UNIVERSIDAD POLITECNICA DE VALENCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

GRADO EN INGENIERÍA DE SISTEMAS DE TELECOMUNICACIÓN, SONIDO E IMAGEN

UNIVERSIDAD
POLITECNICA
DE VALENCIA

ESCUELA POLITECNICA
SUPERIOR DE GANDIA

“Desarrollo de una aplicación basada en sistemas iOS con Xcode y Swift para deportistas de escalada”

TRABAJO FINAL DE

Autor/a:
Harutyun Saghatelyan

Tutor/a:
José Marín-Roig Ramón

GANDIA, 2018

Resumen

El presente trabajo expone la creación de una Aplicación Móvil dirigida a un público deportista, concretamente a los deportistas que se dedican profesional o aficionada al deporte de escalado en montaña o rocódromo. El objetivo de la aplicación es hacer un seguimiento de entrenamiento y poder crear un plan de entrenamiento adaptado a cada deportista. Para garantizar la precisión de los diagnósticos de los entrenamientos, se ha basado en un protocolo de pruebas físicas basadas en validaciones científicas por Pedro Bergua (Doctor en Ciencias del Deporte y entrenador profesional).

Se analizará el estado físico del deportista mediante pruebas "Autotest" creadas por el Dr. Pedro Bergua. Dicha APP interactua con una tabla fabricada por la empresa "Eurohold Climbing".

Abstract

This project exposes the development of a Mobile Application for people who are dedicated professional or amateur to the sport of climbing in mountain or in climbing walls. The aim of the Application is to track the trainings and to create a training plan adapted to each athlete. To ensure the accurancy, the diagnosis of training has been based on a protocol of physical tests based on scientific validation by Pedro Bergua (Doctor of Sport Science and professional coach).

The physical state of the athlete will be analyzed through "Autotest" created by Dr. Pedro Bergua. This App interacts with a board manufactured by the Company "Eurohold Climbing".

Palabras clave, keywords

iOS, xCODE, Aplicación, Swift, Rokodromo, Deporte, Entrenamiento

iOS, xCODE, Application, Swift, Climbing walls, Sport, Training

Agradecimientos

“El camino no ha sido fácil, pero la experiencia que gané es invaluable”
Agradezco a todas las personas que me han acompañado durante este camino.

ÍNDICE

Índice de Figuras

ÍNDICE DE TABLAS

1

Introducción

En actualidad la creación de APP's móviles es uno de los sectores más en auge, emprendedores y empresas importantes del sector IT se involucran en este entorno debido al rol importante del "smartphone" en nuestro día a día. Gracias a estos dispositivos se ha llegado a agilizar muchos de los procesos que no eran tan inmediatos. Prácticamente todos los sectores se involucran e innovan con las nuevas tecnologías, intentan tener su presencia en el mundo digital.

Resulta muy difícil imaginar el mundo sin la tecnología actual y cada vez nos damos cuenta del poder y la capacidad de nuestros móviles y aplicaciones, sea para simplificar tareas de trabajo, pedir comida a domicilio o realizar otras tareas cotidianas.

Actualmente el mercado global de los sistemas operativos para “smartphones” se divide en dos líderes, que prácticamente ocupan el 98 por ciento del mercado global, estos son “Android” y “Apple iOs”. Estos sistemas operativos son programas de bajo nivel que explotan las capacidades del hardware haciendo posible la interactuación y ejecución de las aplicaciones móviles.

En este proyecto se realizará una app para los dispositivos móviles con sistema operativo “Apple iOS”, este sistema operativo ocupa el 29,55 por ciento del mercado global, posicionándose como el segundo sistema operativo después de “Android”. Las principales funcionalidades de la App serán guiar y facilitar el entrenamiento del escalador.

1.1 Objetivos

El objetivo de este trabajo de Fin de Grado es el desarrollo en lenguaje Swift de una aplicación capaz de seguir y crear entrenamientos para los deportistas de escalada.

Dicha aplicación interactúa con una tabla especial, diseñada por la empresa ”Euroholds Climbing”. Los entrenamientos desarrollados son calculados mediante fórmulas basadas en estudios científicos, además el deportista podrá crear de forma libre e individual sus entrenamientos aprovechando las herramientas de la app.

Seguidamente especificaremos los **objetivos principales** a desarrollar:

- Crear una app funcional y de gran calidad para el mercado mundial.
- Crear una interfaz previamente diseñada por el cliente.
- Conseguir la interactuación con la tabla especifica de entrenamiento.

- Funcionamiento coherente con los datos de los estudios científicos realizados.
- Cumplir las fechas previstas para el desarrollo de la app.
- Profesionalidad y responsabilidad durante todo el proceso.

1.2 Etapas de desarrollo y metodología

En este apartado se detallan las diferentes etapas realizadas para conseguir cumplir los objetivos mencionados en el capítulo anterior. En la sección 2.1 se darán a conocer las etapas de creación y en la sección 2.2 mencionaremos las técnicas de metodología adquiridas a lo largo del proyecto.

Hemos decidido realizar el proyecto en cinco etapas:

• Inicio:

En esta etapa crucial, hemos detectado las necesidades y requisitos del cliente. Se han compartido ideas y definido metas para poder garantizar el éxito del proyecto, teniendo en cuenta las posibilidades y restricciones que pueden haber.

Como desarrollador y gestor del proyecto he tenido que tomar desiciones relevantes como pueden ser el presupuesto del proyecto, fecha de entrega y etapas de desarrollo.

• Planificación:

Hemos creado una ruta de planes para mantener el equilibrio entre desarrollo y gestión.

En esta etapa se ha decidido la forma en que se ejecutará el proyecto y que procedimientos se seguirá, tanto de gestión como de desarrollo. Se ha dedicado tiempo para la investigación y aprendizaje, se ha familiarizado con el entorno de desarrollo para las aplicaciones iOS (Xcode).

Se han planificado las fechas de entrega y las etapas de desarrollo a seguir, después de la finalización de cada etapa se han acordado reuniones con el cliente para la verificación y cierre del apartado.

- **Ejecución:**

Basándonos en la planificación, hemos seguido con la ejecución del proyecto. Para el correcto procedimiento se ha garantizado una excelente comunicación con la empresa "Euroholds Climbing".

Se ha procesado a la implementación del software, durante todo el proceso de implementación se ha continuado la formación y aprendizaje del lenguaje de programación Swift.

La metodología seleccionada para gestionar el proyecto es de tipo "Agile", gracias a este tipo de gestión se puede adaptar mejor el proyecto a las necesidades del cliente y a los cambios producidos durante el proceso de realización del proyecto. Debido a su gran flexibilidad, esta metodología de gestión ha ido expandiéndose por las empresas, además "Agile Management" consigue reducir costes de las empresas.

Siguiendo la filosofía de gestión Agile, la implementación del proyecto se ha dividido en diferentes secciones, la involucración del cliente ha llevado a un éxito en la satisfacción y por consiguiente en la fidelización del cliente, ya que se han abierto nuevas oportunidades con el mismo cliente.

- **Seguimiento:**

La monitorización del proyecto es una de las partes vitales, es donde se detectan las desviaciones del proyecto y así poder evitar posibles retrasos en las entregas de los diferentes apartados, además el cliente ha sido activamente involucrado en el proyecto, por tanto todas las modificaciones y aclaraciones se han efectuado en tiempo real, de esta manera se ha conseguido mayor satisfacción del cliente, gracias a un seguimiento continuo.

- **Finalización:**

Por último, para concluir se hará una entrega "llave en mano" del proyecto, se revisarán las obligaciones contractuales en la fase inicial y se concluirá con éxito la finalización del proyecto dejando paso para futuros colaboraciones. La organización, comunicación y corrección de errores durante el desarrollo han sido claves para llegar a realizar el proyecto con éxito.

1.4 Estructura de la memoria

Trataremos de hacer una breve introducción a la estructura general de la memoria, citando el contenido implementado en cada apartado.

En el capítulo dos analizaremos los requerimientos planteados por el cliente, las caracteristicas de la aplicación y el diseño a desarrollar.

En el capítulo tres presentaremos las herramientas utilizadas para el desarrollo de la app, citaremos todas las tecnologías usadas y detallaremos el entorno a usar para la creación de aplicaciones para dispositivos con sistema operativo iOS, además introduciremos el lenguaje a utilizar (Swift), actualmente el lenguaje oficial de desarrollo para el entorno iOS.

En el capítulo cuatro profundizaremos en el diseño y funcionamiento de la aplicación, citaremos los elementos que componen cada vista.

En el capítulo cinco presentaremos la implementación de los apartados relevantes de la app y explicaremos el código fuente de algunas funciones.

Por último, en el capitulo seis hablaremos sobre futuros trabajos con el cliente y la conclusión del trabajo final de grado.

2

Análisis de requerimientos

En este capítulo procederemos a analizar las características de la aplicación a crear tanto de funcionalidad como de diseño, basándonos en las indicaciones del cliente.

2.1 Estado del arte

Antes de empezar con un proyecto, es conveniente realizar un estudio del estado del arte. Las aplicaciones móviles son cada vez más frecuentes en el entorno deportivo, tanto profesional como aficionado.

En el ámbito de aplicaciones deportivas nos podemos encontrar aplicaciones procedentes de marcas deportivas prestigiosas como “Nike Training Club”.

Sin duda esta aplicación ha demostrado que el mundo deportivo necesita la tecnología con el fin de mejorar la calidad de los entrenamientos.

En deportes de escalada, el mercado actual no se encuentra explotado. Al hacer un análisis de mercado, nos hemos encontrado aplicaciones desarrolladas con el mismo propósito, pero no tienen las mismas cualidades que proporciona la app a desarrollar.

Aplicaciones a considerar para entrenamientos de escaladores en tablas:

Boulder Trainer: Disponible para sistemas operativos iOS / Android. Esta app proporciona opciones de entrenamiento, el usuario puede completar rutinas de entrenamientos y programar sus propios entrenamientos.

FIGURA 2.1: APP BOULDER TRAINER

Beastmaker: Disponible para sistemas iOS / Android. Beastmaker app esta creado con el propósito de ser usado con la tabla “Beastmaker hangboard”. Este concepto se acerca mucho a la aplicación que vamos a desarrollar, pero existen ciertas diferencias a considerar.

FIGURA 2.2: APP BEASTMAKER

Los aplicaciones mencionados anteriormente son similares a la app a desarrollar, sin embargo existe un punto fuerte en el cual nos basaremos para desarrollar este proyecto.

Ninguna de las aplicaciones encontradas en el mercado actual esta basada en estudios científicos. La combinación de la tecnología, estudio científico y deporte

pueden traer resultados favorables y destacar en el mercado con un producto completamente diferente de los existentes.

2.1 Especificaciones del proyecto

Al empezar con un proyecto, este debe ir acompañado de especificaciones requeridos por el cliente, por tanto como analista del proyecto, he decidido hacer un estudio de las especificaciones, de esta forma desarrollar el software que satisfaga y resuelva los deseos y necesidades del cliente.

La aplicación junto a la "R-Evolution Board" (R-Evolution Board es una tabla diseñada y creada por la empresa *Euroholds Climbing)*, ayudan al deportista a realizar entrenamientos con máxima precisión.

Esta tabla permite al deportista entrenar de una forma **específica**, dispone de una gran variedad de posibilidades de agarre. Gracias a su sistema de detección de carga la aplicación interactua y permite al usuario crear y planificar entrenamientos o realizar unas pruebas ya determinadas validadas científicamente por **Pedro Bergua**, Doctor en Ciencias del Deporte **y entrenador profesional de escaladores.**

Debemos mencionar que tanto el diseño como la funcionalidad de la app han sido completamente proporcionados por el cliente, nuestra tarea ha sido crear la app que cumpla los requisitos y tenga la funcionalidad exigida por el cliente.

FIGURA 2.3: TABLA FABRICADA PARA LA INTERACTUACIÓN CON LA APP

Los usuarios dispondrán de códigos de licencia para tener acceso a la aplicación. El dispositivo móvil se conectará con la tabla mediante un cable al puerto auxiliar del Iphone. En la figura 2.1 se puede apreciar el aspecto real de la tabla fabricada.

El objetivo general del proyecto a desarrollar es crear una aplicación para dispositivos móviles con sistemas operativos iOS, para los entrenamientos específicos de los dedos con la tabla fabricada por la empresa “Euroholds Climbing”.

FIGURA 2.4: PRUEBAS EN LA TABLA “REVOLUTION BOARD” CON LA APP DESARROLLADA

Durante las primeras reuniones con el cliente, se ha creado un documento de requisitos, este documento contiene todas las características del proyecto a desarrollar. De esta forma se ha concluido la primera versión de la aplicación, este proceso nos ha permitido identificar las necesidades y deseos del cliente.

Características específicas:

- Programación en lenguaje nativo.
- Compatibilidad con diferentes dispositivos de Apple.
- Autenticación de usuarios.
- Desarrollar sobre la base de datos ya existente de la versión Android de la aplicación.
- Diseño gráfico de la app especificado por el cliente.
- Capacidad de crear entrenamientos personalizados.
- Capacidad de guardar los entrenamientos por usuario.
- Sección de Autotest, ésta incluirá unos entrenamientos predeterminados, los usuarios pueden medir sus capacidades físicas mediante esta función.
- Dos tipos de cronómetros para los entrenamientos, ascendente y descendente.
- Idioma: Castellano

Estas son las características principales acordadas al principio del proyecto, durante el desarrollo han habido modificaciones y sugerencias, mayoritariamente referentes al diseño gráfico de la app con motivos de mejorar la experiencia del usuario.

3

Herramientas de desarrollo

En este capítulo introduciremos las herramientas utilizadas para el desarrollo de la aplicación en versión iOS, hablaremos del diseño MVC (Modelo/Vista/Controlador), además haremos una introducción al *software* utilizado y al lenguaje de programación **Swift**.

3.1 IDE Xcode

Xcode es un entorno de desarrollo integrado (IDE, en sus siglas en inglés) para macOS que contiene un conjunto de herramientas creadas por Apple destinadas al desarrollo de software para macOS, iOS, watchOS y tvOS. Su primera versión tiene origen en el año 2003 y actualmente su versión número 9 se encuentra disponible de manera gratuita en el Mac App Store o mediante descarga directa desde la página para desarrolladores de Apple.

El proyecto ha sido desarrollado en la versión “Xcode 9.1”

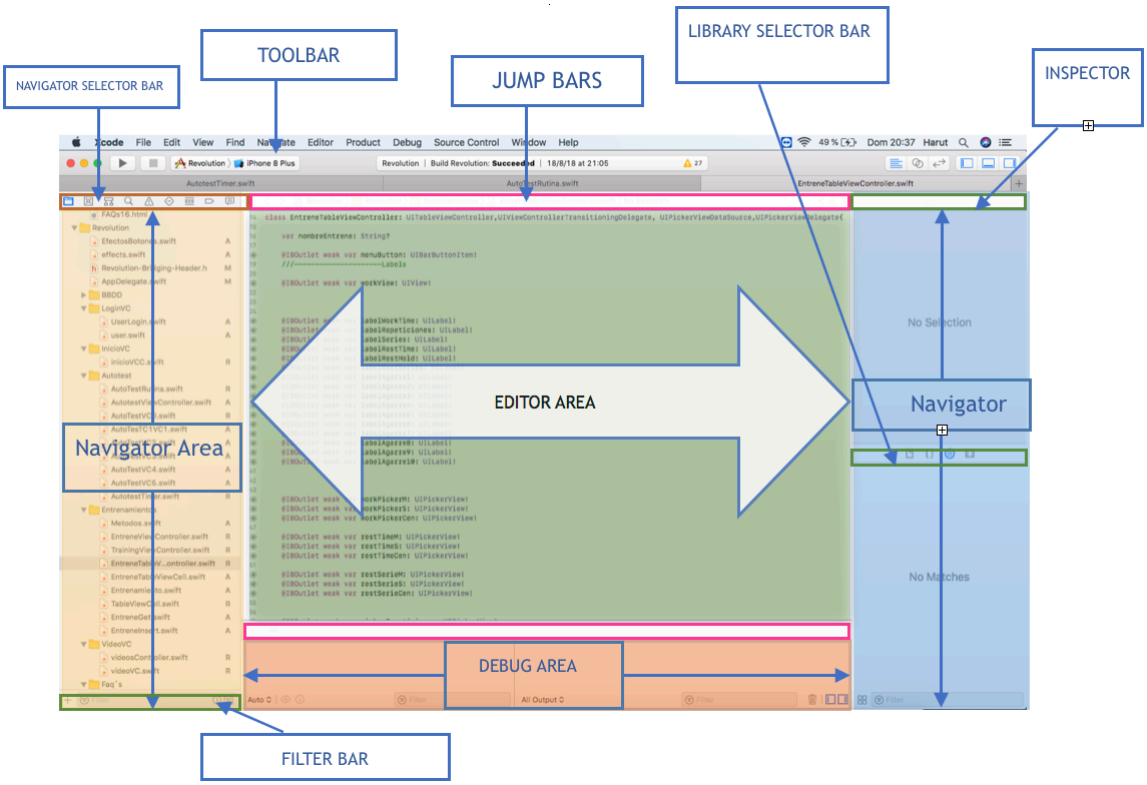

FIGURA 3.1: INTERFAZ DEL ENTORNO DE DESARROLLO

Seguidamente introduciremos las secciones relevantes del entorno, se hará una breve explicación de algunos de los apartados marcados en la Figura 3.1.

Sección de navegación (Navigator Area) :

Como el propio nombre indica en esta sección encontramos los recursos del proyecto, teniendo la posibilidad de acceder a las carpetas y ficheros de nuestro proyecto. La área de navegación del Xcode nos permite organizar nuestro proyecto de la forma deseada.

Es importante destacar que en esta área también aparecen los avisos y los fallos en el programa.

Sección de Utilidades:

En esta área podemos encontrar las propiedades de los diferentes objetos de la interfaz, además podemos modificar las propiedades de las diferentes clases del

proyecto y observar la información disponible de los métodos de las librerías nativas de iOS.

Sección de Edición:

En esta área es donde crearemos nuestro código, la creación de la interfaz gráfica y la configuración de las opciones de la aplicación.

Sección de Depuración:

Por último, en el área de depuración podemos encontrar el valor de las variables a la hora de ejecución. Esta sección es de gran ayuda, se puede observar la causa del error en la compilación.

3.2 Lenguaje de programación

El lenguaje de programación Swift es creado por Apple específicamente para el desarrollo de aplicaciones para iOS y macOS. Swift, que es de código abierto, fue introducido en el año 2014 como lenguaje de programación oficial para aplicaciones iOS y macOS.

Es compatible con el lenguaje de programación “Objective-C” y enfocado a la programación orientada a objetos.

Además, una de las ventajas es que requiere una curva menor de aprendizaje, por eso puede llegara a ser uno de los lenguajes principales para el desarrollo de aplicaciones móviles.

3.2.1 Diseño M-V- C

El diseño Modelo - Vista - Controlador es el recomendado por Apple para el desarrollo de aplicaciones iOS. Como el mismo nombre indica se compone de tres diferentes capas.

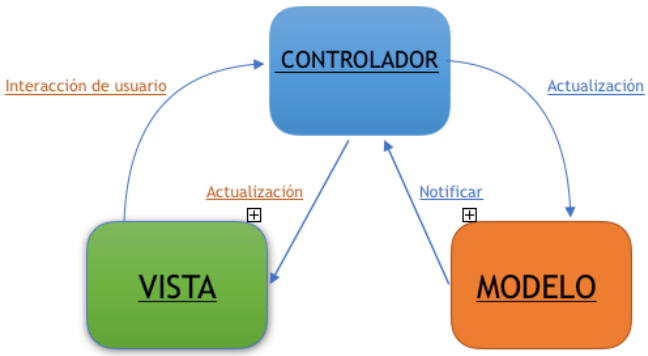

FIGURA 3.2 : ARQUITECTURA MODELO-VISTA - CONTROLADOR

- El modelo es el espacio donde se encuentran los datos con los que operará la aplicación.
- El controlador mediante los delegados interactua entre la vista y el modelo. Por tanto, la interactuación del usuario es invocado mediante peticiones del controlador al modelo y la vista.
- La vista es la capa visible al usuario, uno de los ejemplos es el UIButton, se trata de una vista que representa un Botón en la pantalla.

3.3 Servicio web

La aplicación a desarrollar tiene como objetivo guardar y consultar los resultados de los entrenamientos realizados en una base de datos, por tanto la arquitectura de la aplicación esta basado en el modelo cliente servidor, haciendo uso de un servidor cenralizado.

Mediante peticiones GET de HTTP se han obtenido en formato JSON en la aplicación los parámetros de interés. La actualización o modificación de datos se ha usado la petición POST de HTTP.

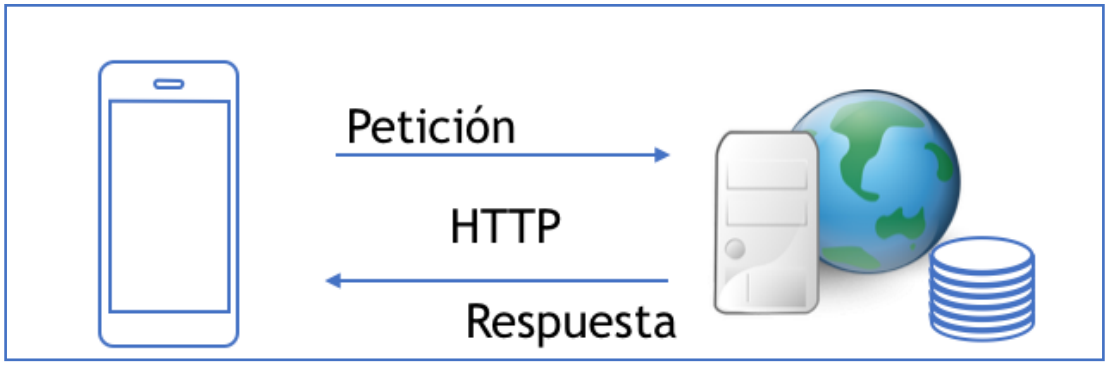

FIGURA 3.3 : ARQUITECTURA DEL SISTEMA

Para alcanzar los objetivos de la conectividad entre el cliente y el servidor, se ha hecho uso del lenguaje PHP (se trata de un lenguaje de código abierto con la posibilidad de gestionar llamadas en los servidores).

Al empezar el proyecto disponiamos implementados la base de datos y los "scripts PHP" para la comunicación con la base de datos, de esta forma disponiamos de dos grupos de scripts PHP, unos para consultar datos y otros para insertar o modificar datos. Uno de los principales "scripts" usados es el "Usuario.php".

```php
<?php
require 'Usuario.php';
if ($_SERVER['REQUEST_METHOD'] == 'GET') {
    if (isset($_GET['id'])) {
        // Obtener parámetro idalumno
        $parametro = $_GET['id'];
        // Tratar retorno
        $retorno = Usuario::getById($parametro);
        if ($retorno) {
            $usuario["estado"] = 1;
            $usuario["usuario"] = $retorno;
            // Enviar objeto json del alumno
            print json_encode($usuario);
        } else {
            // Enviar respuesta de error general
            print json_encode(
                array(
                    'estado' => '2',
                   'mensaje' => 'No se obtuvo el registro'
                      )
            );
        }
    } else {
        // Enviar respuesta de error
        print json_encode(
            array(
                'estado' => '3',
                'mensaje' => 'Se necesita un identificador'
            )
        );
    }
}
```

3.4 Simulador de iOS

Gracias al simulador virtual podemos visualizar la APP creada en cualquier dispositivo iOS, es una ventaja poder visualizar nuestra aplicación en dispositivos con pantallas de diferentes tamaños. En el proyecto se ha utilizado este simulador para poder visualizar la interfaz gráfica, además nos permite comprobar la interactuación del usuario con la aplicación, es importante destacar que el "IDK Xcode" nos permite ejecutar y testar la aplicación en un dispositivo iOS real.

En la parte superior izquierda del IDK Xcode podemos elegir el dispositivo que nos interese, para ejecutar de una forma sencilla disponemos del botón "Run".

FIGURA 3.4: SIMULADOR VIRTUAL DE XCODE

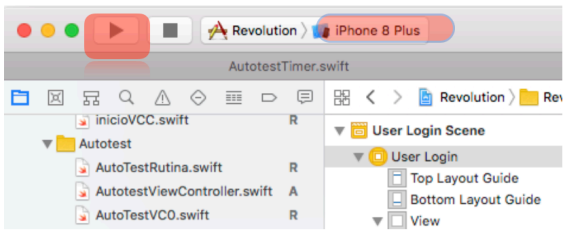

FIGURA 3.5: UTILIDADES DE SIMULACIÓN

4

Diseño y funcionamiento

En este capítulo profundizaremos en el diseño de la interfaz, explicaremos las vistas destacables de la app, así como el funcionamiento de la aplicación con la interactuación del usuario. Consideramos este apartado de gran relevancia para poder entender el funcionamiento por completo desde un punto de vista no muy técnico.

En primer lugar haremos una introducción a cada vista, explicando los elementos que compone la vista, de esta forma una vez conocemos las vistas que componen la aplicación y sus correspondientes denominaciones, pasaremos a elaborar un mapa del funcionamiento de la aplicación respecto a la interactuación del usuario.

4.1 Vista “Autenticación de usuario”

Esta vista es la primera en aparecer al iniciar la aplicación, se trata de la vista de autenticación del usuario.

Cada usuario que desee hacer uso de la aplicación debe de obtener un código de licencia que será otorgada por la empresa distribuidora de la aplicación.

Esta vista corresponde con los requisitos de diseño que previamente han sido acordados.

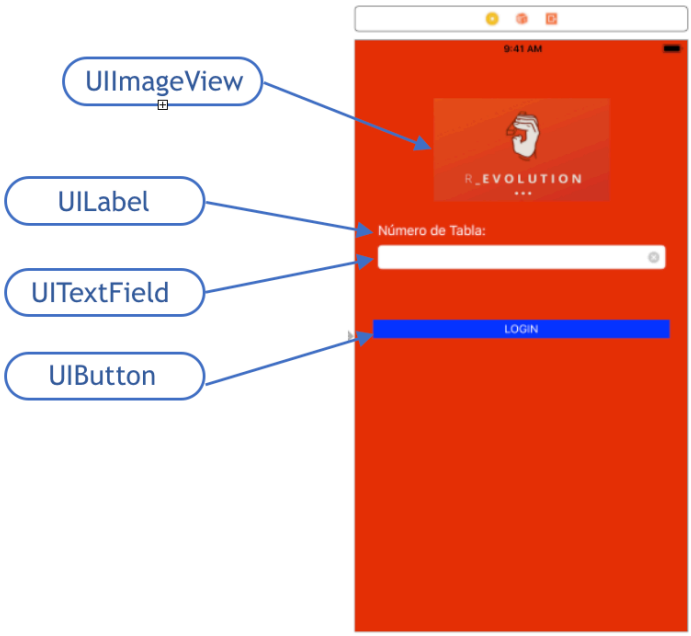

FIGURA 4.1: VISTA DEL LOGIN Y LIBRERIAS UTILIZADAS

Podemos apreciar en la imagen los cuatro elementos que componen la vista. Se ha hecho uso de un "UIImageView" para la incorporación del logotipo en la parte superior.

UIImageview: Esta clase nos permite mostrar imágenes o secuencia de imágenes en la interfaz de la aplicación. Soporta varios formatos de imágenes, entre los cuales los más destacados son "JPEG" Y "PNG". Tenemos la posibilidad de editar la imagen a mostrar, como pueden ser la opacidad o tamaño de éste.

UILabel: Implementa una interfaz para mostrar texto, puede contener texto de tamaño arbitrario siempre y cuando no sobresalga del tamaño de la interfaz creada, en ese caso el texto complementario será cortado o truncado. Esta clase propia de la librería del UIKit, nos permite también editar tamaño y tipo de la fuente.

UITextField: Crea una área de tipo rectangular, cual puede contener texto editable. Cuando el usuario interactua con el espacio creado, aparece el teclado virtual en la pantalla, al presionar "Volver" el teclado virtual desaparece y se crea una variable que contiene el valor de la entrada por teclado.

UIButton: Esta clase del "Framework UIkit" implementa un botón que al interactuar con el usuario ejecuta el código asignado. Nos permite variar aspectos del diseño, como por ejemplo el título del botón, color y otras características de apariencia. En este caso, la acción del botón *Login* es ejecutar la función "servicioWeb()", ésta nos permitirá autentificar el código de licencia introducido por el usuario.

4.2 Vista Inicio

En caso de que el usuario sea autenticado correctamente, la aplicación mostrara al usuario la vista "InicioVC", contiene información básica sobre la tabla "Rokodromo", introduciendo los objetivos principales de la aplicación.

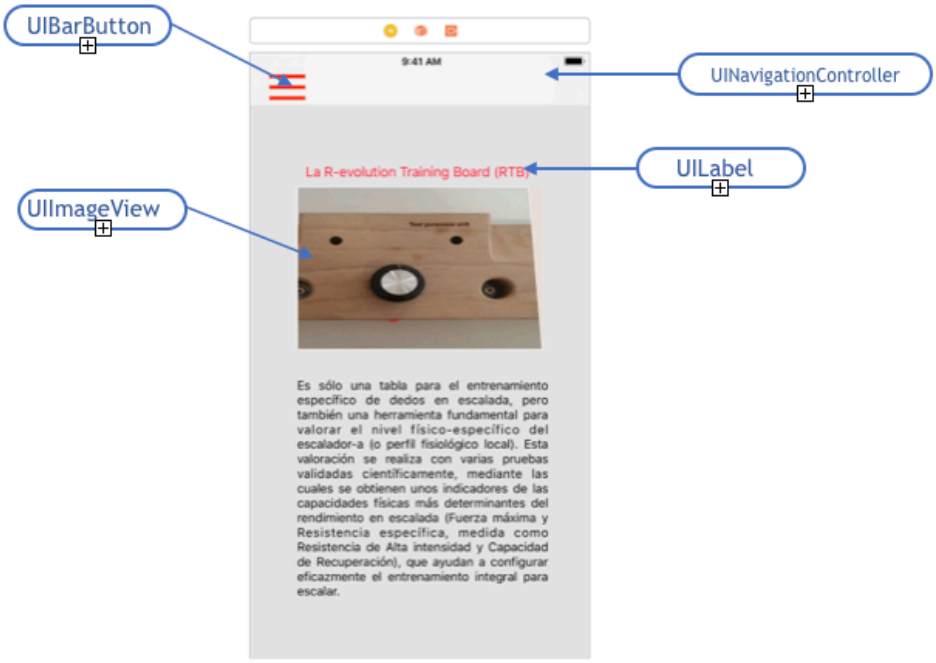

FIGURA 4.2: VISTA DE INICIO Y LIBRERIAS UTILIZADAS

Los elementos principales que componen la vista son: *UIImageView, UILabel, BarButton*, *NavigationView*.

En el apartado anterior "4.1" hemos explicado las funciones de las clases "UILabel" y "UIImageView", por tanto en este apartado nos centraremos en las clases "UIBarButtonItem" y la vista "NavigationView".

UIBarButtonItem: Se puede decir que es muy similar a la clase UIButton, la diferencia consiste en que este tipo de botón se usa para las barras de

herramientas o la barra de navegación de la aplicación. La diferencia con el tipo de botón anterior, es que nos permite hacer uso de unos métodos que nos proporciona la clase "UIBarButtonItem".

UINavigationController: Es una vista que administra la barra de navegación, con lo cual al manejar la aplicación y cambiar de vistas, ésta se actualiza con los elementos correspondientes, por ejemplo el título de la barra de navegación o los botones y funciones disponibles para el caso apropiado.

4.3 Vista "Tabla de entrenamientos"

En este apartado de la aplicación, los usuarios tendrán la opción de crear, guardar y modificar entrenamientos, para ello deben rellenar el campo del "UITextField" con el nombre del entrenamiento deseado y guardar el entrenamiento con el botón "Añadir".

Las celdas del *UITableView* se rellenan con los entrenamientos previamente creados y guardados por el usuario.

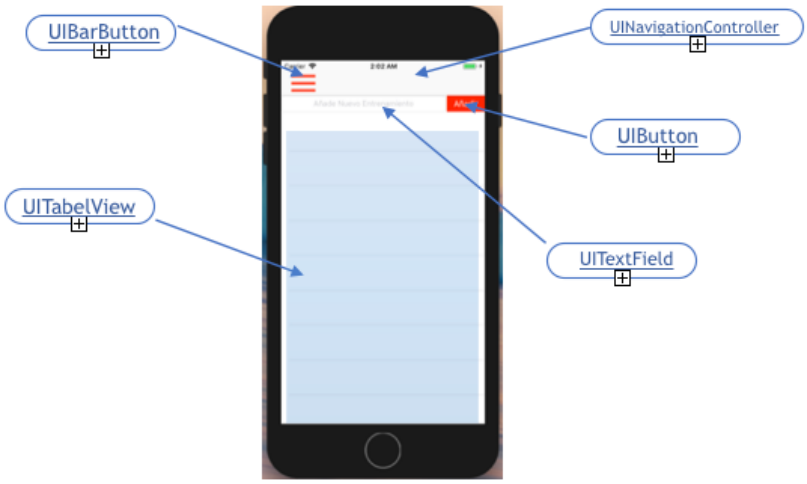

FIGURA 4.3: VISTA DE LA TABLA DE ENTRENAMIENTOS

4.4 Vista “Editor de entrenamientos”

Después de crear un entrenamiento, la aplicación nos mostrará la vista del “TrainingViewController”, desde las opciones de la vista podemos ajustar los parámetros del entrenamiento, también será posible editar los parámetros de los entrenamientos creados previamente.

FIGURA 4.4: VISTA DEL EDITOR DE ENTRENAMIENTOS

Al finalizar la edición de los parámetros del entrenamiento, el usuario podrá guardar su entrenamiento personalizado, pulsando el botón “Done”

Se trata de una vista compuesta por una “UITableViewController” con dieciséis celdas expansibles, al interactuar con cada celda, esta se expande y muestra los elementos que contiene.

Al elegir los tipos de agarre o los tiempos de descanso y entrene, estos parámetros también se reflejan al cerrar la celda.

FIGURA 4.5: CELDA EN FORMATO CERRADO

UITableViewCell: Esta clase dispone de propiedades y métodos para editar, configurar y administrar el contenido de la celda

Al crear celdas, tenemos la opción de personalizar o usar los estilos ya definidos.

UIPickerView: Este tipo de vistas son bien conocidas por los usuarios de "smartphones". Una de las ventajas de usar este tipo de vistas como selectores, es el ahorro de espacio y un diseño atractivo.

El "*UIPickerView*" funciona como una rueda giratoria, el usuario manipula este elemento para seleccionar la opción deseada.

FIGURA 4.6: VISTA DE LA CELDA EN FORMATO EXPANDIDO

En este caso hemos enlazado el selector del "UIPickerView" con la imagen del agarre y el texto en la parte superior de la celda.

De esta forma al interactuar y girar el selector del "UiPickerView" el usuario observa el cambio de las imágenes y el agarre elegido.

4.5 Vista "FAQ´S"

En esta sección se ha creado una vista con el fin de responder a las dudas frecuentes de los usuarios con toda la información relevante para un uso correcto de la aplicación. Esta vista es semejante a la del "TrainingViewController", pero en vez de ser celdas desplegables, al interactuar el usuario en lugar de tratarse de celdas desplegables, nos dirigen a una vista diferente para mostrar el contenido, en la cual se cargan ficheros de tipo "HTML". La visualización de este tipo de archivos se hace posible gracias a la clase "UIWebView", cual crea una vista para incorporar contenido web.

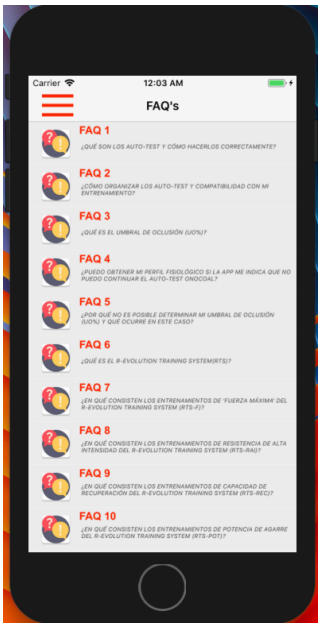

FIGURA 4.7: VISTA FAQ'S

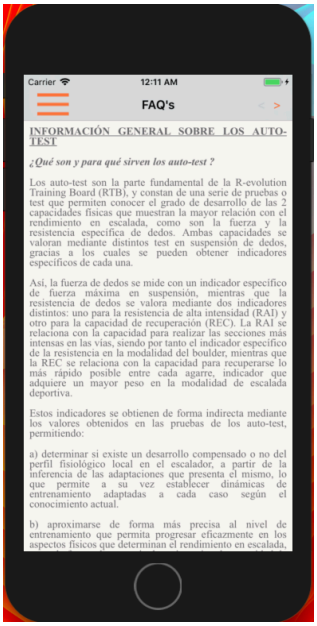

FIGURA 4.8: VISTA AL SELECCIONAR UNA PREGUNTA

4.6 Vista "TimerVC"

Se trata de una de las vistas más relevantes de la aplicación. En este apartado el usuario realizará sus entrenamientos, para ello dispone de un contador de tiempo que estará en funcionamiento cuando el usuario este suspendido en la tabla, es decir, la tabla fabricada por la empresa "Euroholds Climbing" estará conectada al dispositivo móvil del usuario, de esta forma la aplicación controla la actividad del deportista.

Esta es la vista con la cual el usuario entrena, donde se le indicarán las pautas del entrenamiento y el tipo de agarre.

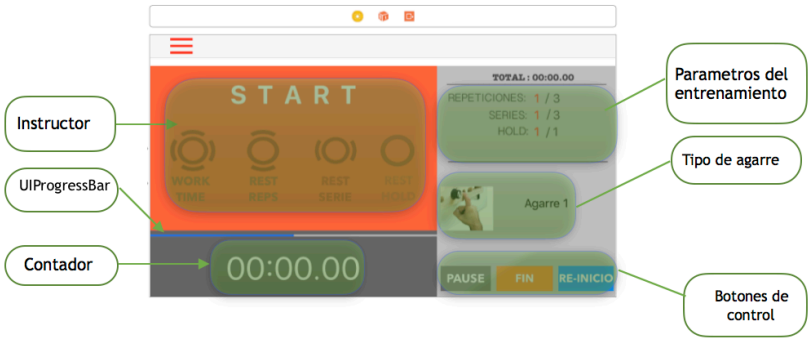

FIGURA 4.9: VISTA DEL ENTORNO DE ENTRENAMIENTO

4.6.1 Funcionamiento del "TimerVC"

En este apartado estudiaremos los elementos que componen la vista. Anteriormente hemos explicado las clases utilizadas con el propósito de no volver a repetirlos, nos centraremos más en la funcionalidad dejando claro la forma de efectuar un entrenamiento.

Al realizar la suspensión en la tabla "Rokodromo", el contador empieza la cuenta atrás, dependiendo de la cantidad de repeticiones y series se efectuarán los descansos previamente programados.

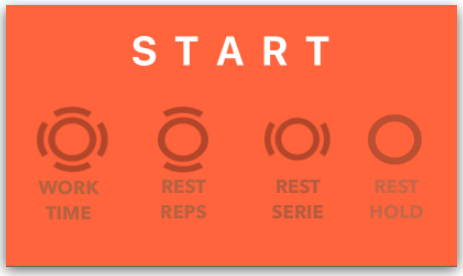

FIGURA 4.10: INDICADOR DE ACCIÓN DEL ENTRENAMIENTO

Como ejemplo, vamos a suponer un entrenamiento que tenga tres repeticiones, tres series y dos agarres:

FIGURA 4.11: PANEL LATERAL DEL ENTRENAMIENTO

Después de efectuar cada repetición se activa el contador del "tiempo de descanso", al finalizar las tres repeticiones el deportista pasará a la segunda serie, de esta forma cada serie tendrá tres repeticiones.

Al complementar las tres repeticiones y sus correspondientes descansos, el usuario pasará al segundo tipo de agarra, por lo tanto el contador del "Hold" se incrementara en uno y si el usuario ha elegido otro tipo de agarre, se le indicará el tipo de agarre mediante una imagen.

4.7 Menú lateral desplegable

En la mayoría de las vistas de la aplicación el usuario dispone de un botón para acceder directamente al menú desplegable de la aplicación. El menú contiene los apartados descritos anteriormente.

En caso de desear cerrar sesión de la aplicación, en el menú lateral, la última opción "Salir" cerrará la sesión y redirigirá al "ViewController User login".

Gracias a este menú lateral, se mejora de forma notable la experiencia de usuario.

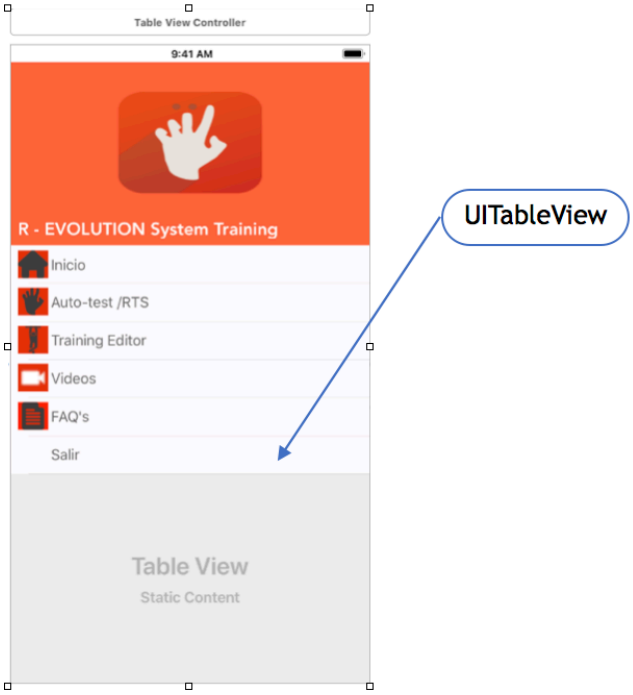

FIGURA 4.12: VISTA DEL MENÚ LATERAL

4.8 Mapa de funcionamiento de la app

En este mapa intentaremos plasmar el funcionamiento de la aplicación con la interactuación del usuario, este apartado es clave para entender como son los saltos entre las diferentes vistas de la app. Podemos separar las vistas en dos grupos:

- Vistas informativas: INICIO, VIDEOS y FAQ'S. Estas tres vistas informan y guían al usuario, con videos explicativos y respuestas a las preguntas más frecuentes.
- Vistas de entrenamiento: TRAINING EDITOR y AUTOTEST, en ambas vistas el usuario podrá efectuar sus entrenamientos, predeterminados o propios.

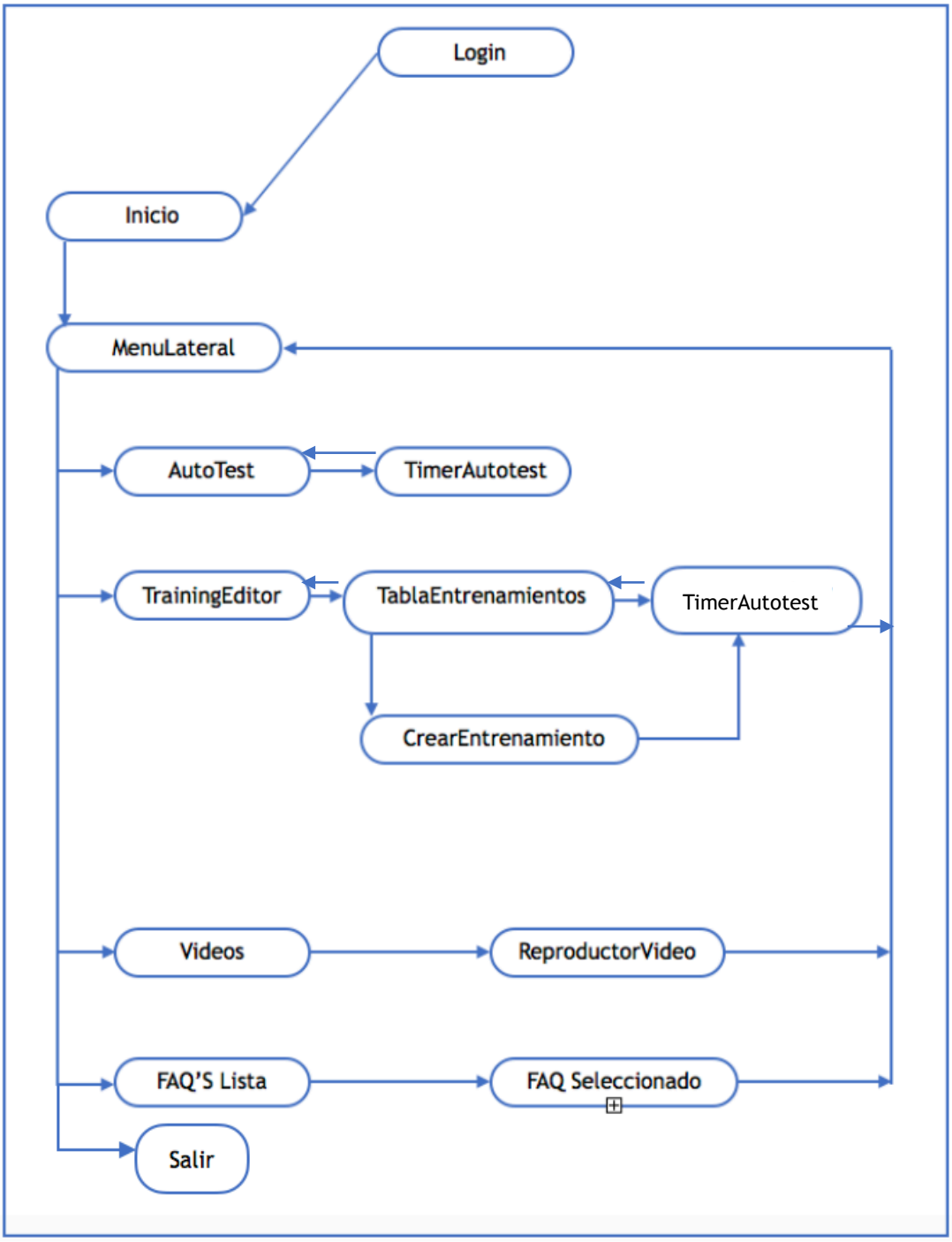

FIGURA 4.13: MAPA DE FUNCIONAMIENTO

5

Implementación

En el presente capítulo presentaremos como se ha realizado la implementación de la aplicación desde un punto de vista técnico. Por consiguiente se verán los métodos implementados y la funcionalidad que proporcionan en la aplicación. Asimismo reflejaremos las dificultades que hemos tenido y la solucione aplicada para cada caso.

5.1 Login del usuario

En primer lugar se ha creado y configurado la vista. Para poder hacer una autenticación del usuario necesitamos un valor de entrada, es el número de la licencia que el usuario dispone.

Para ello se ha hecho uso del "UITextField". En primer lugar se ha creado la conexión desde la vista a la clase "userLogin" y se ha creado una variable llamada "TextFieldLabel" de tipo UITextField.

```
@IBOutlet weak var textFieldLabel:
UITextField!
```

Para poder extraer el valor que contiene esta variable:

```
var codigo_tabla = textFieldLabel.text
```

Por consiguiente, la nueva variable "codigo_tabla" de tipo "String" tendrá el valor de entrada introducido por el usuario.

5.1.1 Función "ServicioWeb"

Esta es la función que se encarga de hacer una consulta a la base de datos, al obtener una respuesta afirmativa (*true*) se efectúa el inicio de sesión y además se obtienen los datos del usuario en caso de que no sea un usuario nuevo.

En resumen, se ha creado una variable que recoge el valor de la licencia introducida por el usuario, seguidamente creamos la variable "myUrl" de tipo "URL" y concatenamos la dirección "URL" del archivo PHP con el valor de la licencia obtenida mediante el UITextField.

```
let myUrl = URL(string: "https://
es.rokodromo.com/RevolutionBoard/
 usuario_obtener_por_codigo.php?
codigo_tabla=\(textFieldLabel.text!)")
```

Este URL nos devolverá un archivo “JSON” , para ello el lenguaje de Swift dispone de una clase llamada “URLSession” cual nos va a permitir descargar el archivo con los datos.

```
let json = try
JSONSerialization.jsonObject(with: data!,
    options: .mutableContainers) as?
NSDictionary
```

A continuación, para gestionar la llamada al URL creado, hacemos uso de la función “dataTask”.

```
let task = URLSession.shared.dataTask(with:
request) { (data: Data?, response: URLResponse?,
error: Error?)
```

En caso de que la licencia no sea correcta, obtendremos un error y al usuario se le mostrará una alerta que indica el error ocurrido:

```
if error != nil {

self.displayMessage(userMessage: "Ups!
Revise su código de  tabla")
print("error=\
(String(describing:error))")

return
```

Por último, al conectarse correctamente, usaremos la clase “JSONSerialization” para convertir los datos obtenidos al tipo de variable con el que trabajaremos en el proyecto.

Creamos un diccionario en Swift que nos permite almacenar datos con el formato de “ clave - valor”:

```
 var userDict =  [String:String]()

         userDict = parseJson["usuario"] as! [String :
String]


           myUser = Usuario(id: userDict[“id"]!,
            imei: userDict["imei"]!,
             codigo_tabla: userDict["codigo_tabla"]!,
             idioma: userDict["idioma"]!,
             fecha_primer_uso: userDict["fecha_primer_uso"]!,
             experiencia_entrenamiento:
             userDict["experiencia_entrenamiento"]!,
             altura: userDict["altura"]!,
             peso: userDict["peso"]!,
             temperatura: userDict["temperatura"]!,
             falange: userDict["falange"]!,

nivel_deportiva_ensayado:userDict["nivel_deportiva_ensayado"
]!,
             nivel_deportiva_vista:
userDict["nivel_deportiva_vista"]!,
             nivel_bloque_ensayado:
userDict["nivel_bloque_ensayado"]!,
             boulder_deportiva:
userDict["boulder_deportiva"]!)
```

En suma, obtenemos los valores necesarios para la estructura “myUser”, cual incorpora los parámetros del usuario.

5.1.2 Botón “Login”

La función “ServicioWeb” explicado en el apartado anterior se ejecuta al presionar el botón *Login*, para ello se ha hecho uso de la clase *UIButton*.

```
@IBAction func userLogin(_ sender: Any) {
        print ("button tapped")

        servicioWeb()

           performSegue(withIdentifier:
"goInicio", sender: self)

  }//botón login
```

En el caso de que se ejecute correctamente la función *servicioWeb()*, el siguiente comando a ejecutar es el método "perfomSegue", cual nos dirige a la vista siguiente con el identificador "goInicio".

5.2 Vista "TimerVC"

Ya que la aplicación se centra en el entrenamiento de precisión y fuerza física, los entrenamientos son cronometrados.

FIGURA 5.1: VISTA DEL CONTADOR

Este cronometro consta de tres variables: minutos, segundos y centésimas de segundo, con el orden correspondiente.

5.2.1 Creación del cronómetro

En primer lugar vamos a crear un cronometro descendiente con un margen de un segundo por cada descenso, para ello hacemos uso de la clase de la librería nativa del Apple "NSTimer", esta clase nos permite ejecutar un código en intervalos determinados, es de gran uso para actualizar contenido automáticamente.

```
 func runTimerWork() {


        timerWork = Timer.scheduledTimer(timeInterval:
0.01, target: self, selector:
#selector(TrainingViewController.updateTimerWork),
userInfo: nil, repeats: true)
        tiempo_trabajo = convertTime(entrada:
tiempoTrabajoString)

      }
```

De esta forma, en cada intervalo de tiempo hacemos una llamada a la siguiente función, cual ejecuta el comando :

```
func updateTimerWork() {



    viewEstados.backgroundColor = UIColor.orange
    workTime.textColor = UIColor.white
    restReps.textColor = UIColor.brown
    restSeries.textColor = UIColor.brown
    labelAccion.text = "W O R K"
    ImageWork.image =  imageLiteral(resourceName:
"work_on")
    ImageRestReps.image =  imageLiteral(resourceName:
"rest_off")
    ImageRestSeries.image =  imageLiteral(resourceName:
"series_off")

    tiempo_trabajo -= 1
    labelTimer.text = convertTimeToString(entrada:
tiempo_trabajo)
```

5.2.2 Método “ConvertTime”

Cuando el usuario programa su entrenamiento, elige en el “PickerView” los parámetros de tiempo, es decir selecciona un tiempo de descanso y tiempo de entrenamiento, estos valores se guardan en la base de datos con el formato “String”, por tanto al obtener estos datos los obtenemos en un formado que no puede ser usado directamente para hacer la cuenta atrás del cronometro, ya que nos hacen falta valores de tipo entero.

Esta función se ha creado para obtener el tiempo total en centésimas de segundo.

```
   func convertTime (entrada: String) -> Int{ // entrada 00-00-00

        var temp =  entrada.split(separator: "-", maxSplits: 3)  //
separaos el formato ""-""-""
        var tiempoConvertidoToInt =  Int( temp[0] )! * 6000 +
Int(temp[1])! * 100 + Int(temp[2] )!


        return tiempoConvertidoToInt
    }
```

Hacemos uso del método “split” de Swift, por tanto podemos separar la cadena de entrada.

5.2.3 Método “ConvertTimeToString”

Por otra parte, también prescindíamos de una función que convierta un entero a un “String” con formato “00:00.00” para mostrar al usuario en el ViewController “TrainingEditor”.

```
 func convertTimeToString (entrada: Int) -> String {

        var min = ""
        var seg = ""
        var cen = “"

        var minutos : Int = Int(entrada/6000)
        var segundos : Int = (entrada - minutos * 6000) /
100
        var centesimas : Int = Int(entrada)%100


        if  minutos < 10  {
            min = "0\(minutos)"

        }

        else  {
            min = "\(minutos)"
        }

        if  segundos < 10  {
            seg = "0\(segundos)"
        }
        else {
           seg = "\(segundos)"
        }
        if  centesimas < 10  {
            cen = "0\(centesimas)"

        }

        else {
        cen = "\(centesimas)"
        }

        var tiempo = "\(min):\(seg).\(cen)"
        return tiempo
    }
```

Como se puede observar en el código, hemos obtenido matemáticamente los valores de minutos, segundos y centésimas.

Recordemos que vamos a obtener un valor entero que representa en centésimas de segundo el total del tiempo, por tanto de la cuantía de minutos lo podemos obtener dividiendo el valor total de la entrada por "6000", ya que un minuto corresponde a 6000 centésimas de segundo.

Para obtener la cuantía de segundos, en primer lugar restamos del valor total de la entrada los minutos y lo dividimos por 100, ya que un segundo son 100 centésimas de segundo, por último hacemos uso del operador aritmético de resto en Swift.

```
var minutos : Int = Int(entrada/6000)

var segundos : Int = (entrada - minutos
* 6000) / 100

var centesimas : Int = Int(entrada)%100
```

Al efectuar el resto sobre el valor total, obtenemos las centésimas de segundo.

Como ejemplo podemos seleccionar el entero 7000:

```
minutos: 7000/6000 = 1.16
```

Para truncar los valores hacemos uso del variable de tipo entero, por tanto el resultado seria "1".

```
segundos: (7000 - 1 * 6000) / 100  = 10
```

Por último obtenemos las centésimas con el operador del resto:

```
(entrada)%100 = 70
```

En conclusión, el resultado final será : 01-10-70.

5.3 Realización de entrenamientos

En este apartado explicaremos el funcionamiento en conjunto del "Timer" y el entrene programado. Para ello supondremos un ejemplo de entrenamiento donde se reflejan los parámetros del entrenamiento.

NOMBRE ENTRENE.	REPETICIÓN	TIEMPO TRABAJO	DESCANSO REPETICIÓN	SERIES	DESCANSO SERIES	AGARRES	DESCANSO AGARRES
TFG	2	*00-30-00*	*00-05-00*	2	*00-15-00*	2	*01-00-00*

TABLA 5.1: PARÁMETROS DE UN ENTRENAMIENTO

En resumen, cuando el usuario comienza la suspensión en la tabla, el botón situado en la tabla se presiona, por consiguiente el entrenamiento empieza con la serie uno, al finalizar cada repetición tendremos los descansos de la repetición hasta completar la serie actual, en ese caso se salta a la siguiente serie con los valores de repetición inicializados.

Al finalizar cada serie el deportista dispone de unos descansos que se denominan "Rest Serie", cada agarre se finalizará al completar las series y las repeticiones correspondientes y se efectuará un descanso llamado "Rest Hold", este intervalo de tiempo permite al deportista cambiar de agarre. Al completar todos los agarres se finalizará el entrenamiento.

Como hemos indicado en los apartados anteriores, para crear el entrenamiento, en primer lugar se introducirá el nombre del nuevo entrenamiento en la vista **"EntrenamientoVC"** y se creará el entrenamiento en la base de datos al apretar el botón "Añadir", además esta acción nos redirige al **"TrainingEditor"** donde podemos configurar los parámetros del entrenamiento.

5.3.1 Diagrama de estados del entrenamiento

En el siguiente diagrama mostraremos los estados en cada situación del entrenamiento, como habíamos mencionado con anterioridad, los entrenamientos se componen de:

Inicio: estado inicial, el usuario aun no ha iniciado la suspensión en la tabla.

WorkTime: el usuario ha empezado la suspensión y el contador se ha iniciado.

Rest: tiempo de descanso después de cada repetición.

RestSerie: tiempo de descanso al completar una serie.

RestHold*:* tiempo de descanso al finalizar el tipo de agarre seleccionado.

FinEntrene: la aplicación pasa a este estado cuando el usuario ha completado todos los agarres con sus respectivas series.

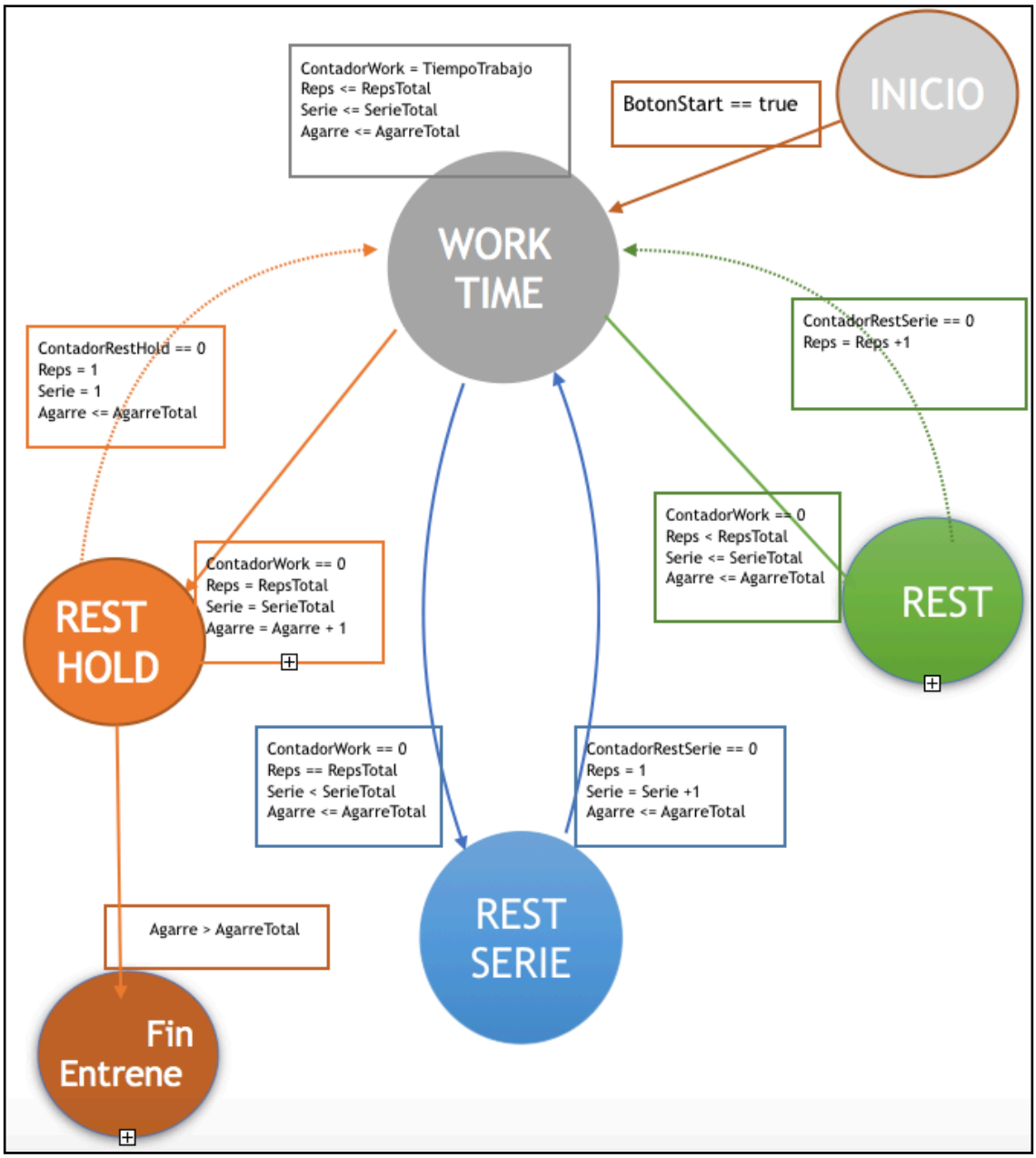

FIGURA 5.2: DIAGRAMA DE ESTADOS

5.4 Conectividad entre la tabla y el dispositivo móvil

En el diagrama de estados apreciamos que el entrenamiento empieza cuando el valor del "*BotonStart"* es "true", es decir, para este botón se ha creado una variable de tipo "*Booleano"*, esto nos ayuda determinar si el usuario esta efectuando la suspensión en la tabla.

Para poder determinar el valor del "*BotonStart"*, se ha optado por una opción sencilla, se ha conectado el dispositivo a la tabla mediante el puerto "auxiliar IN", haciendo uso del conector de tipo "jack 3.5 mm".

Al estar conectado la tabla al iPhone, podemos percibir cada vez que se presione el botón en la tabla, ya que esta acción se refleja en el iPhone como el presionado del botón superior de volumen.

5.4.1 Dificultades

La marca Apple prohibe a los desarrolladores acceder directamente a los botones del "hardware", por tanto en caso de efectuar alguna función que modifique o acceda a las funcionalidades de los botones, la aplicación será eliminada de la plataforma de "AppStore", esta plataforma creada por Apple proporciona el servicio de publicación y descarga de aplicaciones para todos los sistemas operativos de Apple.

FIGURA 5.3: BOTÓN DEL IPHONE QUE INTERACTUA CON LA APP

5.4.2 Solución a la restricción

La solución proporcionada a este problema ha sido crear una función que capte la subida de volumen.

Por tanto, al presionar el botón de volumen o el botón conectado mediante el puerto “auxiliar IN” nos mostrara por pantalla la subida de volumen:

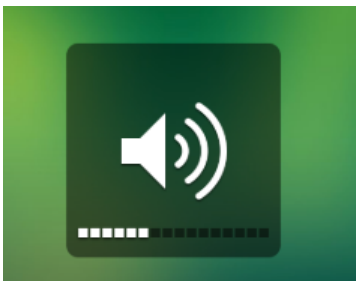

FIGURA 5.4:CAPTACIÓN DE VOLUMEN DEL IPHONE

Aunque el volumen del dispositivo sea máximo, se capta la alerta creada para subir el volumen.

Al capturar esta alerta, el valor de la variable “BotonStart” será igual a “true” y entonces el contador del “*timer*” empieza a efectuar el entrenamiento. La función encargada de efectuar esta acción es “EscucharBotonVolumen()”.

La clase “AudioSession” de Apple actúa como un intermediario entre el sistema operativo, el hardware y la aplicación.

La variable “outputVolume” es de tipo “float” y tiene un valor entre “0.0 - 1.0”, de esta forma se expresa la variación de volumen en el dispositivo.

```
func escucharBotonVolume() {
        do {
            try audioSession.setActive(true)
        } catch {
            print("error")
        }
        audioSession.addObserver(self, forKeyPath: "outputVolume", options:
NSKeyValueObservingOptions.new, context: nil)
    }

    override func observeValue(forKeyPath keyPath: String?, of object: Any?,
change: [NSKeyValueChangeKey : Any]?, context: UnsafeMutableRawPointer?) {
        if keyPath == "outputVolume" {

            BotonStart = true

        }
    }
```

6

Conclusión y futuros proyectos

Se ha logrado realizar el proyecto propuesto por “Euroholds Climbing” con los requisitos previamente especificados. Hemos desarrollado una aplicación para dispositivos móviles con sistemas operativos iOS, ésta primera versión de la aplicación es capaz de proporcionar opciones para crear entrenamientos,

además realizar unas pruebas ya programadas. La aplicación desarrollada se ha validado con diferentes dispositivos móviles de la marca Apple.

Antes de empezar el desarrollo se ha realizado un análisis técnico, se han definido objetivos y soluciones a posibles problemas que pueden surgir.

Desde nuestro punto de vista, queda por completar en el ámbito técnico de conectividad entre la tabla y la app, como hemos indicado con anterioridad tanto el diseño como la funcionalidad han sido sugeridos por el cliente.

La opción de conectividad inalámbrica propuesta por nosotros será estudiado por el cliente para futuros proyectos.

En definitiva, me gustaría destacar la satisfacción por haber realizado este proyecto final de grado. Sin duda el mejor premio es la oportunidad y esa oportunidad es la que he tenido para trabajar en un proyecto real, crear una aplicación cuyo uso será tanto para deportistas profesionales como para aficionados. Interactuar con el cliente desde un ámbito tanto técnico como empresarial, poder llegar a acuerdos en diferentes etapas y mantener un ambiente de confianza y seriedad.

Actualmente ya se han creado acuerdos para **futuros proyectos** con "EuroHolds", la nueva versión a desarrollar incluirá mejoras en la conectividad entre la tabla y el dispositivo móvil, añadirá opciones de estadística y gráficas, además se ha visto la necesidad de crear una versión "Freemium", es decir se bloquearan algunas opciones de la aplicación y únicamente serán desbloqueados cuando el usuario realice la suscripción, por tanto se añadirán opciones de pago a la aplicación.

Bibliografía

[1] THE IPHONE DEVELOPER'S COOKBOOK: BUILDING APPLICATIONS WITH THE IPHONE 3.0 SDK (2ND EDITION)

[2]ADMINISTRACIÓN DE BASES DE DATOS DISEÑO Y DESARROLLO DE APLICACIONES - MICHAEL V. MANNINO

[3] APPLE INC. IOS DEVELOPER LIBRARY. *TOOLS FOR IOS DEVELOPMENT*
https://developer.apple.com

[4] CONSULTAS CONTINUAS AL FORO:
https://stackoverflow.com/

[5] CURSO COMPLETADO EN LA PLATAFORMA DE MOOC'S UDEMY: APRENDE SWIFT 4 PARA IOS Y LO MEJOR EN BASES DE DATOS

[6] BEGINNING IOS STORYBOARDING: USING XCODE BY RORY LEWIS

[7] PAGINA OFICIAL DE LA PLATAFORMA DE DESARROLLO APPLE
https://itunes.apple.com/