UNIVERSIDAD
REY JUAN CARLOS

GRADO EN INGENIERÍA TELEMÁTICA

Curso Académico 2018/2019

Trabajo Fin de Grado

Aplicación móvil para el control del nivel de glucosa en personas diabéticas

Autor : Venancio Romero Moreno

Tutor : Enrique Soriano Salvador

Trabajo Fin de Grado

Grado en Ingeniería Telemática

Autor : Venancio Romero Moreno

Tutor : Enrique Soriano Salvador

La defensa del presente Proyecto Fin de grado se realizó el día de de 2018, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 2018

Dedicado a

mi hija Laura.

Agradecimientos

Quiero dar las gracias a mis padres por su ilusión, por la enseñanza de que los sueños se consiguen con esfuerzo, y por el empuje en todos los momentos difíciles. A mi hermano por esas miles de horas al teléfono aguantándome, dándome su opinión o simplemente haciéndome sentir mejor, y en especial a mi mujer Tamara y a mis hijas Laura y Alba, por el increible día a día que me regalan y por su infinita paciencia.

También quiero dar las gracias a mis compañeros Manuel, Adrián, Javier y Elena por haber estado todas esas horas explicándome lo que no lograba entender, dándome su punto de vista, o empujándome hacia mis objetivos.

Resumen

Este proyecto se centra en el desarrollo de una aplicación móvil que mediante la lectura de un sensor es capaz de monitorizar el nivel de glucosa existente en el líquido intersticial de un paciente con diabetes.

De las opciones en el desarrollo de aplicaciones móviles disponibles actualmente se ha elegido utilizar Android ya que dispone de una cuota de mercado superior al 80 %, posee una comunidad de desarrolladores muy amplia, sus teléfonos resultan mucho más económicos que los de sus competidores, y la imposibilidad de la utilización de ciertos sensores que eran necesarios en este proyecto en otras plataformas.

En la realización del software se ha utilizado Android Studio y a pesar de las distintas alternativas para desarrollar una aplicación para Android, el lenguaje de programación elegido ha sido Java, existiendo además la posibilidad de realizarse en Kotlin o C++.

Este sistema ayudará a las personas diabéticas a mejorar su día a día, ya que al tener monitorizados sus niveles de glucosa de manera constante podrán actuar en consecuencia y así evitar problemas derivados de hiper e hipoglucemias.

Índice general

1. Introducción **1**

1.1. ¿Qué es la diabetes? 1

1.2. ¿Qué tipos de diabetes hay? 2

1.2.1. Diabetes mellitus tipo 1 2

1.2.2. Diabetes mellitus tipo 2 2

1.2.3. Diabetes Lada 3

1.2.4. Diabetes gestacional 3

1.3. La importancia de la tecnología 3

1.4. Estructura de la memoria 4

2. Objetivos **5**

2.1. Descripción del problema 5

2.2. Objetivo del proyecto 6

2.3. Requisitos del proyecto 6

3. Estado del arte **8**

3.1. Android 8

3.2. Aplicaciones móviles similares existentes 8

3.2.1. FreeStyle LibreLink 9

3.2.2. Glimp 10

3.2.3. xDrip 11

3.3. Sensores de glucosa 11

3.3.1. Free Style Libre 11

3.3.2. Dexcom G6 12

3.3.3. Guardian Connect 13

4. Desarrollo del proyecto **14**

4.1. Conceptos y tecnologías básicas 14

4.1.1. Conceptos básicos 14

4.1.2. Tecnologías 15

4.1.2.1. NFC 15

4.2. Software 18

4.2.1. Librerías y frameworks utilizados 20

4.2.2. ¿Qué es una licencias de software? 21

4.2.3. Confidencialidad de los datos 21

4.2.3.1. Formato de compresión ZIP 21

4.2.3.2. Cifrado simétrico 21

4.3. Sensor de glucosa 22

4.3.1. Análisis de los datos 29

4.3.2. Interpretación de las medidas 33

4.3.3. Distribución de los datos obtenidos con respecto a la medida real 34

4.4. Desarrollo de la aplicación móvil 38

4.4.1. Términos básicos de una aplicación Android 39

4.4.2. Estructura de directorios de un proyecto Android 40

4.4.3. Diseño de la base de datos 41

4.4.4. Diagrama de flujo 42

4.4.5. Código del proyecto 43

4.4.6. Descripción de las pantallas 43

5. Conclusiones **48**

5.1. Lineas futuras 49

Bibliografía **50**

Índice de figuras

2.1. Glucómetros con distintos niveles 6

3.1. Búsqueda en google play 9
3.2. Aplicación móvil librelink de Abbott diabetes care inc. 10
3.3. Aplicación móvil glimp 10
3.4. Aplicación móvil xdrip 11
3.5. Sensor Free Style Libre de Abbott Diabetes Care Inc. 12
3.6. Sensor Dexcom G6 fabricado por la empresa Dexcom Inc. 12
3.7. Guardian Connect fabricado por la empresa Medtronic 13

4.1. Representación hexadecimal de la palabra de 32 bits 6B34AFF0 en Big Endian y Little Endian. 15
4.2. Comandos NFC del estándar ISO-15693 16
4.3. Capas presentes en la tecnología NFC 17
4.4. Interior del sensor Free Style Libre 22
4.5. Interior del sensor Free Style Libre 22
4.6. Circuito integrado RF430TAL152H 23
4.7. Esquema general de la solución para la recopilar datos. 24
4.8. Volcado de la memoria del sensor. 29
4.9. Evolución minuto a minuto. 29
4.10. Evolución minuto a minuto. 30
4.11. Primer bloque de los datos del sensor. 31
4.12. Segundo bloque de los datos del sensor. 32
4.13. Tercer bloque de los datos del sensor. 32

4.14. Subbloque de 6 bytes. 33
4.15. Procedimiento para obtener las medidas. 34
4.16. Distribución de la glucosa real. 35
4.17. Distribución de la glucosa medida con el sensor. 35
4.18. Distribución de la temp. 36
4.19. Medida real vs Medida sin tratamiento. 36
4.20. Esquema de la solución 39
4.21. Diagrama de flujo de la aplicación 42
4.22. Pantalla principal, pestaña home 44
4.23. Detalle de la pestaña home 44
4.24. Pantalla principal, pesataña chart. 45
4.25. Pantalla exportación de datos 46
4.26. Capacidad de enviar y compartir el fichero 46
4.27. Detalle del fichero csv con datos reales 47

Capítulo 1

Introducción

En el año 1873 el arqueólogo y novelista alemán George Ebers [1] encontró, cerca de las ruinas de Luxor, un papiro atribuido a un eminente médico de la época que data del año 1.553 a. C.. En él se relata la existencia de enfermos que adelgazan, tienen hambre continuamente, orinan en abundancia y se sienten atormentados por una enorme sed. Aconseja un tratamiento a base de grasa de ternera, cerveza, hojas de menta y sangre de hipopótamo.

Cinco siglos antes de la era cristiana, Ayur Veda escribió el libro más importante de cirugía de la antigua India, Sushruta Samhita. En él se puede encontrar otra referencia en la que se describe una extraña enfermedad, propia de las personas pudientes, obesos, que comen mucho dulce y arroz y cuya característica más peculiar consiste en tener la orina pegajosa, con sabor a miel, que atrae fuertemente a las hormigas, por lo que la llamaron "madhumeha" (orina de miel). De esta manera el padre de la medicina hindú, describió la diabetes mellitus, denominándola "enfermedad de los ricos", llegando incluso a diferenciar una diabetes que se daba en los jóvenes que conducía a la muerte y otra que se daba en personas de una cierta edad. También explica que esta enfermedad habitualmente afectaba a varios miembros dentro de una misma familia.

1.1. ¿Qué es la diabetes?

La diabetes mellitus es una enfermedad que se caracteriza por una elevación crónica de los niveles de glucosa en sangre. Esto ocurre cuando el cuerpo es incapaz de producir insulina o tiene un déficit de la misma.

La exposición crónica a niveles altos de glucosa en la sangre es una de las principales causas

de insuficiencia renal, pérdida visual y otra serie de daños. También predispone a la enfermedad arterial, y a menudo viene acompañada de hipertensión, desórdenes del perfil lipídico y obesidad.

Un nivel bajo de glucosa en sangre o hipoglucemia puede ocasionar graves repercusiones como convulsiones, pérdida de la consciencia, e incluso si no es controlada a tiempo, el coma.

1.2. ¿Qué tipos de diabetes hay?

1.2.1. Diabetes mellitus tipo 1

La diabetes tipo 1 es una enfermedad metabólica, autoinmune y crónica que es más frecuente en niños y adolescentes, aunque puede producirse a cualquier edad. Es producida debido a una respuesta autoinmune que lleva a una destrucción de las células beta del páncreas, que son las encargadas de producir insulina. Esta hormona se encarga de que la glucosa entre al interior de las células y no se acumule en el torrente sanguíneo. Al no poder producir insulina, las personas diabéticas deben aportarla de manera externa. En el caso de la diabetes tipo 1 se suministra mediante inyecciones de insulina.

1.2.2. Diabetes mellitus tipo 2

Este tipo es el que afecta al 90-95 % de las personas con diabetes. Es la forma mas común en los adultos y ancianos, aunque se está empezando a describir su aparición en niños y adolescentes. Habitualmente se diagnostica por encima de los 30-40 años. Normalmente se diagnostica después de muchos años de padecerla sin saberlo, y se caracteriza porque aunque el páncreas produce insulina, el organismo no es capaz de utilizarla de modo eficaz. Hay un componente hereditario importante. En este tipo de diabetes, se produce algo de insulina pero no es suficiente o no funciona de forma apropiada para llevar glucosa hasta el interior de las células. Este tipo de diabetes se puede controlar con un plan de alimentación y actividad física, perdiendo peso si existe sobrepeso u obesidad. En algunos casos puede ser necesario la utilización de pastillas, y a lo largo de los años, muchos pacientes pueden acabar precisando inyectarse insulina para controlar su azúcar.

1.2.3. Diabetes Lada

Diabetes autoinmune latente en adultos [2], también conocida como diabetes 1.5, inicialmente tiene las características de la diabetes tipo 2 pero finalmente el paciente deja de segregar insulina. Se calcula que el 10 % de los diabéticos tipo 2 la padece y no ha sido detectada debido al desconocimiento de su médico. Otro factor que puede llevar a una mala clasificación del paciente es que las células beta del páncreas pueden tardar años en dejar de segregar insulina. Este tipo es considerado como la diabetes tipo 1 de los adultos ya que es causada por una respuesta autoinmune del organismo.

1.2.4. Diabetes gestacional

Conocida como del embarazo, se da en mujeres embarazadas que no tenían previamente diabetes, y desaparece poco después del parto, aunque muchas de estas mujeres con los años pueden llegar a desarrollar diabetes.

1.3. La importancia de la tecnología

Los últimos años han estado caracterizados por un gran avance en la tecnología. Este desarrollo ha favorecido un gran cambio en el ámbito sanitario, facilitando mucho las tareas de control y supervisión de los pacientes. Actualmente, gracias a la tecnología y los desarrollos que se crean para el sector, se puede realizar un control continuo de constantes vitales, nivel de glucosa en sangre, o incluso realizar una operación con paciente y médicos separados por cientos de kilómetros.

En este trabajo nos centraremos en el uso de nuevas tecnologías para realizar un control continuo de glucosa en diabéticos. Su utilización evitará las complicaciones de una mala supervisión continuada de azúcar en sangre tales como problemas visuales, úlceras en la piel, daño en los riñones, etc. Más concretamente se desarrollará una aplicación móvil Android que mediante el uso de tecnologia NFC leerá un sensor que el paciente llevará adherido a su cuerpo.

1.4. Estructura de la memoria

Este trabajo se divide en cinco partes. En la primera, después de una breve introducción histórica y descripción de los tipos de diabetes existentes, se intenta introducir al lector en el contexto de la necesidad de realizar un control continuado de niveles de azúcar mediante el uso de nuevas tecnologías.

En el siguiente capítulo se describe el problema, enunciando los objetivos propuestos y requisitos que se buscan cubrir con la implementación.

El tercer apartado se adentra en el estado del arte para comprender la situación actual de la distintas soluciones y sensores que ya existen.

En la cuarta parte se describe el desarrollo completo de la aplicación comenzando por la investigación necesaria para decodificar los datos que almacena el sensor, y el breve estudio estadístico realizado para intentar minimizar el error en la predicción del valor de la glucosa.

Para finalizar, en el quinto capítulo se exponen las conclusiones obtenidas y se enumeran posibles trabajos futuros.

Capítulo 2

Objetivos

Tras haber introducido el contexto en el que se desarrollará este proyecto, en esta sección fijaremos los objetivos y expondremos los requisitos que presenta la solución.

2.1. Descripción del problema

Para garantizar su salud, personas diabéticas deben controlar constantemente su alimentación y nivel de glucosa en sangre. Diariamente, el número de medidas que se realiza un paciente puede superar la veintena, sobre todo en casos de diabetes mellitus tipo 1. Los datos recopilados y sus hábitos alimenticios deben ser almacenados para que el facultativo asignado al control de la enfermedad pueda comprobar la evolución. Gracias a los *smartphones* y tecnologías actuales, esta necesidad puede ser cubierta con creces con el desarrollo de una aplicación móvil que interprete un sensor de glucosa y almacene los niveles para que más tarde puedan ser exportados o enviados al médico.

Los aspectos a cubrir y por los cuáles se realizó este trabajo son:

- Lectura de sensor de glucosa.
- Almacenaje de medidas de glucosa.
- Generación de gráfica con la evolución de la glucosa en sangre.
- Exportación de datos.

2.2. Objetivo del proyecto

El objetivo principal de este trabajo es **crear una aplicación móvil que sea capaz de leer un sensor de glucosa adherido a la piel de una persona con la capacidad de almacenaje, visualización y exportación de los datos almacenados**.

Para lograr nuestro objetivo, dividiremos el trabajo en dos partes;

- Sensor de glucosa: en este apartado se presentará el sensor utilizado, y se realizará una tarea de ingeniería inversa para poder encontrar la manera de decodificar los datos que el transductor emite.
- Aplicación móvil: desarrollaremos una aplicación móvil que implemente el código necesario para poder decodificar el sensor y que cubra todo los requisitos del trabajo.

2.3. Requisitos del proyecto

A continuación enumeraremos los requisitos marcados para alcanzar el objetivo del proyecto:

- Lectura de sensor con tecnología NFC: es necesario que la aplicación sea capaz de interpretar la información almacenada en el sensor.
- Requisitos relacionados con los datos: no existe ningún sensor sin un cierto grado de incertidumbre, todos disponen de un error. Nuestro caso no es una excepción y trabajaremos para que los datos sean lo más precisos posibles.

Figura 2.1: Glucómetros con distintos niveles

- Presentación de las medidas: la aplicación dispondrá de una tabla en la que poder visualizar de manera ágil los valores y fecha. También proporcionará una gráfica en la que se podrá ver la evolución temporal de manera clara.

- Capacidad de exportación: la aplicación debe ser capaz de exportar las medidas para que sean examinadas por el médico del usuario. Además, el fichero generado debe estar protegido mediante una contraseña.

Capítulo 3

Estado del arte

En este capítulo se menciona el sistema operativo elegido como soporte de la solución y se dan a conocer las aplicaciones móviles existentes relacionadas con el objetivo de este proyecto.

3.1. Android

Android [8] es un sistema operativo basado en el núcleo Linux. Fue diseñado principalmente para dispositivos móviles con pantalla táctil, como *smartphones*, tabletas, relojes inteligentes, televisores y automóviles. Inicialmente fue desarrollado por Android Inc., empresa que Google respaldó económicamente y más tarde, en 2005 compró. Android fue presentado en 2007 junto la fundación del Open Handset Alliance (un consorcio de compañías de hardware, software y telecomunicaciones) para avanzar en los estándares abiertos de los dispositivos móviles. El primer móvil con el sistema operativo Android fue el HTC Dream y se vendió en octubre de 2008. Android es el sistema operativo móvil más utilizado del mundo, con una cuota de mercado superior al 80 % en el año 2017 [3] y más de 2.000 millones de teléfonos, muy por encima del sistema iOS de la empresa Apple [4].

3.2. Aplicaciones móviles similares existentes

Realizando una búsqueda en la tienda de aplicaciones para el sistema Android de Google se puede encontrar multitud de aplicaciones móviles mediante las cuales se puede llevar un control más o menos exhaustivo de la rutina diaria del paciente. La mayoría de ellas requieren que se

introduzca los datos de forma manual, hecho que puede conducir a medidas erróneas, olvidos o incluso al abandono del seguimiento electrónico.

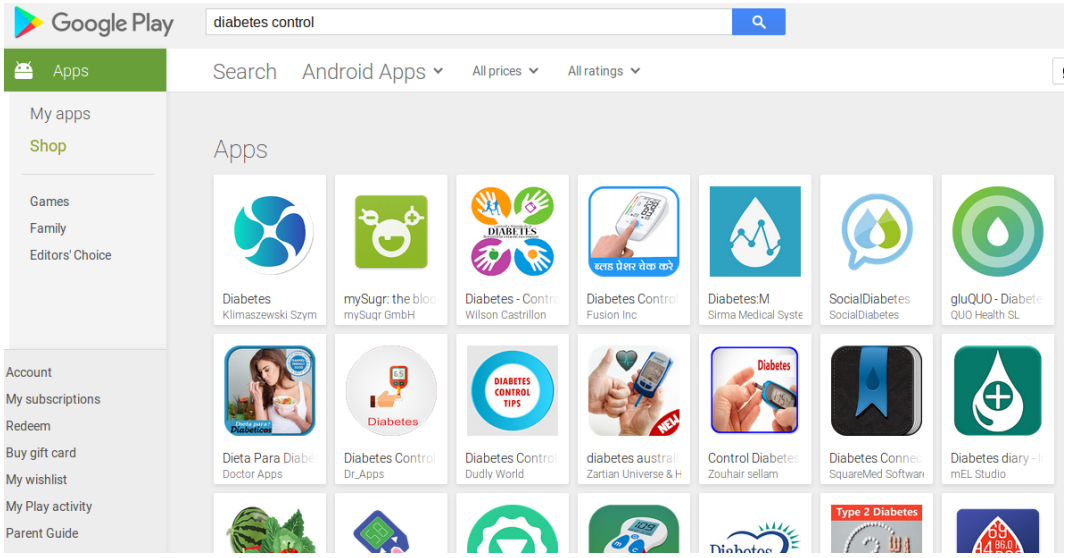

Figura 3.1: Búsqueda en google play

A continuación describiremos las aplicaciones móviles más relevantes que cumplen una función similar a la propuesta en este trabajo.

3.2.1. FreeStyle LibreLink

Aplicación móvil oficial de la compañia farmacéutica Abbott Diabetes Care Inc. [9] para medir su propio sensor de glucosa Free Style libre [10].

Con esta aplicación se puede leer el nivel de glucosa y la tendencia registrada en el sensor con teléfonos que dispongan de tecnología NFC. Estas medidas son almacenadas en el teléfono para su posterior visualización, y tiene la posibilidad de agregar notas a las medidas. El proceso de lectura requiere de la vinculación y activación del sensor, hecho que iniciará un contador de tiempo mediante el cuál será utilizado para determinar el final de la vida del sensor. Requiere el registro gratuito en la plataforma de Abbott Diabetes Care Inc.

Figura 3.2: Aplicación móvil librelink de Abbott diabetes care inc.

3.2.2. Glimp

Aplicación móvil no oficial desarrollada por la grupo italiano CTAPP Software que es capaz de leer el sensor Free Style Libre de Abbott. No requiere la activación ni emparejado con el sensor. Esta aplicación tiene la capacidad de almacenar las unidades de hidratos de carbono consumidos entre otras cosas. Su interfaz puede llegar a ser poco intuitiva para personas con conocimientos poco profundos en la materia.

Figura 3.3: Aplicación móvil glimp

3.2.3. xDrip

Proyecto desarrollado por varios colaboradores y el grupo Nightscout [11]. Es capaz de recoger información de sensores de varios fabricantes, y de sistemas de lectura y transmisión de datos via Bluetooth. Además tiene la capacidad de almacenar los datos sobre la ingesta de alimentos y ejercicio físico. Esta aplicación móvil es muy completa y está en constante evolución.

Figura 3.4: Aplicación móvil xdrip

3.3. Sensores de glucosa

En esta sección se describirán distintos módelos de sensores de glucosa que existen actualmente.

3.3.1. Free Style Libre

Sensor fabricado por la compañia Abbott Diabetes Care Inc. que es capaz de medir la glucosa disuelta en el líquido intersticial existente entre las celulas de la piel y almacenarlo durante un periodo en su memoria interna. La lectura de glucemia se realiza inoculando un filamento de manera hipodérmica, el cual entra en contacto con las capas inferiores de la piel donde se realiza la medición de la glucosa circulante en los tejidos fuera del torrente sanguíneo. El sen-

sor utiliza tecnología NFC y tiene una vida fijada por el fabricante de catorce días. Pasado esta fecha después de su activación cualquier lector oficial dejará de funcionar.

Figura 3.5: Sensor Free Style Libre de Abbott Diabetes Care Inc.

3.3.2. Dexcom G6

Sensor fabricado por la empresa Dexcom Inc. [12], indicado para adultos y niños desde los dos años, registra las medidas tomadas en el líquido interticial de la piel del paciente, para ser transmitido mediante Bluetooth a un dispositivo externo como un *smartphone*, o lector. El funcionamiento es similar al anterior, dispone de un filamento que es introducido en el paciente.

Figura 3.6: Sensor Dexcom G6 fabricado por la empresa Dexcom Inc.

3.3.3. Guardian Connect

Sensor fabricado por la empresa Medtronic [13]. El funcionamiento es similar a los anteriores, dispone de un punzón que es clavado en la piel, que mide la glucosa y la envía cada cinco minutos a traves de Bluetooth a un dispositivo externo. La aplicación oficial de este sensor dispone de la posibilidad de predecir con antelación (entre 10 y 60 minutos) valores atípicos de glucosa en sangre, seguimiento de insulina y carbohidratos. La vida del sensor es de seis dias.

Figura 3.7: Guardian Connect fabricado por la empresa Medtronic

Capítulo 4

Desarrollo del proyecto

En este apartado describiremos la realización del proyecto. Previamente trataremos conceptos que se han utilizado en el desarrollo, seguidamente analizaremos el sensor de glucosa, su interpretación e intimidades, y a continuación nos centraremos en la aplicación móvil.

4.1. Conceptos y tecnologías básicas

En esta sección se mencionarán conceptos y tecnologias básicas que serán utilizadas a lo largo del desarrollo, conviene entenderlos bien, ya que disponemos de casi nula información sobre el funcionamiento del sensor de glucosa y realizaremos un trabajo de ingeniería inversa sobre él. Los conceptos y tecnologías básicas son:

4.1.1. Conceptos básicos

- **Byte**: tiene varias definiciones, según la RAE [14] podemos definir byte como unidad de información compuesta generalmente de ocho bits, también se puede decir que es la unidad fundamental de datos y memoria.

- **Endianness**: término inglés que designa el formato en el que se almacenan los datos de más de un byte en un circuito o componente electrónico. Existen dos posibles configuraciones **big endian** que consiste en representar los bytes en el orden natural y **little endian** que consiste en representar en el orden inverso al natural.

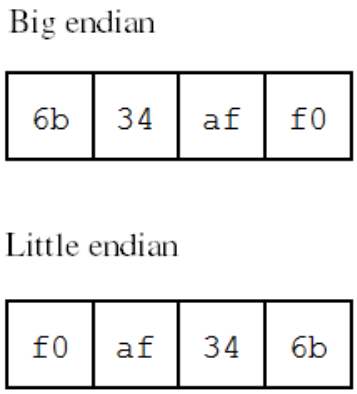

Figura 4.1: Representación hexadecimal de la palabra de 32 bits 6B34AFF0 en Big Endian y Little Endian.

- **Sistema hexadecimal**: sistema de numeración posicional que utiliza como base el número 16. Sus números están representados por los 10 primeros dígitos de la numeración decimal, y el intervalo que va del número 10 al 15 están representados por letras del alfabeto que van desde la A a la F.

- **HTTP**: acrónimo de protocolo de transferencia de hiperTexto. Es un sencillo protocolo cliente-servidor que articula los intercambios de información entre los clientes Web y los servidores HTTP. La especificación completa del protocolo HTTP 1.0 está recogida en el RFC 1945. Fue propuesto por Tim Berners-Lee, atendiendo a las necesidades de un sistema global de distribución de información como el World Wide Web.

4.1.2. Tecnologías

4.1.2.1. NFC

Acrónimo de **Near Field Communication** [7]. Es una tecnología que permite comunicarse entre sí a dispositivos separados por una distancia inferior a 20 centímetros. Esta tecnología funciona en el espectro de radiofrecuencia de 13,56 megahercios.

Las etiquetas NFC suelen almacenar entre 96 y 512 bytes de datos y la transferencia puede alcanzar una velocidad de hasta 848 kilobits por segundo, velocidad suficiente para mover

pequeños paquetes de información prácticamente de forma instantánea.

La tecnología NFC se divide en tres partes:

1.- Software: en esta capa se realiza la interpretación y composición de los datos que la aplicación va a enviar o recibir, para que la capa inferior los trate.

2.- Tags / Peer-to-peer: en esta parte se realiza la comunicación lógica con el tag. Mediante el uso de comandos se le indica al otro dispositivo que acción se va realizar. Estos comandos se indican mediante el valor de un byte en el bloque inicial de comunicación y están definidos por el estándar. Las acciones principales pueden ser leer/escribir un bloque,y leer/escribir varios bloques.

Cmd	Command Type	Tag Type	Function	Option Flag
01	Mandatory	All	Inventory	No
02	Mandatory	All	Stay Quiet	No
20	Optional	Std, Pro, Plus	Read Single Block	No
21	Optional	Std, Pro, Plus	Write Single Block	Yes
22	Optional	Std, Pro, Plus	Lock Block	Yes
23	Optional	Plus only	Read Multiple Blocks	No

Figura 4.2: Comandos NFC del estándar ISO-15693

3.- Hardware: capa hardware compuesta por controladores de radio, bus de comunicaciones internos y elementos hardware que son los encargados convertir las datos en señales de radiofrecuencia, y añadir robustez a la comunicación.

En la siguiente imagen se puede ver una representación sobre las capas de la tecnología:

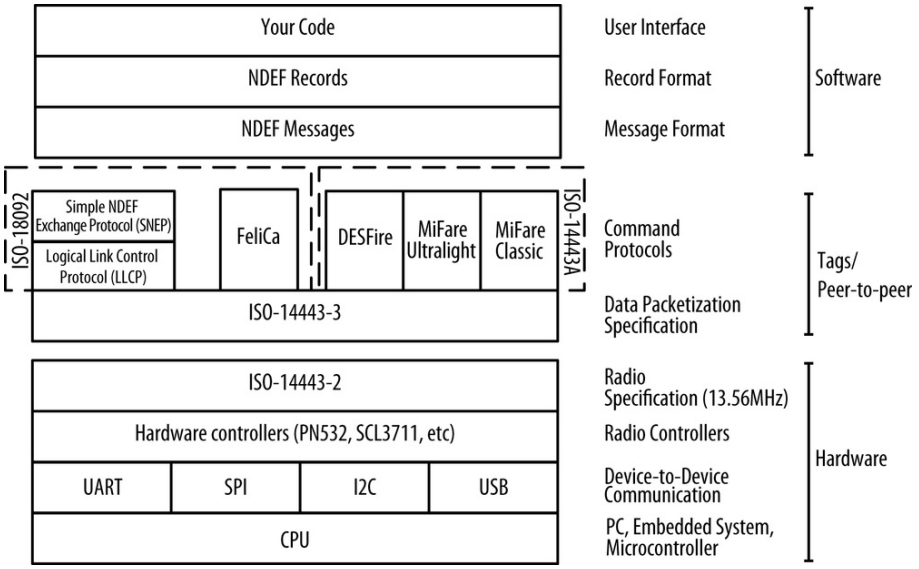

Figura 4.3: Capas presentes en la tecnología NFC

Las normas ISO 14443 e ISO 15693 son los estándares principales que rigen la tecnología NFC. Dentro de ellas se desarrollaron varias versiones que son utilizadas para varias aplicaciones.

Tecnología	Estándar
NFCA	ISO 14443-3A
NFCB	ISO 14443-3B
NFCF	JIS 6319-4
NFCV	ISO 15693
ISODEP	ISO 14443-4
NDEF	NDEF

Las transmisiones de datos se producen de forma “half duplex”. Esto quiere decir que el mismo canal se utiliza tanto para transmitir como para recibir información. Además, para evitar que dos dispositivos transmitan a la vez, existe un protocolo de diálogo mediante el cual el emisor, antes de transmitir, verifica que no hay otro dispositivo transmitiendo en ese momento.

La conexión se realiza entre dos dispositivos de manera automática en el momento en el que se “encuentran” a unos 5 cm el uno del otro, a pesar de que esta distancia puede variar según diferentes factores (tipo de emisor/receptor, temperatura, aislantes, etc.). La distancia máxima

obligatoria para operar se fija en 20 cms. Esta distancia máxima se debe a la necesidad de cubrir la seguridad de la información, y que ésta no se propague más allá.

El estándar NFC define dos tipos de dispositivos: el iniciador y el receptor. Como sus nombres indican, el iniciador es el dispositivo que arranca la comunicación y que controla los intercambios de datos. El receptor es el dispositivo que responde a las peticiones del iniciador.

El estándar NFC define dos modos de operación:

- Modo de comunicación activo: ambos dispositivos generan una señal de radio frecuencia (RF) para transmitir los datos.
- Modo de comunicación pasivo: sólo un dispositivo genera el campo RF. El segundo dispositivo, el pasivo, actúa como receptor, y usa una técnica llamada "modulación de carga" para transferir información de vuelta al dispositivo activo o iniciador.

4.2. Software

A continuación describiremos básicamente el distinto tipo de software que se ha usado:

- **Java 1.8**: lenguaje de progamación de propósito general orientado a objetos. Fue desarrollado con la intención de que los desarrolladores escribieran el código una sóla vez y lo ejecutaran en cualquier dispositivo (concepto conocido en inglés como WORA, o "write once, run anywhere"). Hemos elegido este lenguaje porque permite desarrollar aplicaciones móviles del sistema Android y estamos muy familiarizados con él, ya que ha sido utilizado en distintas asignaturas del grado. Java dispone de licencia GNU GPL/Java Community Process.
- **SQLite**: con licencia de dominio público, es una biblioteca *open source* escrita en C que implementa un sistema de gestión de base de datos transaccional SQL autocontenido, sin servidor ni configurarión. Incluye transacciones de base de datos atómicas, consistencia de base de datos, aislamiento y durabilidad (ACID). Los principios ACID son:
    - **Atomicidad (Atomicity)**: se dice que una operación es atómica cuando es imposible para otra parte de un sistema encontrar pasos intermedios. Si esta operación consiste en una serie de pasos, todos ellos ocurren o ninguno. Por ejemplo, en el caso de una

transacción bancaria o se ejecuta tanto el depósito y la deducción o ninguna acción es realizada. Es una característica de los sistemas transaccionales.

- **Consistencia (Consistency)**: es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto, se ejecutan aquellas operaciones que no van a romper la reglas y directrices de integridad de la base de datos.
- **Aislamiento (Isolation)**: es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que la realización de dos transacciones sobre la misma información nunca generará ningún tipo de error.
- **Permanencia (Durability)**: es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema.

- **Android Studio 3.2**: con licencia Freeware + Source code, es un IDE, o entorno de desarrollo integrado oficial de Android, lanzado en 2014 por Google, para el desarrollo de aplicaciones móviles. Tiene la capacidad de poder trabajar en varias lenguajes de programación entre ellos Java y ha sido empleado en el desarrollo del objetivo de este proyecto.

- **R**: es una implementación de software libre del lenguaje S [18]. Se trata de uno de los lenguajes de programación más utilizados en investigación por la comunidad estadística, siendo muy popular en el campo de la minería de datos, la investigación biomédica, la bioinformática y las matemáticas financieras. Hemos utilizado este lenguaje para intentar disminuir la incertidumbre inherente del sensor de glucosa. R está compuesto de muchos paquetes y tiene varias licencias GPL, BSD, GNU, MIT, Creative commons, Artistic License.

- **RStudio 1.0.153**: es un IDE para el lenguaje de programación R, en él hemos realizado los scripts necesarios para realizar gráficas, y para describir matemáticamente el comportamiento físico del sensor para así lograr un error menor en el resultado del dato. La licencia de este ide es GNU Affero General Public License v3.

- **Python 2.7**: lenguaje interpretado, multiparadigma,y de tipado dinámico. Fue creado con la filosofía de hacer hincapié en una sintáxis que favorezca un código legible. Ha sido empleado en la realización de un script muy básico que recogía las medidas del sensor

sin ningún tipo de tratamiento, para su posterior análisis. Licencia Python software foundation.

- **Gimp 2.8**: software libre dedicado a la edición de imágenes digitales. Esta publicado bajo licencia GNU y ha sido utilizado para pequeños cambios en las imágenes de empleadas en la memoria. La licencia de este software es GPL.

- **LaTeX**: sistema de composición de textos, orientado a la creación de documentos escritos que presenten una alta calidad tipográfica. Por sus características y posibilidades, se usa de forma especialmente intensa en la generación de artículos y libros científicos que incluyen, entre otros elementos, expresiones matemáticas. Lo hemos usado en la redacción de esta memoria. LaTeXutiliza su propia licencia denominada LaTeXproject public License.

- **Microsoft Visual code 1.25.1**: Code es un IDE que soporta múltiples lenguajes de programación tales como C++, C#, Java, Visal Basic, Java, Python,LaTeX, ha sido empleado para escribir esta memoria en LaTeX. Code utiliza licencia MIT.

- **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 estas tienen un gran número de archivos de código. En el proyecto ha sido utilizado para registrar los avances en el desarrollo de la aplicación y escritura de esta memoria. Git utiliza la licencia GPL.

4.2.1. Librerías y frameworks utilizados

Aquí haremos una breve mención a las librerias y frameworks utilizados en el proyecto:

- **Flask 1.0** es un framework minimalista escrito en python que permite crear aplicaciones web rápidamente y con un mínimo de lineas de código. Dispone de licencia BSD.

- **MPAndroidChart v3.0.3** librería para realizar visualizaciones de datos en Android. Dispone de licencia Apache 2.0.

- **Mzip-Android** librería para Android que permite comprimir y descomprimir ficheros y carpetas en formatos de compresión zip y rar.

4.2.2. ¿Qué es una licencias de software?

Es conveniente conocer qué es una licencia de software, ya que en varios puntos del trabajo mencionaremos que el software es open source o que dispone de cierta licencia.

Una licencia de software es un contrato entre el licenciante (autor/titular de los derechos de explotación/distribución) y el licenciatario (usuario consumidor, profesional o empresa) del programa informático, para utilizarlo cumpliendo una serie de términos y condiciones establecidas dentro de sus cláusulas, es decir, es un conjunto de permisos que un desarrollador le puede otorgar a un usuario en los que tiene la posibilidad de distribuir, usar o modificar el producto bajo una licencia determinada. Además se suelen definir los plazos de duración, el territorio donde se aplica la licencia (ya que la licencia se soporta en las leyes particulares de cada país o región), entre otros.

4.2.3. Confidencialidad de los datos

Al tratarse de datos médicos, es importante que el usuario pueda cifrar el contenido de los ficheros que exporta. Para cumplir este requisito y no añadir excesiva complejidad, se ha optado por utilizar un sistema de compresión tipo *zip* que soporta el cifrado simétrico de los ficheros comprimidos.

4.2.3.1. Formato de compresión ZIP

Formato de compresión sin pérdida muy utilizado para la compresión de datos [19]. Fue diseñado por Phil Katz [20] y soporta un sistema de cifrado simétrico basado en clave única.

4.2.3.2. Cifrado simétrico

Método criptográfico en el cual se usa una misma clave para cifrar y descifrar los datos. Aunque este sistema de cifrado es débil ante ataques de texto plano, ataque de diccionario y fuerza bruta nos da un mecanismo simple para cifrar el contenido del fichero.

4.3. Sensor de glucosa

Para la realización del proyecto se ha usado el sensor Free Style Libre. Lo único que sabíamos de este sensor es que utiliza tecnología NFC, tiene una vida de catorce días, y es capaz de medir la glucosa en el líquido intersticial del paciente, por lo que tuvimos que descubrir como funciona internamente, y la estructura utilizada para almacenar los datos.

Nuestro trabajo comenzó indagando en el interior del sensor, para ello mediante unos alicates de corte abrimos el sensor y examinamos el interior.

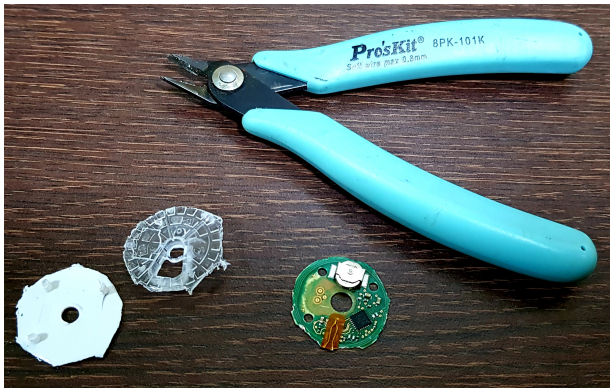

Figura 4.4: Interior del sensor Free Style Libre

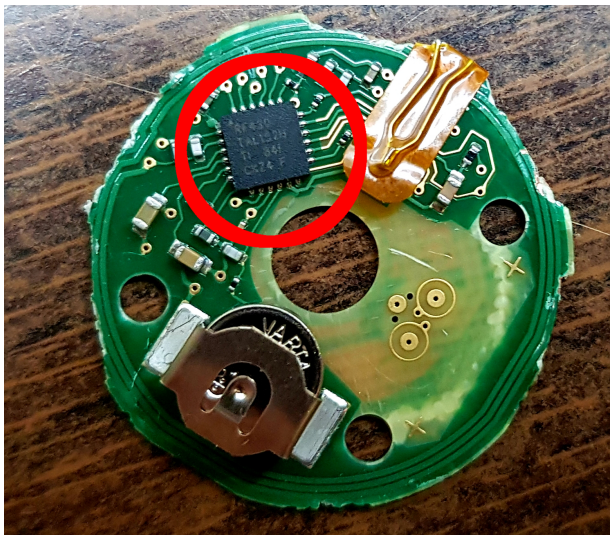

Figura 4.5: Interior del sensor Free Style Libre

Tras realizar una inspección ocular del interior, pudimos observar que el circuito contenía

una pila, varios componentes smd, y un integrado RF430TAL152H que fue clave en nuestra investigación. Mediante la referencia del circuito integrado obtuvimos que el fabricante es Texas Instruments Inc. y que la documentación esta disponible en su página web [21]. Las características más importantes del dispositivos son:

- Utiliza formato little endian para almacenar sus bytes.
- Dispone de un sensor de temperatura.
- Tiene 8 KB de memoria ROM.
- Tiene 4 KB de SRAM.
- Dispone de 2 KB de memoria FRAM.

Figura 4.6: Circuito integrado RF430TAL152H

La siguiente tarea que realizamos fue crear una aplicación móvil Android con la que pudimos averiguar el estándar NFC utilizado y leer los datos del sensor. Una vez conseguido, realizamos un pequeño servidor HTTP que exponía un puerto mediante el cuál la aplicación podía enviar los datos para su posterior análisis sin ningún tipo de tratamiento. En la siguiente imagen ilustramos el esquema general de la solución para recopilar los datos en esta fase inicial.

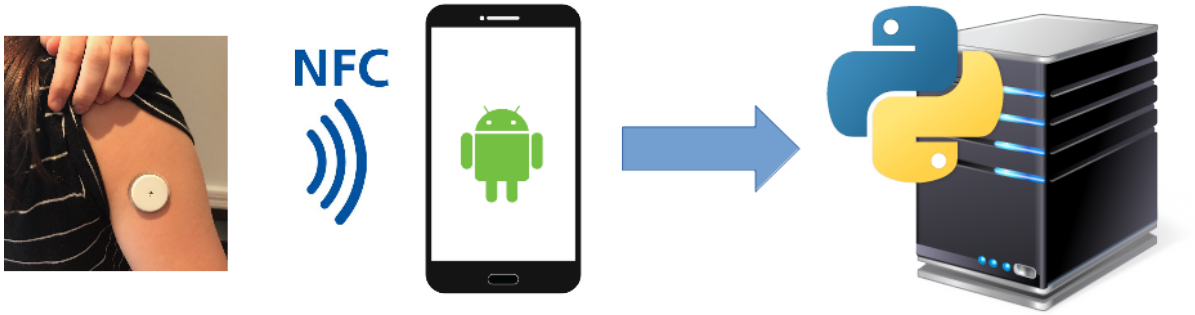

Figura 4.7: Esquema general de la solución para la recopilar datos.

El código de la aplicación móvil Android utilizado en esta fase consta de tres clases Java:

- **NfcCommands.java**: como mencionamos anteriormente, el estándar NFC recoge una serie de comandos que se utilizan para indicar qué acción se realizará. Esta clase contiene las constantes que el estándar define para comunicarse con tags NFC. La implementación ha sido la siguiente:

```
package com.venoty.nfc_reader;

public final class NfcCommands {
    public static final byte INVENTORY             = (byte) 0x01;
    public static final byte STAY_QUIET            = (byte) 0x02;
    public static final byte READ_SINGLE_BLOCK     = (byte) 0X20;
    public static final byte WRITE_SINGLE_BLOCK    = (byte) 0x21;
    public static final byte LOCK_BLOCK            = (byte) 0x22;
    public static final byte READ_MULTIPLES_BLOCKS = (byte) 0x23;
}

```

- **Properties.java**: esta clase es utilizada para manejar las propiedades definidas en un fichero de texto plano externo al código. El código utilizado ha sido el siguiente.

```
package com.venoty.nfc_reader;

import android.content.Context;
import java.io.IOException;

public class Properties {
    private java.util.Properties props;

```

```
    public Properties(Context ctx, String filename) throws
    IOException {
        props = new java.util.Properties();
        props.load(ctx.getAssets().open(filename));
    }
    public String getProperty(String key) {
        return props.getProperty(key);
    }
}

```

- **MainActivity.java**: esta clase contiene toda la lógica de la aplicación.

```
package com.venoty.nfc_reader;

import android.content.Intent;
import android.nfc.Tag;
import android.nfc.tech.NfcV;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;

import org.bouncycastle.util.encoders.Hex;
import java.io.IOException;
import java.util.Arrays;

import okhttp3.MediaType;
import okhttp3.OkHttpClient;
import okhttp3.Request.Builder;
import okhttp3.RequestBody;


public class MainActivity extends AppCompatActivity {

    // Constants
    private final String TAG              = "NFC-Reader";
    private final String PROP_FILENAME    = "app.properties";
    private final String TECH_DISC_ACTION =
    "android.nfc.action.TECH_DISCOVERED";
    private final String NAME_TAG         = "android.nfc.extra.TAG";
    private final int    NUM_MAX_BLOCKS   = 40;

    // Global variables
    private Properties properties;
    private Intent     myIntent;
```

```
32
33      @Override
34      protected void onCreate(Bundle savedInstanceState) {
35
36          super.onCreate(savedInstanceState);
37          setContentView(R.layout.activity_main);
38
39          try {
40              properties = new Properties(this,PROP_FILENAME);
41
42              if (TECH_DISC_ACTION.equals(getIntent().getAction())) {
43                  myIntent = getIntent();
44                  Thread t = new NFCsensor();
45                  t.start();
46                  t.join();
47              }
48          } catch (Exception e) {
49
50              Log.d(this.TAG, Log.getStackTraceString(e));
51          }
52          finish();
53      }
54
55      public class NFCsensor extends Thread {
56
57          public void run() {
58
59              byte[] cmd;
60              byte[] block;
61
62              try {
63                  Tag tag = (Tag) myIntent.getParcelableExtra(NAME_TAG);
64                  NfcV nfcv = NfcV.get(tag);
65                  StringBuilder stringBuilder = new StringBuilder();
66
67                  nfcv.connect();
68                  cmd = new byte[]{ (byte) 0,
69                                    NfcCommands.READ_SINGLE_BLOCK,
70                                    (byte) 0};
71
72                  for (int i = 3 ; i <= NUM_MAX_BLOCKS; i++) {
73                      cmd[2] = (byte) i;
74                      block  = nfcv.transceive(cmd);
75                      block  = Arrays.copyOfRange(block, 1, 8);
76
```

```
                    // Concatenate string hexadecimal representation
                    stringBuilder.append(i * 8)
                            .append(" - ")
                            .append(Hex.toHexString(block)
                                    .replaceAll("(.{2})", "$1 "))
                            .append("\n");
                }

                Log.d(TAG, stringBuilder.toString());
                sendData(stringBuilder.toString());

            } catch (Exception e) {
                Log.e(TAG, Log.getStackTraceString(e));
            }
        }

        private void sendData(String data) throws IOException {

            MediaType mediaType = MediaType.parse("text/plain");
            RequestBody request = RequestBody.create(mediaType, data);
            String server       = properties.getProperty("server");

            new OkHttpClient().newCall( new Builder().url(server)
                                                      .post(request)
                                                      .build())
                                                      .execute();
        }
    }
}
```

El anterior código maneja un evento producido en el sistema, si se trata de un evento NFC, entrará en un ciclo que realiza las siguientes acciones:

- Envía al sensor un comando que le indica que se va a leer un bloque de la memoria.
- Convierte los datos en crudo a una cadena de texto con la representación hexadecimal de los bytes leidos.
- Se concatena la cadena de texto con el resultado de todas las iteraciones anteriores.

Acabado el bucle, la cadena resultante se envía en texto plano a un servidor HTTP.

El servidor almacenará los datos en ficheros separados identificados por la fecha y hora del instante de tiempo en el que se recibió para su posterior análisis. El código utilizado ha sido el

siguiente:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from flask import Flask
from flask import request
import datetime

FOLDER = "glucose_data/"
app    = Flask(__name__)

@app.route('/', methods=['POST'])
def writeData():
    now       = datetime.datetime.now()
    name_file = str(now.strftime("%Y-%m-%d-%H:%M:%S")) + ".data"
    full_path = FOLDER + name_file
    f         = open(full_path,"w")

    f.write(request.data)
    f.close()

    return ""

if __name__ == '__main__':
    app.run(host='0.0.0.0')
```

Con la ayuda de los códigos anteriores fuimos capaces de averiguar que la versión NFC que utiliza es **NFCV** y lograr ver el volcado de la memoria del sensor.

Con todo funcionando estábamos preparados para tomar medidas en difentes días, a distintas horas, minutos y con varios sensores.

```
0   - 2D B7 C8 15 03 00 00 00
8   - 00 00 00 00 00 00 00 00
16  - 00 00 00 00 00 00 00 00
24  - 9A 6B 0E 16 D8 05 C8 68
32  - 1A 80 DB 05 C8 74 5A 80
40  - D6 05 C8 48 5A 80 D5 05
48  - C8 3C 5A 80 CF 05 C8 4C
56  - 5A 80 D5 05 C8 2C 5A 80
64  - D5 05 C8 14 5A 80 D2 05
72  - C8 28 1A 80 D1 05 C8 10
80  - 5A 80 D4 05 C8 18 5A 80
....
....
```

Figura 4.8: Volcado de la memoria del sensor.

4.3.1. Análisis de los datos

Desconocer completamente la estructura de los datos obligó a realizar muchas medidas en distintos intervalos.

Inicialmente tomamos medidas del sensor cada minuto y comprobamos la diferencia. A continuación se muestra una imagen en la que se puede observar que los bytes 24,25 y 26 de las tres medidas y además los bytes 40-45 de la segunda medida con respecto a la primera y los bytes 46-51 con respecto a la segunda cambiaron:

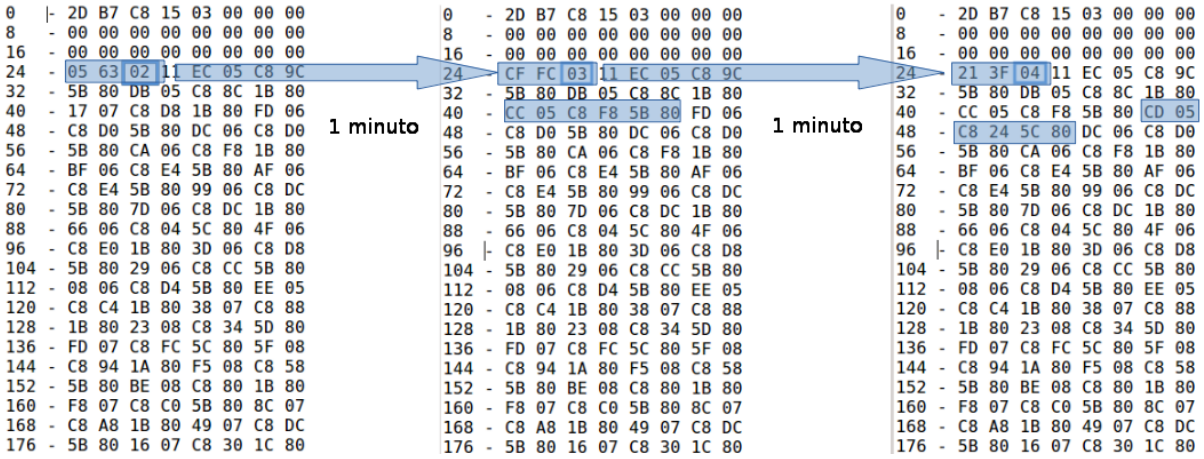

Figura 4.9: Evolución minuto a minuto.

Además, tras varias medidas se comprueba que el byte número 26 pasa del valor 0x0F a 0x00, y que el valor almacenado indica donde se realiza el siguiente cambio de memoria, empezando a contar a partir del byte 28 en bloques de 6 bytes. También que el byte 27 incrementa su valor cada ciclo de 15 minutos. Los bytes 24 y 25 no logramos averiguar su significado.

```
0   - 2D B7 C8 15 03 00 00 00
8   - 00 00 00 00 00 00 00 00
16  - 00 00 00 00 00 00 00 00
24  - 1E 5E 0F 11 EC 05 C8 9C
32  - 5B 80 DB 05 C8 8C 1B 80
40  - CC 05 C8 F8 5B 80 CD 05
48  - C8 24 5C 80 BF 05 C8 28
56  - 5C 80 AF 05 C8 38 5C 80
64  - 9D 05 88 4A 9C 80 91 05
72  - C8 68 5C 80 82 05 C8 78
80  - 5C 80 73 05 C8 80 5C 80
88  - 67 05 C8 90 5C 80 5E 05
96  - C8 94 5C 80 55 05 C8 8C
104 - 5C 80 4F 05 C8 90 5C 80
112 - 4E 05 C8 90 5C 80 EE 05
120 - C8 C4 1B 80 38 07 C8 88
128 - 1B 80 23 08 C8 34 5D 80
136 - FD 07 C8 FC 5C 80 5F 08
144 - C8 94 1A 80 F5 08 C8 58
152 - 5B 80 BE 08 C8 80 1B 80
160 - F8 07 C8 C0 5B 80 8C 07
168 - C8 A8 1B 80 49 07 C8 DC
176 - 5B 80 16 07 C8 30 1C 80
```

```
0   - 2D B7 C8 15 03 00 00 00
8   - 00 00 00 00 00 00 00 00
16  - 00 00 00 00 00 00 00 00
24  - CD 86 00 12 EC 05 C8 9C
32  - 5B 80 DB 05 C8 8C 1B 80
40  - CC 05 C8 F8 5B 80 CD 05
48  - C8 24 5C 80 BF 05 C8 28
56  - 5C 80 AF 05 C8 38 5C 80
64  - 9D 05 88 4A 9C 80 91 05
72  - C8 68 5C 80 82 05 C8 78
80  - 5C 80 73 05 C8 80 5C 80
88  - 67 05 C8 90 5C 80 5E 05
96  - C8 94 5C 80 55 05 C8 8C
104 - 5C 80 4F 05 C8 90 5C 80
112 - 4E 05 C8 90 5C 80 4B 05
120 - C8 98 5C 80 38 07 C8 88
128 - 1B 80 23 08 C8 34 5D 80
136 - FD 07 C8 FC 5C 80 5F 08
144 - C8 94 1A 80 F5 08 C8 58
152 - 5B 80 BE 08 C8 80 1B 80
160 - F8 07 C8 C0 5B 80 8C 07
168 - C8 A8 1B 80 49 07 C8 DC
176 - 5B 80 16 07 C8 30 1C 80
```

Figura 4.10: Evolución minuto a minuto.

Tras lograr averiguar que el byte 27 cambiaba cada 15 minutos, continuamos realizando medidas durante varias horas y comprobamos que el sensor almacenaba las medidas de las últimas 8 horas en intervalos de 15 minutos.

Como conclusión, después de haber analizado más de 1000 medidas con distintos sensores, horas y días, podemos afirmar que la memoria del sensor está dividida aparentemente en tres zonas:

- Primer bloque: consta de 28 bytes, en él se almacena el estado actual del sensor, y punteros a la medidas actuales.

Byte	Descripción
0 - 3	———
4	Estado del sensor
5 - 23	Padding
24-25	———
26	Puntero a la medida actual (15 minutos).
27	Puntero a la medida actual (8 horas).

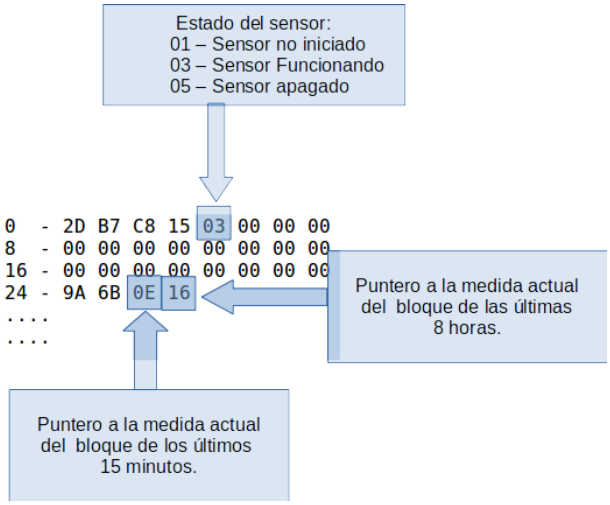

Figura 4.11: Primer bloque de los datos del sensor.

- Segundo bloque: consta de 96 bytes y almacena las medidas de glucosa de los últimos quince minutos. Cada medida es representada por un subbloque de seis bytes con formato little endian.

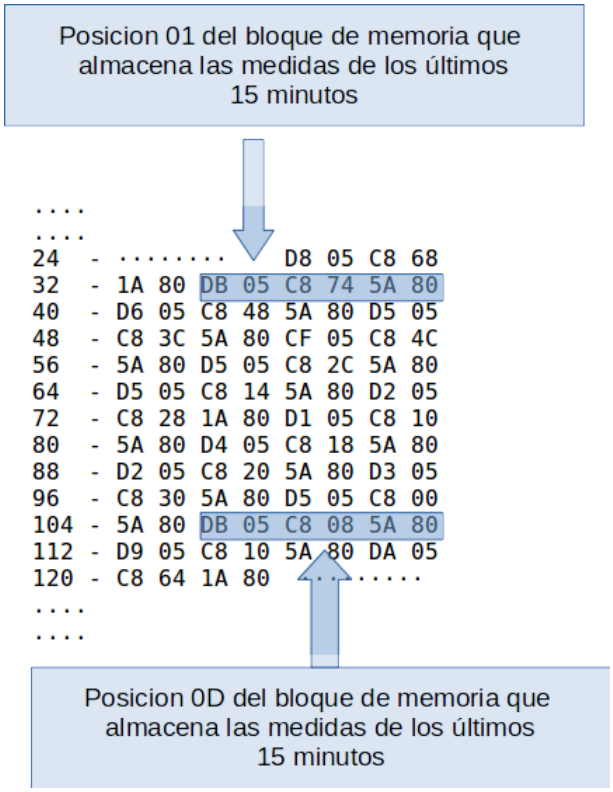

Figura 4.12: Segundo bloque de los datos del sensor.

- Tercer bloque: consta de 192 bytes y almacena las medidas de glucosa de las últimas ocho horas con una granularidad de quince minutos. Al igual que el bloque anterior, las medidas se almacenan en subbloques de seis bytes con formato little endian.

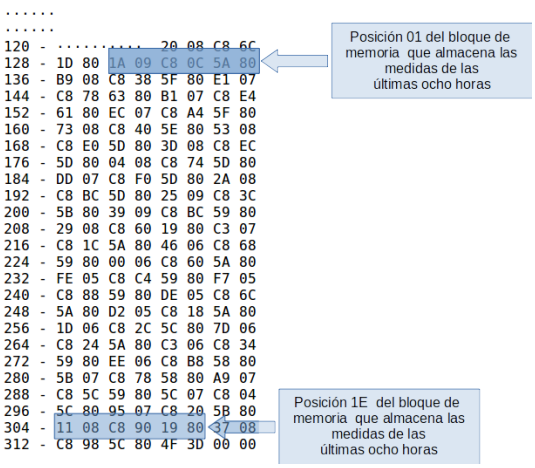

Figura 4.13: Tercer bloque de los datos del sensor.

4.3.2. Interpretación de las medidas

Las medidas se almacenan en grupos de seis bytes en formato little endian. Los primeros dos bytes almacenan el valor de la glucosa en ”crudo” y el cuarto y el quinto guardan el valor de la temperatura en ese instante. Desconocemos el significado del tercer y sexto byte, pero observamos que son diferentes en cada sensor y que no cambiaron en las observaciones que realizamos. Interpretaremos que son separadores de las medidas y que carecen de valor.

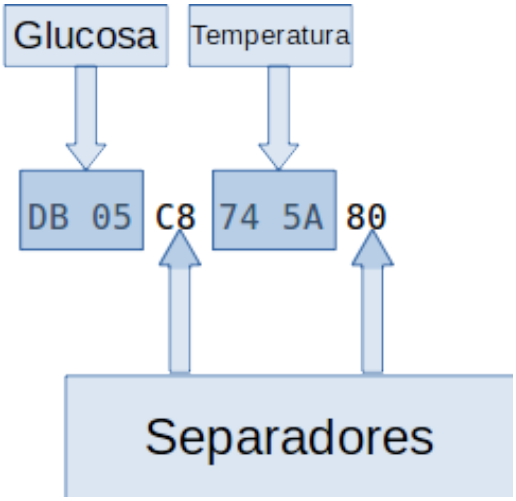

Figura 4.14: Subbloque de 6 bytes.

En el *datasheet* del circuito integrado RF430 [5] menciona que el integrado dispone de un sensor de temperatura y que utiliza un conversor analógico-digital sigma-delta de 14 bits [6]. Por lo tanto, a la medida de temperatura almacenada será necesario aplicarle una máscara de 14 bits (0011 1111 1111 1111), cuya representación hexadecimal es 0x3FFF.

Para obtener los valores, es necesario realizar el siguiente procedimiento:

- Glucosa: convertir el formato de little endian a big endian y pasar el número de base 16 a base 10.
- Temperatura: pasar el formato de little endian a big endian, realizar una máscara con 0x3FFF y finalmente el resultado convertirlo a número decimal.

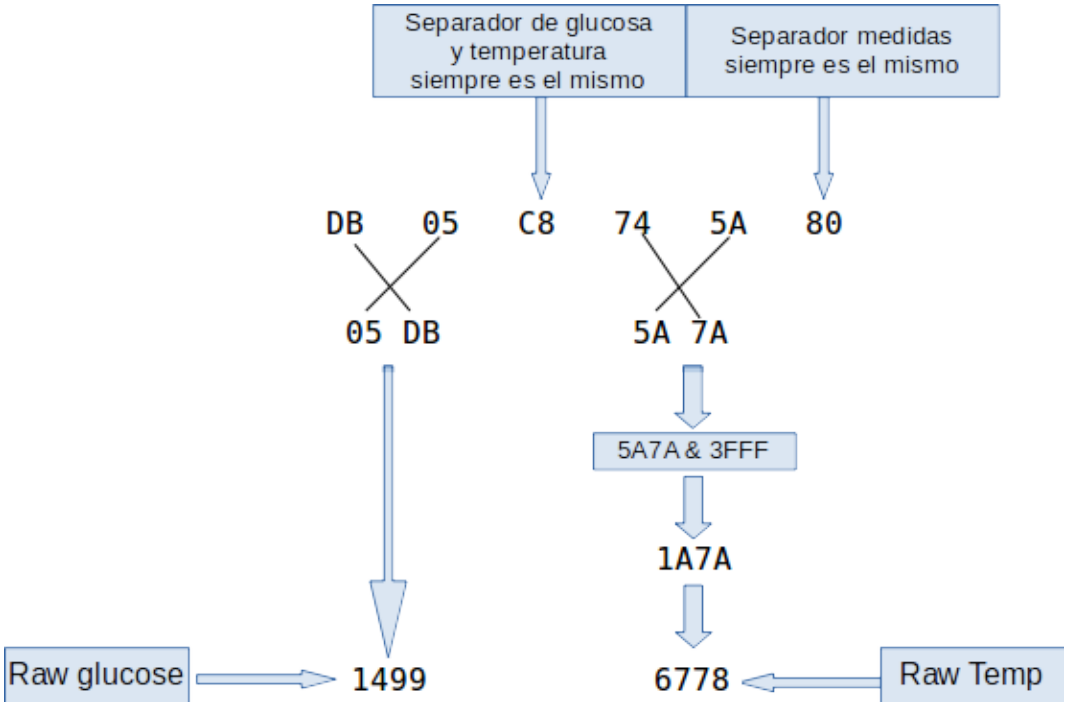

Figura 4.15: Procedimiento para obtener las medidas.

En el ejemplo de la imagen anterior se puede observar que el sensor guarda en memoria un número que representa a la glucosa real (los valores normales de una persona diabética suelen estar entre 80 y 200 miligramos por decilitro). Esta relación no tiene por qué ser lineal, es decir, un incremento en la glucosa del paciente, puede no reflejar un aumento igual o proporcional a la magnitud en el dato almacenado.

Además, suponemos que la temperatura es anotada para ser usada como un offset aplicado al resultado final.

Estas conclusiones, obligaron a analizar estadísticamente los resultados y compararlos con medidas realizadas con un medidor de glucosa en sangre y un paciente.

4.3.3. Distribución de los datos obtenidos con respecto a la medida real

En esta subsección analizamos la distribución que sigue el sensor Free Style Libre. Este análisis requiere de la toma de medidas con el sensor mencionado y además del nivel medido con un medidor capilar en sangre. Después de un periodo de treinta días se obtuvo aproximadamente cien medidas que utilizamos para obtener la distribución individual de cada variable,

la relación existente entre la medida real y la que no tiene ningún tipo de tratamiento. Para esta tarea utilizamos el lenguaje R y el IDE RStudio.

El valor de la glucosa real tiene una densidad que sigue una distribución gaussiana [22] con media 146, como valor mínimo hemos obtenido 68 y máximo 333.

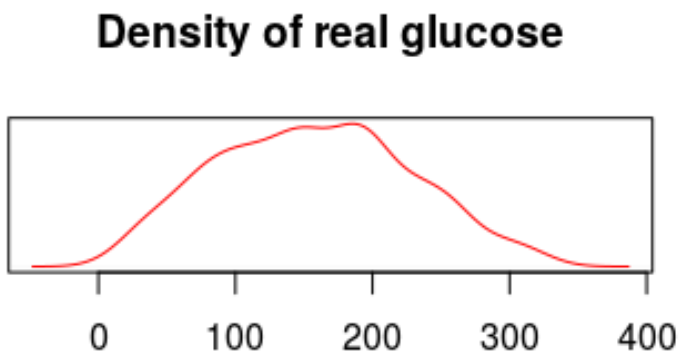

Figura 4.16: Distribución de la glucosa real.

El valor de la glucosa sin transformación tiene una densidad que sigue una distribución gaussiana con media 1712, como valor mínimo hemos obtenido 545 y máximo 3109.

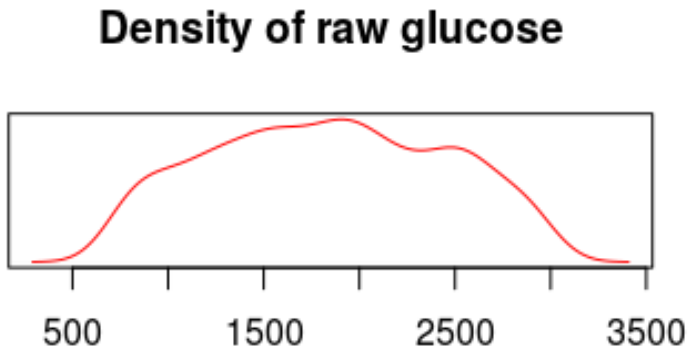

Figura 4.17: Distribución de la glucosa medida con el sensor.

El valor de la temperatura tiene una densidad que sigue una distribución gaussiana con media 6888, como valor mínimo hemos obtenido 6601 y máximo 7203.

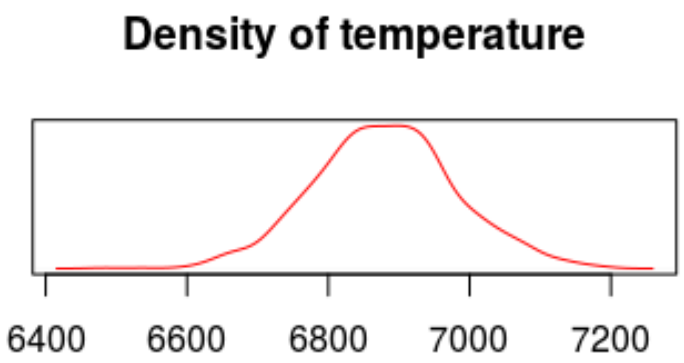

Figura 4.18: Distribución de la temp.

A continuación representamos la medida real contra la medida que no tiene ningún tipo de tratamiento.

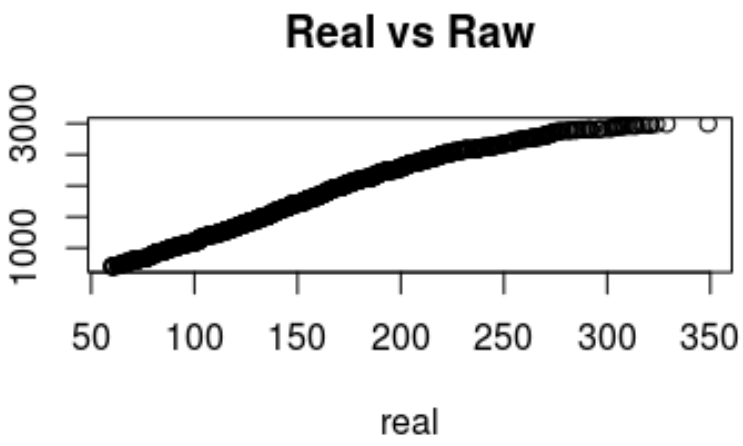

Figura 4.19: Medida real vs Medida sin tratamiento.

Además calculamos la correlación de la glucosa y el valor almacenado:

```
cor(real,raw)
[1] 0.9303742
```

A la vista de la gráfica anterior y el coeficiente de correlación de Pearson [22] podemos afirmar que existe una relación casi lineal entre el valor real y la medida anotada por el sensor, por ello, realizamos una regresión lineal generalizada, junto con la temperatura que determinará el valor estimado de la glucosa real con el error mínimo. El código que hemos utilizado fue el siguiente:

```
data  <- read.csv("/home/venoty/desktop/recta_regresion")
model <- glm(real ~ raw + temp, data=data,family = "gaussian")
summary(model)


Call:
glm(formula = real ~ raw + temp, family = "gaussian", data = data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-21.347   -9.693   -7.921    6.217   30.626

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -86.553645 153.577111  -0.564 0.593459
raw           0.125729   0.019861   6.330 0.000727 ***
temp          0.008764   0.019937   0.440 0.675644
---
(Dispersion parameter for gaussian family taken to be 389.9428)

    Null deviance: 17968.2  on 8  degrees of freedom
Residual deviance:  2339.7  on 6  degrees of freedom
AIC: 83.586

Number of Fisher Scoring iterations: 2
```

En el resultado de la regresión observamos que la medida de glucosa registrada por el sensor tiene un p-valor que indica una significatividad elevada con respecto a la medida real.

Por el contrario la temperatura tiene un peso en la ecuación despreciable, no aporta nada de información al resultado. No obstante no eliminaremos esta variable, por entender de que no disponemos de suficientes registros y además que tras varias observaciones se ha comprobado que cuando este valor es elevado existe una mayor varianza en el resultado de la glucosa.

Después de este breve estudio estadístico pudimos concluir que con los datos de los que disponemos, la medida de glucosa real se puede hallar con la siguiente fórmula.

$$\textbf{Glucosa} = 0{,}125729\ x\ \textbf{rawGlucose} + 0{,}008764\ x\ \textbf{temperature} - 86{,}553645 \quad (4.1)$$

En este punto hemos logrado entender como se almacenan los datos en la memoria del dispositivo de medida, además de describir la ecuación que rige la predicción del dato real con el mínimo error.

4.4. Desarrollo de la aplicación móvil

El diseño de la aplicación, ha ido sufriendo modificaciones a lo largo del desarrollo. Uno de los cambios más significativos fue que inicialmente se ideó como un sistema de alertas y control continuo de glucosa. El teléfono se sostenía mediante un brazalete deportivo al brazo del paciente y la aplicación disponía de un bucle que cada cierto tiempo leía el sensor de glucosa y disparaba alertas cuando los niveles estuvieran fuera de los valores establecidos como normales. Tras varias pruebas se llegó a la conclusión de que no era una buena solución ya que el paciente se sentía incómodo en su día a día y era probable que el teléfono dejara de leer el sensor.

Como resultado de la etapa inicial aprendimos que la aplicación debía ser clara y sencilla, para que ello facilitara al máximo el control diario, histórico de la glucosa y suponiera un uso continuado.

Con la aplicación desarrollada el usuario debe leer el sensor adherido a la piel cuando desee. Esta acción mostrará en el teléfono la pantalla inicial con la medida actual, las medidas de los últimos siete y quince minutos y una tabla con las lecturas anteriores. La aplicación además cuenta con una pantalla para la visualización de la evolución temporal de la glucosa y otra para la exportación de los datos.

Figura 4.20: Esquema de la solución

4.4.1. Términos básicos de una aplicación Android

Para entender el desarrollo de una aplicación móvil Android es necesario tener unos conocimientos previos básicos sobre terminología utilizada. Los principales conceptos son los siguientes:

- **Activivity**: se puede decir que cada pantalla de la aplicación es un activity. Cada una está formada por una parte lógica y otra parte gráfica. La parte lógica es el fichero de código que se implementa para poder manipular, interactuar y definir el comportamiento del activity. La parte gráfica es un fichero xml en el que se define el aspecto visual y componentes que tendrá la pantalla.

- **Fragment**: porción de la interfaz del usuario que puede añadirse o quitarse de forma independiente al resto de contenidos de la actividad. Puede verse como un contendor que

podemos utilizar en cada activity. La aplicación dispone de tres fragmentos, en el primero se muestra las medidas tomadas de forma manual, se muestra en la activity principal, el segundo muestra la gráfica de evolución de la glucosa de los útlimos catorce días, también se muestra en la pantalla principal y, el tercer *fragment* se utiliza dos veces en la pantalla de exportación de datos para mostrar un calendario.

- **Intent**: puede verse como un evento. Es una intención de realizar una acción o comunicar un mensaje. Suele utilizarse como mecanismo para invocar componentes externos o internos a nuestro aplicación. En nuestro proyecto *intents* son usados para detectar que el sensor NFC está cerca, exportar los datos o para lanzar activities.

- **layout**: es una estructura visual para una interfaz de usuario. Se declara mediante un fichero con extensión xml.

4.4.2. Estructura de directorios de un proyecto Android

Todos los proyectos Android tienen un árbol de directorios que permite mantener una estructura, a continuación la describiremos:

- **AndroidManifest.xml**: configurar las opciones básicas de nuestra aplicación.Describe los componentes de la aplicación, activities, services... y permisos necesarios.

- **Java**: en este directorio encontraremos los ficheros.java con el código fuente. Ubicación: /src/main/java/".

- **res**: directorio que contiene recursos, además de él cuelgan los siguientes:

  - **anim**: archivos XML que definen animaciones.Directorio: /src/main/res/anim".
  - **drawable**: en este directorio se encuentran las imágenes en formato PNG o JPEG (Google tiene preferencia por las imágenes PNG). Su ubicación es: /src/main/res/drawable".
  - **layout**: aquí se encuentran todos los layouts de nuestro proyecto. Ya hemos visto qué es un layout y qué tipos de layouts existen. Se ubican en el directorio "/src/main/res/layout".
  - **menu**: archivos XML que definen las plantillas de los menus del proyecto /src/main/res/menu".

- **mipmap**: contiene los iconos de la aplicación con sus diferentes resoluciones. Anteriormente se encontraban en el directorio drawable, pero en las últimas versiones de Android Studio han sido movidos al directorio mipmap. Ubicación ”src/main/res/mipmap/”.
- **raw**: en esta carpeta ubicamos los archivos multimedia descomprimidos, de esta manera Android sabe que no debe procesarlos más tarde. Se encuentra en el directorio /src/main/res/raw”.
- **values**: archivos XML que definen valores constantes.Se encuentra en ”src/main/res/values”
  - **dimens**: este directorio será el contenga los tipos de resoluciones que podrá tomar nuestra aplicación una vez compilada. Ubicación /src/main/res/values/dimens.xml”.
  - **string.xml**: este XML contiene las cadenas de texto de nuestra aplicación. Más adelante veremos cómo nos facilita la tarea de traducir nuestra aplicación a otros idiomas cómodamente. Ubicación /src/main/res/values/strings.xml”.
  - **styles.xml**: en este fichero XML se definen los estilos de nuestro proyecto. Ubicación /src/main/res/values/styles.xml”.
  - **color.xml**: aquí se definen los colores que usaremos en el diseño de nuestra aplicación. Ubicación /src/main/res/values/color.xml”.

- **gradle**: por último encontramos los ficheros gradle de Android Studio. Los ficheros de compilación gradle de Android Studio no son necesarios puesto que son generados automáticamente por Android Studio.

4.4.3. Diseño de la base de datos

La finalidad de este subapartado es la de comprender el diseño elegido para la base de datos. Como cualquier diseño primero debemos de recoger los requisitos. En nuestro caso tenemos la necesidad de almacenar dos números, uno que representa la marca de tiempo de un instante determinado y otro que es la medida de la glucosa calculada en base a un cierto cálculo. Las medidas pueden proceder de dos bloques distintos de la memoria del sensor, uno que almacena las medidas minuto a minuto y otro que almacena los datos de las últimas ocho horas. Los datos

se mostrarán en fragmentos distintos y no deben ser mezclados para evitar incongruencias en la representación del histórico de los datos. Hay varias posibles configuraciones, pero finalmente hemos optado por crear una base de datos SQLite embebida en el teléfono con dos tablas (history y currents). Cada tabla tiene dos columnas númericas (timestamp y measure).

4.4.4. Diagrama de flujo

En esta subsección mostraremos el diagrama de flujo de la aplicación. Cabe descatar que el flujo es el mismo ya sea iniciada por un *intent* producido por la lectura del sensor o bien por el acceso de un usuario a la aplicación, la única diferencia que habrá es que en la pantalla principal se mostrará la última medida hecha con las medidas de los últimos siete y quince minutos.

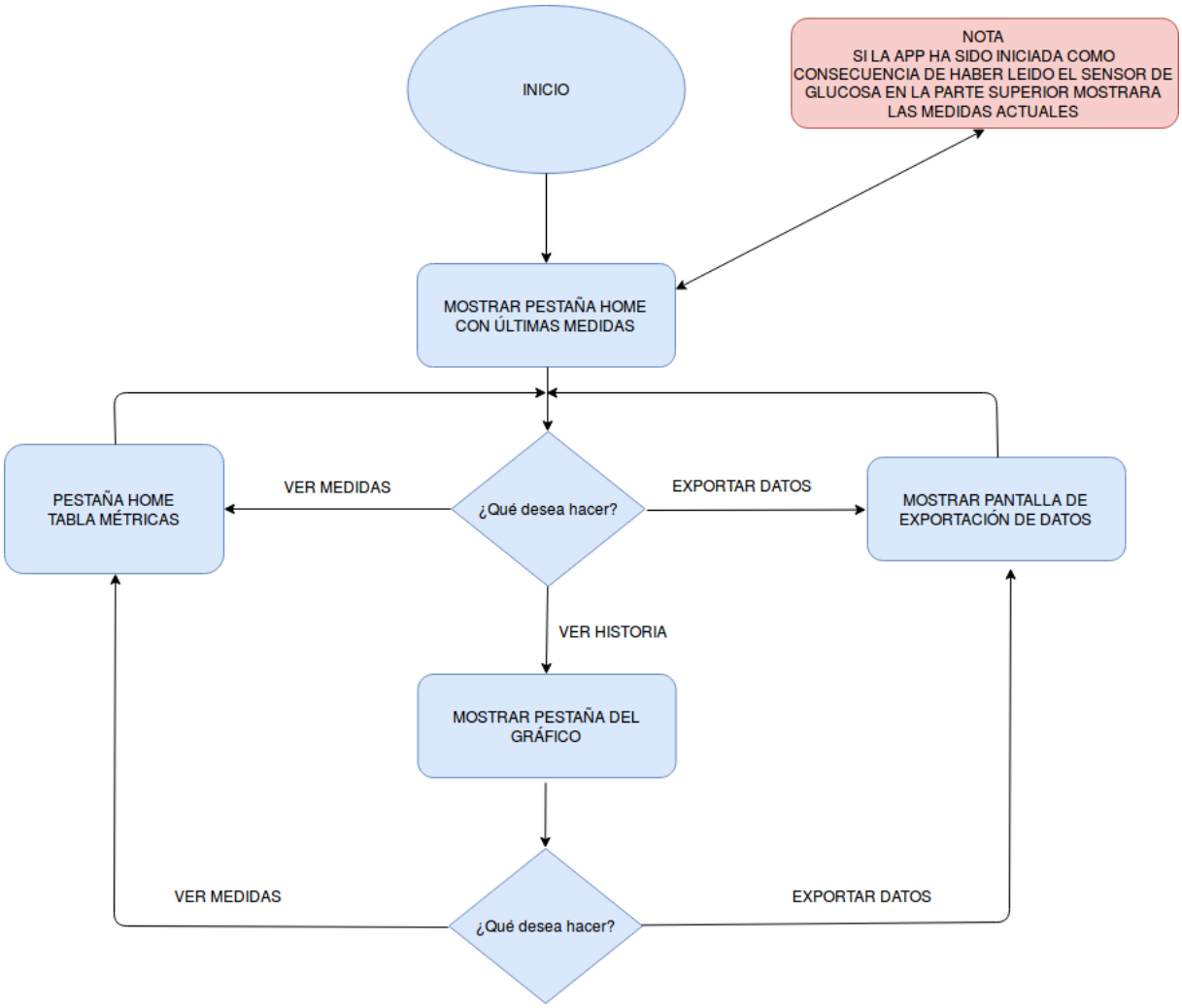

Figura 4.21: Diagrama de flujo de la aplicación

4.4.5. Código del proyecto

El código puede ser encontrado en `https://github.com/venancioromero/TFG`. El proyecto consta de una serie de carpetas,y ficheros con extensión `java` y `xml`. A continuación mencionaremos los ficheros más relevantes que construyen la mayor parte de la lógica y sobre todo las pantallas de la aplicación:

MainActivity.java: fichero del *activity* principal que cargará los *fragments* de *home* y *chart*. Además esta clase es la encargada de manejar los eventos del menú de opciones.

fragment_home.xml: fichero xml del *fragment home*, en él se definen los aspectos visuales del fragmento.

ChartFragment.java : clase java que es utilizada en la lógica del *fragmento chart*. En ella se construye el gráfico en el que se muestra la historia.

fragment_chart.xml:*layout del fragmento chart* que únicamente contiene el bloque del gráfico. Este *widget* será rellenado mediante código en la clase ChartFragment.java.

ExportData.java: clase java que es utilizada en la *activity* de la exportación de datos. Es la encargada de lanzar el *intent* para que una vez generado el fichero protegido mediante una password, pueda ser enviado por mail o compartido.

activity_export_data.xml: *layout* de la *activity* de exportación de los datos. Contiene los vistas necesarias para componer el aspecto visual de la pantalla.

4.4.6. Descripción de las pantallas

Fundamentalmente, la aplicación está compuesta por dos *activities*:

- **Activivity Principal**: este *activity* es la primera que se muestra cuando un usuario entra en la aplicación o esta es disparada por un *intent* generado por la lectura de un sensor. Esta dividida en dos pestañas:
    - **Pestaña home**: *fragment* que se muestra en primer lugar y que está dividido en dos partes.
        - Parte superior: muestra la medida actual y las tomadas hace siete y quince minutos. El motivo por el que se muestran estas observaciones, es porque pueden ayudar a predecir al usuario una posible hipoglucemia o hiperglucemia.

- Parte inferior: contiene una tabla detallada con la fecha y hora de las medidas tomadas anteriormente.

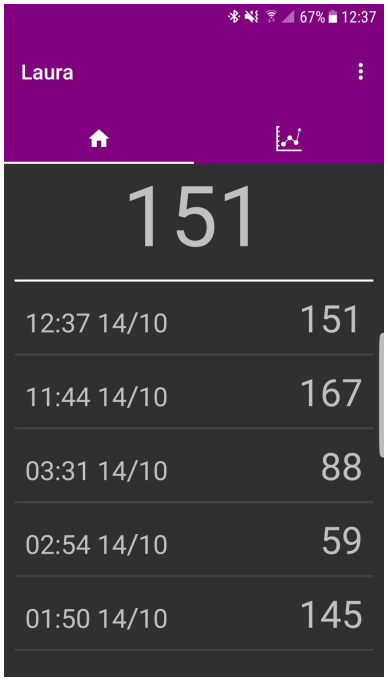

Figura 4.22: Pantalla principal, pestaña home

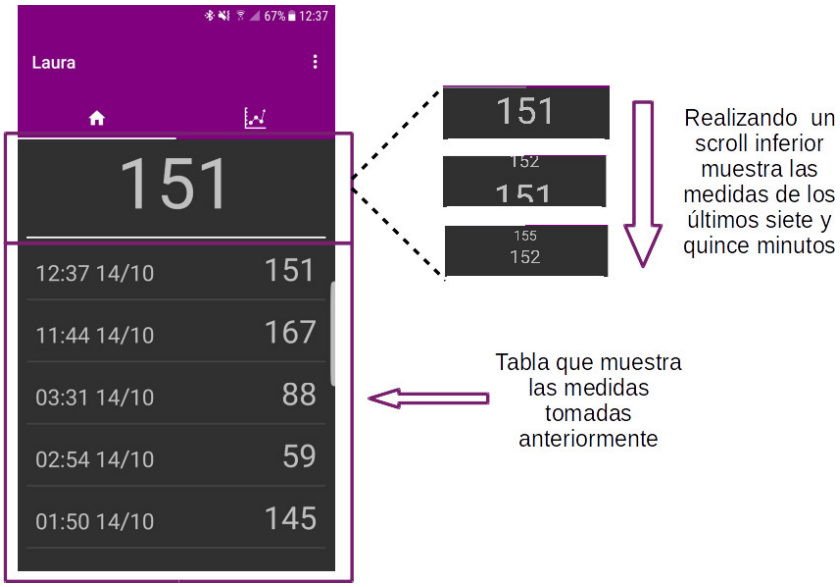

Figura 4.23: Detalle de la pestaña home

- **Pestaña chart**: *fragment* que representa como una serie temporal los últimos catorce

días las medidas de glucosa tomadas del sensor. Cuando se entra la gráfica está centrada en las últimas ocho horas pero tiene la capacidad de hacer zoom sobre las zonas que se desee.

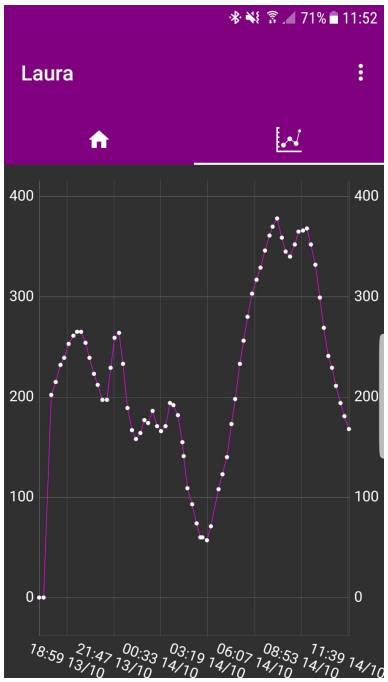

Figura 4.24: Pantalla principal, pesataña chart.

- **Activity de exportación de datos**: este *activity* es donde se realiza la extracción de los datos. En ella es necesario especificar las fechas del intervalo a exportar y la password que se utilizará para proteger el fichero. Generará un fichero con extensión *zip* comprimido y protegido por una contraseña que podremos enviar adjunto en un correo electónico, guardar en un sistema de almacenamiento en la nube o compartir en cualquier cliente de mensajería.

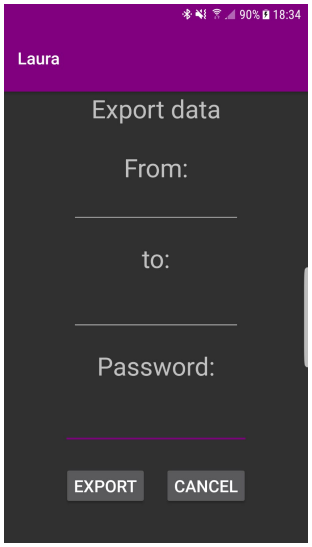

Figura 4.25: Pantalla exportación de datos

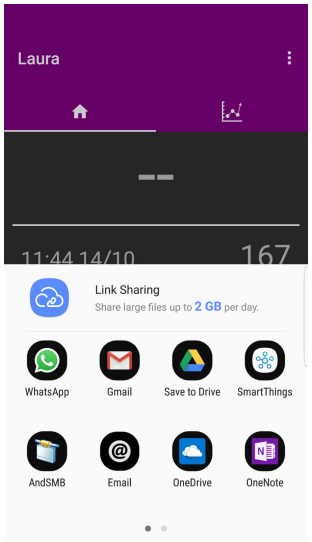

Figura 4.26: Capacidad de enviar y compartir el fichero

Report1691504030.csv

```
date,glucose
18:05 13/10,202
18:20 13/10,215
18:35 13/10,232
18:50 13/10,239
19:05 13/10,253
19:20 13/10,261
19:35 13/10,265
19:50 13/10,265
20:05 13/10,254
20:20 13/10,239
20:35 13/10,223
20:50 13/10,212
21:05 13/10,197
21:20 13/10,197
21:35 13/10,229
21:50 13/10,259
22:05 13/10,264
22:20 13/10,233
22:35 13/10,189
```

Figura 4.27: Detalle del fichero csv con datos reales

Capítulo 5

Conclusiones

Se ha conseguido desarrollar una aplicación móvil Android intuitiva e informativa para cualquier persona. Esta aplicación ayudará a personas que necesiten medir su glucosa a llevar un control y reportar a su médico la evolución de su enfermedad en los meses previos a la consulta.

Uno de los requisitos principales del proyecto era que la aplicación debía de leer un dispositivo del que no teníamos apenas información. No lograr este requisito podía suponer el fracaso de este trabajo. Finalmente, con esfuerzo y tras la toma de muchísimas lecturas logramos detectar el patrón de cambio que tenía el sensor y así poder determinar la estructura de los datos.

Tras la lectura del sensor, el siguiente objetivo fue conseguir que los valores se ajustaran lo máximo posible a la realidad. Para ello, tomamos nuevas medidas anotando el valor de la glucosa que el sensor guardaba (sin ningún tipo de transformación) y la medida real tomada en un glucómetro capilar. Con toda esta información se realizó un pequeñó análisis que determinó la ecuación mediante la cual podíamos obtener el valor de la glucosa real.

Los siguientes requisitos cumplidos fueron que logramos representar las medidas de una forma sencilla, clara y que fuera interpretable por cualquier persona y además que la aplicación fuera capaz de exportar las medidas mediante un fichero protegido con contraseña.

El tiempo empleado en las fases del proyecto se divide en tres partes. La primera, en la que tratamos todos los aspectos del sensor de glucosa, hemos empleado aproximadamente el 60 % del tiempo. Para la segunda, en la que debíamos desarrollar la aplicación, hemos el utilizado aproximadamente el 25 % del tiempo. El 15 % restante se ha destinado en la elaboración de esta memoria y las transparencias de la presentación del trabajo.

Quiero destacar que todas las asignaturas del grado en mayor o menor medida me han servido para comprender muchos aspectos necesarios para este proyecto, por ejemplo arquitectura de computadores, para entender el funcionamiento del sensor, estadística para explicar lo que estaba ocurriendo con los valores atípicos medidos y la relación entre variables, asignaturas de señales para entender que lo que estaba viendo eran series temporales y cómo debía de representarlas, ampliación de sistemas telemáticos que en esa asignatura aprendí a programar en Java y laboratorio de sistemas móviles y ubicuos que en ella aprendí a programar aplicaciones móviles. Además he aprendido la importancia de utilizar un control de versiones en el que se puedan registrar todos los cambios, he profundizado en bastantes aspectos del sistema operativo Android y también he aprendido a realizar un diagrama de flujo de una aplicación Android.

5.1. Lineas futuras

Existen multitud de mejoras que pueden ser implementadas en el sistema propuesto sin que se añada excesiva complejidad. En un trabajo futuro sería interesante que el sistema tuviera la capacidad de sincronización con un servicio externo para el almacenaje y visualización de los niveles a terceras personas. Esta característica añadiría tolerancia a fallos al sistema y cualquier persona autorizada podría ver en todo momento el estado actual del paciente.

Otra propiedad interesante relacionada con la obtención de datos sería que la aplicación fuera capaz de entender otros sensores de glucosa disponibles en el mercado o incluso que fuera capaz de extraer datos de medidores de glucosa capilar. Esta cualidad concentraría en una única base de datos la información de todos los medidores que usa el paciente, incluso podrían utilizarse para alimentar un sistema para disminuir el error en la medida.

Otra característica que agradecerían personas que son nuevas en la enfermedad sería tener una pestaña con recetas de cocina con la cantidad de hidratos de carbono que contiene cada plato. Esta pestaña podría ser una herramienta colaborativa y el contenido debería estar supervisado por nutricionistas.

Bibliografía

[1] Asociación diabetes de Madrid (2015): ¿Conoces la historia de la Diabetes?.

Disponible en:

https://diabetesmadrid.org/conoces-la-historia-de-la-diabetes/

[Consulta: 10 de septiembre de 2018]

[2] Fernández T. (2017): La diabetes. Trabajo final de grado superior en nutrición y dietética inédito. Instituto Inter. Barcelona.

[3] Reseller tech consulting (2017). Android acapara el 86 % del mercado mundial de smartphones.

Disponible en:

https://www.itreseller.es/en-cifras/2017/05/android-acapara-el-86-del-m

[Consulta: 16 de septiembre de 2018]

[4] Apple Inc. (2017): Resumen de ventas del año 2017 divido en trimestres.

Disponible en:

https://www.apple.com/newsroom/pdfs/Q1FY17DataSummary.pdf
https://www.apple.com/newsroom/pdfs/Q2FY17DataSummary.pdf
https://www.apple.com/newsroom/pdfs/Q3FY17DataSummary.pdf

[Consulta: 11 de noviembre de 2018]

[5] Texas Instrument (2012): Datasheet del cirtuito integrado RF430.

Disponible en:

http://www.ti.com/lit/ds/symlink/rf430frl152h.pdf

[Consulta: 1 de julio de 2018]

[6] jvillep (2010): Convertidor análogo-digital Sigma-Delta

Disponible en:

`https://www.monografias.com/trabajos/sigmadelta/sigmadelta.shtml`

[Consulta: 11 de noviembre de 2018]

[7] Igoe T., Coleman D. and Jepson B. (2014): *Beginning NFC*.

Disponible en:

https://www.oreilly.com/library/view/beginning-nfc/9781449324094/

[Consulta:20 de julio de 2018]

[8] Google.(2018):Android - Historia.

Disponible en:

`https://www.android.com/intl/es_es/history`

[Consulta: 12 de septiembre de 2018]

[9] Abbott Diabetes Care Inc, (2018): Página oficial.

Disponible en:

`https://www.diabetescare.abbott/`

[Consulta 14 de julio de 2018]

[10] Abbott Diabetes Care Inc, (2018): Página oficial del sistema de medida de glucosa.

Disponible en:

`https://www.freestylelibre.es/libre/`

[Consulta: 5 de septiembre de 2018]

[11] NightScout Project.(2018):Página oficial del proyecto Nightscout.

Disponible en:

`http://www.nightscout.info/`

[Consulta: 5 de septiembre de 2018]

[12] Dexcon Inc. (2018): Página oficial Dexcom.

Disponible en:

`https://www.dexcom.com/g6-cgm-system`

[Consulta: 5 de septiembre de 2018]

[13] Medtronic (2018): Página oficial de medtronic.

Disponible en:

`http://europe.medtronic.com/xd-en/index.html`

[Consulta: 5 de septiembre de 2018]

[14] Real Academia Española (2018): Diccionario de la lengua española.

Disponible en:

`http://dle.rae.es`

[Consulta: 5 de septiembre de 2018]

[15] Wikipedia (2018): Endianess.

Disponible en:

`https://en.wikipedia.org/wiki/Endianness`

[Consulta: 7 de septiembre de 2018]

[16] Free Software Foundation (2018): What is GNU?.

Disponible en:

`https://www.gnu.org`

[Consulta: 14 de septiembre de 2018]

[17] NFC-Forum.(2018): Página Oficial de la NFC-Forum.

Disponible en:

`https://nfc-forum.org`

[Consulta: junio de 2018]

[18] Wikipedia (2018): S (programming language).

Disponible en:

https://en.wikipedia.org/wiki/S_(programming_language)

[Consulta: 25 de septiembre de 2018]

[19] Wikipedia (2018): Zip_(file_format).

Disponible en:

https://en.wikipedia.org/wiki/Zip_(file_format)

[Consulta: 10 de noviembre de 2018]

[20] Wikipedia (2018): Philz Katz

Dispoible en:

https://en.wikipedia.org/wiki/Phil_Katz

[Consulta: 12 de noviembre de 2018]

[21] Texas Instruments (2018): RF430-TMPSNS-EVM Battery-less NFC/RFID Temperature Sensing Evaluation Module.

Disponible en:

http://www.ti.com/tool/rf430-tmpsns-evm

[Consulta: junio de 2018]

[22] James G., Witten D., Hastie T. and Tibshirani R. (2014): *An introduction to statistical learning*.

Disponible en :

https://www-bcf.usc.edu/ gareth/ISL/

[Consulta: 15 de Agosto de 2018]