsVolume(float.Parse(sr.ReadLine()));
    }
}
```

2. **Patrones de diseño.**
   La mayoría de managers en la aplicación móvil usan un patrón Singelton para asegurar una instancia única de una determinada clase y proporcionar un camino de acceso genérico a dicha instancia.

```
4 references
public class ApiManager
{

    4 references
    private static ApiManager _instance;
    2 references
    private API _api;

    1 reference
    private ApiManager() {
        if (_instance == null) {
            _instance = this;
            _api = new API();
        }
    }

    0 references
    public void sendLogin(User u, API.msgReceivedCallback loginCallback) {

        _api.sendPostToApi("login", u.toJsonStr(true), loginCallback);

    }

    0 references
    public static ApiManager instance {
        get {
            if (_instance == null)
                return new ApiManager();
            else
                return _instance;
        }
    }
}
```

```
public class SoundManager : MonoBehaviour {

    2 references
    public static SoundManager instance;
    3 references
    public AudioSource musicSource;
    2 references
    public AudioSource sfxSource;

    0 references
    public void Awake() {
        if (instance == null) {
            instance = this;
            DontDestroyOnLoad(this.gameObject);
        } else {
            Destroy(this.gameObject);
        }
    }
```

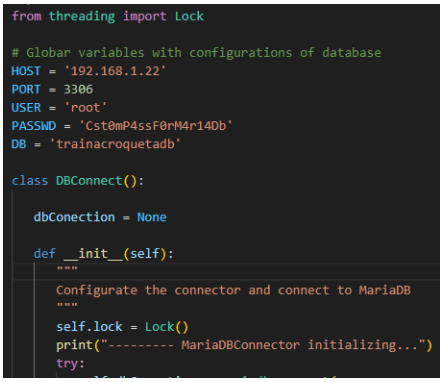
```
public class SettingsManager : MonoBehaviour
{

    2 references
    private float originalMasterVolume;
    2 references
    private float originalEffectsVolume;

    0 references
    void Start() {

        originalMasterVolume = SoundManager.instance.musicSource.volume;
        originalEffectsVolume = SoundManager.instance.sfxSource.volume;

    }
```

3. **Herencia y polimorfismo.**
   El sistema de TimeStamps de la aplicación móvil, para tener historial de las producciones individuales de cada tipo de productor.

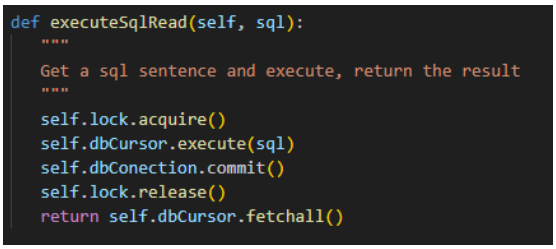
```
public class BaseTimeStamp {

    /// <summary>
    /// The number of Croquetas produced until a moment.
    /// </summary>
    2 references
    protected long _nProduced;

    /// <summary>
    /// A variable that represents a moment.
    /// </summary>
    3 references
    protected long _timeStamp;

    0 references
    public BaseTimeStamp(long nProduced) {
        _nProduced = nProduced;
        _timeStamp = DateTimeOffset.Now.ToUnixTimeSeconds();
    }
```

```
public class ProducerTimeStamp : BaseTimeStamp {

    /// <summary>
    /// The total ammount of producers of this kind in a determinated moment.
    /// </summary>
    3 references
    private int _nProducers;

    /// <summary>
    /// Constructs a ProducerTimeStamp instance.
    /// </summary>
    /// <param name=nProduced> The total ammount of Croquetas produced by all the pr
    /// <param name=nProducers> The total ammount of producers in a determinated mom
    0 references
    public ProducerTimeStamp(long nProduced, int nProducers) : base(nProduced) {
        _nProducers = nProducers;
    }
```

4. **Mecanismos para garantizar la exclusión mutua.**
   El uso de semáforos, en este caso, mutex para que no se acceda a la API varios procesos a la vez y a su vez que el acceso a la base de datos gestione las consultas una por una.
   En C# se usa Mutex() y en Python Lock().

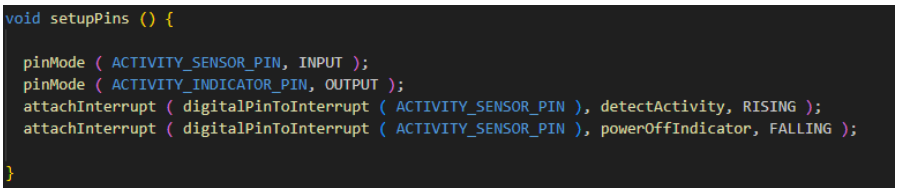


```python
from threading import Lock

# Globar variables with configurations of database
HOST = '192.168.1.22'
PORT = 3306
USER = 'root'
PASSWD = 'Cst0mP4ssF0rM4r14Db'
DB = 'trainacroquetadb'

class DBConnect():

    dbConection = None

    def __init__(self):
        """
        Configurate the connector and connect to MariaDB
        """
        self.lock = Lock()
        print("--------- MariaDBConnector initializing...")
        try:
```



```python
def executeSqlRead(self, sql):
    """
    Get a sql sentence and execute, return the result
    """
    self.lock.acquire()
    self.dbCursor.execute(sql)
    self.dbConection.commit()
    self.lock.release()
    return self.dbCursor.fetchall()
```

5. **Interrupciones**
   Las interrupciones en el código del ActivitySensor se usan para que en el instante en el que se detecta actividad, ésta se quede registrada. De esta forma evitamos tener que hacer *pooling* (evitar estar constantemente preguntado el estado del sensor) .



```
void setupPins () {

  pinMode ( ACTIVITY_SENSOR_PIN, INPUT );
  pinMode ( ACTIVITY_INDICATOR_PIN, OUTPUT );
  attachInterrupt ( digitalPinToInterrupt ( ACTIVITY_SENSOR_PIN ), detectActivity, RISING );
  attachInterrupt ( digitalPinToInterrupt ( ACTIVITY_SENSOR_PIN ), powerOffIndicator, FALLING );

}
```

6. **Callbacks.**
   Usamos los callbacks para obtener el resultado de una operación asíncrona y procesar el resultado de esa operación. Por ejemplo, cuando el la API se realiza una petición, no podemos saber la duración del proceso hasta su fin, y evitar así que el resto de la aplicación se pare, se hace de manera asíncrona y cuando el proceso finalice, éste llama al método callback.

```
public void sendLogin(User u, API.msgReceivedCallback loginCallback) {

    _api.sendPostToApi("login", u.toJsonStr(User.UserToJsonModes.LOGIN_DATA), loginCallback);

}

0 references
public void sendSignup(User u, API.msgReceivedCallback signupCallback) {

    _api.sendPutToApi("signup", u.toJsonStr(User.UserToJsonModes.SIGNUP_DATA), signupCallback);

}

0 references
public void sendGetProducerList(API.msgReceivedCallback respCallback) {

    _api.sendGetToApi("getproducers", respCallback);

}
```

ICD Entre API (Main program) y clientes:

1. login(username,passwd) --> idUser,username,passwd,fullName,email,profileImg

   http://trainacroqueta.ignorelist.com:1109/api/login [POST]

   {
   "username":"silviarc",

```
    "passwd":"silviarc"
}
```

Return Codes:
- [200] -> Accepted
- [404] -> Not Found Not user / Missing Data
- [406] -> Not Acceptable Wrong password
- [500] -> Internal server error
- [415] -> Unsopported media type

2. singup(username,passwd,fullname,email,profileimg) --> idUser

http://trainacroqueta.ignorelist.com:1109/api/signup [PUT]

```
{
    "username":"testing",
    "passwd":"testing",
    "fullname":"User testing",
    "email": "isthetestinguser@gmail.com"
}
```

Return Codes:
- [200] -> Accepted
- [404] -> Not Found User already exists / Missing Data
- [500] -> Internal server error
- [415] -> Unsopported media type

3. newexercise(idGame,dateTime) --> idExercise

http://trainacroqueta.ignorelist.com:1109/api/newexercise [PUT]

```
{
    "idGame":"11",
    "dateTime":"20200309"
}
```

Return Codes:
- [200] -> Accepted
- [404] -> Not Found Missing Data
- [500] -> Internal server error
- [415] -> Unsopported media type

4. updateexercise(qttCroquetas,idExercise) --> Null

http://trainacroqueta.ignorelist.com:1109/api/updateexercise [PUT]

```
{
    "idExercise":4,
    "qttCroquetas":30
}
```

Return Codes:
- [200] -> Accepted Updated successfully
- [404] -> Not Found Missing Data
- [500] -> Internal server error
- [415] -> Unsopported media type

5. finishexercice(idExercise,qttCroquetas,duration) --> Null

   http://trainacroqueta.ignorelist.com:1109/api/finishexercise [PUT]

```
{
   "idExercise":4,
   "qttCroquetas":300,
   "duration": 80
}
```

   Return Codes:
   [200] -> Accepted Updated successfully
   [404] -> Not Found Missing Data
   [500] -> Internal server error
   [415] -> Unsopported media type

6. getproducers() --> Object list (idProd,prodName,prodQuantity,img,description,price)

   http://trainacroqueta.ignorelist.com:1109/api/getproducers [GET]

   Return Codes:
   [200] -> Accepted
   [500] -> Internal server error
   [415] -> Unsopported media type

7. getgamedata(idUser) --> idGame,nCroquetas,lastday,gameproducers
   [Objectlist(idProd,quantity,qttyCroquetas)]

   http://trainacroqueta.ignorelist.com:1109/api/getgamedata [POST]

```
{
   "idUser": 26
}
```

   Return Codes:
   [200] -> Accepted
   [404] -> Not Found Missing Data
   [500] -> Internal server error
   [415] -> Unsopported media type

8. updategamedata(idGame, nCroquetas, lastday,
   gameproducers(idGame,idProd,quantity,qttyCroquetas)) --> Null

   http://trainacroqueta.ignorelist.com:1109/api/updategamedata [PUT]

```
{
   "idGame": 11,
   "nCroquetas": 300,
   "lastday": "20231203",
   "gameproducers": [
      {
         "idGame" : 11,
         "idProd": 1,
         "quantity": 30,
         "qttyCroquetas": 30
      },
      {
         "idGame" : 11,
```

```
            "idProd": 2,
            "quantity": 30,
            "qttyCroquetas": 30
        }  ]
}
```


Return Codes:
[200] -> Accepted
[404] -> Not Found Missing Data
[500] -> Internal server error
[415] -> Unsopported media type

9. getranking() --> Object list (idUser,username,ncroquetas)

http://trainacroqueta.ignorelist.com:1109/api/getranking [GET]

Return Codes:
[200] -> Accepted
[500] -> Internal server error
[415] -> Unsopported media type

Manual de usuario.

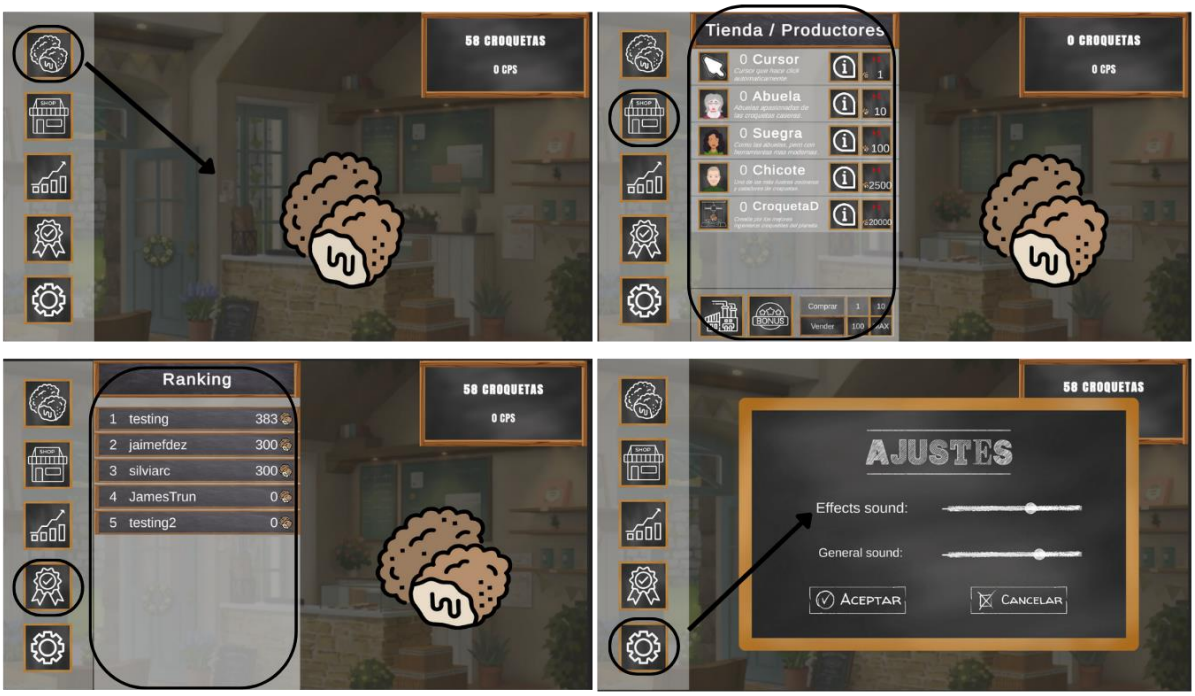
1. Pantalla de inicio.

El botón de Inicio de sesión abre un panel para poder insertar tu nombre de Usuario o Contraseña, en caso de querer registararse, la pulsar el botón de registro se mostrarán los campos Nombre completo y Email. Al pulsar el botón de Aceptar se habilitarar el botón de Inicio para poder comenzar el juego.
Mientras que no se Inicie sesión / Registrarse no se va a poder iniciar el juego.
El botón Ajustes, regula el volumen de sonido de los efectos del juego y el volumen del juego en general.

2. Funcionamiento básico del juego.

Hay 5 botones:

- El botón croqueta muestra la croqueta y el panel de información. Donde se muestra la cantidad de croquetas que tenemos disponibles en total y la producción de croquetas por segundo en total.
  Esta interfaz nos brinda el objecto Croqueta al cual podemos interactuar haciendo clicks.
- El botón Tienda / Productores nos ofrece una lista de cursores donde nos indica el precio de cada uno y la cantidad que nosotros tenemos disponibles. Estos productores nos ayudan a generar más croquetas por segundo.
  En el panel de abajo podemos elegir si Comprar / Vender productores y la cantidad de productores.
- El botón Ranking nos muestra un ranking de usuarios ordenado por mayor número de croquetas generadas.
- El botón Ajustes tienes las mismas funcionalidades que en la ventana inicial.



Bibliografía y fuentes de información. Se incluirían datos del material bibliográfico consultado, tanto de libros como de páginas web consultadas. Si son documentos en formato electrónico descargados de Internet, rogamos que se incluyan junto al proyecto

Sobre gamificación:

https://business.virtuagym.com/es/blog/gamificacion-videojuegos-y-el-impacto-en-la-industria-del-fitness/#:~:text=La%20gamificaci%C3%B3n%20en%20el%20deporte,-La%20gamificaci%C3%B3n%20est%C3%A1&text=Se%20trata%20de%20la%20conexi%C3%B3n,entornos%20de%20aptitud%20no%20deportivos.

https://www.educativa.com/blog-articulos/gamificacion-el-aprendizaje-divertido/#:~:text=La%20Gamificaci%C3%B3n%20es%20una%20t%C3%A9cnica,concretas%2C%20entre%20otros%20muchos%20objetivos.

https://lateralia.es/gamificacion-deporte-y-actividad-fisica/

https://www.bloglenovo.es/7-juegos-para-gamificar-ejercicio/

Sobre passenger:

https://www.phusionpassenger.com/
https://stackshare.io/stackups/apache-httpd-vs-passenger

Sobre ruby on rails:
https://rubyonrails.org/
https://es.wikipedia.org/wiki/Ruby_on_Rails
https://kodigo.org/hablemos-de-ruby-on-rails-ruby-on-rails-que-es-y-para-que-sirve/
https://guides.rubyonrails.org/getting_started.html

Sobre Apache:
https://docs.fedoraproject.org/en-US/quick-docs/getting-started-with-apache-http-server/

Sobre MariaDB:
https://aws.amazon.com/es/compare/the-difference-between-mariadb-vs-mysql/
https://www.hostinger.es/tutoriales/mariadb-vs-mysql
https://voidnull.es/instalar-mariadb-en-debian-12/
https://linuxgenie.net/how-to-install-mariadb-on-debian-12-bookworm-distribution/

Unity:
https://unity.com/es/community

Python:
https://www.python.org/about/gettingstarted/
Libro de PythonNotesForProfessionals

Sobre el ESP-32:
https://es.wikipedia.org/wiki/ESP32

Y mucho https://stackoverflow.com/