Desarrollo de una aplicación iOS de comunicación de aplicaciones de salud con glUCModel

César Abades Ruiz

Departamento de Ingeniería del Software e Inteligencia Artificial

Facultad de Informática

Universidad Complutense de Madrid

Septiembre 2018

Desarrollo de una aplicación iOS de comunicación de aplicaciones de salud con glUCModel

Dirigido por

Jose Ignacio Hidalgo Pérez

Departamento de Ingeniería del Software e Inteligencia Artificial

Facultad de Informática

Universidad Complutense de Madrid

Septiembre 2018

Capítulo 1

Resumen y palabras clave

En este documento se explican todos los aspectos relacionados al desarrollo de una aplicación iOS, que se comunica con iOS Health con el objetivo de subir toda la información relevante sobre la diabetes a la base de datos de glUCModel. Se abarcará desde el inicio de la idea y el objetivo, hasta los detalles más técnicos como las tecnologías utilizadas.

In this document are explained all the aspects related to the development of an iOS communication app that communicates with iOS Health in order to upload all the relevant information about diabetes to glUCModel database. It will cover many aspects going from the birth of the idea and the goal to the more technical details, such as all the technologies i have used.

Palabras clave: iOS, HealthKit, iPhone, Visual Studio Code, glUCModel, Ionic

Key words: iOS, HealthKit, iPhone, Visual Studio Code, glUCModel, Ionic

Índice

Capítulo 2

Introducción

2.1. Un mundo cambiante

La tecnología se ha vuelto indispensable en todos los ámbitos de la sociedad. Pocos son ya los escenarios en los que no encontremos un elemento tecnológico que mejore nuestra gestión, nos haga más productivos, nos otorgue una mayor calidad de vida o nos permita hacer cosas que décadas atrás ni siquiera soñábamos. Es una realidad: la sociedad se ha vuelto tecno-dependiente. Comunicaciones, salud, entretenimiento y muchos otros campos han avanzado sobremanera gracias a esta disciplina de la ciencia.

Sin embargo, este avance global impulsado por la tecnología, nos ha dejado en una situación comprometida, y es que las personas necesitan sumergirse cada vez más en ese complejo e inmenso mundo para poder aprovechar todas las ventajas que ofrece.

Es por ello que nosotros, como desarrolladores y pro-

veedores de servicios informáticos, encontramos entre nuestros deberes acercar la tecnología a la población de una manera sencilla, intuitiva y cómoda, sobre todo a las personas que realmente lo necesitan, personas cuyas vidas podemos mejorar poniendo nuestro granito de arena.

En este escenario, acometeremos el caso de las personas diabéticas. Y es que el crecimiento de la población lleva consigo un aumento proporcional en el número de diabéticos, por lo que cada vez hay un mayor número de personas cuya vida podemos mejorar. Y es en esto en lo que nos vamos a centrar.

2.2. glUCModel y la diabetes

Las personas diabéticas deben mantener un estricto control de sus valores de azúcar en sangre, evitando los problemas que se derivan de la enfermedad, es decir, las hipoglucemias e hiperglucemias. Este control se basa en multitud de parámetros, y es que hay muchos factores que afectan a la glucemia entre los que se encuentran por ejemplo el peso, la comida ingerida, o el deporte realizado entre otros. Una vez conocidos los valores asociados a esto factores, los pacientes deben realizar una estimación de la insulina que necesitan.

Sin embargo, incluso para personas ya veteranas en su lucha interminable contra la diabetes, es muy difí-

cil realizar estimaciones correctas y esta ardua labor se vuelve imposible sin el control ya mencionado. En este punto es donde entra en juego glUCModel.

glUCModel, plataforma ya conocida por muchos, lleva desde sus inicios ejerciendo una labor de seguimiento, control y predicción de la diabetes. Esta tarea, aunque ambiciosa, requiere una cantidad ingente de datos, ya que en su afán por ser precisa, tanto el volumen como la calidad de los datos juegan un papel crucial.

En el momento de su concepción, glucModel nació como una herramienta de control y predicción de la glucemia de los pacientes, apoyándose en las variables anteriormente mencionadas. De esta manera, es posible informar al paciente de cuál es la cantidad de insulina que necesita para mantener el nivel de azúcar en sangre dentro de los valores necesarios.

2.3. Acercándonos al usuario

Como decía, uno de los problemas a los que se enfrenta glUCModel a la hora de realizar predicciones en su día a día, es la necesidad de mantener los datos de sus pacientes actualizados, de manera que estas predicciones sean cada vez más precisas. De esta necesidad, surgió en primera instancia la aplicación web desde la que se puede realizar por ejemplo una subida de datos manual a través de archivos, de los cuáles se extrae la

información y se almacena en la base de datos, de forma que el paciente puede consultarla en todo momento. Esto soluciona el problema... en parte.

No siempre estamos delante del ordenador para poder subir información a la aplicación web. Las situaciones tan diversas por las que pasa una persona a lo largo del día, hacen necesarias otras fuentes de información que permitan enviar los parámetros relevantes, fuentes de información que los pacientes lleven siempre consigo. Y entre las pocas cosas que llevamos siempre encima, aparece una que, si bien no lo estamos mirando a todas horas, lo llevamos siempre encima: el teléfono móvil.

Con esto en mente, se han desarrollado ya algunas aplicaciones móviles que han permitido mejorar la labor de glucModel. Sin embargo, la estrategia de estas aplicaciones era la de que el usuario pudiese introducir manualmente los datos, para posteriormente subir la información a la base de datos y poder explotarla.

No obstante, aunque esto cumple con el objetivo y soluciona el problema, existen algunas cuestiones que hacen que estas aplicaciones no aprovechen todo el potencial que existe a día de hoy en el mundo tecnológico:

- El usuario debe añadir la introducción de datos en una aplicación, que no usará para nada más, a sus tareas diarias
- El histórico de la información que se puede subir,

empieza el mismo día de la instalación de esas aplicaciones, limitando así el rango de tiempo al que tiene acceso glUCModel para poder realizar las predicciones de manera correcta.

- Las aplicaciones nativas, para Android e iOS, tienen muchas ventajas. Sin embargo, el coste de su mantenimiento se multiplica por dos debido a que son aplicaciones completamente independientes.

2.4. iOS Health

En su afán por ejercer cada día una mejor labor, tanto a nivel operacional como tecnológico, y habiendo identificado los puntos con mayor margen de mejora, glUCModel estudió las posibles opciones que existían para poder ponerle fin a las limitaciones que arrastraban las anteriores aplicaciones, al menos en una gran cantidad de casos. Es aquí donde entra en escena la aplicación Health para iOS.

Desde su anuncio en la Worldwide Developers Conference (WWDC) de 2014, la aplicación Health pasó a formar parte de las nuevas versiones de iOS, concretamente desde la versión 8. Además, iOS Health no vino sola, sino que vino acompañada por HealthKit, la interfaz que permite la comunicación con aplicaciones de terceros, otorgando a dichas aplicaciones el acceso a todos los datos alojados en Health, por supuesto siempre

con los permisos del usuario.

Es en este punto en el que glUCModel advirtió que las dos primeras limitaciones ya mencionadas podían quedar resueltas solo aprovechando las bondades de HealthKit:

- Los usuarios que utilicen la aplicación Health para hacer un seguimiento de su estado de salud, constantes vitales y actividad física, no tendrían que volver a introducir de nuevo esos datos solo para la aplicación de glUCModel, ya que se podría utilizar HealthKit para obtener todos los datos y subirlos a la base de datos simplemente pulsando un botón.
- El histórico cambiaría su punto de partida del momento de instalación de la aplicación móvil de glUCModel, a 2014 potencialmente, o al momento en que el usuario haya decidido empezar a utilizar iOS Health.

Como se ha hecho notar en los anteriores párrafos, solo nos estamos centrando en iOS. La razón es que Health viene preinstalado en iOS, por lo que todos los usuarios que adquieran un iPhone tendrán acceso inmediato a dicha aplicación, sin necesidad de instalarla, lo que supone un extra de comodidad respecto a los usuarios en Android, ya que no en todos los *smartphones* con este sistema operativo vienen con una aplicación preinstalada, y no siempre vendrá con la misma aplicación de

salud.

No obstante, a pesar de que solo se haya abordado la versión para iOS, la nueva aplicación de glUCModel ha sido desarrollada como aplicación híbrida, por lo que la integración con Android no supone crear una nueva aplicación, si no adaptar la comunicación con HealthKit a GoogleFit, aplicación para la que el *framework* que ha sido utilizado ya está preparado.

Como colofón y como ya ha sido matizado, se decidió que la nueva aplicación fuese híbrida, permitiendo así que el mantenimiento y la agregación de nuevas características se realizase solo una vez, aplicandose los cambios a ambas plataformas. De este modo, solo es necesario realizar dos desarrollos paralelos en caso de necesitar modificar algún fragmento relacionado con la interfaz de las aplicaciones de salud, ya que son diferentes para cada sistema operativo.

2.5. Definición de características

Habiendo determinado las soluciones, se lanzó la petición de realizar una nueva aplicación que recogiese las mejoras recientemente indicadas y que tuviese un perfil simple, minimalista y muy intuitivo. Todas características que permiten que el usuario se sienta cómodo utilizando la aplicación, y que a su vez resulte tan sencilla de utilizar que apenas suponga esfuerzo en el día a día.

Es por ello que la aplicación final tiene una estética sobria, sin sobrecarga ni de colores ni de elementos innecesarios. En dos toques en la pantalla del teléfono móvil, el usuario, no solo puede llegar al lugar al que quiere, sino que puede realizar la gran mayoría de las interacciones principales, como son la subida de información diaria o la consulta de alguna de las subidas ya realizadas.

2.6. Plan de trabajo

Se ha dividido el trabajo en 3 módulos:

- Core: el código web de la aplicaciÃ$^{3}nfueacometidoenprimerlugar$
- Conexión con glUCModel

Capítulo 3

Las bases de la renovada aplicación de glUCModel

3.1. Seleccionando las tecnologías a utilizar

Una vez determinadas las caraterísticas que se requerían en la nueva aplicación, era necesario elegir minuciosamente la tecnología que se iba a utilizar, ya que eran muchas las restricciones que existían:

- La aplicación tenía que poder desarrollarse en gran medida con tecnologías web, como son HTML5, CSS y JavaScript.
- Debía poder conectarse de manera directa con la interfaz HealthKit provista por Apple para la conexión con iOS Health.
- Era necesario que pudiese conectarse con el servidor ya existente de glUCModel.
- Por último, aun siendo una aplicación híbrida, debía conservar la gran mayoría las funcionalidades

propias de una aplicación móvil, como podrían ser las notificaciones nativas.

El *framework* elegido fue Ionic.

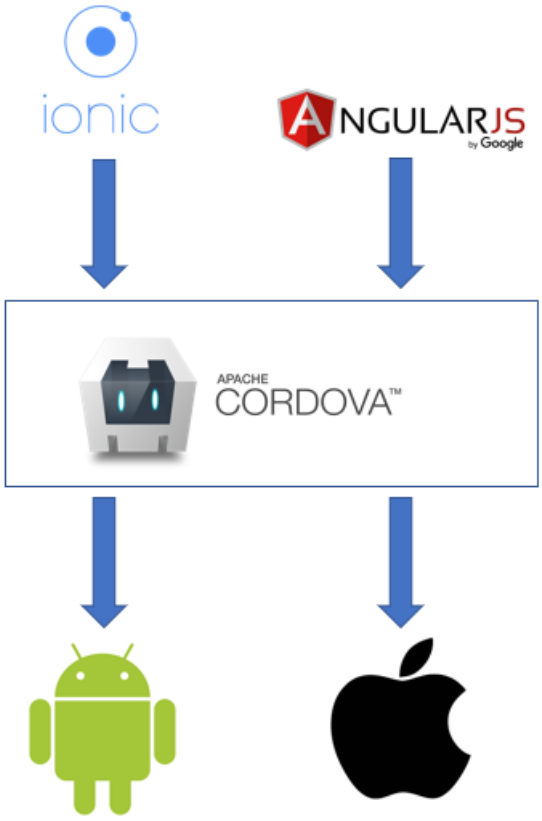

Figura 3.1: Cordova compila los ficheros HTML con los elementos visuales de Ionic y los ficheros Typescript de AngularJS para generar las aplicaciones para cada una de las plataformas.

3.2. Ionic

De entre todos los disponibles se eligió Ionic por varios motivos:

- Orientado a la multiplataforma, permite generar un

único código fuente para Android, iOS y aplicación web.

- Entre sus módulos nativos, se encuentra HealthKit, lo que simplifica enormemente la labor de extracción de datos de iOS Health. Gracias a este módulo podemos gestionar desde los permisos de lectura y escritura, hasta las peticiones de los registros de macronutrientes.
- Permite integrar AngularJS, facilitando sobremanera la gestión de nuevos módulos y la visualización de los datos extraídos de iOS Health.
- Viene con un módulo de peticiones HTTP, por lo que no existe problema alguno en la conexión con el servidor. En este punto es posible incluso elegir entre dos diferentes, ya que AngularJS también proporciona uno.
- Ionic corre sobre Cordova, por lo que proporciona a los desarrolladores una infinidad de módulos que permiten realizar llamadas al sistema operativo, manteniendo de esta manera las funcionalidades de las aplicaciones nativas.
- Otorga la posibilidad de comprobar en tiempo real el resultado de los cambios en la interfaz, a través de la herramienta *ionic-lab* (Figura 3.2).

- Proporciona una gran cantidad de elementos visuales predeterminados, por lo que la creación de la interfaz gráfica resulta sencilla y rica.

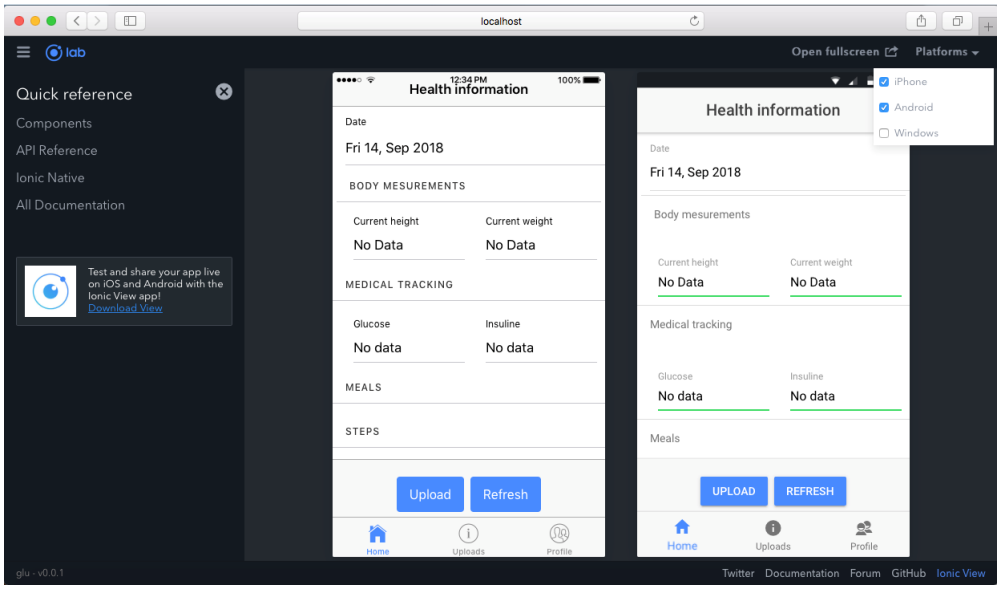

Figura 3.2: Desde ionic-lab podemos ver en tiempo real los cambios realizados en el código tanto para Android como para iOS, así como Windows Phone.

3.3. Los entornos de desarrollo: Visual Studio Code y XCode

Como entorno de desarrollo, para aprovechar las características de Ionic, se ha elegido Visual Studio Code. Entre los motivos que me impulsaron a seleccionarlo, la integración de la consola en el propio editor de texto fue lo más determinante.

Ionic gana potencial a travás de la consola, ya que permite, mediante comandos, realizar acciones tan importantes como crear una nueva página, que aunque

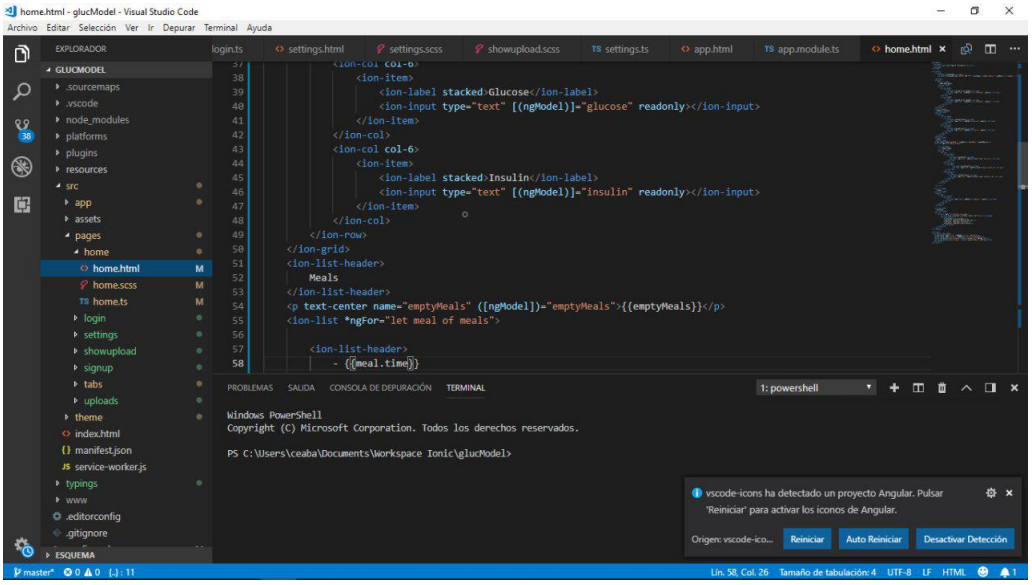

Figura 3.3: En la imagen se puede observar la integración de Visual Studio Code con Ionic, permitiendo lanzar los comandos sin tener que abandonar la herramienta. Además también observamos como detecta que el proyecto utiliza Angular y sugiere un reinicio para aplicar las ventajas del plugin AngularJS instalado.Respecto al fragmento de código que aparece en la imagen, vemos como podemos utilizar la directiva ngFor que facilita Angular para recorrer los elementos de un array de manera fácil y cómoda.

pueda parecer trivial, hacerlo manualmente supone un coste innecesario en tiempo, ya que son varios los ficheros que se deben actualizar, y a medida que la aplicación crece esta tarea se vuelve cada vez más compleja.

Otro gran ejemplo del poder de la CLI de Ionic, que Visual Studio Code explota a la perfección, es el *live-reload*. La ejecución del comando ionic serve lanzará la aplicación en modo local, lo que nos permite comprobar el estado de la aplicación a través del navegador, y gracias al *live-reload* podemos ver en tiempo real los cambios que vamos realizando.

Por último, los plugins que es posible añadir a este editor, permite que el desarrollo en AngularJS, así como la creación de la interfaz con los componentes de Ionic, se vuelva mucho más completo y potente gracias al coloreado, las sugerencias y la resolución de errores.

Por otro lado, y debido a que la aplicación que se ha desarrollado tiene como objetivo iOS, ha sido necesario el uso de XCode, ya que una vez construida la parte del código abstraÃda del sistema operativo, el *testing* de las funcionalidades nativas, como es la conexión con HealthKit, debía hacerse en un entorno con acceso a dicho módulo.

Una vez desarrollado el código, se ha generado el archivo instalable de la aplicación y se ha llevado a XCode para poder instalarla en los simuladores que vienen incorporados con esta herramienta de desarrollo. De esta manera, he podido comprobar la visualización de la interfaz en varios tamaños de pantalla, comprobando que los elementos visuales de Ionic se ajustaban perfectamente a cada uno de ellos, evitando así los problemas típicos de las distintas resoluciones (Figura 3.4).

3.4. Nuestro proveedor de información: iOS Health

Teniendo el código web ya finalizado, era el momento de mirar a la aplicación que nos iba a proporcionar la información de salud, y ver que datos podíamos obtener.

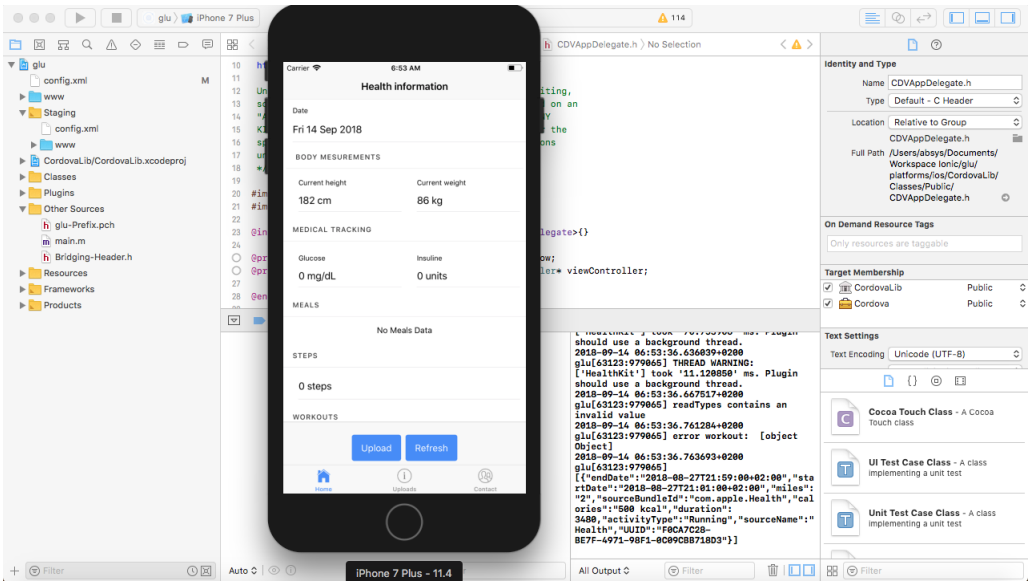

Figura 3.4: XCode nos proporciona una potente herramienta de pruebas para nuestro objetivo, ya que los simuladores integrados vienen con la aplicación Health preinstalada y nos permite hacer pruebas sobre la obtención de información a través de HealthKit.

Sin perder de vista en ningún momento el objetivo, ya se había estudiado que con esta aplicación podríamos obtener datos sobre la diabetes, como son la glucosa en sangre o la insulina liberada en el cuerpo.

Como mínimo debíamos tener acceso a la gran mayoría de factores que afectan al nivel de azúcar en sangre, que ya han sido mencionados al principio del documento:

- Información sobre el peso corporal. Se puede obtener en kilogramos o en libras
- Nivel de azúcar en sangre. Lo obtenemos en mg/dl
- Insulina liberada en el cuerpo. Este dato lo obtene-

nemos como unidades liberadas en el cuerpo.

- Actividad física, tanto suave como intensa. Entendiendo la actividad física suave como caminar, obtenemos cada registro de pasos introducido en la aplciación. No obstante, las personas que posean una pulsera de actividad, pueden registrar los pasos automáticamente de una manera mucho más completa. Para la actividad física intensa o moderada, nos proporciona información sobre los entrenamientos o *workouts*, con el tipo de ejercicio realizado (correr, nadar...), la hora de inicio, la duración y la cantidad de energía consumida
- Ingesta de comida. En este caso, HealthKit nos proporciona información sobre los registros de macronutrientes: grasas, carbohidratos y proteínas.

Además de esta información, nos facilita otra, que aunque menos relevante para la tarea de glUCModel, puede ser interesante en un futuro, como los micronutrientes ingeridos, la presión arterial o las pulsaciones.

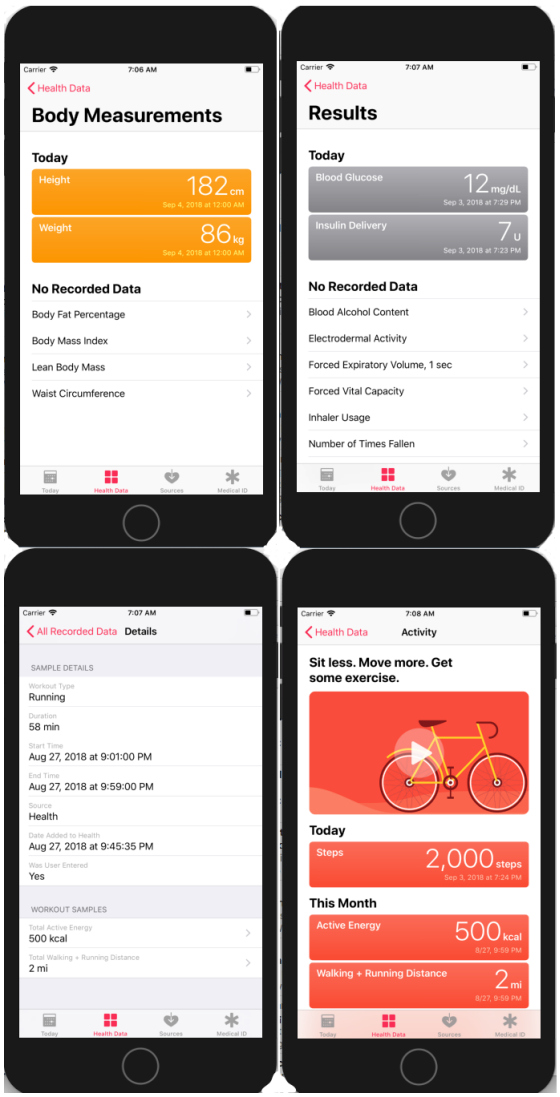

Figura 3.5: iOS Health nos permite introducir todos los datos relevantes para ayudar a glUCModel en su labor de predicción. Peso corporal, glucemia e insulina liberada en el cuerpo, y actividad física, tanto suave como moderada o intensa.

Capítulo 4

El núcleo de la aplicación

4.1. Estructura del código fuente

Una vez sentadas las bases de la información que vamos a recoger y sus características, abordé el desarrollo del código.

Como punto de inicio, la estrucutura viene dada en la creación del proyecto Ionic, utilizando el comando ionic start. A pesar de que el contenido del proyecto contiene un gran número de archivos, nosotros nos centraremos solo en la carpeta src, en cuyo interior se encuentra las carpetas con los elementos que es necesario modificar para desarrollar casi por completo la aplicación:

- app: dentro de este directorio, el archivo más importante es app.module.ts. Es el archivo Typescript que aglutina las características de la aplicación, es decir, contiene todas las referencias a los módulos que están disponibles para ser usados así como las páginas que se han creado y son accesibles

- assets: esta carpeta contiene las imágenes de la aplicación entre las que se encuentran por ejemplo el logo que utilizará nuestra aplicación, o el favicon
- pages: aquí se encuentra el código fuente de la aplicación como tal: los archivos HTML para las vistas, los archivos scss para modificar el estilo de cada página en concreto, y por último los archivos ts o Typescript. Se genera un directorio con cada uno de estos archivos cada vez que se crea un página dentro de la aplicación, entendiendo por página, una vista.
- theme: por último, theme contiene el archivo variables.scss, donde podremos crear variables de estilo comunes para todas las páginas. Este archivo es muy útil sobre todo para utilizar las variables con los colores de nuestra aplicación.

4.2. Comunicación con el servidor

Para la comunicación con el servidor ha sido necesario elaborar una interfaz. Esta interfaz debía ser de doble sentido ya que la aplicación necesita enviar datos al servidor para almacenarlos en la base de datos, y recibir datos de las subidas ya realizadas para que el usuario pueda consultarlas.El lenguaje utilizado ha sido el mismo que ya utilizaba con anterioridad la plataforma web de glUCModel: PHP.

Para que la aplicación funcionase correctamente en

su totalidad, era necesario establecer los siguientes *web services*:

- login: utilizado para la comprobación de credenciales en el login de la aplicación.
- register: destinado a comprobar si el correo electrónico coincide con alguno de los ya registrados y en caso de no ser así, insertar el usuario en la base de datos.
- upload: permite a la aplicación subir los datos de salud de un día concreto a la aplicación.
- uploads: devuelve una lista de todas las subidas de datos realizadas por el usuario para que puedan ser consultadas en la aplicación.

4.3. La interfaz: liviana y directa

Como ya se ha mencionado, desde el principio nos hemos centrado en hacer que esta nueva aplicación fuese lo mÃ¡s útil y fácil de usar que estuviese en nuestra mano. Por ello, solo aparece en las vistas la información esencial y los elementos estrictamente necesarios para liberar al usuario de tener que navegar por complejos menús.

Con este fin, se estableció un sistema de pestañas:

- Status: permite consultar toda la información de salud del día que aparece en el campo Date.

- Uploads: el usuario puede consultar las subidas ya realizadas.
- Settings: en esta pestaña, es posible seleccionar la prefrencias de uso.

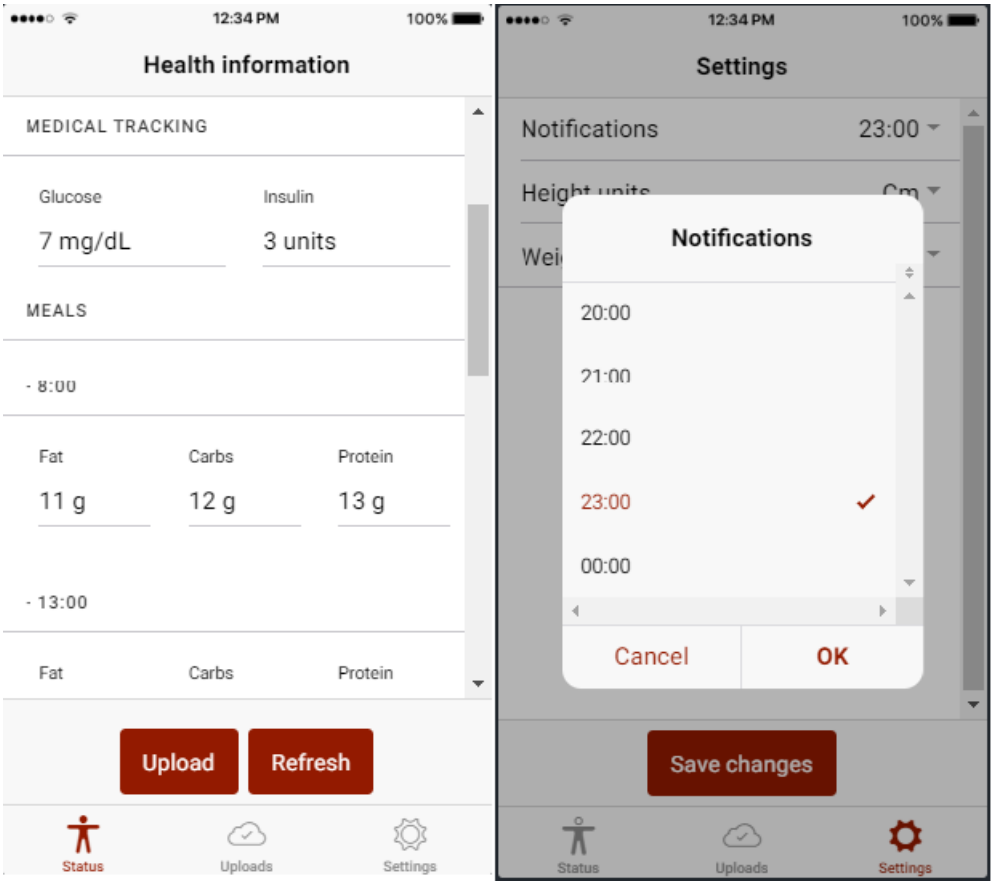

Figura 4.1: La interfaz de la aplicación es simple y fácil de usar

Capítulo 5

Desafíos en la implementación

5.1. Integración de Ionic con HealthKit

Uno de los problemas más grandes que me he encontrado a la hora de desarrollar la aplicación, ha sido la falta de documentación concreta sobre la integración con iOS Health. A pesar de que Ionic posee una página web con una amplia documentación, no detalla lo suficiente como ha de ser la comunicación con HealthKit.

Por mencionar algunas de las tareas que no han sido triviales:

- Unidades: en las peticiones a HealtKit, se puede detallar la unidad en la que se quiere recibir los datos. Sin embargo, este mecanismo solo funciona si reconoce cuál es esa unidad, y no solo eso, sino que además también debía ser una unidad adecuada para esa medición en concreto.

  En la página web de Apple se ofrece una documentación sobre las variables que se pueden consultar,

y especifica cuáles son las unidades, pero no la abreviatura, por lo que llegar a descubrir las unidades ha sido una tarea de ensayo y error.

- Fechas: para el campo Fecha de la pantalla principal, que permite selecciona el día para el que queremos visualizar y subir la información, he utilizado la etiqueta <ion-datetime>. Esta etiqueta, aunque muy útil, se comporta de una manera no deseada en el escenario en el que nos encontramos, ya que al seleccionar una fecha en este elemento, devuelve la fecha en *ISO 8601 datetime format*. HealthKit no reconoce este formato, por lo que ha sido necesaria la creación de una función adicional de conversión entre tipos.

- Meals: otro de los desafíos a los que me he enfrentado, no afectaba solo a la complejidad del desarrollo, si no también a la labor de glUCModel. Este desafío es la agrupación de macronutrientes en ingestas de comida. Cada ingesta, tiene un gran impacto en el nivel de azúcar en sangre. Es por ello que gana especial importancia las horas y la cantidad de carbohidratos ingeridos a una hora concreta, además de la cantidad total diaria. HealthKit tienes totalmente desligados los macronutrientes, por lo que no es posible relacionarlos en primera instancia.

  Por este motivo, se ha desarrollado una función que

a partir de los registros de carbohidratos, grasas y proteínas, genera registros de comidas en función de la hora de introducción de los datos. Siempre que los datos se introduzcan en un intervalo de 5 minutos, la aplicación los agrupará para ser visualizados y almacenados como comidas.

5.2. Conexión con la base de datos

Uno de los desafíos a los que la gran mayoría de desarrolladores se enfrenta cuando va a implementar una aplicación, es la conexión con la base de datos. Para esta tarea se ha utilizado la base de datos SQL ya existente en glUCModel. Solo ha sido necesaria la agregación de una tabla para el almacenamiento de las comidas generadas por la aplicación.

Esta conexión se ha realizado desde la API descrita en el punto 3.2, a través de la librería *mysqli*.

5.3. Almacenamiento local de datos

Otro de los quebraderos de cabeza que suele encontrarse en cada aplicación, es el almacenaje de datos locales, ya que toda la información que estamos manejando es información con un alto grado de sensibilidad, y almacenarla en local es arriesgado. No osbtante, ya que toda la información la obtenemos de glUCModel y se envía directamente al servidor, no ha sido necesario

tomar medidas en este punto.

5.4. Protección de credenciales

Con el objetivo de proteger de posibles intrusiones las credenciales de acceso, se ha establecido en la aplicación que el envío de la contraseña se realice cifrado con el algoritmo MD5. Otro punto a favor es que en este formato es en el que se almacenan las contraseñas en la base de datos de glUCModel, por lo que la comprobación de credenciales es directa.

Capítulo 6

Resultados

Si echamos la vista atrás y revisamos los objetivos que se marcaron en la concepción de la aplicación, podemos decir con firmeza que cumple con los requisitos establecidos:

- El uso del framework Ionic ha permitido que la aplicación sea híbrida, haciendo que el coste de mantenimiento se reduzca notablemente, ya que alrededor del 65 % se ha desarollado en tecnologías web, y teniendo en cuenta que la funcionalidad más importante tiene como objeto las llamadas al sistema operativo, es un porcentaje aceptable.
- Se cumple con el objetivo inicial, ya que no solo extraemos la información de iOS Health y la enviamos a la nube, sino que además la enriquecemos con el agrupamiento de macronuetrientes en ingestas de comida.
- Hemos evitado que los usuarios tengan que intro-

ducir los datos manualmente en nuestra aplicación.

- Potencialmente tenemos acceso a datos más antiguos, en usuarios que hayan utilizado iOS Health desde antes de instalar la aplicación de glUCModel.

Capítulo 7

Conclusiones

En general, la implementación de la aplicación ha salido adelante y se han cumplido los requisitos que en un inicio se plantearon. No obstante, existe un tremendo margen de mejora, que no ha sido posible abordarlo por falta de tiempo:

- Carga masiva de información: actualmente solo es posible cargar la información por días. Una mejora interesante sería permitir que los usuarios cargasen información de un intervalo de tiempo concreto.
- Solo hemos arañado la superficie en lo que a la información que podemos extraer de iOS Health se refiere. Hay muchos datos que se pueden aprovechar y que actualmente no están siendo utilizados.
- Ahora es posible enviar datos desde el teléfono a la plataforma, pero no al revés. Si bien es cierto que existe una función que permite modificar los datos de iOS Health mediante introducción manual, aun-

que ahora mismo no sea visible para el usuario. Esto se puede utilizar para cargar los datos que ya existen en glUCModel en iOS Health. Esto haría que el usuario pudiese consultar, no solo los datos que ha subido desde la propia aplicación, sino también los existentes en la nube.

En lo que respecta a mi experiencia personal desarrollando esta aplicación, ha sido un arduo camino, ya que los únicos conocimientos que he reutilizado son los de HTML, CSS y algo de JavaScript. AngularJS, Ionic y las aplicaciones móviles eran campos totalmente desconocidos para mí, por lo que no ha sido tarea fácil.

Bibliografía

1- Apple HealthKit documentation: https://developer.apple.com/ documentation/healthkit

2- Angular documentation: https://angular.io/docs

3- PHP manual: http://php.net/manual/es/index.php

4- Ionic documentation: https://ionicframework.com/docs/

5- Ionic blog: https://blog.ionicframework.com/

6- Web technologies for developers: https://developer.mozilla.org/es/docs/Web/JavaScript

7- Visual Studio Code documentation: https://code.visualstudio.com/docs