UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

UBUDiabetes 2.0

Presentado por Luis Miguel Inapanta Oyana
en Universidad de Burgos — 27 de junio de 2018
Tutor: Dr. Raúl Marticorena Sánchez

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

Dr. Raúl Marticorena Sánchez, profesor del departamento de Ingeniería Civil, área de Lenguajes y Sistemas Informáticos.

Expone:

Que el alumno D. Luis Miguel Inapanta Oyana, con DNI 71709946-V, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado UBUDiabetes 2.0.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 27 de junio de 2018

Vº. Bº. del Tutor:

Dr. Raúl Marticorena Sánchez

Resumen

El proyecto tienen como objetivo la revisión de la aplicación Android UBUDiabetes 1.1 para el control de la diabetes, incluyendo los nuevos requisitos que surgen de trabajos de fin de grado de validación de la herramienta previa en el Grado en Enfermería.

Por otro lado se pretende mejorar la interfaz gráfica de usuario con el fin de aumentar la interactividad entre usuario y aplicación, proporcionar un entorno más atractivo e intuitivo posible, facilitar el aprendizaje de uso de la aplicación para el usuario, etc., ya que esta se considera como el principal punto de contacto entre usuario y aplicación.

Por último, se requiere la inclusión de la automatización de las pruebas, tanto unitarias, integración, sistema, etc. con el fin de aumentar la calidad y robustez de la aplicación.

Descriptores

Aplicación móvil, Control de la diabetes, Calculador de bolo de insulina, Aplicación para Android, Diabetes tipo I, App sanitaria.

Abstract

The aim of the project is to review the Android UBUDiabetes 1.1 application for the control of diabetes, including the new requirements arising from end-of-degree validation of the previous tool in the nursing degree.

On the other hand, it is intended to improve the graphical user interface in order to increase the interactivity between user and application, provide a more attractive and intuitive environment possible, improve the ease of learning of use of the application for the user, etc., and that this is considered as the main point of contact between user and application.

Finally, requires the inclusion of the automation of the tests, both unitary, integration, system, etc. in order to increase the quality and robustness of the application.

Keywords

Mobile application, Diabetes control, Insulin bolus calculator, Application for Android, Diabetes type I, Sanitary app.

Índice general

Índice de figuras

Índice de tablas

Introducción

Actualmente podemos encontrar más de 1500 apps relacionadas con la diabetes en plataformas como Google Play, iTunes, Windows Market o BlackBery World. La mayoría se clasifican como aplicaciones de seguimiento de la enfermedad (33 %), ya que permiten el control de los registros de la insulina, glucemia, peso, condición física y conteo de carbohidratos. En un porcentaje menor (22 %), encontramos las apps educativas, que facilitan el recuento de carbohidratos mediante juegos o el cálculo de insulina a través de la glucemia. También existen apps nutricionales, centradas en el cálculo de carbohidratos en las comidas (8 %). Otras, en su mayoría pertenecientes a farmacéuticas, prestan información sobre sus productos (8 %). Y, por último, también se han desarrollado redes sociales y foros que ayudan a compartir experiencias entre las personas diabéticas (5 %).

UBUDiabetes 1.0 es una aplicación móvil dirigida a persona que padecen diabetes de tipo 1, que se desarrolló de la mano de Mario López Jiménez, autor del TFG "UBUDiabetes: Aplicación de control de diabetes en dispositivos móviles", marcado por las ideas surgidas de Lara Bartolomé Casado, autora del TFG "Salud electrónica y Diabetes: Elementos para el diseño de una aplicación informática (App) para jóvenes diabéticos de tipo 1".

Tras finalizar una primera versión de la aplicación, Raúl Marticorena Sánchez introdujo algunas modificaciones posteriores, las cuales llevaron a una nueva versión de la aplicación: UBUDiabetes 1.1

Desde el Grado de Enfermería, Bruno Martín Gómez, autor del TFG "Evaluación de la eficacia, efectividad y usabilidad de la aplicación para móviles UBUDiabetes", durante el curso 2016-2017 vio la necesidad de validar la eficacia y efectividad de la aplicación móvil UBUDiabetes 1.0 y posteriormente UBUDiabetes 1.1 abordando los siguientes objetivos[2]:

- Validar la eficacia `in vitro` de las funciones utilizadas en los algoritmos y comprobar los cálculos de hidratos de carbono y del bolo de insulina se realizan de forma correcta.
- Validar la efectividad in vivo de UBUdiabetes 1.1 comparando los datos obtenidos por los usuarios con la app y los que finalmente se han dosificado por su prescripción.
- Valorar el grado de satisfacción de los usuarios con la app y recoger sugerencias de mejora.

Tras una primera la validación de UBUDiabetes 1.1, Bruno Martín llegó a las siguientes conclusiones[2]:

- La eficiencia ha resultado satisfactoria en cuanto la mitad de los usuarios han obtenido resultados acertados respecto a la recomendación del bolo de insulina, y en la otra mitad, los cálculos no han sido tan precisos debido a que los usuarios no tenían la suficiente preparación para el recuento de alimentos. Este factor externo a la app implica que la versión actual de UBUDiabetes 1.1 debe ser recomendada a los usuarios adecuados, con un nivel apropiado de conteo de alimentos.
- Los usuarios han quedado satisfechos con la participación en el proyecto. Coinciden en que es sencilla, útil y de fácil manejo.
- UBUdiabetes 1.1 tiene que evolucionar, para alcanzar un mayor nivel de interacción con los usuarios a la hora del conteo de los alimentos y la visualización de sus registros diarios.

UBUDiabetes 2.0 se centra en la revisión y modificación de la versión 1.1, incluyendo nuevos requisitos surgidos por los compañeros de enfermería, así como en la inclusión de la automatización de las distintas pruebas: unitarias, integración, sistema, etc.

Objetivos del proyecto

Para marcar los objetivos, tanto generales como los de la aplicación, analizamos las propuestas y conclusiones marcadas por Mario López Jiménez y Bruno Martím Gómez en sus respectivos proyectos.

- **UBUDiabetes: Aplicación de control de diabetes en dispositivos móviles**[3].
  - Añadir la posibilidad de exportar los datos almacenados en la base de datos SQLite de la aplicación en algún formato que pueda resultar útil para el usuario.
  - Implementar la opción de borrar o modificar entradas de la base de datos.
  - Adaptar la aplicación para *tablets* aprovechando el diseño de la misma en *fragments*.
  - Añadir la opción de crear diferentes cuentas de usuario. Es una opción que en un teléfono no resulta especialmente útil, pero si podría serlo en caso de migrar la aplicación a `tablets` que puedan ser utilizadas por más de un usuario.
  - La biblioteca *MPAndroidChart* está en continua evolución. Por lo tanto actualizar la versión de la misma y, en caso de que tuviera funcionalidades nuevas útiles, aprovecharlas en la aplicación.
  - Ampliar la base de datos de alimentos y en caso de ser necesario si el numero es muy elevado, adaptar el modo en que estos se seleccionan por pantalla a las necesidades.
  - Añadir más idiomas a la aplicación duplicando para ello los archivos *xml* en los que se encuentran los *Strings* de la misma.

- **Evaluación de la eficacia, efectividad y usabilidad de la aplicación para móviles UBUdiabetes**[2].
  - La posibilidad de mostrar los alimentos que se van introduciendo para el cálculo de hidratos de carbono en una comida.
  - Ampliar la lista de alimentos disponibles. Y mostrar la relación de hidratos de carbono.
  - Dar la posibilidad al usuario de elegir las unidades en las que quieren hacer el recuento (raciones, gramos o directamente los hidratos de carbono que los usuarios sacamos de las etiquetas de los productos que consumen).
  - Incluir en la configuración del perfil del usuario bloques de tiempo para la configuración de distintos valores del Ratio.
  - Añadir decimales a las unidades de insulina, ya que usuarios con mucha sensibilidad a la insulina, media unidad puede ser determinante.
  - Tener en cuenta la insulina residual que hay en el organismo.

Una vez analizadas las propuestas y/o conclusiones anteriores, procedemos a marcar los objetivos principales en este proyecto.

2.1. Objetivos Generales

- Estudio y análisis del lenguaje y herramientas necesarios para el desarrollo de una nueva versión de la aplicación UBUDiabetes 1.1.
- Investigar herramientas para el desarrollo de prototipado.
- Estudiar las opciones que ofrece Android para la automatizacion de las distintas pruebas.

2.2. Objetivos de la aplicación

- Mejorar toda la interfaz gráfica de usuario.
- Añadir decimales a las unidades de insulina.
- Mostrar los alimentos que se van introduciendo para el cálculo de hidratos de carbono en una comida.

- Posibilidad de eliminar los alimentos introducidos para el cálculo de hidratos de carbono en una comida.
- Mostrar registros diarios del usuario.
- Ampliar la lista de alimentos disponibles.
- Mejorar la gráfica del historial de glucemias registradas.
- Incluir Pruebas Unitarias.
- Incluir pruebas Instrumentales-Espresso.

Conceptos teóricos

En este apartado explicaremos aquellos conceptos que consideramos necesarios para la correcta comprensión de este proyecto. Así mismo, al tratarse de una nueva versión, es decir, de la actualización y/o modificación de una versión anterior, se ha decidido incluir únicamente aquellos conceptos en los que la versión anterior de la aplicación daba problemas o no se obtuvieron resultados correctos, tanto orientados a la aplicación como a la Diabetes tipo I. El resto de los conceptos comunes a la versión anterior pueden consultarse en las memorias de los proyectos de Mario López Jiménez [3] y Lara Bartolomé Casado [1].

3.1. Diabetes

Al tratarse de una app dirigida a personas que padecen diabetes tipo 1 vamos a presentar un breve resumen sobre el concepto de Diabetes.

La diabetes es una enfermedad en la que los niveles de glucosa (azúcar) de la sangre están muy altos. La glucosa proviene de los alimentos que consume. La insulina es una hormona que ayuda a que la glucosa entre a las células para suministrarles energía. En la diabetes tipo 1, el cuerpo no produce insulina. En la diabetes tipo 2, la más común, el cuerpo no produce o no usa la insulina de manera adecuada. Sin suficiente insulina, la glucosa permanece en la sangre [5].

A continuación se explicarán aquellos conceptos teóricos en los que la versión anterior de la aplicación (UBUDiabetes 1.1) presentaba ciertos fallos. Para ello, tendremos en cuenta algunas de las recomendaciones de Bruno Martín Gómez en su proyecto [2].

Hidratos de Carbono

Los hidratos de carbono o carbohidratos se definen como biomoléculas compuestas por carbono, hidrogeno y oxígeno, cuyas principales funciones en los seres vivos son el brindar energía inmediata y estructural [20]. El **recuento de carbohidratos** es esencial para el cálculo de bolo corrector. ya que ayuda al usuario a controlar la glucosa en la sangre. Si se consigue el equilibrio adecuado entre carbohidratos ingeridos e insulina, el nivel de glucosa en la sangre se mantendrá dentro de los niveles deseados [1]. Para el cálculo de Grs de HC (Hidratos de Carbono) el usuario deberá:

- Buscar y seleccionar el/los alimento/s que va a ingerir.
- Introducir la cantidad en gramos de dicho/s alimento/s.
- Finalmente, la app deberá realizar el recuento de los gramos de HC en la suma de los distintos alimentos a partir de la tabla de la federación española de la diabetes [9].

Sensibilidad a la insulina

En la Diabetes, la sensibilidad a la insulina se entiende por el valor de glucemia en mg/dl que se consigue reducir al administrar una unidad de análogo de insulina de acción rápida. Así mismo, se debe tener en cuenta que existen usuarios con **resistencia a la insulina**, es decir, que presentan una condición en la cual los tejidos presentan una respuesta disminuida para disponer de la glucosa circulante ante la acción de la insulina; en especial el hígado, el músculo esquelético, el tejido adiposo y el cerebro. Esta alteración en conjunto con la deficiencia de producción de insulina por el páncreas puede conducir después de algún tiempo al desarrollo de una *diabetes mellitus* tipo 2 [24]. Por el contrario, existen usuarios con una mayor sensibilidad a la insulina. Es por esto por lo que Bruno Martín Gómez señala en su TFG [2] en el apartado “Recomendaciones para la versión UBUDiabetes 2.0.”, que se deberían tener en cuenta los decimales en las unidades de insulina, ya que usuarios con mucha sensibilidad a la insulina, media unidad puede ser determinante.

Insulina residual

Toda insulina tiene un **inicio de efecto** que es el periodo desde la inyección de la misma hasta que empieza a funcionar. Un **máximo efecto** o pico que es el periodo donde existe más efecto insulínico, debe coincidir

con la máxima concentración de hidratos de carbono en el organismo, y un **fin de efecto** que es la ***insulina activa residual*** tras el fin del pico de acción.

Indice glucémico

El índice glucémico (glycemic index o GI) mide en qué medida los alimentos que contienen carbohidratos elevan la glucosa en la sangre. Los alimentos se clasifican en base a cómo se comparan a alimentos de referencia, ya sea glucosa o pan blanco. Un alimento con un GI alto eleva la glucosa en la sangre más rápido que los alimentos con un GI mediano o bajo [7].

¿Qué afecta el GI de un alimento?

La grasa y fibra tienden a reducir el GI de un alimento. Como regla general, mientras más cocido o elaborado un alimento, más alto su GI. Sin embargo, esta regla no siempre se aplica.

- Madurez y tiempo almacenado — Mientras más madura la fruta o vegetal, más alto su GI.
- Elaboración — El jugo tiene un GI más alto que toda la fruta; el puré de papas tiene un GI más alto que una papa entera al horno, el pan de trigo integral molido con piedra tiene un GI más bajo que el pan de trigo integral.
- Método de preparación — El tiempo que se han cocinado los alimentos (los fideos al dente tienen un GI más bajo que los bien cocidos).
- Variedad — El arroz blanco instantáneo de grano largo tiene un GI más bajo que el arroz integral pero el arroz blanco de grano corto tiene un GI más alto que el arroz integral.

3.2. Aplicación Móvil

Para comprender como funciona la aplicación UbuDiabetes2.0 primero debemos explicar en que consiste una *Aplicación móvil o App*. El resto de los conceptos comunes a la versión anterior pueden consultarse en las memorias de los proyectos de Mario López Jiménez [3] y Lara Bartolomé Casado [1].

Una app es una aplicación informática diseñada para ser ejecutada en teléfonos inteligentes, tabletas y otros dispositivos móviles. Las aplicaciones permiten al usuario efectuar un conjunto de tareas de cualquier tipo:

profesional, de ocio, educativas, de acceso a servicios, etc., facilitando las gestiones o actividades a desarrollar [17].

Testing en aplicaciones móviles

Probar una aplicación es una parte integral del proceso de desarrollo de la aplicación. Al ejecutar las pruebas en la aplicación consistentemente, se puede verificar la corrección, el funcionamiento funcional y la usabilidad de la aplicación antes de lanzarla públicamente. Las pruebas ofrecen las siguientes ventajas [4]

- Rápida retroalimentación sobre fallos.
- Detección temprana de fallos en el ciclo de desarrollo.
- Refactorización de código más segura, lo que le permite optimizar el código sin preocuparse por las regresiones.
- Velocidad de desarrollo estable, lo que ayuda a minimizar la deuda técnica.

Fundamentos del Testing

Los usuarios interactúan con su aplicación en una variedad de niveles, desde presionar un botón "Enviar" hasta descargar información en su dispositivo. En consecuencia, debe probar una variedad de casos de uso e interacciones a medida que desarrolla iterativamente su aplicación. A medida que una aplicación se expande, puede que sea necesario buscar datos de un servidor, interactuar con los sensores del dispositivo, acceder al almacenamiento local o representar interfaces de usuario complejas. La versatilidad de una aplicación exige una estrategia de prueba integral. Al desarrollar una característica de forma iterativa, comienza escribiendo una nueva prueba o agregando casos y aserciones a una prueba de unidad existente. La prueba falla al principio porque la característica aún no está implementada. Es importante considerar las unidades de responsabilidad que surgen a medida que diseña la nueva característica. Para cada unidad, escribe una prueba de unidad correspondiente. Las pruebas de su unidad casi agotan todas las interacciones posibles con la unidad, incluidas las interacciones estándar, las entradas no válidas y los casos en que los recursos no están disponibles.

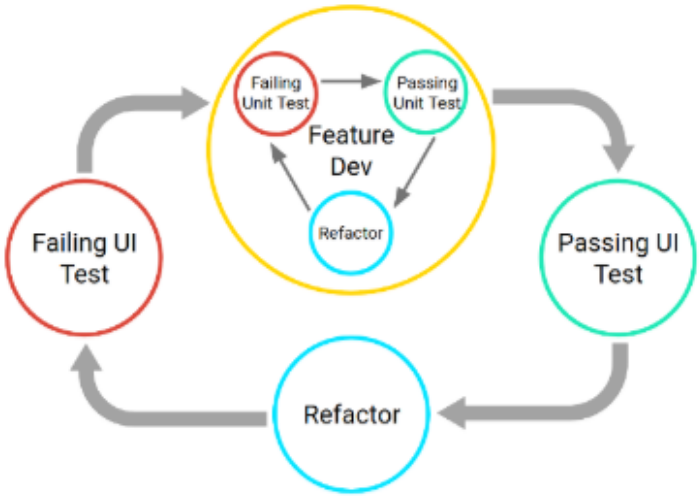

Figura 3.1: Flujo de trabajo-Testing

El flujo de trabajo completo, como se muestra en la Figura 3.1, contiene una serie de ciclos iterativos anidados en los que un ciclo largo, lento y controlado por la interfaz de usuario prueba la integración de las unidades de código. El programador prueba las unidades con ciclos de desarrollo más cortos y más rápidos. Este conjunto de ciclos continúa hasta que la aplicación satisfaga todos los casos de uso.

Interacción Hombre-Máquina

Esta disciplina se enfoca en el estudio de la interacción entre las personas y los sistemas computacionales. Su objetivo principal es mejorar esta interacción, haciendo que los sistemas computacionales sean más usables, de manera que aumente la productividad de las personas al trabajar con ellos

Principios del diseño

Cuando evaluamos una interfaz, o diseñamos una nueva, se tienen que tener en cuenta los siguientes principios de diseño experimental [21]:

- **Fijar quien será el usuario/s y su/s tarea/s**. Se tiene que establecer el número de usuarios necesarios para llevar a cabo las tareas y determinar cuáles serían las personas indicadas. Una persona que nunca lo ha utilizado y no la utilizará en el futuro, no sería un usuario válido.

- **Medidas empíricas**. Sería de gran utilidad llevar a cabo un testeo de la interfaz con usuarios reales, en la situación en que se utilizaría. No podemos olvidar que los resultados se verán alterados si la situación no es real. Habría que establecer una serie de especificaciones cuantitativas, que serán de gran utilidad, como podrían ser el número de usuarios necesarios para realizar una tarea, el tiempo necesario para completarla y el número de errores que se producen durante su realización.
- **Diseño iterativo**. Una vez determinados los usuarios, las tareas y las medidas empíricas se vuelve a empezar: se modifica el diseño, se testea, se analizan los resultados y se repite de nuevo el proceso hasta obtener la interfaz deseada.

UBUDiabetes 2.0

UbuDiabetes es una aplicación dirigida a diabéticos tipo 1 cuya función principal es ayudarlos a calcular con eficacia el bolo corrector de insulina, disminuyendo el riesgo de hipoglucemia. Otras funcionalidades que presenta UBUDiabetes son :

- **Historial de Glucemias**: Gráfico con los valores de las glucemias registradas por el usuario. Muestra los valores de la última semana para las glucemias registradas Antes de desayunar, Después de comer y Antes de cenar.
- **Registro de Glucemias**: Guarda en la base de datos local de la aplicación los valores de las glucemias registradas por el usuario.
- **Consultar Ingestas**: Muestra los registros de todos los cálculos del bolo corrector que se han llevado a cabo por el usuario.

Técnicas y herramientas

A continuación, a pesar de que se trata de una nueva versión, es decir, modificación y/o mejora de una versión previa, incluiremos todas las técnicas y herramientas utilizadas para el desarrollo de este proyecto.

4.1. Android

Android es un sistema operativo basado en GNU/Linux, un núcleo de sistema operativo libre, multiplataforma y gratuito. Inicialmente Android se creó para teléfono móviles. Actualmente lo podemos encontrar en múltiples dispositivos móviles como tablets, relojes inteligentes, televisores, automóviles...

La estructura del sistema operativo Android se compone de aplicaciones que se ejecutan en un framework Java de aplicaciones orientadas a objetos sobre el núcleo de las bibliotecas de Java en una máquina virtual Dalvik con compilación en tiempo de ejecución hasta la versión 5.0, luego cambió al entorno Android Runtime (ART). Las bibliotecas escritas en lenguaje C incluyen un administrador de interfaz gráfica (surface manager), un framework OpenCore, una base de datos relacional SQLite, una Interfaz de programación de API gráfica OpenGL ES 2.0 3D, un motor de renderizado WebKit, un motor gráfico SGL, SSL y una biblioteca estándar de C Bionic. El sistema operativo está compuesto por 12 millones de líneas de código, incluyendo 3 millones de líneas de XML, 2.8 millones de líneas de lenguaje C, 2.1 millones de líneas de Java y 1.75 millones de líneas de C++ [15].

Características

Características y especificaciones en la actualidad [15].

Arquitectura

- **Aplicaciones**: las aplicaciones base incluyen un cliente de correo electrónico, programa de SMS, calendario, mapas, navegador, contactos y otros. Todas las aplicaciones están escritas en lenguaje de programación Java.
- **Marco de trabajo de aplicaciones**: los desarrolladores tienen acceso completo a los mismas API del entorno de trabajo usados por las aplicaciones base. La arquitectura está diseñada para simplificar la reutilización de componentes; cualquier aplicación puede publicar sus capacidades y cualquier otra aplicación puede luego hacer uso de esas capacidades (sujeto a reglas de seguridad del framework). Este mismo mecanismo permite que los componentes sean reemplazados por el usuario.
- **Bibliotecas**: Android incluye un conjunto de bibliotecas de C/C++ usadas por varios componentes del sistema. Estas características se exponen a los desarrolladores a través del marco de trabajo de aplicaciones de Android. Algunas son: System C library (implementación biblioteca C estándar), bibliotecas de medios, bibliotecas de gráficos, 3D y SQLite, entre otras.
- **Runtime de Android**: Android incluye un set de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje Java. Cada aplicación Android corre su propio proceso, con su propia instancia de la máquina virtual Dalvik. Dalvik ha sido escrito de forma que un dispositivo puede correr múltiples máquinas virtuales de forma eficiente. Dalvik ejecutaba hasta la versión 5.0 archivos en el formato de ejecutable Dalvik (.dex), el cual está optimizado para memoria mínima. La Máquina Virtual está basada en registros y corre clases compiladas por el compilador de Java que han sido transformadas al formato.dex por la herramienta incluida dx. Desde la versión 5.0 utiliza el ART, que compila totalmente al momento de instalación de la aplicación.
- **Núcleo Linux**: Android depende de Linux para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos,

Tabla 4.1: Características y especificaciones 1

Diseño de dispositivo	La plataforma es adaptable a pantallas de mayor resolución, VGA, biblioteca de gráficos 2D, biblioteca de gráficos 3D basada en las especificaciones de la OpenGL ES 2.0 y diseño de teléfonos tradicionales.
Almacenamiento	SQLite, una base de datos liviana, que es usada para propósitos de almacenamiento de datos.
Conectividad	Android soporta las siguientes tecnologías de conectividad: GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, HSDPA, HSPA+, NFC y WiMAX, GPRS, UMTS y HSDPA+.
Mensajería	SMS y MMS son formas de mensajería, incluyendo mensajería de texto, además del servicio de Firebase Cloud Messaging (FCM) siendo la nueva versión de Google Cloud Messaging (GCM) bajo la marca Firebase con los nuevos SDK para realizar el desarrollo de mensajería en la nube mucho más sencillo.
Navegador web	El navegador web incluido en Android está basado en el motor de renderizado de código abierto WebKit, emparejado con el motor JavaScript V8 de Google Chrome. El navegador por defecto de Ice Cream Sandwich obtiene una puntuación de 100/100 en el test Acid3.
Soporte de Java	Aunque la mayoría de las aplicaciones están escritas en Java, no hay una máquina virtual Java en la plataforma. El bytecode Java no es ejecutado, sino que primero se compila en un ejecutable Dalvik y se ejecuta en la Máquina Virtual Dalvik, Dalvik es una máquina virtual especializada, diseñada específicamente para Android y optimizada para dipositivos móviles que funcionan con batería y que tienen memoria y procesador limitados. A partir de la versión 5.0, se utiliza el Android Runtime (ART). El soporte para J2ME puede ser agregado mediante aplicaciones de terceros como el J2ME MIDP Runner.
Soporte multimedia	Android soporta los siguientes formatos multimedia: WebM, H.263, H.264 (en 3GP o MP4), MPEG-4 SP, AMR, AMR-WB (en un contenedor 3GP), AAC, HE-AAC (en contenedores MP4 o 3GP), MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF y BMP.

Tabla 4.2: Características y especificaciones 2

Soporte para streaming	Streaming RTP/RTSP (3GPP PSS, ISMA), descarga progresiva de HTML (HTML5 <video>tag). Adobe Flash Streaming (RTMP) es soportado mediante el Adobe Flash Player. Se planea el soporte de Microsoft Smooth Streaming con el port de Silverlight a Android. Adobe Flash HTTP Dynamic Streaming estará disponible mediante una actualización de Adobe Flash Player.
Soporte para hardware adicional	Android soporta cámaras de fotos, de vídeo, pantallas táctiles, GPS, acelerómetros, giroscopios, magnetómetros, sensores de proximidad y de presión, sensores de luz, gamepad, termómetro, aceleración por GPU 2D y 3D.
Entorno de desarrollo	Incluye un emulador de dispositivos, herramientas para depuración de memoria y análisis del rendimiento del software. Inicialmente el entorno de desarrollo integrado (IDE) utilizado era Eclipse con el plugin de Herramientas de Desarrollo de Android (ADT). Ahora se considera como entorno oficial Android Studio, descargable desde la página oficial de desarrolladores de Android.
Google Play	Google Play es un catálogo de aplicaciones gratuitas o de pago en el que pueden ser descargadas e instaladas en dispositivos Android sin la necesidad de un PC.
Multi-táctil	Android tiene soporte nativo para pantallas capacitivas con soporte multitáctil que inicialmente hicieron su aparición en dispositivos como el HTC Hero. La funcionalidad fue originalmente desactivada a nivel de kernel (posiblemente para evitar infringir patentes de otras compañías). Más tarde, Google publicó una actualización para el Nexus One y el Motorola Droid que activa el soporte multitáctil de forma nativa.
Bluetooth	El soporte para A2DF y AVRCP fue agregado en la versión 1.5; el envío de archivos (OPP) y la exploración del directorio telefónico fueron agregados en la versión 2.0; y el marcado por voz junto con el envío de contactos entre teléfonos lo fueron en la versión 2.2.
Videollamada	Android soporta videollamada a través de Hangouts (antiguo Google Talk) desde su versión HoneyComb.
Multitarea	Multitarea real de aplicaciones está disponible, es decir, las aplicaciones que no estén ejecutándose en primer plano reciben ciclos de reloj.
Características basadas en voz	La búsqueda en Google a través de voz está disponible como “Entrada de Búsqueda” desde la versión inicial del sistema.

pila de red y modelo de controladores. El núcleo también actúa como una capa de abstracción entre el hardware y el resto de la pila de software [15].

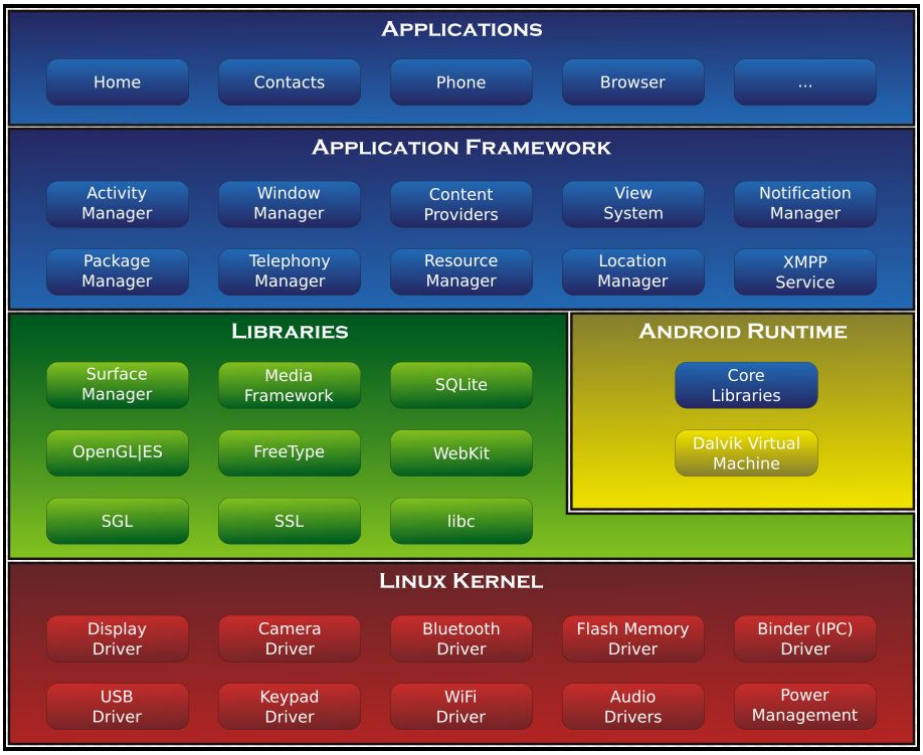

Figura 4.2: Arquitectura del sistema Android.

Versiones

Las versiones de Android reciben, en inglés, el nombre de diferentes postres o dulces. En cada versión el postre o dulce elegido empieza por una letra distinta, conforme a un orden alfabético [15]:

Tabla 4.3: Versiones de Android

Letra	Nombre	Versión	Traducción.
A	Apple Pie	1.0	Tarta de Manzana.
B	Banana Bread	1.1	Pan de plátano.
C	Cupcake	1.0	Cupcake.
D	Donut	1.6	Rosquilla, dónut o Dona.
E	Éclair	2.0/2.1	Pepito o relámpago.
F	Froyo	2.0	Yogur helado.
G	Gingerbread	2.3	Pan de jengibre.
H	Honeycomb	3.0/3.1/3.2	Panal.
I	Ice Cream Sandwich	4.0	Sándwich de helado.
J	Jelly Bean	4.1/4.2/4.3	Gominola o pastilla de goma.
K	KitKat	4.4	KitKat.
L	Lollipop	5.0/5.1	Paleta o Piruleta.
M	Marshmallow	6.0/6.0.1	Malvavisco o Bombón o nube.
N	Nougat	7.0/7.1/7.1.1/7.1.2	Turrón.
O	Oreo	8.0/8.1	Oreo.
P	P	9.0	

4.2. Android Studio

Android Studio es el entorno de desarrollo integrado oficial para la plataforma Android. Fue anunciado el 16 de mayo de 2013 en la conferencia Google I/O, y reemplazó a Eclipse como el IDE oficial para el desarrollo de aplicaciones para Android. La primera versión estable fue publicada en diciembre de 2014. Está basado en el software IntelliJ IDEA de JetBrains y ha sido publicado de forma gratuita a través de la Licencia Apache 2.0. Está disponible para las plataformas Microsoft Windows, macOS y GNU/Linux. Ha sido diseñado específicamente para el desarrollo de Android.

Características

Las siguientes características se proporcionan en la versión estable actual [16]:

- Integración de ProGuard y funciones de firma de aplicaciones.
- Renderizado en tiempo real.

- Consola de desarrollador: consejos de optimización, ayuda para la traducción, estadísticas de uso.
- Soporte para construcción basada en **Gradle**.
- **Refactorización** específica de Android y arreglos rápidos.
- Un editor de diseño enriquecido que permite a los usuarios arrastrar y soltar componentes de la interfaz de usuario.
- Herramientas **Lint** para detectar problemas de rendimiento, usabilidad, compatibilidad de versiones y otros problemas.
- Plantillas para crear diseños comunes de Android y otros componentes.
- Soporte para programar aplicaciones para **Android Wear**.
- Soporte integrado para Google Cloud Platform, que permite la integración con Google Cloud Messaging y App Engine.
- Un dispositivo virtual de Android que se utiliza para ejecutar y probar aplicaciones.

Plataformas

Android Studio está disponible para Windows 2003, Vista, 7, 8, y 10, tanto plataformas de 32 como de 64 bits, GNU/Linux, Linux con GNOME o KDE y 2 GB de memoria RAM mínimo y macOS, desde 10.8.5 en adelante [16].

Requisitos del sistema

Requisitos del sistema para la plataforma utilizada durante el desarrollo del proyecto: **Versión 3.x** [16]

- **OS Version**: Windows 10/8/7 (32- o 64-bit).
- **RAM**: 3 GB RAM mínimo, 8 GB RAM recomendado más 1GB adicional para el emulador de Android.
- **Espacio en disco**: 2 GB de espacio en disco para Android Studio, 4GB recomendados (500MB para la IDE y al menos 1.5 GB para Android SDK, imágenes de sistema de emulador y cachés).
- **Java version**: Java Development Kit (JDK) 8.
- **Resolución de pantalla**: 1280x800 mínimo, 1440x900 recomendado.

AVDManager

Concepto

AVD Manager (Android Virtual Device Manager) es un gestor para los dispositivos virtuales de Android, es decir, los emuladores.

Funcionalidad

Para crear y gestionar los emuladores de Android en los que se probaran las aplicaciones. Estos emuladores pueden ser creados con perfiles predefinidos de dispositivos de fabricantes reales, o con características propias personalizadas por el programador.

Configuración

- Tools>AVD Manager.
- Clic en el botón "+ Create Virtual Device".
- Seleccionamos la categoría del dispositivo virtual que queremos crear: **Phone**.
- Seleccionamos un fabricante con las características deseadas.
- Clic en el botón "Next".
- Seleccionamos la versión del sistema operativo que queremos usar. Si no está disponible, se puede descargar cualquier versión haciendo clic en "Download".
- Clic en el botón "Next".
- Verificamos la configuración del emulador que se va a crear. Se puede editar la configuración recomendada haciendo clic en el botón "Show Advanced Settings".
- Clic en el botón "Finish".

Modo de empleo - Ejecución

Para poder ejecutar nuestra aplicación en un emulador, desde la ventana AVD Manager seleccionamos un emulador (previamente creado) y en la pestaña **Actions** hacemos clic en el botón "Play".

SDK

El SDK (Software Development Kit) de Android, incluye un conjunto de herramientas de desarrollo. Comprende un depurador de código, biblioteca, un simulador de teléfono basado en QEMU, documentación, ejemplos de código y tutoriales. Las plataformas de desarrollo soportadas incluyen GNU/Linux, Mac OS X 10.5.8 o posterior, y Windows XP o posterior. La plataforma integral de desarrollo (IDE, Integrated Development Environment) soportada oficialmente es Android Studio junto con el complemento ADT (Android Development Tools plugin). Además, los programadores pueden usar un editor de texto para escribir ficheros Java y XML y utilizar comandos en un terminal (se necesitan los paquetes JDK, Java Development Kit y Apache Ant) para crear y depurar aplicaciones, así como controlar dispositivos Android que estén conectados (es decir, reiniciarlos, instalar aplicaciones en remoto, etc.). Las Actualizaciones del SDK están coordinadas con el desarrollo general de Android. El SDK soporta también versiones antiguas de Android, por si los programadores necesitan instalar aplicaciones en dispositivos ya obsoletos o más antiguos. Las herramientas de desarrollo son componentes descargables, de modo que una vez instalada la última versión, pueden instalarse versiones anteriores y hacer pruebas de compatibilidad [11].

4.3. SQLite

Concepto

SQLite es una base de datos de código abierto que está integrada en Android. SQLite admite estándar características de bases de datos relacionales como sintaxis SQL, transacciones y declaraciones preparadas. Además, requiere solo poca memoria en tiempo de ejecución (aproximadamente 250 KByte). SQLite admite los tipos de datos TEXT (similar a String en Java), INTEGER (similar a long en Java) y REAL (similar al doble en Java). Todos los demás tipos deben convertirse en uno de estos campos antes guardándolos en la base de datos. SQLite en sí no valida si los tipos escritos en las columnas son en realidad del tipo definido, p. puedes escribir un entero en una columna de cadena y viceversa. Se puede encontrar más información sobre SQLite en el sitio web de SQLite: http://www.sqlite.org [10].

SQLite en Android

SQLite está disponible en todos los dispositivos Android. El uso de una base de datos SQLite en Android no requiere ninguna configuración o administración de base de datos. Solo debe definir las instrucciones SQL para crear y actualizar la base de datos. Luego, la plataforma Android administrará automáticamente la base de datos. El acceso a una base de datos SQLite implica el acceso al sistema de archivos. Esto puede ser lento. Por lo tanto, se recomienda realizar operaciones de base de datos de forma asincrónica, por ejemplo, dentro de la clase `AsyncTask`. Si su aplicación crea una base de datos, esta base de datos se guarda de manera predeterminada en el directorio "DATAdata APPNAMEdatabasesFILENAME". Las partes del directorio anterior se construyen en base a las siguientes reglas. **DATA** es el camino que el método `Environment.getDataDirectory()` devuelve. **APP_NAME** es el nombre de tu aplicación.**FILENAME** es el nombre que especificas en tu código de aplicación para la base de datos.

4.4. DB Browser for SQLite

En este proyecto, DB Browser for SQLite se ha utilizado básicamente para comprobar que los datos creados o registrados en nuestra base da datos (desde nuestra aplicación) son correctos.

Concepto

DB Browser for SQLite es una herramienta de alta calidad, visual y de código abierto para crear, diseñar y editar archivos de bases de datos compatibles con SQLite. Es para usuarios y desarrolladores que desean crear bases de datos, buscar y editar datos. Utiliza una interfaz familiar similar a una hoja de cálculo, y no necesita aprender comandos SQL complicados. Los controles y asistentes están disponibles para que los usuarios [8]:

- Crear y compactar archivos de base de datos.
- Crear, definir, modificar y eliminar tablas.
- Crear, definir y eliminar índices.
- Examinar, editar, agregar y eliminar registros.
- Registros de búsqueda.

- Importar y exportar registros como texto.
- Importar y exportar tablas de / a archivos CSV.
- Importar y exportar bases de datos desde / a archivos de volcado de SQL.
- Emita consultas SQL e inspeccione los resultados.
- Examine un registro de todos los comandos SQL emitidos por la aplicación.

Este programa no es un shell visual para la herramienta de línea de comandos sqlite. No requiere familiaridad con los comandos SQL. Es una herramienta para ser utilizada tanto por los desarrolladores como por los usuarios finales, y debe ser tan simple de usar como sea posible para alcanzar sus objetivos [8].

4.5. Java

Java es un lenguaje de programación de propósito general, concurrente, orientado a objetos, que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo (conocido en inglés como WORA, o "write once, run anywhere"), lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones de cliente-servidor de web, con unos diez millones de usuarios reportados [22]. Webs de la herramienta:

- https://www.oracle.com/java/index.html
- https://www.java.com/es/download/

4.6. JUnit

JUnit es un conjunto de bibliotecas creadas por Erich Gamma y Kent Beck que son utilizadas en programación para hacer pruebas unitarias de aplicaciones Java. JUnit es un conjunto de clases (framework) que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta

como se espera. Es decir, en función de algún valor de entrada se evalúa el valor de retorno esperado; si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente. JUnit es también un medio de controlar las pruebas de regresión, necesarias cuando una parte del código ha sido modificado y se desea ver que el nuevo código cumple con los requerimientos anteriores y que no se ha alterado su funcionalidad después de la nueva modificación [12].

JUnit en Android Studio

Desde la versión 1.1 de Android Studio existe, lo que han llamado desde google, soporte para test unitarios. Esto quiere decir que podemos ejecutar test unitarios sin necesidad de desplegarlos en un dispositivo o emulador, se van a ejecutar en la máquina virtual de java.

Estos test untarios se conocen también como test locales o test de jvm (java virtual machine).

4.7. Espresso

Espresso es un `framework` de testing `open source` lanzado por Google el cuál provee una API que permite crear pruebas de interfaz de usuario (de ahora en adelante UI por sus siglas en inglés) para simular interacciones de usuarios en una aplicación Android (en la versión 2.2 en adelante). Es una buena práctica simular los diferentes escenarios en los que el usuario puede interacturar con una aplicación para evitar que este se encuentre con resultados inesperados o bien tenga una mala experiencia al momento de su uso. Es por esta razón que es recomendable la creación de un entorno de pruebas vinculadas a la UI con el fin de asegurarnos que la aplicación está funcionando correctamente. Su principal ventaja es que nos permite la sincronización automática de las acciones de las pruebas con la interfaz de usuario de nuestra aplicación. Además, permite la ejecución de pruebas en máquinas x86 en un ambiente multihilo, solucionando problemas de concurrencia asociados al testing de UI. Espresso permite realizar pruebas tanto en dispositivos físicos como virtuales (emuladores) [18].

4.8. Git

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos. Al principio, Git se pensó como un motor de bajo nivel sobre el cual otros pudieran escribir la interfaz de usuario o `front end` como Cogito o StGIT. Sin embargo, Git se ha convertido desde entonces en un sistema de control de versiones con funcionalidad plena. Hay algunos proyectos de mucha relevancia que ya usan Git, en particular, el grupo de programación del núcleo Linux. El mantenimiento del software Git está actualmente (2009) supervisado por Junio Hamano, quien recibe contribuciones al código de alrededor de 280 programadores. En cuanto a derechos de autor Git es un software libre distribuible bajo los términos de la versión 2 de la Licencia Pública General de GNU [19].

Características

El diseño de Git mantiene una enorme cantidad de código distribuida y gestionada por mucha gente, que incide en numerosos detalles de rendimiento, y de la necesidad de rapidez en una primera implementación. Entre las características más relevantes se encuentran:

- Fuerte apoyo al desarrollo no lineal, por ende, rapidez en la gestión de ramas y mezclado de diferentes versiones. Git incluye herramientas específicas para navegar y visualizar un historial de desarrollo no lineal. Una presunción fundamental en Git es que un cambio será fusionado mucho más frecuentemente de lo que se escribe originalmente, conforme se pasa entre varios programadores que lo revisan.
- Gestión distribuida. Al igual que Darcs, BitKeeper, Mercurial, SVK, Bazaar y Monotone, Git le da a cada programador una copia local del historial del desarrollo entero, y los cambios se propagan entre los repositorios locales. Los cambios se importan como ramas adicionales y pueden ser fusionados en la misma manera que se hace con la rama local.
- Los almacenes de información pueden publicarse por HTTP, FTP, rsync o mediante un protocolo nativo, ya sea a través de una conexión TCP/IP simple o a través de cifrado SSH. Git también puede emular

servidores CVS, lo que habilita el uso de clientes CVS pre-existentes y módulos IDE para CVS pre-existentes en el acceso de repositorios Git.

- Los repositorios Subversion y svk se pueden usar directamente con git-svn.
- Gestión eficiente de proyectos grandes, dada la rapidez de gestión de diferencias entre archivos, entre otras mejoras de optimización de velocidad de ejecución.
- Todas las versiones previas a un cambio determinado implican la notificación de un cambio posterior en cualquiera de ellas a ese cambio (denominado autenticación criptográfica de historial). Esto existía en Monotone.
- Resulta algo más caro trabajar con ficheros concretos frente a proyectos, eso diferencia el trabajo frente a CVS, que trabaja con base en cambios de fichero, pero mejora el trabajo con afectaciones de código que concurren en operaciones similares en varios archivos.
- Los renombrados se trabajan basándose en similitudes entre ficheros, aparte de nombres de ficheros, pero no se hacen marcas explícitas de cambios de nombre con base en supuestos nombres únicos de nodos de sistema de ficheros, lo que evita posibles, y posiblemente desastrosas, coincidencias de ficheros diferentes en un único nombre.
- Realmacenamiento periódico en paquetes (ficheros). Esto es relativamente eficiente para escritura de cambios y relativamente ineficiente para lectura si el reempaquetado (con base en diferencias) no ocurre cada cierto tiempo.

4.9. GitHub-Desktop

GitHub es una plataforma de alojamiento de código para control de versiones y colaboración. En otras palabras, GitHub es un alojamiento de repositorios Git que nos permite trabajar en proyectos desde cualquier lugar [6].

GitHub Desktop es una aplicación de escritorio de GitHub que nos permite empezar a utilizar un control de versiones sin problemas. GitHub Desktop es una interfaz gráfica de usuario diseñada para facilitar el uso de Git, es decir, permite al usuario interactuar con el programa a través de un dispositivo visual que reemplaza la línea de comandos.

4.10. LaTeX

LaTeX es un sistema de composición de textos, orientado a la creación de documentos escritos que presenten una alta calidad tipográfica. Por sus características y posibilidades, es usado de forma especialmente intensa en la generación de artículos y libros científicos que incluyen, entre otros elementos, expresiones matemáticas. LaTeX está formado por un gran conjunto de macros de TeX, escrito por Leslie Lamport en 1984, con la intención de facilitar el uso del lenguaje de composición tipográfica, TEX, creado por Donald Knuth. Es muy utilizado para la composición de artículos académicos, tesis y libros técnicos, dado que la calidad tipográfica de los documentos realizados en LaTeX, se la considera adecuada a las necesidades de una editorial científica de primera línea, muchas de las cuales ya lo emplean. LaTeX es software libre bajo licencia LPPL [23].

4.11. Texmaker

Texmaker es un editor gratuito distribuido bajo la licencia GPL para escribir documentos de texto, multiplataforma, que integra muchas herramientas necesarias para desarrollar documentos con LaTeX, en una sola aplicación. Texmaker incluye soporte Unicode, corrección ortográfica, autocompletado, plegado de código y un visor incorporado en pdf con soporte de synctex y el modo de visualización continua. Para que Texmaker pueda funcionar es necesario haber instalado TeX previamente [13].

4.12. Photoshop CC 2017

Adobe Photoshop es un editor de gráficos rasterizados desarrollado por Adobe Systems Incorporated. Usado principalmente para el retoque de fotografías y gráficos, su nombre en español significa literalmente “taller de fotos”. Es líder mundial del mercado de las aplicaciones de edición de imágenes y domina este sector de tal manera que su nombre es ampliamente empleado como sinónimo para la edición de imágenes en general [14]. En el desarrollo de este proyecto, se ha utilizado Adobe Photoshop CC 2017 únicamente para la creación y edición del icono de la app.

Aspectos relevantes del desarrollo del proyecto

A continuación, se listarán los aspectos con mayor relevancia para la comprensión y el correcto funcionamiento del proyecto. A su vez, se incluirán únicamente los aspectos relevantes encontrados en esta nueva versión. A sí mismo, se pueden consultar los aspectos relevantes comunes y no comunes con la versión anterior en la memoria del proyecto de Mario López Jiménez [3].

5.1. Formación

Para la realización de este proyecto, se requería algo de experiencia previa en **Java**, **Android** y **Testing**, de los cuales, a excepción de Java, carecía en un principio. Para obtener la experiencia que exigía el proyecto, se llevó a cabo una formación en esos temas mediante lecturas de manuales, documentos online, e incluso, mediante la visualización de pequeños tutoriales encontrados en internet. Una vez que se adquirió cierto nivel de conocimientos, se empezó el desarrollo de una nueva versión para la aplicación **UBUDiabetes** en **Android Studio**. Cabe destacar que, durante el desarrollo del proyecto, se continuó con la formación y adquisición de ciertos conocimientos que eran necesarios para continuar con la realización del mismo.

Por otra parte, al tratarse del desarrollo de una nueva versión, se realizo un estudio previo del TFG de Lara Bartolomé Casado [1] y del TFG de Mario López Jiménez [3] con el propósito de comprender la naturaleza y funcionalidad de la aplicación UBUDiabetes. .

5.2. Ciclo de vida del proyecto

Si tenemos en cuenta el momento en que se empezó a realizar la formación en Android y Testing, se puede decir que el proyecto dio comienzo el 8 de Enero de 2018 tras una previa reunión con Raúl Marticorena Sánchez, tutor de este. El desarrollo del proyecto se llevo a cabo en base a dos tipos de reuniones. Por un lado, reuniones con el tutor del proyecto, cuyos objetivos principales eran revisar el trabajo realizado hasta la fecha y fijar nuevas tareas de cara a la siguiente reunión. Estas reuniones se acordaron realizarlas cada semana, teniendo en cuenta la disponibilidad de los miembros. Por otro lado, se llevaron a cabo reuniones entre Raúl Marticorena Sánchez y los clientes de la aplicación, en este caso, se trataba de miembros del Área de enfermería (Grado de Enfermería) de la facultad de Ciencias de la Salud, como Diego Serrano Gómez y Jesús Puente Alcaraz, tutores del proyecto "Salud electrónica y diabetes: Elementos para el diseño de una aplicación informática (APP) para jóvenes diabéticos de tipo I.", llevado a cabo por Lara Bartolomé Casado. Estas reuniones tenían por cometido mostrarles el trabajo realizado, tomar nota de posibles mejoras y/o nuevos requisitos a incluir y solicitar a los clientes nuevos casos de prueba reales para continuar con la validación de la aplicación. La evolución cronológica del proyecto es la siguiente: Como primer paso, una vez asignado el proyecto y realizada la formación necesaria para su desarrollo, se importó el proyecto de la versión que se tenía hasta el momento de la aplicación. Este proyecto ya incluía ciertas mejoras llevadas a cabo por Raúl Marticorena Sánchez.

Tras la importación del proyecto previo, se realizaron ciertas configuraciones en Android Studio para poder ejecutar el proyecto y visualizar en un emulador creado previamente lo que se tenía hasta el momento. Estas configuraciones fueron necesarias ya que el proyecto que se tenía hasta entonces fue realizado en una versión 1.X de Android Studio y la versión que disponía de Android Studio para realizar una nueva versión era superior a la 3.x, lo que provocó ciertas incompatibilidades. Una vez ejecutada la aplicación por primera vez, se dio comienzo al desarrollo de una nueva versión de la aplicación. Las tareas en cuanto al desarrollo de la nueva versión se ejecutaron en el siguiente orden:

- Corregir las clases y la interfaz gráfica relacionada con el cálculo del Bolo Corrector.
- Llevar a cabo en paralelo el desarrollo de las pruebas unitarias para el cálculo del Bolo Corrector.

- Añadir nuevas clases e interfaces gráficas según los nuevos requisitos definidos.
- Llevar a cabo en paralelo el desarrollo de las pruebas unitarias, integración y/o pruebas de Interfaz de Usuario.
- Realizar versiones beta.

Como resultado de este proceso, se obtuvo una versión final para la aplicación **UBUDiabetes**.

5.3. Decisiones Técnicas

En este apartado se explicaran las decisiones tomadas en cuanto a los aspectos técnicos del mismo. Se han categorizado en varios apartados según las *Activities* (Pantallas) que forman parte de la interfaz gráfica y en las que se han incluido modificaciones respecto a la versión anterior. Las decisiones técnicas comunes con la versión anterior se pueden consultar en la memoria del proyecto de Mario López Jiménez [3].

Nuevo Proyecto

Tras analizar el proyecto del que se partía, es decir, del proyecto de Mario López Jiménez, se comprobó que al realizarse en una versión de Android Studio 1.x no disponía de la estructura necesaria para la creación de las Pruebas Instrumentadas que se requerían en este proyecto 5.3:

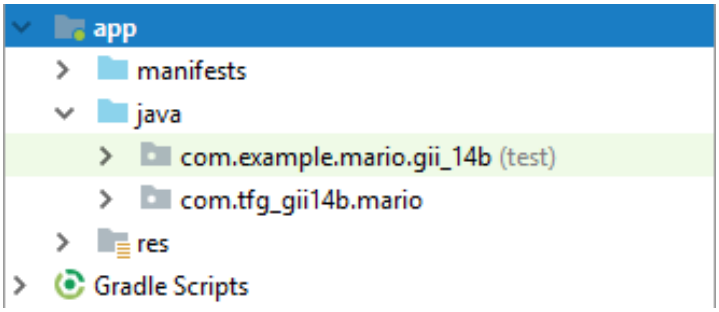

Figura 5.3: Estructura Proyecto UBUDiabetes 1.0

Por ello, se tomó la decisión, de crear un nuevo proyecto desde cero, el cual crearía automáticamente la estructura necesaria para el desarrollo de

las distintas Pruebas Instrumentadas. Una vez creado un nuevo proyecto se migraron los datos del antiguo proyecto al nuevo, el cuál, tenía la siguiente estructura 5.4:

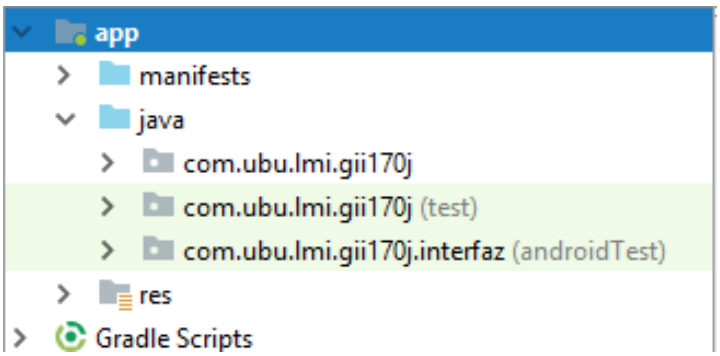

Figura 5.4: Estructura Proyecto UBUDiabetes 2.0

La versión de Android en la que se desarrolló la aplicación es distinta a la versión previa desarrollada por Mario López Jiménez. Se tuvo en cuenta la información proporcionada en la pagina **Android Developer** [4] que muestra datos sobre la cantidad relativa de dispositivos que usan una versión determinada de la plataforma Android.

UBUDiabetes 1.0 5.5:

```
android {
    compileSdkVersion 23
    buildToolsVersion "23.0.2"

    defaultConfig {
        applicationId "com.example.mario.gii_14b"
        minSdkVersion 15
        targetSdkVersion 23
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}
```

Figura 5.5: Versión Android UBUDiabetes 1.0

UBUDiabetes 2.0 5.6:

```
android {

    //Version 8.0 Android
    compileSdkVersion 26
    buildToolsVersion '27.0.3'

    defaultConfig {
        applicationId "com.ubu.lmi.gii170j"
        minSdkVersion 16
        targetSdkVersion 26
        versionCode 2
        versionName "2.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {

        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }

    }
    testOptions {
        unitTests.returnDefaultValues = true
    }
}
```

Figura 5.6: Versión Android UBUDiabetes 2.0

- **`compileSdkVersion`**: Indica desde que nivel de API se ha compilado la aplicación.
- **`minSdkVersion`**: Valor entero que designa el nivel de API mínimo que se requiere para que se ejecute la aplicación. El sistema de Android impedirá que el usuario instale la aplicación si el nivel de API del sistema es inferior al valor especificado en este atributo. Siempre debes declarar este atributo.
- **`targetSdkVersion`**: Valor entero que designa el nivel de API al cual se dirige la aplicación. Si no se configura, el valor predeterminado es igual al valor asignado a la minSdkVersion. Este atributo informa al sistema que has realizado las pruebas en la versión prevista y el sistema no deberá habilitar ningún comportamiento de compatibilidad a fin de mantener la compatibilidad con versiones posteriores de tu aplicación y la versión prevista. La aplicación puede continuar ejecutándose en versiones más antiguas (anteriores a la minSdkVersion).

Splash Screen

Se propuso a Raúl Marticorena Sanchéz, tutor de este proyecto, la idea de implementar una **"Splash Screen"** para presentar la aplicación a los usuarios. Tras el visto bueno de esta idea, se introdujo la "Splash Screen" la cual se lanzará primero siempre que se ejecute la aplicación. Esta pantalla contenía una imagen de fondo con el nombre y color base de la aplicación y una `progressbar` 5.7.

Figura 5.7: Splash Screen

Perfil-Registro

Aunque la pantalla de Perfil-Registro ya estaba implementada 5.8, se tomó la decisión de modificar su diseño desde cero con el fin de conseguir un diseño más atractivo para el usuario y que vaya acorde con la idea planteada por los miembros del Área de Enfermería mencionados anteriormente. UBUDiabetes 1.0 5.8:

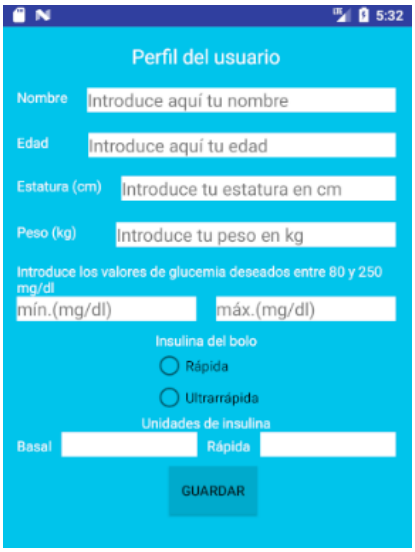

Figura 5.8: Registro-Perfil UBUDiabetes 1.0

UBUDiabetes 2.0 5.9 5.10:

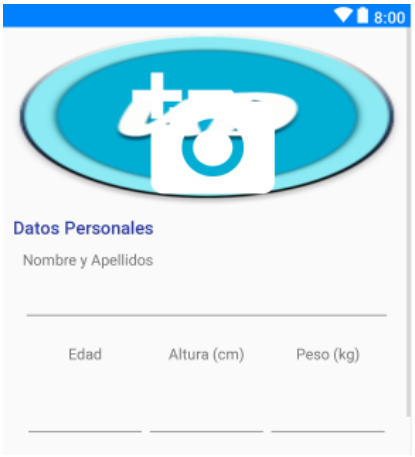

Figura 5.9: Registro-Perfil UBUDiabetes 2.0 (A)

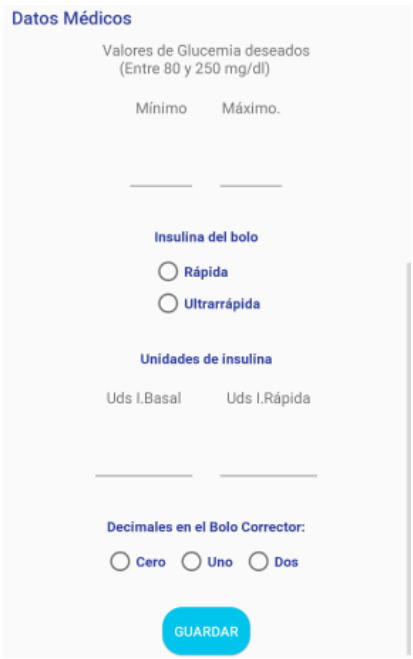

Figura 5.10: Registro-Perfil UBUDiabetes 2.0 (B)

Menú Principal

Se modificó el diseño inicial de esta pantalla al aumentar el numero de opciones principales disponibles para el usuario en la aplicación UBUDiabetes. Así mismo, se busco implementar un diseño mas atractivo e interactivo para el usuario.
UBUDiabetes 1.0 5.11:

Figura 5.11: Menú principal UBUDiabetes 1.0

UBUDiabetes 2.0 5.12:

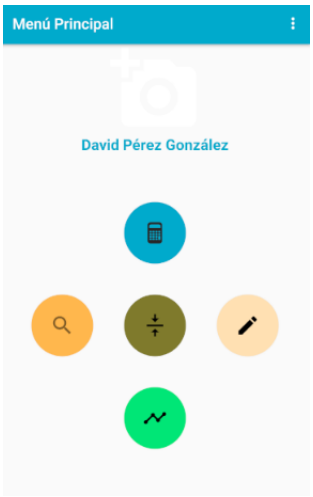

Figura 5.12: Menú principal UBUDiabetes 2.0

Carbohidratos

No solo se modificó completamente el diseño de esta pantalla, sino que, también se modificó su código ya que esta pantalla realiza la función principal de la aplicación, que es, el calculo del bolo corrector. En el trabajo de fin de de Grado realizado por Bruno Martín Gómez [2] se descubrieron

ciertos fallos en la versión previa de UBUDiabetes 2.0. Uno de estos fallos estaba relacionado con el calculo del bolo corrector, el cual no obtenía los resultados esperados en ciertos casos, lo que supuso un problema para los usuarios. Tras corregir este fallo, se añadieron nuevos elementos en el diseño. Estos elementos fueron añadidos teniendo en cuenta los nuevos requisitos solicitados por los clientes.

- **Buscador de alimentos**: Una vista de texto editable que muestra las sugerencias de finalización de forma automática mientras el usuario escribe. La lista de sugerencias se muestra en un menú desplegable desde el cual el usuario puede elegir un elemento para reemplazar el contenido del cuadro de edición. La lista de sugerencias se obtiene de un *adaptador de datos* y aparece solo después de un número dado de caracteres definidos por el umbral.
- **Lista de alimentos**: Lista que contiene todos los alimentos tomados de la tabla de raciones de hidratos de carbono de la **FUNDACIÓN ESPAÑOLA PARA LA DIABETES** [9].
- **Lista de Ingesta**: Lista que muestra los alimentos que el usuario va introduciendo para el cálculo de hidratos de carbono junto con su cantidad en gramos y su índice glucémico.
- **Botón eliminar alimento**: Al mantener seleccionado un alimento de la lista de ingesta, se da la opción al usuario de eliminar dicho alimento de la lista.
- **Bolo corrector e Hidratos de carbono parcial**: Textos que muestran al usuario la suma de hidratos de carbono y el bolo calculado según los alimentos introducidos en la lista de ingesta.

UBUDiabetes 1.0 5.13:

Figura 5.13: Carbohidratos UBUDiabetes 1.0

UBUDiabetes 2.0 5.14:

Figura 5.14: Carbohidratos UBUDiabetes 2.0

Listas de Ingestas y Detalles de la lista

Estas nueva pantallas se añadieron teniendo en cuenta las recomendaciones de los usuarios recogidas en el trabajo de fin de grado de Bruno Martín Gómez [2]. Para su implementación, se propuso a Raúl Marticorena Sanchéz la creación de dos tablas, "listaingesta" y "detalleslistaingesta", las cuales están asociadas entre sí de tal manera que una fila de la tabla "listaingesta" puede estar relacionada con una o varias filas de la tabla "detalleslistaingesta"

- Listas de Ingestas 5.15: Esta pantalla muestra los registros de todos los cálculos del bolo corrector que se han llevado a cabo por el usuario. Cada registro muestra su identificador, su fecha de registro, la suma total de los Hidratos de Carbono consumidos y el bolo calculado.

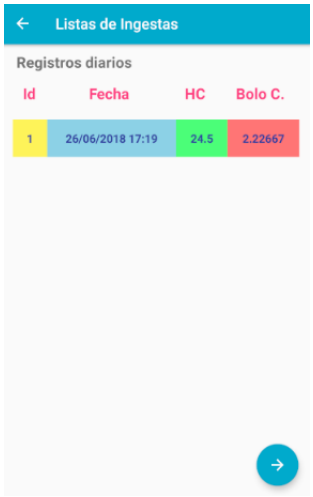

Figura 5.15: Listas de Ingestas

- Detalles de la lista 5.16: Esta pantalla muestra con detalle cada uno de los alimentos ingeridos por el usuario según la lista de ingesta seleccionada en la pantalla "Listas de Ingestas".

Figura 5.16: Detalles de la lista

Ajustes

UBUDiabetes 1.o contaba con un submenú dentro de la pantalla Menú Principal. Este submenú tenía dos opciones: Ajustes de Perfil y Acerca de 5.17.

Figura 5.17: Submenú Principal UBUDiabetes 1.0

Pensando en la escalabilidad de la aplicación, se tomó la decisión de modificar este submenú.

- Se mantuvo la opción Acerca de, que muestra un pequeño mensaje en pantalla con información acerca de la aplicación 5.18.

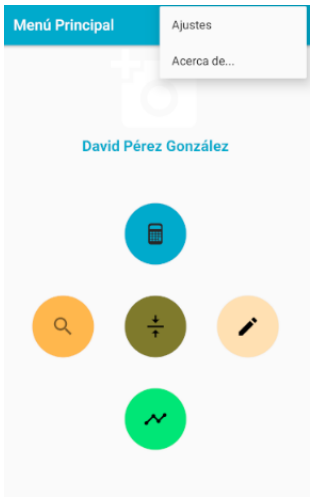

Figura 5.18: Submenú Principal UBUDiabetes 2.0

- Se modifico la opción Ajustes de Perfil. Esta opción paso a llamarse únicamente Ajustes. Dicha opción nos llevara a una nueva pantalla en la que se incluirían todas las opciones de configuración necesarias para el usuario y/o configuración de la aplicación 5.19.

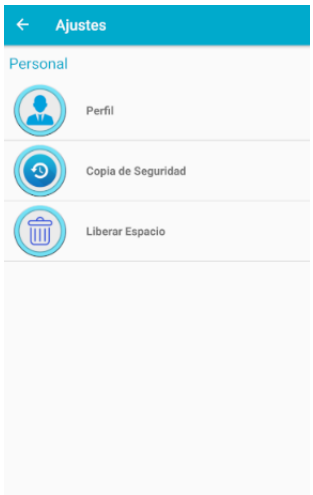

Figura 5.19: Ajustes UBUDiabetes 2.0

Otras decisiones

- Diseño mediante *fragments*: Al igual que UBUDiabetes 1.0, este proyecto mantiene el diseño de sus pantallas mediante *fragments*, posibilitando así, una futura migración a dispositivos tablet.

- *SharedPreferences*: Se mantiene el uso de esta interfaz para acceder y modificar los datos de preferencia del usuario. Estos datos se pueden obtener gracias al *método getSharedPreferences*. Las modificaciones de las preferencias deben pasar por un objeto *SharedPreferences.Editor* para garantizar que los valores de preferencia permanezcan en un estado constante y controlados.

- *ListAdapters* personalizados: UBUDiabetes 2.0 hace uso de varias listas personalizadas: Pantalla Ajustes, Pantalla Listas de Ingestas, Pantalla Detalles de la lista, Pantalla Carbohidratos. Los datos de estas listas se muestran gracias a los Adaptadores que nos proporciona Android. Estos adaptadores funcionan como un puente entre el componente de la interfaz de usuario, en nuestro caso, una *ListView*, y la fuente de datos que nos ayuda a completar los datos en el componente de la interfaz de usuario. De esta forma, la *ListView* puede mostrar cualquier dato siempre que esté envuelto en un *ListAdapter*. Cada ListAdapter está diseñado según el diseño de las filas de las listas que se desean mostrar al usuario. Esta decisión se tomo pensando en la escalabilidad de la aplicación, si en un futuro de desean modificar estas listas, simplemente bastaría con modificador los adaptadores de dichas listas 5.20.

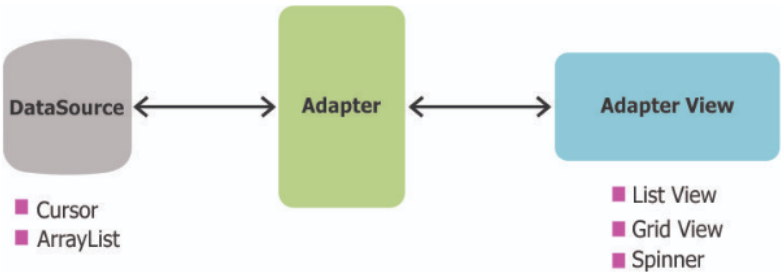

Figura 5.20: Adaptadores Android

- *DataBaseManager y DataBaseHelper*: UBUDiabetes 2.0 mantiene estas clases, que son las encargadas de la persistencia de la aplicación.
- *Arrays.xml*: Para almacenar la información de los alimentos, UBUDiabetes 2.0 hace uso de los archivos .xml que dispone Android Studio, así se evitaría la saturación de código 5.21 en las clases principales. Estos archivos se encuentran separados del código principal 5.22.

```
/**
 * Datos de las raciones en gramos para cada tipo de alimento.
 * Ver: http://www.fundaciondiabetes.org/upload/publicaciones_ficheros/71/TABLAHC.pdf.
 */
public static final int[] raciones =
        {200, 50, 50, 50, 100, // Lacteos - 22
                200, 200, 200, 200, 20,
                25, 300, 50, 70, 250,
                0, 200, 125, 70, 70,
                100, 200, // Fin de lacteos
                //
                13, 38, 13, 40, 12, // Cereales - 74 - Fila 1
                13, 34, 17, 34, 15, // 2
                50, 14, 42, 15, 38, // 3
                15, 20, 15, 65, 50, // 4
                40, 16, 15, 14, 18, // 5
                20, 50, 100, 15, 17, // 6
                70, 30, 24, 20, 50, // 7
                20, 50, 50, 90, 20, // 8
                15, 53, 15, 20, 20, // 9
                20, 18, 23, 15, 15, // 10
                15, 15, 50, 16, 50, // 11
                35, 30, 20, 15, 80, // 12
                19, 48, 14, 90, 30, // 13
                100, 45, 12, 33, 14, // 14
                42, 16, 39, 33,      // 15

                0, 150, 100, 30, 25, // Frutas -  42 - linea 1
                100, 50, 100, 200, 150, // 2
                15, 150, 200, 70, 200,  // 3
```

Figura 5.21: Raciones de HC-UBUDiabetes 1.0

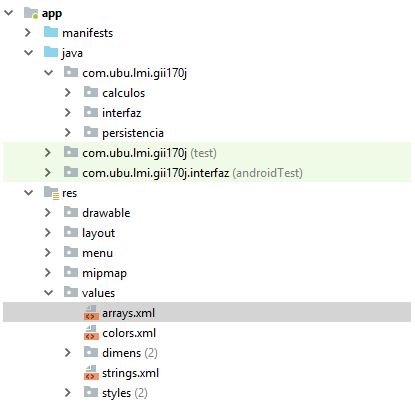

Figura 5.22: Arrays.xlm Android Studio

Figura 5.23: Raciones de HC-UBUDiabetes 2.0

- Iconos personalizados: Se crearon nuevos iconos con la herramienta *Photoshop CC 2017* pensando en la creación de una aplicación amigable e intuitiva. Así mismo, los iconos añaden cierto atractivo y originalidad en cuanto al diseño de la interfaz gráfica del usuario.

- *Testing*: Antes de crear las primeras pruebas unitarias locales o pruebas de Interfaz de usuario es muy importante y necesario comprobar que la estructura del proyecto es la adecuada, es decir, que dentro del directorio **app\java** tiene 3 subdirectorios:

  - Un directorio orientado para la programación del código de la aplicación.
  - Un directorio orientado para la programación de las pruebas unitarias.
  - Un directorio orientado para la programación de las pruebas instrumentales.

  Además, se debe comprobar la correcta configuración de las dependencias de prueba para que el proyecto pueda utilizar las API estándar proporcionadas por el framework JUnit 4 y Espresso. Así mismo, cabe mencionar que para la realización de los *test unitarios* se tomaron los datos de los casos de prueba proporcionados por Bruno Martín Gómez, que se encuentran recogidos en el documento *excel* **"Casos validación UBUDiabetes"**. Dentro de este documento se han utilizado las siguientes hojas:

- **Sin alimentos 1 y Sin alimentos 2**: Estas hojas recogen varios casos de prueba de pacientes con diferentes necesidades de insulina. Estos datos fueron utilizados en la clase "CalculaBolo_NoPreciso_Test" dentro del directorio: **app\java\com.ubu.lmi.gii170j(test)\calculos** Estos test toleran cierto margen de error debido a que los resultados en dichas hojas son valores enteros y sin aplicar ninguna fórmula, lo que no se puede comprobar si los resultados obtenidos por la aplicación son del todo correctos ya que estos son valores con decimales, es decir, valores exactos.
- **Alimentos de 1 en 1**: Esta hoja recoge varios resultados del bolo corrector para un paciente con una glucemia igual a la glucemia objetivo. Estos datos fueron utilizados en la clase "CalculaBolo_Preciso_Test" dentro del directorio: **app\java\com.ubu.lmi.gii170j(test)\calculos** Estos test NO toleran margen de error debido a que los resultados en dicha hoja son valores exactos obtenidos aplicando cierta fórmula, lo que SI se puede comprobar si los resultados obtenidos por la aplicación son correctos ya que estos son valores con decimales, es decir, valores exactos.
- **Casos clínicos**: Esta hoja recoge un caso de prueba concreto para el que la versión anterior de la aplicación obtenía resultados erróneos. Estos datos fueron utilizados en la clase "CalculaBolo_CasoClinico_FalloTest" dentro del directorio: **app\java\com.ubu.lmi.gii170j(test)\calculos** Este test comprueba que el resultado obtenido en esta hoja es igual al obtenido por la aplicación, es decir, comprueba que se ha corregido el error de la versión anterior.

Por otro lado, las pruebas de interfaz, es decir, las pruebas instrumentales se llevaron a cabo dentro del directorio:
app\java\com.ubu.lmi.gii170j\interfaz(androidTest)

Trabajos relacionados

A continuación, se hará un breve resumen acerca de los trabajos relacionados con este proyecto. Cabe mencionar que se puede consultar más información acerca de este apartado en el trabajo de fin de grado realizado por Lara Bartolomé Casado [1]. Como bien menciona Lara, en la actualidad existen alrededor de unas 97.000 aplicaciones móviles de salud disponible para su descarga. Dentro de estas aplicaciones, la mayoría tratan sobre enfermedades endocrinas como la Diabetes. La mayoría de estas ayudan a los usuarios a controlar la dosis de insulina o medicación, la glucemia capilar, o apoyan las tareas de autocuidado, como el ejercicio o la dieta, es decir, funcionas como aplicaciones de"gestión de la diabetes". Algunas de las aplicaciones con mayor interés para los usuarios y que se recogen en el TFG de Lara, son los siguientes:

- **Fundación para la diabetes**: aplicación para iOS y Android en la que se puede consultar recetas para diabéticos, realizar el test Frindisk y leer las ultimas noticias respecto a la diabetes.
- **Diario de la diabetes**: Aplicación para iOS y Android que permite registrar todos los datos importantes de la vida diaria con la diabetes.
- **Diguan**: Juego para iOS y Android con el que los niños se familiarizan con la enfermedad.
- **Diabetes**: La aplicación realiza un seguimiento de todos los aspectos del tratamiento de la enfermedad y proporciona informes detallados, gráficos y estadísticas para compartir a través del correo electrónico.
- **Diabetes hipoglicemia**: Guía para resolver una hipoglucemia de forma adecuada, hasta la resolución del problema.

- **Socialdiabetes**: Aplicación para iOS y Android. En ella se puede introducir las comidas, el nivel de glucosa o la cantidad de insulina. Tiene la opción de consultar las gráficas para seguir una evolución de forma visual.

Conclusiones y Líneas de trabajo futuras

En esta sección expondremos las conclusiones extraídas a lo largo y final del proyecto, así como posibles líneas de trabajo futuras.

7.1. Conclusiones

Este apartado lo dividiremos en varias partes, cada una con una conclusión.

Dinámica del proyecto

El desarrollo del proyecto ha sido muy satisfactorio. Estoy seguro de que esta etapa de la carrera ha sido de gran utilidad para mi en cuanto a adquisición de experiencia profesional y personal. Por otra parte, el hecho de que el proyecto se haya desarrollado con la participación de diferentes actores que podría encontrar en cualquier proyecto del mercado laboral, creo que es una experiencia muy importante y que ha sido de gran ayuda para ver cómo se trabaja en este tipo de proyectos. En cuanto a los objetivos conseguidos, cabe mencionar que se han conseguido cumplir con todos a excepción de uno de ellos: Ampliar la lista de alimentos. A pesar de ello, se ha conseguido una aplicación final muy mejorada y útil con respecto a la versión anterior.

Por otra parte, debo agradecer a Raúl Marticorena Sánchez, tutor del proyecto, quien me ha marcado desde el principio y durante cada semana todas las tareas que debía realizar así como los errores o fallos que se presentaban durante el desarrollo del proyecto.

Técnica y lenguaje de desarrollo

Se ha utilizado el entorno de desarrollo Android Studio, el cual, tiene Java como lenguaje base de programación.

Tras finalizar el proyecto, puedo concluir que, Android Studio ofrece un gran número de alternativas al desarrollador, y que a bien seguro continuaré estudiando y trabajando con ello.

Aprendizaje

El desconocimiento del entorno de desarrollo Android me ha obligado a estudiar nuevas herramientas y considerar los pros y contras cuando se dispone de diferentes opciones para realizar una misma tarea, lo cual ha sido de gran importancia para mí como desarrollador.

Comparativa con la versión anterior

Para hacer una comparativa de nuestra aplicación final con la versión anterior, mencionaremos las distintas funcionalidades que disponían cada una.

UBUDiabetes 1.0

- Calcular Bolo Corrector: Esta opción traía ciertos fallos. Para ciertos casos de pruebas la aplicación no obtenía el resultado correcto.
- Registro de glucemias: OK.
- Incidencias: OK.
- Historial de glucemias: Limitado. Sólo mostraba gráficamente los valores de la última semana.
- Ajustes: Limitado. Sólo se podía realizar ajustes en el perfil de usuario.
- Interactividad usuario-aplicación: Media.
- Interfaz: Estática.

UBUDiabetes 2.0

- Calcular Bolo Corrector: OK.

- Registro de glucemias: OK.
- Incidencias: OK.
- Historial de glucemias: OK.
- Consultar registros: OK.
- Ajustes: OK. Se pueden realizar 3 configuraciones: Perfil, Copia de seguridad y Liberar espacio.
- Interactividad usuario-aplicación: Alta.
- Pruebas: OK. Pruebas unitarias y pruebas de interacción de usuario (UI).
- Interfaz: Dinámica.

7.2. Líneas de trabajo futuras

Como mencionó Raúl MArticorena Sánchez en una de las reuniones, este proyecto servirá de referencia para futuras mejoras.

A partir de esta segunda versión funcional hay que pensar en las opciones que se pueden plantear de cara a una nueva versión, funciones que no se pudieron añadir por falta de tiempo, así como mejoras que se pueden llevar a cabo. Alguna de estas funciones o mejoras coinciden con las mencionadas en el TFG de Mario López Jiménez [3]:

- Añadir la posibilidad de exportar los datos almacenados en la base de datos SQLite de la aplicación en algún formato que pueda resultar útil para el usuario.
- Ampliar la base de datos de alimentos.
- Adaptar la aplicación para tablets aprovechando el diseño de la misma en *fragments*.
- Incluir un ratio de insulina diferente para cada momento del día.
- Tener en cuenta otros factores que puedan influir al paciente, estrés, periodo premenstrual etc.

Bibliografía

[1] Lara Bartolomé Casado. Salud electrónica y diabetes. elementos para el diseño de una aplicación informática (app) para jóvenes diabéticos tipo i., 2015.

[2] Bruno Martín Gómez. Evaluación de la eficacia, efectividad y usabilidad de la aplicación par móviles ubudiabetes., 2017.

[3] Mario López Jiménez. Ubudiabetes: Aplicación de control de diabetes en dispositivos móviles., 2016.

[4] Unknown. Android developer. https://developer.android.com/?hl=es-419, Unknown.

[5] Unknown. Diabetes. https://medlineplus.gov/spanish/diabetes.html, Unknown.

[6] Unknown. Github. https://guides.github.com/activities/hello-world/, Unknown.

[7] Unknown. Indice glucémico. http://www.diabetes.org/es/alimentos-y-actividad-fisica/alimentos/que-voy-a-comer/comprension-de-los-carbohidratos/indice-glucemico-y-diabetes.html, Unknown.

[8] Unknown. Sqlite browser. http://sqlitebrowser.org/, Unknown.

[9] Unknown. Tabla hc. http://www.fundaciondiabetes.org/upload/publicaciones_ficheros/71/TABLAHC.pdf/, Unknown.

[10] Lars Vogel. Android sqlite database and contentprovider-tutorial. *Java, Eclipse, Android and Web programming tutorials*, 8, 2010.

[11] Wikipedia. Android sdk — wikipedia, la enciclopedia libre, 2017. [Internet; descargado 18-mayo-2018].

[12] Wikipedia. Junit — wikipedia, la enciclopedia libre, 2017. [Internet; descargado 21-mayo-2018].

[13] Wikipedia. Texmaker — wikipedia, la enciclopedia libre, 2017. [Internet; descargado 21-mayo-2018].

[14] Wikipedia. Adobe photoshop — wikipedia, la enciclopedia libre, 2018. [Internet; descargado 21-mayo-2018].

[15] Wikipedia. Android — wikipedia, la enciclopedia libre, 2018. [Internet; descargado 17-mayo-2018].

[16] Wikipedia. Android studio — wikipedia, la enciclopedia libre, 2018. [Internet; descargado 21-mayo-2018].

[17] Wikipedia. Aplicación móvil — wikipedia, la enciclopedia libre, 2018. [Internet; descargado 17-mayo-2018].

[18] Wikipedia. Espresso (framework) — wikipedia, la enciclopedia libre, 2018. [Internet; descargado 21-mayo-2018].

[19] Wikipedia. Git — wikipedia, la enciclopedia libre, 2018. [Internet; descargado 21-mayo-2018].

[20] Wikipedia. Glúcido — wikipedia, la enciclopedia libre, 2018. [Internet; descargado 26-abril-2018].

[21] Wikipedia. Interacción persona-computadora — wikipedia, la enciclopedia libre, 2018. [Internet; descargado 7-junio-2018].

[22] Wikipedia. Java (lenguaje de programación) — wikipedia, la enciclopedia libre, 2018. [Internet; descargado 18-mayo-2018].

[23] Wikipedia. Latex — wikipedia, la enciclopedia libre, 2018. [Internet; descargado 21-mayo-2018].

[24] Wikipedia. Resistencia a la insulina — wikipedia, la enciclopedia libre, 2018. [Internet; descargado 26-abril-2018].