glUCModel para iOS

POR: FRANCISCO JAVIER LINDE BLÁZQUEZ

GRADO EN INGENIERÍA DEL SOFTWARE

FACULTAD DE INFORMÁTICA

DEPARTAMENTO DE ARQUITECTURA DE COMPUTADORES Y AUTOMÁTICA

UNIVERSIDAD COMPLUTENSE DE MADRID

PROYECTO DE FIN DE GRADO EN INGENIERÍA DEL SOFTWARE

Madrid, a 18 de Junio de 2014

Director: José Ignacio Hidalgo

Colaborador: J. Manuel Colmenar

AUTORIZACIÓN DE DIFUSIÓN Y UTILIZACIÓN

Francisco Javier Linde Blázquez autoriza a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor, tanto la propia memoria como el código, los contenidos audiovisuales, incluidas imágenes del autor y documentación.

A Gerardo Fernández Moreno.

Por ser un ejemplo de trabajo y constancia.

Por servirme de motivación.

AGRADECIMIENTOS

Mis más sinceros agradecimientos a mi director de proyecto **José Ignacio Hidalgo** por todo el tiempo dedicado, por el interés mostrado y por su paciencia y amabilidad durante el desarrollo del proyecto.

Agradecimientos a **José Manuel Colmenar** por su colaboración en el proyecto, ha arrojado luz en aquellos temas técnicos que requerían una consulta externa.

Agradecimientos a mi compañero **Abdelkhalek El Mansouri** por colaborar en aquellas tareas comunes del proyecto así como por prestar la ayuda que le ha sido posible en el proyecto.

ÍNDICE

Versión en Inglés: Página 65

English Version: Page 65

1. INTRODUCCIÓN

La tecnología debe jugar un papel importante en nuestra sociedad, y es que no debemos contemplarla sólo como una herramienta para ayudarnos a hacer lo que ya sabemos, sino que tenemos que ir mas allá y concebirla como una herramienta para hacer cosas que hasta ahora no podíamos hacer, cosas que ni siquiera imaginábamos.

Sin embargo, en ocasiones el tremendo potencial que tienen las nuevas tecnologías no está siendo aprovechado plenamente, y esto se debe a que muchas veces se han desarrollado las herramientas, pero aún no se han identificado los problemas que queremos resolver.

Son muchas las personas que en su día a día tienen que luchar con problemas, problemas personales, problemas laborales, económicos, familiares, de salud… Es en estos últimos en los que nos vamos a centrar.

Y es que según un estudio realizado por di@bet.es en 2011 el 13,8% de los españoles mayores de 18 años sufre diabetes, y a nivel mundial hay mas de 260 millones de personas diabéticas. Estos datos arrojan suficiente luz acerca de la magnitud del problema.

1.1 - HABLEMOS DE DIABETES

La Diabetes Mellitus (comúnmente conocida como diabetes) es una enfermedad que afecta al metabolismo. No se trata de una enfermedad temporal, dura toda la vida. Se caracteriza por que el paciente muestra unos niveles elevados de glucosa en sangre (hiperglucemia).

La *American Diabetes Association* clasifica la diabetes en 2 tipos:

- Tipo 1: El páncreas del paciente no genera insulina, de tal modo que los niveles de glucosa no pueden ser reducidos adecuadamente. Estos pacientes son *insulinodependientes,* es decir que requieren de un aporte externo de insulina en su día a día. La forma de incorporar este aporte insulínico puede ser mediante una bomba o mediante inyecciones.
- Tipo 2: El paciente genera insulina en dosis inferiores a las correctas y a esto se le debe añadir que el cuerpo muestra tolerancia a la glucosa, lo cual la convierte en menos efectiva reduciendo los niveles de glucosa en sangre. Estos pacientes no son *insulinodependientes*.

Afortunadamente más del 90% de los diabéticos se encuentra en el segundo grupo, y decimos afortunadamente porque este tipo de diabetes es menos incapacitante que la diabetes tipo 1. El principal tratamiento para la diabetes tipo 2 consiste en lograr un peso saludable y mantener una alimentación sana, pero en algunos casos requieren tratamiento en forma de pastillas o en etapas avanzadas necesitan también aporte externo de insulina.

Los diabéticos tipo 1 son por tanto los diabéticos que ven su vida afectada en mayor medida. Deben mantener un control meticuloso de sus niveles de glucemia en sangre para prevenir tanto hiperglucemias (niveles > 120mg/dl) como hipoglucemias (niveles < 40mg/dl). El control meticuloso que mencionamos consiste en mediciones de glucemia en sangre, para realizar posteriores predicciones basadas en la ingesta de comida que van a realizar, el posible deporte que vayan a practicar, el peso que tengan en ese momento, y muchos otros factores que influyan en los niveles de glucemia. En base a esta predicción deben realizar una estimación de la insulina que van a inyectarse para mantener sus niveles de glucemia en unos valores adecuados.

1.2 - GLUCMODEL

La tarea de un diabético tipo 1 no es fácil. En la mayoría de los casos los pacientes perfeccionan sus predicciones con la práctica, pero aún con muchos años de experiencia el paciente yerra en sus

estimaciones. Es en esta tarea diaria en la que glUCModel quiere jugar un papel relevante. La idea detrás de glUCModel es la de ser capaces de predecir los niveles en sangre del paciente basándose en parámetros actuales del sujeto como son el nivel de azúcar o próxima ingesta de carbohidratos para aconsejarle una cantidad de insulina determinada.

Para realizar los cálculos glUCModel está desarrollando gramáticas evolutivas basadas en programación genética. Este tipo de definición permite una mayor libertad a la hora de definir la expresión matemática de la solución, puesto que no se tienen restricciones como por ejemplo se pueden encontrar utilizando ecuaciones lineales.

El proyecto lleva varios años desarrollándose mediante la colaboración de la Universidad Complutense de Madrid, el Hospital Universitario Príncipe de Asturias (Alcalá de Henares, España) y el Hospital Virgen de la Salud (Toledo, España). Los profesionales implicados en el proyecto son:

- *J. Ignacio Hidalgo, José L. Risco-Martin, Juan Lanchares y Oscar Garnica del departamento de Arquitectura y Autómatas de la Universidad Complutense de Madrid.*
- *J. Manuel Colmenar y Alfredo Cuesta-Infante del C.E.S. Felipe II, Universidad Complutense de Madrid.*
- *Esther Maqueda del departamento de nutrición y endocrinología del Hospital Virgen de la Salud (Toledo, España)*
- *Marta Botella y José Antonio Rubio del departamento de nutrición y endocrinología del hospital Universitario Príncipe de Asturias (Alcalá de Henares, España).*

1.3 - EL PROYECTO: LA APLICACIÓN MÓVIL

Una de las principales necesidades de glUCModel es la recogida de datos, puesto que sin ella no se podría por un lado, encontrar la fórmula de estimación de glucosa, y por otro lado tampoco se podría

aplicar esta fórmula para informar al paciente de la insulina que debe administrarse.

Para cubrir esta necesidad de datos se realizó una aplicación web desde la cual introducir datos y acceder a ellos para visualizarlos, pero el usuario necesita introducir datos en todo momento y no cabe duda de que no contamos con un ordenador las 24 horas del día. Cabe la posibilidad de apuntar los datos en un bloc para posteriormente introducirlos pero tampoco es una solución cómoda, por lo que se optó por la mejor solución para el paciente: Realizar una aplicación móvil.

Nadie se separa de su teléfono móvil, los datos son reveladores: En España existen 26 millones de smartphones con acceso a internet y el 53,8% de los españoles navega con su terminal móvil de forma diaria. Estos datos fueron recogidos y publicados por la *Fundación Telefónica* a principios de 2014.

En 2013 fueron ofertados los dos proyectos de fin de grado para el desarrollo de glUCModel en móviles: Una versión para Android y una para iPhone con el fin de proporcionar a los usuarios de Glucmodel una forma sencilla, accesible y amigable para la incorporación de la información a la base de datos. Escogí iPhone porque creo que es una plataforma con mucho potencial. Muchas son las empresas que necesitan desarrolladores para iOS y por otro lado la App Store es sin ninguna duda ni rival la plataforma más rentable de venta de aplicaciones, por tanto poseer conocimientos de programación en iOS me abre como desarrollador autónomo la posibilidad de crear y rentabilizar mis propias aplicaciones móviles.

La aplicación de iPhone desarrollada permite al usuario introducir datos sobre su glucemia en sangre, sus inyecciones de insulina, sus comidas, su peso y su actividad física. Los datos recogidos son directamente enviados a la base de datos de glUCModel la cual los almacenará de forma segura. Todos los datos son también accesibles desde la aplicación móvil para que el usuario pueda visualizarlos.

Este método para añadir datos a la base de datos supone una mejora importante en cuanto a usabilidad se refiere. El usuario puede ahora usar una interfaz adecuada al tamaño de pantalla de su Smartphone, además se añade una ventaja adicional respecto al uso de

la interfaz web: posibilidad de uso offline. En la web lógicamente si perdemos el acceso a internet no podremos almacenar ningún dato, sin embargo en la aplicación para iPhone, podremos añadir datos que se almacenarán localmente hasta que podamos realizar la subida.

El iPhone no almacena ninguno de los datos introducidos por el paciente salvo que no sea posible almacenarlos en el servidor, en tal caso permanecerán en el iPhone hasta que puedan ser subidos. La aplicación no almacena ningún tipo de dato personal del paciente, de tal modo que aunque los datos almacenados temporalmente en el iPhone se vieran comprometidos, no sería posible asociarlos a una persona física, puesto que las transacciones son realizadas utilizando el ID del usuario asignado por glUCModel.

1.4 - MOTIVACIÓN PERSONAL

A la edad de 18 años y tras un año sufriendo las consecuencias de tener el azúcar por las nubes, diagnosticaron diabetes a mi mejor amigo. Pude vivir el proceso en persona puesto que durante todo el tiempo que pasó hasta que fue diagnosticado como diabético vi como se encontraba siempre cansado, como decía que estaba perdiendo vista, le veía beberse botellas de dos litros de agua de una vez… poco a poco se encontraba cada vez peor hasta que tuvieron que ingresarle por que no podía apenas mantenerse en pie. En unos análisis de sangre se pudo ver con un azúcar por encima de 400mg/dl que padecía diabetes.

Tras ser diagnosticado diabético lo primero que debes hacer es lógicamente mantener un control de tu azúcar, en primer lugar para aprender a controlarlo, y en segundo lugar para que tu endocrino pueda llevar un buen control de tu enfermedad.

Durante unos meses mi amigo tenía que apuntar en una libreta todas las mediciones que se hacía, las raciones de hidratos de carbono que había ingerido y la insulina que se había administrado. Esto sucedió en 2009, en ese momento comenzaba el boom de las apps móviles en España, a mí me acababan de regalar un iPhone 3G y miraba el mundo

con otros ojos, con ojos de programador de apps, allá donde miraba veía una app y tenía en mente hacer una app para ayudar a los diabéticos como mi amigo a llevar el control de sus datos. En la lista de proyectos de Trabajo de Fin de Grado se encontraba realizar la app para iPhone de glUCModel, cuando lo vi supe que quería que hacerla, porque ya hacía años que lo tenía en mente.

1.5 - PROBLEMAS A RESOLVER

Como se ha comentado ya glUCModel existía y existe como aplicación web, está desarrollada en Java y su base de datos es SQL. Los primeros problemas surgen rápido: iOS no tiene un buen soporte para SQL. A contrario de otras plataformas y lenguajes de programación iOS no tiene una buena integración con SQL, y la gran mayoría de los desarrolladores optan por crearse sus propios Web Services para acceder a sus bases de datos SQL.

Teníamos que decidir cual sería el comportamiento de la app en caso de no disponer de conexión a internet.

Otra de las preocupaciones era por supuesto la integridad de los datos de los pacientes. Debíamos mantener la seguridad de la base de datos así como de los datos que se almacenaran en el dispositivo.

Otra de las tareas era diseñar una interfaz sencilla y rápida para que cualquier persona con conocimientos de uso básico de un smartphone fuera capaz de usarla.

2. ¿QUÉ ES GLUCMODEL?

glUCModel es una aplicación web disponible *online* que tiene un doble propósito:

- Mejorar el control de la diabetes
- Ayudar a los diabéticos y a sus médicos a controlar la enfermedad.

Los requisitos del proyecto fueron realizar la aplicación multiplataforma, de tal forma que el dispositivo que tuviera el usuario no fuera una restricción. La aplicación web fue probada en múltiples dispositivos como *smartphones, tablets*, portátiles y ordenadores de sobremesa. También se utilizaron en las pruebas diversos sistemas operativos: Mac OS, Windows, Linux, iOS y Android; y diversos navegadores web: Internet Explorer, Mozilla Firefox, Google Chrome, Safari y Opera.

Tras las pruebas se determinó que la aplicación funcionaba correctamente en todas las plataformas, pero la experiencia del usuario se veía afectada al usar dispositivos con pantallas pequeñas, por lo que se decidió realizar aplicaciones nativas para dispositivos móviles iOS y Android.

glUCModel está compuesto por los siguientes cinco módulos conectados entre sí:

1. Interfaz Gráfica: Es el núcleo de la aplicación. Conecta todos los módulos. Se usa para permitir a los pacientes realizar consultas de sus datos, modificaciones y también subidas de datos nuevos. Además los médicos también pueden mantener un control de los datos que los pacientes van añadiendo a la aplicación.
2. Base de datos: La base de datos de glUCModel almacena la información sobre usuarios (pacientes y médicos), los análisis del paciente, las mediciones de glucemia, deporte, alimentación, ejercicio físico, peso… etc.

3. Módulo de aprendizaje o e-learning: Es un espacio de aprendizaje virtual donde el paciente encontrará toda la información necesaria sobre la diabetes. Podrá encontrar documentos con conceptos teóricos acerca de la diabetes y herramientas para ayudarle en su educación com son tests, calendarios, foros y glosarios de términos.

4. Modelo de Glucosa: Realizado con técnicas de programación evolutiva. Este módulo obtiene un modelo personalizado para cada paciente usando la información proporcionada por la base de datos.

5. Sistema Recomendador: La función del sistema recomendador es la de evaluar los datos del paciente. El sistema crea recomendaciones sobre cómo el paciente puede mejorar su calidad de vida. El paciente recibe emails notificándole estas recomendaciones.

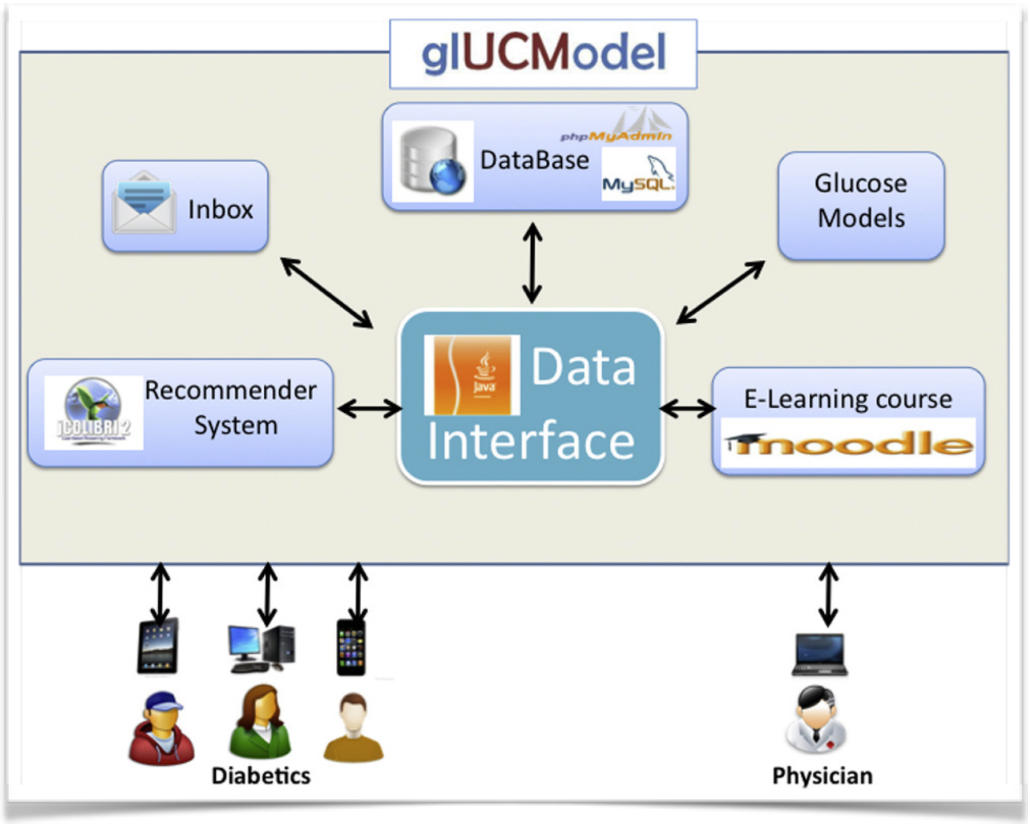


FIGURA 1: Estructura general de gIUCModel (Aplicación Web).

El funcionamiento de la aplicación es el siguiente: los pacientes suben los datos a través de la interfaz gráfica. Los médicos pueden revisar los datos de sus pacientes mediante la misma interfaz gráfica. El modelo de glucosa analiza la información y genera las recomendaciones para el paciente. El módulo de aprendizaje está conectado con la aplicación para comunicar el progreso de los pacientes con los tests.

3. DESARROLLANDO EN iOS

3.1 - ¿POR QUÉ iOS?

La Apple App Store es sin duda alguna la tienda de aplicaciones más rentable del mercado a día de hoy. El día 2 de Julio de 2014 Apple hizo público su número de descargas acumuladas en el App Store desde su salida en 2007 durante la WWDC (World Wide Developer Conference). En estos 7 años de vida del App Store el número acumulado de aplicaciones descargadas asciende a 75.000.000.000 (setenta y cinco mil millones).

Durante estos 7 años Apple ha repartido a los desarrolladores 15.000.000.000$ (quince mil millones) de dólares. Para hacer una comparativa hemos escogido a sus dos principales competidores: Android y Windows Phone. Los datos oficiales que hemos recogido son los siguientes:

	Apple	Google	Microsoft
Número de usuarios (en millones)	600	900	12
Número de Apps (en miles)	1250	800	160
Número de desarroladores (en miles)	235	150	45
N° de descargas (en miles de millones)	50	48	65
Pago a desarrolladores (en millones de $)	5000	900	100

*Todos los datos corresponden al año 2013

FIGURA 2: Tabla comparativa. Datos tiendas de aplicaciones.

Todos estos datos pueden encontrarse en la página web de Forbes, bajo el titular *"How Much Do Average Apps Make?"* http://www.forbes.com/sites/tristanlouis/2013/08/10/how-much-do-average-apps-make/

Basta con fijarnos en que se han descargado aproximadamente el mismo número de aplicaciones en iOS, en Android y en Windows Phone, sin embargo los ingresos de los desarrolladores se multiplican por más de 5 en Apple respecto a Android, y por más de 50 respecto a Microsoft.

Con los datos obtenidos anteriormente decidimos hacer una comparativa de rentabilidad para estudiar qué tienda era más interesante para el desarrollador.

	Apple	Google	Microsoft
Media de aplicaciones por desarrollador	5	5	3
Media de descargas por app	40.000	60.000	4.062
Beneficio medio por descarga	0,10$	0,01875$	0,1538$
Beneficio medio por desarrollador	20.000$	5.625$	1.874$

*Todos los datos corresponden al año 2013

FIGURA 3: Tabla comparativa. Rentabilidad de tiendas de aplicaciones.

Si prestamos atención a la última fila podremos observar que a afectos monetarios, un desarrollador de iOS obtiene 4 veces más ingresos que uno de Android y 11 veces más que uno de Windows Phone. Los datos son claros: La App Store es a día de hoy la tienda de Apps más rentable del mercado.

Por tanto como ingeniero me aporta mucho el hecho de poder desarrollar aplicaciones para iOS, ya que podría llegar a monetizar alguna en el App Store.

3.1 - LENGUAJE: OBJETIVE-C

Actualmente sólo se pueden desarrollar aplicaciones nativas para iOS en el lenguaje de programación Objetive-C. Para la próxima versión de iOS (iOS 8), que se espera esté disponible en Otoño de 2014, se podrá también programar en Swift. Swift fue presentado por Apple el 2 de Junio de 2014 por lo que aún no se sabe demasiado sobre este nuevo lenguaje.

Objetive-C es un lenguaje de programación que apareció en 1980, y como su propio nombre indica está orientado a objetos. Fue creado por Brad Cox y en 1992 fue liberado bajo la licencia GPL.

Objetive-C supone una leve capa por encima de C, por lo que es posible compilar cualquier aplicación de C con un compilador de Objetive-C, así como también es posible incluir código C directamente en nuestra aplicación en Objetive-C.

Semánticamente es diferente a Java o C++ ya que el programador no hace llamadas a métodos de objetos, lo que hace es mandar mensajes a los objetos para que realicen las acciones que necesitemos.

Ejemplo de llamada a un método en C++ y Java

```
objeto->metodo (parametro);          objeto.metodo (parametro);
```

Ejemplo de envío de mensaje en Objetive-C:

```
[objeto metodo : parametro];
```

FIGURA 4: Ejemplo de llamadas en Java, C++ y Objetive-C

Esta peculiaridad lo convierte en un leguaje que en un primer contacto resulta extraño. En ocasiones provoca que haya llamadas a

funciones que ocupen varias líneas de código porque no entran en la pantalla debido a lo largas que se hacen. Este es un ejemplo de una función extremadamente sencilla que se puede encontrar dentro de esta aplicación. Su finalidad es mostrar una sencilla ventana de alerta.

```
- (void) alertStatus:(NSString *)msg :(NSString *)title
{
    UIAlertView *alertView = [[UIAlertView alloc] initWithTitle:title
                                                        message:msg
                                                       delegate:self
                                              cancelButtonTitle:@"Ok"
                                              otherButtonTitles:nil, nil];

    [alertView show];
}
```

FIGURA 5: Ejemplo de llamada a función en varias líneas.

La función alertStatus cuenta tan sólo con dos líneas de código, la primera construye un UIAlertView y la segunda le manda a este nuevo objeto la orden de mostrarse. Como se puede observar la llamada al constructor de UIAlertView se tiene que hacer en varias líneas por motivos de espacio y claridad.

3.2 - EL ENTORNO DE DESARROLLO: XCODE

Apple es conocida por ser una empresa cerrada en muchos de sus aspectos. En programación no han realizado excepciones. Estas son algunas de las condiciones que se deben cumplir:

1. La única manera de programar para iOS o Mac OS en Objetive-C es usar un Mac. Existen alternativas no oficiales pero no resultan viables.

2. El único entorno de desarrollo capaz de desarrollar, compilar y ejecutar aplicaciones de iOS en Xcode (por supuesto desarrollado por Apple).

3. Si deseamos probar una aplicación sobre un dispositivo como un iPhone o un iPad, debemos contratar una cuenta de desarrollador de iOS que tiene un coste de 99$/año.

4. Si deseamos publicar nuestra aplicación debemos además de contar con la cuenta de desarrollador citada en el punto anterior, pasar un estricto control de nuestra app, que garantice que esta cumple con las restricciones legales de la App Store y presenta un correcto funcionamiento.

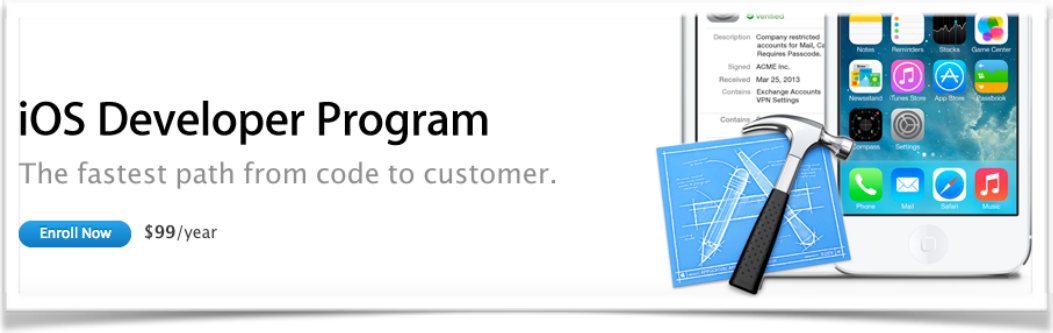


FIGURA 6: Programa de desarrollo para iOS y coste del mismo.

Una vez tengamos lo necesario podemos comenzar a programar con Xcode. Cabe decir que Xcode es gratuita lo cual al contrario de las restricciones anteriores, esto supone una gran ventaja para el programador. Xcode fue lanzando en 2003 y está integrado con *Interface Builder* una herramienta gráfica para la creación de interfaces.

Esto permite desarrollar ciertas partes de la aplicación de una manera sencilla y rápida. Xcode es capaz de compilar aplicaciones en C, C++, Objetive-C, Objective-C++, Java y AppleScript. Compañías externas han desarrollado soportes para Pascal, Ada y Perl.

Una característica muy interesante de Xcode es que es capaz de distribuir trabajos pesados como el de construcción, entre varios ordenadores mediante la tecnología Bonjour.

Xcode proporciona también un gestor de dispositivos conocido como *Organizer*. Mediante este gestor podemos conectar nuestro dispositivo iOS a nuestro Mac para probar las aplicaciones sobre el terminal.

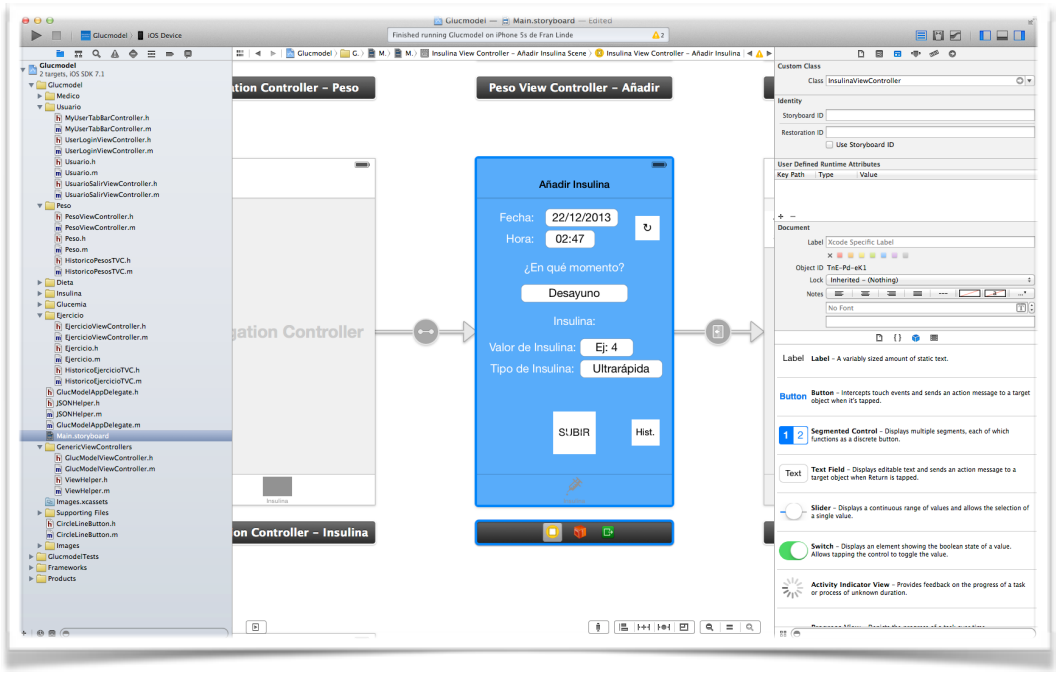


FIGURA 7: Interfaz realizada en Xcode de forma gráfica. A la derecha de la imagen se observan los elementos disponibles para su creación.

Las aplicaciones en iOS cuentan con un nivel de seguridad y control muy alto, por lo que todas las aplicaciones que generemos deben ir firmadas por el desarrollador. Para ello Xcode utiliza un complejo sistema de firmas y certificados digitales. En este apartado Apple ha tratado de simplificar el proceso y ayudarnos mediante Xcode pero lo cierto es que es una tarea ardua.

FIGURA 8: Organizer. Herramienta de gestión de dispositivos iOS para test.

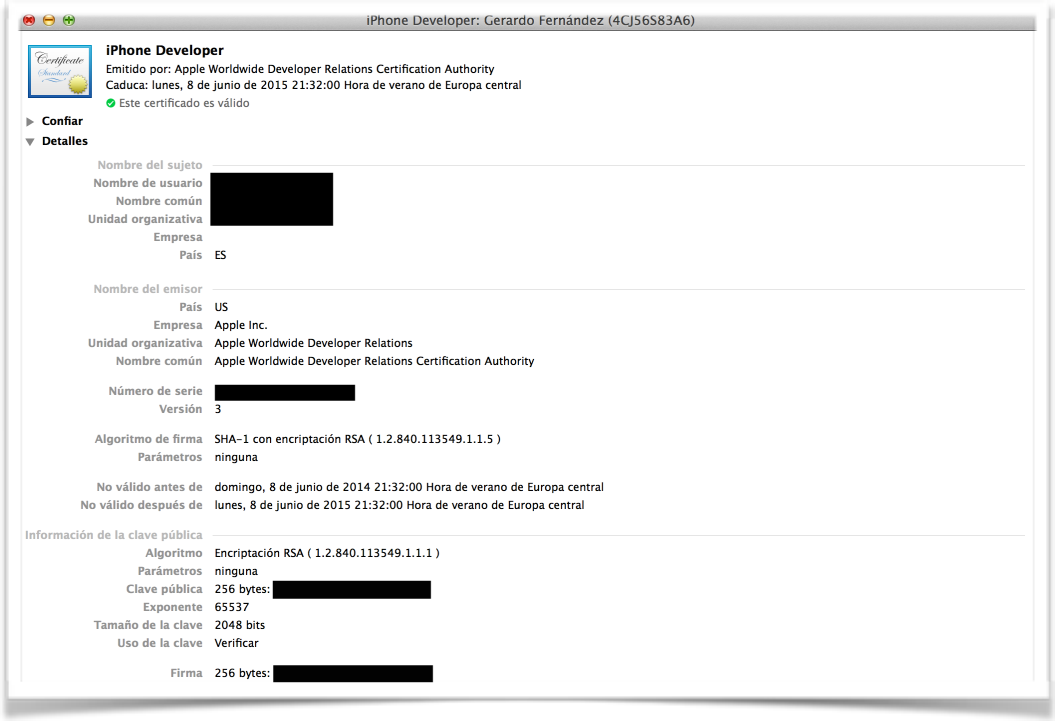


FIGURA 9: Ejemplo de certificado utilizado para firmar la aplicación glUCModel. Obsérvese que se indica el tipo de encriptación así como las claves pública y privada del algoritmo de cifrado.

Xcode cuenta con un simulador en pantalla en el que podemos probar nuestras aplicaciones si no tenemos un terminal físico o la cuenta de desarrollador necesaria. Este simulador es muy rápido, apenas necesita 3 segundos para encenderse y aproximadamente unos dos segundos en cargar la aplicación. Además de permitirnos probar la app si no tenemos las herramientas necesarias, también ofrece una gran ventaja y es que es posible simular todos los dispositivos iOS tanto iPhone, como iPad, como iPod touch… en sus diferentes generaciones y con la versión de iOS que escojamos.

He de añadir que he tenido la oportunidad de utilizar el simulador Android que ofrece Google para desarrollo, y el tiempo de encendido del simulador de Nexus 4 alcanzó los 10 minutos, se trata de una diferencia abismal con el simulador de iOS y supone un punto a favor del desarrollo en iOS.

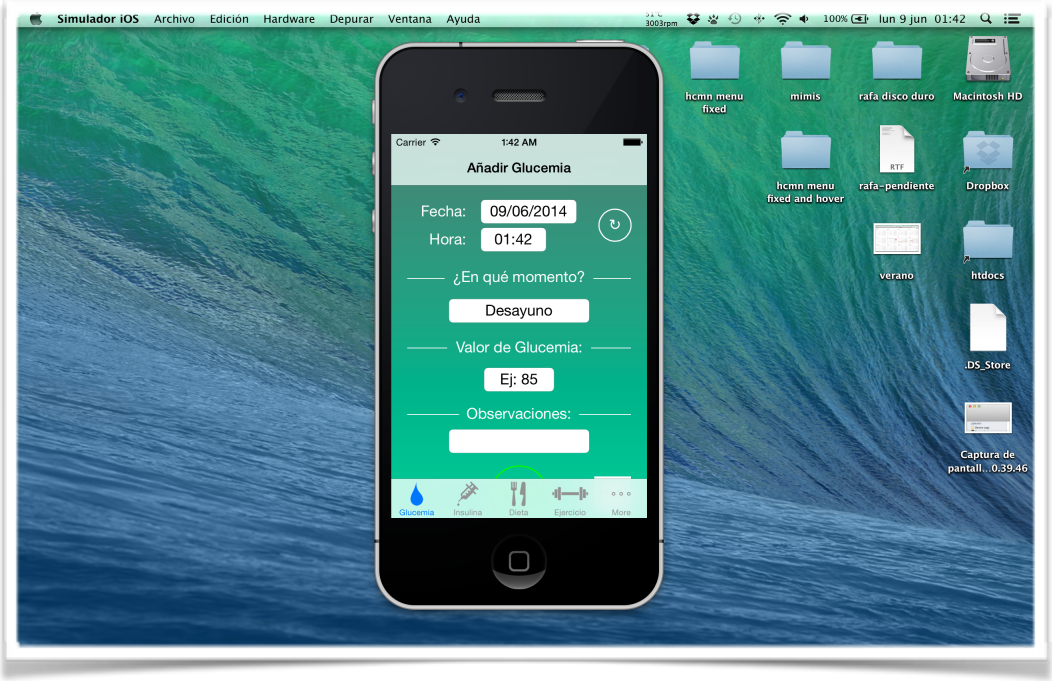


FIGURA 10: Ejemplo de uso del simulador de iPhone 4s con iOS 7.

La aplicación está disponible en la página que la Universidad Complutense de Madrid tiene destinada a las aplicaciones móviles

(http://www.ucm.es/apps). Posteriormente estará disponible en el App Store de Apple.

4. IMPLEMENTACIÓN

4.1 - METODOLOGÍA DE DESARROLLO

Se ha seguido un desarrollo tipo SCRUM, puesto que se han realizado entregas parciales con reuniones de verificación con el director del proyecto. La gran mayoría de los requisitos se definieron al principio del desarrollo en reuniones, posteriormente durante el desarrollo y las verificaciones se han ido adaptando y añadiendo requisitos para lograr el objetivo deseado. A lo largo del las iteraciones realizadas el proyecto ha avanzado de forma correcta y mejorado con cada una de ellas.

4.2 - ORGANIZACIÓN DEL CÓDIGO

El código ha sido distribuido en paquetes según su funcionalidad de tal modo que encontramos los siguientes grupos o paquetes:

1. Médico: Contiene las clases para las interfaces de *login* en médicos, la gestión de los pacientes asociados al médico, la clase médico, la interfaz de *logout* de médico, los controladores de dichas interfaces… etc

2. Usuario: Contiene las clases para las interfaces de login de usuario, la clase usuario, la interfaz de logout de usuario, los controladores de dichas interfaces, la clase que gestiona la interfaces de subida de datos…etc

3. Glucemia: Contiene las interfaces y los controladores necesarios para la subida de datos de Glucemia, y mostrado de datos mediante historial. Contiene las clases que representan una glucemia y almacenan los datos de la misma (nivel de azúcar en sangre, fecha y hora.. etc).

4. Peso: Contiene las interfaces y los controladores necesarios para la subida de datos de peso del paciente y mostrado de datos

mediante historial. Contiene las clases que representan un peso en una fecha determinada y almacenan los datos del mismo (fecha, observaciones y peso en kg).

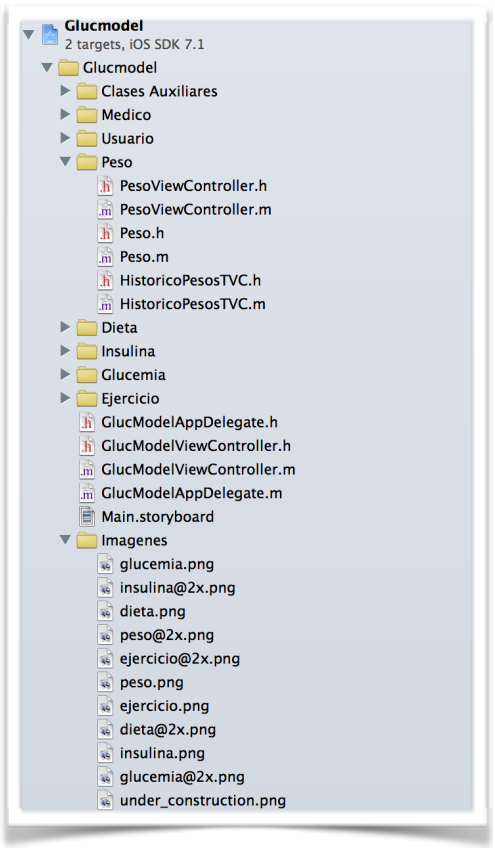
5. Insulina: Contiene las interfaces y los controladores necesarios para la subida de datos de administraciones de insulina, y mostrado de datos mediante historial. Contiene las clases que representan un evento de administración de insulina y almacenan los datos de este (Cantidad de insulina, tipo de insulina, fecha y hora.. etc).

6. Dieta: Contiene las interfaces y los controladores necesarios para el almacenamiento de datos de dietas. Contiene las clases que representan las ingestas anteriores mediante historial. Contiene la clase Dieta que almacena los datos de una ingesta (raciones de hidratos de carbono, fecha y hora, tipo de comida, observaciones.. etc).

7. Ejercicio: Este paquete almacena las clases encargadas de mostrar las interfaces para la subida de datos de ejercicios y el historial de los ejercicios realizados. Contiene la clase que representa y almacena un ejercicio y los datos asociados a este (nivel de exigencia física, fecha, hora y observaciones).

8. Imágenes: En este paquete se encuentran las imágenes utilizadas en las interfaces, en su gran mayoría iconos.

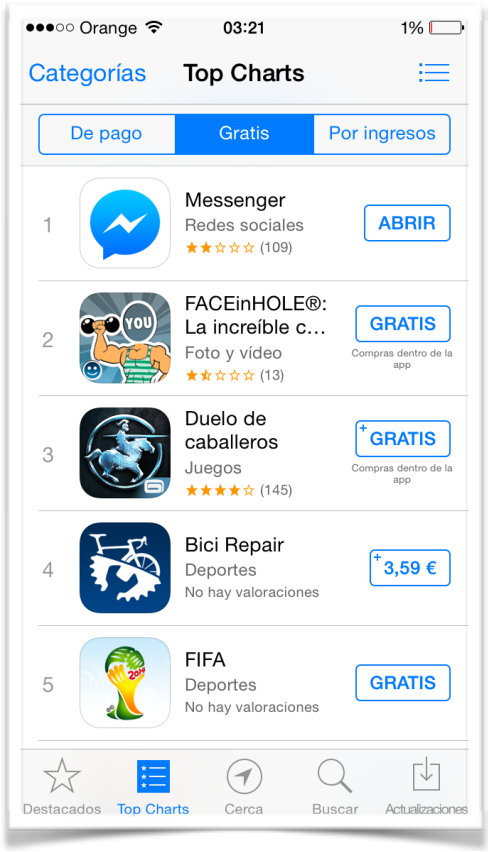
9. Clases Auxiliares: Almacena clases que son utilizadas por diversas interfaces, como es ViewHelper, que es utilizado para dibujar un fondo con colores degradados; o CircleLineButton que también es utilizado por varias interfaces para dibujar un botón circular; o JSONHelper que es utilizado para acceder a la base de datos SQL parseando los datos en formato JSON.



FIGURAS 11 y 12: Ejemplos de paquetes y clases del proyecto vistos con Xcode.

4.3 - DISEÑO DE LA INTERFAZ

El diseño de la interfaz no fue incluida en los requisitos, sino que se incluyó como parte del trabajo de desarrollo. Opté por realizar la interfaz más sencilla posible para que fuera intuitiva para el usuario. También con esta finalidad se siguieron los estándares de diseño de iOS mostrando menús de navegación inferiores que están presentes en las aplicaciones nativas como iTunes y App Store desde la primera versión de iOS.



FIGURAS 13 y 14: Comparativa de aplicación App Store de Apple y glUCModel. Se puede observar el mismo diseño en el menú inferior.

Asimismo se ha procurado realizar un diseño moderno adaptado a la última versión de iOS (iOS 7). Esto se puede observar en detalles como el botón redondeado de subir datos que imita el diseño de los botones que se introdujeron con la iOS 7.

Los fondos de las pantallas son colores degradados que fueron cuidadosamente elegidos, tras pasar por varios procesos de selección contando con opiniones externas de usuarios que probaron la aplicación. Varios de los diseños presentaban tonalidades que causaban incluso sensación de estrés. Los colores fueron sustituidos por gamas de tonalidades pastel, menos agresivas. El resultado es una interfaz que es suave a la vista.

FIGURAS 15 y 16: Muestra de botones numéricos introducidos en iOS 7 y muestra del botón de SUBIR realizado a semejanza de estos.

4.4 - RETOS DE DESARROLLO

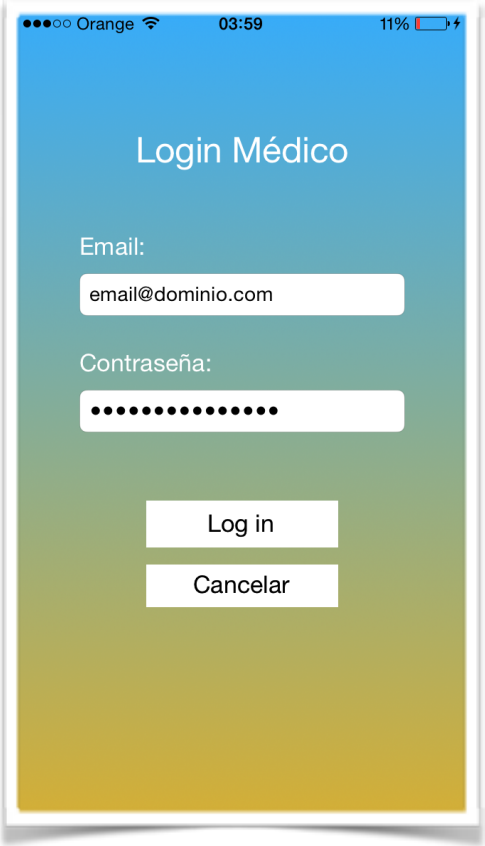
4.4.1 - CONEXIÓN CON SQL DESDE IOS

Sin duda el principal reto de desarrollo supuso lograr el acceso a la base de datos SQL desde Objetive-C. Tras realizar diversas pruebas con librerías los resultados no fueron satisfactorios, por lo que se tomó la decisión de realizar unas API'S (Application Programming Interfaces o Interfaces de programación de aplicaciones) para acceder a los datos y realizar su inserción en la base de datos.

Dichas API´S se realizaron en el lenguaje de programación PHP puesto que serían alojadas en el mismo servidor donde se encontraba la base de datos de glUCModel de tal forma que el acceso fuera más rápido. PHP tiene una buena integración con SQL mediante librerías como mysql, *mysqli o PDO que* están incluidas en PHP. Se optó por usar mysqli ya que es la que presenta mejores resultados de velocidad y rendimiento y está orientada a objetos.

4.4.2 - SEGURIDAD DE LA APLICACIÓN

El acceso a la aplicación debía estar protegido de modo que no se pudieran consultar los datos de un paciente o insertar más datos a su historial si no se poseía la contraseña. Por tanto cuando se abre la aplicación esta solicita las credenciales de acceso del usuario (email y contraseña) como se muestra en las figuras 17 y 18. Esto sucede tanto desde la zona de médicos como desde la zona de usuarios. La contraseña no se muestra mientras es introducida para evitar que sea vista por una tercera persona.



FIGURAS 17 y 18: Pantallas de login de pacientes y médicos. Las contraseñas no se muestran para evitar que puedan ser descubiertas por una tercera persona.

4.4.3 - SEGURIDAD DE LA BASE DE DATOS

La base de datos SQL está protegida mediante usuario y contraseña, por lo que este aspecto no debía ser abordado. La preocupación se centra por tanto en la seguridad de las API'S desarrolladas y en la conexión realizada con las mismas.

4.4.4 - INTEGRIDAD DE LAS CREDENCIALES DE ACCESO

Los datos sensibles que deben ser protegidos son sin duda los de Login. La petición de login se realiza mediante POST y la variable que contiene la contraseña viaja cifrada. Es en la propia aplicación iOS donde se realiza el cifrado. Este cifrado está realizado mediante una función de encriptado MD5, es el mismo algoritmo que utiliza glUCModel en la aplicación web, a diferencia de que el cifrado en este caso es realizado antes de realizar la petición y no después, por lo que la seguridad es aún mayor que en la propia aplicación web.

4.4.5 - PREVENCIÓN DE INYECCIÓN SQL

Otra preocupación fue sin duda la posibilidad de que se inyectara código SQL en las peticiones a la API. Para prevenir esto las API'S están protegidas de modo que todas las peticiones que son recibidas se filtran tras su recepción para realizar un *escapado* de los caracteres especiales que no sean alfanuméricos y puedan provocar un agujero de seguridad.

Tras realizar esta prevención se realizaron diversas pruebas en las que se trató de burlar la seguridad de la API cerrando consultas SQL para realizar volcados de la base de datos, inserciones o borrados. No se consiguió burlar la seguridad, esto demostró que se trataba de API'S seguras.

4.4.6 - ALMACENAMIENTO LOCAL DE LOS DATOS

Cuando el usuario no dispone de conexión a internet los datos son almacenados en el dispositivo para posteriormente insertarlos en la base de datos. Si el usuario cierra la app los datos se perderían, por lo que se planteó que los datos que no se habían podido subir fueran almacenados de forma local.

iOS soporta diversos tipos de almacenamiento local. El más sencillo, ligero y rápido es NSUserDefaults. La desventaja de este sistema es que sólo permite almacenar tipos nativos de Objetive-C. Nuestra necesidad era la de almacenar objetos "insulina", objetos "glucemia", objetos "dieta"… por lo que a priori este método no nos servía. Se tomó la decisión de realizar una función que convertía los datos al tipo String para almacenarnos con NSUserDefaults. Al iniciarse la aplicación los datos son recuperados y convertidos de nuevo a los objetos definidos por nosotros (Glucemia, Insulina, Ejercicio… etc).

FIGURA 19: Pantalla que informa al usuario de que sus datos no han sido subidos. Se almacenarán de forma local.

4.5 - MANTENIBILIDAD DE LA APP

Para que la aplicación pueda ser mantenida más adelante quizá por otro desarrollador o equipo de desarrolladores, el código ha sido debidamente comentado con aclaraciones que indican de forma clara la finalidad de cada parte de código como muestra la figura 19.

```
- (IBAction)uploadPushed:(id)sender {

    //Quito el boton muestro el indicador y lo muevo
    [self.uploadButton setHidden:true];
    [self.indicator setHidden:false];
    [self.indicator startAnimating];

    //retraso un segundo la ejecucion de la subida
    //para que el usuario vea la ruleta y sepa que se está subiendo el dato
    int64_t delayInSeconds = 1;
    dispatch_time_t popTime = dispatch_time(DISPATCH_TIME_NOW, delayInSeconds * NSEC_PER_SEC);
    dispatch_after(popTime, dispatch_get_main_queue(), ^(void){

        //convierto la fecha en un string de mysql
        NSDateFormatter *gmtDateFormatter = [[NSDateFormatter alloc] init];
        gmtDateFormatter.timeZone = [NSTimeZone timeZoneWithName:@"Europe/Spain"];
        gmtDateFormatter.dateFormat = @"yyyy-MM-dd HH:mm:ss";
        NSString *fechaPeso = [gmtDateFormatter stringFromDate:self.datePicker.date];

        //meto los datos en un objecto Peso
        Peso *nuevoPeso = [[Peso alloc]init];
        nuevoPeso.fechaPeso = fechaPeso;
        nuevoPeso.valorPeso = [self.pesoTextField text];

        //Llamo al singleton de usuario para usar el id
        Usuario *usuarioSingleton = [Usuario usuarioSingleton];

        //Llamo a mi JSONHelper para que suba los datos
        [JSONHelper uploadPeso:(usuarioSingleton.userID) :(nuevoPeso)];
        //Paro el indicador, lo escondo y muestro el boton
        [self.indicator stopAnimating];
        [self.indicator setHidden:true];
        [self.uploadButton setHidden:false];
    });
}
```

FIGURA 20: Extracto de código de la aplicación donde se observa que el código contiene comentarios aclaratorios.

El desarrollo de la interfaz se ha realizado casi en su totalidad con la herramienta gráfica, puesto que si se desarrolla con código sería de una dificultad muy elevada tratar de comprender y modificar el código. Cuando se usa la interfaz gráfica se obtienen resultados fáciles de comprender como muestra la figura 20.

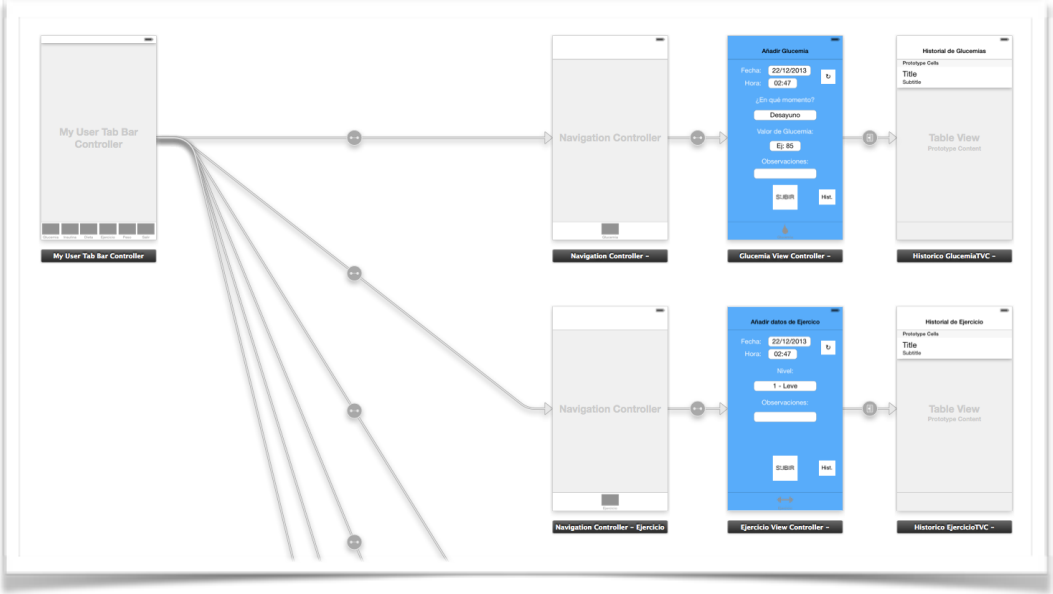


FIGURA 21: Captura del archivo que genera el menú inferior y sus ventanas.

4.6 - LA SALUD DEL USUARIO

Ha sido incorporada una mecanismo de seguridad en la aplicación que informa al usuario si sus niveles de azúcar en sangre suponen un riesgo para su salud.

Cuando se introduce una medición alta que se encuentra en valores considerados de hiperglucemia, o cuando se introduce un valor tan bajo que se encuentra en zona de hipoglucemia, la aplicación confirma este dato, y si el usuario indica que es correcto se le recomienda un plan de actuación.

Los valores considerados como peligrosos han sido descritos por los médicos endocrinos involucrados en el proyecto de glUCModel, los valores hiperglucémicos deterioran la salud del paciente a largo plazo, y los valores hipoglucémicos pueden causar la pérdida del conocimiento al paciente. Las figuras 22 y 23 contienen capturas de los mensajes que son mostrados al usuario.

FIGURAS 22 y 23: Mensajes de aviso en caso de Glucemia elevada.

5. RESULTADO

El resultado del desarrollo ha sido satisfactorio. Nos encontramos ante una aplicación que cumple los requisitos que se propusieron al inicio y durante el desarrollo.

Se trata de una aplicación ligera, segura y rápida.

Tras varios estudios de rendimiento, se ha podido comprobar que inicialmente ocupa unos 20MB en memoria RAM y que tras un uso intenso de aplicación esta ocupa en memoria RAM en torno a 50MB, el uso que se da del procesador nunca supera el 1% puesto que la aplicación no necesita realizar cálculos. La figura 24 muestra los datos de uso de la memoria RAM y el procesador tras un uso intenso.

Se ha estudiado también el espacio que ocupa en disco la aplicación una vez es compilada. Se ha observado que la aplicación no llega a ocupar en disco 1MB. Si contamos con los datos que se almacenan de forma local, puede ocupar 1,5MB o 2MB como se observa en la figura 25.

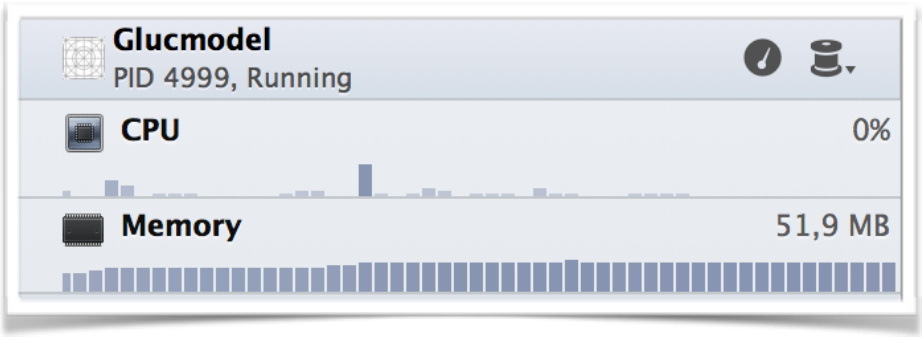


FIGURA 24: Monitor de recursos del iPhone durante la ejecución de glUCModel

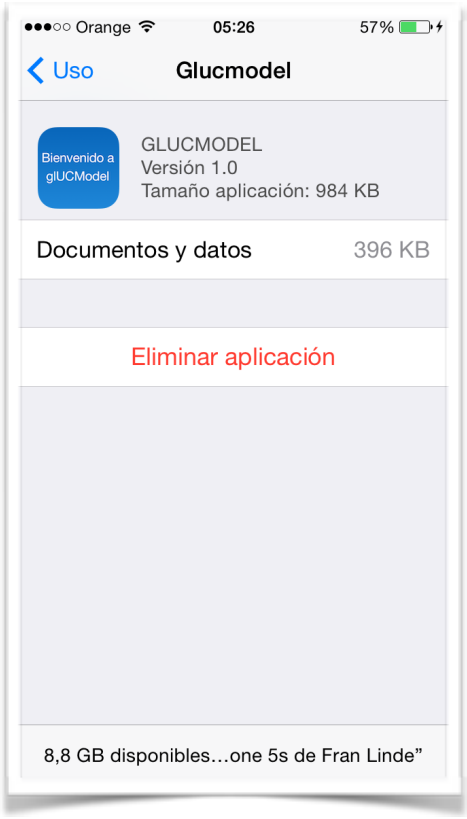


FIGURA 25: Captura de pantalla, espacio en disco de glUCModel.

En cuanto a la interfaz los resultados son a mi juicio buenos. Los test de usabilidad realizados han demostrado que la interfaz es amigable e intuitiva. Las opiniones recabadas acerca de la experiencia de usuario son buenas.

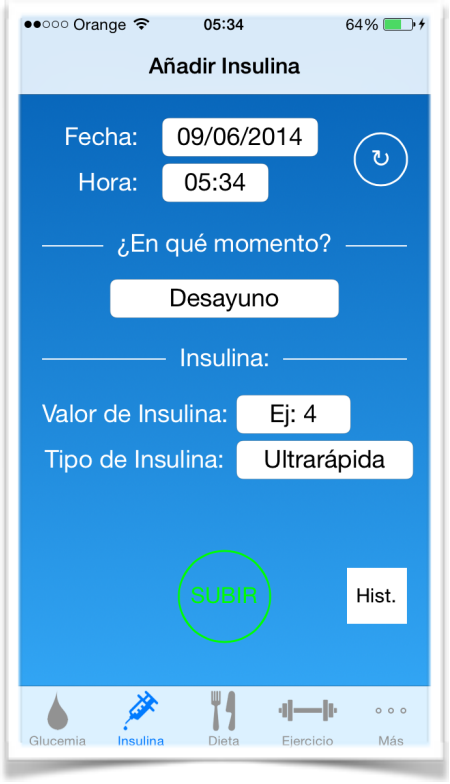


FIGURA 26: Inserción de Insulinas.

FIGURA 27: Historial de Insulina

FIGURA 28: Historial de Glucemias.

FIGURA 29: Inserción de datos de Peso

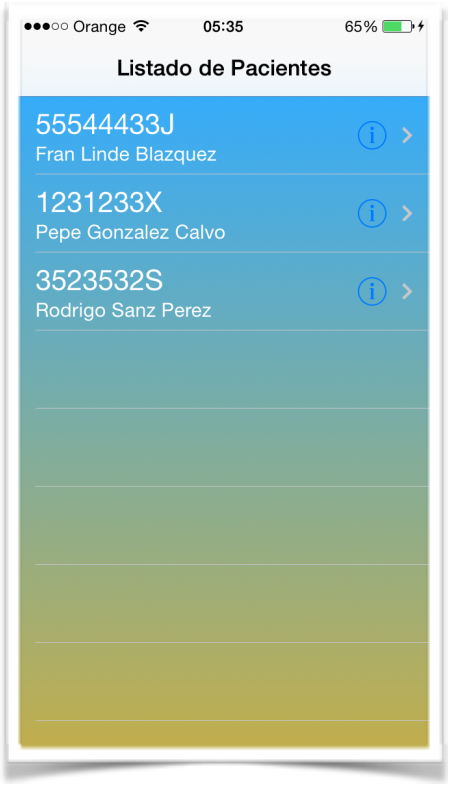


FIGURA 25: Listado de Pacientes de un médico.

6. CONCLUSIONES

El desarrollo del proyecto ha culminado de forma satisfactoria y la colaboración con glUCModel ha sido beneficiosa para ambas partes. Por un lado glUCModel ahora cuenta con una aplicación para iOS nativa que cumple con los requisitos que requería. Por otro lado como desarrollador, he adquirido conocimientos de programación en Objetive-C lo cual le aporta un gran valor a mi curriculum y a mi potencial como autónomo.

Me he enfrentado a problemas durante el desarrollo que he sabido solventar de forma correcta, prueba de ello es que la aplicación ha salido adelante y funciona correctamente.

Lamentablemente el desarrollo ha estado muy condicionado por los plazos de entrega y el calendario universitario, de haber tenido disponibilidad total de tiempo y calendario, la aplicación gozaría de más funcionalidades.

7. FUTURO DE LA APLICACIÓN

La aplicación tiene mucho potencial puesto que el desarrollo ha sido llevado a cabo por una sola persona. Disponiendo con más recursos humanos se podrían incluir más funcionalidades a la aplicación. Detallo una lista de posibles mejoras:

- Incorporar otros idiomas.
- Sistema de predicción de glucemia basado en el algoritmo de predicción que se está desarrollando.
- Sistema de recomendación de dosis de insulina basado en el algoritmo de predicción que se está desarrollando.
- Comunicación directa con médico-paciente mediante la app.
- Comprobación periódica de la conexión a internet para subir los datos pendientes de forma automática.
- Diseño específico mejorado de la interfaz para los diferentes terminales (iPhone 4, iPhone 5, iPad, iPad retina, iPad mini… etc)
- Optimización del código para conseguir mayor velocidad.

8. BIBLIOGRAFÍA

1 - J.I. Hidalgo, et al., Modeling glycemia in humans by means of Grammatical Evolution, Appl. Soft Comput. J. (2013), http://dx.doi.org/10.1016/j.asoc.2013.11.006

2 - Standford University Online Course: Developing iOS 7 Apps for iPhone and iPad, http://online.stanford.edu/course/developing-ios7-apps-fall-2013

3 - J.I. Hidalgo, et al. glUCModel: A monitoring and modeling system for chronic diseases applied to diabetes. J Biomed Inform (2014), http://dx.doi.org/10.1016/j.jbi.2013.12.015

4 - A. American-Diabetes-Association. Standards of medical care in diabetes 2010. Diabetes Care, 33(S1):11–61, 2010.

5 - Apple iOS Developer Library: Programming with Objective-C, https://developer.apple.com/library/ios/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/ProgrammingWithObjectiveC.pdf

6 - Apple iOS Developer Library: iOS 7 Design Resources, iOS Human Interface Guidelines, https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/MobileHIG.pdf

7 - Hidalgo JI et al. glUCModel: Clarke and Parkes Error Grid Analysis of Diabetic Glucose Models obtained with Evolutionary Computation

8 - iOS Developer Library: iOS App Programming Guide, https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/iPhoneAppProgrammingGuide.pdf

9 - Zach Waugh, a Mac & iOS developer: How to use JSON in Cocoa/Objective-C, http://zachwaugh.me/posts/how-to-use-json-in-cocoaobjective-c/

10 - Andrey Prikaznov: Create Your Own XML/JSON/HTML API with PHP, http://css.dzone.com/articles/create-your-own-xmljsonhtml

11 - Manual de PHP, Seguridad Seguridad de Bases de Datos: Inyección de SQL, http://www.php.net//manual/es/security.database.sql-injection.php

12 - iOS Developer Tips: Create MD5 Hash from NSString, NSData or a File, http://iosdevelopertips.com/core-services/create-md5-hash-from-nsstring-nsdata-or-file.html

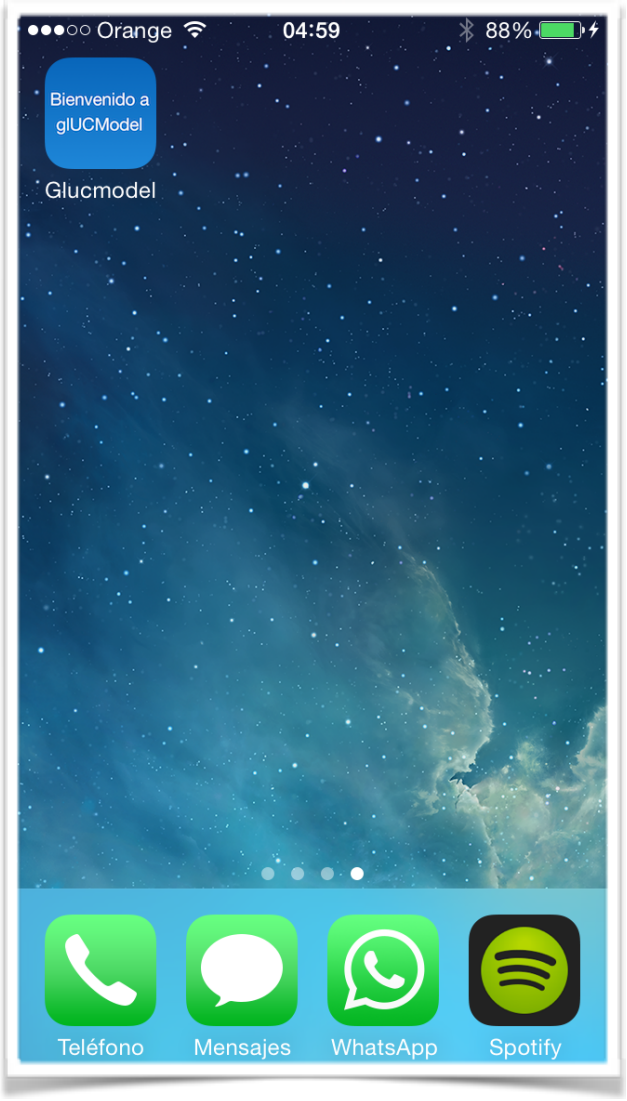
13 - Ray Wenderlich, How To Write A Simple PHP/MySQL Web Service for an iOS App, http://www.raywenderlich.com/2941/how-to-write-a-simple-phpmysql-web-service-for-an-ios-app

14 - Tristan Louis, How Much Do Average Apps Make?, http://www.forbes.com/sites/tristanlouis/2013/08/10/how-much-do-average-apps-make/

APÉNDICE: MANUAL DE USO (PACIENTE)

PASO 1: ABRIR LA APLICACIÓN.

Debemos localizar el icono de la aplicación en nuestro dispositivo. Esta tiene un logo azul con la frase "Bienvenido a glUCModel". Debemos pulsar sobre este icono para abrirla



PASO 2: SELECCIONAR TIPO USUARIO

Debemos seleccionar si vamos a utilizar el modo paciente o el modo médico. Por supuesto esto dependerá del tipo de usuario que seamos.

Vamos a mostrar en primer lugar el uso en el modo paciente. Para entrar en el modo paciente debemos pulsar donde se indica "Paciente".

PASO 3: INTRODUCIR USUARIO Y CONTRASEÑA

Para poder continuar debemos hacer el proceso de Log In. Para ello debemos introducir nuestro email de usuario, y nuestra contraseña. El campo contraseña no será visible por motivos de seguridad.

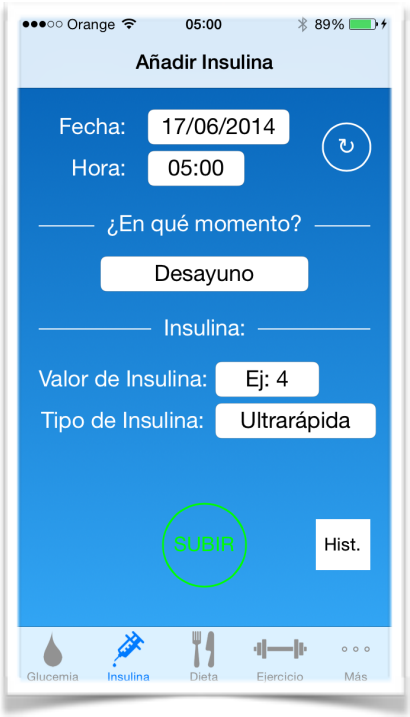
Si pulsamos en Log In se comprobarán las credenciales de acceso, en caso de ser correctas se pasará a la siguiente pantalla, de no ser así deberán volver a introducirse. Si pulsamos en cancelar volveremos a la pantalla del paso 2.

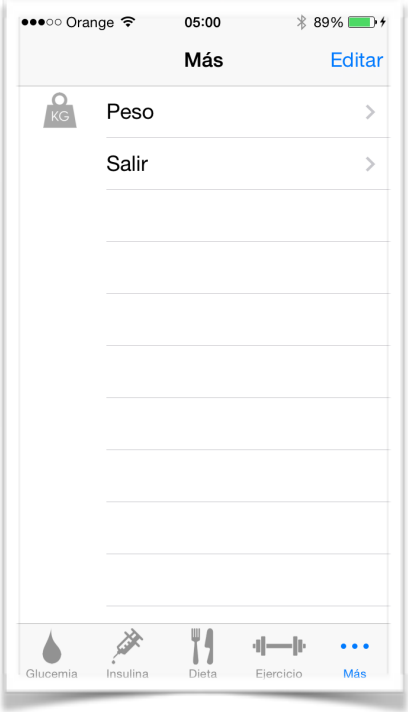
PASO 4: NAVEGACIÓN POR EL MENÚ INFERIOR

Una vez hayamos hecho el proceso de Log In podremos ver esta pantalla:

Esta es la pantalla donde se introducen los nuevos datos de glucemias. En la parte inferior de la pantalla podemos observar el resto de pestañas del menú.

Podemos acceder a Glucemia, Insulina, Dieta, Ejercicio y más. Dentro de más encontraremos Peso y Salir. En las siguientes imágenes se muestran las pantallas disponibles en orden de izquierda a derecha.



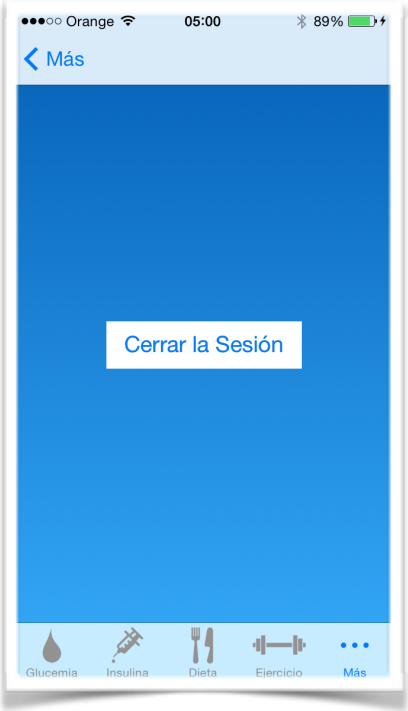


Si pulsamos en más veremos esta pantalla que nos muestra otras dos opciones: Peso y Salir.

Desde la pantalla de peso podremos añadir nuestro peso corporal para realizar un seguimiento de este

Si pulsamos en salir, se cierra la sesión iniciada en el paso 2 de tal modo que se pasa al paso 1 de nuevo.

A continuación se muestran unas capturas de la interfaz de subida de datos de peso y de la interfaz para cerrar sesión o Log Out.



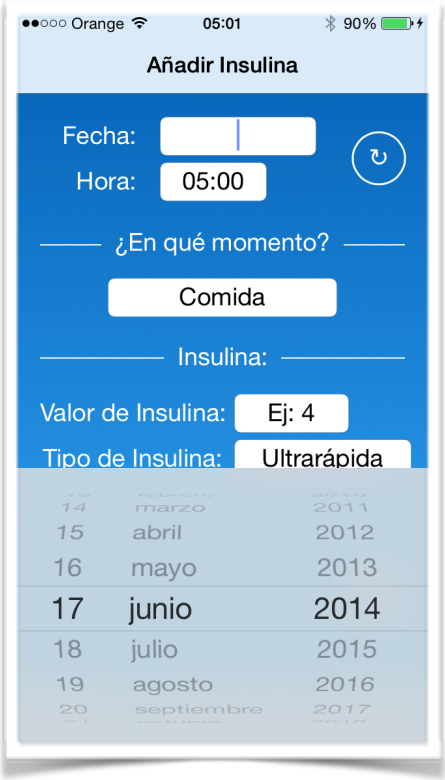
PASO 5: SUBIR UN DATO

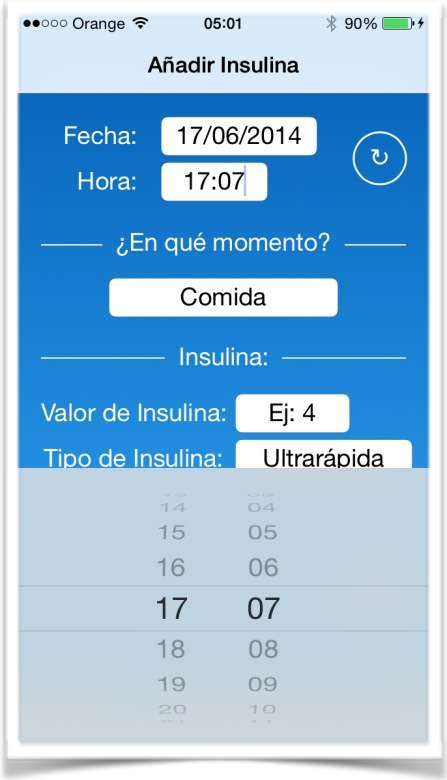
Las interfaces para la subida de datos son similares unas de otras, esto facilita adaptarse a la app de forma rápida. Para subir un dato debemos seleccionar la opción del menú que nos interese subir. Tras esto debemos seleccionar la fecha y la hora, para lo cual disponemos de selectores que sólo nos ofrecerán sólo fechas y horas válidas.

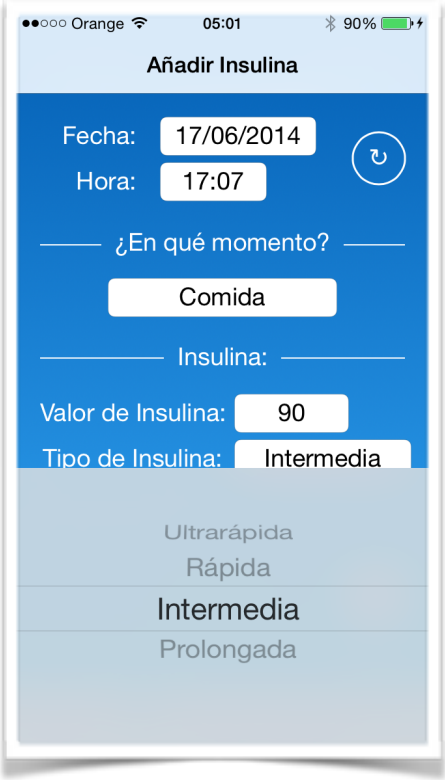
Usaremos la interfaz de subida de datos de insulina como ejemplo para las instrucciones.

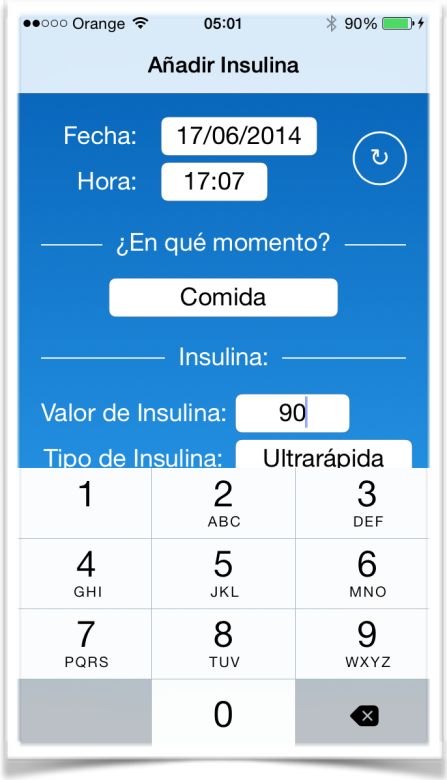
Debemos también indicar en qué momento se ha realizado esta medición, para lo cual disponemos de un selector que sólo nos ofrecerás las opciones de momentos disponibles.

En este caso al tratarse de insulina también debemos indicar la cantidad y el tipo de insulina administrados.



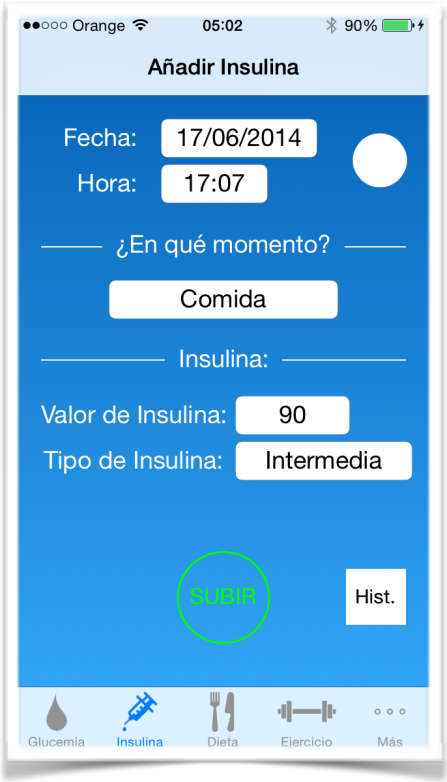






Para facilitar la selección de la fecha y la hora, se ha habilitado un botón con forma de flecha girando que al pulsar hace que la fecha y la hora sean las de ese momento.

En la imagen se muestra el botón pulsado.



Durante la subida del dato el botón de subir se sustituye por una ruleta de proceso que nos inicia que en estos momentos la app está subiendo un dato. Una vez terminada la subida el botón vuelve a estar disponible.

Al terminar de subir el dato también se muestra al usuario un mensaje de éxito.

SUBIR UN DATO SIN CONEXIÓN A INTERNET

Si la subida de los datos fallase la aplicación se lo notificaría al usuario y almacenaría los datos de forma local en el dispositivo para posteriormente subirlos cuando sea posible.

El usuario sabe que hay un dato almacenado por que se muestra un globo en el menú inferior que indica el número de datos pendientes de ser subidos.

LA SALUD DEL PACIENTE: SUBIR UN DATO PELIGROSO

Si el usuario sube un dato cuyos niveles suponen un riesgo para s salud, la aplicación le informará debidamente. Esto sucede por ejemplo con las glucemias elevadas o las glucemias excesivamente bajas. Las siguientes capturas muestran los mensajes que la aplicación genera.

PASO 5: CONSULTAR EL HISTORIAL DE DATOS

Para mostrar su historial, un paciente debe acceder a la pestaña del menú que corresponde a los datos que quiera consultar. Por ejemplo si desea mostrar su historial de insulinas deberá pulsar en insulinas.

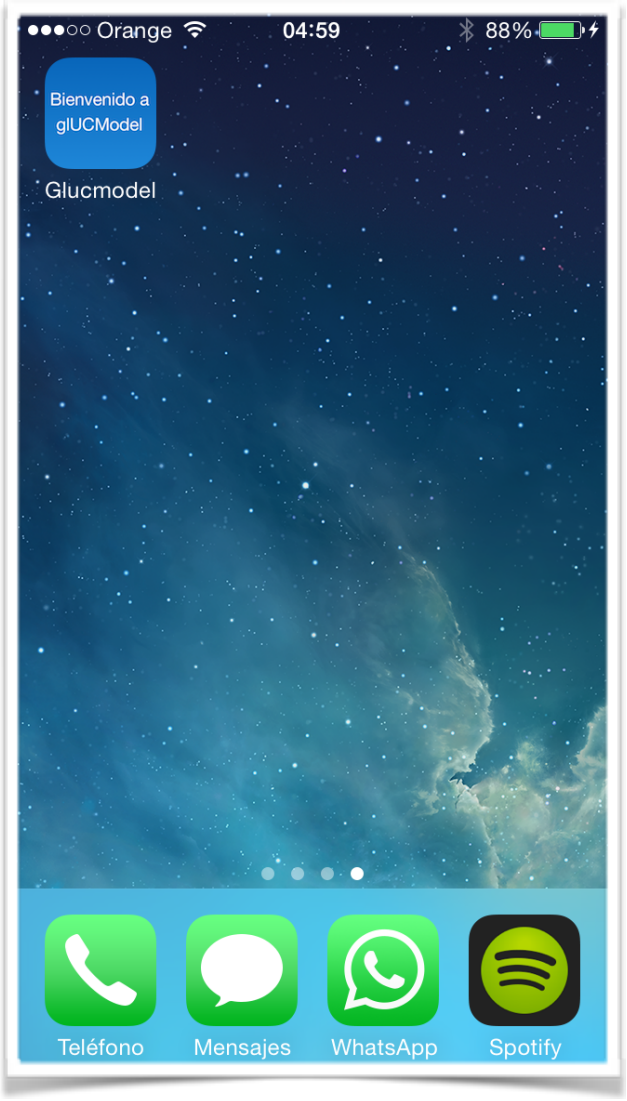
Una vez se muestre la pantalla de subida de datos, deberá pulsar en el botón que se encuentra en la parte inferior derecha que dice "Hist." (Historial).

El usuario podrá ver una lista con sus datos. En caso de querer regresar, el usuario encontrará un botón que dice "Atrás" en la parte superior izquierda de la pantalla.

APÉNDICE: MANUAL DE USO (MÉDICO)

PASO 1: ABRIR LA APLICACIÓN.

Debemos localizar el icono de la aplicación en nuestro dispositivo. Esta tiene un logo azul con la frase "Bienvenido a glUCModel". Debemos pulsar sobre este icono para abrirla



PASO 2: SELECCIONAR TIPO USUARIO

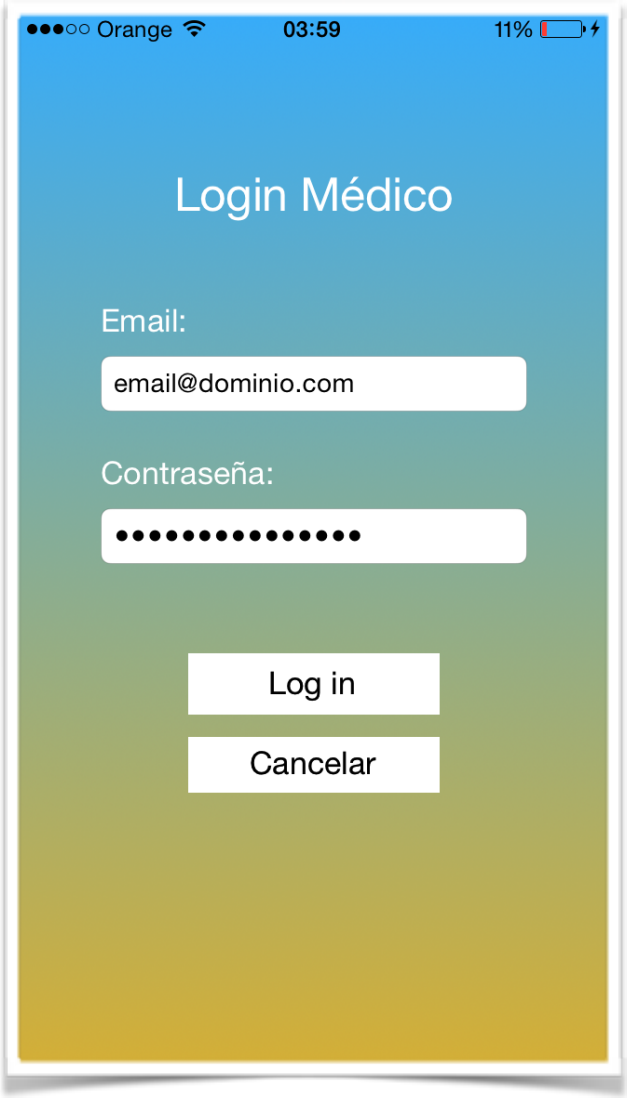
Debemos seleccionar si vamos a utilizar el modo paciente o el modo médico. Por supuesto esto dependerá del tipo de usuario que seamos.

Vamos a mostrar el uso en el modo médico. Para entrar en el modo médico debemos pulsar donde se indica “Médico”.

PASO 3: INTRODUCIR USUARIO Y CONTRASEÑA

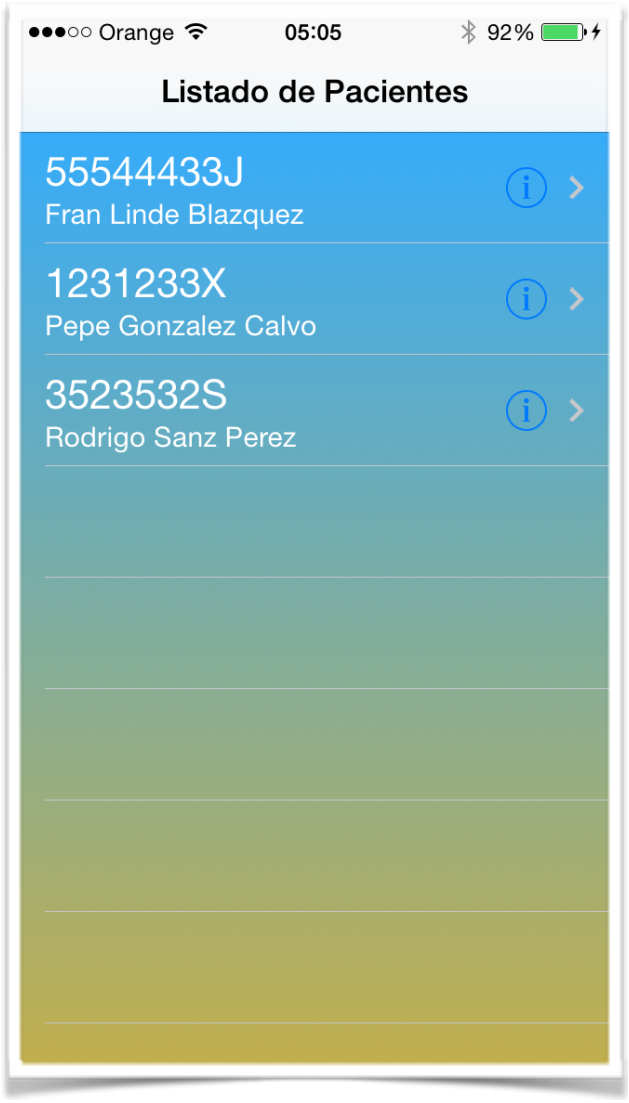
Para poder continuar debemos hacer el proceso de Log In. Para ello debemos introducir nuestro email de usuario, y nuestra contraseña. El campo contraseña no será visible por motivos de seguridad.

Si pulsamos en Log In se comprobarán las credenciales de acceso, en caso de ser correctas se pasará a la siguiente pantalla, de no ser así deberán volver a introducirse. Si pulsamos en cancelar volveremos a la pantalla del paso 2.



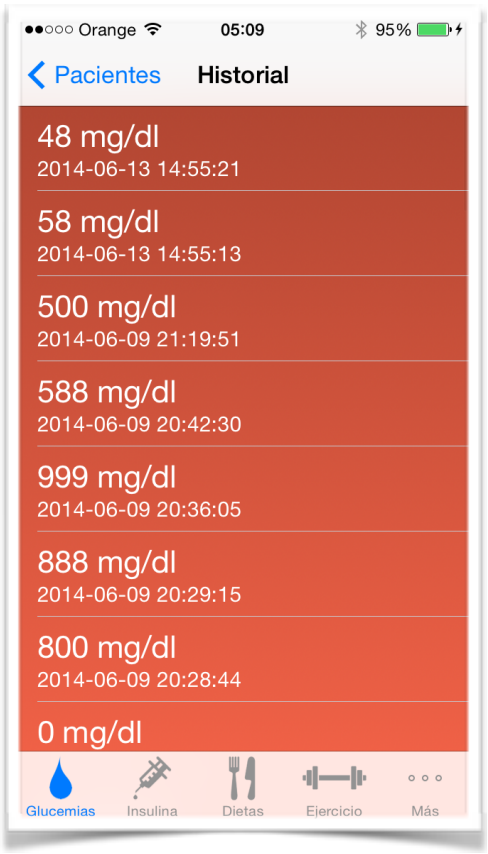
PASO 4: SELECCIONAR UN PACIENTE ASOCIADO

Una vez hayamos hecho el proceso de Log In podremos ver esta pantalla:



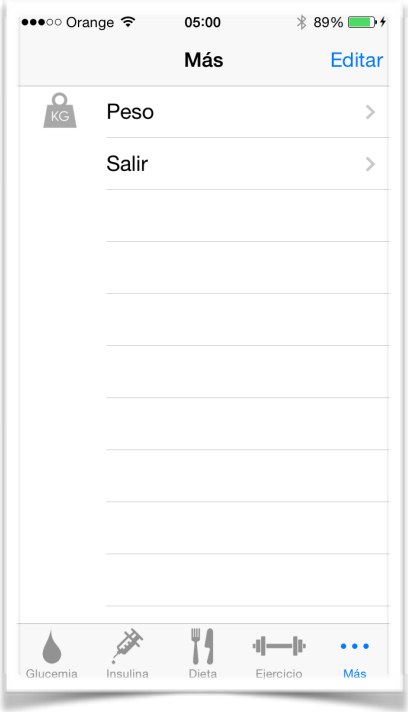
En esta pantalla el médico podrá ver los pacientes que tenga asociados, y podrá seleccionarlos para visualizar los datos que estos hayan subido con la aplicación o mediante la web.

Una vez seleccionado el paciente podemos acceder a sus datos de Glucemia, Insulina, Dieta, Ejercicio y más, seleccionando la opción adecuada dentro del menú inferior. Dentro de "más" encontraremos Peso y Salir. En las siguientes imágenes se muestran las pantallas disponibles en orden de izquierda a derecha.




●●○○○ Orange
05:09
95%
Pacientes
Historial
1 raciones en Recena
2014-04-28 02:57:40
10 raciones en Desayuno
2014-04-28 01:21:30
20 raciones en Recena
2014-03-17 05:42:34
10 raciones en Recena
2014-03-17 05:20:40
8 raciones en Cena
2014-03-12 00:00:00
5 raciones en Merienda
2014-03-11 00:00:00
10 raciones en Comida
2014-03-05 00:00:00
20 raciones en Comida
Glucemias
Insulina
Dietas
Ejercicio
Más

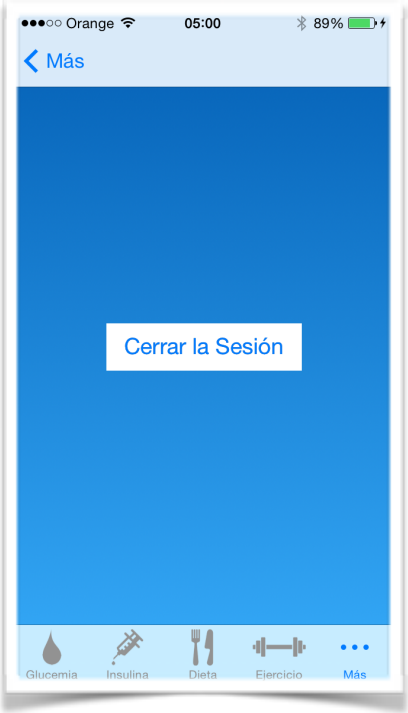

●●○○○ Orange
05:09
95%
Pacientes
Historial
Ejercicio de nivel 9
2014-06-09 05:23:46
Ejercicio de nivel 2
2014-04-28 10:27:37
Ejercicio de nivel 0
2014-04-28 02:57:20
Ejercicio de nivel 8
2014-04-28 01:21:41
Ejercicio de nivel 9
2014-03-17 20:27:25
Ejercicio de nivel 2
2014-03-17 05:43:20
Ejercicio de nivel 4
2014-03-17 05:40:24
Ejercicio de nivel 7
Glucemias
Insulina
Dietas
Ejercicio
Más

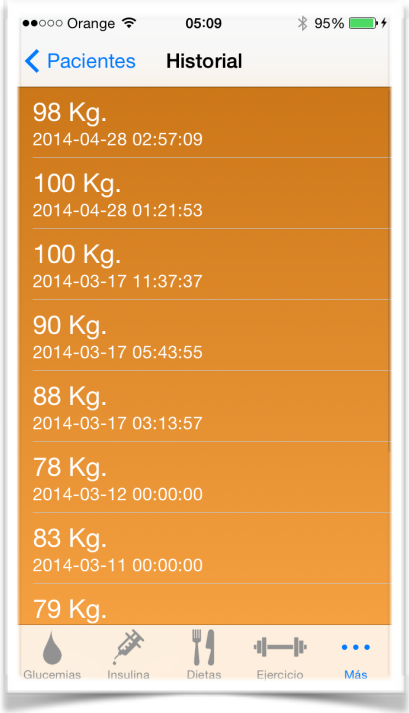


Si pulsamos en más veremos una pantalla que nos muestra otras dos opciones: Peso y Salir. Desde la pantalla de peso podremos visualizar los datos de peso almacenados por el paciente.

Si pulsamos en salir, se cierra la sesión iniciada en el paso 2 de tal modo que se pasa al paso 1 de nuevo.

A continuación muestro unas capturas de la interfaz para cerrar sesión o Log Out.





glUCModel for iOS

BY: FRANCISCO JAVIER LINDE BLÁZQUEZ

GRADO EN INGENIERÍA DEL SOFTWARE

FACULTAD DE INFORMÁTICA

DEPARTAMENTO DE ARQUITECTURA DE COMPUTADORES Y AUTOMÁTICA

UNIVERSIDAD COMPLUTENSE DE MADRID

FINAL YEAR PROJECT IN SOFTWARE ENGINEERING

Madrid, June 18, 2014

Director: José Ignacio Hidalgo

Colaborator: J. Manuel Colmenar

AUTHORIZATION FOR DISTRIBUTION AND USE

Dedicated to: Gerardo Fernández Moreno.

He is an example of hard work and perseverance.

He has been a motivation for me.

ACKNOWLEDGEMENTS

My sincere thanks to my project manager **José Ignacio Hidalgo** for all time spent, for his interest and for his patience and kindness during project development.

Thanks to **José Manuel Colmenar** for his collaboration in the project, has brought light on those technical issues that required consultation.

Thanks to my partner **Abdelkhalek El Mansouri** for working in those common tasks of the project and for providing the help he has been able in the project.

INDEX

User Manual can be found in spanish at page 45.

1. INTRODUCTION

Technology should play an important role in our society, and we must not only contemplate it as a tool to help us do what we already know, but we must go further and think of it as a tool to make things so far could not do things even imagined.

However, sometimes the tremendous potential of new technologies
not being fully exploited, and this because many times the tools have been developed but are not yet identified the problems we want to solve.

Many people in their daily lives have to struggle with personal problems, work problems, financial, family, health ... It's in the last one in which we will focus.

And according to a study by di@bet.es in 2011 13.8% of Spaniards aged over 18 have diabetes, and worldwide there are over 260 million people with diabetes. These data shows the magnitude of the problem.

1.1 - LET'S TALK ABOUT DIABETES

Diabetes Mellitus (commonly known as diabetes) is a disease that affects metabolism. This is not a temporary illness, lifelong. Characterized by that the patient exhibits high levels of blood glucose (hyperglycemia).

The American Diabetes Association Diabetes classified into 2 types:

- Type 1: The patient's pancreas produces insulin, so that blood glucose levels can not be adequately reduced. These patients are insulin dependents, that mean that they require an external supply of insulin in

their daily lives. How to incorporate this contribution may be insulin by a pump or by injections.

- Type 2: The patient produces insulin to lower right and also the body shows glucose tolerance dose, making it less effective in reducing blood glucose levels. These patients are not insulin dependents.

Fortunately over 90% of diabetics is in the second group, and say fortunately because this type of diabetes is less disabling than type 1 diabetes. The main treatment for type 2 diabetes is to achieve a healthy weight and maintain power healthy, but in some cases require treatment in pill form or in advanced stages also need external insulin intake.

Type 1 diabetics are both diabetics who see their lives affected disproportionately. Must maintain a monitoring of your blood glucose levels to prevent both hyperglycemia (levels> 120mg/dl) and hypoglycemia (levels <40mg/dL). The careful control mentioned is blood glucose measurements to make predictions based on subsequent food intake, the sport predicted they are going to do, the weight they have at the moment, and many other factors that influence in blood glucose levels. Based on this prediction should make an estimate of insulin to be injected to keep their blood glucose levels appropriate values.

1.2 - GLUCMODEL

The task of a type 1 diabetic is not easy. In most cases patients refine their predictions with practice, but even with many years of experience the patient error their estimates. It is in this daily task that glUCModel wants to play a big role. The idea behind glUCModel is to be able to predict the patient's blood levels based on current parameters of the subject as are the next level of sugar or carbohydrate intake to advise a certain amount of insulin.

To calculate the future sugar levels glUCModel is developing evolutionary grammars based genetic programming. This type of definition allows greater freedom in defining the mathematical

expression of the solution, since it does not have restrictions such as can be found using linear equations.

The project is several years developed through the collaboration of the Universidad Complutense de Madrid, University Hospital Príncipe de Asturias (Alcalá de Henares, Spain) and el hospital Virgen de la Salud (Toledo, Spain). The professionals involved in the project are: *J. Ignacio Hidalgo, José L. Risco-Martin, Juan Lanchares y Oscar Garnica del departamento de Arquitectura y Autómatas de la Universidad Complutense de Madrid.*

- *J. Manuel Colmenar y Alfredo Cuesta-Infante from C.E.S. Felipe II, Universidad Complutense de Madrid.*
- *Esther Maqueda from departamento de nutrición y endocrinología del Hospital Virgen de la Salud (Toledo, España)*
- *Marta Botella y José Antonio Rubio from departamento de nutrición y endocrinología del hospital Universitario Príncipe de Asturias (Alcalá de Henares, España).*

1.3 - THE PROJECT: THE MOBILE APPLICATION

One of the main needs of glUCModel is collecting data, without it they could not find the formula for estimation of glucose, on the other hand also could apply this formula to inform the patient of insulin be administered.

To meet this need a web application to enter data and access was made, but the user needs to enter data at all times and no doubt that the user does not have a computer 24 hours a day. It is possible to record the information on a notepad to enter them later but it is a convenient solution, so we chose the best solution for the patient: Create a mobile application.

No one leaves his mobile phone, the data are revealing: In Spain there are 26 million smartphones with internet access and 53.8% of Spanish mobile terminal navigate using on a daily basis. These data

were collected and published by the Telefónica Foundation in early 2014.

In 2013 they were offered two draft final project for the development of glUCModel mobile phones: A version for Android and iPhone version in order to provide users of Glucmodel a simple, accessible and friendly way for the incorporation of information a database. I chose iPhone because I think it's a platform with great potential. There are many companies that need developers for iOS on the other hand, the App Store is undoubtedly the most profitable opponent or selling platform applications therefore programming knowledge to me as a freelance iOS developer opens the possibility to create and monetize my own mobile applications.

The iPhone application developed allows the user to enter data on blood glucose, their insulin injections, your meals, your weight and physical activity. The data collected is sent directly to the database which will store glUCModel safely. All data are also accessible from the mobile application for the user to visualize them.

This method to add data to the database is a significant improvement in usability. The user can now use an appropriate interface to the screen size of his Smartphone, and it adds the possibility to use offline. On the Web logically if we lose internet access we can not store any data, but in the iPhone application, we can add data to be stored locally until we can make the upload.

The iPhone does not store any of the data entered by the patient unless it is not possible to store them on the server, in which case they remain in the iPhone until they can be uploaded. The application does not store any personal data of the patient, so that although the data temporarily stored on the iPhone would be compromised, it would be impossible to associate them with an individual, since transactions are conducted using the user ID assigned by glUCModel.

1.4 - PERSONAL MOTIVATION

At the age of 18 and after a year suffer the consequences of having sugar through the roof, was diagnosed with diabetes to my best friend. I could go through the process in person because throughout his time until he was diagnosed as a diabetic I saw he was always tired, as I said he was losing vision, saw him drink two liter bottles of water all at once… In a blood test could show with a sugar above 400mg/dl that he is suffering diabetes.

After being diagnosed with diabetes the first thing to do is logically keep track of your sugar in the first place to learn to control it, and secondly for your endocrine can bring a good control of your disease.

For a few months my friend had to note in a book all the measurements made, the portions of carbohydrate was ingested and insulin was administered. This happened in 2009, then began the boom of mobile apps in Spain, to me I'd just give an iPhone 3G and I looked at the world with different eyes, with eyes of apps developer, wherever you looked I saw an app and had in mind to make an app to help diabetics like my friend to take control of his data. In the list of projects Final Year Work was performing glUCModel app for iPhone, when I saw it I knew I wanted to do it, because for years I had it in mind.

1.5 - PROBLEMS TO SOLVE

As already mentioned glUCModel existed and exists as web application is developed in Java and database is SQL. The first problems arise quickly: iOS does not have good support for SQL. In contrast to other platforms and programming languages do not have iOS integration with SQL, and most developers choose to create their own Web Services to access their SQL databases.

We had to decide what would be the behavior of the app if you do not have internet access.

Another concern was of course the data integrity of the patients. We had to maintain the security of the database and the data that were stored on the device.

Another task was to design a quick and simple interface that anyone with basic knowledge of using a smartphone would be able to use it.

2. ¿QUÉ ES GLUCMODEL?

glUCModel (available online) is a web application that has a dual purpose:

- Improve control of diabetes
- Help diabetics and their doctors to control the disease.

Project requirements were performing platform application, so that the device that the user had is not a constraint. The web application was tested in multiple devices such as smartphones, tablets, laptops and desktops. Various operating systems were also used in the tests: Mac OS, Windows, Linux, iOS and Android; and various web browsers: Internet Explorer, Mozilla Firefox, Google Chrome, Safari and Opera.

After testing it was determined that the application will work correctly on all platforms, but the user experience was affected when using devices with small screens, so it was decided to make native apps for iOS and Android mobile devices.

glUCModel is composed by the following five modules connected together:

1. GUI: The core of the application. Connect all modules. Used to allow patients to query their data, modifications and new data updates. Also doctors can also keep track of the data that patients are added to the application.

2. Database: The database stores information about glUCModel users (patients and doctors), the analysis of patient blood glucose measurements, sport, nutrition, exercise, weight ... etc.

3. Learning Module or e-learning: A virtual learning space where the patient will find all necessary information about diabetes. You can find documents with theoretical concepts about diabetes and tools to help in their education like tests, calendars, forums and glossaries.

4. Glucose Module: Made with evolutionary programming techniques. This module gets a personalized model for each patient using the information provided by the database.

5. Recommender System: The Role of the recommender system is to evaluate patient data. The system generates recommendations on how the patient can improve your quality of life. The patient receives emails notifying these recommendations.

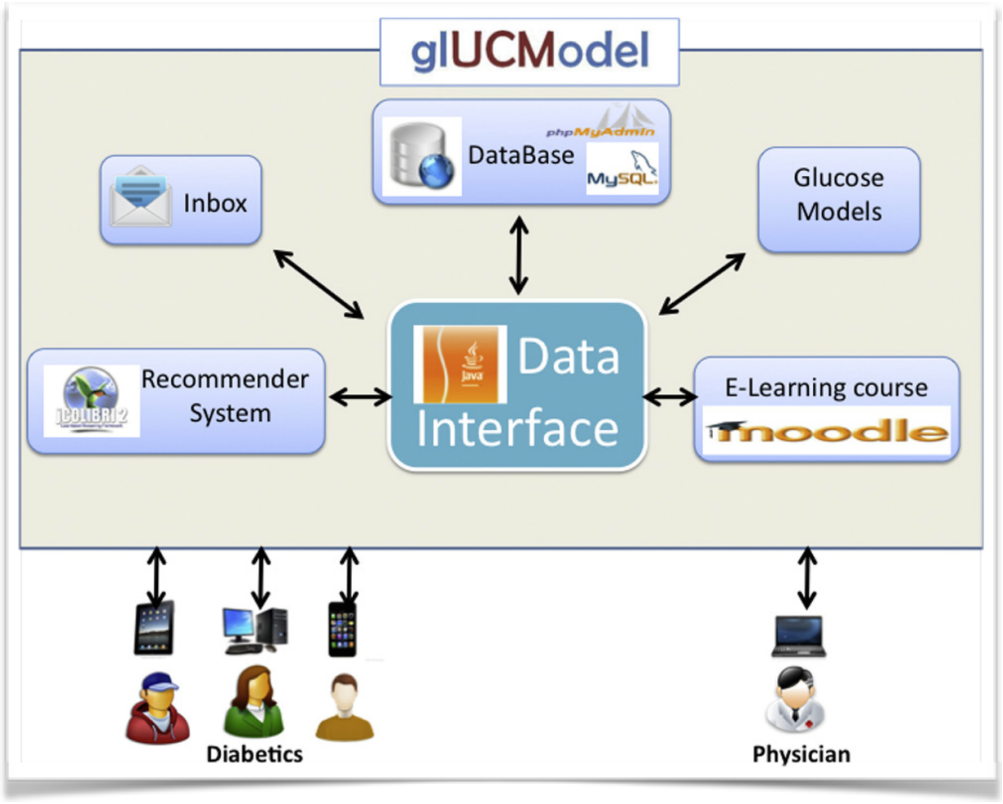


FIGURE 1: General structure of glUCModel (Web Application).

The operation of the application is as follows: patients upload data through the graphical interface. Doctors can review patient data using the graphical interface itself. The glucose model analyzes the information and generates the recommendations for the patient. The learning module is connected to the application to communicate the progress of patients with tests.

3. DEVELOPING ON iOS

3.1 - ¿Why iOS?

The Apple App Store is undoubtedly the most profitable shop in the market today applications. On July 2, 2014 Apple released its number accumulated in the App Store since its release in 2007 at the WWDC (World Wide Developer Conference) downloads. In the 7 years of the App Store cumulative number of downloaded applications amounts to 75,000,000,000 (seventy five billion).

During these seven years Apple has distributed $ 15 billion to developers (fifteen thousand million) dollars. To make a comparison we have chosen two main competitors, Android and Windows Phone. Official data we have collected are:

	Apple	Google	Microsoft
Number of users (in millions)	600	900	12
Number of Apps (in thousands)	1250	800	160
Number of developers (en thousands)	235	150	45
Number of downloads (in billions)	50	48	65
Amount paid to develepores (in million $)	5000	900	100

* All data are for the year 2013

FIGURE 2: Comparative table. Data of app stores and revenues.

All these data can be found on the website of Forbes, with the headline "How Much Do Average Apps Make?"
http://www.forbes.com/sites/tristanlouis/2013/08/10/how-much-do-average-apps-make/

Just look at that have been downloaded approximately the same number of applications on iOS, Android and Windows Phone, but

developers incomes are multiplied by more than 5 to Apple about Android, and more than 50 compared to Microsoft.

With the data obtained previously decided to make a comparative study of profitability which store it was more interesting to the developer.

	Apple	Google	Microsoft
Average of apps by developer	5	5	3
Average of downloads by app	40.000	60.000	4.062
Average benefit by download	0,10$	0,01875$	0,1538$
Average benefit by developer	20.000$	5.625$	1.874$

* All data are for the year 2013

FIGURE 3: Comparative table. Monetizing app stores. Revenues.

If we look at the last row we can see that a currency affects, iOS developer gets 4 times more revenue than Android one and 11 times more than one Windows Phone. The data are clear: The App Store is today the most profitable store for apps.

Therefore as engineer gives me a lot of potential being able to develop applications for iOS, as it could reach some monetize on the App Store.

3.1 - THE LENGUAGE: OBJETIVE-C

Currently you can only develop native applications for iOS in the programming language Objective-C. For the next version of iOS (iOS 8), which is expected to be available in Fall 2014, will also be programmed in Swift. Swift was introduced by Apple on June 2, 2014 so we still do not know much about this new language.

Objective-C is a programming language that appeared in 1980, and as its name suggests is object oriented. It was created by Brad Cox in 1992 and was released under the GPL license.

Objective-C is a thin layer above C, so it is possible to compile any C application with Objective-C compiler, and is also possible to include C code directly into your application in Objective-C.

Is semantically different from Java or C + + since the programmer does not call methods on objects, what he does is send messages to objects to perform the actions we need.

Example of calling a method in C++ and Java

```
object->method (parameter);          object.method (parameter);
```

Example of message sending in Objetive-C:

```
[object method : parameter];
```

Figure 4: Example of calling functions in Java, C + + and Objective-C

This peculiarity makes it a language that at first contact is strange. Sometimes there are causes function calls that occupy several lines of code that do not enter the screen due to how long they are. This is an example of an extremely simple function that can be found within this application. Its purpose is to show a simple alert window.

```
- (void) alertStatus:(NSString *)msg :(NSString *)title
{
    UIAlertView *alertView = [[UIAlertView alloc] initWithTitle:title
                                                        message:msg
                                                       delegate:self
                                              cancelButtonTitle:@"Ok"
                                              otherButtonTitles:nil, nil];

    [alertView show];
}
```

Figure 5: Example of function call on several lines.

The alertStatus function has only two lines of code, builds an UIAlertView the first and the second sends this new object load order. As you can see the constructor call UIAlertView has to be done on multiple lines for reasons of space and clarity.

3.2 - THE DEVELOPMENT ENVIRONMENT: XCODE

Apple is known for being a closed in many aspects of business. In programming have been no exceptions. These are some of the conditions that must be met:

1. The only way to program for iOS or Mac OS in Objective-C using a Mac There is no official alternatives but not feasible.
2. The only development environment can develop, compile and run iOS apps in Xcode (course developed by Apple).
3. If we want to test an application on a device like an iPhone or an iPad, we hire an iOS developer account which costs $ 99 / year.
4. If we want to publish our application in addition to the developer account mentioned in the previous point, go strictly control our app,

to ensure that it meets the legal restrictions of the App Store and has proper operation.

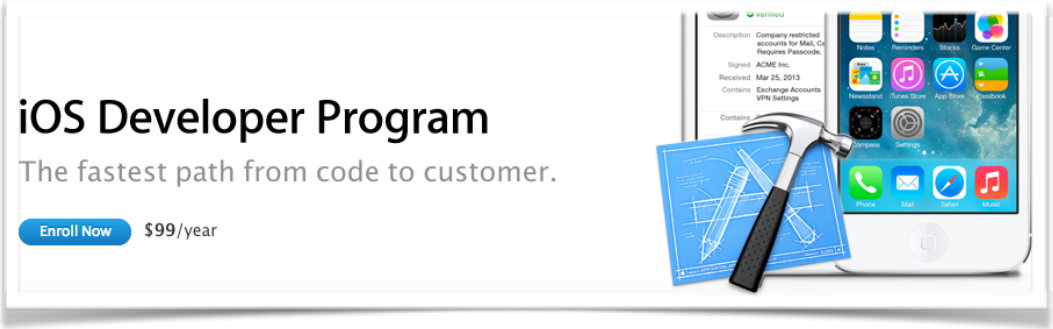


FIGURE 6: Program development for iOS and cost.

Once we have what we need to start programming with Xcode. We can say that Xcode which is free unlike the restrictions above, this is a great advantage for the programmer. Xcode was launched in 2003 and built with Interface Builder is a graphical tool for creating interfaces.

This allows for certain parts of the application in a simple and fast way. Xcode can compile applications in C, C + +, Objective-C, Objective-C + +, Java and AppleScript. External companies have developed supports Pascal, Ada and Perl.

An interesting feature is that Xcode is able to distribute heavy work such as construction, among multiple computers using Bonjour technology.

Xcode also provides a device manager known as Organizer. Using this operator we can connect our iOS device to your Mac to test applications on the terminal.

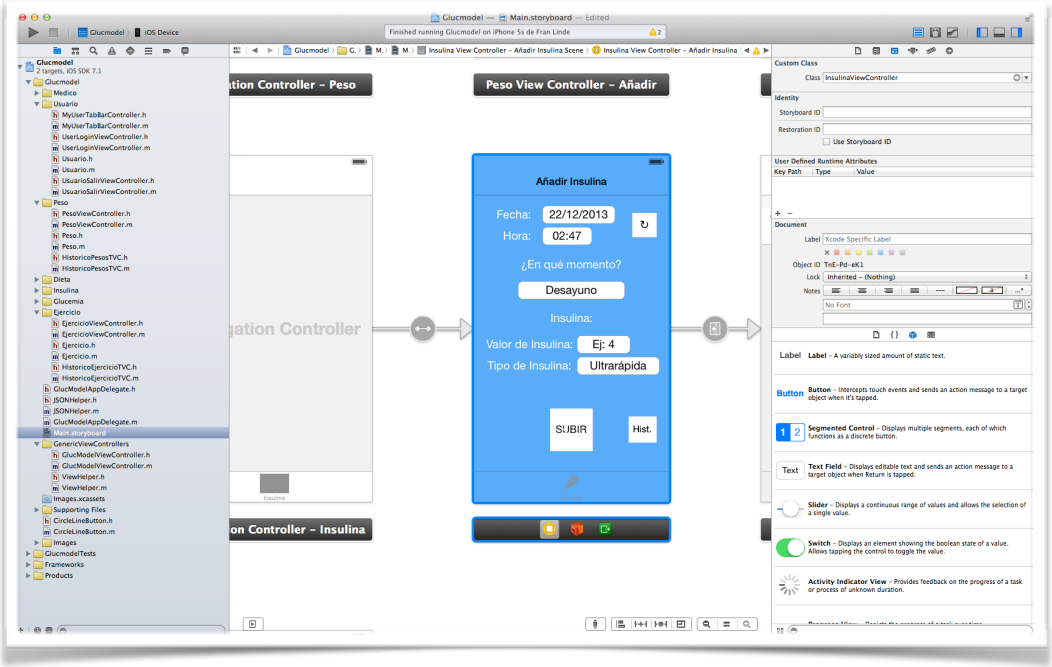


FIGURE 7: Interface made in Xcode with graphical tool. To the right of the picture elements available for creation could be observed.

FIGURE 8: Organizer. Management tool for iOS devices test.

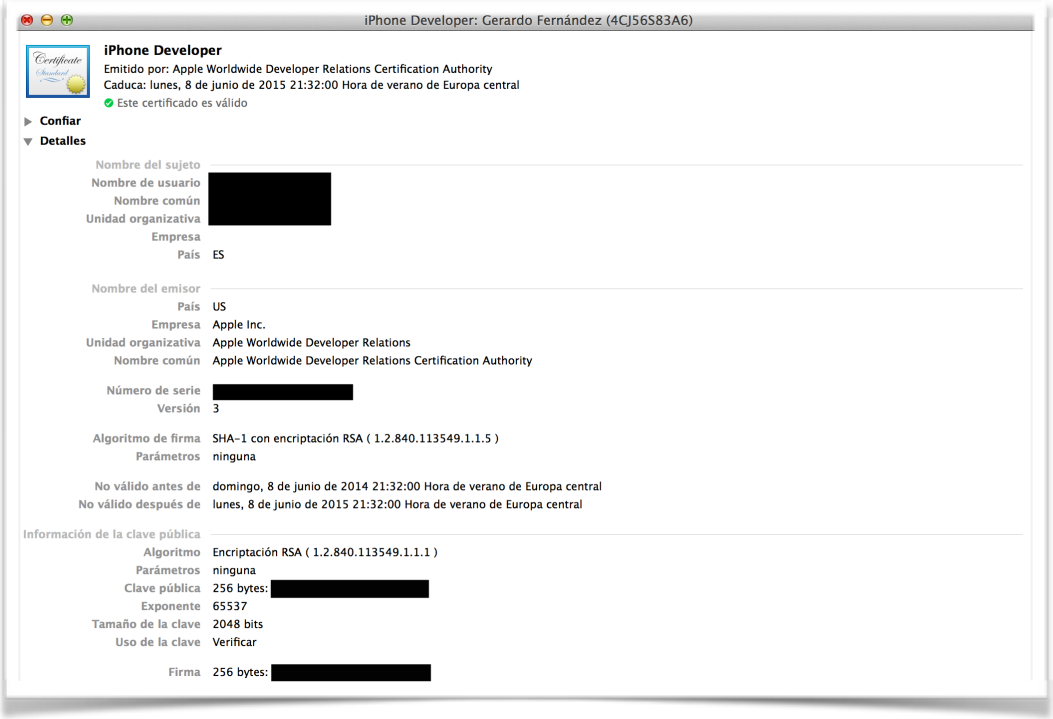


Figure 9: Example of certificate used to sign the application glUCModel. Note that the type of encryption as well as public and private key encryption algorithm is indicated.

IOS applications have a level of security and control too high, so we generate all applications must be signed by the developer. Xcode uses a complex system of digital signatures and certificates. In this section, Apple has tried to simplify the process and help us by Xcode but the truth is that it is an arduous task.

Xcode simulator has a screen where we can test our application if we have a physical terminal or developer account required. This simulator is very fast, just need 3 seconds to turn on and about two seconds to load the application. Besides allowing us to test the app if you do not have the necessary tools, it also offers a great advantage in that it is possible to simulate all iOS devices so iPhone, such as iPad, iPod touch ... as in different generations and with the release of iOS we choose .

I would add that I have had the opportunity to use the Android simulator that Google provides for development, and the ignition timing

simulator Nexus 4 reached 10 minutes is a huge difference with the iOS Simulator and is a point for development in iOS.

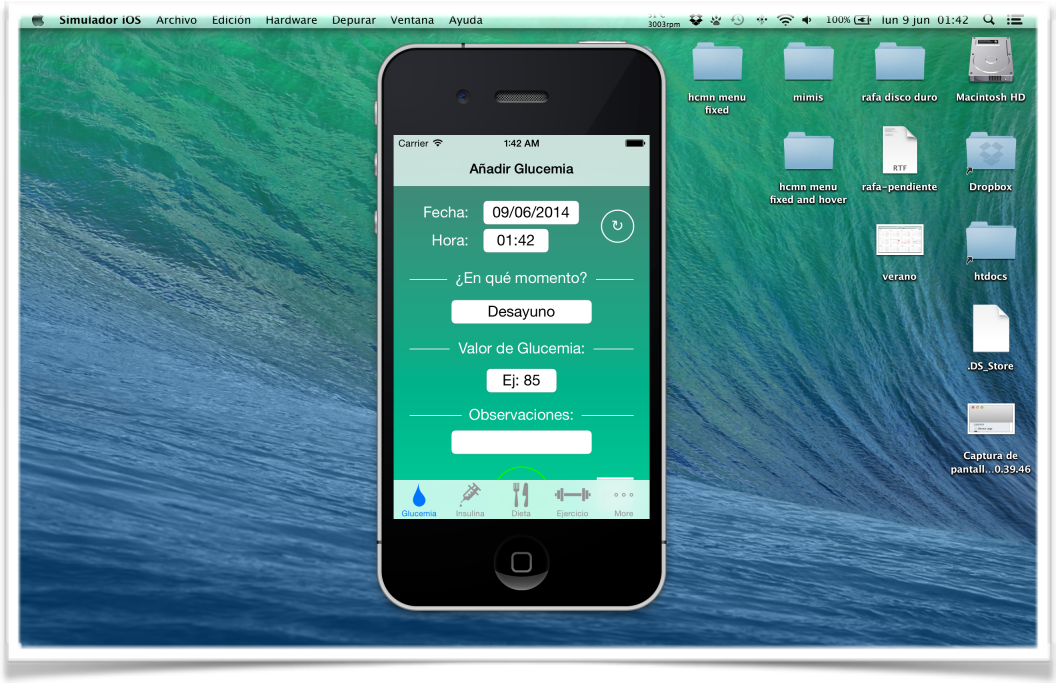


Figure 10: Example of using the simulator iPhone 4s with iOS 7.

The application is available in the page that the Universidad Complutense de Madrid is intended for mobile applications (http://www.ucm.es/apps). Later it will be available in the Apple App Store.

4. IMPLEMENTATION

4.1 - DEVELOPMENT METHODOLOGY

I have followed a development SCRUM type, since they have made partial deliveries verification meetings with the project manager. The vast majority of the requirements are defined early in development meetings, later during development and verifications have been adapting and adding requirements to achieve the desired objective. Throughout the iterations progress the project has made correctly and improved with each.

4.2 - CODE ORGANIZATION

The code has been distributed in packs according to their functionality so that we find the following groups or packages:

1. Doctors: Contains classes for login interfaces in medical, management of patients associated with the physician, medical class, medical logout interface, such interfaces drivers ... etc

2. Username: Contains classes for login user interfaces, class user interface user logout, the drivers of these interfaces, the class that manages the data upload interfaces ... etc

3. Glucose: Contains interfaces and drivers needed for rising glycemia data, and data shown by history. Contains classes that represent glucose and store the data for the same (blood sugar level, date and time .. etc).

4. Weight: Contains interfaces and drivers needed for uploading data and patient weight data shown by history. Contains classes that represent a weight on a certain date and store its data (date, observations and weight in kg).

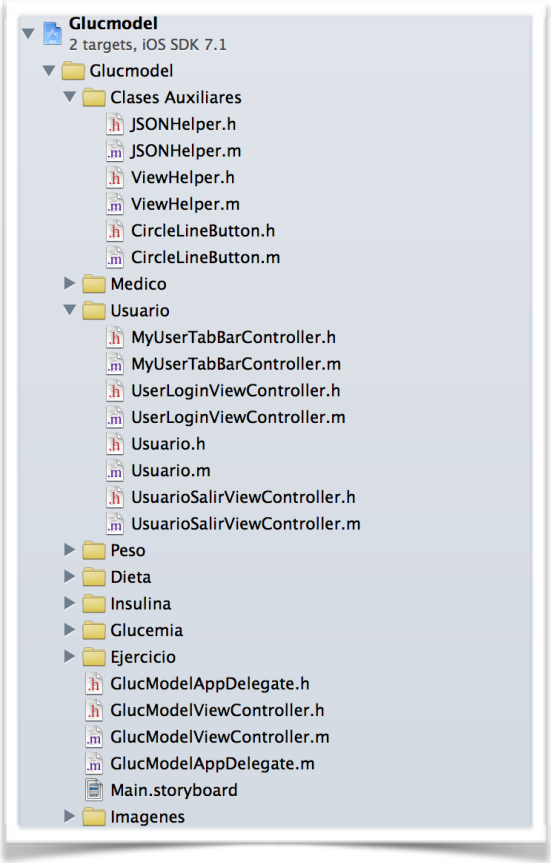
5. Insulin: Contains interfaces and drivers required for uploading data from insulin administration, and data shown by history. Contains classes that represent an event of insulin and store this data (amount of insulin, insulin type, date and time .. etc).

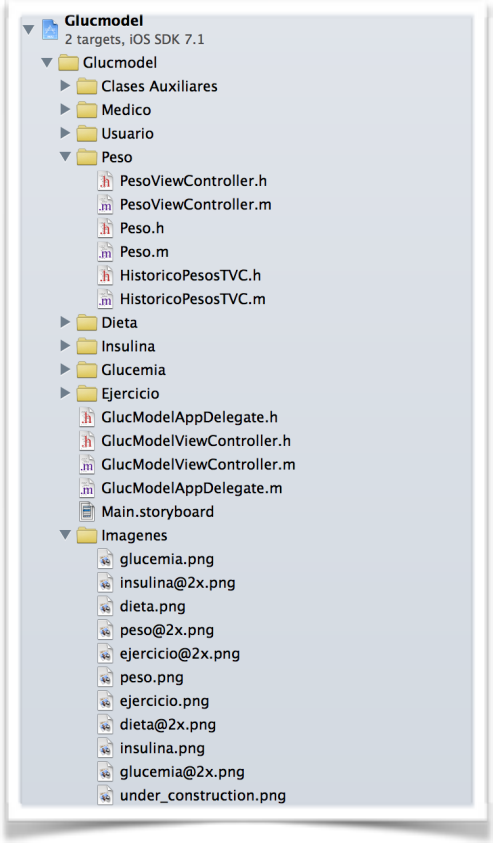
6. Diet: Contains interfaces and drivers needed for data storage diets. Contains classes that represent the previous intakes by history. Diet Contains class that stores data intake (servings of carbohydrates, date and time, type of food, comments .. etc).

7. Exercise: This package stores the classes responsible for displaying interfaces for uploading workout data and history of the exercises. Contains the class that represents and stores a year and the data associated with this (level of physical demand, date, time and observations).

8. Images: In this pack you will find the images used in interfaces, mostly icons.

9. Auxiliary Classes: Stores classes that are used for various interfaces, such as ViewHelper, which is used to draw a line with gradient color; or CircleLineButton which is also used by several interfaces to draw a circular button; or JSONHelper that is used to access the database SQL parsing JSON data format.

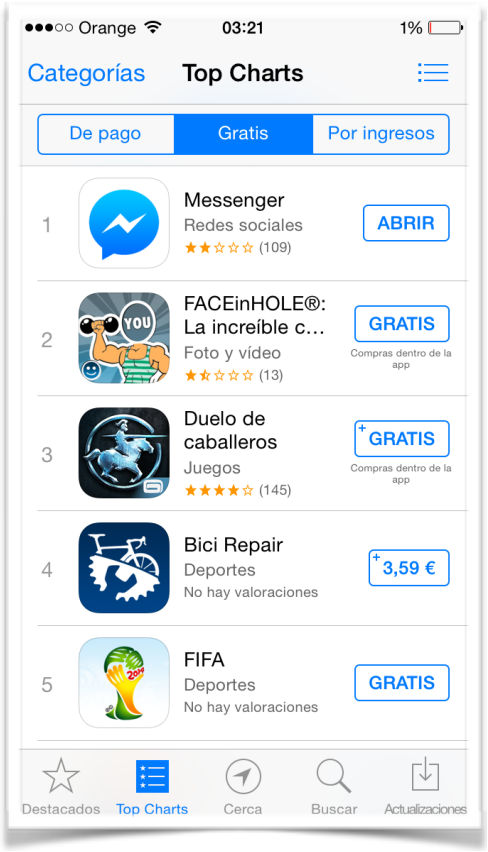




FIGURES 11 and 12: Examples of packages and classes viewed the project with Xcode.

4.3 -INTERFACE DESIGN

The interface design was not included in the requirements, but are included as part of the development work. I chose to make the easiest way to make it intuitive user interface. Also for this purpose standard iOS design showing lower navigation menus that are present in native applications like iTunes and App Store from the first version of iOS is followed.



FIGURES 13 and 14: Comparison of Apple App Store application and glUCModel. You can see the same design on the bottom menu.

It also has sought to make a modern design geared to the latest version of iOS (iOS 7). This can be seen in details like the rounded uploading data that mimics the design of the buttons that were introduced with the iOS button 7.

The bottoms of the screens are broken colors were carefully chosen, after passing through several selection processes relying on outside opinions of users who tested the application. Several of the designs presented tones causing even feeling stress. The colors were replaced by pastel hues ranges, less aggressive. The result is an interface that is gentle to the eye.

FIGURES 15 and 16: Sample number buttons introduced in iOS 7 and UP button shows made in the likeness of these.

4.4 - DEVELOPMENT CHALLENGES

4.4.1 - CONNECTING TO SQL FROM IOS

Definitely the main development challenge meant gaining access to SQL database from Objective-C. After performing various tests with libraries the results were not satisfactory, so the decision to perform'S APIs (Application Programming Interfaces or Application Programming Interfaces) to access the data and make insertion into the database was taken.

These API'S were coded in the programming language PHP as would housed on the same server where the database so glUCModel was that access was faster. PHP has a good integration with SQL using libraries like mysql, mysqli or PDO that are included in PHP. We chose to use mysqli as it is the one with better speed and performance and is object oriented.

4.4.2 - APPLICATION SECURITY

Access to the application must be protected so that there could query data from a patient or add more data to your records if the password is not possessed. So when the application is requesting access credentials of the user (email and password) as shown in Figures 17 and 18 are opened. This happens both from the medical area and from the user area. The password is not displayed as it is introduced to avoid being seen by a third person.

FIGURES 17 and 18: login screens patients and physicians. Passwords are not displayed to prevent them from being discovered by a third party..

4.4.3 - DATABASE SECURITY

The SQL database is protected by username and password, so that this aspect should not be approached. The concern is for the safety of both the developed and API'S connection made with them.

4.4.4 - INTEGRITY OF LOGIN CREDENTIALS

Sensitive data must be protected are certainly of Login. The login request is made by POST and the variable containing the password is encrypted. It is in the IOS application itself where encryption is performed. This encryption is performed by a function of MD5 encryption is the same algorithm used glUCModel web application, unlike the encryption in this case is done before the request and not later, so that security is still higher than in the web application itself.

4.4.5 - PREVENTING SQL INJECTION

Another concern was definitely the possibility that inject SQL code in API requests. To prevent this the API'S are protected to all requests that are received are filtered upon receipt for an escaped special characters that are not alphanumeric and could pose a security hole.

After making these various prevention trials in which they tried to breach the security API for SQL queries closing dump database, insertions or deletions were performed. Not got past security, this showed that it was safe API'S.

4.4.6 - LOCAL STORAGE OF THE DATA

When the user does not have internet data is stored on the device for later insertion into the database. If the user closes the app data will be lost, so it was suggested that the data were not able to be uploaded were stored locally.

iOS supports various types of local storage. The simplest, light and fast is NSUserDefaults. The disadvantage of this system is that it allows to store only native Objective-C types. Our need was for storing objects "insulin" objects "glucose" objects "diet" ... so this method seems that not served to us. The decision to perform a function that converted to the String data type for almacenarnos with NSUserDefaults was noted. When the application starts the data is recovered and converted back to the objects defined by us (Glucose, Insulin, Exercise ... etc).

Figure 19: A screen that informs the user that their data has not been uploaded. They will be stored locally.

4.5 - MAINTAINABILITY OF THE APP

For the application to be developed later perhaps by another developer or team of developers, the code has been properly discussed with explanations that clearly indicate the purpose of each piece of code as shown in Figure 19.

```
- (IBAction)uploadPushed:(id)sender {

    //Quito el boton muestro el indicador y lo muevo
    [self.uploadButton setHidden:true];
    [self.indicator setHidden:false];
    [self.indicator startAnimating];

    //retraso un segundo la ejecucion de la subida
    //para que el usuario vea la ruleta y sepa que se está subiendo el dato
    int64_t delayInSeconds = 1;
    dispatch_time_t popTime = dispatch_time(DISPATCH_TIME_NOW, delayInSeconds * NSEC_PER_SEC);
    dispatch_after(popTime, dispatch_get_main_queue(), ^(void){

        //convierto la fecha en un string de mysql
        NSDateFormatter *gmtDateFormatter = [[NSDateFormatter alloc] init];
        gmtDateFormatter.timeZone = [NSTimeZone timeZoneWithName:@"Europe/Spain"];
        gmtDateFormatter.dateFormat = @"yyyy-MM-dd HH:mm:ss";
        NSString *fechaPeso = [gmtDateFormatter stringFromDate:self.datePicker.date];

        //meto los datos en un objecto Peso
        Peso *nuevoPeso = [[Peso alloc]init];
        nuevoPeso.fechaPeso = fechaPeso;
        nuevoPeso.valorPeso = [self.pesoTextField text];

        //Llamo al singleton de usuario para usar el id
        Usuario *usuarioSingleton = [Usuario usuarioSingleton];

        //Llamo a mi JSONHelper para que suba los datos
        [JSONHelper uploadPeso:(usuarioSingleton.userID) :(nuevoPeso)];
        //Paro el indicador, lo escondo y muestro el boton
        [self.indicator stopAnimating];
        [self.indicator setHidden:true];
        [self.uploadButton setHidden:false];
    });
}
```

FIGURE 20: Extract application code which shows that the code contains explanatory comments.

The development of the interface is made almost entirely with the graphical tool, because if developed code would be of a very high trying to understand and modify the code troubles. When the GUI is used easy to understand results as shown in Figure 20 are obtained.

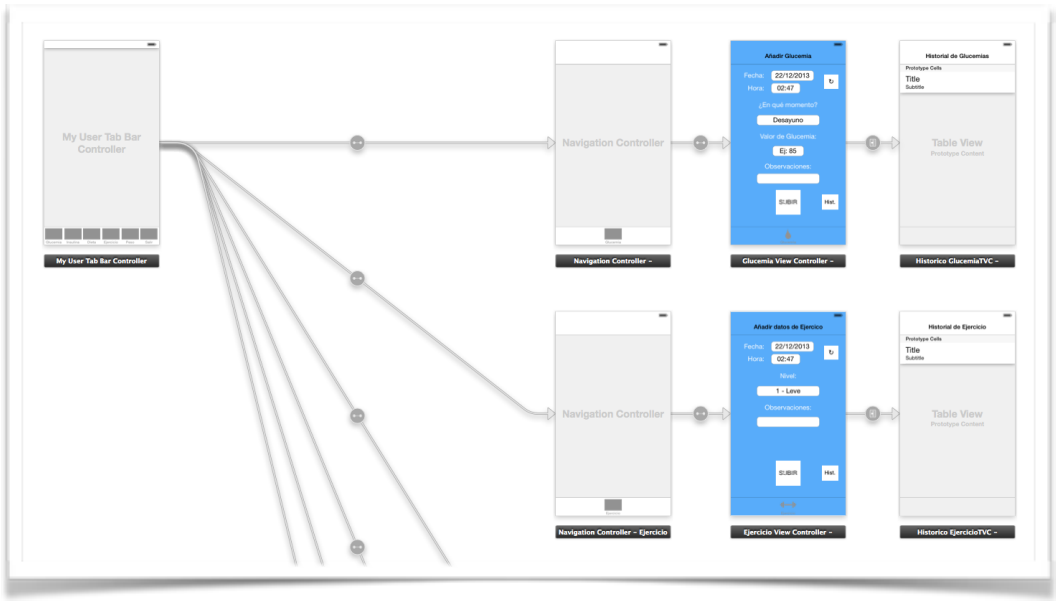


Figure 21: Screenshot of the file that generates the bottom menu and windows.

4.6 - USER'S HEALTH

It was a built-in safety mechanism that the application informs the user if their blood sugar levels pose a risk to your health.

When a high measurement values is considered hyperglycemia, or when you enter a value that is as low in hypo area is entered, the application confirms this fact, and if the user indicates success it is recommended a plan action.

The values considered hazardous are described by endocrine physicians involved in the project glUCModel, hyperglycemic values deteriorate the patient's health in the long term, and hypoglycemic values can cause unconsciousness the patient. Figures 22 and 23 contain screenshots of the messages that are displayed to the user.

FIGURES 22 and 23: warning messages in case of high glycemia.

5. RESULTADO

The result has been satisfactory development. We are faced with an application that meets the requirements set out at the beginning and during development.

It is a lightweight, secure and fast application.

After several studies of performance, it has been found that initially occupies about 20MB in RAM and after intense use of this application in RAM occupies around 50MB, the use to which the processor never exceeds 1% since the application does not need to perform calculations. Figure 24 shows the use of data RAM and the processor after intense use.

He has also studied the disk space occupied by the application once it is compiled. It has been observed that the application fails to occupy disk 1MB. If we have the data that is stored locally, you can occupy 1.5MB or 2MB as shown in Figure 25.

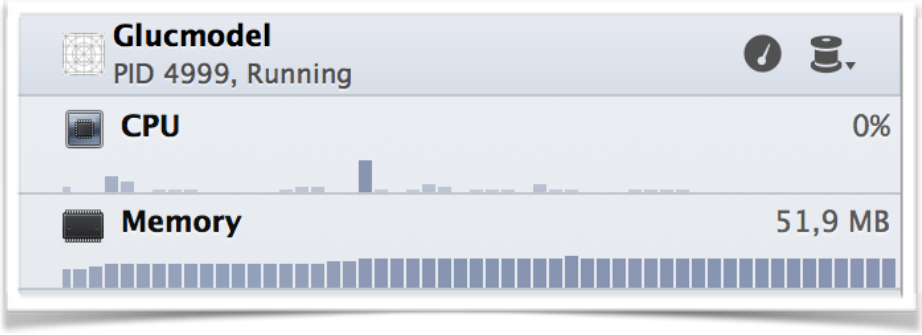


Figure 24: Resource Monitor iPhone while running glUCModel

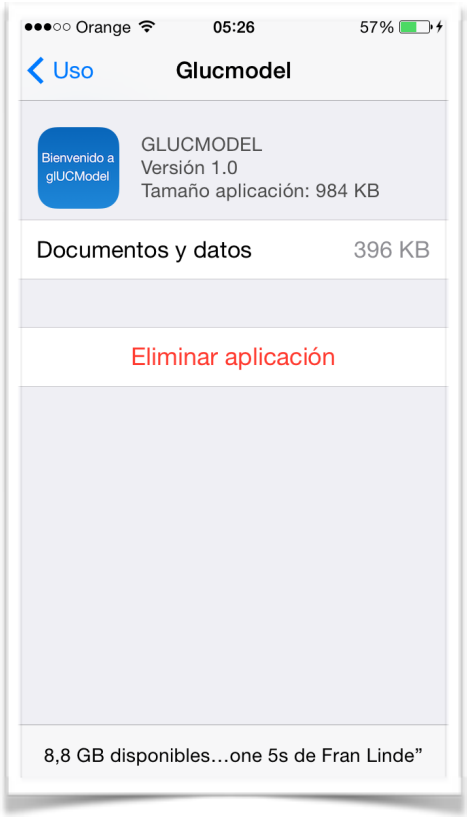


Figure 25: Screenshot, glUCModel disk space.

Regarding the interface results are my good. Usability tests have shown that the interface is friendly and intuitive. These opinions about the user experience is good.

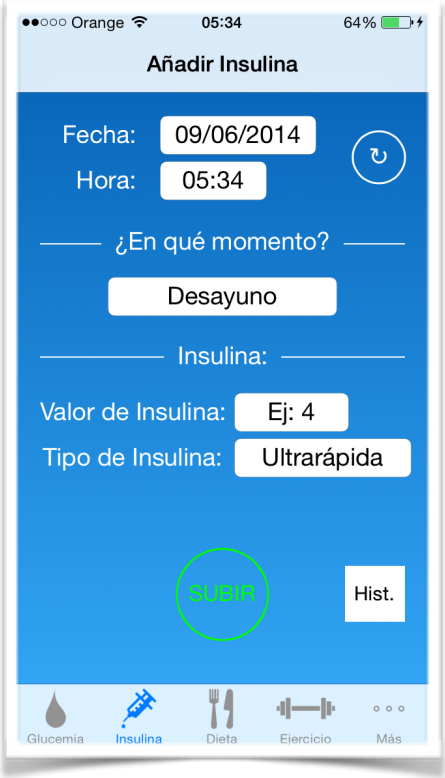


Figure 26: Inserting insulins.

Figure 27: History of Insulin

Figure 28: History of hyperglycemia.

Figure 29: Inserting data of Weight.

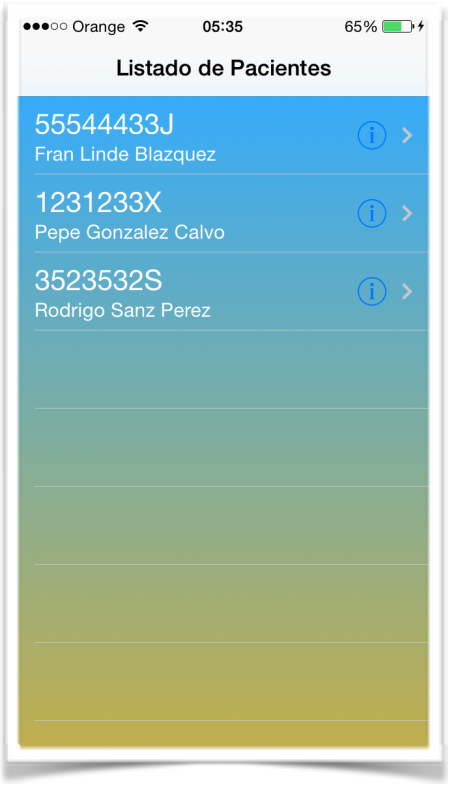


Figure 30: Patients of a doctor.

6. CONCLUSIONS

The development of the project has successfully completed and collaboration with glUCModel has been beneficial for both parties. On one side glUCModel now has a native iOS application that meets the requirements needed. Furthermore as a developer, I have acquired knowledge of programming in Objective-C which brings great value to my resume and my potential as autonomous.

I have faced problems during development that have learned to solve correctly, the proof is that the application has gone ahead and working properly.

Unfortunately the development has been largely influenced by deadlines and the university calendar, having had full time availability and scheduling, implementation would enjoy more features.

7. FUTURO DE LA APLICACIÓN

The application has great potential since the development has been carried out by only one person. By providing more human resources could include more functionality to the application. Detail a list of possible improvements:

- Add other languages.
- Glucose prediction system based on the prediction algorithm that is developing.
- Recommendation system insulin dose based on the prediction algorithm that is developing.
- Direct communication with doctor-patient by app.
- Periodically checking the internet for uploading the outstanding data automatically.
- Improved specific design of the interface for different terminals (iPhone 4, iPhone 5, iPad, iPad retina, iPad mini ... etc)
- Code optimization for higher speed.

8. REFERENCES

1 - J.I. Hidalgo, et al., Modeling glycemia in humans by means of Grammatical Evolution, Appl. Soft Comput. J. (2013), http://dx.doi.org/10.1016/j.asoc.2013.11.006

2 - Standford University Online Course: Developing iOS 7 Apps for iPhone and iPad, http://online.stanford.edu/course/developing-ios7-apps-fall-2013

3 - J.I. Hidalgo, et al. glUCModel: A monitoring and modeling system for chronic diseases applied to diabetes. J Biomed Inform (2014), http://dx.doi.org/10.1016/j.jbi.2013.12.015

4 - A. American-Diabetes-Association. Standards of medical care in diabetes 2010. Diabetes Care, 33(S1):11–61, 2010.

5 - Apple iOS Developer Library: Programming with Objective-C, https://developer.apple.com/library/ios/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/ProgrammingWithObjectiveC.pdf

6 - Apple iOS Developer Library: iOS 7 Design Resources, iOS Human Interface Guidelines, https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/MobileHIG.pdf

7 - Hidalgo JI et al. glUCModel: Clarke and Parkes Error Grid Analysis of Diabetic Glucose Models obtained with Evolutionary Computation

8 - iOS Developer Library: iOS App Programming Guide, https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/iPhoneAppProgrammingGuide.pdf

9 - Zach Waugh, a Mac & iOS developer: How to use JSON in Cocoa/Objective-C, http://zachwaugh.me/posts/how-to-use-json-in-cocoaobjective-c/

10 - Andrey Prikaznov: Create Your Own XML/JSON/HTML API with PHP, http://css.dzone.com/articles/create-your-own-xmljsonhtml

11 - Manual de PHP, Seguridad Seguridad de Bases de Datos: Inyección de SQL, http://www.php.net//manual/es/security.database.sql-injection.php

12 - iOS Developer Tips: Create MD5 Hash from NSString, NSData or a File, http://iosdevelopertips.com/core-services/create-md5-hash-from-nsstring-nsdata-or-file.html

13 - Ray Wenderlich, How To Write A Simple PHP/MySQL Web Service for an iOS App, http://www.raywenderlich.com/2941/how-to-write-a-simple-phpmysql-web-service-for-an-ios-app

14 - Tristan Louis, How Much Do Average Apps Make?, http://www.forbes.com/sites/tristanlouis/2013/08/10/how-much-do-average-apps-make/