VNiVERSiDAD D SALAMANCA

CAMPUS DE EXCELENCIA INTERNACIONAL

DiabeticLog: Diario de seguimiento de una persona diabética

-

Trabajo de Fin de Grado - Ingeniería Informática

Curso 2020/2021

-

Autor:

Adrián Antonio González Pazos

-

Tutores:

Ana Belén Gil González

Belén Pérez Lancho

José Luis Vidal de la Rosa

Las Dras. Ana Belén Gil González y María Belén Pérez Lancho, profesoras del Departamento de Informática y Automática de la Facultad de Ciencias, y D. José Luis Vidal de la Rosa, como tutor del Observatorio HP

HACEN CONSTAR

que el presente documento titulado "***DiabeticLog:* Diario de seguimiento de una persona diabética**" ha sido realizado bajo su supervisión por **D. Adrián Antonio González Pazos,** con DNI 70922606J, y constituye la memoria del proyecto realizado para la superación de la asignatura **Trabajo de Fin de Grado** de la Titulación Grado en Ingeniería Informática de la Universidad de Salamanca,

para que así conste a todos los efectos oportunos.

Salamanca, 6 de julio de 2021

GIL GONZALEZ ANA BELEN - 07973126S
Firmado digitalmente por GIL GONZALEZ ANA BELEN - 07973126S
Fecha: 2021.07.06 12:31:15 +02'00'

PEREZ LANCHO MARIA BELEN - 07858727H
Firmado digitalmente por PEREZ LANCHO MARIA BELEN - 07858727H
Nombre de reconocimiento (DN): c=ES, serialNumber=IDCES-07858727H, givenName=MARIA BELEN, sn=PEREZ LANCHO, cn=PEREZ LANCHO MARIA BELEN - 07858727H
Fecha: 2021.07.06 12:16:13 +02'00'

VIDAL DE LA ROSA JOSE LUIS - 72890587E
Digitally signed by VIDAL DE LA ROSA JOSE LUIS - 72890587E
Date: 2021.07.06 16:56:18 +02'00'

Fdo.: Ana Belén Gil González | Fdo.: Belén Pérez Lancho | Fdo.: José Luis Vidal de la Rosa

Resumen

El proyecto “DiabeticLog: Diario de seguimiento de una persona diabética” tiene como objetivo principal la creación de una aplicación móvil que permita ayudar a las personas con diabetes a controlar diferentes aspectos de su rutina como diabéticos. El proyecto ha sido desarrollado dentro del convenio de colaboración de la Facultad de Ciencias con el Observatorio HP.

Los usuarios de la aplicación podrán registrar aquellos aspectos que afectan directamente a su nivel de azúcar en sangre, por ejemplo, las dosis de insulina, el número de raciones de hidratos de carbono consumidas, o el diferente deporte realizado.

Además, la aplicación permite almacenar otros aspectos que también afectan al nivel de glucosa. Por un lado, el estado emocional, ya que hay ciertas emociones que pueden subir el azúcar en sangre y, por otro lado, el ciclo menstrual que también puede afectar de manera indirecta.

Aunque estos datos se registran en la aplicación de forma manual, se ha realizado un estudio de las posibilidades que ofrece la tecnología existente en este ámbito de cara a una futura integración en la aplicación.

Los datos recogidos por la aplicación móvil se envían a un servidor creado para ese fin. Para ello la aplicación Android se conectaría a una API REST desarrollada en Golang.

Por último, la recolección de datos relacionados con la diabetes podría permitir la creación de algoritmos de inteligencia artificial capaces de predecir el nivel de azúcar en sangre de una persona con diabetes.

Palabras clave

Aplicación móvil, Diabetes, Servidor, Base de datos, Android, Golang

Summary

The main goal of the project “DiabeticLog: Diary of a diabetic person” is the creation of a mobile application to help people with diabetes to control different aspects of their diabetic routine. The project has been developed within the collaboration agreement between the Faculty of Science and the HP Observatory.

Users of the application will be able to record those aspects that directly affect their blood sugar level, for example, insulin doses, the number of carbohydrate servings consumed, or the different sports performed.

In addition, the application allows you to store other aspects that also affect blood glucose level. On the one hand, the emotional state, as there are certain emotions that can raise blood sugar, and on the other hand, the menstrual cycle, which can also affect it indirectly.

Although this data is recorded manually in the application, a study of the possibilities offered by existing technology in this area has been carried out with a view to future integration into the application.

The data collected by the mobile application is sent to a server created for that purpose. To do this, the Android application would connect to a REST API developed in Golang.

Finally, the collection of diabetes-related data could enable the creation of artificial intelligence algorithms capable of predicting the blood sugar level of a person with diabetes.

Keywords

Mobile application, Diabetes, Server, Database, Android, Golang

Tabla de contenido

Índice de ilustraciones

Índice de tablas

1. Introducción

La diabetes es una enfermedad metabólica caracterizada por niveles de azúcar en sangre elevados. En las últimas décadas el número de afectados por la diabetes ha ido aumentando.

Actualmente, la diabetes es la cuarta causa de muerte en la mayoría de los países desarrollados. Además, está también comenzando a afectar en países en vías de desarrollo [1].

En 2014 había 387 millones de afectados en el mundo. Se prevé que esta cifra incremente en las próximas décadas, llegando a los 592 millones de personas en 2035. Teniendo un mayor incremento en los países menos desarrollados [1].

En cuanto a España, en 1995 había 2,1 millones de personas con diabetes. Un estudio de 2011 dio como resultado que 5,3 millones de personas mayores de 18 años padecían diabetes de tipo 2, lo que equivale a un 13,8% de la población [1].

El aumento de estas cifras se debe en gran medida a un estilo de vida cada vez más sedentario, donde abunda la llamada "comida basura" [1].

Para los afectados, supone un cambio en su estilo de vida y un seguimiento constante, teniendo que llevar un control de diferentes aspectos de la enfermedad.

1.1. Estado actual de la tecnología en el ámbito de la diabetes

En la actualidad, diferentes empresas farmacéuticas y tecnológicas han ido desarrollando diferentes soluciones que ayuden a controlar la enfermedad.

La empresa estadounidense Abbot ofrece un sensor de glucosa denominado FreeStyle (https://www.freestylelibre.es/libre/). Se trata de un sensor intracutáneo el cual actualiza los resultados de glucosa cada minuto, y los envía a una aplicación móvil que han desarrollado para este fin. Una de las ventajas de esta propuesta es que elimina la necesidad de pinchazos rutinarios en el dedo para medir los niveles de glucosa. Sin embargo, una de sus desventajas es que el sistema solo permite medir el nivel de azúcar en sangre, dejando fuera aspectos importantes para una persona con diabetes, como, por ejemplo, la aplicación de insulina.

Existen varios sensores de medición continua de glucosa similares a Freestyle en el mercado, como el sensor Enlite, desarrollado por la empresa Medtronic, o el dispositivo Dexcom. Sin embargo, todos presentan características similares a las que tiene FreeStyle.

Ilustración 1 Sensor FreeStyle y aplicación

Por otro lado, una empresa española ha desarrollado Insulclock, un dispositivo que permite registrar de forma automáticamente la aplicación de insulina (https://insulclock.com/patient/). Según afirman ellos, se trata del único sistema del mundo que permite el registro automático del uso de la pluma desechable de insulina. Insulclock también cuenta con una aplicación móvil que, al contrario que la de FreeStyle, permite registrar otros aspectos de la diabetes como las raciones de hidratos de carbono o el ejercicio físico.

Ilustración 2 Sensor Insulclock y aplicación

En este contexto, uno de los principales problemas es que cada empresa lanza su solución sin tener en cuenta las propuestas ya existentes en el mercado. Por ejemplo, si quisiéramos usar los sistemas Freestyle e Insulclock tendríamos que usar sus dos aplicaciones oficiales, que pueden suponer un problema para gente mayor o para personas que no sepan utilizar muy bien la tecnología, limitando de esta manera el acceso a estos dispositivos. Además, estas empresas cierran la posibilidad de poder desarrollar otras aplicaciones usando sus sensores, al no contar con un SDK[1] abierto para su uso. Esto limita que desarrolladores puedan crear alternativas que mejoren las aplicaciones oficiales.

Por último, hay que mencionar que actualmente los dispositivos descritos no están financiados por el sistema de salud español, lo cual hace que haya menos gente que use esta serie de dispositivos. Solo financia desde 2018 los medidores de glucosa de tipo *flash*, los cuales no permiten una monitorización continua de la glucosa [2].

Ilustración 3 Medidor tipo flash de glucosa

1 Un SDK (Software Development Kit) es un conjunto de herramientas que permite a un desarrollador crear una aplicación informática para un hardware o software concreto.

1.2. Motivación

En este contexto de propuestas expuestas para ayudar a una persona con diabetes a controlar la enfermedad se enmarca este proyecto, que ha sido desarrollado dentro del convenio de colaboración de la Facultad de Ciencias con el Observatorio HP.

El proyecto pretende unificar, en una sola aplicación, el registro de todos los aspectos que afectan a la glucosa para, de esta manera, facilitar a las personas el control de la diabetes. Como punto de partida, se pretende dar la posibilidad a los usuarios introducir los diferentes aspectos relacionados con la diabetes de forma manual. Los principales datos serían niveles de azúcar en sangre, ingesta de hidratos de carbono y unidades de insulina aplicadas. Además, se analizarán las posibilidades técnicas que permitan conectar los sensores de Freestyle e Insulclock a la aplicación, obteniendo así estos datos de forma automática.

Se pretende también que la aplicación pueda mantener un diario de otros aspectos que puedan afectar a la glucosa, ya que no solo la insulina o las raciones de hidrato de carbono afectan al nivel de azúcar en sangre. De esta manera se busca obtener un diario de seguimiento lo más completo posible que permita, por ejemplo, el registro de actividad física, el cual también puede realizarse de forma manual o mediante pulseras de actividad.

Además de la utilidad inmediata, la idea que subyace en este proyecto es que la recolección sistemática de todos los datos relacionados con la diabetes pueda posteriormente servir de ayuda al desarrollo de algoritmos que permitan, por ejemplo, predecir el nivel de glucosa. Esto puede ser muy útil para desarrollar sistemas inteligentes y adaptativos que ayuden a controlar automáticamente el nivel de azúcar en sangre.

2. Objetivos del trabajo

Se presentan a continuación los objetivos que deberá cumplir el sistema. Estos se dividirán en dos tipos de objetivos: funcionales y no funcionales.

Esta especificación de objetivos surge de los requerimientos indicados en la propuesta realizada por el tutor de la empresa HP, José Luis Vidal de la Rosa, donde se explicaba de forma general qué funcionalidades debería tener la aplicación móvil a desarrollar. En etapas iniciales del proyecto se consideró la posibilidad de incluir sensores para obtener algunos de los datos de forma automática, sin embargo, se dio prioridad al registro manual frente al de estos sensores. No obstante, durante el desarrollo del proyecto se ha hecho un estudio de estas posibilidades, y se ha tenido en cuenta durante el diseño, con el objetivo de una futura integración en la aplicación

2.1. Objetivos funcionales

Se definen a continuación aquellas funciones que el sistema deberá ser capaz de realizar, los llamados objetivos funcionales, que también se encuentran descritos en el "Anexo II – Especificación de requisitos".

- El primer objetivo es "Registro manual de datos", el cual consiste en el que sistema deberá ser capaz de registrar y guardar diferente información relacionada con la diabetes.

  La información que se tendrá que guardar será la fecha y hora de las diferentes aplicaciones de insulina, el número de raciones de hidratos de carbono consumidas, y el ejercicio o el deporte que el usuario vaya realizando. Esta información es importante que esté controlada en una persona con diabetes porque afectan directamente al nivel de azúcar en sangre.

  También se deberá guardar el estado emocional y la fecha de inicio y fin del ciclo menstrual, en el caso de mujeres en edad fértil. En el caso del estado emocional existen cuatro emociones que pueden afectar al nivel de glucosa en sangre: estrés, ansiedad, angustia y alegría. Las primeras tres pueden hacer subir el azúcar, mientras que la última mantiene los niveles de azúcar estables. En el caso del ciclo menstrual, este puede afectar a la sensibilidad a la insulina. Es por ello, que, aunque afecte en menor medida al nivel de azúcar en sangre que las tres anteriores, es interesante tenerlo en cuenta, y mantener un registro sobre ello.

  Por descontando, todos los datos anteriormente mencionados deberían estar disponibles para que el usuario pueda verlos de forma sencilla en la aplicación.

  Finalmente, se debería dar flexibilidad al usuario para poder editar o eliminar cualquiera de los datos guardados.

OBJ - 001	Registro manual de datos
Versión	1.0.1 (09-02-21)
Autores	Adrián Antonio González Pazos
Descripción	La aplicación deberá poder registrar de forma manual información referente a la ingesta, la aplicación de insulina, la actividad realizada, el estado emocional y el ciclo menstrual. Además, se deberá poder ver, editar y eliminar estos registros
Subobjetivos	Ninguno
Urgencia	Media
Estado	Verificado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 1 OBJ - 001 Registro manual de datos

- El siguiente objetivo es "Visualización de datos y perfil". Con él se pretende formalizar la existencia de un perfil de usuario, donde las personas que utilicen la aplicación podrán visualizar sus datos.

  Estos datos podrían ser los típicos de un perfil de usuario, como nombre, u otros más relevantes en el ámbito de aplicación de este sistema, como son el peso y la altura.

  Se sobrentiende también la posibilidad de que el usuario puede editar cualquiera de los datos mostrados en su perfil.

OBJ - 002	Visualización de datos y perfil
Versión	1.0.1 (09-02-21)
Autores	Adrián Antonio González Pazos
Descripción	Los datos registrados se podrán visualizar en el perfil del usuario. Además de su peso y altura.
Subobjetivos	Ninguno
Urgencia	Media
Estado	Verificado
Estabilidad	Alta

Tabla 2 OBJ - 003 Visualización de datos y perfil

- El tercer objetivo es "Sincronización con el servidor", el cual consiste en el desarrollo de un servidor que reciba los datos capturados por los usuarios de la aplicación móvil.

  La información que deberá guardar el servidor en su base de datos es la indicada en el primer registro: la aplicación de insulina, el número de raciones de hidratos de carbono, la actividad diaria, el estado emocional y el ciclo menstrual.

  La existencia del servidor pone la obligación de la existencia de un método en la aplicación móvil que envíe todos estos datos. Este método podrá realizarse mediante subidas manuales o automáticas, siendo esta última transparente para el usuario. También se deberá implementar un mecanismo que reintente el envío de los datos en caso de que haya fallado, ya sea porque el dispositivo no tiene conectividad a la red o por fallo del propio servidor. Por último, el servidor deberá tener también un mecanismo que devuelva los datos guardados y que puedan ser mostrados en la aplicación móvil.

OBJ - 003	**Sincronización con el servidor**
Versión	1.0.1 (09-02-21)
Autores	Adrián Antonio González Pazos
Descripción	Deberá existir un servidor que reciba toda la información capturada por la aplicación móvil. Para ello la aplicación móvil deberá tener un mecanismo manual o automático para realizar las subidas de datos al servidor
Subobjetivos	Ninguno
Urgencia	Media
Estado	Verificado
Estabilidad	Alta

Tabla 3 OBJ – 003 Sincronización con el servidor

- El cuarto objetivo es “Seguridad en la aplicación”. Hoy en día es muy importante que las aplicaciones cumplan un mínimo de seguridad que garantice a los usuarios la privacidad de sus datos. Es por ello por lo que la aplicación deberá contar con un mecanismo de seguridad para poder acceder a ella.

  Este mecanismo de seguridad podrá ser usuario y contraseña contra el servidor, un pin de seguridad o incluso datos biométricos. En este caso se ha optado por usuario y contraseña contra el servidor.

  La elección de un sistema de acceso mediante usuario y contraseña tiene como consecuencia la necesidad de implementar un mecanismo de registro donde se le pida al usuario la creación de unos datos de acceso. Asimismo, se deberá contar con un sistema de inicio de sesión que compruebe con el servidor si los datos introducidos por el usuario son correctos, y se le permita entrar a su cuenta. Finalmente, se le dará al usuario la posibilidad de cerrar sesión o de editar los datos de acceso.

OBJ - 004	Seguridad aplicación
Versión	1.0.1 (09-02-21)
Autores	Adrián Antonio González Pazos
Descripción	Para poder acceder a la aplicación móvil, se implementará un mecanismo de seguridad
Subobjetivos	Ninguno
Urgencia	Media
Estado	Verificado
Estabilidad	Alta
Comentarios	El mecanismo de seguridad puede ser usuario y contraseña contra el servidor, datos biométricos o pin de seguridad

Tabla 4 OBJ - 004 Seguridad aplicación

- El último objetivo es “Notificaciones de insulina e ingesta”. Este objetivo pretende recordar al usuario la aplicación de insulina, así como el registro del número de hidratos de carbono o ingesta.

  Para ello la aplicación deberá permitir que el usuario programe las horas en las que debe realizar las acciones mencionadas y su registro, comprobar cuando no se ha introducido cualquiera de los datos mencionados y mostrar a través de una notificación en el dispositivo móvil el recordatorio correspondiente.

  Además, el usuario deberá poder activar o desactivar las notificaciones, así como cambiar la hora en la que se mostrarían.

OBJ - 005	Notificaciones insulina e ingesta
Versión	1.0.1 (09-02-21)
Autores	Adrián Antonio González Pazos
Descripción	La aplicación será capaz de detectar cuándo no se han introducido los datos y mostrar una notificación para que registre la ingesta o la insulina
Subobjetivos	Ninguno
Urgencia	Media
Estado	Verificado
Estabilidad	Alta
Comentarios	Ninguno

Tabla 5 OBJ - 005 Notificaciones insulina e ingesta

2.2. Objetivos no funcionales

Se especifican ahora los objetivos no funcionales, los cuales, aunque no formalizan una funcionalidad en concreto, sí que son requisitos importantes para tener en cuenta, y que mejoran el proyecto. Como en el caso anterior, también se encuentran recogidos en el "Anexo II – Especificación de requisitos".

NFR - 001	Seguridad datos
Versión	1.0.1 (09-02-21)
Autores	Adrián Antonio González Pazos
Descripción	El sistema deberá ser capaz de tener seguridad en los datos almacenados de tal forma que nadie pueda acceder a los datos registrados a excepción del usuario al que pertenezcan

Tabla 6 NFR - 001 Seguridad datos

NFR - 002	Interfaz
Versión	1.0.1 (09-02-21)
Autores	Adrián Antonio González Pazos
Descripción	El sistema deberá tener una interfaz sencilla y que sea fácil de usar, con el fin de que la pueda usar gente de todas las edades

Tabla 7 NFR - 002 Interfaz

2.3. Objetivos Personales

El primer objetivo personal es poder aplicar los conocimientos aprendidos durante los cuatro años del grado en el desarrollo de un proyecto.

Asimismo, otro de los objetivos es aprender los conocimientos básicos que se requieren para realizar una aplicación móvil para el sistema operativo Android, utilizando el lenguaje de programación Kotlin, y el entorno de desarrollo Android Studio. Estos conocimientos no se ven a lo largo de la carrera, y se han tenido que aprender de manera autónoma para el desarrollo del proyecto.

Relacionado con el anterior, también está presente el objetivo de aprender a realizar un servidor API-REST usando el lenguaje de programación Go (conocido también como Golang)

Por último, que los conocimientos aprendidos durante el transcurso del trabajo me puedan ayudar en el futuro ampliando las posibles salidas al mercado laboral.

3. Aspectos Teóricos

3.1. Aspectos teóricos de la diabetes

Debido a que se trata de una aplicación con un ámbito médico se explicara a continuación algunos aspectos relevantes de la diabetes, los cuales se han tenido en cuenta para el desarrollo del proyecto. De esta manera se busca exponer de forma breve el marco teórico del ámbito de la aplicación.

La diabetes es un conjunto de trastornos metabólicos, cuya característica común principal es la presencia de concentraciones elevadas de glucosa en la sangre. Esto implica que una persona diabética debe llevar un control sobre el índice de glucosa en sangre, o lo que es lo mismo, el nivel de azúcar en sangre.

Un nivel aceptable de azúcar en sangre está dentro del rango 70-120. Donde un nivel debajo de 70 se considera hipoglucemia, y un valor por encima de 120 es hiperglucemia. Ambos casos traen riesgos para la salud:

- **Hipoglucemia**: Podría producirse una pérdida de conocimiento impidiendo la ingesta de azúcares de absorción rápida. Esto se produce debido a que el cerebro necesita azúcar para funcionar.
- **Hiperglucemia:** El azúcar en sangre se dispararía si no se toma tratamiento de insulina. El cuerpo humano obtiene el azúcar de la grasa corporal, generando cetona. Este compuesto es tóxico en grandes cantidades, pudiendo provocar cetoacidosis. Si no se corrige con insulina puede implicar el ingreso de un paciente diabético en la UCI debido a complicaciones cardiacas o cerebrales.

En la diabetes de tipo 1, las personas diabéticas se consideran insulinodependientes, y deben aplicarse una serie de unidades de insulina, las cuales puede dividirse en tres tipos:

- **Insulina lenta o basal:** Tiene una duración de 24 horas aproximadamente en el cuerpo humano. Se usa para controlar el azúcar en sangre durante la noche, en periodos de sueño, mientras se está en ayunas y entre comidas. Se suele aplicar una vez al día.
- **Insulina rápida:** Este tipo de insulina comienza a hacer efecto a los 5-15 minutos de su aplicación. Es capaz de corregir el pico máximo en el rango de 1 a 2 horas desde su aplicación. Se usa para corregir el azúcar en sangre que se produce en las digestiones de las comidas principales del día: desayuno, comida y cena. También cuando hay picos de azúcar en sangre descontrolados. Esta insulina hace el trabajo que debería hacer el páncreas cuando hay demasiado nivel de azúcar en sangre. Un paciente se inyectará este tipo de insulina unas 3 veces al día.
- **Insulina intermedia:** Empieza a hacer efecto entre 1 a 2 horas después de su aplicación. Se usa igual de forma que la insulina lenta, pero se necesitan más aplicaciones al cabo del día.

Respecto a la insulina, se debe tener en cuenta también que cada persona puede ofrecer resistencia a la insulina, lo cual hace que no siempre se necesite aplicar el mismo número de unidades para una misma ingesta de hidratos de carbono. Cada persona tiene un momento en el día donde puede ofrecer más resistencia a la insulina y no tiene por qué ser igual entre personas, aunque sí que suele ser constante en la misma persona.

Las emociones también pueden afectar directamente a la glucosa pudiendo producir variaciones. En concreto el estrés, ansiedad y la angustia producen un aumento del azúcar en sangre. Por otro lado, la alegría mantiene los niveles de azúcar estables.

Por último, el ciclo menstrual afecta a la sensibilidad a la insulina, provocando que en ciertos días del ciclo se necesiten más unidades de insulina de lo habitual. Si no se corrige, se tendrá el nivel de azúcar más alto.

3.2. Arquitectura

3.2.1. Modelo cliente-servidor

El modelo cliente-servidor es la arquitectura que se usará en el proyecto para conectar la aplicación móvil con el servidor. Se trata de la arquitectura más común y usada en aplicaciones distribuidas.

Este tipo de arquitectura consta de dos partes:

- **Cliente:** Solicita peticiones al servidor y procesa resultados.
- **Servidor:** Ejecuta las peticiones del servidor y devuelve los resultados.

Aunque un servidor puede también ser cliente de otros servidores.

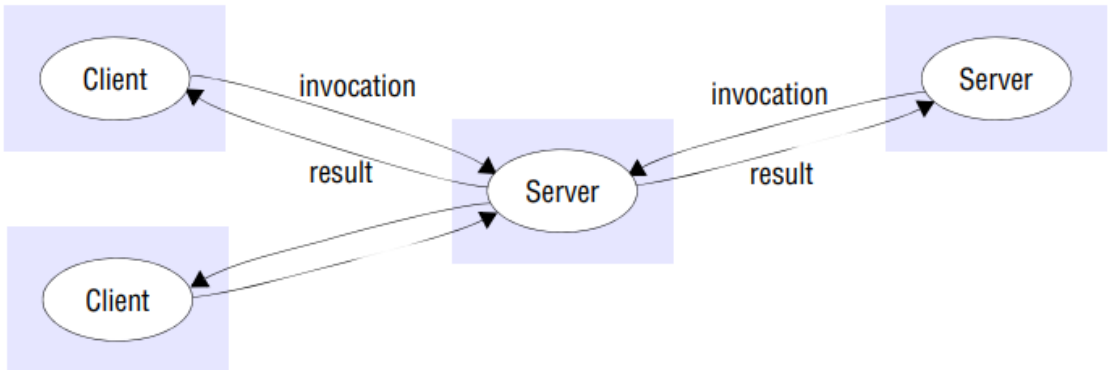

Ilustración 4 Modelo cliente-servidor [3]

Este tipo de modelo se diferencia al *peer-to-peer*, el cual se trata de sistemas de igual a igual. Aunque presenta ventajas como la reducción en los retardos en la comunicación o la eliminación de intermediarios, en el contexto de este proyecto no tiene sentido usar este tipo de arquitectura [3].

3.2.2. API REST

Con respecto a la comunicación entre el dispositivo móvil y el servidor, se usará la arquitectura API REST.

REST (***RE**presentational **S**tate **T**ransfer)* es un tipo de arquitectura de desarrollo web que utiliza el protocolo HTTP para comunicarse. Fue definida por Roy Fielding en el año 2000 [4]. El término REST en español significa "transferencia de estado representacional", lo que quiere decir que, entre dos llamadas cualquiera, el servicio no guarda datos [5].

El cliente de una API REST puede ser cualquier dispositivo que pueda usar HTTP: una aplicación Android o de iOS, un navegador web, un altavoz inteligente como Alexa, o incluso una lavadora.

Una de las características de REST es que el cliente no necesita conocer los detalles de implementación del servidor, y el servidor no se preocupa de cómo utiliza el cliente los datos [5].

Para desarrollar una API REST hay que tener en cuenta los métodos soportados por HTTP. Los principales métodos usados son los siguientes:

- **POST**: Crear un recurso nuevo
- **PUT:** Modificar un recurso existente
- **GET:** Consultar información de un recurso
- **DELETE:** Eliminar un recurso

Cuando se realiza una petición, usando alguno de los métodos indicados, se necesita conocer si la operación se ha podido llevar a cabo de forma correcta. Para esta tarea HTTP presenta códigos de error o éxito que se devuelven al realizar la petición. Entre los más comunes se encuentran:

- **200 OK:** La petición se ha realizado de manera correcta
- **201 *Created*:** La petición se ha sido completada de forma correcta y se ha creado un nuevo recurso.
- **400 *Bad Request*:** La solicitud contiene sintaxis errónea. Por ejemplo, intentar realizar una petición POST en un método que solo admite DELETE.
- **403 *Forbidden*:** La petición no se pudo realizar debido a que el cliente no tiene los permisos necesarios para realizarse.
- **404 *Not Found*:** No se ha podido encontrar el recurso solicitado
- **500 *Internal Server Error*:** Error genérico cuando se ha encontrado situaciones de error ajenas a la naturaleza del servidor web.

La principal ventaja del uso de una API REST es la independencia que proporciona frente a cualquier cliente, permitiendo que cualquier dispositivo pueda usar los servicios de la API REST. Solo se requiere que el dispositivo sea capaz de usar HTTP e intercambie la información en los formatos soportados, que generalmente son JSON y XML. Esta característica proporciona fiabilidad, escalabilidad y una fácil portabilidad a cualquier otra plataforma, al asilar por completo el cliente del servidor [4].

3.2.3. Aspectos teóricos de diseño

La interacción persona-ordenador cobra importancia en este proyecto al ser una aplicación móvil la parte fundamental del mismo. Es por ello por lo que es importante seguir las guías y recomendaciones de diseño que se usan para crear las interfaces de usuario.

Una primera aproximación es seguir los cuatro principios de diseño descritos por Robin Williams [6]. Aunque en un principio están descritos para su uso en cualquier diseño, se pueden usar y tener en cuenta para el diseño de interfaces. Estos cuatro principios son conocimos como "C.R.A.P.".

- **Contraste**: La idea del contraste es evitar usar elementos en el diseño que sean similares. El contraste puede ser de color, tipografía, tamaño o posición. Si se quiere destacar o jerarquizar información es conveniente usar algún tipo de contraste. De esta manera se ayuda a entender el contenido que está en el diseño.
- **Repetición**: Se trata de utilizar la misma forma o elemento más de una vez, creando una sensación de armonía que tiende a unir el diseño. La repetición se puede aplicar, por ejemplo, en figuras, colores, tamaños o posición.
- **Alineación:** Ningún elemento debería ser colocado de forma arbitraria. Cada uno de los elementos debería tener alguna conexión visual con otro elemento del diseño. Si un diseño está alineado, se capta cohesión y orden, facilitando la compresión del mensaje que se quiere transmitir.
- **Proximidad:** Aquellos elementos que se relacionen entre sí deberían estar colocados juntos. De esta manera se apreciarán como un grupo relacionado ayudando a organizar la información.

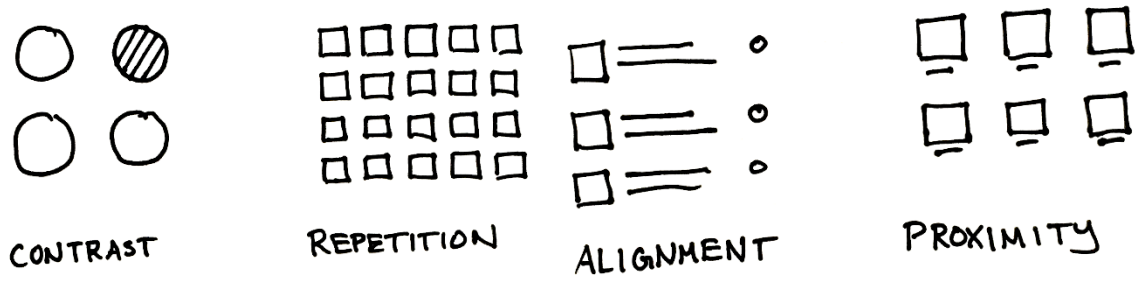

Ilustración 5 Principio de diseño C.R.A.P

Sin embargo, en una buena interfaz de usuario no solo hay que tener en cuenta el diseño visual. También es importante la usabilidad. Para este apartado se han tenido en cuenta las 10 heurísticas de usabilidad descritas por Nielsen [7].

1. **Visibilidad del estado del sistema:** El sistema siempre debe mantener informado al usuario de lo que está ocurriendo, mediante una realimentación en un tiempo razonable.
2. **Relación entre el sistema y el mundo real:** El diseño debe hablar el lenguaje de los usuarios. Se deben usar palabras, frases y conceptos que sean familiares a los usuarios, en lugar de términos internos del sistema.
3. **Control y libertad del usuario:** Los usuarios suelen realizar acciones por error. Necesitarán una "salida de emergencia" claramente marcada que les permita abandonar la acción no deseada.
4. **Consistencia y estándares:** Los usuarios no deberían preguntarse si acciones, situaciones o palabras significan lo mismo. Se debe seguir las convenciones de la plataforma y de la industria.
5. **Prevención de errores:** Los mejores diseños evitan que los errores se produzcan. O bien se eliminan aquellas condiciones propensas a error, o bien se presenta a los usuarios una opción de confirmación antes de realizar la acción.
6. **Reconocer en lugar de recordar:** Un usuario no tendría que recordar información de una parte a otra de la interfaz. La información necesaria para utilizar el diseño debe ser visible o ser fácilmente recuperable cuando se necesite.
7. **Flexibilidad y eficiencia de uso:** La presencia de atajos, ocultos para los nuevos usuarios, pueden acelerar la interacción con un usuario experto. De esta manera el diseño puede ser usado tanto para usuarios expertos como inexpertos.
8. **Estética y diseño minimalista:** Las interfaces no deberían contener información irrelevante o poco utilizada.
9. **Ayuda al usuario a reconocer, diagnosticar y recuperarse de los errores:** Los mensajes de error deben expresarse en lenguaje sencillo, sin códigos de error, que indiquen el problema y sugieran una solución.
10. **Ayuda y documentación:** Es mejor si el diseño no necesita ninguna explicación adicional. Sin embargo, puede ser necesario proporcionar documentación para ayudar a los usuarios.

4. Técnicas y herramientas

En esta sección se expondrán las diferentes herramientas y técnicas usadas en el transcurso del proyecto.

4.1. Técnicas y herramientas utilizadas en el servidor

4.1.1. Go/Golang

Para el desarrollo del servidor que recibirá los datos capturados por la aplicación móvil se ha usado el lenguaje de programación Go, también conocido como Golang. Este lenguaje de programación de código abierto ha sido desarrollado por Google y permite de forma sencilla crear aplicaciones web.

Entre las características más relevantes de Go encontramos que tiene un recolector de basura, que permite más paradigmas aparte de la programación orientada a objetos, que utiliza la concurrencia [8]. Además, su sintaxis es similar a C, lo que ha hecho que, al ser un lenguaje aprendido durante la carrera, su curva de aprendizaje se haya aplanado.

Actualmente Go es un lenguaje que está en crecimiento, y según la última encuesta realizada por Stack Overflow se encuentra en el quinto puesto de los lenguajes preferidos por los usuarios, y el tercero que más quieren aprender los programadores [9].

La sencillez de Golang permite tener un servidor simple usando muy pocas líneas de código. A continuación, se muestra el código necesario para que cuando accedamos a "localhost:8080" el navegador nos devuelva "DiabeticLog-Server". Este método se ha usado durante el desarrollo del servidor para comprobar que estaba operativo.

```
func handler(w http.ResponseWriter, r *http.Request)
{
    fmt.Fprintf(w, "DiabeticLog-Server")
}

func main() {
     router := mux.NewRouter().StrictSlash(true)
     router.HandleFunc("/", homeLink)
     log.Fatal(http.ListenAndServe(":8080", router))
}
```

Ilustración 6 "Hola mundo" lenguaje programación Go

Además, Golang permite añadir librerías de terceros, a lo que ellos llaman paquetes. Esto permite, por ejemplo, usar la librería GORM, la cual facilita las conexiones con los principales sistemas de bases de datos, simplificando el código para realizar operaciones en la base de datos. Debido a estas ventajas, se eligió el lenguaje Go para realizar la API REST del proyecto, con la cual se conectará la aplicación móvil para la carga de los datos.

4.1.2. PostgreSQL

PostgreSQL ha sido el motor de base de datos elegido para la persistencia de la información.

Las ventajas de PostgreSQL incluyen que se trata de un software gratuito y multiplataforma, que sigue el estándar SQL o que es capaz de ajustarse de forma óptima al hardware del equipo donde se encuentre. También tiene una herramienta gráfica, pgAdmin, que permite administrar de forma fácil la base de datos [10].

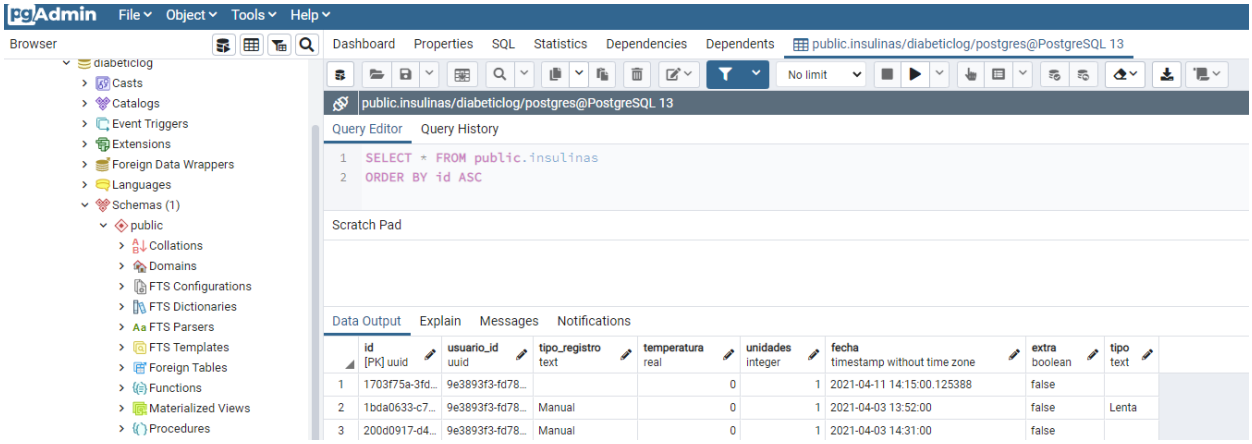

Ilustración 7 Herramienta gráfica de PostgreSQL, pgAdmin

Sin embargo, su principal desventaja es que es relativamente lento en inserciones y actualizaciones en una base de datos pequeña, ya que PostgreSQL está orientado para ambientes de alto volumen. [10]

A pesar de la desventaja, se eligió este sistema porque ya había trabajado con este motor de base de datos, y así poder centrar los esfuerzos y el aprendizaje en Android o Go, los cuales se partía con nulo conocimiento.

4.1.3. Insomnia y OpenAPI

Para las pruebas de la API realizada se usó el cliente REST Insomnia, el cual por un lado permite realizar peticiones a la API REST creada con Go, y por otro permite documentar los *endpoints* con el lenguaje OpenAPI.

Insomnia permite fácilmente realizar diferentes peticiones REST, basta con indicar el enlace donde se encuentra, el tipo de petición (POST, GET, UPDATE, etc.) y los datos de esa petición, donde admite múltiples formatos como JSON o XML.

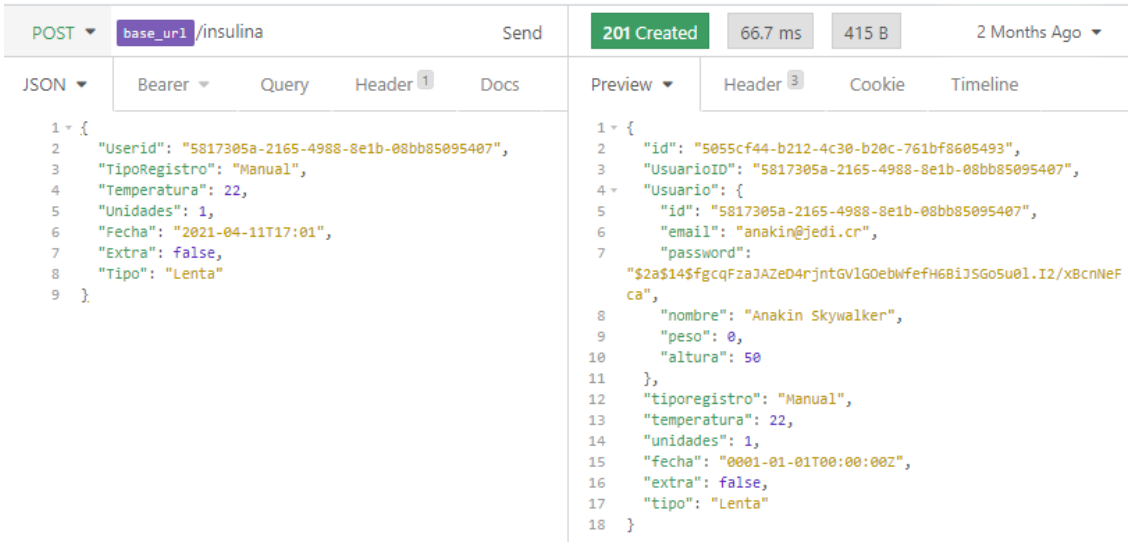

Ilustración 9 Ejemplo de petición POST realizada con Insomnia

Por otro lado, la herramienta Insomnia permite documentar la API usando la especificación OpenAPI, conocida anteriormente como Swagger. Esto permite visualizar de forma fácil los diferentes *endpoints* creados y facilita las pruebas, ya que Insomnia crea automáticamente las peticiones REST basándose en lo especificado con OpenAPI.

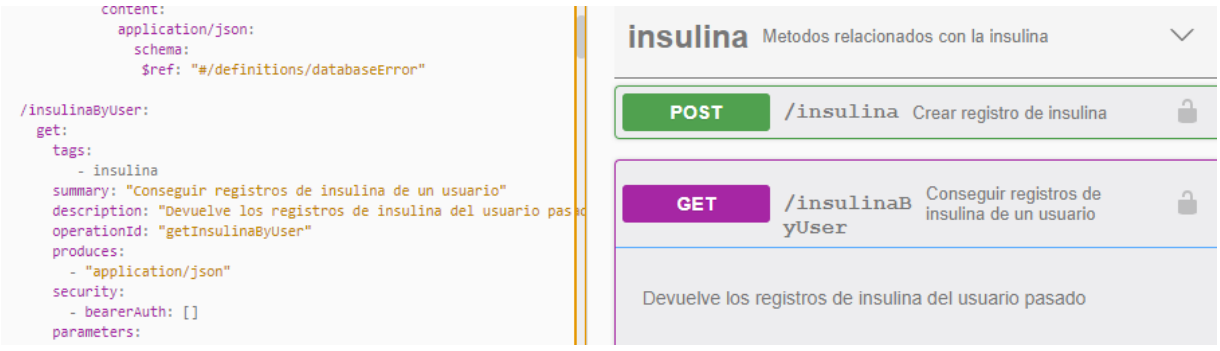

Ilustración 8 Documentación endpoints con OpenAPI

4.2. Técnicas y herramientas utilizadas en la aplicación móvil

4.2.1. Kotlin

La aplicación móvil será para el sistema operativo Android, y para su realización se ha usado el lenguaje de programación Kotlin y el entorno de desarrollo Android Studio.

El lenguaje de programación Kotlin es un proyecto de código abierto desarrollado por JetBrains con contribuciones de Google. Kotlin es un lenguaje de programación que usan más del 60% de desarrolladores de Android. Entre sus características están que es interoperable con Java, que proporciona seguridad sobre las variables nulas (lo que hace que sus aplicaciones tengan un 20% menos de probabilidades de fallar) o que es más conciso, necesitando menos líneas de código estándar. Además, el lenguaje está optimizado para el desarrollo de aplicaciones Android [11].

```
class LoginActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_login)

        val botonLogin = findViewById<View>(R.id.botonLogin)
        botonLogin.setOnClickListener {
            loginUser()
        }

        val botonRegistro = findViewById<View>(R.id.botonRegistro)
        botonRegistro.setOnClickListener {
            loadRegistroActivity()
        }
         …
      }
}
```

Ilustración 10 Fragmento del código Kotlin usado en la pantalla de inicio de sesión

Asimismo, el entorno de desarrollo por defecto de Kotlin es Android Studio, el cual está optimizado para crear aplicaciones Android. Android Studio cuenta con emulador para probar las aplicaciones creadas o la posibilidad de usar plantillas que facilitan el desarrollo, entre otra serie de características.

Por esta serie de características se eligió Kotlin frente a otros lenguajes, como Java o C++, con los que se podría haber realizado la aplicación móvil en Android.

5. Aspectos relevantes

5.1. Planificación Temporal

Para realizar la planificación temporal del proyecto se siguió la metodología ágil Scrumban, la cual combina las metodologías Scrum y Kanban.

Dentro del "Anexo I – Planificación Temporal" se detalla con más profundidad los aspectos teóricos de la planificación temporal, se resaltan las metodologías ágiles y se especifica con más detalles el enfoque elegido y cómo se ha aplicado.

5.1.1. Herramienta de planificación: Gitlab

La plataforma elegida para gestionar el proyecto es Gitlab, la cual ha servido como doble propósito: gestionar el repositorio donde está el código, y gestionar el proyecto. Esto último se ha realizado en el apartado "*issues*".

Dentro de este apartado, se crearon los diferentes requerimientos que tendrá el proyecto, con el objetivo de tener el *backlog* del proyecto. Los requerimientos se especificaban a través de *issues*, también conocidos como historias de usuario, los cuales tendrán la siguiente estructura:

- Título: Breve descripción de lo que se va a implementar o realizar
- Descripción: La cual tendrá la siguiente plantilla:
    - *As a <user, develop, other people>*: Como <Usuario, Desarrollador, otra gente>
    - *I want to <brief explanation about what the person specified in the "As a" needs>*: Quiero <breve explicación de lo que necesita la persona especificada en "*As a*">
    - *So that <why this will be useful for that person>:* De manera que <porque será útil para esa persona>
- *Acceptance Criteria*: Criterios de aceptación, serán el mínimo que deberá cumplir el *issue* para que se considere como realizado.
- *Label:* Se trata de una etiqueta que clasifica los *issues* dependiendo de qué tipo de desarrollo sea. Por ejemplo, se usaron dos principalmente: "Android" y "Server", que indicaban cuando se trataba de una tarea en el desarrollo de la aplicación y cuando del servidor.

La herramienta también contaba con otras opciones, pero que eran poco relevantes en el proyecto actual y que prácticamente no se usaron. Están eran, por ejemplo, quien tenía asignado cada issue, que siempre era el mismo, o especificar a que hito pertenecía.

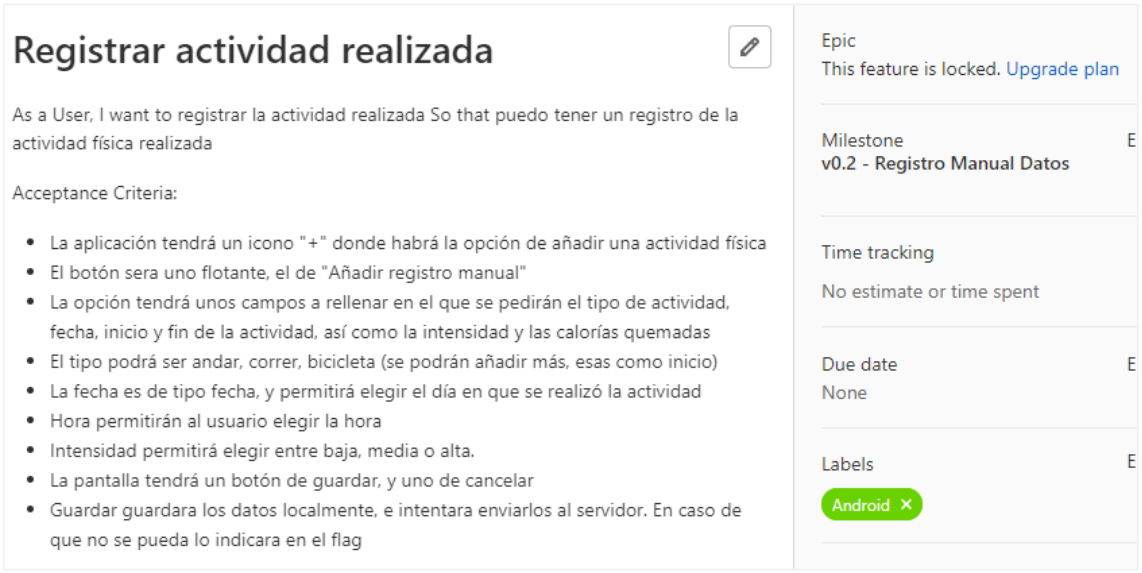

Ilustración 11 Ejemplo de issue

Los issues pasan a formar el *Product Backlog* del proyecto, los cuales se podrían visualizar en un tablero de Kanban. Donde se encuentran divididos en cuatro columnas:

- *Open:* En esta columna se encuentran todos los issues que se han creado, y que forman el *Product Backlog*. Estas tareas están pendientes de ser planeadas en un próximo *sprint.*
- *To Do:* Las tareas que se encuentran en esta columna son issues pendientes de realizar, los cuales se han planeado para realizar en el sprint actual.
- *Doing:* Indican los issues que se están realizando ahora mismo. Normalmente esta columna estaba limitada a una tarea a la vez, al existir un único desarrollador.
- *Closed:* Son los issues que se han finalizado. Para saber si una tarea ha finalizado deberá estar realizado los criterios de aceptación descritos en la misma.

En la ilustración siguiente se puede ver el tablero usado en el proyecto, con las cuatro columnas descritas anteriormente, e issues creados y/o realizados en el proyecto.

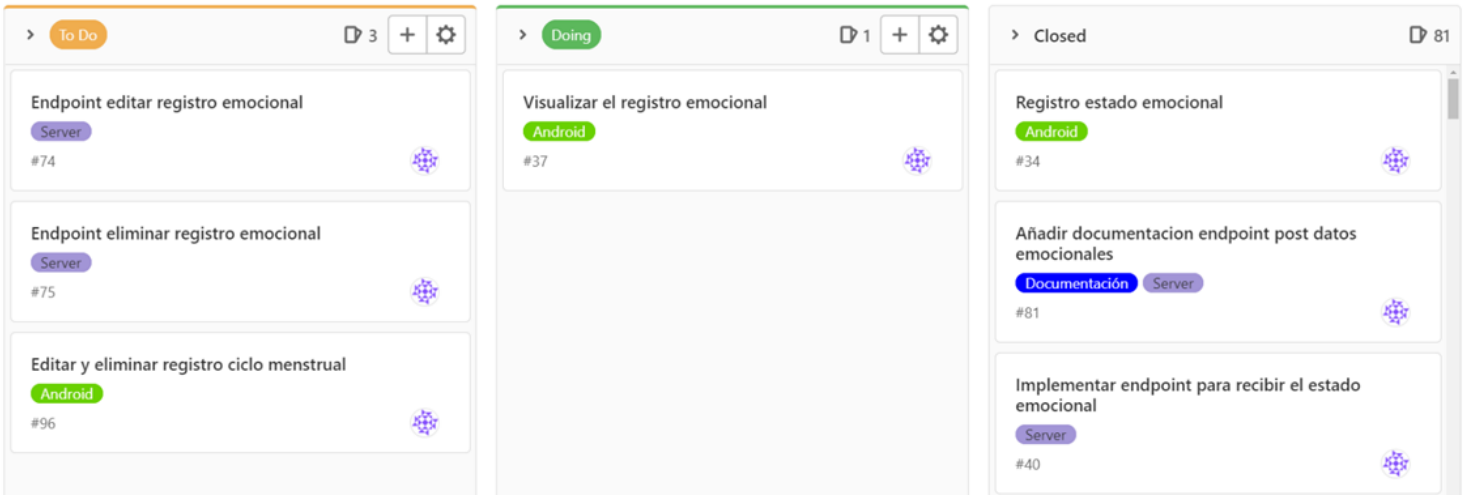

Ilustración 12 Tablero Kanban de un sprint realizado con tareas relacionadas con el estado emocional

5.1.2. Sprints

A lo largo del proyecto se han realizado 6 *sprints,* los cuales tenían una duración de tres semanas cada uno. Al inicio de cada *sprint* se tenía una reunión con los tutores del proyecto, donde se revisaba lo realizado en el *sprint* y se planificaba el siguiente.

En las revisiones del *sprint* se comentaba si se habían realizado cambios en cuanto a la planificación inicial, si se ha tenido que dividir algún *issue* para que sea más asequible de implementar, o incluso si se han añadido más tareas que las previstas inicialmente.

Para la planificación del siguiente, se priorizan las tareas a realizar, teniendo en cuenta lo que se necesita para implementar la funcionalidad deseada, y lo que dará tiempo a realizar en esas tres semanas.

En cuanto a los roles de Scrum en este proyecto, el *Product Owner* estaría representado por José Luis Vidal de la Rosa y Adrián Antonio González Pazos. En el rol del *SCRUM Master* se encontraría los tutores del proyecto; Ana Belén Gil González, Belén Pérez Lancho y José Luis Vidal de la Rosa. El equipo de desarrolladores estaría compuesto únicamente por Adrián Antonio González Pazos.

Se comenta a continuación un resumen de las actividades realizadas en los sprints:

- Sprint 1: Comenzó con la reunión inicial, también llamada *Kick Off.* Durante este sprint se realizó principalmente tareas de la fase de requisitos, análisis y diseño donde se fueron creando issues en Gitlab para realizar en posteriores sprints.
- Sprint 2: Se empezaron a realizar tareas de implementación, aunque también tareas de análisis y diseño.
- Sprint 3 y 4: Se realizaron issues realizados principalmente con la implementación.

- Sprint 5: Se realizaron issues de implementación orientados a corregir errores o mejorar funcionalidad implementada en anteriores sprints. Y se comenzó a realizar la documentación.
- Sprint 6: En el último sprint se realizaron principalmente tareas de documentación del proyecto, así como alguna de corrección de errores y mejoras en el sistema.

Por último, se muestra la planificación temporal de los sprints, así como las reuniones realizadas, mediante un diagrama de Gantt.

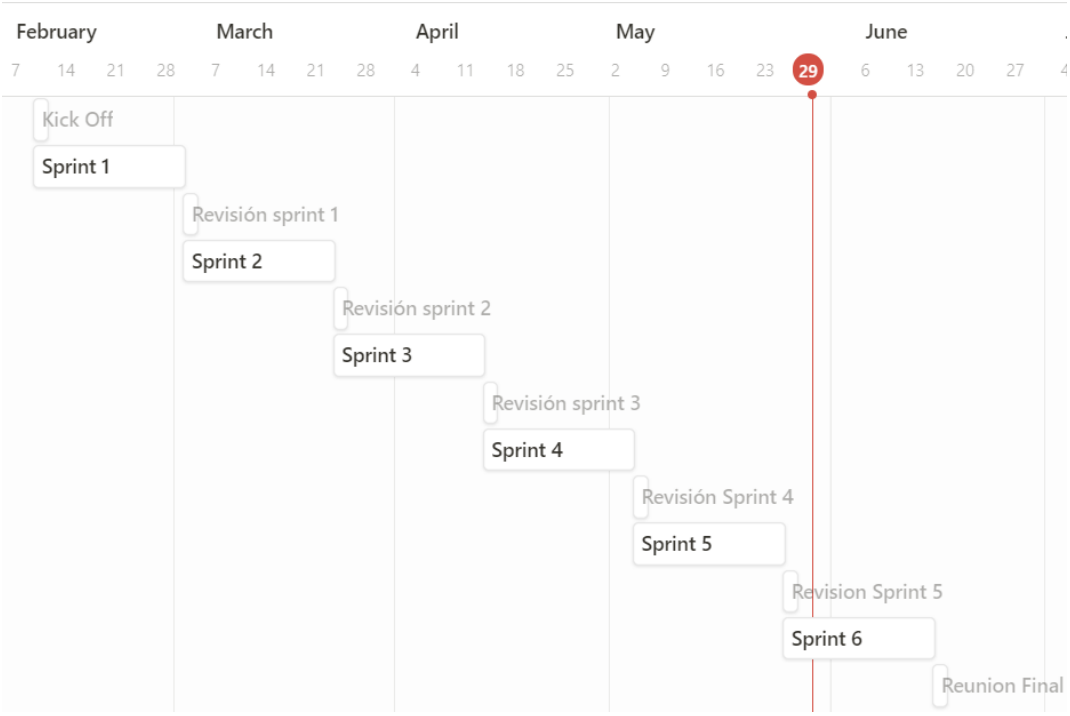

Ilustración 13 Planificación temporal sprints

5.2. Especificación de requisitos

En este punto se presenta un resumen de la especificación de requisitos llevada a cabo para el proyecto. Esta información se encuentra detallada en el "Anexo II – Especificación de requisitos", donde se recogen las tablas de los diferentes requisitos, así como los diagramas de caso de uso y su especificación.

Los primeros requisitos por detallar serían los de información, donde nos encontramos uno referente a los datos del usuario, y el resto son datos referentes a los diferentes registros que el usuario puede registrar: insulina, ingesta, actividad, estado emocional y ciclo menstrual. Se especifica ahora que datos tendrá que cada uno:

- **Usuario:** Correo electrónico, contraseña, nombre, peso y altura
- **Insulina:** Fecha de aplicación, tipo (rápida, intermedia o lenta), unidades aplicadas, si se trata de una aplicación extra o no, y la temperatura. Por el momento la incorporación de estos datos la realiza el usuario de forma manual, pero también se abre la posibilidad de que un futuro esta información provenga de un sensor, y por ello se añade la especificación del tipo de registro, que puede ser manual o sensor.
- **Ingesta:** Fecha y hora de la ingesta, número de raciones de hidrato de carbono, y un dato de tipo texto que indicará la comida ingerida.
- **Actividad:** Fecha y hora de inicio y de fin, el tipo de actividad, la intensidad (baja, media o alta) y la estimación de las calorías quemadas.
- **Estado emocional:** Fecha del registro, y tipo de emoción. Además, se añadirá un valor numérico que corresponderá a las cuatro emociones que afectan a la glucosa, las cuales son estrés (1), ansiedad (2), angustia (3) y feliz (4). Si no corresponde a ninguna de esas cuatro el valor será 0.
- **Ciclo menstrual:** Fecha de inicio y fecha de fin de la menstruación.

Descritos los requisitos de información pasamos a describir los casos de uso, los cuales tendrán cuatro actores: Usuario, Sistema (la propia aplicación móvil), el servidor, y el temporizador de notificaciones.

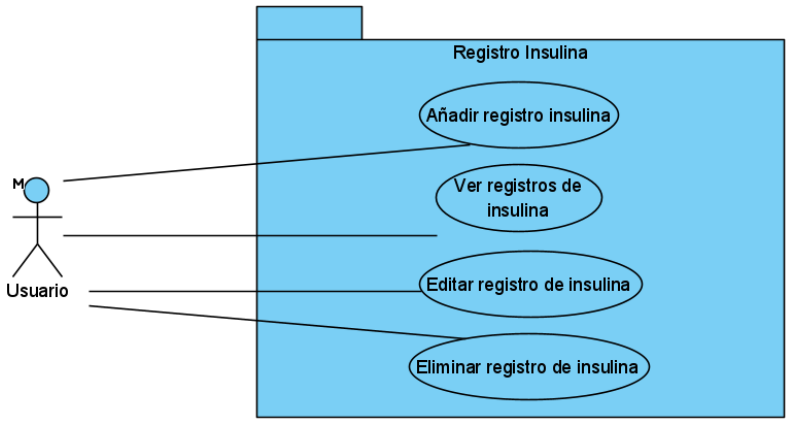

Ilustración 14 Diagrama de caso de uso "Registro insulina"

Entre los casos de uso a destacar se tendrían los referentes a los diferentes tipos de registro: añadir, ver, editar y eliminar, que se encargarían de formalizar la gestión de los datos de insulina, ingesta, actividad diaria, estado emocional y ciclo menstrual.

Existirían también los casos de uso referentes a la gestión de usuario: ver perfil de usuario y editar los datos de usuario. Asimismo, también habría casos de uso de registro, inicio sesión y cerrar sesión.

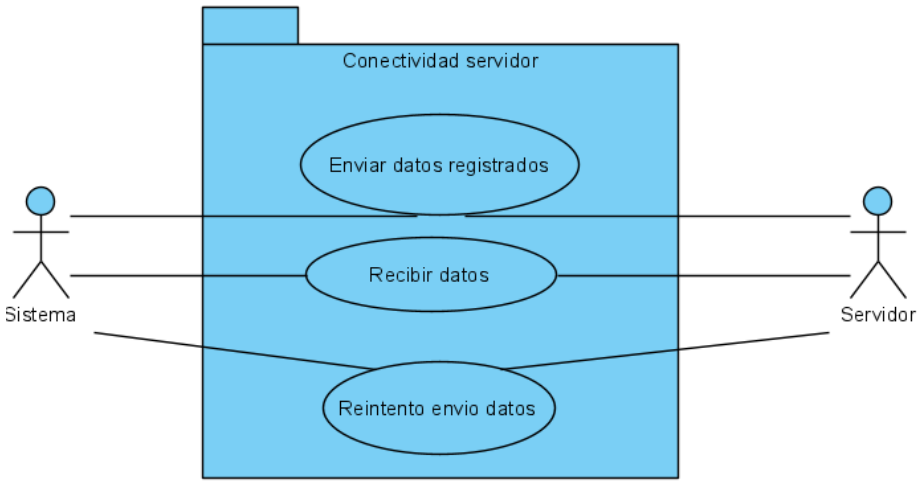

Ilustración 15 Diagrama de caso de uso "Sincronización con el servidor"

Enviar datos registrados al servidor, recibir datos del servidor y reintentar envío de datos serían los casos de uso que servirían para gestionar la sincronización con el servidor del proyecto.

Por último, para el objetivo de las notificaciones el temporizador activaría el caso de uso comprobar si hay datos, que revisaría si hay datos de insulina o ingesta en la última hora y determinaría si se tiene que mostrar la notificación, ejecutando el caso de uso correspondiente. Además, los usuarios podrán editar la hora de las notificaciones que tendrá el caso de uso correspondiente.

5.3. Análisis de requisitos

Como en el punto anterior se presenta un resumen del Anexo III: Análisis de requisitos, donde se presenta el diagrama de clases y los diagramas de secuencia de los diferentes casos de uso.

En el diagrama de clases del proyecto tendremos las clases: usuario, insulina, ingesta, actividad, emoción y ciclo, como se muestra en la siguiente ilustración.

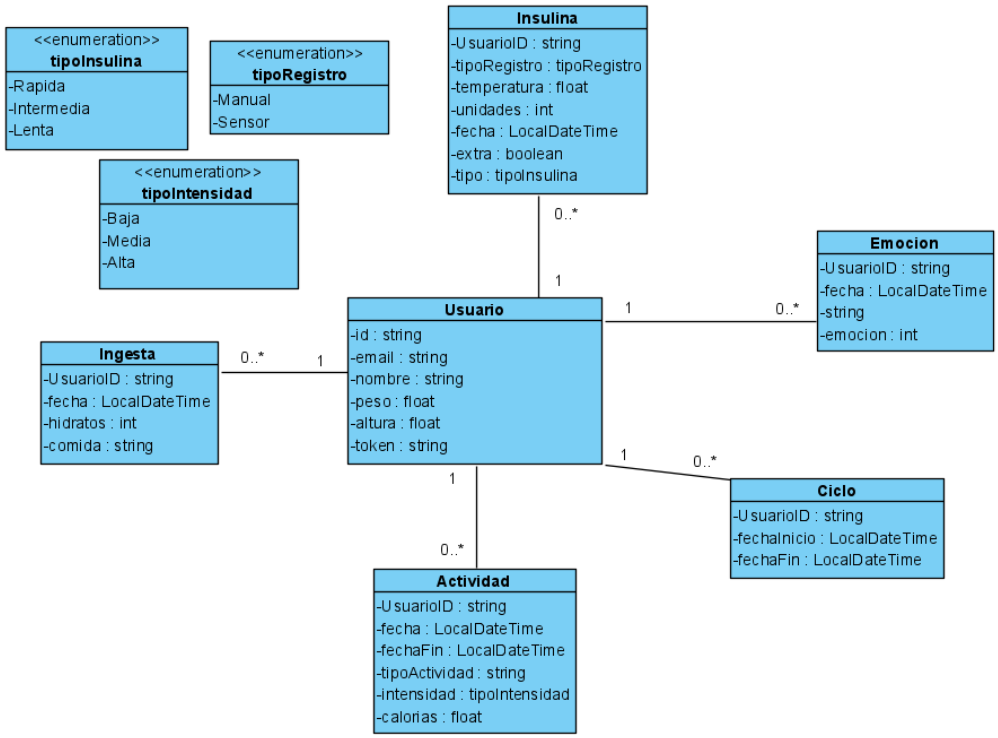

Ilustración 16 Diagrama de clases

Por otro lado, se realizaron los diagramas de secuencia de los diferentes casos de uso que ayudan a ver como se implementarían.

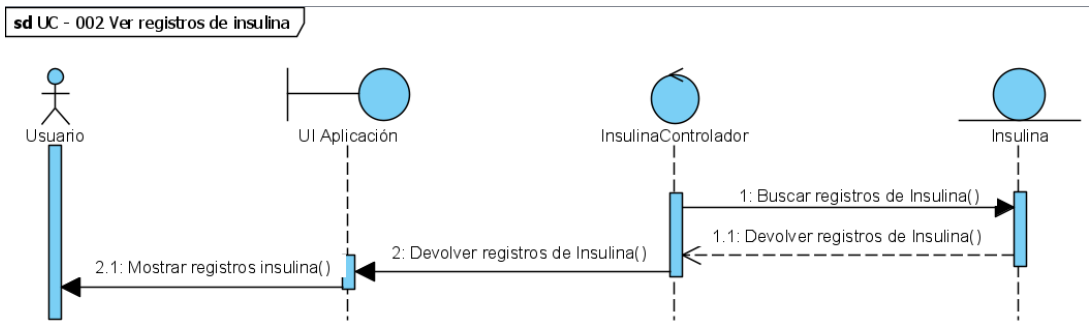

Ilustración 17 Diagrama de secuencia "Ver registros de insulina"

5.4. Diseño del sistema

En lo relativo al diseño del sistema, se comenzó por crear primero la arquitectura del proyecto, basándose en el modelo de cliente-servidor. Se obtuvo la siguiente arquitectura:

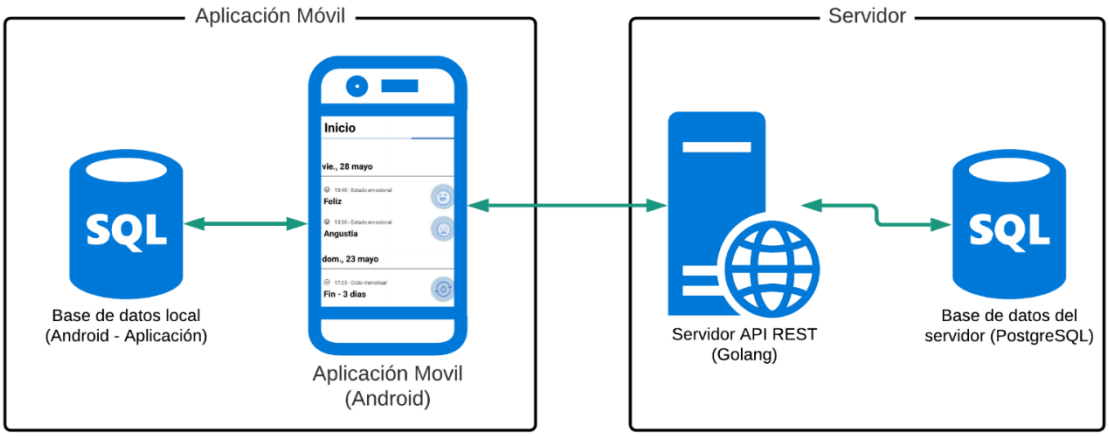

Ilustración 18 Arquitectura del sistema

De esta forma se consigue dos partes diferenciadas: la aplicación móvil (el cliente) y el servidor.

5.4.1. Diseño de la interfaz

Para el diseño de la aplicación móvil se realizó primero un prototipado en papel. Con ello se consiguió visualizar de forma rápida cómo serían las diferentes pantallas de la aplicación y cómo se realizaría la interacción. Cabe destacar que el diseño se ha basado en Material Design, que es una normativa enfocada en la visualización para el sistema operativo Android. De esta manera, el usuario va a saber usar más rápidamente la aplicación, al estar familiarizado con este diseño.

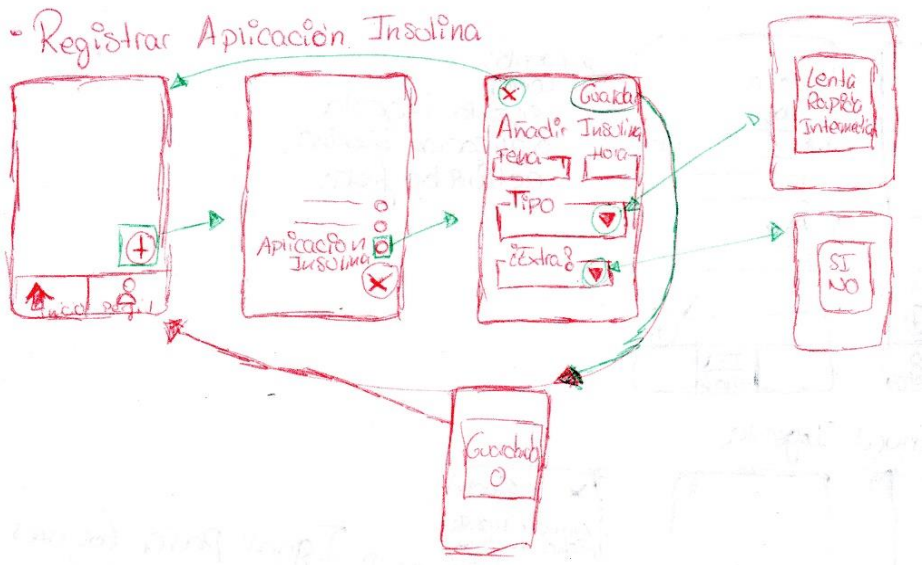

Ilustración 19 Prototipo del registro de aplicación de insulina

Una vez ideado el contenido y la distribución de los elementos la elección de colores y tipografía usada cobra importancia en el diseño de una aplicación móvil.

En el tema de colores se eligió como color principal el azul, cuya tonalidad exacta se eligió basada en el símbolo que le da la ONU a la diabetes. Este color, además, es usado en muchos logos relacionados con temas de la salud, ya que suele transmitir tranquilidad y confianza, lo cual es acorde con la aplicación del proyecto. Para el color secundario se eligió el color complementario de la tonalidad de azul elegida, dando como resultado el color tostado.

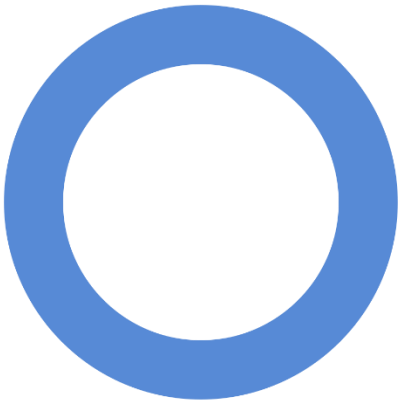

Ilustración 20 Símbolo de la ONU para la diabetes

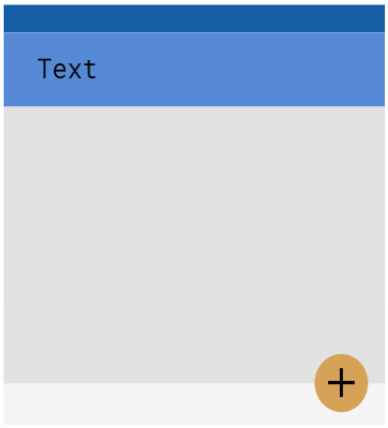

Ilustración 21 Ejemplo de uso de los colores de uso

Otro punto para tener en cuenta en la elección de colores es la accesibilidad eligiendo colores que no tengan problemas de legibilidad. Por esta razón, la tonalidad del color secundario elegida inicialmente tuvo que cambiar por una más oscura que presentara mayor legibilidad a los textos cuando se usara como fondo.

La posibilidad de seleccionar el modo oscuro de las aplicaciones supone tener que elegir también tonalidades de colores acordes a ese modo. Esto da como resultado dos paletas diferentes, una para cada modo. Se muestran a continuación las dos paletas resultantes. De izquierda a derecha, los dos primeros recuadros muestran el color primario y el secundario. Los dos siguientes serían las variantes de estos colores, usados en algunas ocasiones. Los dos siguientes son el color de texto y el color de fondo y el último, el color usado en los mensajes de error.

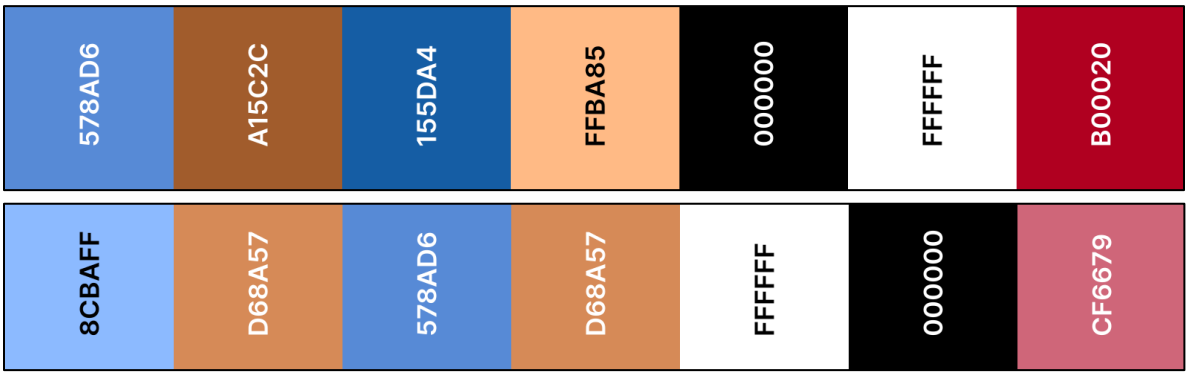

Ilustración 22 Paleta modo claro y modo oscuro

Un ejemplo de las diferencias entre el modo claro y oscuro se puede ver en la pantalla de inicio, como se muestra en las siguientes capturas:

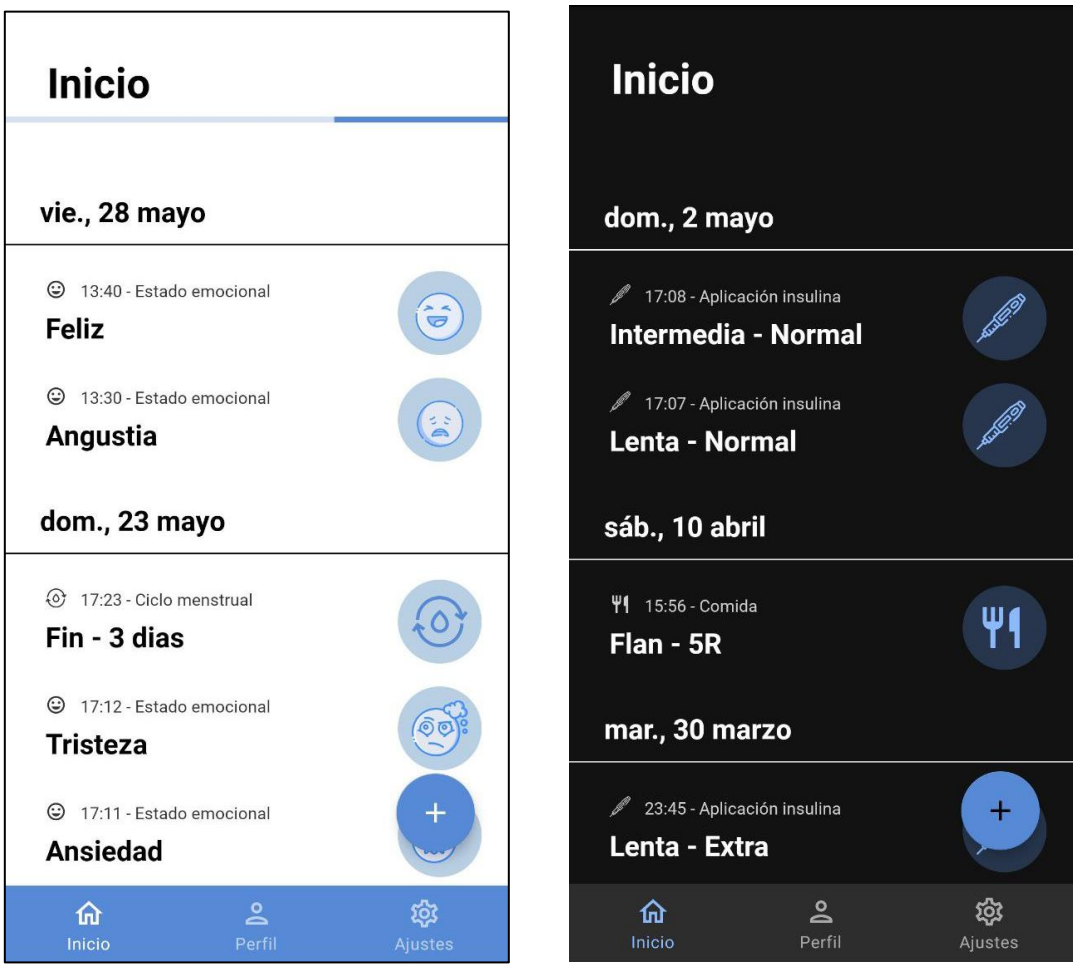

Ilustración 23 Pantalla de inicio en modo claro y oscuro

En lo relativo a la tipografía, se eligió la fuente Roboto. La decisión se basó en que es la tipografía que recomienda Google para las aplicaciones Android, debido a que es fácil de leer, entre otras características.

5.4.2. Diseño del servidor

Para el diseño del servidor, se plantean una serie de *endpoints* que deberá tener la API REST a desarrollar. Se tratará de una serie de direcciones web que al ser accedidas el servidor ejecutará la petición solicitada. La aplicación móvil, mediante los *endpoints*, enviará al servidor los datos capturados, que deberá tener *endpoints* que abarquen todos los tipos de datos de la aplicación.

La mayoría de *endpoints* estarán protegidos por un *token* de autenticación, de tal forma que solo podrán ser usados por aquellos clientes que estén registrados en el sistema. Se resume a continuación los *endpoints* según el tipo:

- **Usuario:** Se tendrá uno de registro y otro de inicio de sesión de tipo POST. El de registro creará un usuario con los datos pasados, y el de inicio de sesión comprobará los datos de correo electrónico y contraseña, devolviendo el *token* de autenticación. También habrá otro método PUT que permitirá editar los datos de usuario.
- **Insulina, ingesta, actividad, estado emocional y ciclo menstrual:** Se tendrán cuatro diferentes *endpoints* para manejar los cinco diferentes tipos de datos que gestionará el sistema. Primero se tendrá uno de tipo POST, que permitirá crear un nuevo registro del tipo indicado con los datos pasados. Para que el servidor devuelva los datos, se tendrá uno de GET, al cual se le pasará el id de usuario, y devolverá todos los datos del tipo indicado que pertenezcan a ese usuario. Para editar y eliminar los datos habrá un método PUT y otro de DELETE.

En el "Anexo IV – Diseño del sistema" se expone con mayor detalle todos estos aspectos del diseño del sistema. En dicho anexo se puede encontrar todos los prototipos de la aplicación realizados, una mayor explicación de la elección de colores, y se detallan los *endpoints* del servidor.

5.5. Implementación

5.5.1. Servidor

Como se ha mencionado en apartados anteriores, el servidor tendrá dos partes diferenciadas: la base de datos y el servidor API REST.

Para la parte de la base de datos, la persistencia de datos, se hará uso del sistema de base de datos PostgreSQL. Se guardarán todos los datos relativos a los usuarios y tendrá el siguiente diseño:

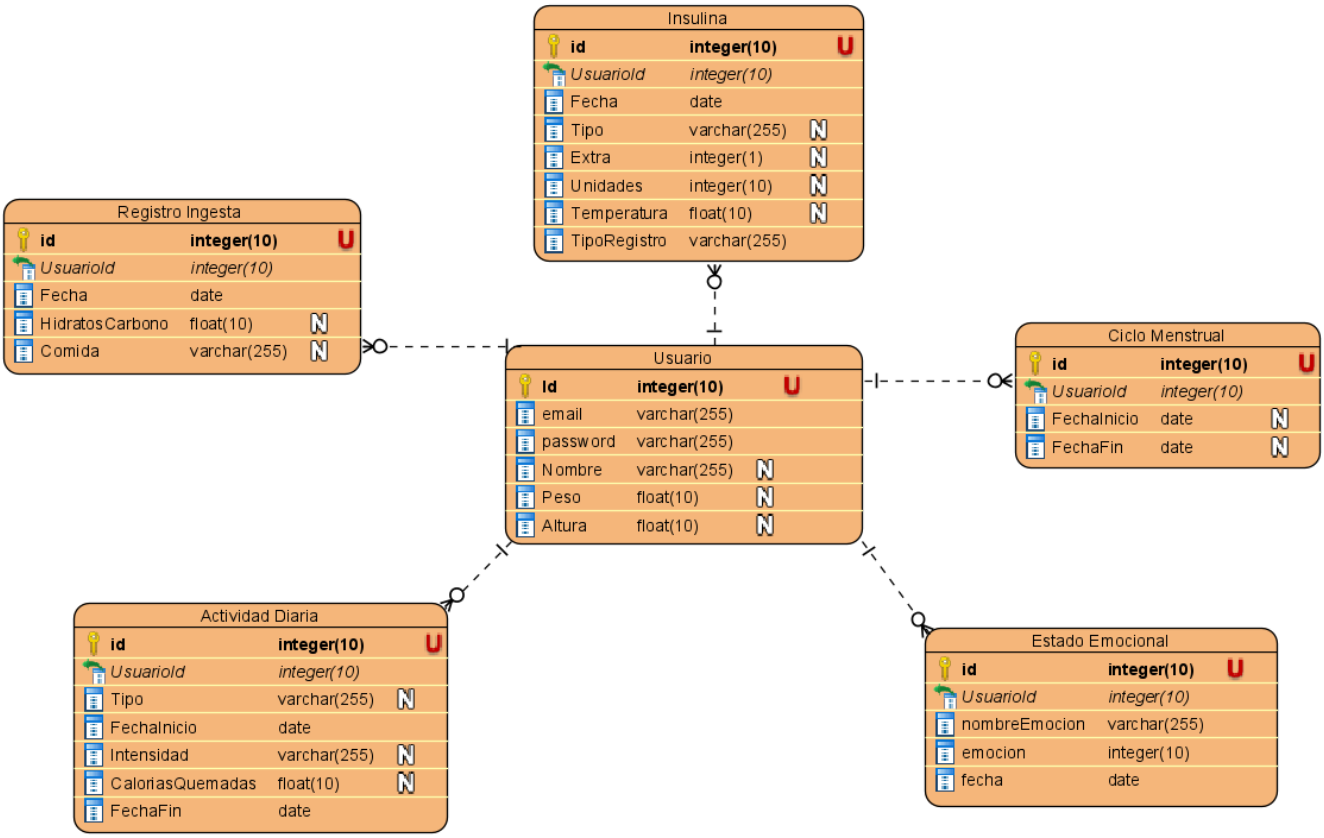

Ilustración 24 Modelo base de datos

- **Usuario:** Representa a un usuario del sistema. A cada usuario le corresponden 0 o más registros de insulina, de ingesta, de actividad diaria, de estado emocional y de ciclo menstrual.
    - **ID:** Valor alfanumérico que identifica al usuario. Se crea automáticamente la primera vez que se inserta en la base de datos. Será la clave primaria de la tabla.
    - **Email:** Correo electrónico necesario para iniciar sesión en el sistema.

    - **Password:** Contraseña necesaria para iniciar sesión en el sistema. Está almacenada de forma cifrada.
    - **Nombre:** El nombre del usuario en el sistema. Se trata de un campo opcional.
    - **Peso:** Representa en kilogramos el peso del usuario
    - **Altura:** Representa en centímetros la altura del usuario
- **Insulina:** Representa un registro de insulina. Cada registro de insulina pertenecerá a un usuario del sistema.
    - **ID:** Valor alfanumérico identifica el registro de insulina. Se crea automáticamente la primera vez que se inserta en la base de datos. Será la clave primaria de la tabla.
    - **idUsuario:** Clave foránea que se relaciona con el ID del usuario, y que relaciona el registro con el usuario al que pertenece.
    - **Fecha:** Marca de tiempo que identifica la fecha y hora en la que se realizó la aplicación de insulina.
    - **Tipo:** Representa el tipo de insulina aplicada. Puede ser de tipo lenta, rápida o intermedia.
    - **Extra:** Booleano que representa si la insulina aplicada es una dosis extra o no.
    - **Unidades:** Representa las unidades de insulina inyectadas en ese registro. Este atributo está preparado para un sensor que mida las unidades automáticamente.
    - **Temperatura:** Representa la temperatura de la insulina aplicada. Como el anterior campo, está preparado para un sensor que midiera la temperatura de la insulina aplicada.
    - **TipoRegistro:** Indica si el registro se ha realizado de forma manual o a través de un sensor.
- **Ingesta:** Representa un registro de ingesta. Cada registro de ingesta pertenecerá a un usuario de un sistema.
    - **ID:** Valor alfanumérico que identifica el registro de ingesta. Se crea automáticamente la primera vez que se inserta en la base de datos. Será la clave primaria de la tabla.
    - **idUsuario:** Clave foránea que se relaciona con el ID del usuario, y que relaciona el registro con el usuario al que pertenece.
    - **Fecha:** Fecha y hora en la que se realizó la ingesta.

- **HidratosCarbono:** Representa el número de raciones de hidratos de carbono consumidas.
- **Comida:** Valor de tipo texto que representa el nombre de la comida tomada en dicha ingesta.

- **Actividad Diaria:** Representa un registro de actividad. Cada registro de actividad pertenecerá a un usuario del sistema.
  - **ID:** Valor alfanumérico que identifica el registro de actividad. Se crea automáticamente la primera vez que se inserta en la base de datos. Será la clave primaria de la tabla.
  - **idUsuario:** Clave foránea que se relaciona con el ID del usuario, y que relaciona el registro con el usuario al que pertenece.
  - **Tipo:** Indica el nombre de la actividad realizada. Por ejemplo, correr.
  - **FechaInicio:** Fecha y hora de inicio de la actividad.
  - **Intensidad:** Indica la intensidad de la actividad en un rango de alta, media o baja.
  - **CaloriasQuemadas:** Representa las calorías quemadas en kilocalorías.
  - **FechaFin:** Fecha y hora de fin de la actividad.
- **Estado emocional:** Representa un registro de estado emocional. Cada registro pertenecerá a un usuario del sistema.
  - **ID:** Valor alfanumérico que identifica el registro de estado emocional. Se crea automáticamente la primera vez que se inserta en la base de datos. Será la clave primaria de la tabla.
  - **idUsuario:** Clave foránea que se relaciona con el ID del usuario, y que relaciona el registro con el usuario al que pertenece.
  - **NombreEmocion:** Representa que emoción se ha registrado.
  - **Emoción:** Hace referencia al número que le corresponde las cuatro emociones que afectan al índice glucémico: estrés (1), ansiedad (2), angustia (3) y feliz (4). Si no corresponde a ninguna de esa cuatro, el valor será 0.
  - **Fecha:** Fecha y hora en la que se produjo la emoción registrada.
- **Ciclo menstrual:** Representa un registro de ciclo menstrual. Cada registro pertenecerá a un usuario del sistema.
  - **ID:** Valor alfanumérico que identifica el registro de ciclo menstrual. Se crea automáticamente. Será la clave primaria de la tabla.

- **idUsuario:** Clave foránea que se relaciona con el ID del usuario, y que relaciona el registro con el usuario al que pertenece.
- **FechaInicio:** Fecha y hora de inicio de la menstruación.
- **FechaFin:** Fecha y hora de fin de la menstruación.

Para conectar la base de datos descrita anteriormente con el servidor creado con Golang, se usó la librería GORM (https://gorm.io/). Esta librería permite conectar de forma fácil los principales sistemas de bases de datos con un programa Golang.

Otro aspecto importante del servidor fue la seguridad, que tiene como objetivo que solo los usuarios autorizados puedan usar los *endpoints* de la API. Para ello se usó la librería Go-Guardian (https://github.com/shaj13/go-guardian). Cuando un usuario inicia sesión con el *endpoint* correspondiente se le devuelve un *token* aleatorio de autenticación. Al ejecutar cualquiera de los métodos Go-Guardian, antes de ejecutar el *endpoint* correspondiente, comprueba que se le pasa el *token* y que este es correcto. Si está autorizado, se ejecutará la petición solicitada. En caso contrario, devolverá un código de error 403, indicando que no está autorizado. De esta forma, se asegura que solo los usuarios registrados en el sistema podrán acceder a los datos guardados.

Durante el desarrollo del proyecto, el servidor estuvo lanzado de forma local. Suficiente para poder acceder a los métodos que ofrece desde el dispositivo móvil con el que se fueron haciendo las pruebas.

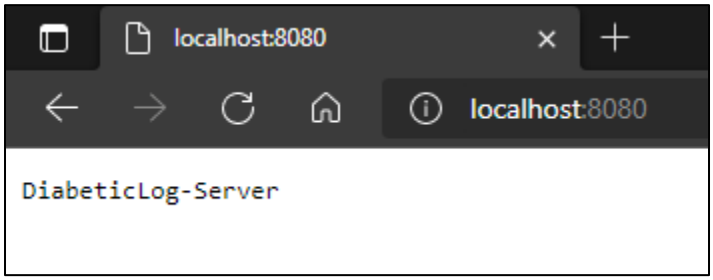

Ilustración 25 Servidor lanzado de forma local

Los *endpoints* del servidor son los descritos en al apartado 4 del "Anexo IV – Diseño del sistema". Estos métodos usan el formato JSON para el intercambio de datos, tanto para las peticiones como para las respuestas.

5.5.2. Aplicación móvil

Como se ha mencionado anteriormente la aplicación móvil se ha desarrollado en Android, usando el lenguaje de programación Kotlin y la herramienta de desarrollo Android Studio.

Para conectar la aplicación con el servidor creado anteriormente se hizo uso de la librería OkHTTP (https://square.github.io/okhttp/). Se trata de una librería que hace más rápido el envío de peticiones HTTP. De esta forma se hizo más fácil la comunicación entre el servidor y la aplicación móvil.

Como se mencionó en el apartado anterior, el servidor responde a las peticiones usando el formato JSON. Para facilitar la conversión de las respuestas recibidas en las clases creadas en la aplicación, se hizo uso de la librería Gson, la cual ha sido desarrollada por Google (https://github.com/google/gson). Esta librería convierte de forma fácil un texto en JSON en un objeto Kotlin (o Java), y viceversa.

A continuación, se adjunta algunas de las capturas del resultado final de la aplicación.

- **Inicio Sesión y Registro:** Estas dos pantallas son las primeras que el usuario encontraría al acceder a la aplicación. O bien introduciría sus datos de acceso o bien podría registrarse en el sistema. En caso de introducir algún dato incorrecto, se indicaría el tipo de error debajo del campo correspondiente. Por ejemplo, en caso de introducir un correo electrónico que no es válido se indicaría debajo de dicho campo. En caso de error de servidor, también se indicaría con un mensaje en un *snackbar*, pudiendo el usuario reintentar la petición. Si es todo correcto se accedería a la aplicación.

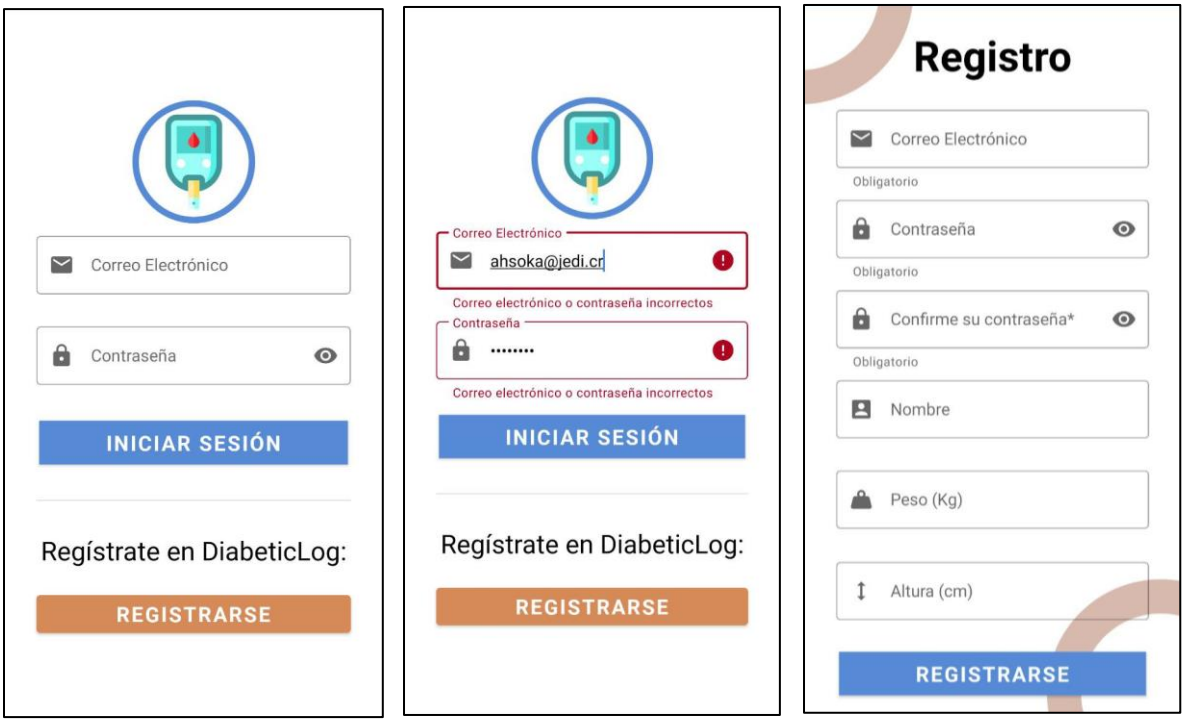

Ilustración 26 Capturas de inicio de sesión y registro

- **Inicio:** Cuando un nuevo usuario acceda por primera vez le aparecerá un mensaje de bienvenida, que de forma breve explica qué puede realizar en la aplicación. Según vaya añadiendo registros aparecerán en la pantalla de inicio, donde se mostrarán los datos más relevantes de cada registro ordenados por fecha. Si el usuario pulsa uno de los registros accederá a los detalles de este, donde podrá editarlo o eliminarlo. Por último, para añadir un nuevo registro, el usuario pulsará el "+". Se abrirá el llamado FAB (*floating Action Button*), donde seleccionará qué tipo quiere añadir, abriéndose la pantalla para ello.

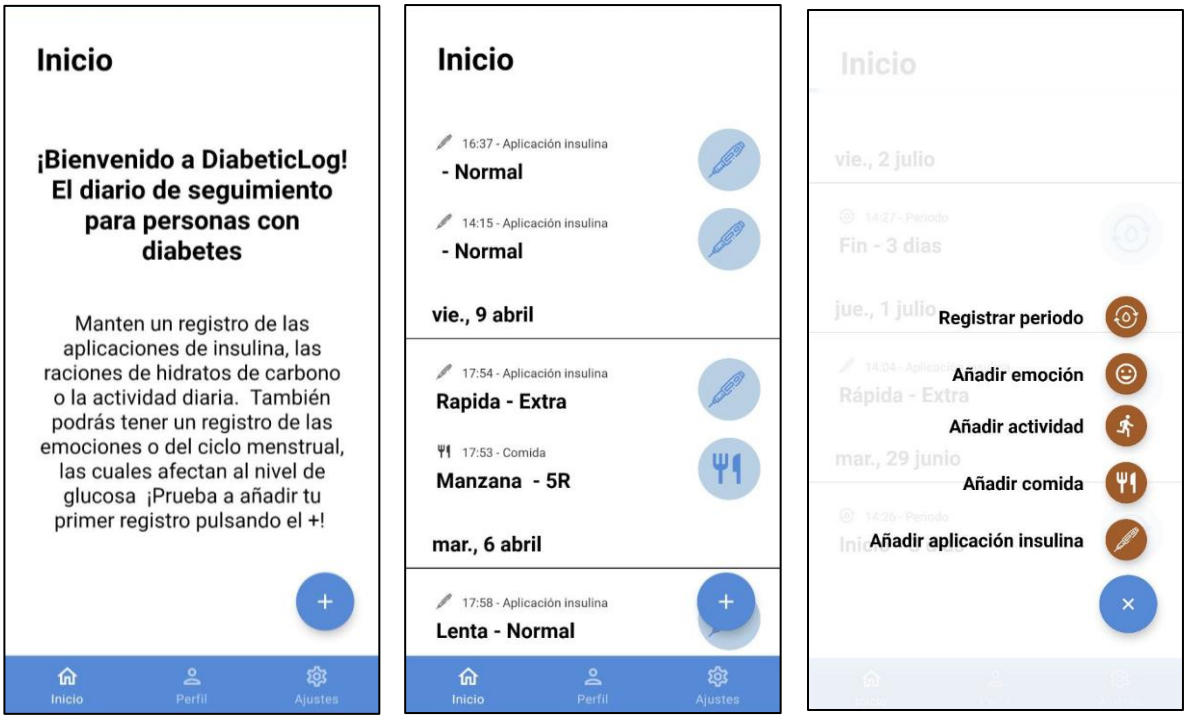

Ilustración 27 Capturas de inicio

- **Registro aplicación insulina:** Para añadir un registro de insulina, se pedirán los datos de fecha, el tipo y si es extra o no. Una vez guardado, el usuario podrá abrir los detalles desde inicio y ver los valores almacenados. El usuario además podrá borrarlo o editarlo. En caso de borrarlo, se le pedirá confirmación. En caso de editarlo, se abrirá la pantalla donde podrá editar cualquiera de los datos introducidos anteriormente.

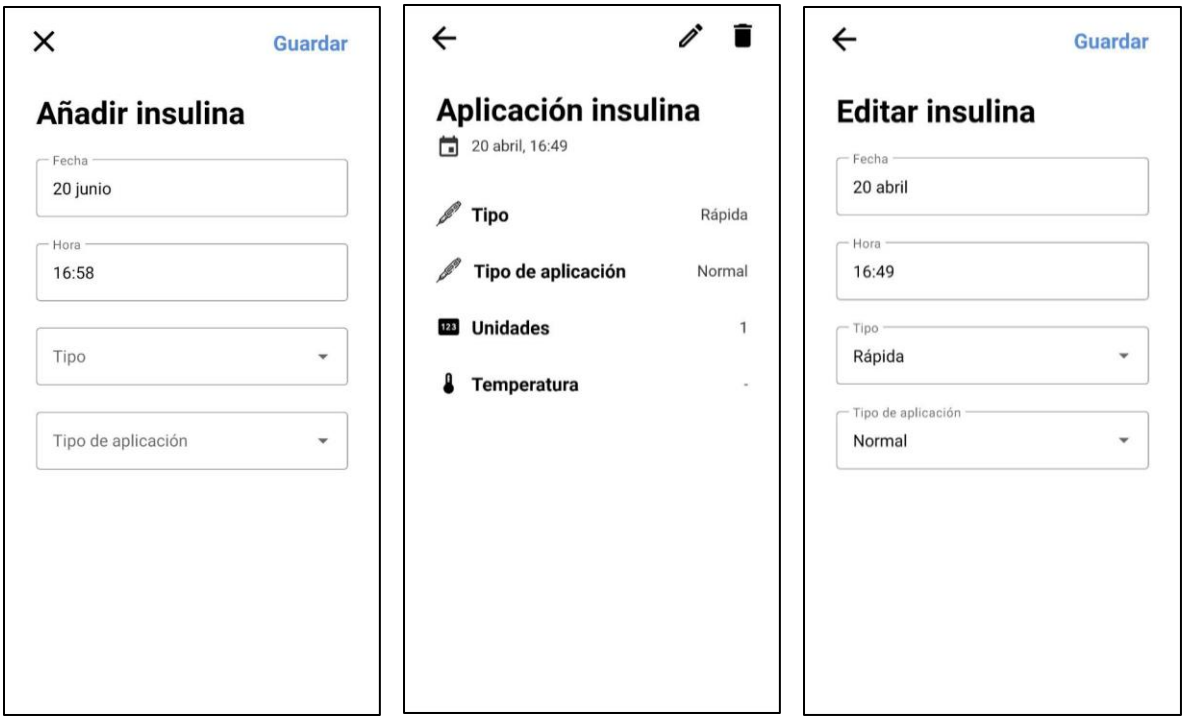

Ilustración 28 Capturas de registro de insulina

- **Ingesta:** En el caso del registro de ingesta se haría de forma similar a lo visto en la insulina. En este caso cabe destacar la opción de abrir una tabla de alimentos de la fundación para la diabetes (www.fundaciondiabetes.org), la cual facilita al usuario conocer el número de raciones de hidratos de carbono equivalente a los alimentos que ha consumido.

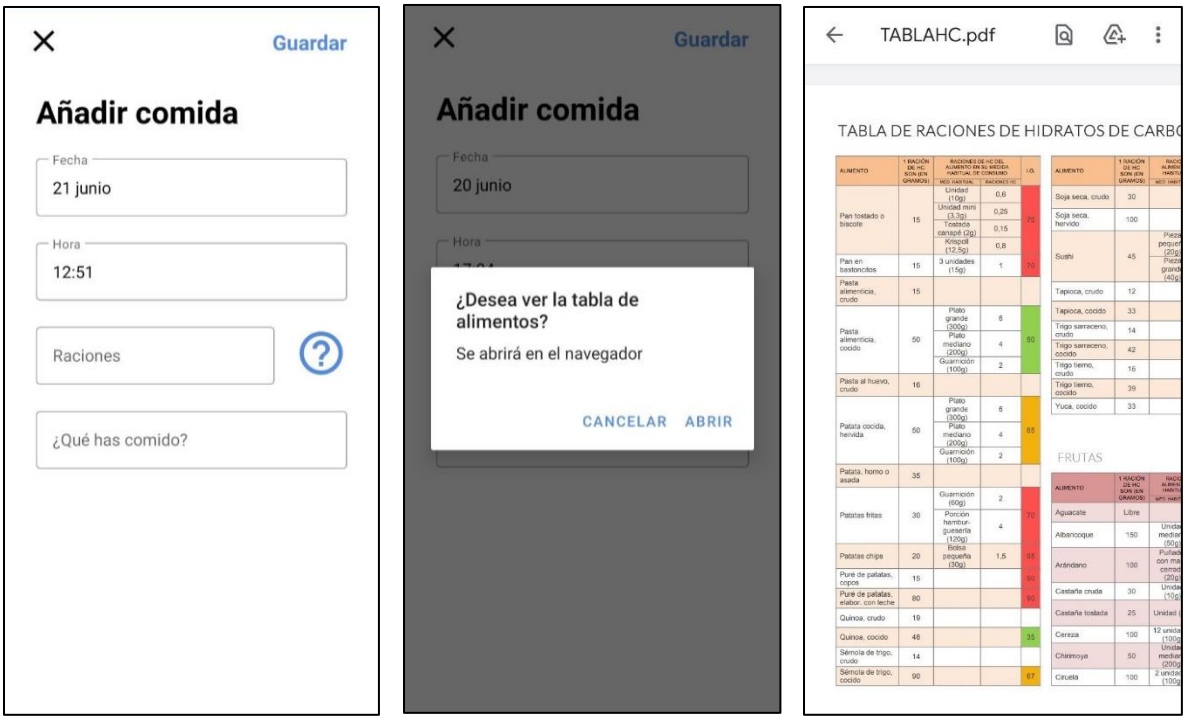

Ilustración 29 Capturas registro ingesta

- **Registro de actividad diaria:** El registro de actividad diaria se hace de la misma manera que los dos vistos anteriormente, tanto el añadir, como el editar o eliminar. Se destacan dos aspectos. Por un lado, el usuario puede guardar aquellas actividades que realiza de forma habitual, para no tener que escribirlas cada vez que las añada un nuevo registro. Por otro lado, tiene la posibilidad de abrir una calculadora de calorías quemadas (https://www.yazio.com/es/calculadora-calorias-quemadas) que le permite estimar de forma fácil las calorías quemadas en la realización de una actividad o un ejercicio.

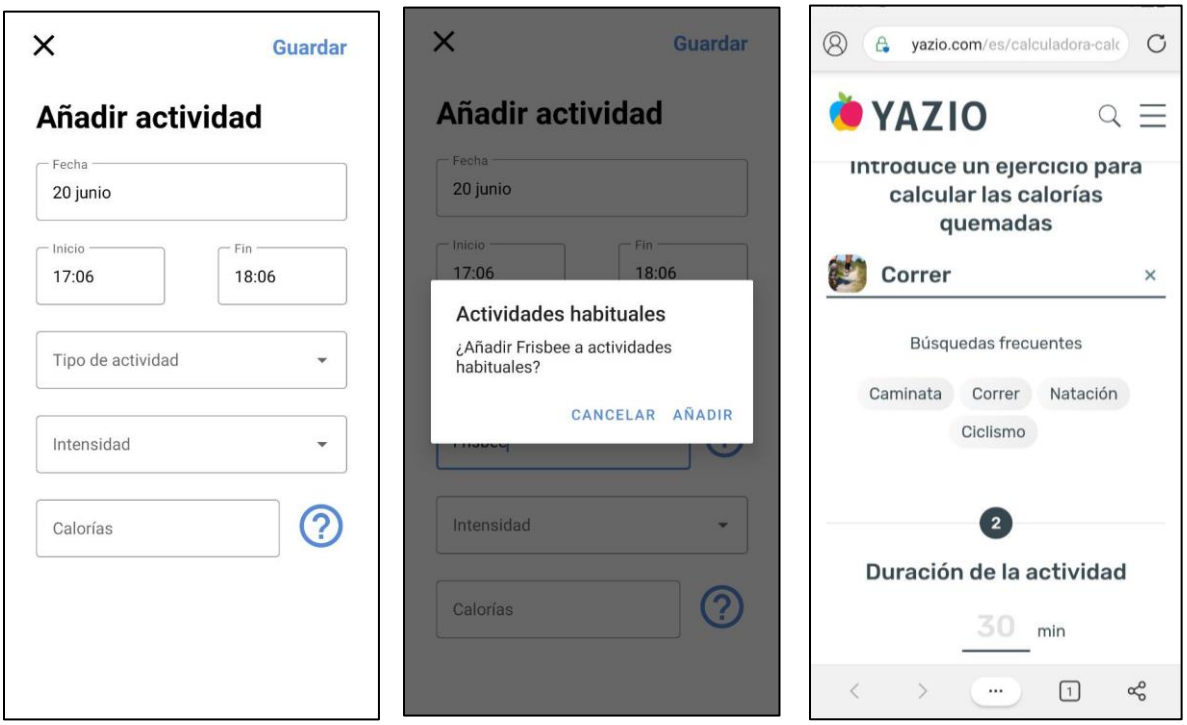

Ilustración 30 Capturas actividad diaria

- **Registrar estado emocional y duración del periodo:** Las pantallas también mantienen un aspecto similar a lo visto en los registros anteriores. Se destaca únicamente el uso de emoticonos en el estado emocional para la representación de las emociones en lugar de usar un campo de texto o un desplegable.

Ilustración 31 Capturas de estado emocional

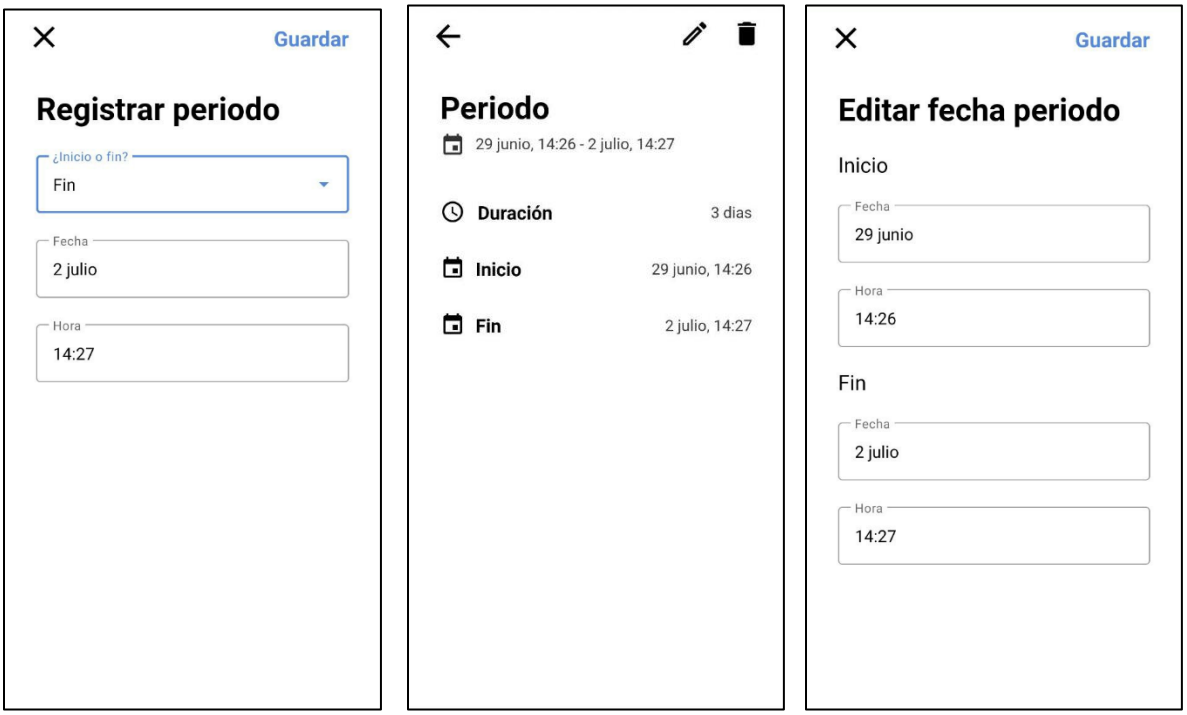

Ilustración 32 Capturas registro del periodo

- **Perfil:** En el perfil el usuario podrá consultar y editar los datos de acceso, el peso y la altura. En el caso de que intente modificar los datos de acceso se le pedirá la confirmación mediante la introducción de la contraseña actual para poder ser aplicados dichos cambios.

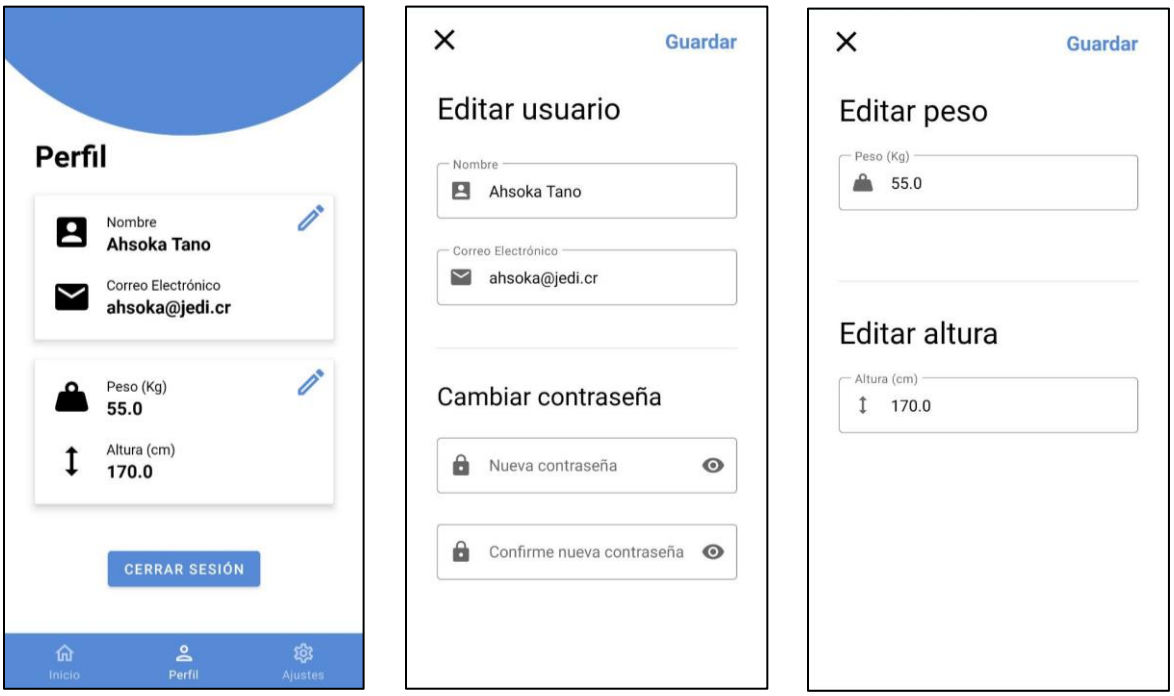

Ilustración 33 Capturas de perfil

- **Ajustes:** En esta sección el usuario podrá realizar varias configuraciones en la aplicación. En primer lugar, podrá activar o desactivar las notificaciones. Y también podrá cambiar la hora a las que están programadas. Otro aspecto que también es posible realizar es la gestión de las actividades habituales, permitiendo añadirlas, editarlas o eliminarlas. Por último, el usuario podrá desactivar la opción del registro del periodo. De esta forma, al abrir el FAB en inicio no le aparecerá esta opción. Además, el usuario encontrará en ajustes aspectos informativos de las licencias de terceros usados y un "acerca de" de la aplicación.

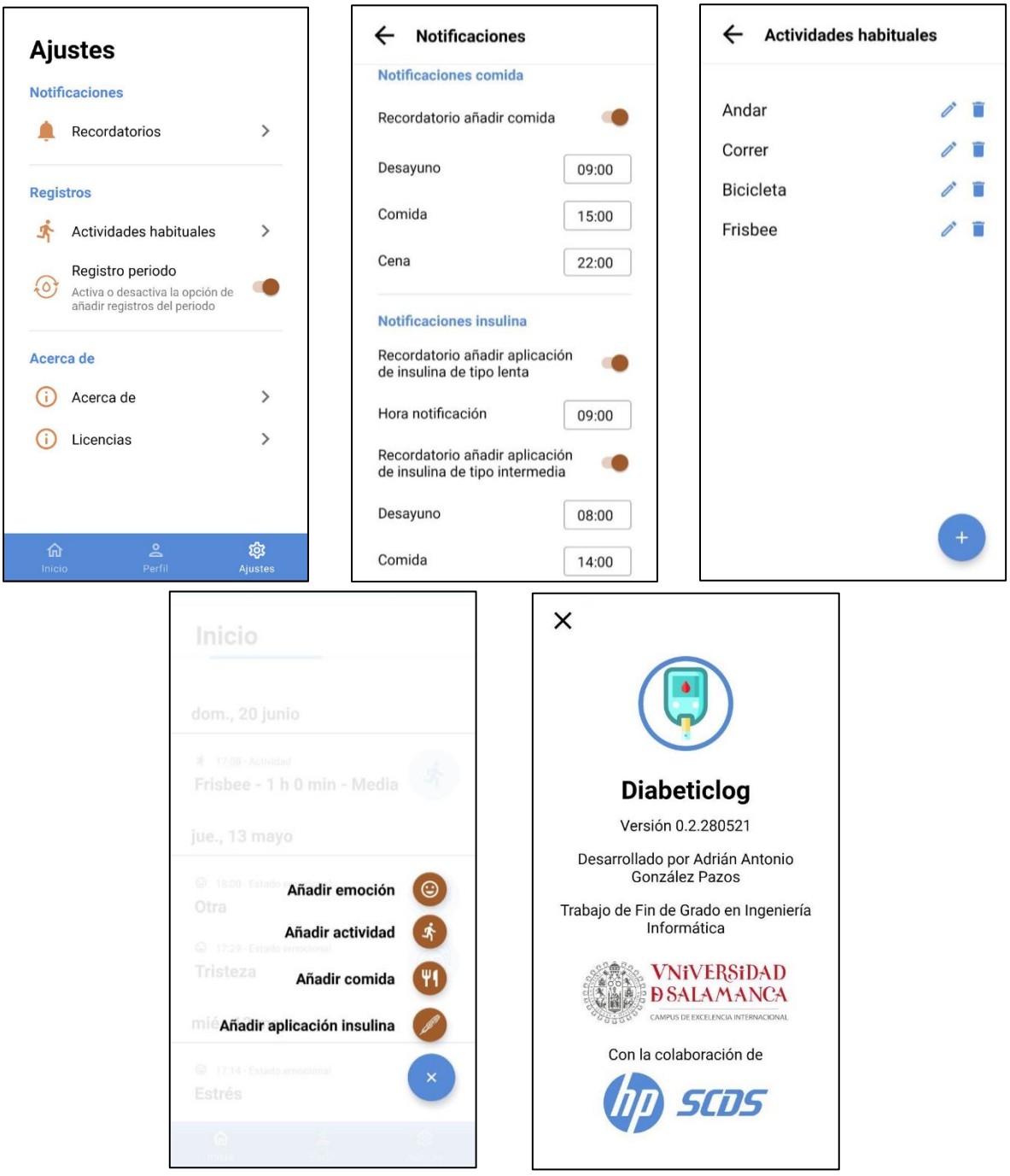

Ilustración 34 Capturas de ajustes

6. Conclusiones y trabajo futuro

6.1. Futuras líneas de trabajo

El trabajo desarrollado ha cumplido con el objetivo de crear un sistema que permita registrar diferentes datos relacionados con la diabetes. Sin embargo, solo se ha podido llegar a realizar el registro de estos datos de forma manual. Esto deja abiertas muchas posibilidades de mejora y de futuras líneas de trabajo que se puede realizar en la aplicación.

En primer lugar, a la aplicación le faltaría el poder registrar el nivel de glucosa en sangre del usuario. Una forma fácil de añadir esta opción sería como un registro manual, de forma parecida al resto de registros implementados. No obstante, la aplicación mejoraría si se le pudiera conectar el sensor FreeStyle, el cual mide de forma continua el nivel de azúcar en sangre. Esa posibilidad se barajó inicialmente, pero se ha descartado en este primer prototipo debido a que el sensor no cuenta con un SDK que permita conectarlo con la aplicación de forma sencilla. Para poder hacerlo, se tendría que realizar ingeniería inversa a la aplicación oficial del sensor, y ver cómo se podría conectar la aplicación desarrollada con el Bluetooth y NFC del sensor.

Continuando con los sensores, se mencionó en la introducción el dispositivo Insulclock. Este dispositivo permitía registrar las aplicaciones de insulina de forma automática. Otro posible trabajo futuro sería conectar este dispositivo con la aplicación. Sin embargo, y siendo el mismo caso que con FreeStyle, no es una tarea fácil de hacer al no contar con un kit de desarrollo oficial que lo permita, teniendo que realizar también ingeniería inversa.

Otra posible mejora viene relacionada con la actividad física, donde encontramos dos posibilidades. Por un lado, se podría conectar alguna pulsera de actividad o reloj inteligente que permita obtener de forma automática la actividad diaria. Un problema de esta idea es que en el mercado existen múltiples dispositivos de este tipo realizados por diferentes fabricantes que hace muy difícil abarcar todas en la aplicación. Por otro lado, otra posibilidad sería obtener los datos de actividad de Google Fit. una aplicación que tiene como propósito el registro de actividad. Su principal ventaja es que tiene una API abierta que permite a los fabricantes de pulseras de actividad enviar los datos recogidos por los dispositivos de los usuarios y a los desarrolladores obtener estos datos. De esta manera, se podría obtener de forma automática los datos de deporte de los usuarios de la aplicación.

Cabe destacar la pulsera de actividad Amazon Halo lanzada recientemente. Esta pulsera cuenta con dos micrófonos que escuchan la voz e interpretan nuestro estado emocional. La pulsera analiza la intensidad, el ritmo y el tempo de la voz, y la clasifica en varias emociones como aburrido, feliz, cariñoso o preocupado. Si se pudiera conectar esta pulsera con la aplicación desarrollada tendría un doble propósito al poder registrar de forma automática la actividad y el estado emocional del usuario. Actualmente esta pulsera solo se encuentra disponible en Estados Unidos, y una de

sus principales desventajas es que el reconocimiento de emociones solo está disponible para aquellos usuarios que paguen una suscripción mensual [12].

Además de estos aspectos relativos a la conexión de la aplicación con dispositivos externos también se encuentran otra serie de propuestas que se podrían realizar. Una de ellas es utilizar algoritmos de inteligencia artificial que permitan, por ejemplo, predecir el nivel de glucosa de un usuario. Para lograrlo, se entrenaría el algoritmo con todos los datos recogidos de forma sistemática en la aplicación. Además, si se incluyen en la aplicación datos como la edad o el género, el algoritmo podría tenerlo en cuenta, y dar resultados más precisos de esa predicción.

Otra última línea de trabajo futuro podría ser realizar una versión de la aplicación para el sistema operativo iOS, ya que ahora mismo solo las personas con diabetes que tengan un dispositivo Android podrían usarla.

Como se ha expuesto en este apartado, hay muchas posibilidades para mejorar el sistema creado. Tanto de añadir funcionalidades a la aplicación que mejoraría la experiencia del usuario como otro tipo de trabajos que parten de lo desarrollado en este proyecto.

6.2. Conclusiones

El resultado final del sistema desarrollado en este proyecto podría considerarse una aplicación funcional. Con la aplicación móvil desarrollada los usuarios pueden registrar de forma manual los datos de su administración de insulina, el número de hidratos de carbono, el ejercicio realizado, el estado emocional y el ciclo menstrual. Además, se permite al usuario gestionar estos datos pudiendo actualizarlos o eliminarlos, así como funcionalidad extra que le facilita realizar el registro, como la opción de guardar actividades habituales. Asimismo, el usuario podrá gestionar sus datos de usuario y de acceso, manteniendo un total control de la información almacenada y estando todos estos datos sincronizados con un servidor API REST. Todo este sistema desarrollado tendría valor para una persona con diabetes y se podría considerar que estaría listo para salir al mercado en forma de versión beta.

El desarrollo de este proyecto no hubiera sido posible sin poner en práctica los conocimientos aprendidos durante el transcurso del grado en Ingeniería Informática. La realización de la planificación temporal realizada fue posible gracias a los conocimientos aprendidos en la asignatura “Gestión de Proyectos”. El proyecto empezó desde las etapas iniciales de diseño, donde se usaron conceptos vistos en “Ingeniería de Software”. El diseño de la interfaz de la aplicación móvil era un aspecto importante para el proyecto y en ella tuvieron importancia las herramientas vistas en la asignatura “Interacción Persona-Ordenador”. Por último, para la realización del servidor se hizo uso de conocimientos vistos en “Sistemas Distribuidos” y en “Diseño de base de datos”, el primero para realización de los *endpoints* del servidor API REST y el segundo para la creación y gestión de la base de datos donde se guarda la información de los usuarios.

Además, la base de programación aprendida durante el transcurso del grado ha ayudado a aprender los lenguajes de programación Kotlin y Golang. En este aspecto, se han cumplido los objetivos personales de aprender los conocimientos básicos de estos dos lenguajes, siendo capaz de desarrollar tanto una aplicación móvil Android como un servidor API REST. Estos conocimientos me podrían ayudar en el futuro a ampliar las posibles salidas laborales.

Por otro lado, y como se ha comentado en el apartado anterior, el proyecto presentando tiene muchas posibilidades de trabajo futuro. Se podrían añadir los sensores Freestyle, de lectura de glucosa, e Insulclock, de aplicación de insulina, que podría ser muy útiles para los usuarios de la aplicación. Igualmente, la posibilidad de conectarla con alguna pulsera de actividad facilitaría el registro del ejercicio realizado. También se abren posibilidades de realizar algoritmos de inteligencia artificial que sirvan de utilidad para las personas con diabetes. Todas estas posibilidades llegan a la conclusión de que el ámbito de la aplicación móvil desarrollada es bastante amplio, y se hace difícil abarcarlo todo en un proyecto con una temporización limitada. Al final se ha tenido que priorizar los aspectos que se recogen en el resultado final frente a otros que hubieran sido interesantes explorar, pero que debido a limitaciones de tiempo no se han podido realizar.

Uno de los problemas que se han encontrado durante el transcurso del proyecto se relaciona con la posibilidad de añadir los sensores anteriormente mencionados a la aplicación. Estos dispositivos se tratan de hardware y software propietario que no cuentan con herramientas de desarrollo que permitan añadir estos sensores de forma sencilla a la aplicación. Es posible que a las empresas que han desarrollado estos dispositivos no les interese que otros desarrolladores puedan usar sus productos hardware en otro software que no sea el oficial, probablemente por motivos económicos. La consecuencia de esto es que la única manera de poder conectarlos es usando técnicas como ingeniería inversa. Esto dificulta desarrollar software alternativo que podría mejorar la experiencia de usuario con estos dispositivos, y ayudar a más diabéticos a controlar la enfermedad.

Finalmente, hay que mencionar que se ha podido desarrollar un producto que es totalmente funcional y que, a pesar de que se le pueden añadir más funcionalidades, dispone de una utilidad real para personas que tengan diabetes.

7. Referencias

[1] Fundación para la Diabetes, «Diabetes, una epidemia del siglo XXI» [En línea]. Available: https://www.fundaciondiabetes.org/prensa/298/diabetes-una-epidemia-del-siglo-xxi. [Último acceso: 22 junio 2021].

[2] Sociedad Española de Farmacéuticos de Atención Primaria, «Medidores continuos de glucosa: SEFAP» [En línea]. Available: https://www.sefap.org/2020/02/26/medidores-continuos-de-glucosa-que-quien-donde-cuando-como-para-que-por-que/. [Último acceso: 22 Junio 2021].

[3] G. Coulouris, «Section 2.3.1 Architectural elements» de *Distributed Systems: Concepts and Design*, Addison-Wesley, 2012, pp. 46-47.

[4] M. Civantos, «¿Qué es una API REST?: Tribalyte Technologies» 24 mayo 2021. [En línea]. Available: https://tech.tribalyte.eu/blog-que-es-una-api-rest. [Último acceso: 19 junio 2021].

[5] M. Pérez Esteso, «Qué es una API REST y para qué se utiliza: Geeky Theory» [En línea]. Available: https://geekytheory.com/que-es-una-api-rest-y-para-que-se-utiliza. [Último acceso: 19 junio 2021].

[6] R. Williams, «The four basic principles» de *The Non-Designer's Design Book*, California, Peachpit Press, 2004, p. 13.

[7] J. Nielsen, «10 Usability Heuristics for User Interface Design: Nielsen Norman Group» 24 abril 1994. [En línea]. Available: https://www.nngroup.com/articles/ten-usability-heuristics/. [Último acceso: 19 junio 2021].

[8] Wikipedia, «Go (lenguaje de programación)» [En línea]. Available: https://es.wikipedia.org/wiki/Go_(lenguaje_de_programaci%C3%B3n). [Último acceso: 31 Mayo 2021].

[9] Stack Overflow, «2020 Developer Survey» [En línea]. Available: https://insights.stackoverflow.com/survey/2020#most-loved-dreaded-and-wanted. [Último acceso: 31 Mayo 2021].

[10] J. Segovia, «Ventajas y Desventajas de PostgreSQL: TodoPostgresql» 30 Agosto 2018. [En línea]. Available: https://www.todopostgresql.com/ventajas-y-desventajas-de-postgresql/. [Último acceso: 31 Mayo 2021].

[11] Google Developers, «Desarrolla apps para Android con Kotlin» [En línea]. Available: https://developer.android.com/kotlin?hl=es. [Último acceso: 1 Junio

2021].

[12] J. García, «Amazon Halo. La pulsera fitness de Amazon tiene una app que analiza nuestra grasa corporal con un modelo 3D de nuestro cuerpo: Xataka,» 27 agosto 2020. [En línea]. Available: https://www.xataka.com/wearables/amazon-halo-caracteristicas-precio-ficha-tecnica. [Último acceso: 23 junio 2021].