Facultad de Informática
UNIVERSIDADE DA CORUÑA

TRABAJO FIN DE GRADO
GRADO EN INGENIERÍA INFORMÁTICA
MENCIÓN EN SISTEMAS DE INFORMACIÓN

Producto de apoyo basado en la placa Micro:bit para personas con diversidad funcional

Estudiante: Alejandro López Fernández
Dirección: Adriana Dapena Janeiro
Dirección: Javier Pereira Loureiro

A Coruña, septiembre de 2020.

A mis padres, por apoyarme en todas las decisiones de mi vida.

Agradecimientos

A ASPACE, y en especial, a su fisioterapeuta, Rubén Carneiro, por su gran ayuda durante este proyecto y por su disposición a colaborar en todo lo que ha estado a su alcance. A los directores del proyecto, Adriana y Javier, por estar siempre disponibles para guiarme durante todo el desarrollo. A mis padres, por apoyarme durante todo este tiempo y aconsejarme para que todo saliera bien finalmente.

Resumen

El principal objetivo del proyecto es la creación de un entorno que facilite a terapeutas la realización de terapias basadas en juegos. Aunque el proyecto puede ser empleado para distintos colectivos de personas con diversidad funcional, el desarrollo se ha basado en las especificaciones dadas por ASPACE A Coruña (asociación sin ánimo de lucro dedicada a personas con parálisis cerebral). La aplicación desarrollada permite la gestión de los usuarios asociados a distintos servicios de esta asociación, la personalización de los juegos utilizados en las terapias y el seguimiento de los resultados de cada usuario.

En el proyecto se han desarrollados varios juegos controlados por el microcontrolador Micro:bit, que resulta un interfaz idóneo para personas con graves problemas de movilidad debido a que puede ser adherido en un objeto del mundo real. Más concretamente, nos hemos centrado en el trabajo de equilibrio. Se crearon varios escenarios que permiten trabajar de una forma lúdica el control de elementos en la pantalla del ordenador mediante el control de elementos de la vida real a los que se conectó un Micro:bit.

Abstract

The main goal of the project is to create an environment that eases game therapies to the therapists. Even if the project could be used for different colectives of people with functional disability, the development was based on the specifications given by ASPACE A Coruña (non-profit association which works with people with cerebral palsy). The app allows user management, having users assigned to different services offered by this association, customizing games in order to have successful therapies and tracing of the scores of each individual.

Some games were developed for the project, which are controlled by the Micro:bit controller, which turns a suitable interface for people with severe mobility problems, because it can be attached to a real-world object. We created some scenarios that lets the users work on a entertaining way the control of the elements on the computer monitor with a Micro:bit-attached object.

Palabras clave:

- Micro:bit
- Juegos
- Diversidad funcional

Keywords:

- Micro:bit
- Games
- Functional disability

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introdución

Este proyecto se centra en la realización de una aplicación de apoyo para las terapias que realizan personas con parálisis cerebral. Los usuarios de esta aplicación serán los terapeutas, encargados del diseño de las sesiones y del análisis del seguimiento, y personas con parálisis cerebral que realizarán las actividades lúdicas integradas en la aplicación.

La parálisis cerebral abarca un conjunto de trastornos crónicos debidos a una lesión o defecto en el desarrollo del cerebro inmaduro o a un trastorno neuromotor. El término ”parálisis” hace referencia a un problema en la utilización de los músculos, que se manifiesta con alteraciones en el control del movimiento, el tono muscular y la postura. Mientras que el término ”cerebral” quiere resaltar que la causa de la parálisis cerebral radica en una lesión en las áreas motoras del cerebro que controlan el movimiento y la postura. El tratamiento está fundamentado en cuatro pilares básicos [1]: fisioterapia, prótesis, fármacos y tratamiento quirúrgico.

El empleo de juegos en la fisioterapia es una herramienta poderosa para favorecer el aprendizaje, a la vez que se mejora la capacidad motora. Mediante el juego, se perciben sensaciones y estímulos agradables que hacen disfrutar de su movimiento, mejorando así la condición física y psicológica. Además, aporta satisfacción a nivel profesional y familiar [2, 3].

En ASPACE Coruña, institución fundada en 1977 como una asociación de apoyo a personas con parálisis cerebral, prestan servicio a personas con este tipo de discapacidad y otras de características similares. En el año 2019 contaban con 114 personas usuarias de sus servicios, de las cuales la amplia mayoría son totalmente dependientes a la hora de satisfacer sus necesidades básicas.

ASPACE Coruña lleva varios años utilizando actividades lúdicas como parte de las terapias de usuarios de distintos servicios, obteniendo muy buenos resultados. Sin embargo, en la actualidad no cuenta con un entorno que integre estos juegos con un sistema de gestión de usuarios y de seguimiento de los resultados obtenidos en las sesiones de terapia.

En el desarrollo del presente proyecto, además de apoyar el trabajo de los fisioterapeutas,

se tuvo en cuenta de forma muy especial a los otros usuarios de la aplicación: las personas con parálisis cerebral. Dado que tienen serias dificultades a la hora de moverse y comunicarse con los demás, fue necesario diseñar y adaptar juegos para que fuesen accesibles y adecuados con su forma de realizar este tipo de actividades.

1.1 Objetivos del proyecto

El principal cometido de este proyecto es el desarrollo de una aplicación que integre los sistemas que necesita el terapeuta y los usuarios para realizar una sesión, evitando pérdidas de tiempo innecesarias y unificando toda la información de la que disponen en un único sistema. Además, se pretende que el estudiante adquiera las competencias propias de la titulación de Grado en Ingeniería Informática, a la vez que realiza un servicio a una asociación sin ánimo de lucro y a un colectivo con discapacidad funcional.

Cabe mencionar que se pretende que los usuarios de la aplicación disfruten de unas terapias más amenas y divertidas gracias a los juegos incorporados dentro de la aplicación, lo que además permite un acercamiento a las tecnologías, que de otra manera no podrían usar. Es por ello que los juegos incorporados tienen una temática arcade, por la sencillez y facilidad de comprensión de este tipo de juegos, lo cual permite a alguno de los usuarios recuperar recuerdos.

Es imprescindible que los usuarios puedan interactuar con los juegos utilizando un medio adaptado a ellos, ya que no pueden usar un teclado o un mando como correspondería a cualquier juego que nos podamos imaginar. De ahí que el desarrollo esté basado en un controlador BBC Micro:bit, el cual contiene diferentes sensores que permiten controlar los movimientos que el jugador quiere realizar, que se transmiten a través de la comunicación Bluetooth Low Energy al ordenador para que estos puedan ser procesados, plasmados en el juego, y por último, almacenados en la base de datos correspondiente.

El desarrollo pretende facilitar la labor de los terapeutas. Es por ello que, al mismo tiempo que el usuario juega, todos los datos que están siendo recolectados por la aplicación a través de los juegos, se almacenan en una base de datos y son visibles desde la propia aplicación, de manera que se puede extraer información útil para los terapeutas, además de poderlos usar como estímulo hacia los pacientes para que estos se esfuercen más y mejoren en las sesiones siguientes.

1.2 Organización de la memoria

La presente memoria consta de seis capítulos que resumimos a continuación.

El capítulo 2 recoge las tecnologías empleadas en el desarrollo de la aplicación y hace una

revisión de otros proyectos que fueron tomados como punto de partida.

El capítulo 3 detalla el desarrollo realizado siguiendo la metodología ágil Scrum, indicando los roles, eventos y artefactos. Además, se comenta la planificación y la estimación de costes del proyecto.

A continuación, durante el capitulo 4, se desarrolla el proceso de realización del proyecto, paso por paso, durante todas sus fases, desde la recopilación de información hasta la implementación.

En el capítulo 5, relatamos cómo transcurrió la prueba realizada con usuarios reales de la aplicación, con la colaboración de fisioterapeutas y usuarios en las instalaciones de ASPACE Coruña.

Por último, trataremos en el capítulo 6 una serie de conclusiones y futuras líneas de desarrollo en el proyecto.

Capítulo 2

Tecnologías usadas

En este capítulo, se realiza una revisión de las diferentes tecnologías que se han empleado durante el desarrollo del proyecto, explicando en líneas generales en qué consisten y cuál es su funcionamiento básico. Además, se realiza una revisión de proyectos que fueron de gran utilidad para entender el funcionamiento y la programación de la placa Micro:bit.

2.1 Funcionamiento de la placa Micro:bit

La placa Micro:bit es el punto central de todo el proyecto y del desarrollo del código del mismo. De ella sale toda la información útil que le dará sentido a la aplicación y que permitirá jugar en los diferentes escenarios propuestos.

Micro:bit es una tarjeta programable desarrollada por BBC [4] con la intención de que los alumnos de las escuelas empiecen a familiarizarse con la informática y con la programación desde edades tempranas. Es codificable, principalmente, en lenguaje de bloques Scratch, lo cual resalta lo sencillo que resulta realizar programas que funcionen en ella, aunque también se pueden utilizar otros lenguajes de programación como JavaScript o Python.

La placa, que podemos observar en la figura 2.1 (extraída de [5]), incluye varios sensores que almacenan información en tiempo real sobre la aceleración de la placa, su temperatura, etc. Además, permite conectividad Bluetooth y a través de USB. Contiene también varios conectores que permiten añadirle otros tipos de sistemas, como sensores o lo que queramos conectar mediante pinzas a la parte inferior de la propia placa. Los 25 LED en la parte delantera de la placa, permiten formar una especie de pantalla, en la que podremos escribir palabras, formas o lo que queramos dentro de las limitaciones de tamaño de la misma, lo cual puede ser de utilidad en los proyectos que realicemos.

Otra característica de la placa Micro:bit es que si tenemos más de una, podemos programar una de ellas para que actúe como receptor de radio de las otras, de manera que cada una de las placas envíe datos a la que actúa como receptora, que podremos conectar a nuestro ordenador

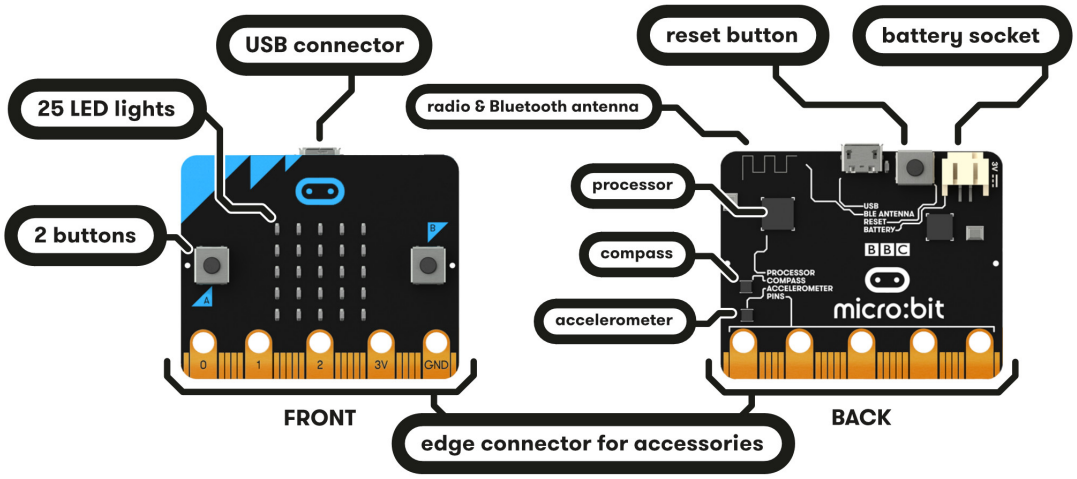

Figura 2.1: Placa Micro:bit y sus componentes básicos.

para poder extraer toda la información que queramos.

El paquete que recibimos al encargar la placa Micro:bit incluye también una batería que nos permitirá no tener siempre la placa conectada al ordenador, lo cual necesitaremos para poder manejarla durante los juegos cómodamente. Esto nos obliga también a comprender los fundamentos de la comunicación por Bluetooth para el envío de datos, lo cual se tratará en apartados posteriores.

2.2 Lenguajes de programación

Durante el desarrollo del proyecto se ha utilizado principalmente un único lenguaje de programación para favorecer la simplicidad a la hora de comunicar los diferentes módulos de la aplicación. Se escogió el lenguaje Python [6] por su sencillez y porque ofrece una variante del lenguaje para desarrollar con Micro:bit, llamada Micropython [7].

Las primeras pruebas de desarrollo para el código incluido en la placa Micro:bit se hicieron en Micropython que, entre otras capacidades, permite acceder de manera muy sencilla a los valores de los diferentes sensores de la misma, como el acelerómetro, el termómetro, entre otros. Sin embargo, en fases más avanzadas del desarrollo, posteriores al conocimiento de los métodos de comunicación por Bluetooth, se descartó usar Micropython para programar el Micro:bit, ya que al usar este lenguaje se deshabilita la posibilidad de emparejar nuestro Micro:bit al ordenador por Bluetooth. Esto nos deja con una única opción para su programación, ya que para mantener habilitada la posibilidad de emparejar dispositivo, el código debe estar codificado en bloques Scratch y, mas en particular, en el editor de MakeCode. Este código contiene únicamente unas pocas líneas en las que, tras conectarse por Bluetooth al ordenador

en cuestión, se encienden todos los servicios necesarios para la comunicación.

El resto de la aplicación se ha codificado totalmente en Python versión 3.7.5, tanto todas las pruebas intermedias que se han realizado como el resultado final del proyecto. Prácticamente todos los archivos que contienen código se estructuraron de una manera muy similar, ya que además casi todos ellos utilizan las mismas librerías. En el resto de apartados de este capítulo se expondrá con mayor detalle la estructura general de estos archivos, cuando se trate cada librería en particular.

A continuación, indicamos con más detalles los lenguajes y tecnologías empleadas para el desarrollo de todos los módulos que forman la aplicación.

2.2.1 Desarrollo de juegos

Teniendo en cuenta el lenguaje escogido para el desarrollo del proyecto y las necesidades que se habían expuesto para el control de los videojuegos, era necesario escoger una librería que permitiera un amplio control de los dispositivos periféricos y la creación de un videojuego sencillo sin utilizar demasiados recursos y de una manera intuitiva y no demasiado complicada para alguien que nunca ha realizado un desarrollo de estas características.

Por ello, tras visitar algunos proyectos similares, se escogió la librería Pygame [8], ya que cumple con todos los requisitos necesarios para el desarrollo.

Pygame permite la creación de diferentes sprites como clases de Python. Además, permite añadir las características que deseemos y la modificación de aquellas que sean interesantes para crear movimiento, como características entre las que se encuentran el centro del rectángulo que conforma el sprite, el punto superior o inferior del mismo, etc. De esta manera, aplicando sencillas ecuaciones para los sprites, podemos hacer que estos se muevan en función a la velocidad que indiquemos. En nuestro caso, el movimiento se consigue inclinando el Micro:bit hacia cualquiera de los lados, lo cual es fácilmente traducible a movimiento en un juego. Si realizamos una primera aproximación controlando nuestros sprites con teclas del teclado, Pygame nos permite accionar condiciones dentro de nuestro código para realizar ciertas acciones. Sustituyendo las pulsaciones del teclado por la condición que se tenga que dar con el valor extraído de la placa Micro:bit, obtendremos la movilidad que queríamos.

Para la parte gráfica, la librería nos permite seguir gozando de una extrema sencillez para nuestro código. Bastará con tener una función que nos permita obtener imágenes y poderlas incluir en la ventana del juego. Una vez tengamos todas las imágenes, tanto las de los sprites como el fondo del juego, bastará con hacerlas actualizarse en un orden coherente dentro del bucle del que consta el programa, para que no ocurran cosas extrañas durante la ejecución del juego.

Por último, cabe mencionar que todos los archivos que contienen código de los juegos se han estructurado de la misma manera para facilitar su lectura y su entendimiento, además

de estar convenientemente comentados con la misma intención. La estructura consta de la definición de variables globales, como la longitud y amplitud de la pantalla, la definición de las clases que conforman los sprites del juego, seguido de los métodos convenientes para la correcta ejecución del programa, como la inclusión de imágenes, conexión Bluetooth o subida y recolección de datos de la base de datos. Para terminar, el bucle principal del juego, en el que se inicializan los sprites, se comprueban los cambios que hayan podido sufrir los sprites en consecuencia a movimiento en el Micro:bit y se actualizan todos los sprites y textos que hay en la pantalla. Finalmente, dentro del bucle se especifica en que momentos se interrumpe la ejecución de los juegos, que será cuando se haga click sobre la cruz en la esquina superior derecha de la pantalla de juego o cuando se acaba el tiempo de la partida.

2.2.2 Comunicación por Bluetooth

El último añadido que tenemos que realizar para poder tener juegos que funcionen tal como queremos es la comunicación por Bluetooth. La placa Micro:bit cuenta con un sistema de conexión por Bluetooth Low Energy, que como dice su propio nombre, se diferencia del Bluetooth convencional por abaratar costes energéticos y recortar ligeramente los costes económicos sin tener diferencia alguna en cuanto a alcance y capacidad de la conexión.

Como se comentó anteriormente, Pygame presenta una interfaz que permite manipular los valores de las pulsaciones del teclado para manejar los sprites de los juegos, por lo que necesitamos extraer los valores de los sensores de Micro:bit para sustituirlos e introducir el control remoto desde la placa.

Los datos se almacenan dentro de la placa Micro:bit en estructura de servicios y características, esto es, cada grupo de funcionalidades, datos y otros valores se agrupan en función del sensor que los recoge o del componente al que hacen referencia. Por ejemplo, dentro del servicio de acelerómetro del Micro:bit, encontramos características como el valor que almacena el sensor, la frecuencia de actualización de los datos, entre otros. Para poder acceder a ellos, cada servicio y cada característica cuenta con un identificador único que nos permitirá referirnos a ellos de manera unívoca. La tabla 2.1 ejemplifica con el caso del servicio de acelerómetro.

Tipo	Descriptor universal (UUID)	Descripción
Servicio	E95D0753251D470AA062FA1922DFA9A8	Servicio de acelerómetro
Característica	E95DCA4B251D470AA062FA1922DFA9A8	Datos del acelerómetro
Característica	E95DFB24251D470AA062FA1922DFA9A8	Periodo de recogida de datos

Tabla 2.1: Identificación de servicios y características del acelerómetro de la placa Micro:bit

Para el acceso a estos datos, se han usado dos bibliotecas muy similares: Bluepy, variante para sistemas GNU/Linux, y Bleak, para versiones Windows. La librería Bluepy fue muy sen-

cilla de usar, ya que con tres métodos (conexión por dirección MAC, acceso a servicio y acceso a característica) se extraen los datos deseados. Gracias a ella, aprendí los fundamentos de este tipo de comunicación y se facilitaron en gran medida los pasos siguientes, que fueron cuestión de localizar los datos y extraerlos en el formato correcto. En cuanto a la librería Bleak, funciona siguiendo los mismos principios, pero de una manera ligeramente más compleja, ya que contempla funciones asíncronas que hay que considerar a la hora de estructurar el código.

Los datos que se extraen de las características no vienen en un formato amigable, sino que estos se encuentran en su mayor parte en enteros de 8 o 16 bits, los cuales debemos traducir antes de ser utilizados. Para ello se usa el método struct.unpack(), que nos permitirá realizar la traducción y usar los datos en los juegos.

2.2.3 Base de datos

Para que la aplicación sea completamente útil para las personas que la deseen usar, los datos relevantes han de ser almacenados en una base de datos. Por ello se diseñó un modelo de datos que contempla todas aquellas necesidades expuestas para que la aplicación funcione de acuerdo con las necesidades del personal de ASPACE Coruña. Se decidió desarrollar la base de datos en un sistema MySQL [9], debido a la facilidad de la integración con el código en Python y por lo sencillo que resulta la codificación de la base de datos gracias a la herramienta gráfica que incluye, MySQL Workbench, de gran utilidad a la hora de diseñar la base de datos, modificarla, realizar consultas de prueba, etc.

A la hora de integrar el código con las consultas y otras operaciones de la base de datos, debemos de instalar previamente el conector de Python para MySQL, el cual permitirá realizar todas las que necesitemos, y dará total libertad para estructurar los métodos que necesiten obtener o introducir datos. La estructura de almacenamiento/recogida de datos en la aplicación es muy sencilla: se crea una conexión, se inicializa un cursor, al cual le indicamos la operación que queremos realizar con los correspondientes parámetros, y se ejecuta. Posteriormente, podremos acceder al cursor para extraer los datos que este haya encontrado conforme a la consulta realizada. Es importante mencionar dos aspectos importantes. En primer lugar, el cursor sólo almacena una única consulta al mismo tiempo, lo cual significa que si tenemos varias consultas para hacer dentro de un método, tendremos que hacer una consulta, extraer datos, y realizar la segunda consulta. En segundo lugar, al finalizar el trabajo que hayamos realizado, tendremos que finalizar la transacción y cerrar la conexión para no dar pie a ningún error de lectura o escritura, o bien de saturación del servidor al cual le estamos haciendo peticiones.

2.2.4 Interfaces gráficas

Para que la aplicación sea cómoda y sencilla de usar, es importante desarrollar una interfaz gráfica para que los usuarios tengan todas las funciones a mano y que la iteración entre las diferentes páginas que constituyen la aplicación sea intuitiva y no requiera ningún tipo de conocimiento previo.

Con ayuda de los terapeutas de ASPACE Coruña, se diseñaron una serie de pantallas preliminares de lo que sería la aplicación en un futuro, que se irían refinando hasta obtener la interfaz final.

Para programar las interfaces en Python, existen numerosas librerías. Por la sencillez y claridad del código que ofrece, la decisión final fue la de usar la librería PySimpleGui [10], basada en otra más compleja, Tkinter[11]. La principal característica que ofrece esta librería frente a otras es la capacidad de esbozar la interfaz directamente en el código, es decir, de una manera intuitiva, mediante un conjunto que a su vez contiene otros conjuntos, podremos ir introduciendo todos aquellos componentes como botones, campos de texto, desplegables... que nuestra interfaz necesite, de manera que cada uno de los conjuntos que son parte del todo constituyen una línea en el formato gráfico, como podemos observar en el siguiente fragmento de código. El resultado de ejecutarlo puede verse en la Figura 2.2.

```
import PySimpleGUI as sg

sg.theme('DarkAmber')    # Color de fondo

layout = [[sg.Text('Persistent window')], #Diseño de la interfaz
          [sg.Input(key='-IN-')],
          [sg.Button('Read'), sg.Exit()]]

window = sg.Window('Window that stays open', layout)

while True:     #Bucle de eventos, que hacer en cada situación
    event, values = window.read() #Leer el estado de cada evento
    print(event, values)
    if event == sg.WIN_CLOSED or event == 'Exit':
        break

window.close()
```

Esto además nos permite configurar de manera muy sencilla la colocación de todos los widgets que introduzcamos, ya que podemos introducir espacios en blanco entre ellos a nuestro gusto. En líneas generales, todo es configurable con esta librería, además de hacerlo posible de una manera sencilla y intuitiva.

Otra ventaja que nos ofrece PySimpleGui es lo fácil que se hace integrar las demás librerías

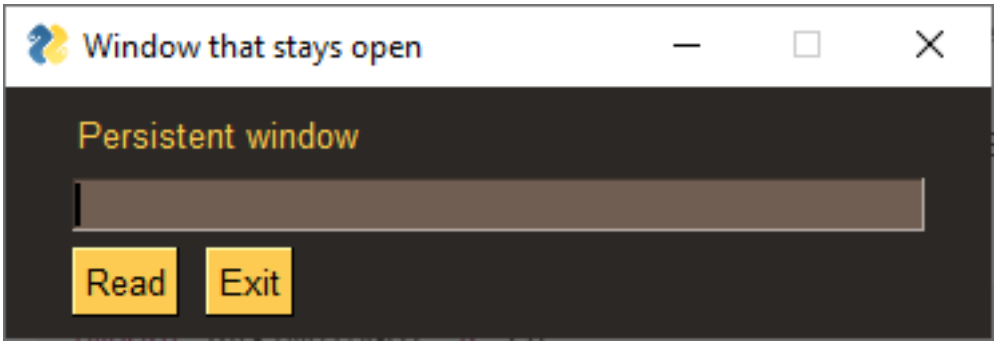

Figura 2.2: Resultado de ejecutar el código anterior

que se usan en la aplicación. En el resultado final, observamos que es imposible no mezclar en el código métodos de varias librerías diferentes y que podrían chocar entre ellos. Sin embargo, esto no es así, no solo no ha dado problemas sino que agiliza ciertos aspectos que igual con otra librería costarían más esfuerzo, principalmente, en lo que se refiere a configurar dentro de la interfaz valores obtenidos de la base de datos.

2.3 Proyectos similares

En esta sección se comentarán tres proyectos en los cuales se han encontrado características útiles para incluir en la aplicación o que simplemente han servido de apoyo o inspiración para continuar el desarrollo o descubrir alguna herramienta necesaria para el correcto funcionamiento de la aplicación.

2.3.1 Control remoto por BLE para Spotify en MacOS

Este proyecto [12] permite configurar un Micro:bit para controlar una instancia de Spotify en un sistema operativo MacOS, y realizar acciones dentro de él como subir y bajar el volumen, pasar canciones o pausar y reanudar una de ellas.

Para que la aplicación funcione correctamente, es neceario colocar nuestro Micro:bit en funcionamiento, que este se comunique mediante Bluetooth Low Energy con un script de Node.js que interpretará los mensajes transmitidos por Bluetooth para que finalmente, dicho script utilice la API de Spotify para que se realicen las acciones que estamos señalando.

El controlador Micro:bit tendrá programados varios estados: conectado, desconectado, ”play”, ”volumen” y ”siguiente”. Cuando el ordenador recibe una señal BLE, el Micro:bit pasará de estado desconectado a conectado. Para alcanzar el resto de estados, habrá que ir pasando uno por uno por todos ellos, pulsando los botones A+B del Micro:bit. En cada uno de los estados, los botones tendrán una función especial, por ejemplo, en el modo ”volumen”, el botón A baja el volumen y el B lo sube.

Este proyecto ha sido útil para nuestro desarrollo en un punto principal: la comprensión de como funciona la comunicación por Bluetooth Low Energy. Investigando más allá de la descripción del mismo, se consigió obtener la información necesaria para saber cómo establecer una conexión entre el Micro:bit y el ordenador. Además, la estructura de máquina de estados también fue útil para determinar cuando activar los servicios del controlador Micro:bit, de manera que cuando se conecte por BLE, estos son activados.

2.3.2 MicroBike: convertir tu Micro:bit en un controlador para juegos

Durante los primeros pasos del desarrollo, se observó en detalle el funcionamiento del proyecto descrito en [13], con la intención de conocer como integrar el controlador Micro:bit con los juegos que había que desarrollar. El proyecto en cuestión permite controlar juegos en un ordenador a través del Micro:bit utilizando dos librerías de Python: Pynput, que permite controlar y monitorizar los periféricos como el ratón o el teclado, y Bitio, que permite interactuar directamente con el Micro:bit y ejecutar su código directamente en otras plataformas como en el código escrito en el ordenador.

Esta última librería funciona de manera muy similar a una alternativa que existe para los juegos desarrollados en el lenguaje de bloques Scratch, Scratch.hex, que fue sugerida por los tutores del proyecto para que investigara acerca de su funcionamiento, y de esta manera encontré este proyecto. Lo que nos permite la librería Bitio es, básicamente, incluir un archivo ya codificado en el Micro:bit para que se comunique con el PC, y de esta manera podamos codificar todo lo referente con las funciones del Micro:bit dentro del resto del código del programa que estemos desarrollando.

El proyecto tiene un funcionamiento muy sencillo, mediante las dos librerías comentadas anteriormente, sustituimos las funciones del ratón o del teclado en el juego, por variaciones en los valores del acelerómetro del Micro:bit. Aunque la idea de incluir este funcionamiento en mi proyecto era interesante, finalmente se descartó porque la estructura del código del juego no era demasiado afín con esta manera de funcionar.

Sin embargo, el hecho de buscar algo que permitiera sustituir los valores del teclado por valores extraídos del Micro:bit me llevó a encontrar la librería que usaría para implementar los juegos, Pygame, que también permite interactuar con los valores de la pulsación de teclas del teclado y cambiar valores en función a otros eventos.

2.3.3 Micro:bit BLE

Este pequeño proyecto [14] consta de varios ejemplos de como funciona la comunicación Bluetooth Low Energy entre el Micro:bit y un ordenador con sistema operativo Linux, principalmente, la lectura de datos para su posterior tratamiento. Para ello, usa como intermediario una librería llamada Bluepy.

Expone un código en bloques Scratch único para todos los ejemplos, cuya función es activar los servicios y notificar la conexión a través de los LEDs del Micro:bit. A partir de ahí, mediante otras herramientas como ”gatttool” o ”bluetoothctl”, permite leer o escribir las características de los servicios que contienen los valores que nos interesen. Para ello debemos conocer los identificadores de los servicios y de las características para saber a cuales tenemos que acceder.

Por otra parte, los ejemplos codificados en Python consisten en pequeñas funcionalidades como leer la temperatura ambiente recogida por el Micro:bit, usar el Micro:bit como un control remoto para un coche de juguete, entre otros casos.

Aunque esto solo funcione en sistemas operativos Linux y todo el desarrollo de nuesttro proyecto estaba siendo en Windows hasta el momento de encontrar este proyecto, ha sido extremadamente útil para terminar de entender como funciona la comunicación BLE. Probablemente podría añadir que esa fue la pieza final del puzle que compone toda la información necesaria para poder codificar toda la aplicación, de manera que por fin pude unir los servicios del Micro:bit con los juegos desarrollados en el ordenador.

Junto con la librería Pygame comentada en el apartado anterior, que permite manipular los valores de las pulsaciones en el teclado, más los valores que puedo extraer de los servicios y características del Micro:bit, podemos controlar el juego en función a determinados parámetros de velocidad y sensibilidad que expongamos de manera externa al juego.

Una vez entendido el funcionamiento de este proyecto, solo faltaba conocer todos los identificadores necesarios para extraer los datos y como traducir estos desde el hexadecimal a un formato que el juego pueda entender, lo cual se explicará en capítulos posteriores. Otro detalle a tratar es cómo se hizo el cambio desde el sistema Linux a uno Windows. De una manera prácticamente idéntica a la librería Bluepy, encontramos la librería Bleak, con las mismas características pero exclusiva para sistemas Windows, además de elevar un poco más la dificultad de aprendizaje de la misma al contar con métodos asíncronos y una estructura de funcionamiento ligeramente diferente.

Capítulo 3

Metodologías de desarrollo

Una metodología proporciona un marco de trabajo a través de el cual se planifica, gestiona y administra un proyecto [15]. En función de las características de nuestro proyecto, podemos seguir dos ramas principales de metodologías: si nuestro proyecto es estático, esto es, no está sujeto a cambios a lo largo del tiempo de desarrollo y mantenimiento, lo más apropiado será escoger una metodología tradicional, como el modelo en cascada, o el incremental; en cambio, si nuestro proyecto debe adaptarse a numerosas variables, las cuales no se pueden controlar directamente, lo más apropiado será escoger una metodología ágil, como Scrum [16]. Las metodologías ágiles permiten usar ciclos de vida similares a los tradicionales, pero priorizando la gestión del cambio dentro del desarrollo.

El hecho de tener en cuenta los cambios que pueden afectar a los proyectos suele producir una mayor calidad en el producto final, además de una mayor participación por parte de los clientes, lo cual también repercute en la satisfacción de los mismos cuando finalmente obtienen un producto con el que están conformes, ya que es algo igual o muy similar a lo que ellos deseaban en primera instancia.

Este capítulo describe los aspectos más importantes del desarrollo del proyecto utilizando la metodología Scrum, elegida por ser la que más se adapta al desarrollo de la aplicación. Además, presenta la planificación y estimación de costes asociados al proyecto.

3.1 Metodología empleada para el desarrollo

Debido a la necesidad de hacer una aplicación que se adapte a las necesidades concretas de la asociación y de poder adaptarnos a nuevas situaciones, se ha decidido llevar a cabo un desarrollo siguiendo la metodología ágil Scrum. Para conseguir un desarrollo exitoso de esta metodología, se ha procurado seguir de la manera más fiel la guía de referencia de Scrum [16].

Dicho esto, se ha determinado los roles de las personas que forman parte del proyecto, los artefactos y los eventos.

3.1.1 Roles de Scrum

Al seguir esta metodología, debemos tener en cuenta que ciertas personas deberán asumir ciertos roles para que todo fluya y se puedan asignar responsabilidades a cada uno. Dicho esto, tendremos un ”Product Owner”, un ”Scrum Master” y un equipo de desarrollo.

- Javier Pereira, co-director del proyecto, ha sido el ”Product Owner” dado que es la persona más cercana a ASPACE Coruña, se encargó de velar por sus intereses frente al equipo de desarrollo. Hizo de puente entre los fisioterapeutas de la asociación y el equipo de desarrollo de este proyecto trasladando las necesidades de los mismos entre ambos puntos.

- Adriana Dapena, co-directora del proyecto, actuó como ”Scrum Master”. Se encargó de la correcta ejecución de las recomendaciones de Scrum, así como de controlar que sus eventos se lleven a cabo y que los artefactos se empleen conforme a su utilidad. También determinó las funcionalidades que se tenían que llevar a cabo y realizó otras tareas asociadas a supervisión del proyecto.

- Alejandro López fue el desarrollador del proyecto. Se ocupó de implementar todas aquellas funcionalidades recogidas y catalogadas como necesarias. Habitualmente, estos equipos tienen un tamaño de entre 5-9 personas, cantidad justa para favorecer el trabajo en equipo y que ningún trabajador entorpezca en el trabajo de los demás. Sin embargo, al tratarse del un Trabajo Fin de Grado, el desarrollo fue realizado íntegramente por el alumno.

3.1.2 Artefactos de Scrum

Los artefactos de Scrum son objetos que se usan para facilitar el desarrollo de los proyectos y al mismo tiempo, recopilar toda la información sobre el mismo y que puedan servir como documentación para el propio equipo. Los artefactos más importantes y de los que hablaremos para este proyecto son el ”Sprint Backlog” y el ”Product Backlog”.

El ”Product Backlog” contiene todo aquello que puede ser necesario para el producto final. Cada funcionalidad debe reflejar quién es el usuario de esa funcionalidad, qué debe hacer y con qué finalidad la quiere, para su correcta comprensión y priorización. A medida que el proyecto va avanzando, en cada Sprint se irán refinando las entradas del ”Product Backlog”, y se irán añadiendo a los ”Sprint Backlog”.

No se explicará en profundidad todo lo que contiene el ”Product Backlog”, ya que en el capítulo 4 se detalla todo el proceso de desarrollo y búsqueda de información que ha habido durante el proyecto. A continuación, se expondrá en que apartados encontraremos lo que se ha realizado en cada uno de los sprints.

- **Primer Sprint: Búsqueda de información sobre Micro:bit** → Podremos encontrar lo realizado durante este Sprint desde la sección 4.1 hasta la sección 4.3.1. Tuvo una duración de un mes, y consistió en toda la extracción de información de los lenguajes de programación que se han utilizado y como aplicarlos dentro de la aplicación.

- **Segundo Sprint: Comunicación por Bluetooth** → Este sprint, cuya duración fue de aproximadamente otro mes, consistió en conocer los fundamentos de la comunicación Bluetooth y qué librerías serían útiles para esta aplicación. Consta de este contenido las secciones entre la sección 4.4 y la sección 4.4.4

- **Tercer Sprint: Programación de los juegos** → Comprende las secciones 4.5 y 4.6, que tratan sobre el desarrollo de los escenarios de juego de forma independiente y como comunicarnos con el ordenador para controlar el propio juego con la placa Micro:bit. Tuvo una duración de tres semanas.

- **Cuarto Sprint: Diseño de la aplicación** → Dentro de este Sprint, que duró aproximadamente un par de semanas, se ha llevado a cabo todo el diseño de la base de datos utilizada, y de una primera fotografía de como resultaría ser la interfaz gráfica. Está contenido dentro de la sección 4.7.

- **Quinto Sprint: Implementación de la aplicación** → Con todos los recursos necesarios listos y aprendidos, y con el diseño realizado, el trabajo de implementación es totalmente directo. Este trabajo se describe en las secciones 4.8.1 y 4.8.2, y duró aproximadamente un mes.

3.1.3 Eventos de Scrum

Los eventos de Scrum consistieron en una serie de reuniones que sirvieron para controlar el desarrollo del proyecto siguiendo los preceptos de la metodología, haciendo partícipe a todo el mundo y favoreciendo la comunicación dentro del equipo.

Dentro de estos eventos encontramos el **Scrum Diario**, que consiste en explicitar que se ha hecho durante el día, y que se hará durante el día siguiente, además de comentar los problemas que se hayan encontrado durante la jornada. Se realiza de manera diaria, siempre pudiendo ser a la misma hora, y con una duración de quince minutos. En el marco de este proyecto, no se han realizado estas reuniones de la manera convencional que se aconseja en el manual de Scrum, sino que se han realizado en momentos en los que se han alcanzado ciertos hitos dentro de cada Sprint, en los cuales el equipo de desarrollo comunica al resto de los integrantes del proyecto la situación actual, los pasos siguientes, y los problemas encontrados.

Por otra parte, otro evento como es la **Planificación del Sprint**, si se ha llevado de una manera constante durante el desarrollo. Es un evento que sirve para detectar que contenido

del Product Backlog se va a refinar para introducirlo en el Sprint Backlog que se vaya a a iniciar en ese momento. Se realiza siempre antes de iniciar el siguiente Sprint, y debe tener una duración máxima de ocho horas para Sprints de duración máxima (1 mes).

La **Revisión del Sprint** es otro evento, que se realiza a la finalización de cada Sprint para comprobar el progreso, de manera que se hayan cumplido todos los objetivos marcados en el Sprint Backlog. Para este proyecto, cuando se han alcanzado los objetivos marcados en la Planificación del Sprint, se ha convocado una reunión con el resto del equipo para hacer una reunión de revisión del progreso. Para agilizar el proceso, en este proyecto, la revisión del Sprint y la Planificación del Sprint siguiente se realizan en la misma reunión física, pero respetando las funciones que ofrece cada una de los eventos.

Existe una última reunión, la **Retrospectiva del Sprint**, que consiste en revisar como se ha trabajado durante el Sprint, que mejorar y que ha salido bien. En este proyecto, esta reunión no se ha llevado a cabo en ningún momento, ya que por el tamaño del equipo de trabajo, no tendría demasiado sentido debatir este tipo de información.

3.2 Planificación y estimación de costes

En esta sección vamos a presentar la planificación del proyecto y el seguimiento a tiempo cumplido. También estudiaremos cuáles han sido los gastos de este proyecto, y cuál ha sido el coste de los diferentes elementos que han intervenido en él.

3.2.1 Planificación y Diagrama de Gantt

Siguiendo los pasos de la metodología Scrum, hemos de ser conscientes a la hora de planificar que las tareas a realizar van a ser divididas en Sprints, de duración máxima de un mes. Por lo tanto, las tareas que comprendan cada Sprint han de ser planificadas de tal manera que se respeten los tiempos, por lo que tendremos que ser muy cuidadosos con las estimaciones, y preferir hacer previsiones más realistas en vez de aventurarse a incluir demasiadas tareas en el mismo Sprint. Aunque se haga una planificación inicial general de todas las tareas a hacer, antes de cada Sprint se realiza el Sprint Planning, que acotará un poco más los tiempos estimados para las tareas que estén implicadas en cada momento.

Aclarado todo esto, el resultado de la planificación inicial se ha plasmado en el Diagrama de Gantt que se muestra en la figura 3.1 .

El diagrama de Gantt no refleja el tiempo estimado de realización de la memoria, pero para realizar la planificación se ha contemplado dejar un mes para realizarla, ya que hasta el último momento no podemos estar seguros de como se estructurará y cual será el contenido de la misma.

Gracias a haber realizado una planificación realista y no optimista, no nos hemos quedado

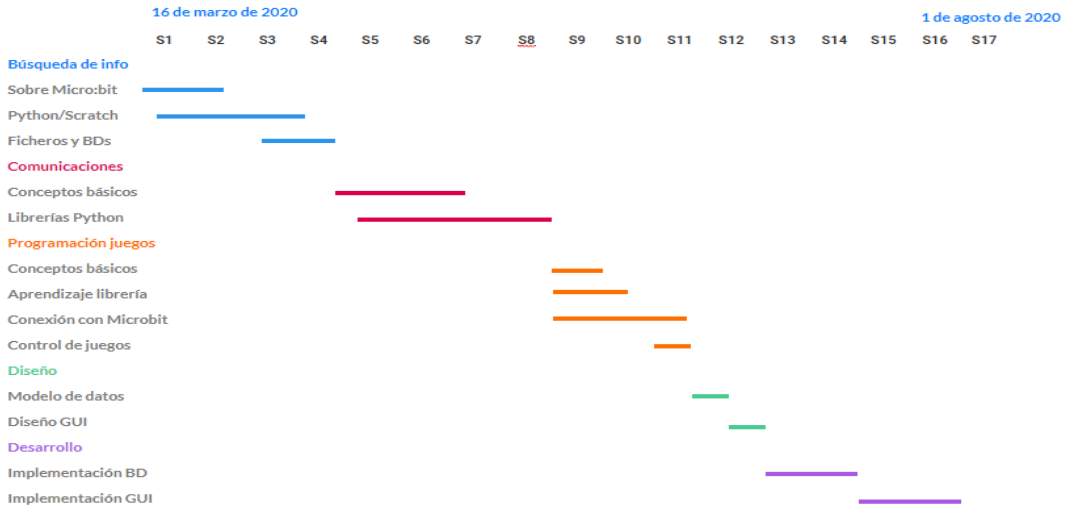

Figura 3.1: Diagrama de Gantt de la planificación inicial del proyecto

cortos de tiempo conforme lo planificado y han sobrado dos semanas, en las cuales se ha añadido una tarea de revisión de los diferentes módulos de la aplicación, que ha sido útil para encontrar algunos errores o detalles que no eran demasiado convincentes en el resultado final y se han corregido en esta tarea extra.

Para contrastar el trabajo realizado finalmente con la estimación se intenta aproximar lo más posible el trabajo realizado, que ha sido, de alrededor de 4 horas diarias, trabajando 5 días a la semana, lo cual resulta en 20 horas/hombre a la semana, que si multiplicamos por las 17 semanas de trabajo, implica que se han planificado un total de 340 horas/hombre. En la tabla 3.1 [1] podemos ver un desglose por Sprints de las horas de trabajo fina1les.

Sprint	Objetivo	Fecha Inicio	Fecha Fin	Horas
1	Búsqueda de información	16/03/20	10/04/20	80
2	Comunicaciones	13/04/20	30/04/20	56
3	Programación de juegos	04/05/20	27/05/20	72
4	Diseño	28/05/20	15/06/20	48
5	Desarrollo (BD y GUI)	26/06/20	17/07/20	64
Extra	Revisión de módulos	20/07/20	31/07/20	20

Tabla 3.1: Resumen de horas empleadas por Sprint

En comparación con la planificación inicial, que contaba con aproximadamente con 340 horas de trabajo, finalmente resultan en 320 horas, contando únicamente con lo planificado. Las 20 horas restantes se cuadran con el Sprint extra de revisión, durante el cual se ha trabajado de manera más esporádica, sin seguir la media de 4 horas diarias mencionada anteriormente.

[1] No se ha realizado trabajo entre el 16/07 al 26/07 debido a unas vacaciones.

3.2.2 Estimación de costes

Para realizar las estimaciones referentes a los costes, se tuvieron en cuenta los recursos hardware, software y humanos. En cuanto a los recursos hardware, se contó con el ordenador usado para todo el desarrollo y el coste de la placa Micro:bit usada para el desarrollo. El equipo usado es un Lenovo B50 del año 2015, con un precio de 500 €, que ha sido usado para este proyecto, durante aproximadamente 4 meses, lo cual resulta en un coste de 500€/5 años = 100 €/año, 100€ * 1/3 de año = 30 € por los cuatro meses en los que se ha utilizado el ordenador. La placa Micro:bit ha tenido un coste de 34,90 €, por lo que el coste del hardware asciende a 64,90 €. Por parte del software, todo el descrito en el capítulo 2 es totalmente gratuito, por lo que no existe ningún tipo de coste asociado a estas herramientas.

Por último, el coste de los recursos humanos. Diferenciamos el coste de dos recursos: desarrollador y directores de proyecto. Aunque el coste que se imputa a cada uno varía de manera muy significativa en función de la empresa que tratemos, para este proyecto se consideró que el desarrollador cobra 20 € la hora, mientras que el director de proyecto cobra a razón de 30 € cada hora de trabajo. Sabiendo esto, se calculó que el desarrollador, si ha trabajado 340 horas, tiene un coste total de 6.800 €. Para calcular el tiempo invertido por los directores se consideró que, a través del tiempo consumido por los eventos Scrum, han intervenido en 5 Sprint Planning (1 hora de duración, aproximadamente) y otros tantos Sprint Review (1 hora de duración, aproximadamente), además de las consultas esporádicas que conforman los Daily Scrum de este proyecto (estimación de aproximadamente 10 horas), de lo que resulta un total de 20 horas en las que los dos directores han intervenido en el proyecto, que se traduce en un coste de 600 €. En la tabla 3.2 se especifican todos estos costes.

	Sueldo	Horas	Coste
Desarrollador	20€/hora	340	6.800 €
Director de proyecto (x2)	30€/hora	20	600 €
		Total de recursos humanos	7.400 €

Tabla 3.2: Costes de recursos humanos

Finalmente, sumando todos los costes mencionados hasta el momento, el proyecto tiene un coste estimado de 7.464,90 €, contando recursos software, hardware y humanos.

Capítulo 4

Desarrollo

En este capítulo se irá detallando de manera precisa el desarrollo del proyecto desde el inicio hasta el último día. El desarrollo, contando con todas las fases a continuación descritas, tuvo una duración aproximada de 4 meses, contando con todas las etapas de búsqueda de información, análisis, diseño e implementación de los diferentes módulos de la aplicación.

Los apartados se irán exponiendo en orden cronológico, comenzando con los primeros pasos, sobre cómo se encontró toda la información necesaria para el posterior desarrollo, avanzando hasta los últimos detalles de la implementación.

4.1 Recopilación de información sobre las herramientas a usar

El primer paso al comenzar con este proyecto fue recopilar información sobre las herramientas que se iban a utilizar para el desarrollo. Para este proyecto en particular, se usaron diversos elementos desconocidos por el estudiante en el momento de comenzar la investigación, por lo que hubo que dedicarle un tiempo considerable a buscar fuentes de información y aprender a usar correctamente todas las herramientas y funcionalidades.

La primera herramienta a tratar fue la placa Micro:bit (sección 2.1), que es el punto central del desarrollo de la aplicación, ya que es el controlador para los juegos, y el lugar del que extraeremos los datos para registrar en la aplicación y que podrán ser útiles para los fisioterapeutas que la usarán.

Para comenzar el desarrollo, lo primero sobre lo cual se ha buscado información es la programación de la placa. Es posible programarla en tres lenguajes de programación diferentes: Scratch, Python o JavaScript. Para el trabajo que se comentará en este apartado y en los siguientes, no se explicará en mayor profundidad la programación en JavaScript, ya que no ha sido relevante para el trabajo.

La intención durante los primeros pasos del desarrollo fue crear una pequeña aplicación que fuera capaz de extraer información de los diferentes sensores de Micro:bit, sacarlos al PC

y posteriormente introducirlos en una base de datos. De esta manera, obtendría información sobre cómo codificar la placa Micro:bit, con los datos que me harían falta en pasos posteriores, además de empezar a gestionar la persistencia de los datos en una base de datos. En los apartados siguientes se comentará a través del desarrollo de la aplicación de prueba cómo fue el proceso de aprendizaje para la codificación de la placa Micro:bit.

A continuación, lo siguiente a realizar era comprender cómo realizar la integración del código con la base de datos, es decir, el camino desde que se extraen los datos de los sensores hasta que están almacenados en una columna en una tabla de una base de datos.

Otro paso intermedio que también se ampliará en apartados siguientes es un pequeño aprendizaje introductorio al lenguaje de bloques Scratch, ya que existen varios proyectos similares realizados en este lenguaje, y buscando similitudes con Python, se facilitó la labor en muchos puntos.

4.2 Programación con Micro:bit

Como se comentó anteriormente, para el desarrollo de este proyecto, se han usado dos lenguajes a la hora de programar la placa Micro:bit: Python y Scratch. Los dos lenguajes son muy sencillos a la hora de aprenderlos, y existe mucha documentación sobre ellos y como codificar cosas similares a las que estamos intentando alcanzar.

4.2.1 Programación en Micropython

Como se comentó en la sección 2.2, Micropython [7] es una librería del lenguaje de programación Python, que extiende las funciones básicas del propio lenguaje para el manejo de las funcionalidades de la placa Micro:bit como, por ejemplo, métodos que nos permitan acceder al valor del termómetro, o cualquiera de las componentes del acelerómetro... Cuenta con una documentación bastante amplia [17], en la cual se recogen todos los métodos necesarios para interactuar con todos los sensores y demás funcionalidades de la propia placa.

Continuando con el ejemplo del programa de prueba, se escogieron unos pocos valores para ser extraídos, como son la componente en el eje x del acelerómetro, la temperatura y el nivel de luz recogida por los leds. Con esos datos, abrimos un archivo .csv en el cual volcamos todos los datos por columnas. La ejecución se realiza mediante un bucle infinito, que cortaremos a voluntad del usuario a través de la terminal de comandos. El código es el siguiente:

```
1 from microbit import *
2 from math import sqrt
3
4 acc = 0
5 temp = 0
6 luz = 0
```

```
datos = open("datos.csv", "w")
display.scroll("Escribiendo...", delay=150, wait=False, loop=True,
    monospace=False)
datos.write("aceleracion_x,temperatura,luz\n")
while True:
    x = accelerometer.get_x()
    datos.write(repr(x) + ",")
    temp = temperature()
    datos.write(repr(temp) + ",")
    luz = display.read_light_level()
    datos.write(repr(luz) + "\n")
    sleep(1000)
datos.close()
```

Con esto, obtenemos un archivo .csv generado dentro de la memoria de la placa Micro:bit. Cabe mencionar que cuenta con un sistema de ficheros muy sencillo, que cuenta con 30 KBytes de almacenamiento, lo cual no nos permite almacenar grandes archivos pero en ciertas ocasiones no es suficiente. Por este motivo, el método escogido para esta primera prueba fue totalmente descartado.

4.2.2 Programación en Scratch

De la misma manera que en el lenguaje MicroPython, en la sección 2.2 se explicó en qué consiste el lenguaje de bloques Scratch. Para ponernos un poco en contexto y recopilar formas de realizar nuestro desarrollo, los directores del proyecto presentaron al alumno varios escenarios que podrían adaptarse más tarde para convertirlos en juegos. Dichos juegos están codificados en Scratch, como podemos ver en la figura 4.1.

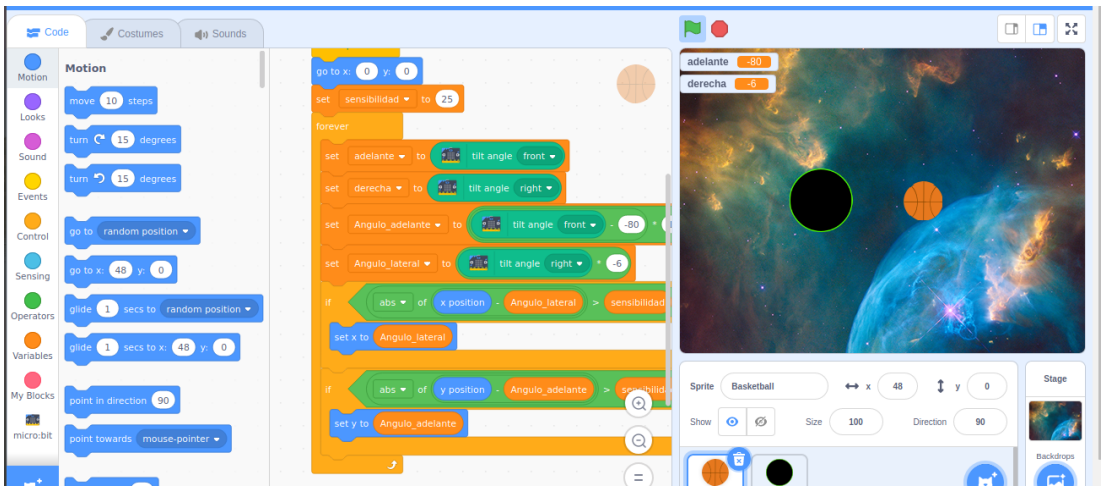

Figura 4.1: Ejemplo de código en Scratch para uno de los ejemplos de juegos propuestos.

Teniendo en cuenta los requisitos para nuestro proyecto, en esta fase se tuvo que tomar la decisión sobre qué lenguaje de programación se iba a utilizar más adelante para programar la placa Micro:bit. Se realizó el siguiente razonamiento:

- Scratch ofrece una ventaja clave sobre MicroPython para la codificación de la placa Micro:bit. Por una parte, MicroPython nos facilita mucho, gracias a la cantidad de métodos que ofrece, el hecho de acceder a cada uno de los datos que necesitemos y almacenarlos en un fichero externo, algo que con Scratch nos resultaría imposible, ya que no permite interactuar con otros ficheros.

- Sin embargo, la ventaja principal de Scratch es que únicamente el código contenido en la placa Micro:bit que esté codificado en este lenguaje nos permite activar la función de emparejamiento Bluetooth de la propia placa, lo cual será de vital importancia en fases posteriores del desarrollo.

Todo esto se explicará en mayor detalle en la sección 4.4, en el que se tratarán todos los temas relacionados con la comunicación entre el Micro:bit y el ordenador.

De esta fase dentro del desarrollo, la conclusión obtenida fue que la codificación de la placa Micro:bit debería ser en lenguaje de bloques Scratch, para permitir que el emparejamiento Bluetooth se pueda realizar y no necesitemos estar conectados por cable al ordenador.

4.3 Extracción información de Micro:bit al PC

Una vez entendido cómo codificar la extracción de datos de los diferentes sensores de la placa Micro:bit, es necesario tener alguna manera de poder manejar esos datos fuera de la placa, para poderlos almacenar de alguna manera.

Como veíamos en la sección 4.2.1, con ese código obteníamos un archivo con todos los datos extraídos, pero, ¿cómo extraemos ese archivo del sistema de ficheros de Micro:bit?

En este sentido, debemos mencionar que la extracción de este fichero de la placa no es para nada trivial. En el caso que estamos tratando, se extrajo dicho fichero a través de un editor de código (Code with Mu for Python [18]). Este editor permite acceder al sistema de ficheros y extraer los ficheros generados. No hemos encontrado ninguna otra manera de poder extraer los datos del sistema de almacenamiento.

Sabiendo esto, todo apunta a que en fases posteriores del desarrollo se deberá usar otro método de extracción de datos.

4.3.1 Generación de ficheros / introducción en base de datos

En las fases anteriores, hemos decidido el método para codificar la placa Micro:bit y encontrado la manera de extraer los datos a nuestro ordenador, de lo que obtenemos un fichero

.csv con el resultado de la ejecución durante unos instantes del código incluido en la placa. Una vez extraído el fichero csv de la placa Micro:bit, era turno de convertir esos datos en valores que insertar en una base de datos que, como se comentó en la sección 2.2.3, es MySQL. Para emplear métodos propios de una base de datos en nuestro programa de prueba, debemos de usar un driver que nos permita gestionar la base de datos desde el código Python. Para ello, descargamos la librería mysql.connector desde la página de descargas de MySQL.

Para poder introducir/modificar/borrar/leer datos de una base de datos debemos de tener en cuenta que las operaciones dentro de una base de datos son transaccionales, y que dichas transacciones deben de cumplir las propiedades ACID [19]. Conociendo todos estos detalles, se plantea una estructura a seguir siempre que interactuemos de aquí en adelante con la base de datos:

```
import mysql.connector

    try:
        cnx = mysql.connector.connect(
            user="xxx", password="xxx", host="127.0.0.1",
    database="xxx"
        )
        cur = cnx.cursor(buffered=True)
        print("Me conecté")
    except:
        print("No me pude conectar bien")
    try:
        '''Aqui va el codigo para realzar operaciones CRUD sobre
    las filas en la bd '''
        sql = ''' Sentencia SQL '''
        val = ''' Valores obtenidos como parámetros '''
        cur.execute(sql, val)
        cnx.commit()
        cnx.close()
    except:
         '''Gestión de los errores'''
```

Esta estructura intenta en primer lugar establecer una conexión con la base de datos, a través de los datos que solicita la función *mysql.connector.connect()*. No será estrictamente necesario tener estos datos escritos directamente en esa parte del código, sino que pueden ser extraídos de ficheros de configuración. Por sencillez, a lo largo de la aplicación, se reflejarán los datos en el código. Habrá que tener en cuenta que la instalación de la base de datos concuerde con los datos aquí expuestos, para evitar los errores de conexión.

4.4 Comunicación por Bluetooth

Una vez descrito el lenguaje para codificar la placa Micro:bit y cómo nos comunicaremos con la base de datos, tenemos que concretar el último punto que nos queda de los que hablamos en el apartado anterior: la extracción de los datos de la placa Micro:bit al ordenador. Una vez descartada la idea de mantener la placa conectada al ordenador por USB, la única idea posible para desarrollar es la comunicación a través de las funcionalidades Bluetooth que ofrece Micro:bit. Esto enlaza perfectamente con el lenguaje escogido para programarla, ya que si no hubiéramos escogido la idea de programar con Scratch, no podríamos emparejar el dispositivo con el ordenador, y por tanto, no saber cómo enviar los datos. Ahora que tenemos una explicación convincente de porqué hacer esto, se irá detallando en mayor medida todo el proceso hasta tomar la última decisión en cuanto a comunicación entre dispositivos.

4.4.1 Situación inicial: conexión por cable

En los primeros momentos del desarrollo de la aplicación, la comunicación entre la placa Micro:bit y el ordenador se realizaba por conexión USB. Esta conexión nos permite el inmediato funcionamiento de la placa, e incluso si contamos con alguna librería específica como Scratch.hex o Bitio, nos permite ahorrar en código y poder comunicar ambos dispositivos de manera directa.

Nos vimos obligados a descartar totalmente la idea de mantener el Micro:bit unido al ordenador por cable, principalmente pensando en la utilidad de la aplicación cuando esté terminada: las terapias. Dependeríamos de tener un cable USB compatible con ambos dispositivos lo suficientemente largo para poder adherir la placa Micro:bit a otros objetos del mundo real, pero aun así, estaríamos recortando la cantidad de movimientos que se podrían hacer de manera fluida con el, ya que puede haber tiranteces en los cables, o que entorpezca con las personas, por ejemplo.

Debido a que existían motivos suficientes para pensar que la comunicación por Bluetooth es la única solución viable, se intentó conocer cuál es el funcionamiento de esta tecnología, teniendo en cuenta que además en el caso de Micro:bit, tratamos con Bluetooth Low Energy. Esto condiciona la forma de trabajar a la hora de codificar, a la hora de la extracción de los datos, ya que tienen que ser "transportados", y a la hora de entenderlos, ya que estarán codificados de una manera óptima para minimizar el almacenamiento (recordamos que la capacidad de memoria de la placa Micro:bit es de 30 KBytes).

4.4.2 Aprendizaje de librerías de comunicación Bluetooth

Previo al desarrollo de esta fase, la comunicación Bluetooth era algo de lo cual no contaba con el conocimiento suficiente para hacer algo de estas características, por lo tanto, se incluyó

una pequeña fase de aprendizaje sobre los fundamentos del mismo. El primer paso es cómo conectar por Bluetooth el ordenador y la placa Micro:bit [20]. Para poder mantener activa la opción de emparejamiento de dispositivos de Micro:bit, tenemos que tener codificado nuestro código en lenguaje Scratch, y como se nos recomienda desde la documentación de la placa, hacerlo en el editor de MakeCode. Este editor online además nos permite la opción de enviar nuestro código a la placa Micro:bit sin necesidad de conectarnos por cable al ordenador.

Para poder continuar, debemos desarrollar el código que permita realizar todas las operaciones que posteriormente necesitemos e introducirlo de manera casi definitiva en la placa Micro:bit. Debido a este motivo, es importante conocer cómo se estructuran los datos dentro de la placa. Cada uno de los sensores que contiene está catalogado como un **servicio**, que contará con un identificador único con el cual nos podremos referir a él.

Nuestro código, por tanto, deberá activar estos servicios para que sus datos sean alcanzables, y de esa manera, podamos llamar a otros métodos que accedan a ellos y así recuperar los datos que necesitemos. Dicho esto, el código que incluiremos en la placa Micro:bit es el que podemos observar en la figura 4.2.

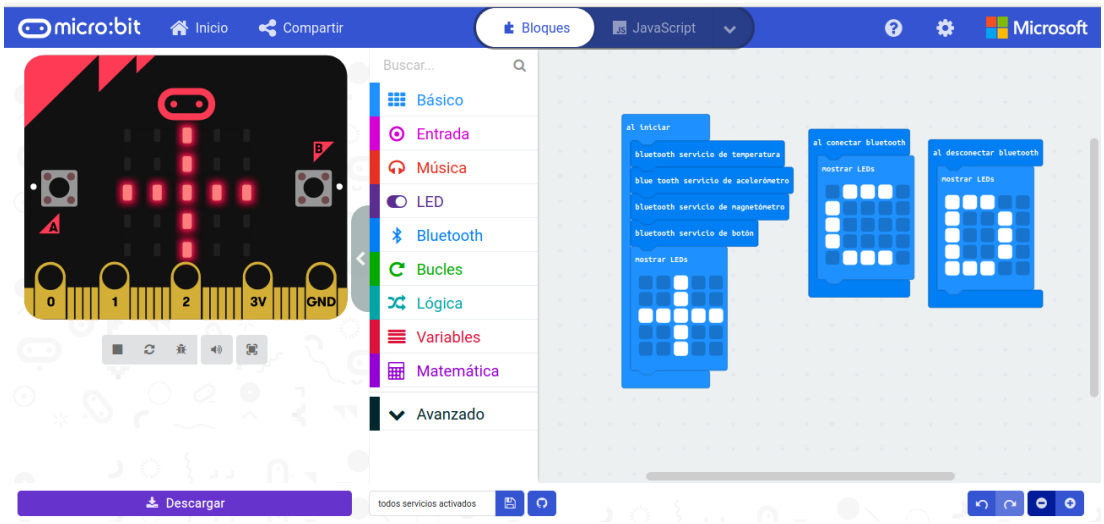

Figura 4.2: Código contenido en la placa Micro:bit para su comunicación con el ordenador

En este código podemos observar que tenemos tres bloques diferenciados, que son, al iniciar la placa Micro:bit, al establecer conexión y cuando nos desconectamos. En el momento de iniciar el programa, activamos todos los servicios necesarios, además de mostrar por el display una aspa que permitirá al usuario saber que el dispositivo está listo para usarse. Cuando nos conectemos, y mientras estemos conectados, cambiaremos la disposición de los leds para mostrar una letra C, que permitirá al usuario saber que la placa Micro:bit y el ordenador están conectados. En la situación de desconexión, mostraremos una letra D, con la intención de notificar que nos acabamos de desconectar de la conexión que tuviéramos activa.

En cuanto a la estructura de datos que tiene Micro:bit, ésta no se acaba con el sistema de servicios. Los servicios se componen, a su vez, de una serie de slots relacionados con el propio servicio. En estos slots podemos encontrar información como la frecuencia de actualización de los datos, números de serie, y los propios datos. Esta información se conoce como **características de los servicios** y son los puntos a los que debemos de acceder para obtener y extraer la información que necesitemos. Las características también tendrán un identificador único que nos permitirá acceder fácilmente a ellas, ahora debemos conocer con qué método lo haremos.

Lo único que nos falta es conocer los identificadores de los servicios y características que necesitemos para la aplicación. Para ello, existe un recurso, publicado por la Universidad de Lancaster en su página de Github [21] que contiene una descripción completa del perfil de Bluetooth de la placa Micro:bit, con los identificadores para todos los servicios y características que existen en ella.

4.4.3 Librería Bluepy (versión para GNU/Linux)

Las primeras fases de aprendizaje sobre librerías que permitan acceder a las características de un dispositivo, me llevaron a trasladar el desarrollo de esta parte a un entorno GNU/Linux para probar diferentes funcionalidades. En esta situación entra en juego el proyecto mencionado en la sección 2.3.3. Como se comenta en ese punto, este proyecto nos muestra varios fragmentos de código para incluir en la placa Micro:bit que nos permitan activar servicios para posteriormente extraer datos de sus características. En dicho proyecto se menciona que la manera más sencilla de leer los datos es a través de una librería Python llamada Bluepy (sección 2.2.2), otro proyecto similar que permite acceder a dispositivos que cuenten con tecnología Bluetooth Low Energy, como la placa Micro:bit.

En el proyecto Bluepy podemos encontrar algunos archivos que contienen código fuente en Python que nos pueden ayudar a guiar nuestro desarrollo, por ejemplo, este fragmento de código que nos permite descubrir dispositivos con el Bluetooth activo:

```
1  #!/usr/bin/python
2  from __future__ import print_function
3
4  from time import gmtime, strftime, sleep
5  from bluepy.btle import Scanner, DefaultDelegate, BTLEException
6  import sys
7
8
9  class ScanDelegate(DefaultDelegate):
10
11     def handleDiscovery(self, dev, isNewDev, isNewData):
```

```
12          print(strftime("%Y-%m-%d %H:%M:%S", gmtime()), dev.addr,
    dev.getScanData())
13          sys.stdout.flush()
14
15 scanner = Scanner().withDelegate(ScanDelegate())
16
17 scanner.scan(10.0, passive=True)
```

Con un ligero conocimiento de la librería, falta descubrir las funciones que nos permitan conectarnos, encontrar los datos y extraerlos, por ello, buscaremos más información relativa a estas necesidades en la documentación de la librería [22]. Es aquí donde encontramos la clase Peripheral, que contiene todos los métodos necesarios para las funcionalidades necesarias. Recopilando todo lo necesario, implementamos un pequeño código de prueba para comprobar que es posible con estos métodos extraer los datos tal y como los necesitamos:

```
1  # This Python file uses the following encoding: utf-8
2
3  from bluepy import btle
4  import struct,time
5
6  iterations = 0
7
8  p = btle.Peripheral("E9:0A:58:88:06:18", btle.ADDR_TYPE_RANDOM)
       #Método que nos permite establecer conexión
9
10 # Sin esta línea, falla la lectura de características
11 p.setSecurityLevel("medium")
12
13 svc_a = p.getServiceByUUID("e95d0753-251d-470a-a062-fa1922dfa9a8")
       #Accedemos a uno de los servicios
14
15 while True:
16   ch_acc =
       svc_a.getCharacteristics("e95dca4b-251d-470a-a062-fa1922dfa9a8")[0]
       #Accedemos a una de las características
17
18
19   testResult = struct.unpack('<hhh',ch_acc.read()) #Leemos y
       descodificamos los datos
20   print testResult
21   time.sleep(1)
```

En el perfil de Bluetooth que se comentaba antes, además de los correspondientes identificadores de servicios y características, también encontramos metadatos, que nos especificarán que contiene cada característica y que tipo de datos son, que en su mayor medida son unsigned

int en formato de 8 o 16 bits.

Con este código conseguimos una lectura exitosa, independientemente del identificador que tengamos que usar, y siempre teniendo en cuenta la traducción que haya que hacer del dato extraído, por lo que el siguiente paso sería adaptar este código para que pueda ser usado durante la ejecución de los juegos.

4.4.4 Librería Bleak (versión para Windows)

La librería Bluepy tiene un funcionamiento sencillo y que aporta los métodos que necesitamos, pero debemos descartarla por dos motivos principales: sólo funciona en sistemas Linux y no es compatible con la versión de Python instalada para el resto del desarrollo, ya que estamos usando la versión 3.7.5 y sólo es compatible para Python 2.7, 3.3 y 3.4.

Por tanto, debemos trasladar de vuelta el desarrollo a un sistema Windows, para que sea utilizable en los ordenadores que utilizan en ASPACE Coruña. Por suerte, encontramos una librería de funcionaiento muy similar a Bluepy pero para Windows, Bleak [23]. También es compatible con algunas versiones de otros sistemas operativos.

Bleak es una librería que ofrece una conexión asíncrona a través de código Python para comunicarse con objetos con Bluetooth Low Energy como esta placa Micro:bit. Su API contiene métodos muy similares a los vistos anteriormente en la biblioteca Bluepy, como podemos ver en este código útil para descubrir dispositivos. Este código también nos sirve para comparar ambos sistemas.

```
import asyncio
from bleak import discover

async def run(): #Importante que los métodos sean asíncronos
    devices = await discover()
    for d in devices:
        print(d)

loop = asyncio.get_event_loop()
loop.run_until_complete(run()) #Si tenemos varios métodos,
    indicamos cual ejecutamos de manera asíncrona
```

En el caso de Bleak, para acceder a las características no necesitamos acceder previamente al servicio, como podemos ver el siguiente fragmento de código, también extraído del proyecto en cuestión, que ha sido el utilizado para extraer los datos:

```
import asyncio
from bleak import BleakClient

address = "24:71:89:cc:09:05"
MODEL_NBR_UUID = "00002a24-0000-1000-8000-00805f9b34fb"
```

```
6
7  async def run(address, loop):
8      async with BleakClient(address, loop=loop) as client:
9          model_number = await client.read_gatt_char(MODEL_NBR_UUID)
       #Con este método leemos la característica
10         print("Model Number: {0}".format("".join(map(chr,
       model_number))))
11
12 loop = asyncio.get_event_loop()
13 loop.run_until_complete(run(address, loop))
```

4.5 Programación de juegos

Luego de haber conseguido las herramientas necesarias para comunicarnos por Bluetooth, necesitamos elaborar los escenarios para poder usar la información que extraemos de la placa Micro:bit, y ese será el siguiente paso, programar los diferentes juegos que utilizarán dichos datos para controlar los elementos de la pantalla.

Como se explicó en la sección 2.2.1, la librería Pygame, además de exponer una API muy sencilla, nos permite sustituir los valores extraidos de Micro:bit y sustituirlos en donde había interacción con el teclado.

En una primera etapa, para probar los juegos de una manera más cómoda, el control de los elementos en la pantalla se realiza con las flechas del teclado.

Dicho esto, comenzamos el desarrollo de los escenarios. Desde la dirección del proyecto, se plantean tres ideas: un juego consistente en recoger manzanas que caen de la parte superior de la pantalla con un cesto, otro que, controlando una nave espacial, dispare a unos meteoritos y, por último, un balón de baloncesto que hay que introducir en un agujero que se desplaza automáticamente por la pantalla.

De manera didáctica y como ayuda para la programación de los juegos, se ha seguido un tutorial online [24], a partir del cual, se ha sacado la idea para un juego a mayores para la aplicación, un clásico del arcade como es el Pong. Otra ventaja extraída del tutorial es usar una plantilla para el código, de manera que todos los juegos están estructurados de la misma manera y que el mantenimiento del código sea lo más sencillo posible. La estructura es la siguiente:

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3
4 # Módulos
5
6 # Constantes
7
```

```
# Clases
# ------------------------------------------------------------

# ------------------------------------------------------------

# Funciones
# ------------------------------------------------------------
# ------------------------------------------------------------

def main():
    return 0

if __name__ == '__main__':
    main()
```

En el apartado de módulos, iremos colocando todas las librerías que sean necesarias, entre las que tendremos que ubicar Pygame, Bleak o mysql.connector. En constantes lo único que reseñaremos es el tamaño de la ventana, anchura y altura. Las clases representarán los objetos con los que podemos interactuar durante una partida al escenario en cuestión: los sprites. Cada sprite contendrá una imagen y una serie de propiedades como la velocidad o la posición inicial. Es recomendable que las imágenes que usemos para los sprites tengan formato .png, de manera que no podamos ver el recuadro blanco que rodea muchas imágenes. Esto, por ejemplo, en el caso de la imagen de fondo no es necesario. A cada sprite habrá que añadirle una serie de físicas para sus movimientos, para que cuando nosotros lo decidamos, se mueva. Este tipo de interacción estará contenida dentro de unos métodos de la clase del sprite al que le corresponda. Otro aspecto a tener en cuenta dentro de este apartado es la colisión de elementos, para la cual Pygame nos ofrece una manera muy sencilla de comprobarla, siempre y cuando coincidan las coordenadas de los sprites en cuestión, o uno esté contenido dentro de otro.

Para el apartado de funciones, escribiremos aquí métodos de conveniencia que no corresponden a ningún sprite en particular, como una función para subir las fotos al juego, los métodos que implican usar la base de datos o la conexión por Bluetooth.

El método main contendrá el bucle principal de ejecución del juego, en el que se irán comprobando los valores para asegurarse de que se cumplen todas las reglas del juego, por ejemplo, que se compruebe la posición de un sprite para evitar que se nos salga de los márgenes de la pantalla. También debemos de incluir en este bucle la condición de finalización del juego, bien por tiempo o por pulsación de alguna tecla. Por último, contemplar que todos los elementos de la pantalla se actualicen en el orden que les corresponde, para evitar que haya bugs visuales durante el juego.

Pygame también permite introducir texto como un sprite dentro de la pantalla del juego.

Utilizaremos un método de conveniencia para configurar tamaño, fuente y lugar dentro de la imagen, para así poner cualquier texto que nos interese. Usaremos esta característica para la puntuación de los juegos.

Una vez todo está codificado, tenemos cuatro juegos totalmente funcionales con diferentes teclas del teclado. Cuando son sometidos a revisión por los directores del proyecto en colaboración con los fisioterapeutas de ASPACE Coruña, se llega a la conclusión de que uno de los juegos no es válido para el propósito. Se trata del juego de la nave espacial que dispara a los meteoritos ya que para utilizar todos los controles, no podemos usar únicamente la placa Micro:bit, principalmente para accionar el disparo. A raíz de esto surge el debate de si usar los botones como disparador de dicho evento, pero rápidamente es descartado al comprobar que el retardo de actualización del servicio de botones de Micro:bit es elevado y no es configurable.

El siguiente paso en el desarrollo, una vez tenemos los juegos desarrollados y funcionando, será introducir los datos necesarios para mover los elementos del juego con la placa Micro:bit únicamente, además de añadir la propia conexión Bluetooth.

4.6 Comunicación juego-Micro:bit a través de Bluetooth

Una vez hemos conseguido que nuestros juegos funcionen de manera correcta con el teclado como control, faltaba integrar la conexión y comunicación Bluetooth con el funcionamiento del juego. Ya tenemos escogidos los métodos que usaremos para este proceso, por lo que lo único que necesitamos es adaptarlos a la situación en la que estamos trabajando. Por suerte para el desarrollo, los tres juegos seleccionados son muy similares, y al haberlos desarrollado con la misma estructura, el código también es muy parejo, también usan los datos del acelerómetro, lo único que cambia en cada caso es que ejes están implicados.

Cabe mencionar que hay contenido en la característica del servicio de acelerómetro (el necesario para captar movimiento) una vez traducido el valor a un número entero. La característica de los datos del acelerómetro viene dividida en tres valores diferentes que se almacenan en un array, de manera que quedan distribuidos con el eje x en el primer espacio del array, el eje y en el segundo espacio, y el eje z en el tercero. El valor que contienen es un número entero entre -1024 y 1024, que simboliza, en el caso de los ejes x e y, la inclinación hacia los lados o hacia adelante o hacia atrás. En el eje z indica la orientación de la placa Micro:bit, si bien está mirando hacia arriba o hacia abajo.

El eje x nos indicará los movimientos captados en el eje horizontal, si inclinamos hacia la derecha, el valor irá aumentando, mientras que decrecerá si la inclinación es en la otra dirección. Para el eje y la situación es idéntica, salvo que la inclinación es en el eje vertical, si inclinamos hacia adelante, el valor correspondiente aumenta, y decrece si inclinamos hacia atrás.

Sabiendo esto, debemos de establecer un umbral a partir del cual hacer que el movimiento del Micro:bit se traslade a un movimiento dentro del juego. Tras hacer varias pruebas, se establece un primer umbral en 300 y -300, dependiendo de la dirección del movimiento.

Dicho todo esto, el método que nos permitirá mover los sprites será similar al que vemos en el siguiente fragmento. Se ha escogido el caso del juego de baloncesto ya que podemos observar movimiento en ambos ejes.

```
raw_data = await
    client.read_gatt_char("e95dca4b-251d-470a-a062-fa1922dfa9a8")
data = struct.unpack('<hhh',raw_data) #Traducimos los valores que
    recibimos.
#Valores -> 0: eje x, 1: eje y, 2: eje z
if self.rect.left >= 0:  # Comprueba que no se sale de la pantalla
        if data[0] < -300 :  # Aquí introducimos captura de datos
    por bluetooth (izquierda -> eje x < -100)
      self.rect.centerx -= self.speed[0] * time
if self.rect.right <= WIDTH:  # Comprueba que no se sale de la
    pantalla
    if data[0] > 300 : # Aquí introducimos captura de datos por
    bluetooth (derecha-> eje x > 100)
      self.rect.centerx += self.speed[0] * time
if self.rect.top >= 0: # Comprueba que no se sale de la pantalla
    if data[1] < -300 : #Nos movemos hacia arriba
      self.rect.centery -= self.speed[1] * time
if self.rect.bottom <= HEIGHT: # Comprueba que no se sale de la
    pantalla
    if data[1] > 300 : #Nos movemos hacia abajo
      self.rect.centery += self.speed[1] * time
```

En este código, previo al valor de umbral escogido, incluíamos la función de la librería que nos permitía acceder a los valores del teclado. Una vez sustituidos, el funcionamiento sigue siendo correcto y ya se adapta a las necesidades de los clientes.

Ya tenemos completamente codificados todos los juegos que se van a incluir en el proyecto y toda la comunicación por Bluetooth, únicamente falta añadir la gestión de usuarios y el diseño general de la aplicación que integrará todos los módulos comentados, además de la implementación del modelo de datos y de la interfaz gráfica.

4.7 Diseño de la aplicación

Al finalizar el desarrollo de los juegos, se convocó una reunión para comprobar que cumplían el propósito por el que fueron diseñados, lo cual se cumplió. En esa misma reunión, se expusieron unas cuántas funcionalidades a mayores que giran en torno a los juegos y sus re-

sultados. De esta manera, la terapia que los fisioterapeutas de ASPACE Coruña realizan con las personas con parálisis cerebral se hará en su totalidad con la aplicación.

A lo largo de este apartado, se expondrán las peticiones de los clientes y que decisiones motivaron a la hora de idear un modelo de datos y una interfaz gráfica, así como un código subyacente que realice todo aquello que se solicitó.

- **Gestión de usuarios:** La aplicación debe contener un sistema que permita crear, eliminar y modificar usuarios, que serán los responsables de utilizar los juegos y de los cuales los fisioterapeutas esperan obtener información útil. La idea más cómoda para este sistema es elaborar una lista que en cada fila contenga información de cada usuario, y una serie de funciones que nos permitan realizar las operaciones sobre los usuarios, así como acceder al contenido de cada uno. Para el código de esta parte, debemos confiar totalmente en las potencialidades que nos pueda ofrecer la librería que utilicemos para las interfaces gráficas, ya que será algo que implica únicamente el factor visual y la conexión a la base de datos.

- **Gestión de claves:** La aplicación debe estar organizada de la misma manera que lo están las instalaciones de ASPACE Coruña. Cuentan con cuatro servicios diferentes: Atención temprana, centro educativo, centro de día y centro residencial. De la misma forma, debemos contemplar estos cuatro servicios dentro de la aplicación, de manera que cuando entremos en cada uno de ellos, sólo veamos a los usuarios del mismo. Otro detalle a tener en cuenta es que para acceder a cada uno de ellos, debemos de introducir una contraseña propia de cada servicio, y que esta sea posible de recuperar si nos olvidáramos de ella.

  Para este tema, escogeremos realizar un menú en el que podamos visualizar todos los servicios al mismo tiempo, y que podamos escoger en el que queramos entrar mediante un botón, que al ser pulsado, nos pida la contraseña. Si nos olvidamos, nos permitirá recuperarla en la ventana que nos pide la contraseña. Esta opción activará un mecanismo que enviará un correo a una persona responsable de cada servicio, que se especificará en otro menú.

- **Existencia de un usuario administrador:** Se planteó primero la idea de que hubiera un usuario con permisos superiores, encargado de la gestión de usuario y de claves. Para acceder a este usuario, deberemos conocer su contraseña, que de manera idéntica a la de los servicios, es recuperable, salvo que en vez de enviar un correo a un responsable, lo envía a los responsables de los cuatro servicios, ya que, en teoría, esos cuatro responsables serían quienes usarían ese perfil administrador.

  Finalmente, se descartó la idea de que sea el administrador quien controle la gestión de usuarios, ya que son los encargados de cada terapia quienes están en contacto diario

(o más cercano) con la aplicación y los usuarios, de manera que si hubiera algún cambio, ellos son los que lo aprecian de manera más temprana. De esta manera, el usuario administrador que se ideó, es encargado únicamente de la gestión de claves, por lo tanto podrá cambiarla, además de poder cambiar el nombre y el correo electrónico de la persona encargada.

- **Datos de usuario:** Cuando se tenga la lista de usuarios, es necesario mostrar datos para que los usuarios sean reconocibles. Por ello se nos pidió que se vea el nombre del usuario, que terapeuta lo atiende y una imagen que permita reconocerlo rápidamente.

  Por ello, y gracias a las facilidades que nos aporta PySimpleGui, se añadió una línea con todos los widgets necesarios para mostrar estos datos por cada usuario que haya registrado. También debe contemplarse estos datos a la hora de añadir usuarios al registro. Para los nombres esto es algo trivial, pero en el caso de las fotos, se facilita una carpeta en la estructura de directorios de la aplicación para añadir ahí las fotos que se vayan a utilizar de perfil, y así diferenciarlas del resto de imágenes del programa y de la misma manera se hace un poco más sencillo el código que hará posible que sean visibles.

- **Estadísticas de usuario:** Se solicita que, con el fin de poder tener una trazabilidad de la evolución de los usuarios, haya un sistema de puntuaciones y estadísticas que nos permita analizarlas para poder extraer información útil para los terapeutas y que de esta manera puedan personalizar las terapias en función del desarrollo de las personas.

  Para esto, se hace necesario que los juegos envíen información a la aplicación de la puntuación y de parámetros importantes como la sensibilidad del movimiento, el tiempo de las partidas... Estos datos, además se mostrarán dentro de la aplicación en una ventana dedicada, donde cada usuario podrá ver un resumen de todas sus partidas, desde las más nuevas hasta las más antiguas, pudiendo analizar su puntuación en cada una de ellas y con que valores la ha realizado. La idea que se tiene para esta pantalla es similar a la lista de usuarios, sustituyendo usuarios por partidas.

- **Parámetros de partida:** Los parámetros que se comentaban anteriormente son parte fundamental del estudio que quieren realizar los terapeutas, por lo que se deben tratar por separados. La idea que nos hacen saber es que será necesario poder cambiar los diferentes valores que tengan la sensibilidad, la velocidad de los diferentes sprites, el tiempo y la puntuación umbral a partir de la cual la partida se considera una ”victoria”, en algún momento durante la partida, para agilizar las terapias sin tener que reiniciar toda la aplicación.

  Para poder realizar esto sin interrumpir ningún proceso clave, se decidió habilitar un botón durante la ejecución del juego que abrirá una pestaña donde se pueda ver los

parámetros actuales y cambiarlos si así se desea. Una vez introducidos, se parará el juego y se podrá volver a empezar con los nuevos datos.

- **Exportar datos a Excel:** Como venimos comentando, todos los datos extraídos de los juegos son útiles para los terapeutas, por lo que es interesante poder sacarlos de la aplicación a un archivo Excel para que se puedan analizar mejor y realizar gráficas o similares.

  Se decidió hailitar una opción dentro de nuestra ventana de estadísticas que permita extraer datos a partir de una fecha que seleccionemos, para evitar extraer todos los datos de golpe, ya que podría resultar en un archivo demasiado grande. De esta manera también será más fácil para el terapeuta agrupar los datos por etapas.

- **Posibilidad de cambiar de dispositivo:** El hecho de que la placa Micro:bit se estropee por cualquier causa es un problema que amenazaría gravemente toda la aplicación, se nos pide crear un sistema que haga que esto tenga el menor impacto posible.

  Se plantean dos ideas: solicitar la conexión a un dispositivo Bluetooth antes de iniciar un juego o tener una opción en algún menú para indicar a que dispositivo nos vamos a conectar. Finalmente nos decidimos por la segunda, ya que la primera entorpecería demasiado las terapias, ya que antes de jugar habría que realizar toda la configuración. Como sabemos del desarrollo anterior, en el código reconocemos los dispositivos a través de su dirección MAC, por lo que guardaremos en un archivo de configuración (que nos servirá en futuro desarrollo para almacenar más datos de configuración), ya que al ser un dato único no tendría sentido almacenarlo en una base de datos. La aplicación accede a este archivo, y cuando se inicien los juegos, se conecta al dispositivo que le hayamos indicado.

- **Detalles de accesibilidad:** Como esta aplicación está pensada para trabajar con personas con parálisis cerebral, debemos de tener en cuenta ciertos detalles que les hagan más fácil la interacción con la misma. Los fisioterapeutas de ASPACE Coruña sugieren que añadamos sonidos e imágenes en los juegos que permitan a los usuarios saber cuando lo están haciendo bien de manera visual o auditiva.

  La librería Pygame permite añadir sonidos, por lo que haremos eso mismo cuando se haga un punto dentro de las partidas o cuando se pierda la oportunidad de hacer uno, en diferentes tonos para que sean fáciles de reconocer, es decir, un tono alegre cuando se haga el punto, y uno más serio cuando no sea el caso. De la misma manera, añadiremos una ventana al final de cada partida, indicando la puntuación final de la partida, añadiendo una imagen amigable cuando se supere la puntuación umbral establecida, y una triste cuando no sea así.

Ya conocemos todos los detalles sugeridos para la aplicación y cómo se van a desarrollar, por tanto, lo único que nos falta ahora es realizar un modelo de datos acorde a la información relevante que necesitamos y una interfaz gráfica que utilice estos datos y nos permita realizar todas las funcionalidades.

4.7.1 Diseño del modelo de datos

Al estudiar toda la información necesaria para la aplicación, se ha analizado y se ha extraído un modelo de datos sencillo que unifique todos los datos y nos permita acceder a ellos sin perder información, relacionando todos aquellos que estén implicados en los mismos procesos. A continuación se muestra el modelo Entidad-Relación que ha resultado de analizar los datos necesarios:

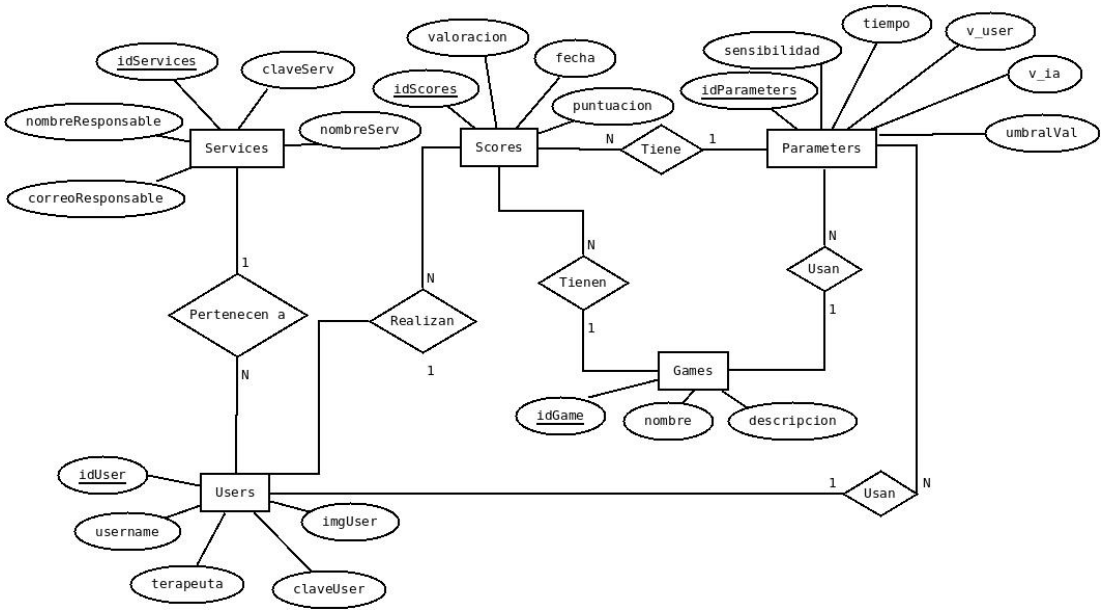

Figura 4.3: Modelo Entidad-Relación de la aplicación

En la imagen 4.3 podemos ver que nuestro modelo de datos cuenta con cinco entidades, que representan a grandes rasgos los diferentes módulos de la aplicación. Ya que el programa no está diseñado para ser excesivamente complicado, podemos agrupar todos los datos por los diferentes módulos sin problemas aparentes.

En primer lugar, nos encontramos con la entidad que representa los juegos, **game**, que consta de un identificador, el nombre del juego en cuestión y una pequeña descripción del juego. Este último atributo, aunque en esta versión de la aplicación no se utiliza, en el futuro nos serviría para añadir un pequeño apartado de ayuda para cada uno de los juegos, explicando a grandes rasgos en que consiste cada uno. A esta entidad no se puede acceder a través de la

aplicación, debido a la complejidad que entraña por el momento el hecho de añadir nuevos juegos a la aplicación. En un momento posterior del desarrollo podríamos llegar a plantearnos si añadir esta funcionalidad, lo cual haría necesario acceder a esta entidad. Además, nos sirve para referirnos a que juego corresponde una partida o unos parámetros en particular.

Por otra parte, el otro pilar fundamental de la aplicación son los usuarios, entidad **user**. Los usuarios cuentan con un identificador, que se usará para enviar información de un módulo a otro dentro de la aplicación, un nombre de usuario para reconocerlos, el terapeuta a cargo de el usuario en cuestión, el servicio de ASPACE al que pertenecen, una clave de usuario y su imagen de perfil. El hecho de que exista una clave de usuario es por el caso particular del usuario administrador, un usuario especial dentro de esta tabla, cuyos datos serán totalmente intrascendentes a excepción de esta clave. Será el primer elemento contenido en esta tabla, por lo que para el resto de casos de uso que necesiten esta tabla, siempre se accederá a partir del segundo elemento.

En cuanto a los servicios, la entidad **services**, constituyen las diferentes áreas de trabajo que siguen en ASPACE, siendo estas Atención Temprana, Centro Educativo, Centro de Día y Centro Residencial. Cada una de ellos tendrá, además de su nombre e identificador, una clave para poder acceder a la información sobre los mismos, un nombre de un terapeuta responsable del mismo, y un correo electrónico, perteneciente al responsable. Esto último existe para la situación en la que necesitemos recuperar la contraseña del servicio, en caso de que esto suceda, la aplicación mandará un correo electrónico a la dirección que hayamos especificado.

Ahora pasamos a estudiar las dos entidades centrales del modelo, ya que son las que relacionan los juegos y los usuarios, y son aquellas que contienen la información útil para el estudio de los usuarios, siendo estas entidades **scores** y **parameters**.

La primera de las dos representa las puntuaciones de cada una de las partidas que se realizan en los diferentes juegos de la aplicación, guardando información sobre quién, cuándo y cómo lo ha hecho. Se referencian el usuario que lo ha hecho, en que juego y con que parámetros, además del momento de finalización de la partida, la puntuación y que valoración ha obtenido en función de la puntuación umbral. En cuanto a los parámetros, tal y como mencionábamos antes, almacenamos la sensibilidad de captura de movimiento, el tiempo de la partida, la velocidad del sprite controlado por el usuario, la velocidad del sprite controlado por el juego y la puntuación umbral. Además, se indica a que usuario pertenece estos parámetros y sobre que juego se aplican. Cabe mencionar que al añadir un usuario al sistema, se introducen una serie de parámetros por defecto para que no haya ningún tipo de problema al iniciar los juegos, ya que los parámetros no son modificables sin haber accedido al juego en cuestión.

4.7.2 Diseño de la interfaz gráfica de la aplicación

Desarrollaremos este apartado de manera similar al anterior, exponiendo los diferentes mockups propuestos por los terapeutas de ASPACE Coruña para la aplicación y desarrollando la transformación hasta la idea final incluida en el programa. Las imágenes presentadas a continuación fueron proporcionadas por los fisioterapeutas de ASPACE Coruña, a modo de guía de cómo quieren que sea la aplicación, por lo que no son pantallas idénticas a las que pretenden tener, sino algo similar, que sirva como ejemplo de algo que les parezca oportuno.

La aplicación cuenta con un total de seis pantallas distintas, que agrupan todas las funcionalidades descritas hasta el momento, y estas son: la pantalla de inicio, pantalla de servicios como administrador, pantalla de servicios como usuario, lista de usuarios de cada servicio, lista de juegos y estadísticas. Los juegos no se tratan como pantallas ya que su desarrollo no necesita de interfaz gráfica, además de que son archivos totalmente externos al resto de la aplicación, es decir, se pueden usar de manera independiente.

La pantalla inicial (figura 4.4) propuesta por ASPACE Coruña es muy simple y proponen que cuente con la imagen corporativa de todos los organismos implicados en el proyecto. En ella se incluyen, además de la imagen, mecanismos para acceder a la siguiente pantalla de la aplicación. También se decidió incluir aquí la opción de cambiar la dirección MAC del dispositivo usado para controlar los juegos, de manera que no haya más interrupciones a lo largo de la aplicación en cuanto a este tema.

Figura 4.4: Mockup proporcionado por ASPACE Coruña a partir del cual se desarrolló la pantalla inicial

La pantalla inicial puede comunicar con dos diferentes en función del usuario que la vaya usar, administrador o usuario normal. Si entramos como administrador, nos pedirá la clave, y en caso de acceder nos mostrará información sobre los servicios, que podremos cambiar. En el caso de entrar como usuario normal, veremos los servicios y se nos permitirá acceder a ellos.

En ambos casos (figura 4.5)[1], la estructura será la misma, ya que veremos los cuatro iconos de los servicios con un botón que nos permitirá la acción que corresponda en cada caso.

Figura 4.5: Mockup proporcionado por ASPACE Coruña a partir del cual se desarrollaron las pantallas de servicios

Cuando entramos a la lista de usuarios, debemos ver los datos de cada uno de ellos, y también debemos contemplar que debemos incluir elementos de navegación, es decir, ir a la pantalla anterior o acceder a cada uno de los usuarios (esto se detallará más adelante), sin olvidarse de las opciones de añadir, modificar o eliminar los usuarios. En la figura 4.6 podemos ver un ejemplo de cómo se nos pide que sea esta pantalla.

Figura 4.6: Mockup proporcionado por ASPACE Coruña a partir del cual se desarrolló el listado de usuarios

Al acceder a un usuario en particular, debemos de ver una lista con los tres juegos seleccionados para la aplicación, y esto debe ser algo similar a lo reseñado en la figura 4.7. Esta

[1]En algunos mockups se especifica que el control es por barrido, esta idea fue descartada debido a la complejidad que supondría introducirla.

pantalla también debe dar acceso a las estadísticas de cada usuario, además de las opciones de navegación que venimos comentando hasta ahora. La pantalla también contendrá una imagen que nos permita reconocer el juego, además de un botón con el cual lo activemos y podamos jugarlo.

Figura 4.7: Mockup proporcionado por ASPACE Coruña a partir del cual se desarrolló el listado de juegos disponibles

Para concluir, falta tratar la página en la que veremos todas las estadísticas (figura 4.8). Un detalle importante para esta página es poder filtrar los datos por juegos. Será una página bastante similar al listado de usuarios, ya que consiste en lo mismo, mostrar los datos de cada partida, pudiendo acceder a un menú para cada una de las partidas que nos mostrará los parámetros con los que se jugó. Además, debemos de ubicar un elemento que nos permita volver atrás y otro que nos permita exportar los datos a Excel.

Figura 4.8: Mockup proporcionado por ASPACE Coruña a partir del cual se ha desarrollado la pantalla de estadísticas

4.8 Implementación de la aplicación

Una vez realizado el diseño de toda la aplicación, se pasó a realizar la implementación del modelo de datos a partir del modelo relacional presentado en la sección 2.2.3 y el desarrollo de la interfaz gráfica a partir de las especificaciones presentadas en la sección 4.7.2.

4.8.1 Implementación del modelo de datos

Una vez tenemos un modelo de datos completo, falta traducirlo al modelo relacional para poder aplicarlo en nuestro programa. Como mencionábamos en la sección 2.2.3, usaremos una base de datos MySQL, que cuenta con un editor especial para la manipulación de la base de datos, MySQL Workbench, el cual usaremos para gestionar la base de datos que tenemos que crear.

Siguiendo los pasos para convertir un modelo Entidad-Relación en un modelo relacional correcto, obtenemos las tablas que podemos ver en la figura 4.9:

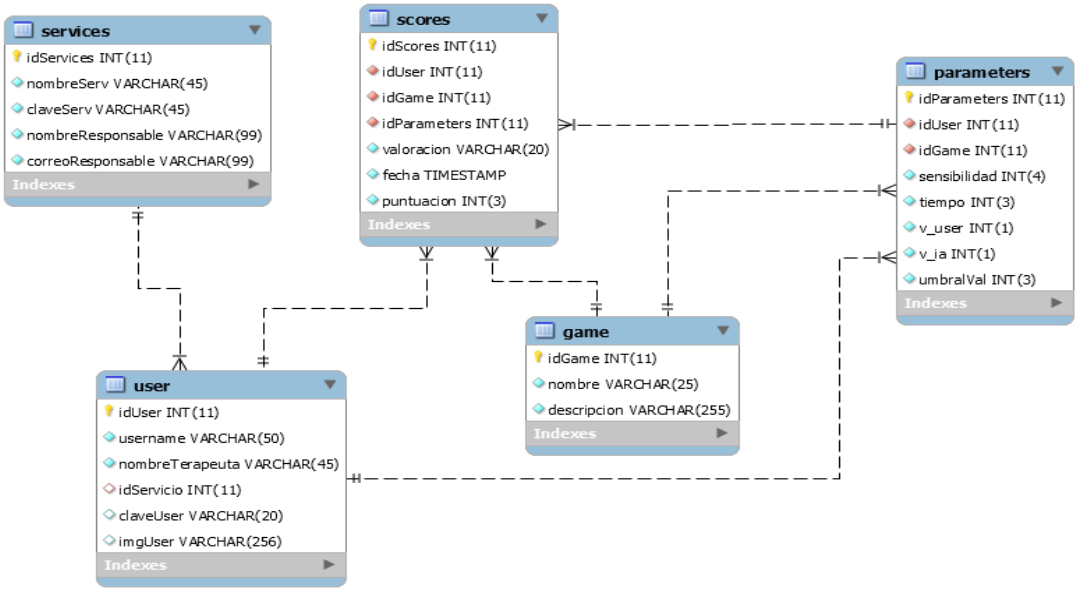

Figura 4.9: Modelo relacional de la aplicación

Mediante el conector de MySQL para Python, se puede realizar llamadas directamente desde nuestro código a la base de datos. En el apartado 4.3.1 hablamos de la estructura que utilizamos para acceder a la base de datos. Con las tablas tal y como las tenemos ahora mismo, los datos que necesitamos obtener son totalmente accesibles y utilizables en la aplicación, y eso explicaremos ahora, cómo son utilizados estos datos dentro de los diferentes módulos que tenemos. Vamos a resumir las principales llamadas a la base de datos que se hacen para resumir cómo es este proceso en la aplicación.

- Para la pantalla de servicios, para la parte de administrador, necesitaremos obtener los datos de configuración de cada servicio, así que simplemente hacemos un SELECT que extraiga el identificador, la clave del servicio, el nombre del responsable y su correo electrónico. Podríamos obtener también el nombre de cada servicio, pero al ser algo inmutable, podemos ahorrarnos esta llamada y escribirlo directamente en la interfaz gráfica. Como detalle, de ahora en adelante, para obtener los datos de la consulta, deberemos de llamar a uno de los métodos del cursor que creemos para acceder a ellos: cursor.fetchone(), cursor.fetchall() o cursor.fetchmany(). Una vez hecho esto, podremos usar nuestros datos como variables de la aplicación y usarlo en la aplicación. El identificador del servicio lo usaremos para mandárselo a la siguiente pantalla, para que allí sepamos a que servicio nos tenemos que referir.

- A continuación, tendremos que mostrar la lista de los usuarios de cada servicio. Ya que hemos traído el identificador de servicio de la pantalla anterior, tendremos que hacer una consulta de los usuarios que cumplan la condición de tener un identificador de servicio igual al que recibamos de la pantalla anterior. Los usuarios deberán reconocerse por su imagen, nombre y nombre del terapeuta, por lo que esos serán los datos que tenemos que pedir a la base de datos. En el siguiente paso, tenemos que tratar los casos de añadir, modificar y eliminar usuarios. Son casos triviales, ya que en el caso de añadir, tendremos que hacer una sentencia INSERT INTO, cubriendo todos los datos de la tabla, cuando los modifiquemos, haremos un UPDATE con aquellos datos que se pueden modificar (es decir, los que podemos ver), y en el caso de la eliminación, para facilitarlo, haremos un DELETE FROM a partir del identificador del usuario que se seleccione.

- Una vez accedamos a un usuario, se mostrará la pantalla de selección de juegos, la cual no requiere llamadas a la base de datos, ya que estos valores son inmutables en principio, pero si necesitaremos conocer el identificador de cada juego para enviárselo al propio juego para que este pueda rellenar los datos de cada partida (además del identificador de usuario), o bien para seleccionar las puntuaciones que queremos ver.

- Los juegos deben cubrir las puntuaciones al acabar cada partida, además de mostrar y almacenar parámetros, por lo que tienen una carga bastante pesada de operaciones contra la base de datos. En primer lugar, recibimos el identificador del usuario y del juego, de manera que podremos hacer un SELECT que nos permita recuperar los parámetros de juego y aplicarlos en la partida en curso. Cuando estemos en situación de cambiar los parámetros, los que estén en uso deben de mostrarse, a la vez que se actualizarán mediante un UPDATE en la base de datos cuando hayamos acabado. Para almacenar los datos de la partida, necesitaremos conocer el usuario, el juego y los parámetros, además de los datos propios del resultado de la partida, como son la valoración, los puntos y la

fecha. Los puntos los calcula el juego automáticamente, mientras que los otros datos los podemos generar fácilmente mediante una única línea de código cada uno. Finalmente, haremos un INSERT INTO con todos estos datos para almacenarlos.

- Por último, para mostrar las estadísticas, deberemos de acceder a la tabla **scores**, de la cual mostraremos la fecha, la puntuación, la valoración y los parámetros de la partida. Para ello, necesitaremos el usuario en cuestión y el juego del que queremos ver las puntuaciones, que obtenemos de la misma manera que en el juego, desde la pantalla anterior. Una vez más, hacemos una consulta que nos permita obtener todos los datos, siendo esta un poco más compleja, ya que tenemos que realizar un JOIN para obtener únicamente los parámetros de cada partida, y no siempre los mismos.

Ya tenemos los métodos que nos permiten acceder a todos los datos necesarios para que la información mostrada en la aplicación sea clara y que a su vez, la aplicación sea fácil de usar gracias a estos datos. Para finalizar, en la siguiente sección, comentaremos cómo con estos datos y las ideas provenientes del diseño previo, obtenemos la interfaz gráfica final de la aplicación.

4.8.2 Desarrollo de la interfaz gráfica

Esta sección trata el último paso realizado durante el desarrollo, que fue la elaboración de una interfaz gráfica que facilitara el uso de la aplicación. Se han creado un total de 8 pantallas, contando con los juegos y los parámetros. A continuación hablaremos de cómo están formadas dichas pantallas y que funcionalidades permiten hacer y cómo. Además aprovecharemos este apartado para comentar la imagen de los juegos y cómo funcionan.

Cuando iniciamos la aplicación, aparecerá una pantalla que muestra la imagen corporativa de las entidades que han colaborado durante este proyecto, así como la posibilidad de acceder a la aplicación como administrador o como usuario normal, tal como podemos ver en la figura 4.10. Además desde esta pantalla, podremos cambiar la MAC del dispositivo que tendremos conectado por Bluetooth, y al cual se intentará conectar automáticamente según arranquen los juegos. Es una pantalla muy sencilla, ya que cuenta con un menú en la barra superior, una imagen y tres botones, que accionan otros archivos que ejecutan las pantallas siguientes.

Como mencionamos, en esta pantalla se permite cambiar la MAC del dispositivo Bluetooth en el menú Configuración en la parte superior de la ventana. Una vez entremos en esta opción, se abrirá una ventana que permitirá introducir la nueva dirección, tal como podemos observar en la figura 4.11. En caso de necesitarse, en el mismo menú existe una opción de ayuda con los pasos a seguir. La dirección MAC se almacena en un fichero de configuración independiente, que en el futuro se puede emplear para otros datos que sean independientes del resto de datos, como es este.

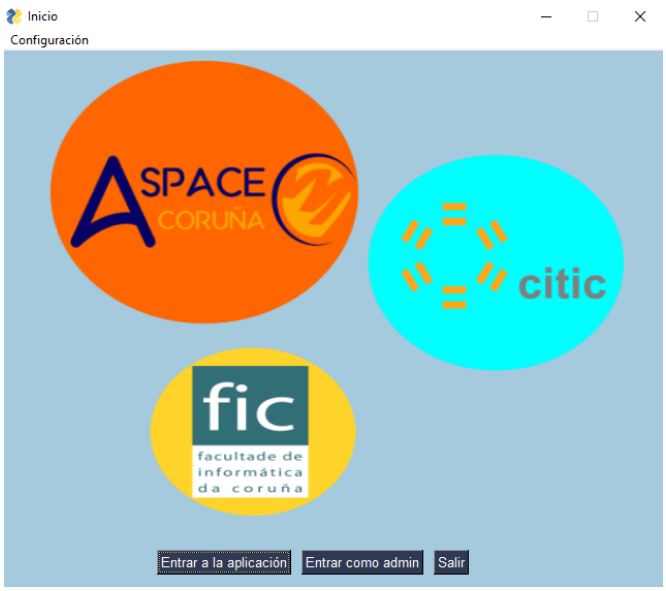

Figura 4.10: Pantalla inicial de la aplicación

El paso siguiente será pulsar uno de los botones que nos permita acceder al resto de la aplicación. Si entramos como usuario normal, se cargará directamente la siguiente pantalla. Al mismo tiempo, se abrirá una terminal de comandos en segundo plano, que servirá como log de errores. Esto se debe a que el código que permite la interacción entre pantallas, os.system(« comando de terminal cmd ») ejecuta los comandos que le mandemos a través de la propia terminal.

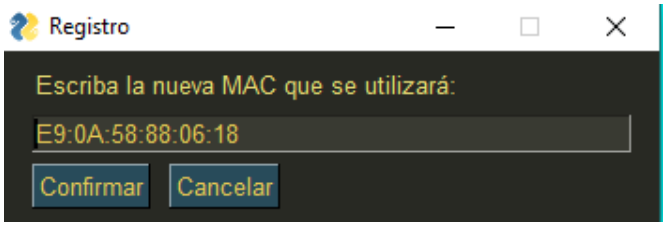

Figura 4.11: Posibilidad de cambiar la dirección MAC

Si elegimos la opción de entrar como administrador, se nos pedirá la contraseña (figura 4.12). Al mismo tiempo, se ofrece la posibilidad de recuperarla si no nos acordamos de ella. Este botón conecta con un fichero que ejecuta una serie de métodos que envían un correo a las direcciones que tengamos habilitadas como responsables de los diferentes servicios de ASPACE Coruña. Lo mismo pasará más adelante con las claves de los propios servicios, salvo que el correo se enviará al responsable del servicio del cual queramos recuperar la contraseña únicamente.

Si la introducimos de manera correcta, veremos los cuatro servicios de ASPACE Coruña,

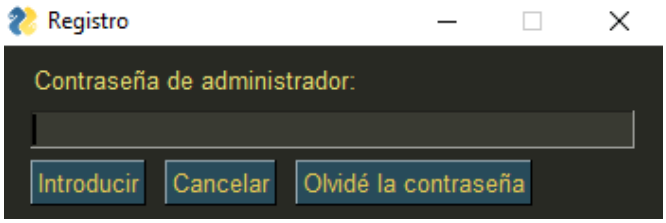

Figura 4.12: Introducir clave de administrador

con los pictogramas que los identifican, en dos filas de dos elementos cada uno. Debajo de cada pictograma encontraremos un botón, el cual nos permitirá modificar la configuración de cada uno de los servicios, cambiar la clave, el responsable, y el correo del mismo. Todo esto, lo podemos observar en las figuras 4.13 y 4.14.

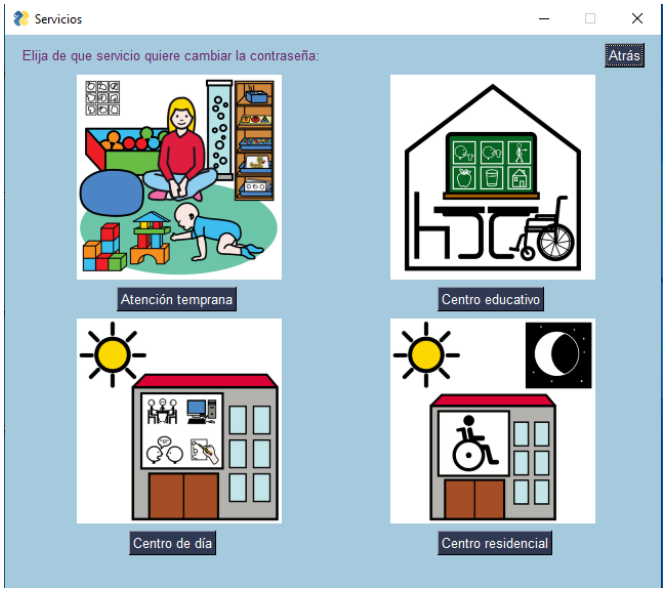

Figura 4.13: Pantalla de servicios para el administrador

Si entramos en la aplicación como usuario normal, la pantalla que veremos es prácticamente idéntica a la figura 4.13, pero en este caso, cuando hagamos clic sobre uno de los botones que corresponden a los servicios, se nos pedirá la contraseña del servicio para poder acceder a la lista de los usuarios. Dicha lista, como ya hemos mencionado en otros apartados, consta de una imagen y dos recuadros de texto donde irán el nombre del usuario y el del terapeuta que lo atiende. Al lado de esta información encontraremos un botón que permite acceder al menú de selección de juegos de cada usuario, además de contar con tres botones en la parte inferior que permitirán añadir, modificar y eliminar usuarios (figura 4.15).

Esta página quizás sea la que más complejidad entraña, debido a la necesidad de crear una línea para cada usuario, sin saber de cuantos usuarios hará falta información. Para ello

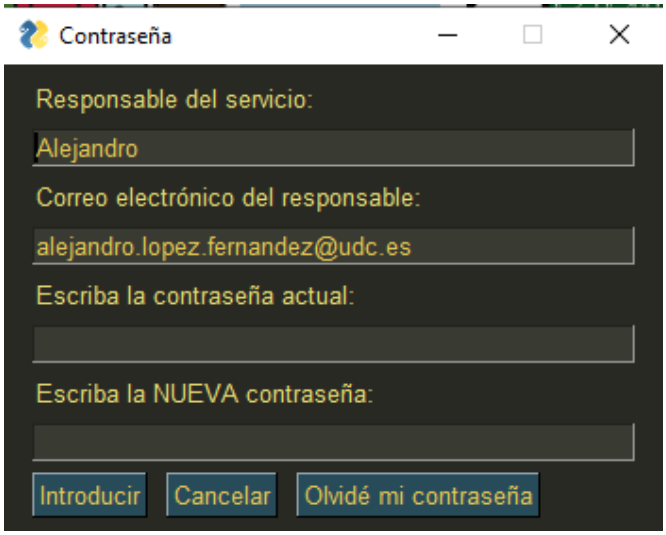

Figura 4.14: Configuración de los servicios

Figura 4.15: Lista de usuarios

usaremos el siguiente bloque de código:

```
1  layout = [
2        [sg.Text(" "*46),sg.Text("Nombre",font = 'Arial 15
     bold'),sg.Text(" "*19),sg.Text("Terapeuta",font = 'Arial 15
     bold')],
3       ]
4  for i in range(0,times):
5    if data:
6      if data[i][2] == None:
7        imagen = "images/images/nophoto.jpg"
8      else:
9        imagen = data[i][2]
10     layout += [sg.Text(" "*10),sg.Image(data=get_img_data(imagen,
     first=True)),sg.Text("
     "*7),sg.InputText(data[i][0],size=(15,1),disabled = True,
     justification='center'),sg.Text("
     "*14),sg.InputText(data[i][1],size=(15,1),disabled = True,
     justification='center'),sg.Text(" "*8),sg.Button("Entrar")],
```

Para resumir, tenemos un layout que conforma la vista de la ventana, que consta con los títulos de nombre y terapeuta en las columnas que corresponde para que quede alineado. A mayores, debemos extraer de la base de datos el número de elementos que hay en la tabla de usuarios (variable ”times”). Sabiendo esto, si la variable data (info de usuario) contiene algo, crearemos una línea para esa información. En cuanto a las líneas 6-9, sirven para comprobar si se le ha asignado una imagen al usuario. En caso contrario, se asigna una por defecto.

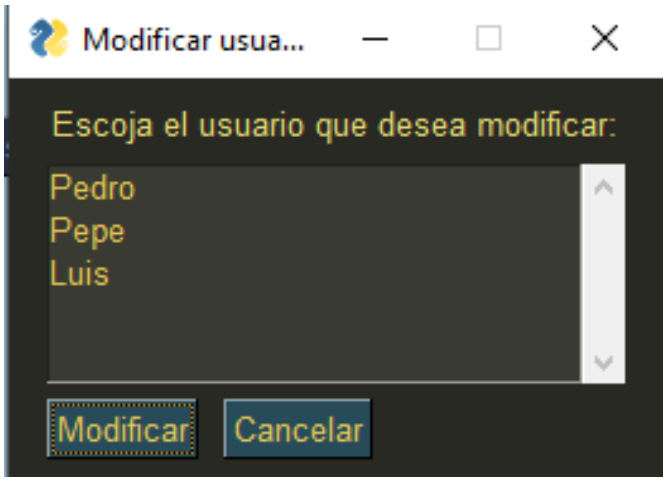

Figura 4.16: Lista de usuarios para modificar o eliminar usuario

En cuanto a los botones, debemos de mencionar que la primera idea fue añadir botones de modificar y eliminar para cada usuario, en su propia línea. Esto sin embargo no se puede

realizar sin aumentar la complejidad del código, por la manera que tiene la librería de nombrar los botones que tienen el mismo nombre, ya que si, por ejemplo, tenemos dos botones que se repiten (BotonA y BotonB), la segunda ocurrencia de los mismos los nombrará como BotonA0 y BotónB1, y así sucesivamente, lo cual nos hace complicado relacionar cada botón con un identificador de usuario. Por este motivo, cuando queramos modificar o eliminar un usuario, deberemos de seleccionarlo de una lista auxiliar que se desplegará al pulsar el botón correspondiente. Para ambos casos la lista es idéntica (figura 4.16). En el caso de que queramos añadir un usuario, se nos mostrará una pantalla en la que rellenaremos los datos necesarios para ser mostrados. En cuanto a la imagen, para evitar desajustar la estructura de los datos en la interfaz, deberá tener unas dimensiones iguales o superiores a 75x75 píxeles. La aplicación bloqueará cualquier intento de añadir una imagen más pequeña que esas dimensiones.

Si finalmente accedemos a uno de los usuarios, veremos la selección de juegos que ofrece la aplicación (figura 4.17). Como vemos, tenemos tres juegos, consistentes en atrapar manzanas con un cesto, introducir un balón de baloncesto en un agujero y jugar al ping pong. Los tres juegos se distribuyen en la pantalla en dos filas, que contienen una imagen que los identifica y un botón que nos permite jugarlos. Además, esta pantalla contiene un botón en la parte inferior derecha que permite acceder a las estadísticas de cada uno de los juegos. Tendremos que seleccionar de que juego queremos ver las estadísticas para acceder (figura 4.18).

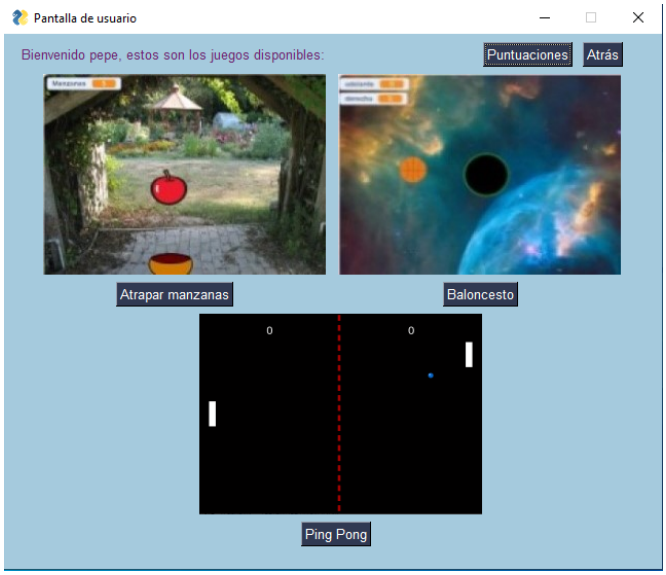

Figura 4.17: Pantalla de selección de juegos

Si hemos seleccionado uno de los juegos para ver sus estadísticas, nos encontraremos una pantalla como la de la figura 4.19. Con el mismo funcionamiento que en el caso de los usuarios, mostramos la fecha de la partida, la puntuación, y si el umbral seleccionado por el terapeuta fue superado o no. Al lado de estos datos, encontraremos un botón que nos mostrará

Figura 4.18: Selección de juego para ver estadísticas

los parámetros con los que se jugó la partida. Tanto la pantalla de usuarios como esta, cuentan con la posibilidad de desplazarnos verticalmente por la pantalla en caso de que haya múltiples ocurrencias de los datos.

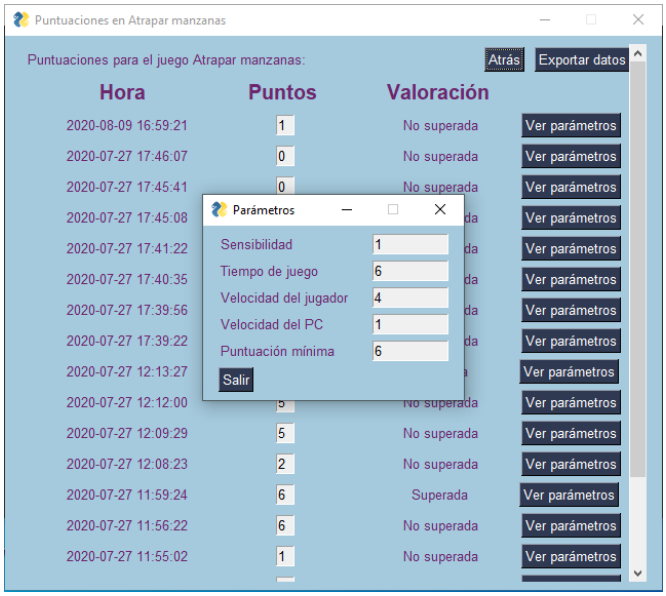

Figura 4.19: Estadísticas de las partidas jugadas junto con los parámetros de partida

Como se indicó anteriormente, en esta pantalla se permite exportar los datos que vemos en pantalla a formato Excel. Para ello, pulsaremos el botón que está en la esquina superior derecha, y nos hará falta seleccionar una fecha a partir de la cual queremos exportar, para evitar tener que meter todos los datos de golpe (figura 4.20).

Por último, nos queda analizar la imagen de los juegos. Desde la pantalla de selección de juego, si pulsamos cualquiera de los botones accederemos directamente al juego en cuestión. Funcionan de la siguiente manera:

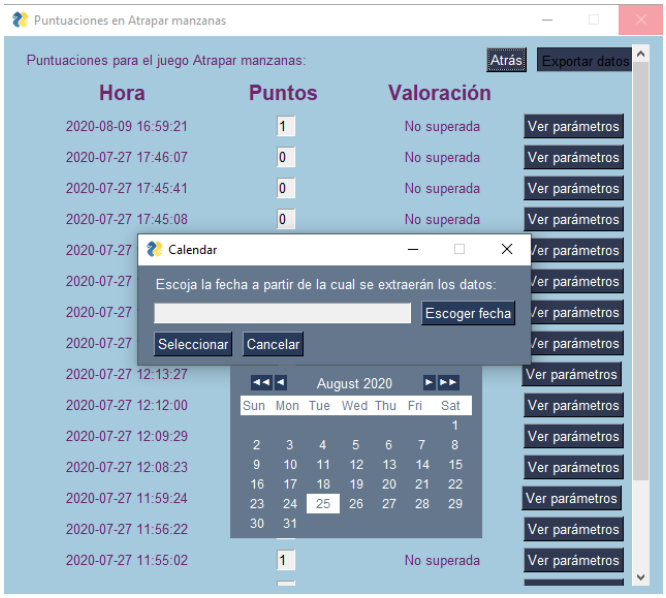

Figura 4.20: Pantalla de exportación de datos

- **Atrapar manzanas:** Consiste en desplazar una cesta para recoger las manzanas que caen desde la parte superior de la pantalla. Inclinando la placa Micro:bit en el eje horizontal, moveremos la cesta. Se consigue un punto atrapando la manzana que cae. Sonará un tono alegre si esto sucede, y en caso contrario, un tono de error (figura 4.21).

- **Baloncesto:** Trata de introducir un balón de baloncesto en un agujero que se posicionará por la pantalla de forma aleatoria. Cada vez que lo introduzcamos, el agujero cambiará su ubicación, sonando un tono alegre cada vez que esto suceda (figura 4.22).

- **Ping Pong:** Partida de ping pong al estilo arcade. Si conseguimos que el rival no devuelva la pelota, sonará un tono alegre y se nos sumará un punto, si es el usuario quién no la devuelve, sonará un tono de error (figura 4.23).

Una partida puede terminar de tres maneras diferentes: pulsando sobre la cruz de la ventana, pulsando el botón ESC o cuando se acabe el tiempo de la partida. Si pulsamos la cruz, el juego se terminará sin más, sin guardar la puntuación. En caso de pulsar el botón ESC, podremos acceder a la modificación de parámetros del juego en cuestión (figura 4.24), que una vez almacenados, se cierra el juego. Cabe mencionar que dicha pantalla cuenta con una opción de ayuda que describe los parámetros y los valores que acepta, ya que el código cuenta con validación para evitar valores absurdos. Finalmente, si dejamos que se acabe el tiempo, nos encontraremos una pantalla con el resultado de la partida, que mostrará la hora, la puntuación y unos dibujos alegres si hemos superado el umbral o tristes si no es así (figura 4.25).

Con esto concluimos el repaso al funcionamiento y apariencia de la interfaz gráfica desa-

Figura 4.21: Juego de atrapar manzanas

Figura 4.22: Juego de baloncesto

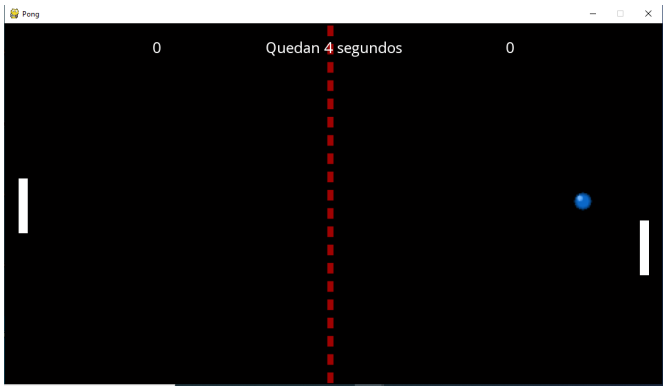

Figura 4.23: Juego de ping pong

rrollada para la aplicación y el desarrollo de la misma. Hubo una fase final de revisión de errores por toda la aplicación, pero que no merece mención a mayores, ya que fue retocar el código ya existente y comentado en estos apartados.

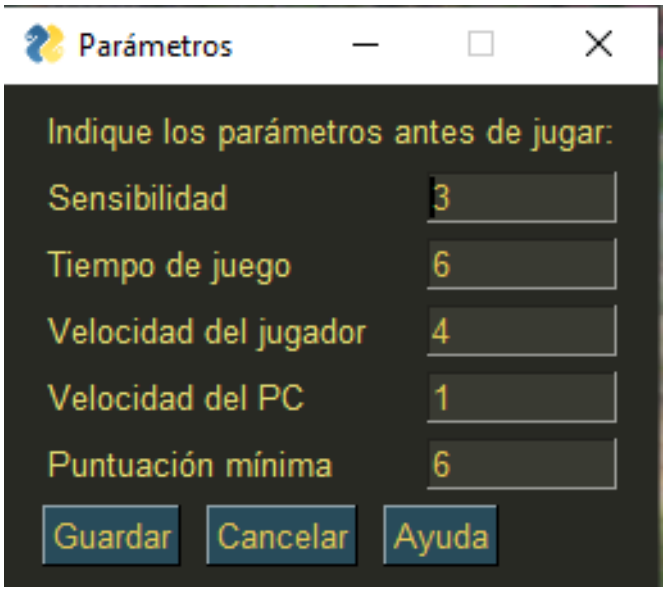

Figura 4.24: Elección de parámetros para la partida

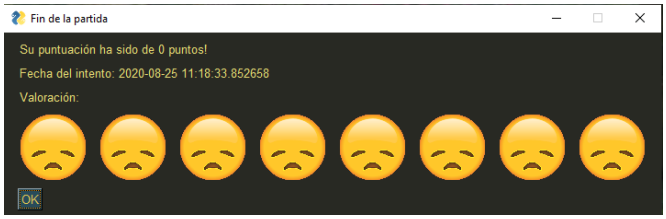

Figura 4.25: Resumen de partida

Capítulo 5

Pruebas con usuarios reales

El día 27 de julio de 2020 se realizó, en las instalaciones de ASPACE Coruña, una sesión de pruebas de la aplicación con usuarios reales, programada en una reunión anterior con los propios fisioterapeutas. La complicada situación sanitaria vivida durante el desarrollo del proyecto provocó que no se pudiesen realizar todas las pruebas deseadas, pero los resultados obtenidos durante la sesión han sido fructíferos tanto para el desarrollo del proyecto como para que los fisioterapeutas pudieran comprobar cómo adaptar las sesiones de terapias mediante el uso de la aplicación. Es conveniente mencionar que, aunque el proyecto se encontraba en una fase final del desarrollo, no estaba completamente terminado, y la retroalimentación obtenida de las pruebas fue importante para finalizar la revisión de todos los módulos que se estaba realizando en ese momento.

En este capítulo se describe el desarrollo de la sesión de pruebas y los principales resultados obtenidos.

5.1 Descripción de la sesión

Durante la sesión de pruebas han participado siete personas con diferentes roles, pertenecientes a ASPACE Coruña y al equipo de desarrollo del proyecto.

Podemos diferenciar los roles dentro de la prueba como:

- Rubén Carneiro Medín. Colegiado nº: 1841. Fisioterapeuta que trabaja en el ámbito de la Neurorehabilitación Pediátrica. Especialista en neurorehabilitación intesiva con órtesis Spidersuit. Actualmente desarrolla su trabajo con niños con Parálisis Cerebral y alteraciones del neurodesarrollo. Coordinador del proyecto Jóvenes Tech-In para una tecnología inclusiva que desarrolla ASPACE Coruña conjuntamente con Citic (Universidade da Coruña) financiado por la Fundación Española de Ciencia y Tecnología.

- Laura Quiroga Andrade. Colegiada nº: 2813. Fisioterapeuta que trabaja en el ámbito de la neurorehabilitacíon en personas adultas con parálisis cerebral y alteraciones afi-

nes. Desarrolla su trabajo en el Centro Residencial y Centro de Día ASPACE Coruña. Responsable de nuevas tecnologías en Centro de día ASPACE Coruña.

- Dos usuarios con distinto grado de parálisis cerebral medido con dos sistemas: sistema de la clasificación de la función motora gruesa (GMFCS) para la parálisis cerebral está basado en el movimiento auto-iniciado por el paciente con énfasis en la sedestación (control del tronco), las transferencias y la movilidad [25] y el sistema de clasificación de la habilidad manual (MACS) que describe cómo las personas con parálisis cerebral usan sus manos para utilizar objetos de uso cotidiano[26].

  Usuario 1. Usuaria residente en ASPACE Coruña. Diagnóstico: parálisis cerebral infantil tipo espástica. GMFCS: 4. MACS: 5

  Usuario 2. Usuario residente en ASPACE Coruña. Diagnóstico: parálisis cerebral infantil. GMFCS: 3. MACS: 1

- Los directores del proyecto, que se encargaron de supervisar el funcionamiento de la aplicación, comentando detalles para mejorar y asegurándose de que la herramienta funcione tal como se acordó en las reuniones.

- El alumno, que llevó a cabo la interacción con la aplicación y guió el manejo para aprovechar las capacidades de la misma.

5.2 Sesión No. 1

En primer lugar, se realizó un acercamiento a lo que sería una sesión para el usuario 1. El usuario usa una silla de ruedas (figura 5.1) para desplazarse, tiene grandes problemas para manejar sus manos y agarrar objetos como sería la placa Micro:bit, y para comunicarse oralmente, por lo cual utiliza un documento plastificado con una gran cantidad de pictogramas que le ayudan en esta tarea, tal y como podemos ver en la misma figura.

Se comenzó la sesión cediendo el control de la aplicación a los fisioterapeutas, para que puedan crear un perfil con el usuario a tratar en estos momentos. Aquí se encuentran dos problemas que afectan a la aplicación:

- Al escoger una imagen pequeña para el perfil, se descolocan los diferentes slots que existen para los datos de usuario, lo cual se solucionaría más adelante exigiendo un mínimo de dimensiones para las imágenes.

- El otro problema viene generado por los nombres de usuario compuestos, ya que se está pasando como parámetro a las pantallas siguientes el nombre de usuario, el cual se entiende como dos parámetros diferentes. Esto se arregla sustituyendo el parámetro del nombre por el número identificador de cada usuario.

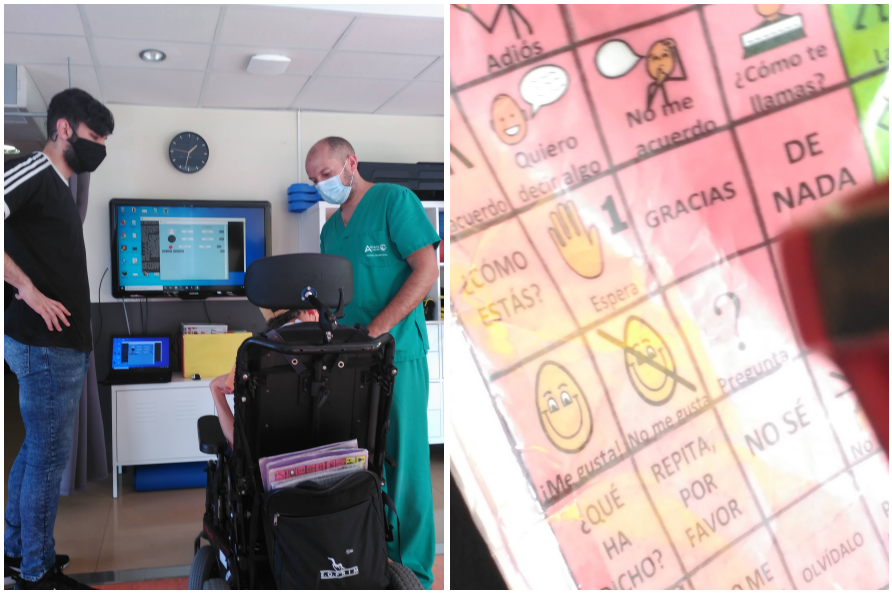

Figura 5.1: Usuario 1 preparándose para su sesión y documento con pictogramas usado para comunicarse

Para que el usuario pueda controlar el juego, utilizará una gorra que llevará pegada la placa Micro:bit, de manera que se pueda respetar la orientación de los movimientos tal como están codificados. El usuario probó los tres juegos implementados en la aplicación. Se observó que el usuario no tiene problemas cuando se trata de movimientos en el eje horizontal, sin embargo, cuando hay movimientos en el eje vertical, tal y como está codificado (inclinar hacia delante implica subir, mientras que inclinar hacia atrás implica bajar), no resulta del todo intuitivo ya que los movimientos son opuestos a lo que el movimiento requerido implica, por ejemplo, con el Micro:bit pegado a la gorra, para bajar el personaje controlado haría falta subir la cabeza.

El usuario termina una partida de cada uno de los juegos con resultado exitoso. Los fisioterapeutas muestran satisfacción con el funcionamiento para el primer caso, mientras que el usuario expresa felicidad a través de los pictogramas, por haber realizado satisfactoriamente la sesión.

5.3 Sesión No. 2

Para el usuario 2 se siguen los mismos pasos que se siguieron en la primera sesión a la hora de registrar sus datos en el sistema. El usuario tiene menos problemas de movilidad y de uso de sus manos, sin embargo se siguen usando diferentes aparatos para que trabaje en su equilibrio. En primera instancia, se usó para controlar los juegos una tabla de equilibrio, a la cual pegamos la placa Micro:bit en uno de sus extremos, tal como vemos en la figura 5.2 (izquierda). Para la segunda parte de la terapia, el usuario se sienta sobre un balón sobre el

cual tendrá que equilibrarse. Adherimos la placa Micro:bit a una gorra, de igual manera que en el caso del usuario 2, para que pueda hacer movimientos en el eje vertical, como vemos en la figura 5.2 (derecha).

El usuario expresa gratitud al finalizar las pruebas con éxito, de la misma manera que los fisioterapeutas señalan que la aplicación es útil para las terapias, al ahorrarles una gran cantidad de tiempo y esfuerzo.

Figura 5.2: Tabla y balón de equilibrio preparada para que el usuario 2 realice su terapia

5.4 Seguimiento de resultados

Una vez finalizadas las sesiones con los dos usuarios, es hora de que los fisioterapeutas puedan visualizar los datos de las partidas realizadas para poder extraer información sobre ellas. Como ya comentamos anteriormente, en la pantalla de puntuaciones podemos ver la fecha de la partida, la puntuación, los parámetros de la partida y si ha superado la puntuación umbral seleccionada.

Los fisioterapeutas se muestran contentos con el formato en el que se muestran los datos, comentando que quizás sea más útil mostrar las partidas en orden cronológico que por puntuación, por lo que se añade a la lista de tareas para realizar posteriormente.

Para concluir, se comenta la funcionalidad referente a exportar los datos a formato Excel. Aunque esta función ya estaba codificada, no se mostraba en la aplicación ya que había dudas sobre cómo seleccionar los datos para extraer. Se sugiere a los fisioterapeutas la posibilidad de acceder a los datos por fecha, de manera que seleccionen una, y se extraigan todas las

coincidencias de datos que sean posteriores a esa fecha seleccionada, a lo cual responden positivamente, por lo que será finalmente esa la forma de implementar dicha funcionalidad.

Capítulo 6

Conclusiones y líneas futuras

El proyecto se da por finalizado una vez obtenido el visto bueno tanto de los fisioterapeutas de ASPACE Coruña como de los directores del propio proyecto. A continuación, presentaremos una reflexiones sobre el desarrollo de todo el proyecto y algunas líneas para el trabajo futuro.

6.1 Conclusiones

La aplicación desarrollada es un programa completo y funcional que ha sido desarrollada acorde con lo que necesitan los fisioterapeutas para realizar sus terapias de la manera más ágil posible, manteniendo unificados los datos y facilitando la tarea lo más posible. Por otro lado, la buena valoración recibida por parte de los usuarios, muestran que los juegos desarrollados son atractivos para ellos y que el empleo de la placa Micro:bit permite que realicen una interacción acorde con sus limitaciones motoras.

El desarrollo del proyecto está relacionado con la mención de Sistemas de Información cursada. En primer lugar, los conocimientos adquiridos han sido de gran utilidad para el desarrollo del sistema que gestiona la información generada por las diferentes fuentes y que, como ya se ha mencionado, constituye la funcionalidad más importante de la aplicación, y es con ella con la que los fisioterapeutas estudian más en conciencia el comportamiento de los usuarios de la aplicación. Para ello, fue fundamental conocer los principios de diseño, tanto de base de datos como de aplicaciones en general, desarrollados durante el año y medio que comprende la mención realizada, a lo largo de diferentes asignaturas.

Otros conocimientos requeridos durante el desarrollo del proyecto fue la utilización de una metodología ágil, las cuales son desarrolladas en asignaturas como Metodologías de Desarrollo, entre otras o, por ejemplo, el desarrollo de interfaces de usuario en asignaturas como Interfaces Persona-Máquina y otras similares.

También reseñar la amplia utilidad a nivel personal de este proyecto, debido a que gran

parte del conocimiento obtenido durante el desarrollo es completamente nuevo, como todo el referido al lenguaje Python y sus diferentes librerías, o de los fundamentos de la comunicación por Bluetooth y cómo establecerla entre diferentes dispositivos. Aunque en cierta medida, el alumno tenía una buena base sobre los temas comentados en el párrafo anterior, durante todo el desarrollo se ha ido ampliando notablemente.

Por otro lado, se ha cumplido también el objetivo de realizar un Trabajo Fin de Grado que preste un servicio a un colectivo que necesita un apoyo especial. En este sentido, está previsto que ASPACE Coruña utilice la aplicación desarrollada en todos sus servicios.

Trabajar con colectivos desfavorecidos siempre es una labor agradable, principalmente por la gratitud que suelen mostrar hacia la gente que pretende ayudar, como ha sido el caso. La felicidad y agradecimiento demostrado por los usuarios del programa durante las pruebas, ya por si sólo, hace que el haber prestado el tiempo para desarrollar algo como este proyecto merezca totalmente la pena.

6.2 Líneas futuras

La continuación del desarrollo serviría de igual manera para mejorar la aplicación y su propósito, así como para ampliar los conocimientos sobre las herramientas usadas. Algunas de las funcionalidades descartadas durante el desarrollo, se recogerán a continuación como líneas futuras de desarrollo. Estas son, como se comentaba antes, ideas propuestas que final han sido descartadas por diversos motivos, pero aun así existe un pequeño de plan sobre cómo realizarlas o, al menos, unas líneas maestras que seguir para alcanzar el conocimiento necesario para realizarlas.

Estas líneas son las siguientes:

- **Navegación por barrido :** Los fisioterapeutas de ASPACE Coruña sugirieron desarrollar una interfaz que fuera totalmente navegable a través de la placa Micro:bit, para que los usuarios pudieran usarla en prácticamente su totalidad, desde que se selecciona el servicio, hasta ver sus puntuaciones para ver como han mejorado. Existen librerías de Python que permiten modificar el comportamiento del cursor, por lo que lo que se intentaría hacer en este caso sería asociar movimientos de la Micro:bit al cursor, de forma análoga a como se hace en los juegos. A mayores, habría que incluir unas zonas de pulsación, en las cuales, al esperar unos segundos sobre las propias, se hiciera la interacción con el botón correspondiente.

  Esta funcionalidad se descartó finalmente por la complejidad que entraña y por que no cuadra con la manera con la que se ha enfocado la aplicación, ya que los datos que nos pide para acceder y para navegar en algunas pantallas no son de conocimiento de los usuarios, como son las claves.

- **Inclusión de juegos nuevos:** El problema a tratar en este caso es que la aplicación solo contiene tres juegos y en caso de que se desarrolle otro que pueda ser útil para los usuarios, habría que retocar bastante código, además de reestructurar toda la pantalla de selección de juegos. No es un problema excesivamente complejo, pero la falta de tiempo y el hecho de que, de momento no hay más juegos desarrollados y no se necesita esta funcionalidad han llevado a finalmente no codificarla. En el apéndice B se explicará más en profundidad cómo se realizaría esta tarea.

- **Mejora de conectividad Bluetooth:** Actualmente la única configuración Bluetooth que se maneja es la dirección MAC a la que conectarse. La idea que surgió fue el conseguir que antes de abrir la aplicación, o en un momento determinado, siempre antes de los juegos, se buscaran dispositivos Bluetooth, y que conectara automáticamente con el que sea la placa Micro:bit, o que seleccione la dirección MAC de la misma. Esta idea no se desarrolló al no haber sido capaz de unificar un programa que localizara los dispositivos y al mismo tiempo obtuviera la información necesaria para conocer que se trata de la placa Micro:bit y finalmente, extraer la dirección MAC.

- **Inclusión de otras tecnologías:** El desarrollo del presente proyecto se centraba en la placa Micro:bit, pero sería muy interesante abordar el empleo de otras placas y tecnologías, como realidad aumentada y realidad virtual, en terapias de personas con dificultades motoras severas.

6.3 Difusión

Como se ha mencionado anteriormente, la aplicación desarrollada va a ser utilizada en los distintos servicios de ASPACE Coruña. La primera sesión de pruebas ha tenido muy buena acogida por parte de esta asociación y por distintos medios de comunicación. En particular, nos gustaría mencionar dos artículo de prensa:

> ”Un estudiante de Informática de la UDC crea una aplicación para personas con parálisis cerebral”, La Voz de Galicia, 27/07/2020

> ”Informática social sobre el terreno”, La Opinión, 28/07/2020

Consideramos, además, que es importante dar a conocer este desarrollo en el ámbito de la universidad, por lo que está previsto que la aplicación sea presentada en el congreso XoveTIC que se celebrará en el CITIC en octubre. Asociado a este congreso, se ha elaborado la siguiente publicación:

Alejandro Lopez-Fernandez, Ruben Carneiro-Medin, Thais Pousada, Betania Groba-González, Adriana Dapena, ”Development of Recreational Content with Micro:Bit for Intervention with People with Cerebral Palsy”, Proceedings MDPI, vol. 54, 2020.

Apéndices

Apéndice A

Manual de usuario de la aplicación

Este apéndice es un manual de usuario que explica que necesitamos y cómo tenemos que manejar la aplicación. El contenido abarca desde la instalación hasta el manejo de la aplicación. Además, se incluyen varias fichar realizadas para que los terapeutas puedan acceder de forma rápida y ágil a la información que necesitan para realizar la terapia. Además, pueden ser utilizadas para interactuar con algunos usuarios.

En general, el manejo de la aplicación es sencillo e intuitivo. Además, en la mayoría de las pantallas se ha incluido un botón para acceder a ayuda on-line.

A.1 Configuración previa

La aplicación requiere tener instalado Python 3.7.5 para Windows. Deben instalarse varias librerías, alguna son nativas del lenguaje, por lo que no será necesario descargarlas. El resto de librerías pueden instalarse con el gestor de paquetes pip (pip install <nombre del paquete>). A continuación se muestra el listado de librerías necesarias:

- PySimpleGUI, necesaria para crear las interfaces.
- Pygame, para la lógica interna de los juegos.
- PIL, para el procesamiento de las imágenes.
- Smtplib, para la gestión de los correos electrónicos.
- Random, para generar números pseudoaleatorios.
- Asyncio, para funciones asíncronas.
- Platform, para acceder a detalles identificativos de la plataforma utilizada.
- Json, para realizar parsing y envío de datos en formato JSON.

- Datetime, para crear fechas en formato cómodo.
- Bleak, para gestionar la comunicación Bluetooth.
- Xlsxwriter, para crear archivos con formato Excel.

Para poder interactuar con la base de datos, debe añadirse la librería MySQLConnector que debe ser descargada desde la página oficial de MySQL (`https://dev.mysql.com/downloads/connector/python/`). Por último, es necesario tener la base de datos configurada y montada en MySQL y configurar los parámetros de conexión a la base de datos. Es recomendable para gestionarla de manera más cómoda, utilizar MySQL Workbench.

A.2 Ejecución de la aplicación

Se proporciona una carpeta con todos los ficheros de la aplicación. Esta carpeta contiene un archivo ejecutable (TFGApp.exe), además de todo el código fuente, sonidos, imágenes y demás archivos de configuración, todo estructurado en directorios para tener cada archivo localizado.

La figura A.1 muestra la pantalla principal que tiene tres opciones: Entrada a la aplicación, Entrada como admin y Salir.

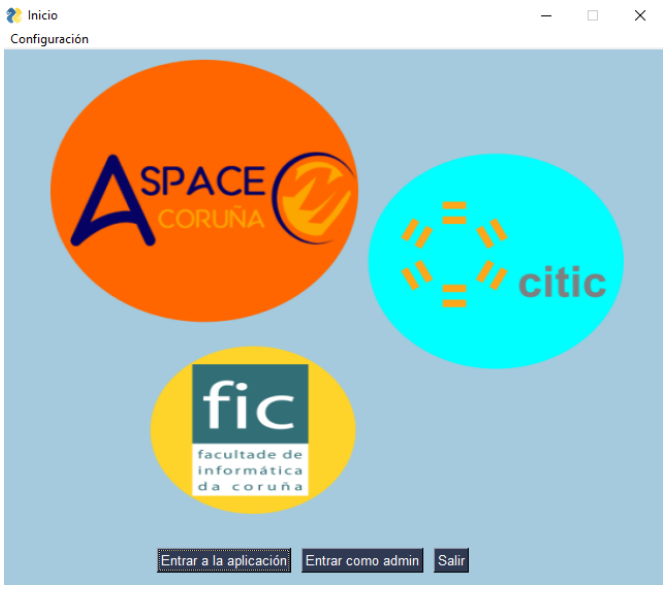

Figura A.1: Pantalla de inicio de la aplicación

A.2.1 Entrar a la aplicación

Una vez arrancado el ejecutable mencionado antes, aparece la pantalla de inicio de la aplicación, con las opciones para navegar por la aplicación, además de un menú superior con la configuración Bluetooth.

La ficha mostrada en la figura A.3 recoge el funcionamiento asociado a la navegación por la aplicación y la ficha A.4 muestra con más detalle las funcionalidades. Por otro lado, las fichas A.5, A.6 y A.7 recogen la parte de configuración y manejo de cada juego por separado.

La configuración de Bluetooh se realiza a través de la pantalla que aparece en la figura A.2 muestra la pantalla para realizar la configuración. En esta pantalla un recuadro de texto que contiene la dirección MAC del dispositivo que esté conectado actualmente. Debemos de sustituir dicha dirección por la nueva que vayamos a usar. Para que todo funcione correctamente, debemos seguir estos pasos:

1. Desemparejar el dispositivo anterior del ordenador.
2. Utilizando alguna aplicación de rastreo de dispositivos Bluetooth, localizar nuestro dispositivo nuevo (por ejemplo, uno de los que aparecen en el proyecto Micro:bit BLE [14]). En él, podremos obtener también la dirección MAC del dispositivo.
3. Emparejar el dispositivo a nuestro ordenador, siguiendo las instrucciones para ello que están en la documentación de Micro:bit [20]).
4. Introducir la dirección MAC obtenida en la ventana de configuración Bluetooth de la aplicación.

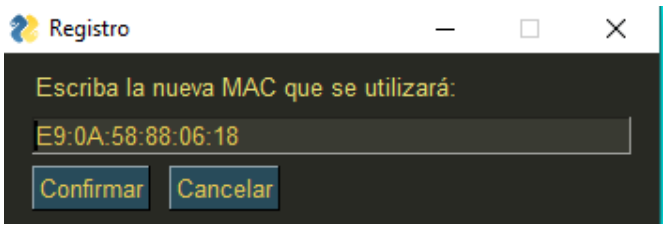

Figura A.2: Pantalla que nos permitirá cambiar la dirección MAC a usar

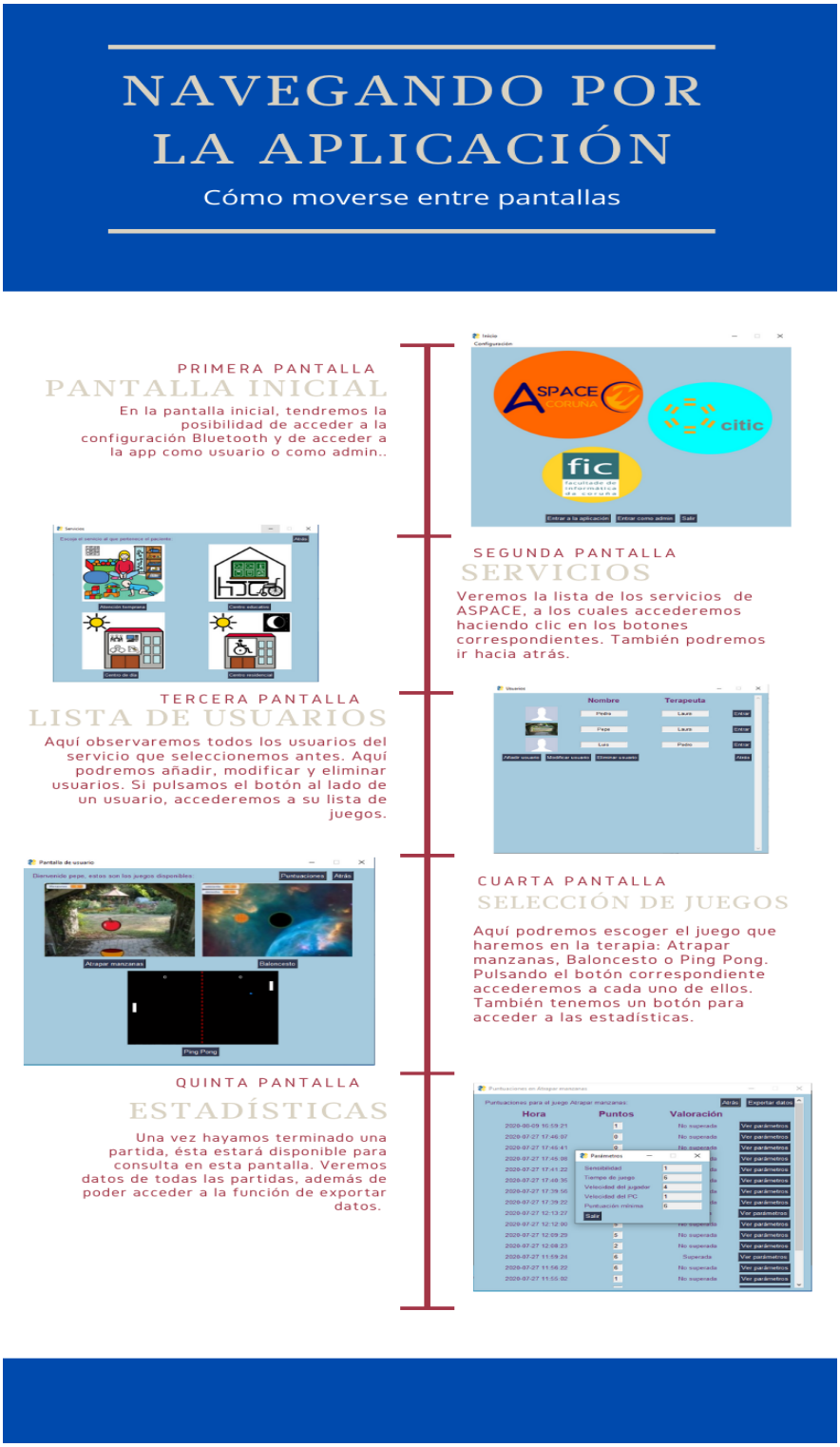

Figura A.3: Cómo navegar por la aplicación

Figura A.4: Funcionalidades de la aplicación

Figura A.5: Detalles del juego Atrapar Manzanas

Figura A.6: Detalles del juego Baloncesto

Figura A.7: Detalles del juego Ping Pong

A.2.2 Entrar para admin

El acceso se realiza introduciendo su contraseña. Aparecerá la pantalla de los servicios que permite realizar las siguientes tareas: cambiar las contraseñas de acceso a los servicios, cambiar el responsable de cada uno de ellos y su correo electrónico. De la misma manera, será posible recuperar las contraseñas. Se enviará un correo a la persona que aparezca como responsable del servicio con una contraseña temporal. En la ficha de la figura A.8 se recogen todas las funcionalidades.

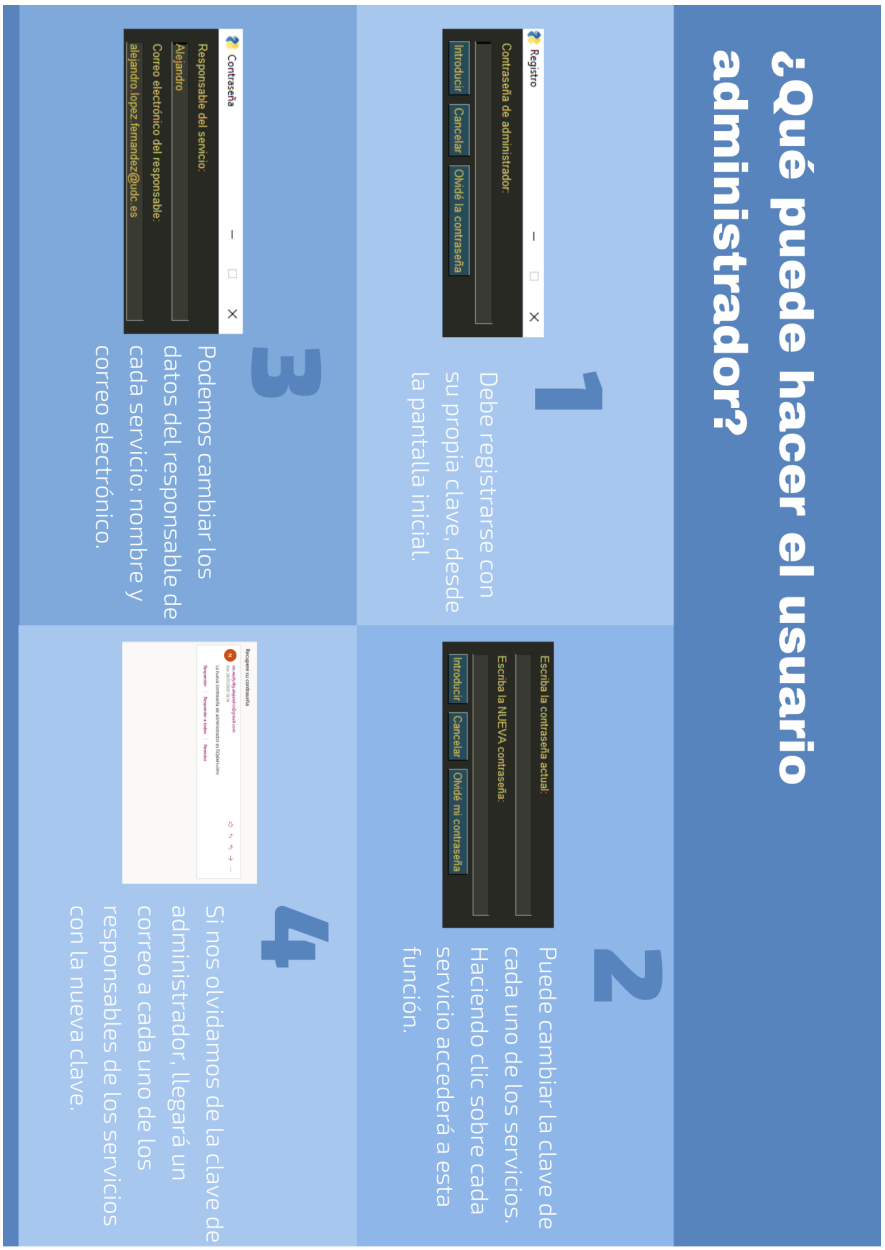

Figura A.8: Tareas del usuario administrador

Apéndice B

Añadir juegos a la aplicación

En la versión actual de la aplicación, no está configurada para incorporar de forma sencilla nuevos juegos, siendo esta una línea de futuro desarrollo. Sin embargo, en la actualidad, es posible realizarlo siguiendo las instrucciones contenidas en este apéndice. Además, se muestra las líneas a seguir para mejorar la inclusión de juegos en un nueva versión de la aplicación.

B.1 Añadidos al modelo de datos

Con el modelo de datos actual, el nombre permite diferenciar los juegos. Por ello, sería conveniente añadir un atributo a la entidad/tabla que represente el archivo que contiene el juego, de manera análoga a cómo se almacena la imagen en la entidad **user**. De esta manera, cuando queramos añadir un juego nuevo, añadimos la ruta del archivo, para que en el momento de ejecutarlo, únicamente tengamos que hacer una llamada a dicha ruta. Siguiendo el diseño ya realizado para las imágenes de perfil,también añadiremos un atributo para las imágenes que representan cada uno de los juegos. Para hacer esto tenemos dos opciones: podemos hacerlo a través de algún script que manipule la base de datos, o podemos hacerlo a través de la herramienta MySQL Workbench (recomendado, mucho más cómodo, figura B.1).

No parece que sea necesario ningún otro cambio referente al modelo de datos. Sin embargo, esto no quiere decir que si se comenzase el desarrollo, se pudieran descubrir otras modificaciones necesarias para hacer que funcione.

B.2 Añadidos al código

Sabiendo que prácticamente el modelo de datos es idéntico al de la versión actual del proyecto, únicamente nos tendremos que centrar en qué llamadas a la base de datos serán necesarias para obtener los datos necesarios o para almacenarlos correctamente.

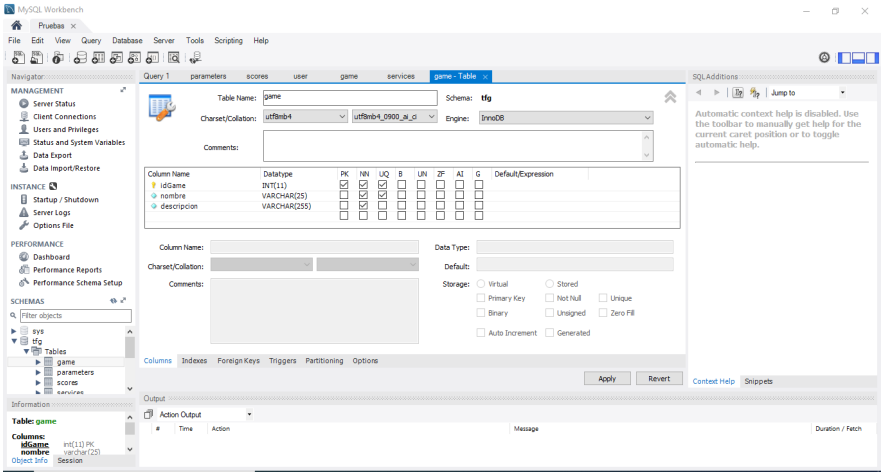

Figura B.1: Lugar en el que tenemos que modificar la base de datos

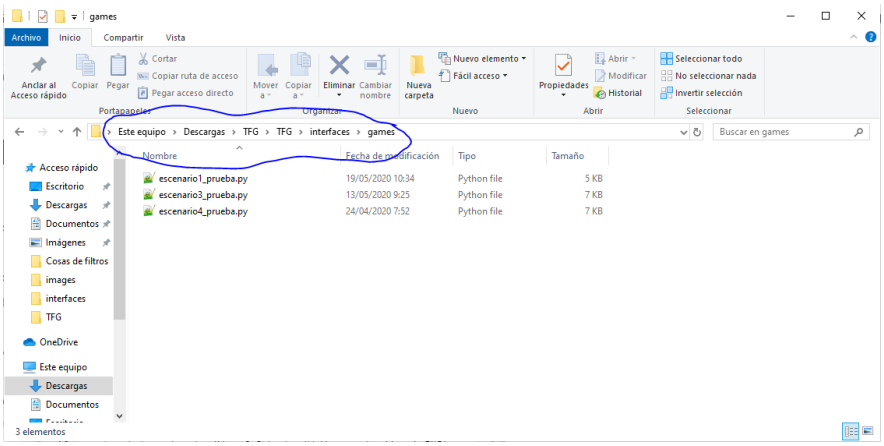

Figura B.2: Carpeta que contendría todos los juegos incluidos en la aplicación. La ruta marcada indica su ubicación.

Actualmente, la aplicación está configurada de tal manera que obtiene directamente el acceso a los juegos llamando a los archivos correspondientes, es decir, los archivos como tal o información sobre los mismos no están almacenados en ningún sitio, nada más que en el sistema de directorios del proyecto. Quizás el hecho de tenerlos referenciados en la base de datos nos permita crear un directorio a mayores que permita tener todos los juegos localizados (figura B.2), y en caso de querer añadir, tener una carpeta ”raíz” donde se almacenarán los nuevos juegos para ser insertados, que como se comentaba antes, es el mismo método empleado para las imágenes de perfil de los usuarios.

Dicho esto, necesitaremos una consulta que nos permita acceder a todos los juegos que haya almacenados en la base de datos para así mostrarlos posteriormente. Bastará con extraer el nombre del juego y la ruta del archivo correspondiente, aunque podemos usar la descrip-

ción y añadir al mismo tiempo un apartado que nos permita ver una pequeña ayuda sobre el juego. La parte fundamental de este apartado es añadir juegos, por lo que necesitaremos una sentencia INSERT INTO para esto (figura B.3). Necesitaremos que la aplicación pida el nombre del juego, la descripción y la ruta de la imagen y del archivo, que, como decíamos antes, tendremos una carpeta dedicada para los archivos de juegos. Como validación, sería útil que los únicos archivos que podamos incluir aquí sean de código fuente de Python (extensión .py). De la misma manera, para completar la funcionalidad, debemos tener algo que nos permita eliminar o modificar el juego si nos hemos equivocado en algún detalle, o simplemente ya no nos es útil, entonces, tendremos un método que utilice una sentencia UPDATE o DELETE, en función de la opción que hayamos escogido.

```
def insertar_datos(username,terapeuta,img):
    try:
        cnx = mysql.connector.connect(user="root", password="root", host="127.0.0.1", database="tfg")
        cur = cnx.cursor(buffered=True)
    except:
        print("Error conectándose a la base de datos.")

    #Do work
    try:
        id_query = "SELECT idUser FROM user ORDER BY idUser DESC LIMIT 1"
        cur.execute(id_query)
        id_parameter_new = cur.fetchone()[0]+1
        sql = "INSERT INTO user (idUser,username,nombreTerapeuta,idServicio,claveUser,imgUser) VALUES (%s, %s, %s, %s, null,%s)"
        values = (id_parameter_new,username,terapeuta,sys.argv[1],img)
        cur.execute(sql,values)
        num_games = "SELECT count(*) FROM game"
        id_query = "SELECT idParameters FROM parameters ORDER BY idParameters DESC LIMIT 1"
        cur.execute(id_query)
        id_parameter_new1 = cur.fetchone()[0]+1
        cur.execute(num_games)
        times = cur.fetchone()[0]
        i = 0
        while i < times:
            sql = "INSERT INTO parameters (idParameters,idUser,idGame,sensibilidad,tiempo,v_user,v_ia,umbralVal) VALUES (%s,%s,%s,%s,%s,%s,%s,%s)"
            values = (id_parameter_new1,id_parameter_new,i+1,3,60,2,2,20)
            cur.execute(sql,values)
            i = i+1
            id_parameter_new1 = id_parameter_new1+1
        cnx.commit()
        sg.popup(f"Usuario registrado con éxito!")
        cnx.close()
    except:
        print("Error inesperado insertando los datos.\n")
```

Figura B.3: Ejemplo de función con INSERT INTO para el caso de usuarios (análogo a lo que sería para los juegos)

Para concluir este apartado, debemos de mencionar cómo vamos a almacenar la ruta del archivo con el juego, ya que hay que tener en cuenta que no todos los sistemas de archivadores de los ordenadores en los que se pueda usar esta aplicación son iguales, por eso necesitamos algo que haga que la ruta que usemos sea siempre la misma, o que cualquier ordenador pueda entender la ruta a la que estamos llamando. Hay que tener en cuenta que tal como está programada la aplicación, todo el código se ejecuta desde la misma carpeta, y de forma idéntica a como se hace con las imágenes de perfil, cortaremos la ruta a partir de la primera ocurrencia del nombre de la carpeta, seguido de una barra (/). De esta manera, si por ejemplo, tuviéramos una ruta como esta: C:/Usuarios/usuario/appTFG/juegos/juego1.py, obtendríamos dos cadenas, C:/Usuarios/usuario/appTFG/ y juegos/juego1.py, siendo esta segunda la que almacenaríamos en la base de datos. Con este fragmento de la ruta, seremos capaces de ejecutar ese código en cualquier ordenador, ya que esa ruta existirá siempre. Usaremos el mismo método para las imágenes de los juegos, siguiendo paso por paso lo que se hizo con las imágenes de perfil de los usuarios. Todas estas modificaciones deben realizarse

en el archivo que trata la pantalla de usuario, es decir, donde vemos los juegos (en nuestro caso: ejemplo_pantalla_usuario.py)

B.3 Añadidos a la interfaz gráfica

Quizás esta parte sea la que más complejidad le añada al nuevo programa, ya que el resto de retoques hechos son, al final, cosas que ya se han hecho en otros puntos de la aplicación. Sin embargo, la interfaz gráfica complicará un poco el proceso, especialmente a la hora de ubicar las imágenes que representen los juegos.

Actualmente tenemos dos filas, la primera con dos imágenes colocadas una a cada lado de la mitad de la pantalla, mientras que en la segunda fila tenemos una imagen centrada. Debajo de cada una de las imágenes tendremos un botón que ejecutará la llamada al juego correspondiente.

Dicho esto, teniendo en cuenta que desconocemos en todo momento el número de juegos que hay almacenados en la base de datos, hay que plantearse diversos escenarios. En principio, seguiremos colocando los juegos en filas de dos elementos, para respetar las dimensiones de las pantallas de la aplicación. A mayores también añadiremos la posibilidad de desplazar verticalmente la pantalla (figura B.4) en caso de que haya numerosos juegos y no se puedan visualizar todos en la pantalla.

```
window = sg.Window('Usuarios',size = (640,540)).Layout([[sg.Column(layout=layout, scrollable=True,size = (620,540),vertical_scroll_only=True)]])
```

Figura B.4: Código que nos permite incluir scroll vertical

Ahora debemos de plantearnos dos situaciones: que haya un número par de juegos o que haya un número impar. Como comentábamos antes, si son pares, nos quedarán justos para colocar de dos en dos, si son impares, el último será el único de su columna. Por tanto, debemos de lanzar una consulta a la base de datos que nos diga cuantos juegos hay almacenados, y una vez lo sepamos, ejecutaremos un código del estilo al que mostramos en el apartado 4.8.2, página 48, sólo que en vez de usar un bucle for, usaremos un bucle while, cuya condición de parada sea no ser superior al número de juegos entre 2. Dentro de este bucle, tendremos una variable que nos indique el número de juegos, que deberemos comprobar si es par o impar. Si este es par, creará una fila con las imágenes y otra con los correspondientes botones, cada uno con el nombre del juego en cuestión. Si el número fuese impar, cuando acabe el bucle, añadirá la fila a mayores del juego que falta, junto con la fila que le corresponde a su botón. De esta manera, en principio, deberíamos tener una lista ordenada con los juegos que tengamos, independientemente del número que sea.

Finalmente, debemos de tener en cuenta cómo vamos a gestionar los eventos de la interfaz con las tareas que tenemos que realizar en la base de datos. En interfaces de PySimpleGUI, podemos identificar los eventos producidos por los botones porque el evento tendrá el mismo nombre que el texto contenido por el propio botón, por lo que, si dentro del bucle de lectura de pantalla, creamos otro bucle como el del párrafo anterior, en el cual comprobaremos que el evento se corresponda con el nombre de alguno de los juegos, y si esto es así, que lo ejecute, enviando los parámetros necesarios, que son el usuario y el identificador de juego. En principio, este sería el código perteneciente a la interfaz gráfica que haría posible visualizar esta función.

Recordemos que este es un plan a seguir en caso de realizar esta funcionalidad en un futuro, no quiere decir necesariamente que esto se vaya a hacer así y que funcione como es debido, simplemente es la idea que se ha tenido para esto y es así cómo se probaría en un primer intento.

Lista de acrónimos

ASPACE *Asociación de Padres de Personas con Parálisis Cerebral.*

BBC *British Broadcasting Corporation*

BLE *Bluetooth Low Energy*

USB *Universal Serial Bus*

LED *Light-Emitting Diode*

MAC *Media Access Control*

CSV *Comma-Separated Values*

API *Application Programming Interfaces*

GMFCS *Gross Motor Function Classification System*

MACS *Manual Ability Classification System*

Glosario

Sprite Tipo de mapa de bits dibujado en la pantalla de ordenador por hardware gráfico especializado sin cálculos adicionales de la CPU. A menudo son pequeños y parcialmente transparentes, dejándoles así asumir otras formas a la del rectángulo.

Dirección MAC Identificador de 48 bits que corresponde de forma única a una tarjeta o dispositivo de red. Se la conoce también como dirección física, y es única para cada dispositivo.

Diagrama de Gantt Herramienta gráfica cuyo objetivo es exponer el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado. No indica relaciones entre tareas.

Propiedades ACID Propiedades que necesita tener una transacción confiable: atomicidad, consistencia, aislamiento y durabilidad.

Layout Propiedad de las interfaces gráficas que representa la colocación de los diferentes componentes de los que consta.

Mockup Fotomontaje o esquema que permite al cliente o al diseñador esbozar una idea sobre cómo se ha diseñado un elemento.

Lenguaje de bloques (Scratch) Lenguaje que, de forma visual, encadena bloques de código a modo de puzle para facilitar su aprendizaje y entendimiento.

Pictograma Signo icónico dibujado y no lingüístico, que representa figurativamente un objeto real, o un significativo.

Periféricos Denominación genérica para designar a los dispositivos auxiliares e independientes conectados a la unidad central de procesamiento de una computadora.

Scroll Movimiento en 2D (horizontal o vertical) de los contenidos que conforman el escenario de un videojuego o la ventana que se muestra en una aplicación informática.

Bibliografía

[1] P. Póo-Argüelles, "Parálisis cerebral infantil," *Asociación Española de Pediatría*, 2008. [En línea]. Disponible en: https://www.aeped.es/sites/default/files/documentos/36-pci.pdf

[2] "El juego en niños con parálisis cerebral," 2018. [En línea]. Disponible en: https://interactua.es/el-juego-en-ninos-con-paralisis-cerebral/

[3] S. Lopes, P. Magalhaes, A. Pereira, J. Martins, C. Magalhaes, E. Chaleta, and P. Rosario, "Games used withserious purposes: A systematic review of interventions in patients with cerebral palsy," *Front. Psychol.*, 9 2019. [En línea]. Disponible en: https://doi.org/10.3389/fpsyg.2018.01712

[4] BBC, "Micro:bit educational foundation." [En línea]. Disponible en: https://microbit.org/

[5] "Comprar placa bbc micro:bit." [En línea]. Disponible en: https://www.xplora360.es/producto/placa-bbc-micro-bit/

[6] "Python homepage." [En línea]. Disponible en: https://www.python.org/

[7] "Micropython homepage." [En línea]. Disponible en: https://www.micropython.org/

[8] "Pygame library homepage." [En línea]. Disponible en: https://www.pygame.org/news

[9] "Mysql homepage." [En línea]. Disponible en: https://www.mysql.com/

[10] "Pysimplegui library documentation." [En línea]. Disponible en: https://pysimplegui.readthedocs.io/en/latest/

[11] "Tkinter reference page." [En línea]. Disponible en: https://wiki.python.org/moin/TkInter

[12] J. J. Sánchez-Hernández, "Bluetooth low energy remote control for spotify." [En línea]. Disponible en: https://josejuansanchez.org/projects/2018/08/20/bluetooth-low-energy-remote-control-for-spotify.html

[13] M. Kılıç, "Microbike, turn your micro:bit into a controller." [En línea]. Disponible en: https://github.com/musabkilic/MicroBike/

[14] "Read micro:bit data from linux via bluetooth (ble)." [En línea]. Disponible en: https://github.com/alcir/microbit-ble

[15] J. Ruiz-Vanoye, A. Fuentes-Penna, O. Díaz-Parra, D. Díaz, S. López, A. Medina, and A. González, *Metodologías de Desarrollo de Software.* Editorial Académica Dragón Azteca, 03 2017.

[16] K. Schwaber and J. Sutherland, "The definitive guide to scrum: The rules of the game." [En línea]. Disponible en: https://www.scrumguides.org/scrum-guide.html

[17] "Bbc micro:bit micropython documentation." [En línea]. Disponible en: https://microbit-micropython.readthedocs.io/en/v1.0.1/

[18] "Code with mu: a simple python editor for beginner programmers." [En línea]. Disponible en: https://codewith.mu/

[19] A. Silberschatz, H. Korth, S. Sudarshan, and F. Pérez, *Fundamentos de bases de datos.* McGraw-Hill, 2006.

[20] "Pairing and flashing code via bluetooth." [En línea]. Disponible en: https://support.microbit.org/support/solutions/articles/19000051025-pairing-and-flashing-code-via-bluetooth

[21] "Bluetooth profile for bbc micro:bit." [En línea]. Disponible en: https://lancaster-university.github.io/microbit-docs/resources/bluetooth/bluetooth_profile.html

[22] I. Harvey, "Bluepy library documentation." [En línea]. Disponible en: https://ianharvey.github.io/bluepy-doc/index.html

[23] H. Blidh, "Bleak project." [En línea]. Disponible en: https://github.com/hbldh/bleak

[24] A. Guerra Marrero, "Tutoriales pygame." [En línea]. Disponible en: http://razonartificial.com/tutoriales-pygame/

[25] R. Palisano, P. Rosenbaum, D. Bartlett, and M. Livingston, "Clasificación de la función motora gruesa extendida y revisada." [En línea]. Disponible en: https://canchild.ca/system/tenon/assets/attachments/000/000/079/original/GMFCS-ER_Translation-Spanish.pdf

[26] "Manual ability classification system for children with cerebral palsy 4-18 years." [En línea]. Disponible en: https://www.macs.nu/