Universidad Internacional de La Rioja

Escuela Superior de Ingeniería y Tecnología

Grado en Ingeniería Informática

Desarrollo de aplicación de visualización de glucemia de sistemas de insulina MiniMed™ 780G para Apple Watch.

Ubicación del código fuente:

https://github.com/arnaldopxm/GlucosApp

Trabajo fin de estudio presentado por:	Arnaldo Alberto Quintero Segura
Línea de investigación:	Diseño de aplicaciones web
Director/a:	Dr. Salvador Cobos Guzmán
Fecha:	16/03/2022

Resumen

La diabetes es una enfermedad cada vez más común, y con el avance de las tecnologías avanzan también los tratamientos para estos pacientes. En particular hablamos de las bombas de insulina de asa cerrada, las cuales controlan de manera casi autónoma la infusión de insulina del paciente.

Este trabajo enmarca el modelado, desarrollo y validación de una aplicación para la familia de dispositivos Apple Watch, siguiendo un modelo de arquitectura de software limpia. Cuya finalidad es proveer a los usuarios y cuidadores de pacientes usuarios del sistema de insulina MiniMed™ 780G, la posibilidad de ver en sus dispositivos Apple Watch los valores de su glucemia, y la tendencia de esta, a través de la aplicación desarrollada. Especialmente centrado en la lectura de datos a través de las complicaciones, mostradas en las distintas carátulas de los dispositivos Watch.

Palabras clave: Diabetes, Insulina, Apple Watch, Medtronic 780G.

Abstract

Diabetes disease is becoming more and more frequent, and as technology advances, so do the treatments for patients. Specially this focuses on closed loop insulin pumps, devices that control the patient's insulin infusion almost autonomously.

This document presents the modeling, development, and validation of an application for the Apple Watch devices family, following a clean software architecture model. The main goal is to provide the MiniMed™ 780G insulin system patients and caregivers, the ability to see on their Apple Watch devices their blood glucose and trend values through the developed application. Especially focused on the reading of data over the complications, displayed on the different faces of the Watch devices.

Keywords: Diabetes, Insulin, Apple Watch, Medtronic 780G.

Índice de contenidos

Índice de figuras

Índice de tablas

1. Introducción

En los últimos 10 años la tecnología ha evolucionado de manera muy veloz. Y gracias a esto han podido evolucionar otras áreas como las telecomunicaciones, el entretenimiento, la salud y la medicina. Siendo en estos últimos ámbitos en donde se puede enmarcar el desarrollo propuesto en este trabajo de fin de grado.

El área de la tecnología asociada a la medicina, más específicamente a los tratamientos para la diabetes, no se ha quedado atrás en estos años de evolución. Y es así como se ha logrado avanzar desde los tratamientos básicos, hasta tener dispositivos inteligentes de infusión automática de insulina para el tratamiento de pacientes de diabetes tipo 1.

1.1. Justificación del tema elegido

A pesar de la evolución y las mejoras incorporadas a lo largo del desarrollo de estos tratamientos, dada la naturaleza de rápido cambio de los ambientes tecnológicos, siempre existirán deseos de los usuarios que , por razones diversas, los fabricantes no podrán cumplir.

Partiendo de un paciente que padece de diabetes tipo 1, en adelante usuario, el cual posee un sistema de insulina MiniMed™ 780G. Y además utiliza como dispositivo móvil un iPhone, el cual tiene asociado un dispositivo de muñeca Apple Watch.

Se desea poder visualizar los valores de glucemia de la manera más cómoda posible, y qué sería más cómodo para el usuario, que poder ver estos datos críticos para su salud en su dispositivo de su muñeca.

Comenta (Féans, 2021) que existen desde hace 5 años opciones de software libre que permiten al usuario visualizar en su muñeca el valor de su glucemia. Pero el fabricante de estos dispositivos, Medtronic, se caracteriza por ser una compañía hermética con sus tecnologías y APIs, y no posee actualmente esta opción para sus usuarios.

Desde que este sistema ha salido al público, y desde que existe la aplicación móvil para ver los datos de la glucemia de este sistema. Medtronic no ha desarrollado aún una extensión de su aplicación, para dispositivos Apple Watch, que permita a los usuarios ver dichos datos en la comodidad de su muñeca, a pesar de la insistencia de sus usuarios para obtener esta facilidad.

1.2. Problema y finalidad del trabajo

El problema que intenta solventar en este trabajo es: la falta de una herramienta que permita a los usuarios del sistema MiniMed™ 780G con dispositivos Apple Watch, ver los datos de su glucemia, obtenidos del sistema 780G, en la comodidad de su muñeca.

A lo largo del desarrollo de este trabajo se planteará una solución de software al problema propuesto anteriormente, esta solución estará específicamente diseñada para dispositivos Apple iPhone y Apple Watch, y proporcionará a los usuarios el acceso a sus valores de glucemia en sus dispositivos de muñeca, cerrando así la brecha actualmente existente en este grupo específico de usuarios.

1.3. Objetivos del TFE

El objetivo de este trabajo final de grado es brindar una herramienta a todos los usuarios de Apple Watch, que sean también usuarios del sistema MiniMed™ 780G, mediante un desarrollo de software con la intención de brindar una facilidad a la hora de ver los valores de glucemia.

Para ello se planteará un objetivo general del proyecto, el cuál será subdividido a su vez en objetivos específicos. Donde cada uno será por sí mismo realizable, medible y verificable; y en conjunto, todos estos objetivos específicos, satisfarán el objetivo general.

1.3.1. Objetivo General

El objetivo general de este trabajo de fin de grado es: el diseño, desarrollo y validación de una aplicación para dispositivos Apple iPhone y Apple Watch, dirigida a cuidadores y pacientes portadores de sistemas de insulina MiniMed™ 780G.

Esta aplicación permitirá a sus usuarios la lectura de su glucemia actual, así como las tendencias de subida o bajada de esta. Obtenidas a través de la plataforma CareLink™ y mostradas en la comodidad de su muñeca, específicamente en las complicaciones de los dispositivos Apple Watch.

A grandes rasgos, la intención de la aplicación es convertirse en una nueva herramienta informativa para el paciente, brindándole a este mayor comodidad y flexibilidad a la hora de monitorizar sus niveles de glucemia.

1.3.2. Objetivos específicos

Para tener un mejor entendimiento de los objetivos a desarrollar en este trabajo, y de lo que se le presentará al usuario final, se procede a descomponer el objetivo general en varios objetivos específicos.

De esta manera se obtiene una lista de elementos, los cuales son por sí mismos realizables, valorables y auditables. Que serán la guía estructural a lo largo de todo el proceso de desarrollo del software.

También se incorporan en la lista de objetivos específicos los conocimientos transversales que se obtendrán a lo largo del proceso de desarrollo de la aplicación. Como podría ser el aprendizaje de nuevos lenguajes de programación, o la utilización de arquitecturas o metodologías durante el proceso de desarrollo. A continuación, se presenta la lista de objetivos específicos propuestos para el desarrollo de software en cuestión:

- Adquirir conocimientos sobre el desarrollo de aplicaciones para dispositivos Apple iPhone y Apple Watch.
- Adquirir conocimientos sobre el lenguaje de programación Swift.
- Mostrar los valores de glucemia del sistema MiniMed™ 780G del paciente en la aplicación a desarrollar.
- Mostrar las tendencias de subida y bajada de la glucemia del paciente en la aplicación a desarrollar.
- Crear una complicación que permita al usuario ver los valores de glucemia y tendencias desde una carátula del Apple Watch, a través de complicaciones.
- Realizar el desarrollo y planificación del software, siguiendo metodologías ágiles.
- Realizar el desarrollo de software utilizando una arquitectura que permita que el código sea fácilmente validable.

2. Marco Teórico

El desarrollo de este trabajo se enfoca en pacientes diabéticos, concretamente en pacientes usuarios de sistemas de insulina MiniMed™ 780G. Para entender cómo contribuye el desarrollo de este software a estos usuarios, se debe estudiar todo el contexto alrededor de este amplio campo entre la ingeniería y la medicina.

Para proveer de contexto al desarrollo de software propuesto, se procederá a revisar la diabetes como enfermedad crónica, la evolución de sus tratamientos a lo largo de los años, enfocado en las bombas y sistemas de insulina. Y los efectos de la informática, y del software en general, en los tratamientos de los pacientes.

2.1. Diabetes Mellitus

El término diabetes proviene del griego, y significa "correr a través". Dicho término fue acuñado en la antigua Grecia (250 a.C.) por Apolonio de Menfis (Cámara Argentina de Especialidades Medicinales, 2019). Pero existen descripciones que concuerdan con la diabetes desde una época mucho más antigua, como por ejemplo el Papiro de Ebers (1550 a.C.), donde se puede encontrar una descripción de los síntomas de esta enfermedad. Siendo este uno de los primeros precedentes conocidos de lo que podría ser esta enfermedad.

Según la OMS (Organización Mundial de la Salud, 2021) la diabetes mellitus, comúnmente llamada diabetes, es una enfermedad crónica que sucede cuando el páncreas de un individuo no secreta suficiente insulina, o cuando el organismo de este individuo no utiliza eficazmente la insulina que produce.

A lo largo de los años, la cantidad de personas con diabetes ha ido aumentando considerablemente, y la edad de las personas afectadas ha ido cambiando. Pasando de ser una enfermedad predominante en personas mayores, a ser cada vez más común en personas más jóvenes (International Diabetes Federation, 2021).

Así como se ha visto esta evolución a lo largo del tiempo, los tratamientos y las opciones para los pacientes también han ido evolucionando y mejorando para intentar hacer que la vida de estos pacientes sea como la vida de una persona sin esta condición.

Los síntomas principales que puede presentar una persona diabética vienen asociados a los desniveles en sus valores glucémicos. El biólogo (Campillo, 2019) expone que la glucemia es una de las variables más importantes que regulan nuestro cuerpo. Haciendo referencia a la concentración de glucosa libre en la sangre, para los cuales los valores entre 70 y 100 mg/dL. son considerados normales.

A partir de esto, (Campillo, 2019) denota los términos hipoglucemia e hiperglucemia, donde se dice que un paciente tiene una hipoglucemia cuando su valor glucémico se encuentra por debajo de 70 mg/dL. Y un paciente tiene una hiperglucemia, cuando sus valores de glucemia se encuentran por encima de 126 mg/dL.

Cualquiera de estos trastornos, sostenidos por largos períodos de tiempo en el cuerpo, pueden llegar a causar daños en órganos del cuerpo. Tales como neuropatías, nefropatías, enfermedades cardiovasculares, problemas en las retinas, e incluso amputaciones de miembros de la parte baja (International Diabetes Federation, 2021). Todos estos trastornos pueden ser controlados y prevenidos mediante un correcto tratamiento y control.

2.2. Tipos de diabetes

Dentro de la diabetes se pueden encontrar distintos tipos, de los cuales el tipo 1 y el tipo 2 son los más comunes, a pesar de existir otros tipos. Cada uno de estos tipos se puede caracterizar según los niveles de insulina que produce el páncreas, entre otros factores. A continuación, se explican de manera general los dos tipos más comunes de diabetes, y se mencionan otros tipos de diabetes menos comunes, pero igual de importantes.

2.2.1. Diabetes tipo 1

La IDF (International Diabetes Federation, 2021) expone que la diabetes es generada por un proceso autoinmune. El sistema inmunológico del cuerpo ataca las células beta, productoras de insulina, dando como resultado una producción mínima o incluso nula de insulina. Las causas aún no son del todo comprendidas, aunque se cree que se debe a una combinación de factores genéticos y factores ambientales.

Por otro lado, la OMS (Organización Mundial de la Salud, 2021) define la diabetes tipo 1 como una enfermedad caracterizada por un déficit en la producción de insulina, cuyo tratamiento necesita obligatoriamente de una administración diaria de insulina.

Se estima que para el 2021 un 10% de toda la población, más de 1,2 millones de niños y adolescentes de entre 0 y 19 años, padecían de diabetes tipo 1. Esta condición puede afectar a personas de cualquier edad, aunque usualmente se desarrolla en niños y adultos jóvenes (International Diabetes Federation, 2021).

La IDF indica que alrededor de 240 millones de adultos viven con diabetes y no han sido diagnosticados. Esta condición genera alrededor de 966.000 millones de dólares en gastos de salud, y ha causado cerca 6,7 millones de muertes (International Diabetes Federation, 2021).

2.2.2. Diabetes tipo 2

La diabetes tipo 2 es el tipo de diabetes más común en el mundo, con un porcentaje estimado del 90% de toda la población diabética del mundo (International Diabetes Federation, 2021). La IDF define la diabetes tipo 2 como la incapacidad de las células del cuerpo para responder correctamente a la insulina. Condición conocida también como resistencia a la insulina, la cual conlleva a producción de insulina más alta, para intentar equilibrar el organismo, generando eventualmente daños en las células beta que producen la insulina.

De acuerdo con la OMS (Organización Mundial de la Salud, 2021), la diabetes tipo 2 se caracteriza por un uso ineficaz de la insulina producida por el organismo. Mayormente debido a exceso de peso y falta de actividad física.

Actualmente, alrededor de 541 millones de adultos están en riesgo de desarrollar diabetes tipo 2 debido a los factores de riesgo asociados con esta enfermedad, tales como: familiares con diabetes, sobrepeso, dieta poco equilibrada, etnicidad, alta presión sanguínea, entre otros (International Diabetes Federation, 2021).

De esta misma manera, la IDF (International Diabetes Federation, 2021) estima que el 50% de los casos de diabetes tipo 2 pueden ser prevenidos a través de una dieta sana y actividad física, factores esenciales para mantener los niveles de glucemia bajo control.

2.2.3. Otros tipos

Además de los dos mencionados anteriormente, existen otros tipos de diabetes. Según la (Fundación para la Diabetes Novo Nordisk, 2020) podemos destacar: la diabetes gestacional, la cual es una intolerancia a la glucosa producida durante el embarazo.

La diabetes relacionada con fibrosis quística: como consecuencia de esta enfermedad que afecta a distintos órganos, entre ellos el páncreas. O La diabetes secundaria a medicamentos, los cuales podrían alterar la secreción de insulina, o la producción de esta (Fundación para la Diabetes Novo Nordisk, 2020).

Por último, (Fundación para la Diabetes Novo Nordisk, 2020) nos presenta la diabetes MODY, la cual a su vez se clasifica en distintos tipos, de los cuales actualmente se conocen 7. Esta es producida por defectos genéticos de las células beta, encargadas de la secreción de insulina, afectando su cantidad de secreción, pero sin afectar su acción.

2.3. Tratamientos para la diabetes

Teniendo registros de esta condición desde hace más de 3000 años, no fue hasta 1921 que se descubrió la insulina (Novalab, 2018) como una posible esperanza de una cura o tratamiento. Y no es hasta 1922 que se comienza a utilizar la insulina cómo tratamiento en un paciente con diabetes tipo 1.

Para la diabetes tipo 2 se emplean, normalmente fármacos de vía oral que estimulan el páncreas o que bajan el nivel de glucosa en la sangre (Cámara Argentina de Especialidades Medicinales, 2019). Pero tal y como dice la (Fundación para la Diabetes Novo Nordisk, 2020), se puede llegar a utilizar insulina en pacientes de diabetes tipo 2 en casos necesarios.

La insulina es el tratamiento principal para pacientes de diabetes tipo 1, la cual inicialmente solo se encontraba disponible en forma de jeringas de distintos tipos. Es a partir de 1960 que se comienzan a ver los primeros prototipos de infusión continua de manera subcutánea (González Fernández, 2019). A partir de este momento comienza la evolución de lo que serían las bombas de insulina como tratamiento para la diabetes. Pero no es hasta 25 años después, en 1985, que se aprueba el uso de bombas de insulina como una terapia intensiva para tratamiento de pacientes de diabetes tipo 1 (González Fernández, 2019).

Además de la insulina, para pacientes que sufren de hipoglucemias graves, se utiliza el glucagón como medicamento también inyectable, para generar un incremento en los niveles de glucosa de manera rápida, en un tiempo entre 5 y 10 minutos (Suárez, 2017).

2.4. Bombas de Insulina

Una bomba de insulina es un pequeño dispositivo móvil, que intenta imitar la funcionalidad de un páncreas sano, a través de una ISCI (infusión subcutánea continua de insulina) adaptándose a las necesidades del cuerpo. La bomba de insulina brinda al paciente múltiples beneficios clínicos tales como: menos hipoglucemias y mayor estabilidad glucémica. Todo esto con una dosificación más fácil, exacta y controlada; y una mayor flexibilidad (Medtronic, 2017).

El funcionamiento de las bombas de insulina se basa en dos tipos de infusión de insulina: los bolos, también llamados bolus, y la infusión basal. Según (Sáez de Fuente, y otros, 2008) la infusión basal es una infusión continua de insulina rápida regular, para mantener los valores de glucemia controlados a lo largo del día. Y los bolos son dosis extras para cubrir las necesidades al momento de ingerir alimentos o en caso de hiperglucemias.

2.5. MCG: Monitoreo Continuo de Glucosa

Un sistema de monitoreo continuo de glucosa, de abreviatura MCG, es un sistema que "permite controlar continuamente tus niveles de glucemia y realizar así un seguimiento de las tendencias y los patrones.” (Medtronic, 2021b).

Figura 1. MCG Guardian® REAL-TIME y bomba de insulina (Thomas, 2009).

El cual “requiere normalmente de un pequeño equipo, que consta de un transmisor y un sensor que mide los niveles de glucosa intersticial cada 5 minutos" (Medtronic, 2021b), tal y como se muestra en la Figura 1. Permitiendo así a los pacientes tener los valores de glucemia al alcance las 24 horas del día, sin necesidad de una intervención física para obtener la medición.

2.6. Sistemas de Infusión de insulina

Un sistema de infusión de insulina es un sistema electromecánico de administración de insulina, compuesto por al menos una ISCI, conectada al paciente, y opcionalmente: un MCG, también conectado al paciente, y un algoritmo de control, encargado del proceso y control y coordinación entre la ISCI y el MCG (González Fernández, 2019).

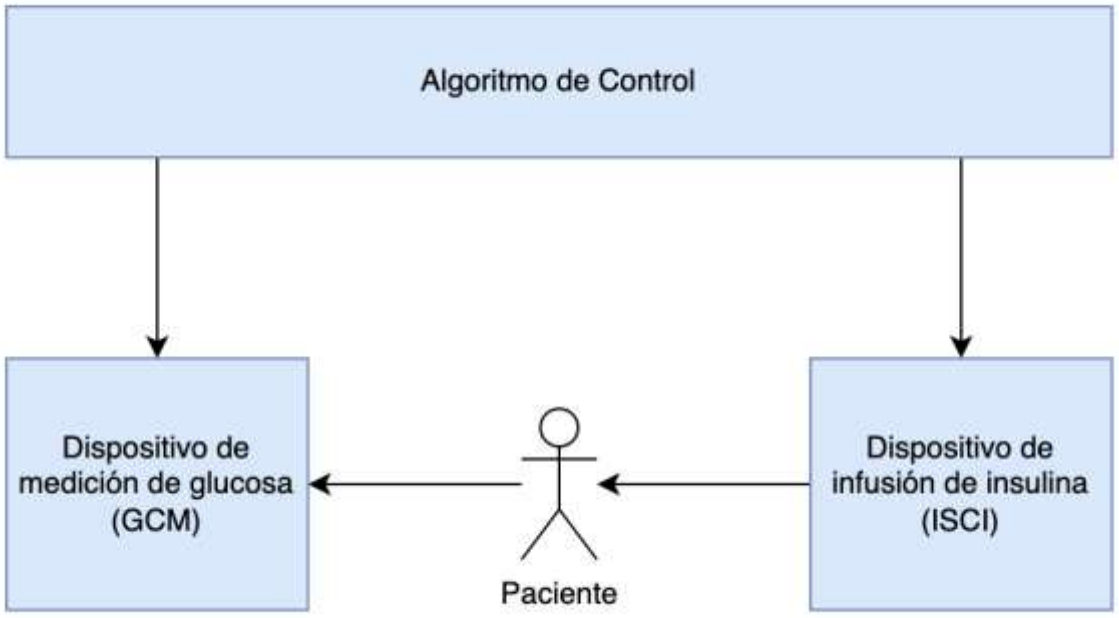

Figura 2. Esquema de un sistema de infusión de insulina completo. Adaptado de (Capel Flores, 2017, pág. 32).

Tal y como nos muestra la Figura 2, los 3 elementos que componen el sistema se ven interconectados para un correcto funcionamiento, pero estos pueden ser de capacidades

variables, y según la inteligencia que posean, el sistema puede ser considerado como un sistema de asa abierta, o como un sistema de asa cerrada.

2.7. Sistemas de asa abierta

Un sistema de asa abierta, según (González Fernández, 2019) se compone únicamente de una ISCI. Este dispositivo se encarga únicamente de administrar insulina a partir de los parámetros introducidos anteriormente por un especialista, sin tomar decisiones ni tener retroalimentación alguna.

Figura 3. Bombas de insulina de Asa abierta, de izquierda a derecha: Animas IR 1250; Animas IR2020; Insulet OmniPod; Medtronic/MiniMed™ Paradigm 522/722; Roche/Disetronic Accu-Check Spirit; Smith/Deltec Cozmo 1800 (Clinidiabet, S.L., 2018).

Todos los dispositivos que se muestran en la Figura 3, son dispositivos de asa abierta. Lo cual significa que son dispositivos que simplemente proveen al paciente con una infusión más exacta y continua que las jeringuillas tradicionales. Proporcionándole así un mejor control sobre sus niveles de glucemia, y ejemplifican la gran cantidad de opciones disponibles.

2.8. Sistemas de asa cerrada

Un sistema de asa cerrada debe contener, según (González Fernández, 2019), los 3 componentes presentados anteriormente: una ISCI, un MCG y un algoritmo de control, siendo este último el elemento de mayor importancia en el sistema.

El algoritmo de control toma el valor más importante en el sistema, ya que es el encargado de conectar los otros componentes y coordinar la comunicación entre ellos. De esta manera

encauza el flujo de datos entre la ISCI y el MCG, que toma parte en el proceso de realimentación.

Según el nivel de inteligencia que posea el algoritmo del sistema de infusión, se pueden clasificar, tal y como nos muestra la Tabla 1, en 6 niveles distintos, cada uno perteneciente a una de 3 generaciones. Siendo el nivel 1 el menos inteligente, y el nivel 6 correspondiente a el funcionamiento de un páncreas artificial.

Tabla 1. Clasificación de las bombas de insulina según la JDRF.

GENERACIÓN	NIVEL	CARACTERÍSTICAS
Primera Generación	1	**Suspensión por umbral de hipoglucemia:** El sistema suspende la infusión de insulina ante una hipoglucemia.
	2	**Minimizado de Hipoglucemia:** El sistema realiza una predicción, e intenta suspender la infusión de insulina para evitar las hipoglucemias.
	3	**Minimizado de hipo e hiperglucemia:** El sistema realiza una predicción, e intenta suspender la infusión de insulina para evitar la hipoglucemia o aumentarla en caso de hiperglucemia.
Segunda Generación	4	**Controlador Híbrido:** El sistema es completamente automático con la insulina basal, pero requiere intervención humana al ingerir alimentos.
	5	**Control automático completo:** El sistema tiene control completamente automático de la insulina sobre el individuo.

Tercera Generación	6	**Control automático completo multihormonal:** El sistema tiene todo el control sobre la infusión de insulina. Y además sobre el glucagón.

Adaptación de (Capel Flores, 2017, pág. 35)

2.9. Algoritmos de Control

Los algoritmos de control son la parte central de los sistemas de insulina, y como dice (Capel Flores, 2017) el algoritmo debe tener en cuenta todas las variables que intervienen en la homeóstasis de la glucosa. Pero dada la complejidad matemática para modelarlo, la mayoría son diseñados usando como entrada principal la medición continua de la glucosa, y como salida la cantidad de insulina a administrar. Según el punto de vista de la ingeniería de control, se puede clasificar en 4 grupos, los cuales se verán a continuación.

2.9.1. Algoritmos con control Proporcional-Integral-Derivativo PID

Son algoritmos con un esquema clásico, usado previamente para control de procesos industriales, los cuales evalúan constantemente cómo evoluciona la desviación entre el valor deseado y el valor actual de la variable a controlar en función de tres componentes: proporcional, derivativo e integral (Capel Flores, 2017).

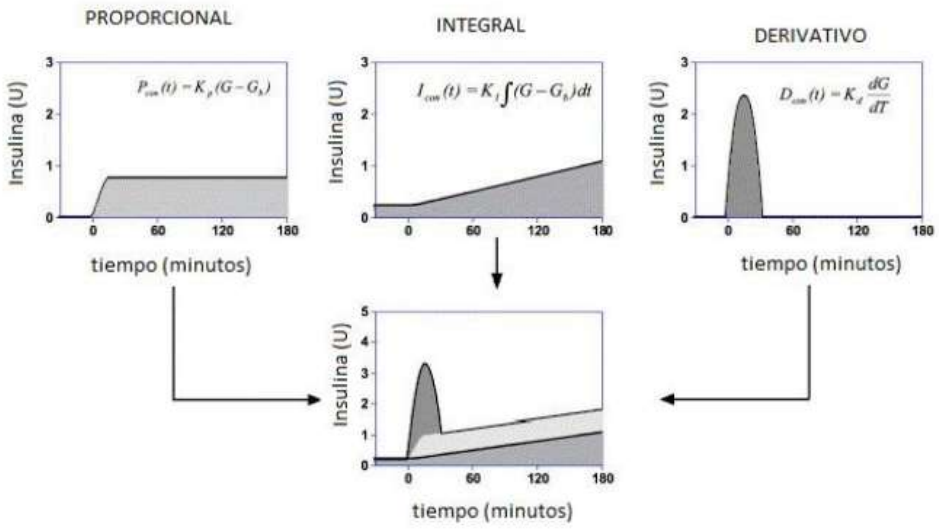

Figura 4. Componentes de un algoritmo PID (Capel Flores, 2017, pág. 59).

2.9.2. Algoritmos con predicción basada en modelo MPC

Según (Capel Flores, 2017) son algoritmos basados en la homeostasis de la glucosa, los cuales realizan una predicción sobre cómo evolucionará la glucemia, y se propone una solución que luego será reevaluada para ser corregida. La mayoría de estos algoritmos son adaptativos y poseen más valores de entrada, además de la glucemia del paciente.

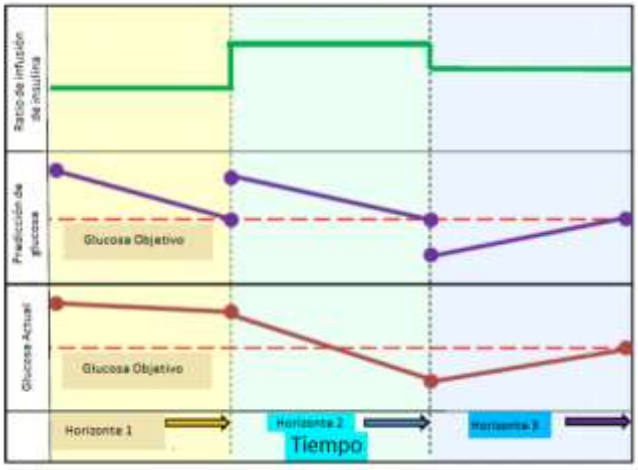

Figura 5. Esquema de un algoritmo con predicción MPC (Capel Flores, 2017, pág. 60).

2.9.3. Algoritmo predictivo basado en reglas (pRBA)

El algoritmo pRBA es un algoritmo propuesto por (Rodriguez Herrero, 2010) donde este requiere de dos fuentes de información: la realimentación (*feedback*) y los datos de control previo (*feedforward*). Conteniendo dos partes principales: el control basal y el calculador de bolus de corrección (paso de insulina). Este algoritmo incluye además predictores basados en redes neuronales y cuantificadores de insulina.

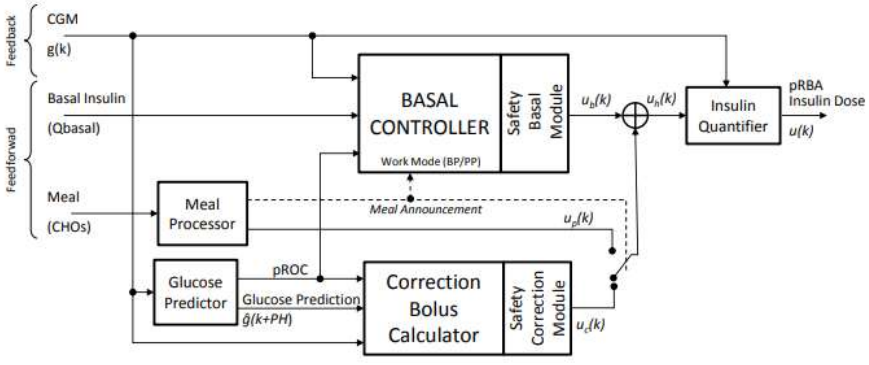

Figura 6. Topología de un algoritmo pRBA (Capel Flores, 2017, pág. 71).

2.9.4. Otros

En la categoría de otros de (Capel Flores, 2017), se engloban los algoritmos que intentan resolver de formas varias el problema de control. Algunos basándose en modelos matemáticos y otros en modelos clínicos, tal y como se muestran a continuación:

2.9.4.1. Redes neuronales

Las Redes Neuronales son una metodología de inteligencia artificial que simula el comportamiento de las neuronas humanas (Capel Flores, 2017), y permite modelar un proceso de pensamiento a través de interconexiones entre neuronas.

2.9.4.2. Lógica Difusa

La lógica difusa, según (Capel Flores, 2017) es un método de la ingeniería de control que permite abordar a un fenómeno con múltiples variables y se basa en evitar categorías absolutas, asignando varias categorías a cada entrada.

2.9.4.3. Control por inversión

Los algoritmos de control por inversión son modelos obtenidos a partir de la inversión de un modelo matemático a partir de la dinámica entre la insulina y la glucosa (Capel Flores, 2017), y se enfoca en preguntar cuánto se necesita para aproximarse al valor deseado.

2.9.4.4. Inspirado en el funcionamiento de la Célula beta

Es un algoritmo desarrollado por investigadores británicos que consiste en un modelado matemático de la secreción de insulina por la célula beta, y una adaptación en el medio subcutáneo (Capel Flores, 2017).

2.10. Sistemas de insulina Medtronic

Medtronic es actualmente uno de los grandes fabricantes de sistemas de infusión de insulina. No es el único, pero es en el que se centrará el estudio ya que el desarrollo está enfocado directamente a sus productos. Y debido a que han desarrollado dispositivos de asa abierta y cerrada para casi cada nivel alcanzable a lo largo de los años (Medtronic, 2021c). A continuación, se muestran algunos de los modelos desarrollados y comercializados por Medtronic respecto a sistemas de infusión de insulina.

2.10.1. MiniMed™ 640G

Según (Medtronic, 2018) el sistema 640G permite a sus pacientes mejorar el control ante los episodios de hipoglucemia, en particular predice con una antelación de 30 minutos si se producirá una hipoglucemia y detiene la infusión de insulina en esos casos.

Figura 7. Sistema de insulina MiniMed ™640G (Medtronic, 2018).

Esta descripción nos permite encasillar el sistema MiniMed™ en el nivel 2 de la clasificación propuesto por la JDRF, ya que es capaz de predecir hipoglucemias e intenta prevenirlas, sin embargo, ante las hiperglucemias no actúa de ninguna manera.

2.10.2. MiniMed™ 670G

El sistema MiniMed™ 670G, según su fabricante (Medtronic, 2019), es el primer sistema híbrido de asa cerrada que se ajusta automáticamente cada 5 minutos. Esto permite a los pacientes tener una vida más espontánea y con menos preocupaciones.

Figura 8. Sistema de insulina MiniMed ™670G (Medtronic, 2018).

La descripción de este sistema, lo posiciona en el nivel 4 de clasificación de la JDRF, ya que trata la insulina basal de manera automática, pero aún requiere interacción humana para las comidas.

2.10.3. MiniMed™ 780G

El sistema MiniMed™ 780G es, según (Medtronic, 2020), un sistema de asa cerrada que ajusta la insulina basal, para regular los niveles de glucosa durante el día y la noche, además de realizar correcciones automáticamente.

Figura 9. Sistema de insulina MiniMed™ 780G (Medtronic, 2021a).

El sistema 780G utiliza un algoritmo llamado AHCL (*Advanced Hybrid Closed Loop*). Algoritmo de control avanzado de asa cerrada (Cai, 2021), el cual es una versión mejorada a partir de su predecesor, el del sistema 670G, brindando al usuario una operación más simple y con menos alarmas.

Respecto a su predecesor, este algoritmo está clasificado también dentro del nivel 4 de la JDRF. Pero a pesar de esto, los estudios (Cai, 2021) demuestran gran cantidad de mejoras en el funcionamiento respecto a su antecesor, en especial en la mejora de los controles de los niveles de glucosa del paciente.

2.11.APS: Sistema de Páncreas Artificial de Software Libre

Además de las soluciones propuestas por compañías privadas, tal como dice (Omer, 2016), existen grupos de desarrolladores que han aunado sus esfuerzos para crear una solución abierta, que permita a los pacientes de diabetes tener lo más parecido a un páncreas artificial. De este grupo de personas, se deriva la idea que luego se convertiría en el proyecto #OpenAPS, *The Open Artificial Pancreas System*.

Estos APS (Sistema de Páncreas Artificial) son sistemas de páncreas artificial, manejados por los usuarios, diseñados con la intención de ajustar automáticamente el suministro de insulina de una bomba de insulina compatible, para mantener la glucosa en rango, mediante la comunicación con una ISCI y un MCG compatibles y previamente aprobados (Lewis & The #OpenAPS Community, OpenAPS Reference Design, 2015).

2.11.1. Comunidad #OpenAPS

Según su propia web (Lewis & The #OpenAPS Community, What is #OpenAPS?, 2018), la comunidad #OpenAPS se define como una comunidad que intenta generar tecnologías eficientes y seguras respecto a los sistemas páncreas artificiales, proveyendo a las personas de: un diseño de referencia enfocado en la seguridad, una implementación de código libre y su documentación.

Actualmente se estima que hay más de 2400 personas alrededor del mundo usando un sistema APS de lazo cerrado de software libre (Lewis & The #OpenAPS Community, OpenAPS Outcomes, 2021), las cuales dependen siempre de una serie de componentes muy específicos para que este sistema sea funcional.

2.11.2. Componentes necesarios

Para que un páncreas artificial pueda ser ensamblado, se requieren de ciertos componentes físicos, tal y como se muestra en su documentación (The OpenAPS Foundation, 2021), de entre

los cuales es mínimamente necesario tener: una bomba de insulina compatible, un MCG compatible, un pequeño ordenador tal como un Raspberry Pi o Intel Edison, y una batería.

En cuanto a estos elementos mencionados, cabe destacar que no cualquier dispositivo es permitido, y especialmente las ISCI y los MCG son una gran limitante para poder usar este software. De igual manera, la comunidad alienta a las personas a implementar el soporte a nuevos dispositivos para poder incrementar la cantidad de pacientes.

2.11.3. Algoritmos presentes

Actualmente, #OpenAPS (The OpenAPS Foundation, 2021) provee 2 posibles algoritmos para su uso, siendo uno de ellos, el oref0, la versión básica y recomendada, con un mayor historial de versiones y refinamiento. El cual determina las dosis de insulina basándose en distintos escenarios, los cuales obtiene mediante distintas predicciones.

Este algoritmo oref0 utiliza como parámetros de entrada los valores de: glucosa, dosis actual de insulina, desviación de la glucosa, sensibilidad a la insulina, glucosa esperada, impacto de los carbohidratos, y más parámetros que deben ser configurados para cada paciente de manera específica, para adaptarse a sus necesidades (The OpenAPS Foundation, 2021).

El segundo algoritmo presente es el oref1, que nace como una mejora de la primera versión, incorporando nuevas funcionalidades tales como los supermicrobolos, que no son soportadas por todos los dispositivos, pero que permiten una simulación más real a la absorción natural de la insulina.

Además de esto, la versión oref1 (The OpenAPS Foundation, 2021) propone, de manera innovadora, la funcionalidad de UAM, de sus siglas en inglés *Unannouced Meals*, el cual traduce a: "comidas sin anunciar". Esta funcionalidad permite detectar de manera autónoma la subida de los niveles de glucemia debido a comidas del paciente, sin necesidad de necesitar una interacción humana. Esto lograría clasificar a un sistema de insulina con el algoritmo oref1 dentro del nivel 5 de la JDRF, ya que este sistema sería capaz de controlar completamente la infusión de la insulina del paciente de manera autónoma sin necesidad de interacciones al momento de las comidas.

3. Contextualización

El desarrollo de este trabajo se encuentra enmarcado en el ámbito de las aplicaciones móviles, específicamente aplicaciones para dispositivos Apple iPhone y Apple Watch. Ligado directamente a pacientes usuarios de sistemas de insulina específicos de Medtronic, y es en el ámbito de esta específica combinación que se ha visto un vacío que se intenta llenar con el desarrollo planteado.

En este apartado se presentará el estado actual del dominio de la aplicación, en el cual se muestran diversas aplicaciones que: poseen ya un modelado de la solución que se desea plantear en este trabajo, o resuelven el problema planteado. Pero que, de alguna manera no funcionan para esta combinación de dispositivos, o que no lo hacen de la manera deseada.

Se presentarán entonces aplicaciones de otros sistemas de insulina, de otros MCG, y la solución que propone la comunidad #OpenAPS. Con la intención de evaluar cómo solventan estas aplicaciones el problema en cuestión, y utilizar este conocimiento previo como base del desarrollo a realizar a continuación, haciendo hincapié en las tecnologías que utilizan estas aplicaciones para la comunicación y tratado de datos entre los dispositivos involucrados.

3.1. Freestyle Libre

Dependiendo de la versión de Freestyle Libre de la que se trate, como comenta (Diabetes Mall, 2021) , puede ser considerado un MCG o no. Ya que la versión Freestyle2 requiere que el móvil o lector sea acercado al sensor en el cuerpo del paciente para obtener una nueva lectura. Mientras que la versión 3, entrega los datos de manera automática al dispositivo móvil, haciendo así que sea un MCG completamente dependiente del móvil del usuario.

Figura 10. MCG Freestyle Libre y aplicación móvil asociada (FreeStyle Libre, s.f.).

En ambos casos, Freestyle presenta una aplicación para dispositivos iOS y Android, la cual se puede ver en la Figura 10, mediante la cual el usuario puede ver sus valores de glucemia y la tendencia de esta, e incluso compartirla. Pero hasta la fecha no se presentan, por parte del fabricante, extensiones ni aplicaciones para dispositivos Apple Watch, que permitan al usuario ver estos valores en sus dispositivos de muñeca.

3.2. Eversense

Eversense es, según (Diabetes Mall, 2021), el único sensor implantable en el cuerpo, haciéndolo además el único MCG en proveer vibraciones en el cuerpo ante hipo o hiperglucemias, sin necesidad de tener un dispositivo físico con el usuario.

Figura 11. MCG Eversense, aplicación móvil y de muñeca asociadas (Diabetes Mall, 2021).

Este dispositivo sí presenta una aplicación para móviles y dispositivos de muñeca Apple Watch, tal y como se muestra en la Figura 11. Cumpliendo esta parcialmente los objetivos detrás del desarrollo de este proyecto.

Se debe observar que no existen referencias sobre la lectura de los datos en el Apple Watch mediante complicaciones. Siendo este un punto importante y crítico en el dominio de estudio actual, que esta aplicación no cumple.

3.3. Dexcom

Dexcom es una compañía con gran experiencia en dispositivos MCG, comenta (Diabetes Mall, 2021) que estos han sido de los primeros desarrolladores de las tecnologías MCG. Su modelo G6 es por sí solo un MCG, pero tiene la capacidad de conectarse con la ISCI Tandem:t-slim, convirtiendo esta ISCI en un sistema de insulina completo con predicción y prevención de hipoglucemias (Diabetes Mall, 2020). Esto conlleva que el sistema resultante pueda ser clasificable en el nivel 2 propuesto por la JDRF.

Figura 12. MCG Dexcom, aplicación móvil y de muñeca asociadas (Diabetes Mall, 2021).

El sistema presentado en la Figura 12 presenta, en cuanto a aplicaciones, soporte para dispositivos Android y Apple, al igual que todos los demás. Y presenta además soporte para dispositivos de muñeca tales como el Apple Watch.

A diferencia de las alternativas mostradas anteriormente, se puede observar que (Smith, 2020) muestra cómo esta aplicación sí posee un soporte directo para las complicaciones del Apple Watch, mostrando clara y concisamente los valores de glucemia y su tendencia, al alcance de la muñeca de los usuarios.

3.4. Medtronic

Por parte de Medtronic, se habla específicamente de la familia de MCG Guardian™ Connect. Los cuales, tal y como expone (Diabetes Mall, 2021), pueden ser utilizados de manera

autónoma como un MCG independiente. O pueden ser utilizados como sistemas de insulina, junto con un ISCI MiniMed™ 640G, MiniMed™ 670G o MiniMed™ 780G. Constituyendo así un sistema de insulina de niveles variables, posiblemente nivel 2 o 3 según la JDRF, dependiendo del ISCI utilizado.

Se pueden encontrar por parte del fabricante 2 aplicaciones para móvil que permiten a los usuarios de los sistemas de insulina, y de los MCG, ver en sus dispositivos móviles las tendencias de insulina y obtener alarmas en caso de híper o hipo glucemias del paciente.

Figura 13. Aplicaciones MiniMed™ Mobile App y CareLink™ Connect App (Medtronic, 2020).

Estas 2 aplicaciones son: MiniMed™ Mobile App y CareLink™ Connect App, mostradas en la Figura 13. Ambas aplicaciones están disponibles para móviles iPhone y Android, y ambas permiten a los usuarios ver la misma información. La diferencia entre estas aplicaciones, según (Medtronic, 2020), radica en que la primera es la encargada de obtener los datos directamente desde la bomba de insulina, y de subir los datos a la plataforma CareLink™. Mientras que la segunda es una aplicación utilitaria para terceros, ideada inicialmente para cuidadores, permitiéndoles tener acceso a la información del paciente en tiempo real.

La primera de estas es necesaria para que la segunda tenga acceso a los datos. Ya que es el puente de conexión directo entre el MCG y el guardado de datos en la plataforma CareLink™,

Una vez se tienen ya los datos guardados en la API, es cuando se puede dar acceso a terceros a estos datos, y para esto existe la segunda aplicación. Para dar acceso a los datos y notificaciones del paciente a cuidadores o familiares.

Es importante notar que, a pesar de la integración entre este sistema y los dispositivos físicos involucrados, no se presenta por parte del fabricante ningún tipo de aplicación o extensión para dispositivos de muñeca.

3.5. Nightscout

Nightscout es una aplicación de código libre que permite visualizar, guardar y compartir los datos de un sistema de MCG. Que necesita de al menos dos partes: el sitio de Nightscout para guardar, procesar y visualizar los datos. Y un dispositivo para subir los datos. (Nighscout Foundation, 2017). En sí Nightscout no es un MCG ni un sistema de insulina, sino una utilidad de manejo y conexión de otros MCG e ISCI, con la intención de proporcionar soluciones software a los usuarios para permitirles visualizar, tratar, reportar y analizar sus datos de manera cómoda.

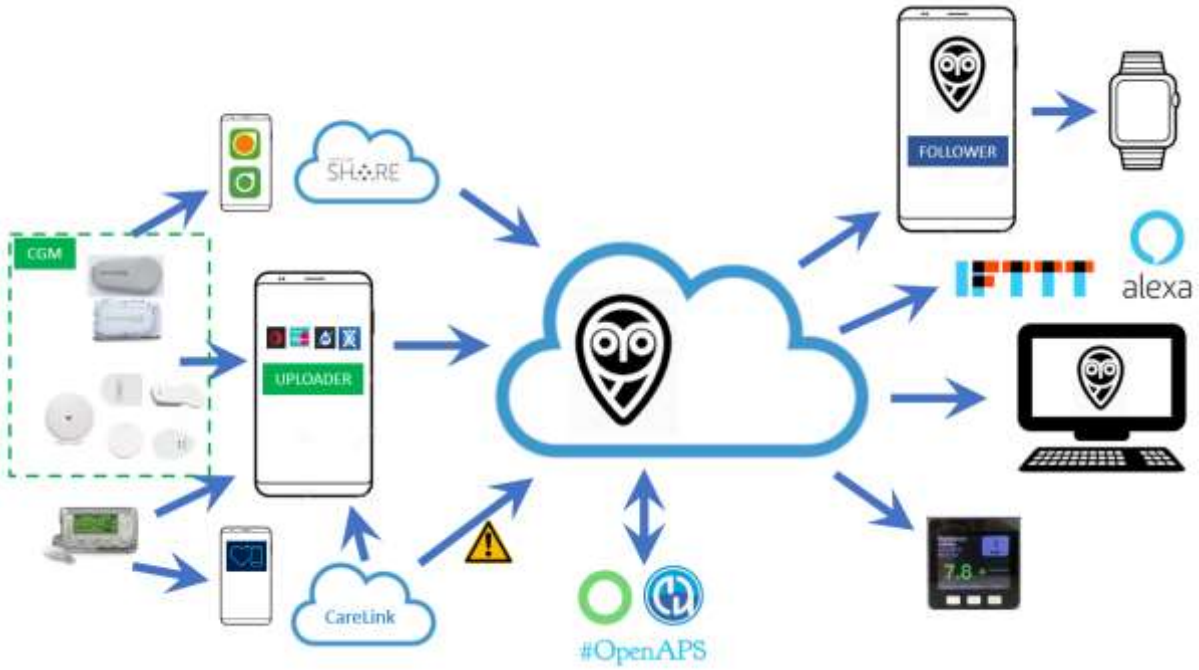

Figura 14. Esquema de funcionamiento de Nightscout (Nighscout Foundation, 2017).

Tal y como muestra la Figura 14, Nightscout podría estar compuesto por una gran cantidad de partes, siendo una de ellas un software completo, el cual necesita ser desplegado a un servidor web, además de un MCG compatible. Luego de configurado el sistema, como expone (Tejedor, 2018), con cualquier dispositivo con conexión a internet se pueden visualizar los datos, mediante distintas aplicaciones las cuales soportan gran cantidad de dispositivos.

Teniendo ya un sistema Nightscout funcional, dado la naturaleza de ser un software libre, se presentan varias alternativas para la visualización de datos. Una de ellas es Nightguard, aplicación de código libre desarrollada por (nightscout, 2021). La cual es una aplicación para dispositivos iPhone y Apple Watch que permite mostrar los valores de glucemia obtenidos a partir de un servidor Nightscout.

Figura 15. Aplicación Nightguard para iOS y WatchOS (nightscout, 2021).

Esta aplicación presenta, tal y como muestra la Figura 15, para dispositivos de la familia Apple Watch, la posibilidad de visualizar los datos desde una aplicación, y desde las complicaciones en las carátulas.

Dado que esta es una aplicación de software libre, y que cumple la funcionalidad deseada a desarrollar en este proyecto, se analizará el código fuente de dicha aplicación con la intención de validar las propuestas de comunicación de datos desde y hacia el Apple Watch.

4. Diseño de la propuesta

En este apartado se describe el proceso de estructuración y planeación llevado a cabo para el correcto desarrollo de la aplicación. Se presenta inicialmente una visión general de la aplicación, y su arquitectura a un nivel técnico simple. A continuación, se expone el alcance del desarrollo de software propuesto, acompañado de los requisitos funcionales y no funcionales, los cuales serán la base para el desarrollo del producto, obtenidos a partir de los objetivos propuestos en el apartado 1.3.

Para poder modelar el desarrollo del software en cuestión, de una manera más cercana a un producto final, se procede a introducir un pequeño apartado que explica la identidad visual del software: logotipos, estilos, y prototipos de diseño; Obtenidos con la finalidad de cumplir los requisitos especificados y proporcionar una experiencia amigable al usuario.

Seguidamente se presenta la metodología utilizada para el desarrollo ágil del proyecto. Esta se divide el desarrollo en pequeñas unidades de trabajo manejables y realizables, como lo son las historias de usuario y, dentro de ellas, las tareas. Posteriormente, se expone el plan de desarrollo propuesto para la aplicación, haciendo hincapié en la planeación de lapsos de tiempo, para una mejor organización del software.

Para culminar, se presenta en detalle el apartado de la arquitectura de la solución desarrollada, en el cual se desglosa la aplicación en las distintas capas. Y se ejemplifica cada una de estas capas con los diagramas pertinentes, tales como los diagramas de clases de uso y los diagramas de clases.

Antes de finalizar, de manera informativa, se presenta un listado de las tecnologías y Frameworks utilizados a lo largo del desarrollo, con la finalidad de proveer contexto técnico respecto a las tecnologías usadas en el ámbito específico del desarrollo de software, y la escritura de código.

Finalmente, se presenta un apartado de evaluación, en el cual se explican las pruebas de validación utilizadas, y se presentan las capturas de pantalla de la aplicación en funcionamiento. Además, se verifica la adecuación del diseño resultante respecto a los prototipos planteados; y la funcionalidad de la aplicación respecto a los objetivos y requisitos fijados, validados por un usuario.

4.1. Visión general

Para poder lograr los objetivos propuestos, se debe mostrar al usuario sus valores de glucemia y la tendencia de esta, en sus dispositivos Apple iPhone y Watch. Estos datos son obtenidos a partir de la plataforma CareLink™. El usuario, para esto, debe ser portador de un sistema de insulina MiniMed™ 780G, y tener instalado en su dispositivo móvil la aplicación MiniMed™ Mobile. La cual es la encargada, luego de la correcta configuración y habilitación de la sincronización de datos, de subir a la plataforma CareLink™ las mediciones de glucemia del MCG del usuario en un intervalo de tiempo de 5 minutos.

Los datos que debe mostrar la aplicación a desarrollar son: la glucemia del usuario y su tendencia actual. Estos datos se obtendrán a través de la API de CareLink™, y para esto el usuario debe conocer, y proporcionar, sus credenciales de acceso a esta plataforma. Estas credenciales serán guardadas en el dispositivo mediante el *Framework* Keychain de Apple, de manera que se encontrarán encriptadas y bajo la propiedad del usuario.

Para la extracción de los datos necesarios de la API, se realizará una adaptación de la librería de código abierto creada por (Szász, 2021). Originalmente escrita en el lenguaje de programación Java, realizándose un trabajo de reescritura al lenguaje Swift, para permitir así que este código pueda ser ejecutado directamente desde los dispositivos Apple.

De esta manera se simplifica la arquitectura de la aplicación, al no haber necesidad de un servidor para el procesado de datos. Esto a su vez hace que toda la lógica y flujo de datos de los usuarios, no se vea afectado por problemas de protección de datos. Ya que todos los accesos a los datos de los usuarios son directos desde sus dispositivos personales.

Sabiendo que los valores de la ISCI, y por consiguiente de la API CareLink™, se actualizan cada 5 minutos. El flujo de datos de la aplicación consistirá en preguntar cada 5 minutos por datos nuevos a la API CareLink™. Este proceso es realizado desde la aplicación del iPhone únicamente, luego de que el usuario haya introducido sus credenciales válidas de la plataforma CareLink™.

Al recibir los datos en el iPhone, se comprueba si el Apple Watch posee la aplicación instalada y se encuentra disponible. En caso de que la aplicación Watch esté disponible, se procede a enviar un mensaje desde el iPhone con los valores actuales de la glucemia del usuario y demás

datos necesarios. Utilizando como medio de transporte el *Framework* WCSession de comunicación entre iPhone y Watch, propuesto por Apple.

Cuando el Apple Watch recibe los datos desde el iPhone, procede a actualizar sus entidades internas con los nuevos datos, y en consecuencia a actualizar las complicaciones, mostradas en las carátulas, con estos nuevos datos recibidos.

Dada la naturaleza de las complicaciones, las cuales deben ser actualizadas sin necesidad de interacción del usuario, la aplicación del Apple Watch registrará tareas en segundo plano, cuya intención es pedir, al iPhone conectado, cada 5 minutos los datos actualizados de los valores de glucemia del usuario.

De esta misma manera el Apple Watch puede solicitar al iPhone, también mediante el sistema de mensajería WCSession, una actualización de los datos. Haciendo entonces que el iPhone genere una nueva solicitud de datos a la API externa o enviando directamente los datos guardados en memoria, si estos son lo suficientemente recientes.

Teniendo conocimiento del flujo básico de información de la aplicación, se presenta a continuación en detalle, todas las funcionalidades que debe tener la aplicación. Expresadas mediante requisitos, tanto funcionales como no funcionales. La intención es sentar las bases para lo que será la arquitectura y desarrollo en detalle de la aplicación.

4.2. Requisitos del Sistema

En este apartado se presentarán los requisitos funcionales y no funcionales que han sido tomados en consideración, con la finalidad de completar los objetivos propuestos en el apartado 1.3. Y a su vez, satisfacer las necesidades que puede tener el usuario de dicha aplicación. Pero para esto, se debe conocer qué son los requisitos y qué expresan en el contexto del desarrollo de una aplicación.

Un requisito es un atributo necesario de un sistema, una declaración que identifica una característica, una capacidad o un factor de calidad, con la finalidad de aportar valor y utilidad a un cliente o usuario (Young, 2004). Los requisitos pueden ser presentados desde la perspectiva del usuario, del sistema o de la documentación.

Según (Sommervile, 2005) los requisitos se clasifican en funcionales y no funcionales. Donde los funcionales son especificaciones de qué debe hacer el sistema y cómo debe comportarse. Mientras que los no funcionales son en su mayoría restricciones sobre el sistema, especialmente sobre el desarrollo o procesos internos.

Con toda esta información necesaria ya definida, se procede a detallar a continuación los requisitos funcionales y no funcionales de la aplicación. Especificando primero el alcance y limitaciones que tendrá este proyecto, los cuales deben ser tomados en cuenta para la creación de los requisitos.

4.2.1. Alcance y limitaciones

El presente trabajo tiene como finalidad el análisis, diseño, desarrollo y validación de una aplicación para Apple Watch, que permita a un usuario del sistema de insulina MiniMed™ 780G ver sus valores de glucemia y su tendencia, en su dispositivo Apple Watch mediante las complicaciones de Apple, así como en la aplicación misma.

Para que los datos puedan ser obtenidos en tiempo real, el usuario debe tener instalada y en funcionamiento la aplicación MiniMed™ Mobile en su móvil, y configurada la sincronización con la plataforma CareLink™. Permitiendo así la subida de datos a la fuente de información de la aplicación, dicha plataforma.

De esta manera, como limitaciones para el funcionamiento, se observa la directa dependencia que existe entre el software y: la plataforma de datos, la aplicación móvil del usuario, y la correcta configuración necesaria para la subida de datos. Pero que, a su vez deberían ser una configuración básica para usuarios de este tipo de sistemas.

4.2.2. Requisitos funcionales

Para la obtención de los requisitos funcionales, se toma como base los objetivos planteados en el apartado 1.3. Y para la representación de los requisitos funcionales, de manera estandarizada, se utilizará la plantilla EARS (*Easy Approach to Requirements Syntax*). La cual fue desarrollada por la empresa Rolls-Royce, para mejorar la especificación de sus proyectos (Mavin, Wilkinson, Harwood, & Novak, 2009) y ha sido usada a lo largo de los años en diversas áreas de la ingeniería.

Tabla 2. Requisitos Funcionales

IDENTIFICADOR	DESCRIPCIÓN
RF-1	Cuando el usuario introduzca sus credenciales de CareLink™, la aplicación deberá validar las credenciales contra la API CareLink™, y en caso de ser correctas guardarlas.
RF-2	Mientras haya credenciales guardadas, y estas sean válidas, la aplicación deberá ser capaz de obtener los datos de la API CareLink™.
RF-3	La aplicación deberá mostrar en el iPhone los datos de glucemia y tendencia de la glucemia obtenidos de la API CareLink™.
RF-4	La aplicación deberá mostrar en el Apple Watch los datos de glucemia y tendencia de la glucemia obtenidos de la API CareLink™.
RF-5	Cuando habiendo credenciales guardadas, el usuario desea eliminar dichas credenciales, la aplicación deberá eliminar las credenciales guardadas del usuario.

4.2.3. Requisitos no funcionales

Para la obtención de los requisitos no funcionales, además de los objetivos planteados, se procede a especificar restricciones propuestas directamente como decisiones de diseño. Tales como la arquitectura de código a seguir, o limitaciones en cuanto al guardado de datos.

De esta manera, los requisitos no funcionales, describen las formas y restricciones que debe seguir la aplicación a momento de desarrollo, especificando incluso en algunos casos qué no debe hacer la aplicación. Para su representación se utilizará también la plantilla EARS explicada en el apartado anterior.

Tabla 3. Requisitos No Funcionales

IDENTIFICADOR	DESCRIPCIÓN
RNF-1	La aplicación deberá actualizar los datos siempre que la API CareLink™ obtenga datos nuevos.
RNF-2	El código de la aplicación deberá estar estructurado según la arquitectura "Easy Architecture" propuesta por (Martin, 2018).
RNF-3	La aplicación no guardará ningún tipo de dato fuera del dispositivo en el que se ejecuta.
RNF-4	La aplicación no será dueña de los datos guardados en el dispositivo en el que se ejecuta.

4.3. Prototipado y diseño de la aplicación

Dados los requisitos y funcionalidades de la aplicación ya especificados, se procede a desarrollar una identidad visual para el producto, que será el desarrollo de software en cuestión. Esto con la intención de poder, en un futuro, publicar en la tienda de aplicaciones de Apple: AppStore.

Con esta intención, se ha desarrollado para la aplicación una serie de elementos visuales, tales como un logotipo y un isotipo. De manera que permitan que la aplicación tenga una identidad visual por sí misma, y sea fácilmente reconocible por los usuarios.

Se incluye también en este apartado una serie de prototipos visuales desarrollados, a partir de los requisitos y objetivos a cumplir en la aplicación. Estos prototipos serán la base visual que será seguida a lo largo del proceso de desarrollo, para generar así una estética amigable para el usuario de la aplicación.

4.3.1. Nombre comercial: glucosapp

Para esta aplicación, se ha escogido como nombre comercial, o lo que será el nombre público de la aplicación, la combinación de palabras: "glucosapp". Esta proviene de una mezcla entre

las palabras "glucosa" y el diminutivo de aplicación: "app". Intentando generar una familiaridad al usuario entre una aplicación, y sus valores de glucemia o glucosa, los cuales son el centro del desarrollo.

4.3.2. Logotipo e Isotipo

El logotipo de la aplicación consiste en el nombre comercial, escrito todo en letras minúsculas. Usando la fuente Outfit, en estilo mediano. Mientras que el isotipo desarrollado consiste en dos gotas, de color rojo. Una de la mitad de tamaño de la otra, y posicionada en la parte superior derecha de la más grande, solapando parte de su figura.

El isotipo de la aplicación debe generar una asociación directa a la aplicación, en particular, este elemento será utilizado como icono de la aplicación. Al momento de ser instalada en dispositivos móviles y, por ende, en las tiendas de aplicaciones desde donde se descarga.

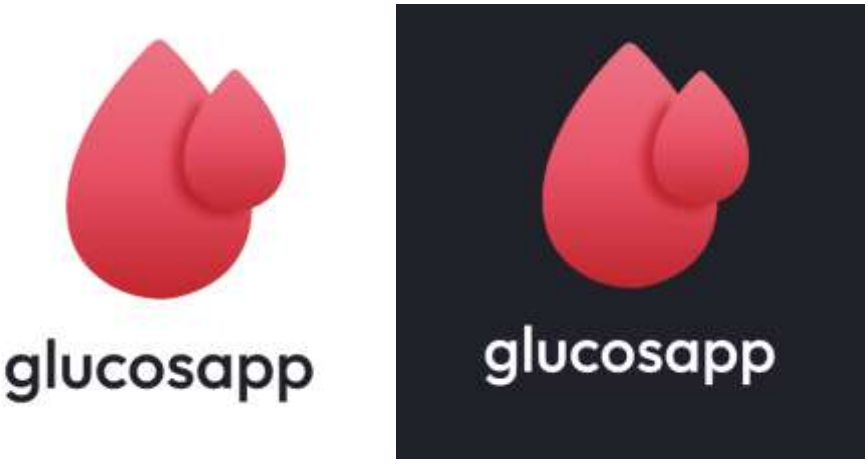

Figura 16. Logotipo e isotipo de la aplicación glucosapp en fondo claro y oscuro.

La Figura 16 nos muestra la combinación del logotipo e isotipo desarrollados en cuestión. Se proponen además 2 versiones: para el modo normal, y para el modo oscuro. Todos los prototipos presentados a continuación serán presentados siempre en la versión para modo claro y modo oscuro.

4.3.3. Prototipo de la aplicación móvil

La aplicación móvil consiste inicialmente de 2 pantallas: la primera de ellas, donde el usuario debe introducir sus credenciales, o iniciar sesión. Y la segunda pantalla, es donde un usuario ya con credenciales guardadas puede ver sus valores de glucemia.

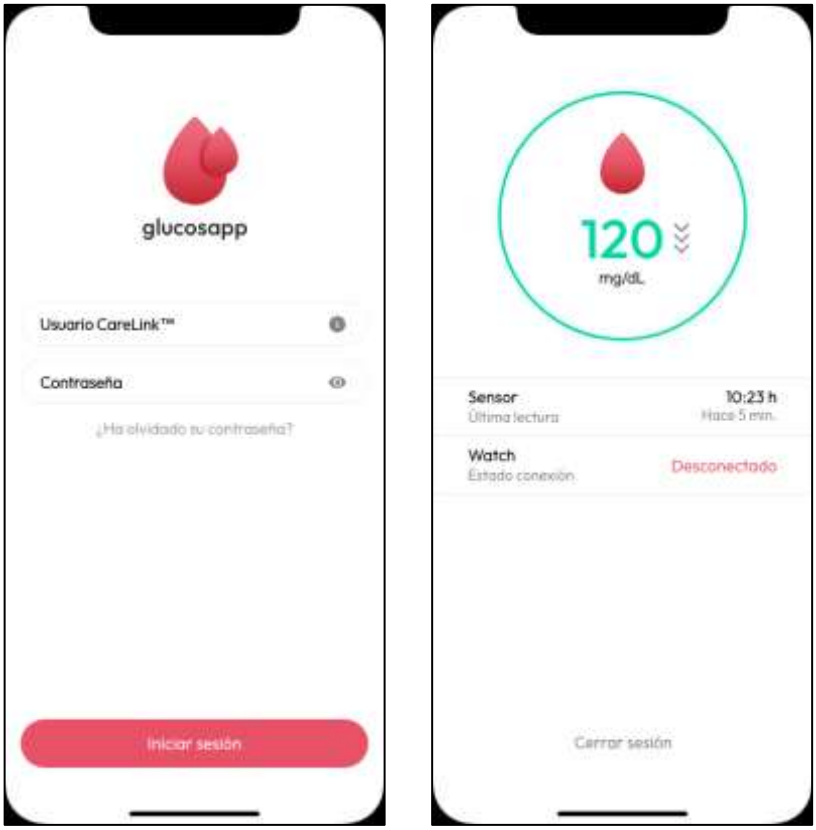

Figura 17. Prototipo de las vistas de la aplicación móvil glucosapp en fondo claro.

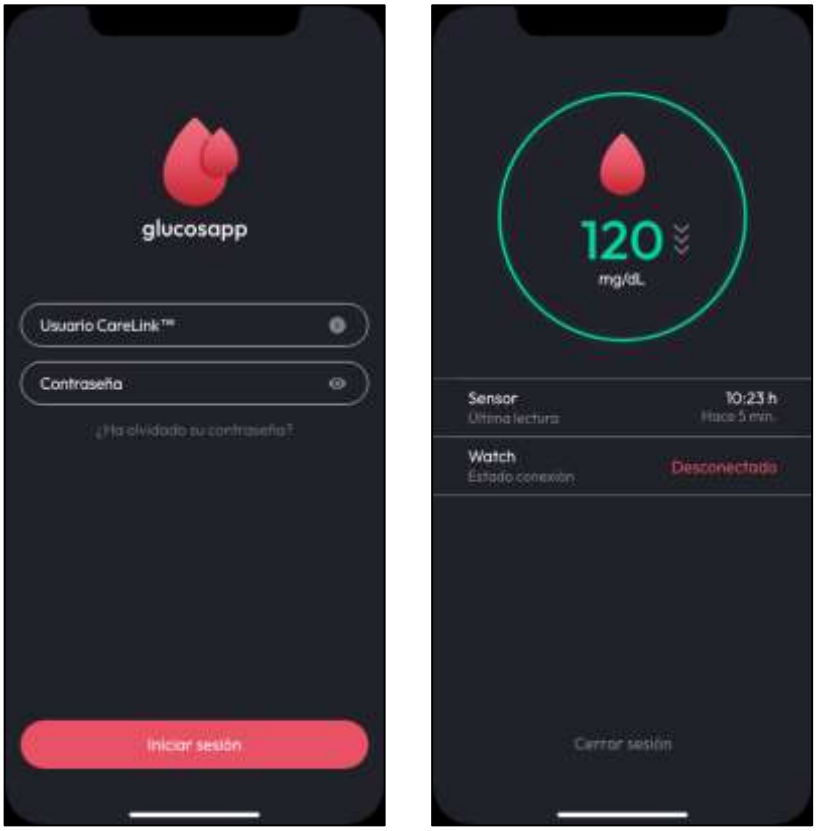

Figura 18. Prototipo de las vistas de la aplicación móvil glucosapp en fondo oscuro.

Las Figuras 17 y 18 presentan entonces la primera y segunda pantalla de la aplicación, presentadas en modo claro y oscuro correspondientemente. Se debe denotar que el prototipo presenta funcionalidades fuera del alcance de este desarrollo, como lo es la posibilidad de cambio de contraseña del usuario, o el estado de la conexión del Watch respecto al iPhone.

4.3.4. Prototipo de la aplicación Watch

Se ha realizado el mismo proceso para la aplicación Watch, en la cual se intenta reutilizar las vistas ya creadas para la aplicación móvil. Tal y como muestra la Figura 19. En este caso, se diseña únicamente para un fondo negro, debido a las restricciones que provee la plataforma Apple en cuanto a estilos y complejidad de estas aplicaciones.

Figura 19. Prototipo de la vista de la aplicación Watch glucosapp.

4.3.5. Prototipo de las complicaciones

Por último, se presenta un prototipo de un tipo específico de complicaciones que se planea soportar, debido a que Apple tiene a su disposición en estos dispositivos una gran cantidad de familias de complicaciones para sus distintas carátulas.

Sabiendo que las complicaciones son mayormente datos, y que deben estar al alcance de manera rápida y fácil para los usuarios. La estilización en este diseño se ve muy limitada, y se enfoca puramente en la funcionalidad, y no en una estética asociada a la identidad visual.

Figura 20. Prototipo de las complicaciones de la aplicación glucosapp.

La Figura 20 ejemplifica 3 posibles complicaciones. El primer prototipo, a la izquierda, nos muestra una versión simple en la esquina superior izquierda. Y una versión con un poco más de información, el tiempo de la lectura, en la parte baja del reloj. El segundo prototipo en cambio nos muestra en la parte baja una alternativa con un poco más de diseño y estilización.

Ya teniendo un esquema inicial, el cual incluye: requisitos, funcionalidades, restricciones y prototipos. A continuación, se presenta el proceso y la metodología seguida a lo largo del proceso de desarrollo, con la intención de realizar el desarrollo de manera ágil y pudiendo así desgranar los requisitos en tareas más sencillas y realizables.

4.4. Metodología

Para el desarrollo de este proyecto se seguirá una metodología de tipo ágil, las cuales define (Sommervile, 2005) como metodologías diseñadas para apoyar el desarrollo de aplicaciones, y entregar software funcional de forma rápida, durante varias iteraciones.

En este apartado se expresarán las historias de usuario de la aplicación, obtenidas a partir de los requisitos funcionales y no funcionales del apartado 4.2. A continuación estas historias de usuario serán desglosadas en tareas. Para posteriormente presentar una primera planificación de un plan de trabajo a seguir.

El plan de trabajo que se plantea pretende seguir una metodología ágil de desarrollo llamada Kanban. De acuerdo con (Kniberg & Skarin, 2010) el Kanban puede ser resumido en los siguientes pasos:

1. Visualizar el flujo de trabajo: descomponiéndolo en pequeñas tareas, escritas en un papel y puestas en la pared. Usando columnas para ilustrar el estado actual de la tarea.
2. Limitar el trabajo actual: especificar un máximo de tareas a ser resueltas a la vez
3. Medir el tiempo: que tarda en promedio en realizarse una tarea, para intentar optimizarlo lo más posible.

Ya que el desarrollo es digital, y no se dispone de un equipo físico de trabajo. Se presentarán los elementos físicos necesarios de Kanban de manera digital, mediante imágenes. El tablero que se utilizará en el desarrollo propuesto contiene las siguientes columnas:

- Pendiente: Una tarea que se encuentra esperando por otra tarea para poder ser desarrollada.
- En Progreso: Si una tarea se encuentra en esta columna significa que su trabajo ha sido comenzado, pero no está del todo terminado.
- Revisado: La tarea está hecha y cumple las funcionalidades requeridas
- Hecho: La tarea ha sido terminada y validada.

De igual manera, cada tarea dentro del tablero debe seguir una estructura específica, para poder mantener un proceso de trabajo correcto, y una homogeneidad a la hora de definir y especificar las tareas. De esta manera, cada tarea dentro de este proceso de desarrollo debe contener como mínimo los siguientes elementos:

- Identificador: identificador único como referencia de la tarea.
- Historia de usuario: historias de usuario a la cual pertenece la tarea, obtenidas a partir de los requisitos funcionales y no funcionales. Ya que una historia de usuario podría conllevar más de una tarea.
- Título: Título descriptivo de la tarea a realizar
- Descripción: Descripción detallada de la tarea a realizar.
- Dependencias: En caso de existir, significa que esta tarea depende de una o varias tareas, y que para poder ser realizada se necesita que estas tareas estén ya hechas.

4.4.1. Historias de Usuario

Para poder generar tareas para el desarrollo, se deben tener primero las historias de usuario. En este apartado se presentan las historias de usuario, obtenidas a partir de los requisitos mostrados en el apartado 4.2, representados en la tabla a continuación.

Tabla 4. Historias de Usuario

IDENTIFICADOR	REQUISITOS ASOCIADOS	DESCRIPCIÓN
HU-01	RF-1	Como usuario no identificado quiero introducir mis credenciales para ser un usuario identificado.
HU-02	RF-2 RF-3	Como usuario identificado quiero ver el valor de la glucosa y su tendencia en mi aplicación instalada en el dispositivo iPhone.
HU-03	RF-2 RF-4	Como usuario identificado quiero ver el valor de la glucosa y su tendencia en mi aplicación instalada en el dispositivo Apple Watch.
HU-04	RF-2 RF-4	Como usuario identificado quiero ver el valor de la glucosa y su tendencia en mis complicaciones del Apple Watch.
HU-05	RF-5	Como usuario identificado quiero poder eliminar mis credenciales para convertirme en un usuario no identificado.

La Tabla 4, muestra para cada historia de usuario su identificador único, la descripción de la historia de usuario. Y la referencia que existe entre cada historia de usuario respecto a los requisitos funcionales y no funcionales.

4.4.2. Desglose de tareas

A partir de las historias de usuario expuestas anteriormente, junto con los requisitos funcionales y no funcionales del apartado 4.2, se ha desglosado cada historia de usuario en tareas, para su implementación y desarrollo, de manera manejable.

Tabla 5. Tareas

ID	TÍTULO	DESCRIPCIÓN	H.U.	DEP.
T-01	Reescritura librería CareLink™	Reescritura de Java a Swift, de la librería de conexión a la API CareLink™, escrita por (Szász, 2021).	HU-02 HU-03	
T-02	Inicio de Sesión	La aplicación debe ser capaz de guardar las credenciales de CareLink™ del usuario en el Keychain del dispositivo en caso de haber introducido las credenciales correctamente.	HU-01	T-01
T-03	Obtención de datos	A partir de la librería CareLink™ reescrita y las credenciales guardadas del usuario, se debe ser capaz de obtener la glucosa y la tendencia de esta, guardando estos datos dentro de los modelos del dominio de la aplicación de iPhone.	HU-02 HU-03	T-02
T-04	Refresco de datos iPhone	La aplicación debe refrescar los datos desde la API, siempre que la última medición haya sucedido hace 5 minutos o más. Y actualizar dichos datos en los modelos del dominio de la aplicación de iPhone.	HU-02	T-03

T-05	Visualización de datos en el iPhone	Habiendo datos guardados en el dominio de la aplicación del iPhone, se deben visualizar los valores de glucosa y su tendencia en la pantalla principal de la aplicación.	HU-02	T-03
T-06	Enviar datos al Apple Watch	A partir de los datos guardados en los modelos del dominio de la aplicación, y teniendo un Apple Watch conectado. Se deben enviar los datos de glucemia y su tendencia desde el iPhone hasta el Apple Watch para guardarlos en los modelos del dominio de la aplicación del Apple Watch.	HU-03	T-03
T-07	Visualización de Datos Apple Watch	Habiendo datos guardados en el dominio de la aplicación del Apple Watch, se deben visualizar los valores de glucosa y su tendencia en la pantalla principal de la aplicación.	HU-03	T-06
T-08	Visualización de datos en las complicaciones	Habiendo datos guardados en el dominio de la aplicación del Apple Watch, se deben visualizar los valores de glucosa y su tendencia en las complicaciones del Apple Watch.	HU-04	T-06
T-09	Actualización de datos de las complicaciones	La aplicación debe solicitar los datos actualizados desde el Apple Watch al iPhone, siempre que la última medición haya sucedido hace 5 minutos o más. Actualizando así dichos datos en los modelos del dominio de la aplicación del Apple Watch.	HU-04	T-06
T-10	Cierre de Sesión	Habiendo credenciales guardadas, la aplicación debe ser capaz de eliminar las credenciales de	HU-05	T-02

		CareLink™ del usuario, del Keychain del dispositivo.		
T-11	Diseño de interfaces	Diseño e implementación de interfaces estilizadas para una mejor experiencia de usuario.	HU02 HU-03 HU-04	T-05 T-07 T-08

En la Tabla 5 se muestra de manera ordenada cada una de las tareas propuestas para el desarrollo del software en cuestión. Es especialmente importante notar las dependencias entre tareas, ya que esto será usado como punto de partida para la organización del plan de trabajo planteado a continuación.

4.4.3. Plan de Trabajo

En este apartado se presenta una propuesta de plan de trabajo, en la cual se segmenta el desarrollo de las tareas en tres iteraciones, con la intención de realizar 3 entregas, donde cada entrega tiene la intención de estar acorde a la fecha de entrega del trabajo de grado, y debería corresponder con las siguientes fechas:

- Primera entrega: 11 de noviembre de 2021.
- Segunda entrega: 23 de diciembre de 2021.
- Tercera entrega: 10 de febrero de 2022.

Figura 21. Plan de trabajo de tareas segmentado en 3 entregas.

Se presenta en la Figura 21 el plan de trabajo propuesto, donde se muestran las restricciones que proveen las dependencias entre las tareas, de manera horizontal. Y a su vez se engloban en 3 grupos de tareas, generando así una planeación de desarrollo de todas las tareas presentes en el proyecto.

De esta manera, con el plan de desarrollo ya estructurado, esta planificación pretende en la primera entrega: realizar una prueba de conceptos válida. Validando que los datos a acceder, que serán los pilares de la aplicación, son accesibles. Ya que el acceso a los datos de la API CareLink™ viene dado por la reescritura de una librería ya existente en otro lenguaje.

La segunda entrega se enfoca en el desarrollo de la transferencia y presentación de los datos en el dispositivo Apple iPhone, así como el envío de los datos al Apple Watch, sin entrar en detalles de estilos ni visualización.

Al final de la segunda entrega, se debería tener como resultado un prototipo básico, con los datos necesarios en los dispositivos necesarios. Pero aun faltando la presentación correcta y estilizada de los datos.

Por último, en la tercera entrega el foco del desarrollo será mayormente en la parte visual. Enfocándose en la presentación de los datos ya disponibles, y a su vez la estilización y estética de estas presentaciones de datos.

4.5. Arquitectura del software

En este apartado se procederá a describir en detalle la arquitectura de código utilizada para el desarrollo de la aplicación en cuestión. Pero antes de esto, se debe saber qué es la arquitectura de un software, y qué problemas resuelve.

Según (Martin, 2018), la arquitectura de software es la forma dada al sistema por aquellos que lo construyeron. Cuyo propósito es facilitar el desarrollo, despliegue y mantenimiento del código contenido. De manera que se pretende facilitar el desarrollo de software, separando las responsabilidades en distintas capas. Y especialmente facilitar el mantenimiento a la hora de querer cambiar componentes o agregar nuevas funcionalidades, de manera eficiente respecto al código ya escrito.

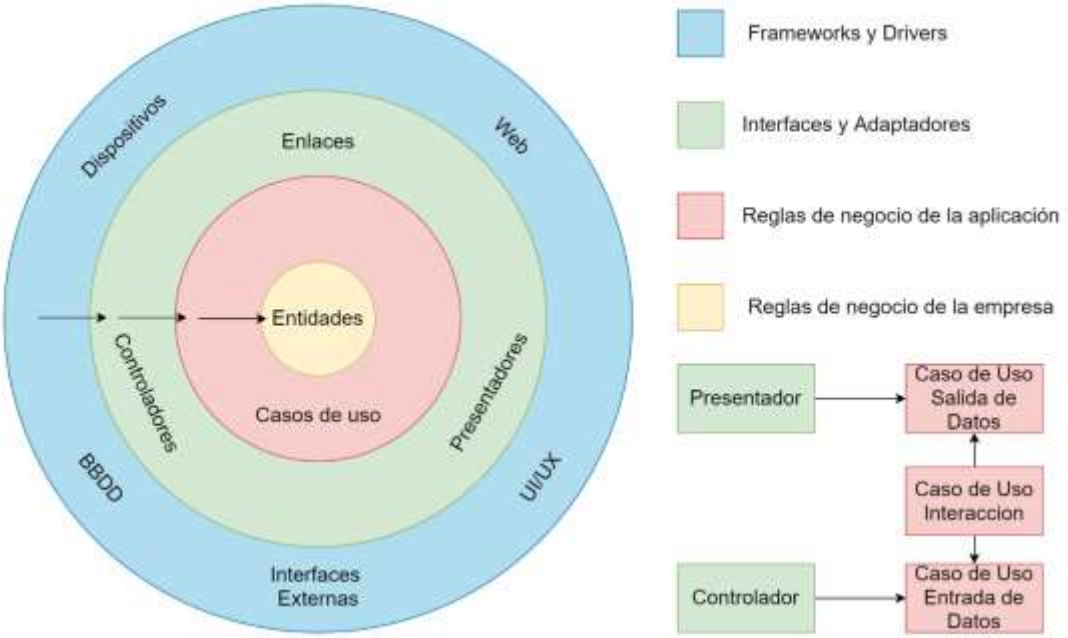

Figura 22. Diagrama concéntrico de la Arquitectura Limpia. Adaptado de (Martin, 2018).

La arquitectura de software que se utilizará en este trabajo se basa en la arquitectura propuesta por (Martin, 2018), mostrada en la Figura 22. Esta arquitectura nace como una unión de otros tipos de arquitecturas basadas en capas, las cuales tienen entre ellas en común, que producen sistemas que cumplen las siguientes características:

- Son independientes de las librerías que se utilizan.
- Las reglas de negocio son corroborables mediante pruebas unitarias, sin depender de elementos externos como un servidor web o una base de datos.
- Son independientes respecto a las interfaces de usuario. Estas pueden ser cambiadas sin cambiar el resto del sistema.
- Son independientes de las bases de datos, el manejador utilizado puede ser cualquiera y las reglas de negocio no están ligadas a este.
- Son independientes a cualquier agente externo, donde las reglas de negocio no conocen nada acerca de las interfaces fuera de ellas.

A partir de esto, Martin nos presenta la arquitectura limpia, una arquitectura que separa el software en al menos 4 capas. Estas capas son representadas como círculos concéntricos, siendo cada aro interno un nivel de la arquitectura. Acoplando además todas las reglas y

restricciones presentadas. Esto permite que cada capa tenga un bajo acoplamiento con sus capas adyacentes, pero a su vez tengan una alta cohesión.

Martín añade como novedad a esta arquitectura la regla de la dependencia, que denota que una capa interna no puede depender de una capa externa. Es decir, las capas internas no pueden saber, referenciar ni depender de las capas externas, pero sí las capas externas de las capas internas (Martin, 2018).

A continuación, se muestra la arquitectura del software a desarrollar organizado en capas, tal y como propone la arquitectura limpia. Se separa la aplicación global en 3 bloques, con la intención de hacer los diagramas más legibles, y agruparlos en base a funcionalidades.

A lo largo de los diagramas, se presentan 4 capas, donde cada capa tiene un color característico: Azul para *Frameworks* y Drivers, verde para Interfaces y Adaptadores, Rojo para Casos de Uso y amarillo para Entidades.

Además de esto, existen 2 casos especiales en los diagramas: el primero se presenta en la capa más externa, donde las cajas con borde más grueso denotan que son generadores de acciones. Tales como interacciones de usuario o recepción de mensajes. El segundo caso se ve en las dos capas intermedias, donde existe para cada capa una subcapa. Esta subcapa es representada sin fondo, y su finalidad es, como propone (Martin, 2018), generar un control de inversión mediante interfaces con la finalidad de solventar el problema de dependencia cuando se necesita llamar a una capa superior, sin romper la regla de dependencias de la arquitectura limpia de Martin. Veamos a continuación los bloques propuestos que, como un todo, componen la arquitectura de la aplicación.

El primer bloque corresponde a los requisitos referentes al inicio y cerrado de sesión, es decir, las historias de usuario HU-01 y HU-05. En este diagrama, presentado en la Figura 23, se muestra en la capa más exterior: 2 vistas de usuario y 2 entes externos. Las vistas de usuario pueden ambas generar una interacción de usuario que conllevará en una acción específica, el inicio y cerrado de sesión correspondientemente. En cambio, los 2 entes externos corresponden a: la fuente de datos y el Framework usado para el guardado de las credenciales del usuario, haciendo que no sean generadores de acciones.

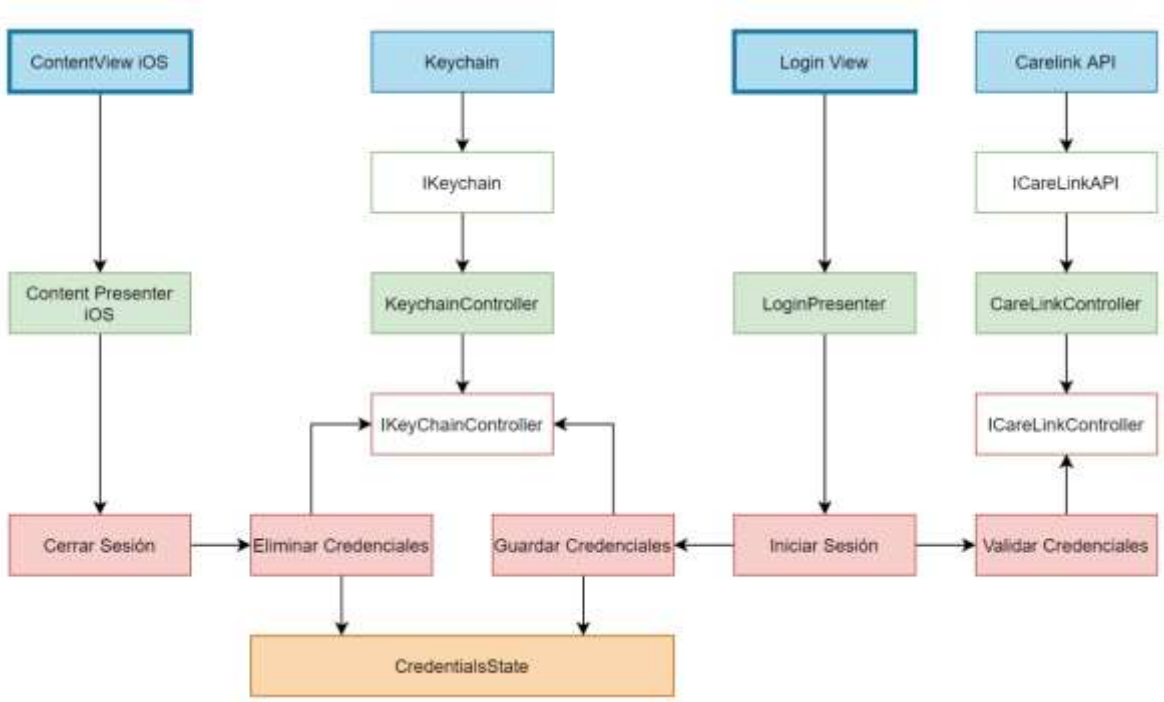

Figura 23. Representación en arquitectura limpia de las historias de usuario HU-01 y HU-05.

Se pueden observar en la Figura 23 los casos de uso primarios "Cerrar Sesión" e "Iniciar Sesión", pero estos dependen de otros casos de uso más específicos. Los cuáles serán explicados en detalle en el apartado de casos de uso. Especialmente se debe notar que las acciones de usuario siguen líneas de dependencia directamente descendentes. Pero estas necesitan usar los entes externos, haciendo que existan interfaces para el uso, mediante control de inversión, de estas utilidades desde capas más internas de la aplicación.

El segundo bloque corresponde a las interacciones del usuario en su dispositivo Apple iPhone, y al flujo de los datos asociados a estas interacciones, en particular, se habla de la historia de usuario HU-02. Partiendo de un usuario con credenciales guardadas, se muestra cómo debe funcionar el flujo de datos desde que el usuario quiere ver sus datos, cómo se obtienen estos datos de la API CareLink™ y el guardado de los datos en las entidades correspondientes.

Se muestra también el flujo de envío de mensajes, mediante el *Framework* WCSession, desde el dispositivo móvil hacia el Apple Watch. Debido a que esta interacción pertenece directamente al dispositivo móvil, no se hace referencia al dispositivo de muñeca ya que todos los detalles de transporte y conexión son manejados por el *Framework* en cuestión.

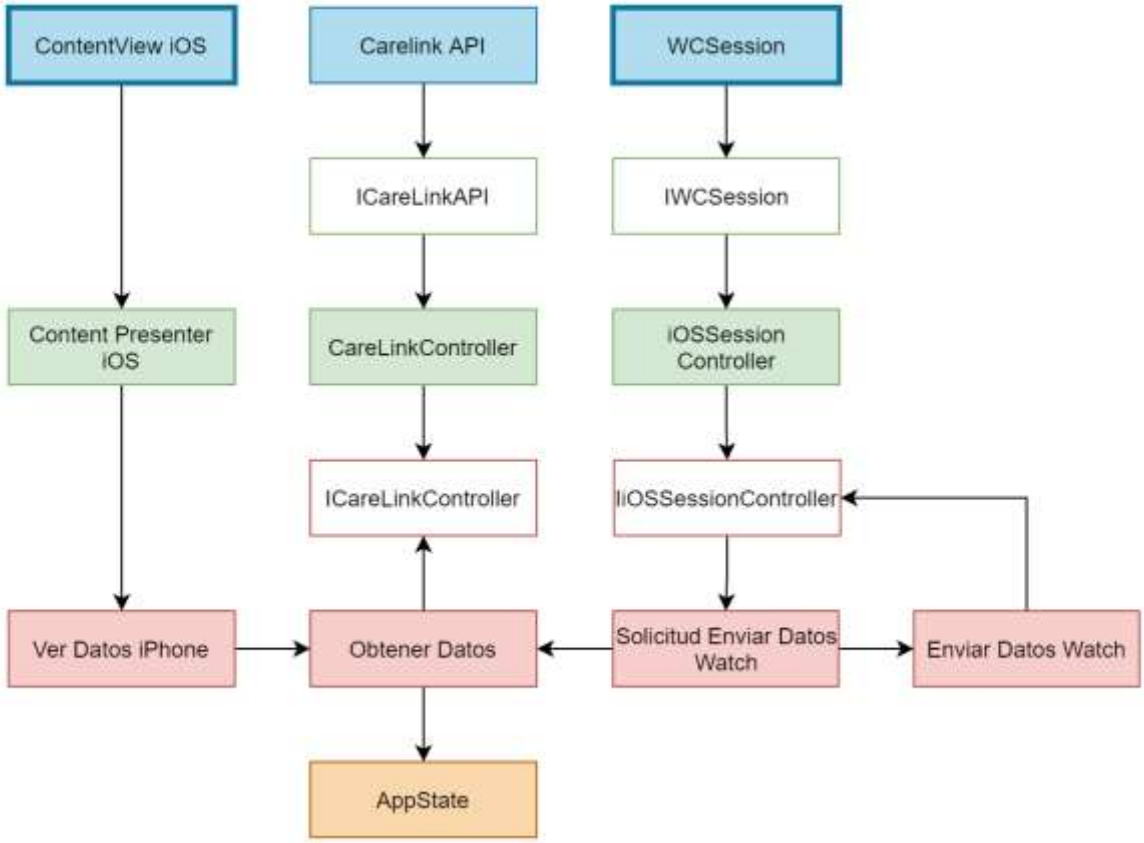

Figura 24. Representación en arquitectura limpia de la historia de usuario HU-02 y el flujo de datos desde el iPhone hacia el Apple Watch.

En la Figura 24 se puede observar que el Framework WCSession está marcado como un ente que puede iniciar una acción. Y esto se debe a que, desde el dispositivo de muñeca, se generarán peticiones de actualización que, mediante el Framework, serán recibidas en el dispositivo móvil. Y se encargará de llamar a los casos de uso correspondientes para obtener los nuevos datos y proceder a enviar estos datos, mediante un nuevo mensaje.

Por último, el tercer bloque corresponde a las interacciones del usuario con su Apple Watch, las cuales incluyen las interacciones en la aplicación misma, y en las complicaciones. Esto se corresponde con las historias de usuario HU-03 y HU-04. Este bloque contempla también la recepción de los datos desde el iPhone, mediante el Framework WCSession, y el envío de mensajes, desde el Watch hacia el iPhone, para la petición de datos.

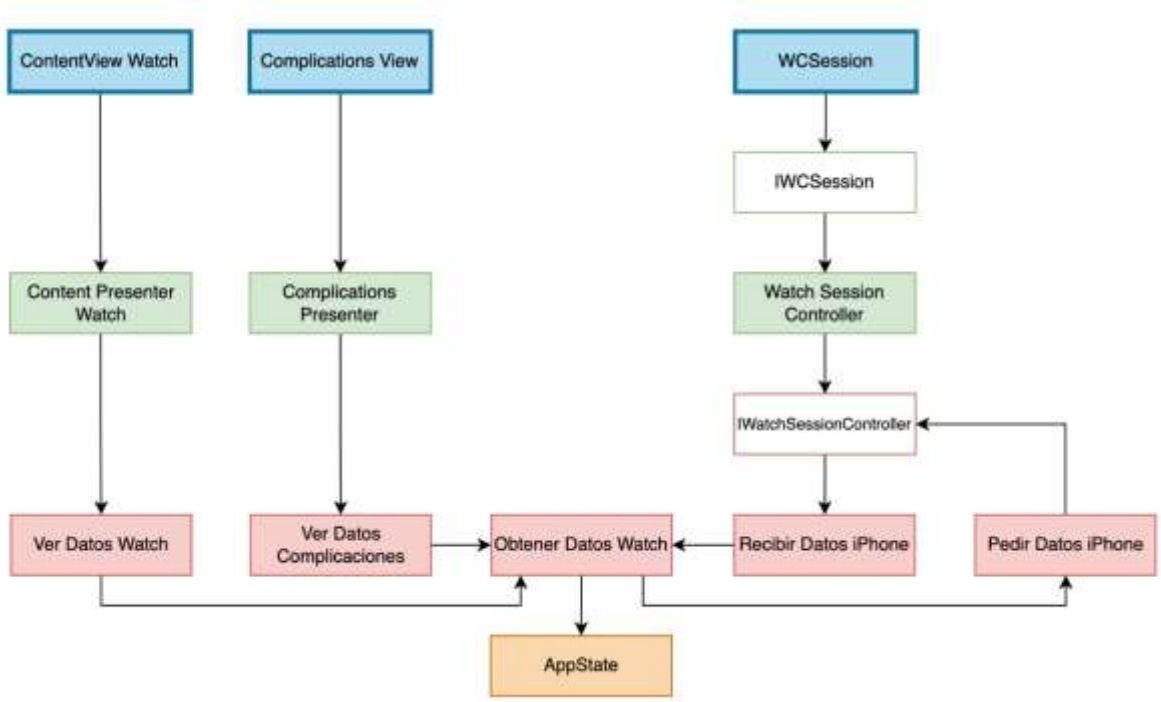

Figura 25. Representación en arquitectura limpia de las historias de usuario HU-03, HU-04 y el flujo de datos desde el Apple Watch hacia el iPhone.

En la Figura 25 se puede observar que todos los elementos de las capas externas son iniciadores de acciones, y estas son relativamente sencillas, ya que no incluyen ninguna lógica. Más allá de la petición y recepción de mensajes, y el guardado de los datos para la presentación. A continuación, se presenta en detalle cada una de las capas de la aplicación.

4.5.1. *Frameworks* y *Drivers*

Esta es la capa más externa de la aplicación, generalmente compuesta de *Frameworks* y herramientas tales como Bases de Datos, interfaces de usuario o dispositivos externos. Generalmente no se escribe mucho código en esta capa, más allá de la comunicación con la capa siguiente (Martin, 2018).

En los diagramas de la aplicación, esta capa se ve reflejada como las cajas de color azul. Y si el borde se encuentra presentado de forma más gruesa, denota que ese elemento es un desencadenante de una acción, por ejemplo: la entrada de datos del usuario.

En esta capa se encuentran elementos tales como: las vistas de la aplicación, las cuales serán escritas utilizando el *Framework* SwiftUI, nativo para escribir vistas en aplicaciones para

dispositivos Apple. Se incluye también en esta capa el *Framework* WCSession de comunicación mediante mensajes entre el iPhone y el Apple Watch. EL fabricante (Apple Inc., 2022e) define este Framework como: un objeto que inicia la comunicación entre una aplicación de iOS y su contraparte en el Apple Watch. Y será utilizado siguiendo los lineamientos y restricciones propuestos por el fabricante.

Por último, se incluye en esta capa el *Framework* Keychain, también de Apple, el cual permite guardar de manera segura los datos, en específico las credenciales, a nombre del usuario de manera cifrada. (Apple Inc., 2022b). Y la API externa de CareLink™, la cual será la fuente de datos dentro de la aplicación es realmente el único código escrito en esta capa. Correspondiendo este a la reescritura de la librería de (Szász, 2021) en el lenguaje Swift.

```swift
//
//  CareLinkAPI.swift
//
//
//  Created by Arnaldo Quintero on 9/2/22.
//

import Foundation

public class CareLinkAPI: ICareLinkAPI {

    public static let singleton: ICareLinkAPI = CareLinkAPI()

    public func login(username: String, password: String) async throws -> HTTPCookie {...    }

    public func getCountrySettings() async throws -> CountrySettings {...    }

    public func getUserRole(token t: String) async throws -> UserSettings {...    }

    public func getData(url dataUrl: String, username: String, role: String, token t: String) async throws -> DataResponse {
        let headers = [
            "Content-Type": "application/json",
            "Authorization": "Bearer \(t)"
        ]
        let body = [
            "username": username,
            "role": role
        ]
        guard let jsonBody = try? JSONEncoder().encode(body) else {
            throw HttpErrors.JsonSerializationError
        }

        guard let (data, _) = try? await makeRequest(url: dataUrl, method: .POST, headers: headers, body: jsonBody) else {
            throw HttpErrors.GetDataError
        }

        let stringData =  data.data(using: .utf8)!
        guard let json = try? JSONDecoder().decode(DataResponse.self, from: stringData)
        else {
            throw HttpErrors.JsonSerializationError
        }

        return json
    }

}
```

Figura 26. Código para la obtención de datos de la API CareLink™.

En la Figura 26 se puede observar parcialmente el código correspondiente a la reescritura de la librería mencionada anteriormente, dónde existen llamadas al método *makeRequest*, el cual no se encuentra referenciado dentro de este fichero, pero es el encargado de realizar cualquier tipo de llamada HTTP.

A pesar de que este también es dependiente de Frameworks de Apple para realizar llamadas HTTP, dado que su única intención es ser usado por esta API, se decide englobarlo dentro del elemento de la API CareLink™ en los diagramas. Y se ve planteado en el código bajo la carpeta de servicios y utilidades, las cuales son únicamente una colección de métodos estáticos y sin contexto disponibles para el uso genérico de la aplicación.

4.5.2. Interfaces y Adaptadores

La siguiente capa, de interfaces y adaptadores, corresponde a una serie de clases y entidades que se encargan de transformar los modelos de las capas externas, a los modelos más convenientes para los casos de uso y entidades de la capa siguiente (Martin, 2018). En esta capa se encontrarán 3 tipos de elementos: presentadores, controladores e interfaces, los cuales se verán reflejados en los diagramas como los recuadros de color verde.

Los presentadores son clases, cuya funcionalidad es únicamente ser usado por las vistas para mostrar datos. Mientras que los controladores se encargan de modelar datos de interfaces externas tales como WCSession, Keychain o la API de CareLink™ para el uso de estas. Funcionalmente realizan ambos el mismo tipo de trabajo, pero por facilidades de calificación y clasificación, se separan en estos 2 términos distintos.

El tercer elemento, las interfaces, es una herramienta que utiliza (Martin, 2018) para no romper la regla de dependencia de la arquitectura. Estas se generan siguiendo el principio de inversión de dependencia, de manera que la interfaz genere dependencias de código siempre hacia las capas internas del círculo, exponiendo así métodos de las capas superiores a un nivel más interno dentro del diagrama.

Son las interfaces las que se encuentran en una subcapa especial, ya que son usadas casi únicamente por la inversión de dependencia. Por lo tanto, cuando estas aparecen es porque se quiere acceder a una capa superior desde una llamada de una capa inferior, o del mismo nivel. Por ejemplo: cuando se quiere acceder a la API de CareLink™, para la obtención de datos, luego de una interacción del usuario. Desplazándose a través de los casos de uso necesarios para guardar y presentar los datos sin romper la regla de las dependencias.

```swift
//
//  ICareLinkAPI.swift
//
//
//  Created by Arnaldo Quintero on 9/2/22.
//

import Foundation

public protocol ICareLinkAPI {
    func login(username: String, password: String) async throws -> HTTPCookie

    func getCountrySettings() async throws -> CountrySettings

    func getUserRole(token t: String) async throws -> UserSettings

    func getData(url dataUrl: String, username: String, role: String, token t: String) async throws -> DataResponse
}
```

Figura 27. Código para la interfaz de la API CareLink™.

Siguiendo con el caso mostrado en el apartado anterior, se puede observar en la Figura 27, la definición de la interfaz correspondiente a los métodos de la Figura 26. Estos son realmente los métodos públicos que utilizarán las capas internas de la aplicación en cualquier momento que se quiera acceder a los datos de la API CareLink™.

4.5.3. Casos de uso

La capa de casos de uso, también llamada de reglas de negocio de la aplicación, la define (Martin, 2018) como la capa que contiene el software que encapsula e implementa todos los casos de uso del sistema. Orquestando el flujo datos desde y hacia las entidades de manera que se pueda cumplir el objetivo del caso de uso.

Con este enfoque, Martin pretende aislar todas las reglas de negocio críticas mediante agrupaciones por funcionalidad. De manera que se encuentren completamente desligadas de entes externos, haciendo así que estas sean fácilmente verificables mediante pruebas unitarias. Y dado que los casos de uso se ven normalmente relacionados entre sí, se pueden mantener las referencias de código dentro de la misma capa para obtener una alta cohesión, pero un bajo acoplamiento entre estas distintas partes del software.

En esta capa se modelarán directamente los casos de uso de la aplicación, los cuales son una representación de las historias de uso obtenidas anteriormente. Detallados de manera que, tal y como las historias de usuario, cumplan con los requisitos funcionales y no funcionales propuestos para el desarrollo. Dando a su vez una facilidad para el desarrollo y validación de software de la aplicación de manera modular y ordenada.

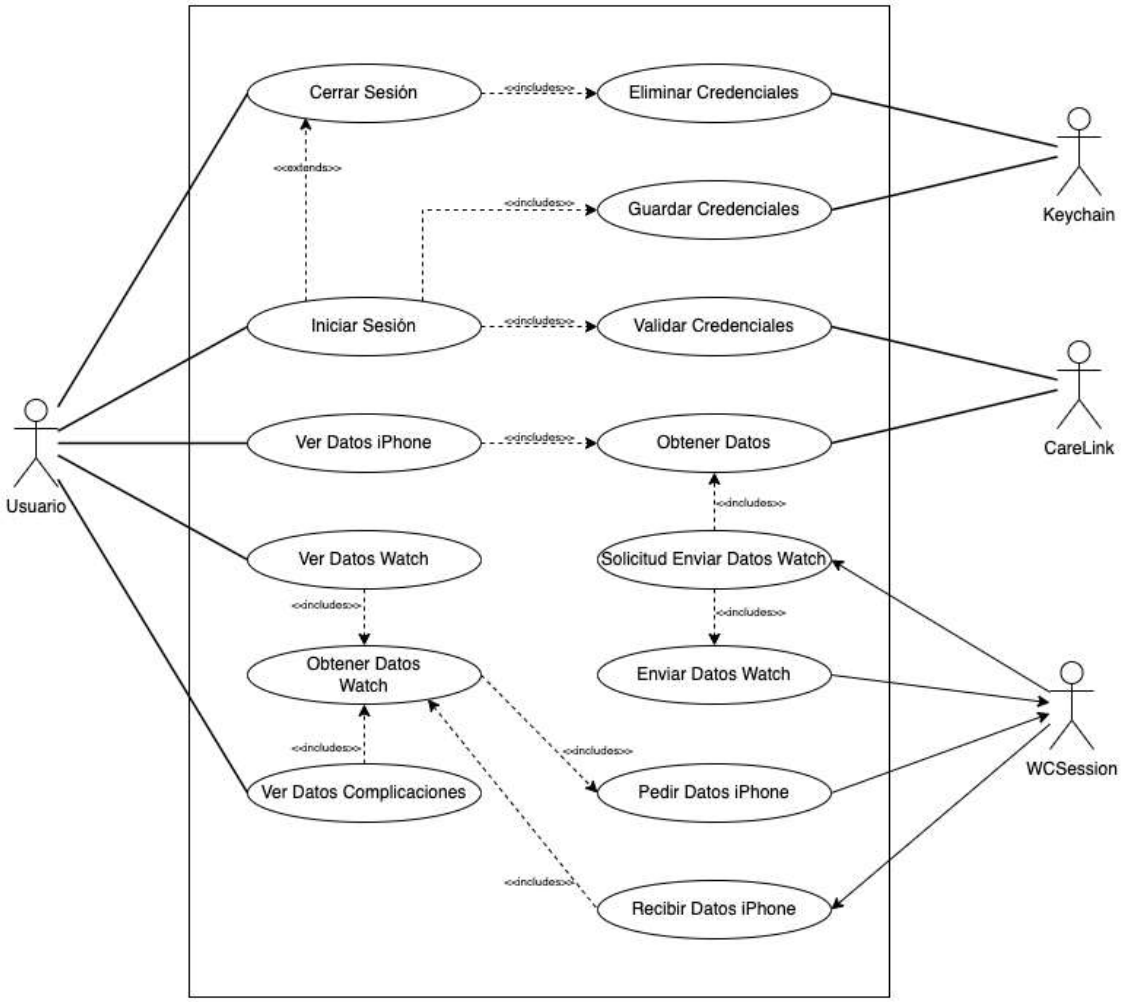

Figura 28. Diagrama de casos de uso.

Se puede observar en la Figura 28 que las 5 historias de usuario presentadas en el apartado 4.4.1 son reflejadas directamente como casos de uso ligados a los usuarios. Y que a su vez estos se encuentran desglosados, de manera que puedan corresponderse con los casos de uso propuestos por la aplicación de manera general. Tal y como se vieron reflejados en las Figuras 23, 24 y 25, en la capa de casos de uso.

El diagrama de casos de uso deja ver además qué casos de uso se encuentran ligados a qué agentes externos, siendo los agentes externos los representados en la parte externa a la derecha e izquierda del recuadro. De esta manera se puede interpretar que todo lo que se encuentra dentro del recuadro pertenece a la capa de casos de uso del software, y lo que se encuentra fuera de este pertenece a la primera capa, de Frameworks y Drivers.

A efectos del diagrama de casos de uso, la capa de controladores y presentadores, intermedia entre las dos mencionadas anteriormente, es ignorada. Debido a que su funcionalidad es principalmente traducción de modelos entre una capa y la otra.

```swift
//
//  GetDataUseCase.swift
//
//  Created by Arnaldo Quintero on 9/2/22.
//

import Foundation

class GetDataUseCase {

    public static let singleton = GetDataUseCase()
    private let carelink: ICareLinkController = CareLinkController.singleton
    private var appState: AppState?
    private var semaphore = DispatchSemaphore(value: 1)

    public func getLatestData(completion: ((AppState) -> Void)? = nil) {
        if let data = appState, data.isValid() {
            self.handleCompletion(completion)
            return
        } else {
            semaphore.wait()
            if appState == nil || !appState!.isValid() {
                getNewData() { newData in
                    self.semaphore.signal()
                    let sendDataToWatch = SendDataToWatch.singleton
                    sendDataToWatch.send(newData)
                    self.handleCompletion(completion)
                }
            } else {
                semaphore.signal()
                self.handleCompletion(completion)
            }
        }
    }

    private func getNewData(completion: ((AppState) -> Void)? = nil) {
        Task.init {
            let data = try await carelink.getLastSensorGlucose()
            let newGs = GlucoseModel(value: data.lastSG.sg)
            let newGsTrend = GlucoseTrendModel(trend: data.lastSGTrend)
            let newGsTime = GlucoseTimeModel(dateTime: data.lastSG.datetime)
            let newSensorState = data.lastSG.sensorState
            let newState = AppState(gs: newGs, gsTrend: newGsTrend, gsTime: newGsTime, sensorState: newSensorState)

            if (appState == nil){ appState = newState }
            else { appState?.updateNeededFields(from: newState) }
            self.handleCompletion(completion)
        }
    }

    private func handleCompletion(_ completion: ((AppState) -> Void)?) {
        if (completion != nil) { completion!(self.appState!) }
    }
}
```

Figura 29. Código para el caso de uso de obtención de datos.

En el caso de uso de obtención de datos mostrado en la Figura 29, el cual es un caso de uso interno, se pueden ver las referencias al agente externo CareLink, mediante su interfaz; a la entidad correspondiente, y al caso de uso de envío de datos al Watch. Este último no puede ser inicializado al momento de creación de la clase, por errores del ciclo de vida, lo cual

conlleva que su uso sea mediante un *singleton* directamente cuando se obtengan datos nuevos que deban ser enviados al Watch.

Es importante además mencionar que la petición de datos está regulada mediante un semáforo, para evitar así que se realicen una serie de peticiones a la API sin haber esperado por la respuesta de la petición anterior.

4.5.4. Entidades

La capa más interna, de entidades, según (Martin, 2018) es la capa que encapsula las reglas de negocio que son comunes a nivel de empresa. Las cuales pueden ser objetos con métodos, o una serie de estructuras de datos y funciones. Siempre y cuando puedan ser usados por distintas aplicaciones a lo largo de la empresa.

Para este desarrollo de software, las entidades representan la fuente de datos y el manejo de estos para su tratamiento. Es decir, las entidades son una serie de clases, las cuales poseen a su vez métodos adecuados para que dichos datos sean presentados en el formato correcto a lo largo de todas las aplicaciones.

En las Figuras 23, 24 y 25 se puede observar que solo se reflejan 2 entidades: *AppState* y *CredentialsState*, pero por simplicidad del diseño no se detallan las subentidades que las componen, las cuales se verán en detalle a continuación:

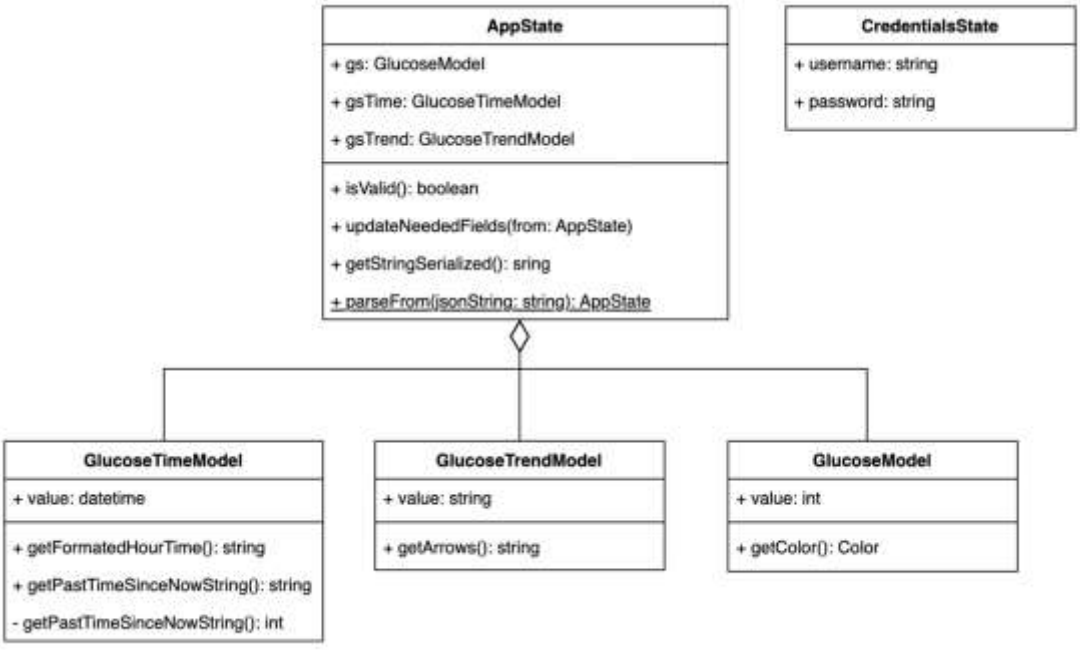

Figura 30. Diagrama de clases correspondiente a las entidades de la aplicación.

La intención de estas entidades de la aplicación es que guarden directamente la información actual que compete a la aplicación. La Figura 30 muestra las entidades presentes en la aplicación, de las cuales se estudiará especialmente la clase *AppState*.

Esta clase o entidad se encarga del guardado del estado actual de la glucemia: la tendencia, el último resultado obtenido, entre otras propiedades. Así como los métodos correspondientes para obtener el lapso en minutos desde la última actualización de datos a la fecha actual ,mediante el uso de otras clases o subentidades, que a su vez tienen métodos para la obtención y tratamiento de los datos, siguiendo la lógica deseada en la aplicación.

Por otro lado, la clase *CredentialsState* se presenta aislada al resto de las clases, ya que las credenciales en sí se guardan en el dispositivo, a nombre del usuario, usando el Framework Keychain de Apple. Haciendo así que esta clase sea usada únicamente como modelado de datos entre los casos de uso relacionados al guardado de credenciales.

4.6. Tecnologías utilizadas

En este apartado se listarán las tecnologías utilizadas para llevar a cabo el desarrollo de la aplicación en cuestión, explorando desde los lenguajes de programación hasta los *Frameworks* y librerías utilizadas. Con el fin de proveer un poco de contexto técnico respecto a las herramientas utilizadas.

4.6.1. Swift

Según su desarrollador (Apple Inc., 2019) “Swift es un intuitivo lenguaje de programación creado por Apple que permite diseñar apps para iOS, Mac, el Apple TV y el Apple Watch … de código abierto y fácil de usar.” Dado que la aplicación está enmarcada para dispositivos Apple, en particular para iPhone y Apple Watch, es Swift la elección propuesta por el mismo fabricante de estos productos, para el desarrollo de aplicaciones.

4.6.2. SwiftUI

SwiftUI es un conjunto de herramientas que permiten a los desarrolladores construir interfaces visuales mediante el uso del lenguaje Swift, utilizando una sintaxis declarativa. Estando así optimizado para la accesibilidad y la experiencia de usuario en aplicaciones desarrolladas con Swift (Apple Inc., 2022f). Este set de herramientas es relativamente nuevo,

y se utiliza debido a su sencillez respecto a su predecesor, y su sintaxis declarativa que permite la lectura y escritura del código de manera más amigable.

4.6.3. WatchKit

Según (Apple Inc., 2022d) WatchKit es el *Framework* que provee la infraestructura para desarrollar aplicaciones para Apple Watch, incluyendo el manejo de servicios tales como tareas en segundo plano. Este Framework es absolutamente necesario a la hora de desarrollar aplicaciones para Apple Watch, ya que son el conjunto básico de herramientas de trabajo para cualquier tarea a realizar en este entorno.

4.6.4. ClockKit

ClockKit es un *Framework* para mostrar información en las carátulas de un Apple Watch. Provee las herramientas necesarias para el desarrollo de complicaciones. Las cuales son pequeñas interfaces que aparecen en las carátulas y proveen al usuario un acceso rápido a datos que normalmente usa (Apple Inc., 2022a).

Dado que la finalidad de la aplicación es mostrar los datos en las carátulas, mediante complicaciones, se debe utilizar ClockKit como herramientas de trabajo y gestión. Ya que es esta la única manera proporcionada por Apple para realizar estas tareas.

4.6.5. Keychain

Los servicios de Keychain son un conjunto de herramientas que permite guardar de manera segura trozos de datos en una base de datos encriptada, a nombre del usuario. (Apple Inc., 2022b). Suelen ser utilizados para guardar datos tales como contraseñas, certificados de identidad, claves criptográficas, y en general cualquier tipo de dato sensible.

Para el servicio de Keychain, se utiliza la librería de código libre *SwiftKeychainWrapper*. Elaborada por (jrendel, 2020), cuya funcionalidad es proveer una interfaz simplificada de acceso a la librería Keychain de Apple. Por lo tanto, el uso de este Framework para el guardado de credenciales se realizará siempre a través de esta librería de terceros.

4.6.6. WCSession

El Framework WCSession de comunicación entre el iPhone y el Apple Watch es un objeto que inicia la comunicación entre una aplicación de iOS y su contraparte en el Apple Watch, permitiendo el paso de mensajes entre ambas partes. (Apple Inc., 2022e). Esta es una de las

posibles formas de comunicación de mensajes entre dispositivos Apple, y es la opción escogida ya que permite enviar mensajes desde ambos dispositivos entre sí, permitiendo de esta manera una comunicación fluida, asíncrona y manejada completamente por Apple.

4.6.7. XCode

XCode es el IDE (entorno de desarrollo integrado) de Apple, que permite a los usuarios desarrollar, validar y publicar las aplicaciones de todo el ecosistema Apple, haciendo uso de su entorno de desarrollo. Incluyendo utilidades tales como simuladores de dispositivos físicos, modelos de Machine Learning y herramientas de depuración (Apple Inc., 2022g).

Dado que se desarrollan aplicaciones para Apple, lo más recomendable es usar su IDE recomendado, de manera que se puedan aprovechar las funcionalidades tales como los simuladores, o incluso la previsualización de vistas sin necesidad de simuladores. Haciendo así el proceso de desarrollo del software más amigable y ameno.

4.7. Evaluación

Luego de desarrollado el software, se debe validar que este cumple con los requisitos planteados, y que realiza de manera correcta las tareas que debe realizar. Para ello, en este apartado se presentan una serie de evaluaciones y validaciones realizadas sobre la aplicación ya desarrollada, con el objetivo de corroborar qué tan bien cumple los objetivos y requisitos planteados al comienzo de este documento.

Para ello, inicialmente se presenta una serie de pruebas, las cuales son realizadas a nivel de código, conteniendo pruebas unitarias, pruebas de interacción y pruebas de lanzamiento. Con la intención de validar que las reglas de negocio sean cumplidas, que el tratamiento de los datos se genere de forma correcta, que la interacción con los usuarios sea correcta, y que el tiempo de carga de la aplicación sea apto.

Seguidamente, se presenta una comparativa entre las interfaces desarrolladas y los prototipos planteados, junto a una corta discusión de las diferencias entre estos, en caso de existir, con intención de validar la completitud y validez de los prototipos desarrollados en los apartados anteriores.

Por último, se plantea una breve evaluación punto a punto sobre los requisitos planteados, realizada mediante una encuesta de usuario, la cual incluye además posibles sugerencias sobre el funcionamiento y desarrollo de la aplicación.

4.7.1. Pruebas unitarias

Dado que una de las ventajas principales de la arquitectura limpia de Martin, es que el código pueda ser fácilmente probado y validado. Se han realizado pruebas unitarias sobre las entidades generadas, en el proceso de desarrollo de la aplicación, con la intención de verificar la fiabilidad y correctitud de la capa de entidades de la aplicación, la encargada de manejar la lógica base según la arquitectura de Martin.

En este apartado se validarán, para cada entidad, sus constructores y la funcionalidad esperada de cada uno de sus métodos, tales como los comparadores de igualdad para cada modelo, o la obtención de datos en formatos especiales requeridos por la aplicación.

Para las pruebas unitarias, se utilizó la herramienta de pruebas XCTest de Apple, la cual permite realizar pruebas que se integran con el entorno de desarrollo de aplicaciones de Apple en su IDE XCode. Estas pruebas se basan en aserciones, que son condiciones propuestas para validar la lógica de la aplicación. Y en caso de no ser satisfechas, se denota la prueba como fallida. En caso contrario como exitosas (Apple Inc., 2022h).

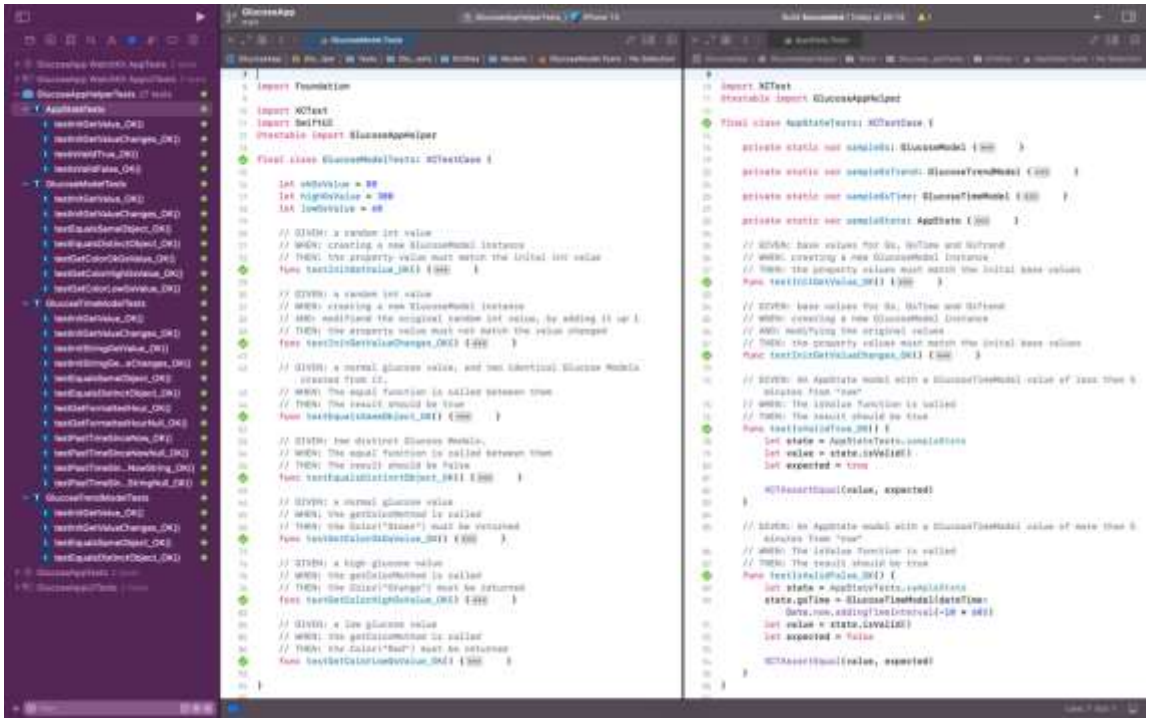

Figura 31. Ejecución de las pruebas unitarias.

En la Figura 31 se pueden observar, a la izquierda, todas las definiciones de las pruebas de validación creadas para cada una de las entidades de la aplicación. De la misma manera, en el editor se encuentran abiertas las pruebas de las entidades *GlucoseModel* y *AppState*.

Para todas las clases validadas, se comprueba que los constructores de la clase guarden de manera correcta los valores introducidos, y que además este valor no cambie indeseadamente. Luego de esto, para cada clase se valida el comparador de igualdad, que compara los valores de las propiedades internas respecto a otro elemento del mismo tipo. Esta validación tiene especial complejidad para la clase *GlucoseTimeModel*, debido a que el manejo del tiempo viene dado en cadenas de texto, y se debe usar la misma precisión en segundos para realmente validar si ambas cadenas son iguales o no.

Por último, para cada clase, se procede a validar cada uno de sus métodos internos, en caso de existir. Probando los casos especiales y casos borde que puedan surgir. Por ejemplo, en la Figura 31 se pueden observar las validaciones al método *isValid* de la clase *AppState*. El cual denota que el estado es válido si su fecha está dentro de los últimos 5 minutos.

Estas validaciones nos permiten realizar cambios o actualizaciones en el código, sabiendo que, si se llegase a cambiar algo de manera no deseada, las pruebas de validación fallarían y se sabría exactamente qué está fallando. Haciendo el proceso de revisión y reparación más rápido y eficiente.

4.7.2. Pruebas de interacción

En el caso de las pruebas de interacción, se utilizó la familia *User Interface Tests* de Apple, las cuales permiten validar dos partes importantes de la aplicación: las interacciones de los usuarios y la conectividad entre dispositivos (Apple Inc., 2022c). Para probar las interacciones, se procederán a validar los siguientes casos:

1. Un usuario sin credenciales guardadas introduce credenciales incorrectas en el iPhone, presiona el botón de iniciar sesión y debe mantenerse en la página de inicio de sesión.
2. Un usuario sin credenciales guardadas introduce credenciales en el iPhone correctas, presiona el botón de iniciar sesión y debe ser redirigido a la página principal de la aplicación.

3. Un usuario con credenciales guardadas presiona el botón de cerrar sesión y debe ser redirigido a la página de inicio de sesión de la aplicación.
4. Un usuario con credenciales guardadas, sin interacción alguna, debe poder ver los datos de glucosa y tendencia en la pantalla principal.
5. Un usuario sin credenciales guardadas, al abrir la aplicación en el Apple Watch, no debe ser capaz de ver los datos de glucosa y tendencia en la pantalla principal.
6. Un usuario sin credenciales guardadas, y con una complicación activa en el Apple Watch, no debe poder ver ningún valor en la complicación.
7. Un usuario con credenciales guardadas, al abrir la aplicación en el Apple Watch, debe poder ver los datos de glucosa y tendencia en la pantalla principal.
8. Un usuario con credenciales guardadas, y con una complicación activa en el Apple Watch, debe poder ver el valor de la glucosa en la complicación.

Dado que se desean realizar pruebas para el iPhone y el Apple Watch, y a nivel de código son 2 proyectos separados. Se debe separar las pruebas de uno respecto al otro, siendo las del iPhone las pruebas del 1 al 4; y las del Apple Watch del 5 al 8.

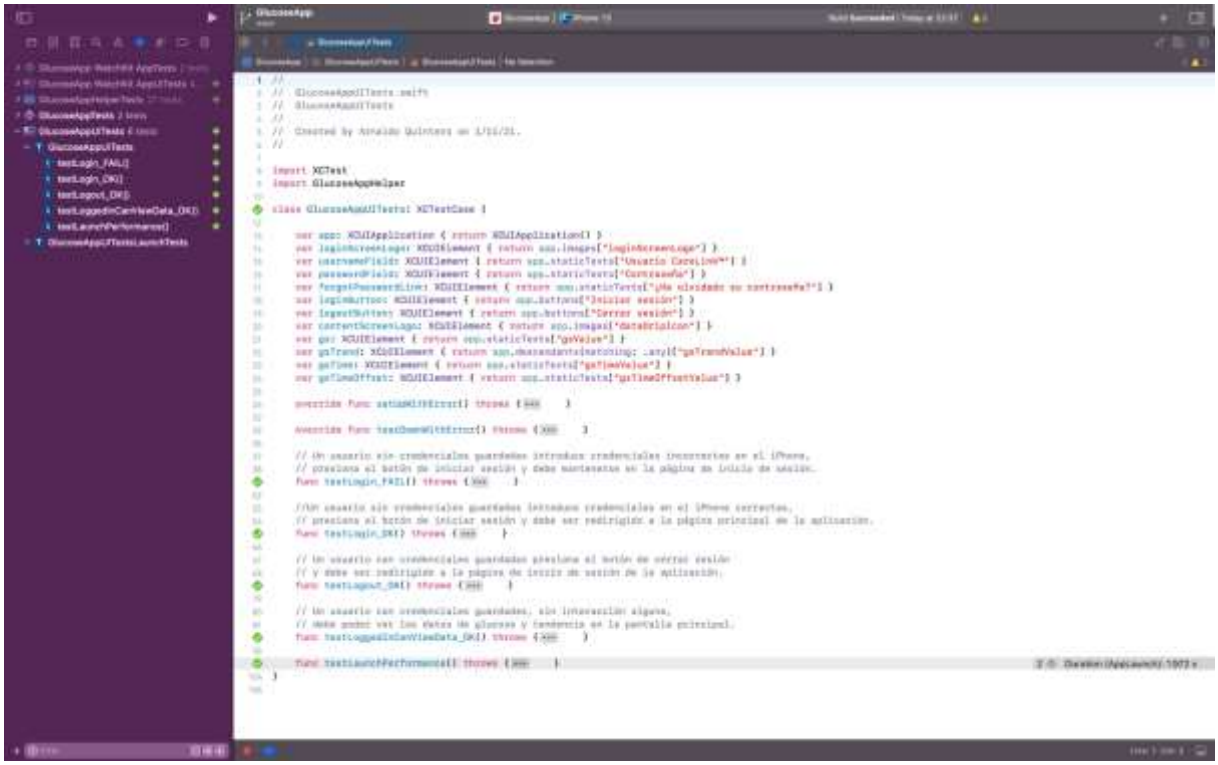

Figura 32. Ejecución de las pruebas de interacción del dispositivo Apple iPhone.

En la Figura 32 se pueden observar las capturas de la ejecución de las pruebas de interacción, donde se valida exactamente lo propuesto en los puntos 1 al 4 expuestos anteriormente. Se puede observar además todos los elementos que se utilizan para validar la aplicación. Tales como los logotipos mostrados, los campos de texto, y los valores de: glucemia (*gs*), tendencia de glucemia (*gsTrend*), tiempo de captura de la glucemia (*gsTime*), y el tiempo pasado desde la captura de esa glucemia (*gsTimeOffset*). Los cuales son los valores guardados normalmente en la entidad *AppState*, y son además los valores mostrados al usuario.

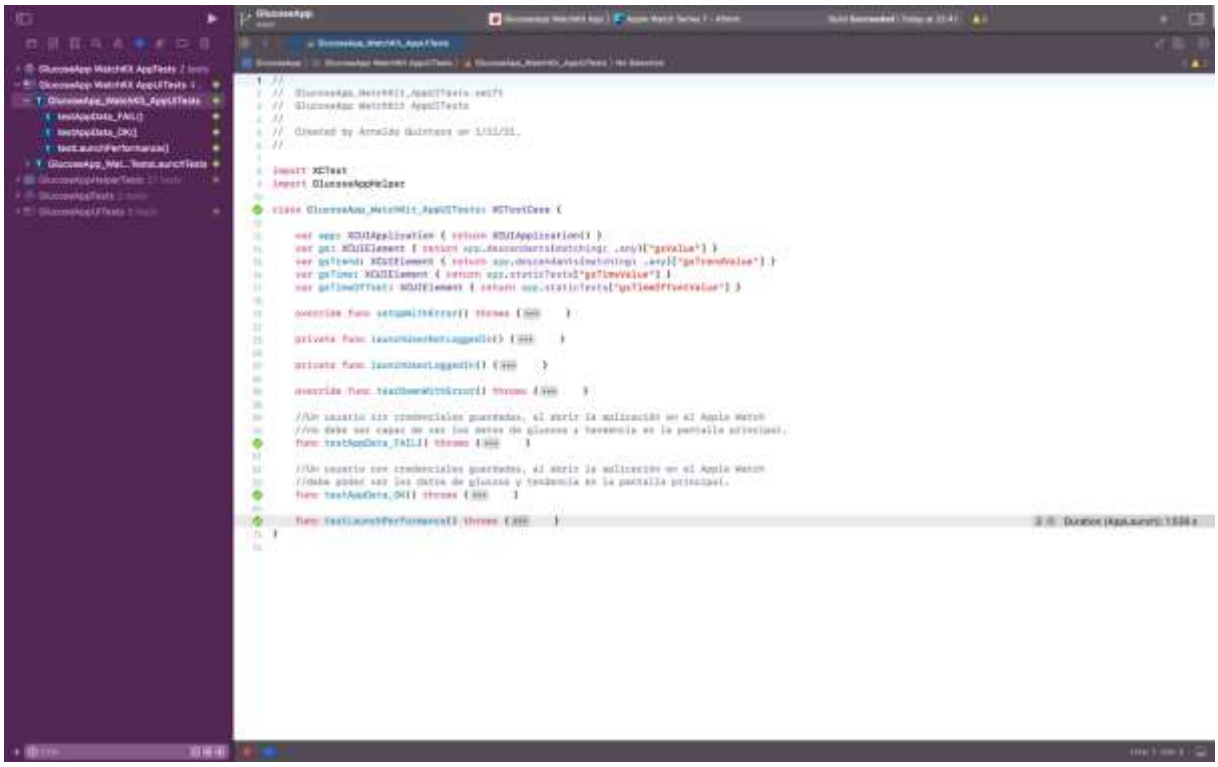

Figura 33. Ejecución de las pruebas de interacción del dispositivo Apple Watch.

Las pruebas planteadas para el Apple Watch incluyen pruebas para la aplicación misma, y pruebas para las complicaciones. Pero actualmente Apple no dispone de ningún *Framework* que permita validar o probar las complicaciones, ni nada asociado al ClockKit.

Por lo tanto, las pruebas que se muestran en la Figura 33, corresponden únicamente a las pruebas referentes a la aplicación del Apple Watch, es decir, las pruebas 5 y 7 de la lista anterior. Estas pruebas nos permiten validar que los usuarios pueden ver los datos al estar con credenciales válidas. Y que no ven datos si no tienen credenciales válidas.

4.7.3. Pruebas de lanzamiento

Las pruebas de lanzamiento nos permiten ver el tiempo que tarda cada aplicación en convertirse usable para el usuario. Esto nos permite validar qué cantidad de tiempo se quedará el usuario esperando, y validar si este tiempo es demasiado largo para hacer que el usuario se vea desinteresado respecto a la aplicación.

Según comenta (Harrison, 2019) las últimas recomendaciones de Apple, expuestas en el WWDC 2019, dicen que una aplicación debería mostrar su primer *frame*, o fotograma, antes de que pasen 4000 ms, es decir, se tomará como un tiempo de lanzamiento válido, si dicho tiempo se encuentra por debajo de los 4 segundos.

En las Figuras 32 y 33 se puede observar que sobre la prueba llamado *testLaunchPerformance*, a la derecha sobre la barra gris, se muestran los valores correspondientes a los tiempos de lanzamiento de las aplicaciones de iPhone y Apple Watch correspondientemente.

Estos valores son de 1,972 segundos para la aplicación móvil y 1,539 segundos para la aplicación del Watch. Al ver estos valores, respecto al valor de referencia propuesto por Apple, se puede observar que las aplicaciones se encuentran por debajo de la mitad del valor de referencia. Haciendo así que ambas partes de la aplicación tengan un tiempo de lanzamiento más que aceptable.

4.7.4. Pruebas de Interfaces

Ya teniendo las pruebas de validación de código, queda validar el diseño desarrollado respecto al diseño propuesto en los prototipos del apartado 4.3. Para ello, en este apartado, se procederá a mostrar el acabado final de cada una de las pantallas desarrolladas, obtenidas a partir de los simuladores de desarrollo del IDE XCode, respecto a los prototipos planteados anteriormente. Presentándose siempre la versión desarrollada a la izquierda y la versión prototipada a la derecha.

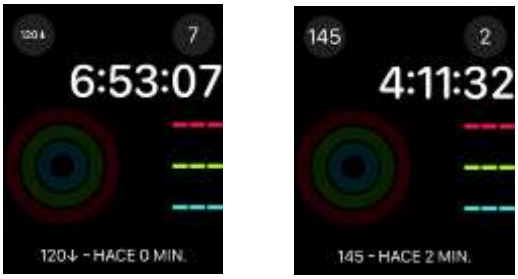

Figura 34. Comparativa de interfaces desarrolladas: complicaciones Apple Watch.

La figura 34 nos muestra un caso de las complicaciones desarrolladas, junto a el prototipo de este mismo caso. Se puede observar que la versión prototipada y la versión realizada son iguales. Esto se debe a que el desarrollo para complicaciones se ve siempre enmarcado por las directrices Apple, y la libertad existente para realizar cambios en las presentaciones de las complicaciones se ve muy limitada por el mismo fabricante. Debe comentarse también que en el apartado 4.3.5 se presentaba además una complicación conteniendo color y un pequeño icono. Pero dados los plazos de tiempo, y el poco valor que le aporta a la funcionalidad del proyecto, dicha complicación no ha sido desarrollada.

Figura 35. Comparativa de interfaces desarrolladas: aplicación Apple Watch.

Por otro lado, en la Figura 35 se puede ver que existen claras diferencias de tamaño entre el círculo que se muestra en la versión desarrollada, respecto a la versión diseñada. Esto se debe a que, a la hora de desarrollarlo, se ha reutilizado el componente visual usado en el iPhone, y simplemente se ha re escalado al tamaño correspondiente. Tratando así que encaje dentro de los márgenes del disponibles por parte de WatchKit.

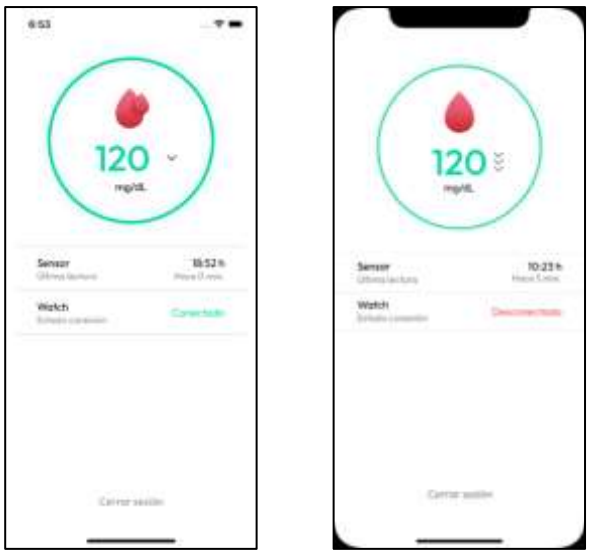

Figura 36. Comparativa de interfaces desarrolladas: pantalla principal iPhone.

Respecto a las interfaces desarrolladas para el iPhone, en la Figura 36 se puede ver la página principal de la aplicación. La cual nos muestra los datos de la glucemia y tendencia del paciente. En cuanto a estructuras, el diseño es completamente igual, sólo con la diferencia en el círculo superior, donde las líneas son más gruesas y el número no se encuentra del todo centrado. Además de esto, se ha decidido usar el isotipo de la aplicación en vez de una gota simple, para mantener una mejor estética y consistencia a lo largo de la aplicación.

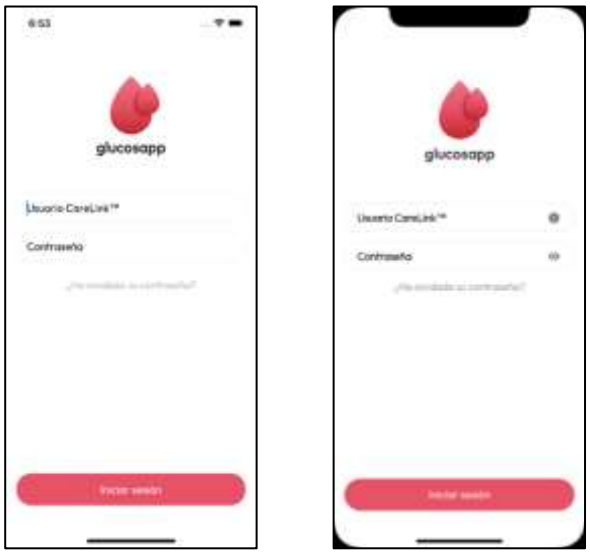

Figura 37. Comparativa de interfaces desarrolladas: inicio de sesión iPhone.

En la página de inicio de sesión mostrada en la Figura 37 se ve reflejada de manera casi exacta el diseño entre el prototipo y el desarrollo. Existiendo como diferencia únicamente la falta de los iconos a la derecha de los campos de texto, y la posición del botón de inicio de sesión.

Finalmente, se puede concluir que el desarrollo de interfaces, utilizando SwiftUI, ha logrado el objetivo de reproducir los prototipos planteados, con la intención de dar al usuario una interfaz amigable y usable.

4.7.5. Pruebas de usuarios

Dado que el software desarrollado es un software especializado, y requiere de una serie de equipos especiales para su correcto funcionamiento y validación. Se ha procedido a utilizar a un sujeto de prueba, el cual cumple con todos los requerimientos y posee los equipos necesarios para hacer uso de este software, con la intención de que este usuario utilice la

aplicación por un período de 1 semana, para proceder a dar un resultado, respondiendo a un cuestionario.

La encuesta realizada al usuario de prueba fue realizada utilizando la plataforma Google Forms. En esta encuesta se pregunta al usuario uno a uno si los objetivos, referentes al funcionamiento de la aplicación y muestra de datos, han sido cumplidos. Y en caso negativo, qué fallos ha presentado la aplicación. Así como posibles mejoras para cada uno de estos subapartados.

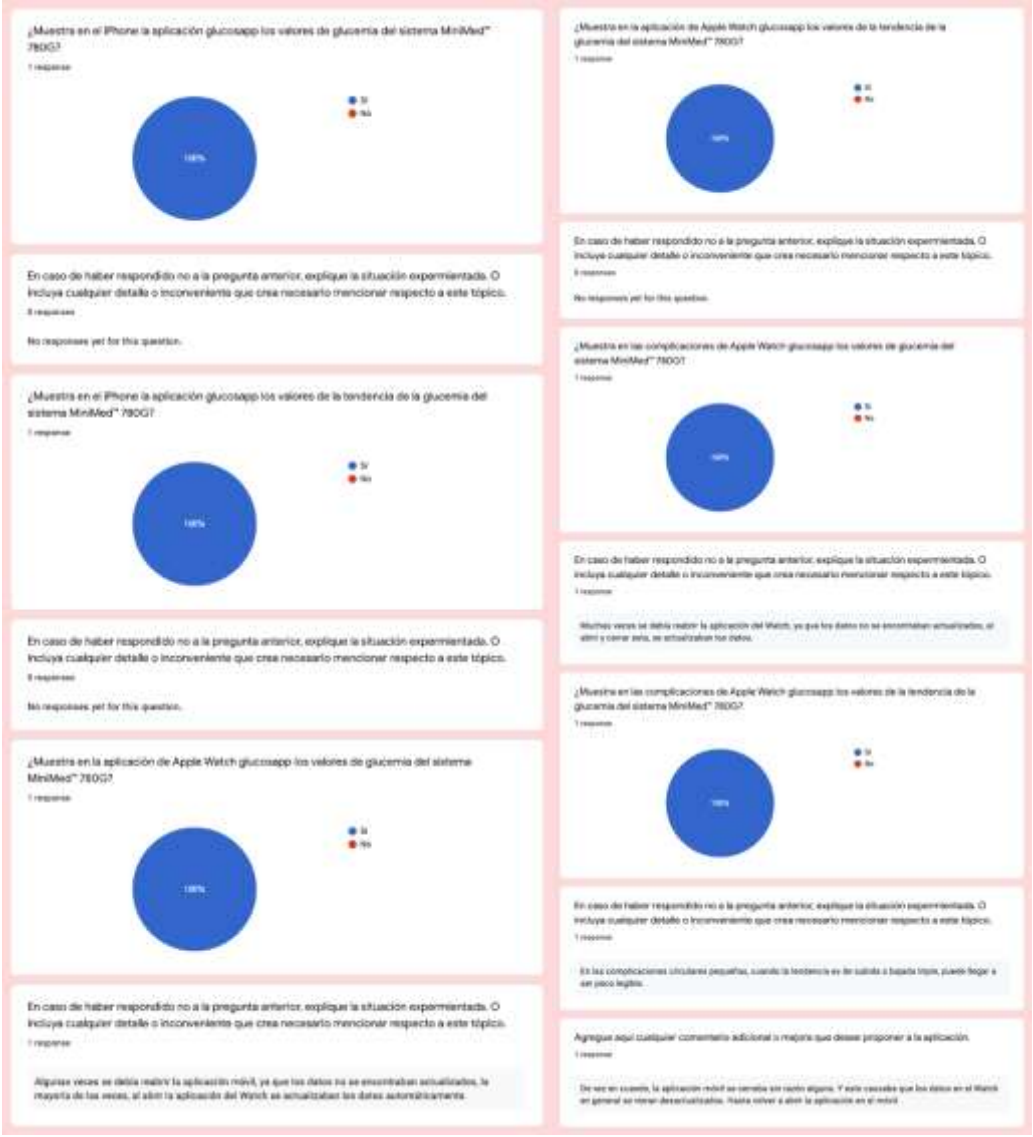

Figura 38. Extracto de la encuesta de las pruebas de usuario.

El resultado obtenido luego de la encuesta, el cual se presenta en la Figura 38, nos muestra que el resultado para cada una de las preguntas es afirmativo. Confirmado que la muestra de

datos es exitosa tanto en la aplicación móvil, como en la aplicación del Apple Watch y sus respectivas complicaciones. De igual manera, el sujeto de prueba ha comentado que repetidas veces se veía forzado a reabrir la aplicación, debido a que los datos se encontraban desactualizados, además de experimentar algunas veces que la aplicación móvil se cerrara de forma repentina.

Esto nos deja ver que el desarrollo no se encuentra del todo completo respecto a la usabilidad de los usuarios, pero en lo que atañe al proceso de software y a los objetivos planteados en este trabajo de grado, se pueden dar como completados los objetivos.

Si es cierto que el proceso de pruebas con usuarios no ha sido exhaustivo, para que este lo fuera, sería necesario un ambiente controlado con una población muy específica, la cual realice las pruebas utilizando distintos modelos de Apple Watch y distintos modelos de iPhone, tomando en cuenta además que todos deben ser usuarios del sistema de insulina MiniMed™ 780G. Lo cual excede en tamaños y tiempos, el alcance que puede abarcar este trabajo desarrollado.

4.8. Acceso al código fuente

Todo el software desarrollado en los apartados anteriores, incluyendo las pruebas de validación, han sido desarrollados manteniendo un control de versiones utilizando git. En particular, se ha utilizado la plataforma GitHub para versionar y guardar el código de manera externa en la nube.

Además, en esta plataforma se puede visualizar todo el proceso que se ha seguido a lo largo del desarrollo del proyecto, y se habilita la contribución de usuarios externos, en caso de ser necesaria alguna mejora o arreglo propuesta por estos.

El código se encuentra disponible en la URL: https://github.com/arnaldopxm/GlucosApp.

5. Conclusiones y trabajo futuro

Este apartado tiene como finalidad, la validación de los objetivos generales y específicos propuestos en el apartado 1.3 de este documento, y comprobar qué objetivos han sido alcanzados luego del desarrollo del software. Para esto, se presentará de manera resumida el desarrollo y los resultados obtenidos a lo largo del proyecto, además de los obstáculos y fallas encontradas a lo largo del proceso.

Para concluir, se plantearán posibles líneas de trabajo de desarrollo que se pueden desprender a partir del de esta aplicación, las cuales nacen desde posibles mejoras que se pueden presentar a la aplicación en cuestión, o incluso de limitaciones o inconvenientes encontrados en el proceso de desarrollo. Veamos a continuación por qué se considera de interés el trabajo desarrollado, y cuál es su aportación para la comunidad.

5.1. Conclusiones

A lo largo de este trabajo se ha diseñado, a nivel de arquitectura de software, desarrollado y validado una aplicación para dispositivos Apple Watch y iPhone. Dicha aplicación permite a cuidadores y pacientes de usuarios del sistema Medtronic MiniMed™ 780G, ver en sus dispositivos, Apple iPhone y Apple Watch, los datos de glucemia del paciente y su tendencia actual. Partiendo de esta afirmación, se puede concluir que el objetivo principal de trabajo ha sido cumplido. Veamos en detalle los matices respecto a las posibles excepciones o problemas encontrados en dicho proceso.

Se ha validado que la aplicación permite a los usuarios leer los datos a partir de la plataforma CareLink™ y los muestra en la comodidad de su muñeca, en un Apple Watch. Tanto en la aplicación como en las complicaciones, en las carátulas de los dispositivos Apple Watch.

En el desarrollo de las carátulas del Apple Watch, debido a la falta de herramientas, no ha sido posible validarlas mediante pruebas de interacción. Pero mediante interacciones manuales de usuarios, se ha validado que funcionan correctamente.

Dado que la fuente de los datos del Apple Watch es directamente el iPhone, se ha creado una dependencia directa entre el dispositivo móvil y el dispositivo de muñeca. Causando además que, si la aplicación es cerrada accidentalmente por el usuario, o si Apple en su manejo interno

de memoria decide terminar los procesos de fondo de esta, entonces el usuario quedaría con los datos desactualizados en su dispositivo Apple Watch; hasta que este vuelva a abrir la aplicación en el dispositivo móvil.

De igual manera, el paso de mensajes entre los dispositivos Apple iPhone y Apple Watch, se ve condicionado de gran manera por reglas internas, no siempre expresadas por Apple. Donde se establece un límite máximo de mensajes, o donde existen una variedad de tipos de mensajes, generando así un obstáculo a la hora de elegir de manera correcta la forma y el tipo de comunicaciones a utilizar para el paso de mensajes entre ambas partes.

Con todo lo anteriormente planteado, se corrobora que todos los objetivos planteados, correspondientes a las funcionalidades del software a desarrollar, han sido cumplidos. Obteniéndose como resultado un software funcional y verificable.

La aportación principal de este software a la comunidad es: proporcionar una nueva herramienta a los usuarios de los sistemas de insulina MiniMed™ 780G, y dispositivos Apple Watch, que les permita ver sus valores de glucemia en la comodidad de su muñeca mediante complicaciones, en las carátulas del Apple Watch.

Además de estos objetivos, se planteaban inicialmente objetivos referentes a las metodologías a seguir para el proceso del desarrollo de software, y la arquitectura de software a desarrollar. En este ámbito los objetivos han sido cumplidos satisfactoriamente, ya que el código ha sido desarrollado utilizando la arquitectura limpia de Martin. Y esta se ha hecho siguiendo una metodología ágil, en particular Kanban, a lo largo de todo el proceso de planificación y desarrollo del código.

Respecto a los objetivos restantes, se han adquirido conocimientos específicos en el uso del lenguaje Swift para el desarrollo de aplicaciones para dispositivos iPhone y Apple Watch, ya que se ha desarrollado de manera exitosa una aplicación para iPhone con su contraparte correspondiente para Apple Watch. De la misma manera, se debe destacar que se han obtenido los conocimientos necesarios en las herramientas utilizadas en el proceso validación de código mediante pruebas unitarias, de interfaces y de lanzamiento.

5.2. Trabajo futuro

Luego de haber presentado un breve resumen del desarrollo de software, para el cual todos los objetivos iniciales han sido cumplidos. Se procede a presentar en este apartado posibles mejoras que se pueden realizar sobre el software desarrollado. E incluso nuevas líneas de trabajo que pueden ser derivadas a partir de carencias encontradas a lo largo del desarrollo, las cuales se muestran a continuación.

- Soporte de otros sistemas de bomba de insulina, no directamente dependientes de Medtronic o MiniMed ™.
- Integración con el sistema Nightscout de código abierto.
- Adaptación para dispositivos Android y sus equivalentes relojes inteligentes.
- Nueva funcionalidad para la visualización de los valores de glucemia y tiempo en rango de las últimas 24 horas del paciente.
- Creación de un servidor, con la intención de desligar la dependencia entre el Apple Watch y el iPhone. Tomando en cuenta el guardado de credenciales, y el traslado de datos personales respecto a la ley de protección de datos.
- Integración con el Framework HealthKit de Apple, especializado en el manejo de datos relacionados con la salud y bienestar de los usuarios.
- Desarrollo de un Framework de pruebas para desarrollos realizados utilizando el Framework ClockKit.
- Validación de la funcionalidad y usabilidad del software en un entorno médico, mediante pruebas con distintos grupos de usuarios y dispositivos.

Referencias bibliográficas

Apple Inc. (2019). *Swift - Apple Developer*. Recuperado el 7 de febrero de 2022, de Apple.com: https://developer.apple.com/swift/

Apple Inc. (2022a). *ClockKit | Apple Developer Documentation*. Recuperado el 7 de febrero de 2022, de Apple.com: https://developer.apple.com/documentation/clockkit

Apple Inc. (2022b). *Keychain Services | Apple Developer Documentation*. Recuperado el 7 de febrero de 2022, de Apple.com: https://developer.apple.com/documentation/watchconnectivity/wcsession

Apple Inc. (2022c). *User Interface Tests | Apple Developer Documentation*. Recuperado el 9 de febrero de 2022, de Apple.com: https://developer.apple.com/documentation/xctest/user_interface_tests

Apple Inc. (2022d). *WatchKit | Apple Developer Documentation*. Recuperado el 7 de febrero de 2022, de Apple.com: https://developer.apple.com/documentation/watchkit

Apple Inc. (2022e). *WCSession | Apple Developer Documentation*. Recuperado el 7 de febrero de 2022, de Apple.com: https://developer.apple.com/documentation/watchconnectivity/wcsession

Apple Inc. (2022f). *Xcode - SwiftUI- Apple Developer*. Recuperado el 7 de febrero de 2022, de Apple.com: https://developer.apple.com/xcode/swiftui/

Apple Inc. (2022g). *Xcode | Apple Developer Documentation*. Recuperado el 7 de febrero de 2022, de Apple.com: https://developer.apple.com/documentation/xcode

Apple Inc. (2022h). *XCTest | Apple Developer Documentation*. Recuperado el 9 de febrero de 2022, de Apple.com: https://developer.apple.com/documentation/xctest

Cai, A. (1 de febrero de 2021). *Study Compares MiniMed 780G and MiniMed 670G Algorithms*. Recuperado el 20 de diciembre de 2021, de diaTribe: https://diatribe.org/study-compares-minimed-780g-and-minimed-670g-algorithms

Cámara Argentina de Especialidades Medicinales. (14 de noviembre de 2019). *La historia del descubrimiento de la diabetes y su control*. Recuperado el 12 de 2021, de CAEME:

https://www.caeme.org.ar/la-historia-del-descubrimiento-de-la-diabetes-y-su-control/

Campillo, S. (19 de enero de 2019). *Hiperglucemia e hipoglucemia: causas, síntomas y tratamiento*. Recuperado el 23 de diciembre de 2021, de Vitonica.com: https://www.vitonica.com/enfermedades/hiperglucemia-e-hipoglucemia-causas-sintomas-tratamiento#:~:text=Con%20esta%20palabra%20hacemos%20referencia,descienden%2C%20estaremos%20ante%20una%20hipoglucemia.

Capel Flores, I. (febrero de 2017). Recuperado el 20 de diciembre de 2021, de https://www.tdx.cat/bitstream/handle/10803/457142/icf1de1.pdf?sequence=1&isAllowed=y

Clinidiabet, S.L. (2018). *Comparativa de bombas de insulina 2010*. Recuperado el 9 de febrero de 2022, de Clinidiabet.com: https://clinidiabet.com/es/infodiabetes/bombas/46.htm

Diabetes Mall. (23 de marzo de 2020). *Tandem t:slim X2*. Recuperado el 8 de marzo de 2022, de Diabetesnet.com: https://www.diabetesnet.com/diabetes-technology/insulin-pumps/current-pumps/tandem-tslim-x2/

Diabetes Mall. (19 de octubre de 2021). *Comparison of Current Continuous Glucose Monitors (CGMs)*. Recuperado el 8 de marzo de 2022, de Diabetesnet.com: https://www.diabetesnet.com/diabetes-technology/meters-monitors/compare-current-monitors/

Féans, S. (4 de enero de 2021). *Mi experiencia con sistema integrado 780G*. Recuperado el 20 de diciembre de 2021, de RepúblikaDiabetes: https://republikadiabetes.com/mi-experiencia-780g/

FreeStyle Libre. (s.f.). *LibreLinkUp - Diabetes app | Freestyle Libre 2*. Recuperado el 8 de marzo de 2022, de Freestylelibre.co.uk: https://www.freestylelibre.co.uk/libre/products/mobile-app-librelinkup.html

Fundación para la Diabetes Novo Nordisk. (2020). *Fundaciondiabetes.org*. Recuperado el 20 de diciembre de 2021, de Tipos de diabetes: https://www.fundaciondiabetes.org/infantil/177/tipos-de-diabetes-ninos

González Fernández, M. (2019). *Revisión de la evolución de los sistemas de infusión subcutáneos de insulina en diabetes tipo 1.* Recuperado el 15 de diciembre de 2021, de Universidad Complutense de Madrid: https://eprints.ucm.es/id/eprint/61563/

Harrison, K. (28 de octubre de 2019). *Testing App Launch Time*. Recuperado el 6 de marzo de 2022, de Use Your Loaf: https://useyourloaf.com/blog/testing-app-launch-time/

International Diabetes Federation. (2021). IDF Diabetes Atlas, 10th edn. Bruselas, Bélgica. Recuperado el 15 de diciembre de 2021

jrendel. (28 de septiembre de 2020). *jrendel/SwiftKeychainWrapper: A simple wrapper for the iOS Keychain to allow you to use it in a similar fashion to User Defaults. Written in Swift.* Recuperado el 10 de diciembre de 2021, de GitHub: https://github.com/jrendel/SwiftKeychainWrapper

Kniberg, H., & Skarin, M. (2010). *Kanban and Scrum making the most of both.* Lulu.com.

Lewis, D., & The #OpenAPS Community. (2015). *OpenAPS Reference Design*. Recuperado el 10 de diciembre de 2021, de OpenAPS.org: https://openaps.org/reference-design/

Lewis, D., & The #OpenAPS Community. (2018). *What is #OpenAPS?* Recuperado el 10 de diciembre de 2021, de OpenAPS.org: https://openaps.org/what-is-openaps/

Lewis, D., & The #OpenAPS Community. (2021). *OpenAPS Outcomes*. Recuperado el 10 de diciembre de 2021, de OpenAPS.org: https://openaps.org/outcomes/

Martin, R. C. (2018). *Clean Architecture : a craftsman's guide to software structure and design.* Prentice Hall.

Mavin, A., Wilkinson, P., Harwood, A., & Novak, M. (agosto de 2009). Easy Approach to Requirements Syntax (EARS). *2009 17th IEEE International Requirements Engineering Conference*, 317-322.

Medtronic. (10 de agosto de 2017). *Terapia con Bomba de insulina*. Recuperado el 10 de diciembre de 2021, de Medtronic-diabetes.com: https://www.medtronic-diabetes.com/es/sobre-la-diabetes/terapia-con-bomba-de-insulina

Medtronic. (12 de marzo de 2018). *Bomba de insulina MiniMed™ 640G con SmartGuard™*. Recuperado el 10 de diciembre de 2021, de Medtronic España:

https://www.medtronic-diabetes.com/es/terapia-con-bomba-de-insulina/que-es-la-tecnologia-smartguard

Medtronic. (2018). *MiniMed670g Spain*. Recuperado el 7 de febrero de 2022, de mmc.medtronic-diabetes.com.

Medtronic. (30 de enero de 2019). *Minimed 670G System*. Recuperado el 10 de diciembre de 2021, de Medtronic España: https://www.medtronic-diabetes.com/es/terapia-con-bomba-de-insulina/sistema-minimed-670g

Medtronic. (2 de junio de 2020). *Minimed 780G System*. Recuperado el 10 de diciembre de 2021, de Medtronic España: https://www.medtronic-diabetes.com/es/sistema-minimed-780g

Medtronic. (7 de noviembre de 2021). *Minimed 780G System automated to help more patients reach glycaemic targets with less effort*. Recuperado el 7 de febrero de 2022, de UC202107103 EN_780G_HCP_Brochure UK v4: https://resources.cloud.medtronic-diabetes.com/sites/prd/files/documents/2021-11/UC202107103%20EN_780G_HCP_Brochure%20UK%20v4.pdf

Medtronic. (2021). *Monitorización continua de glucosa*. Recuperado el 10 de diciembre de 2021, de Medtronic-diabetes.com: https://guardianconnect.medtronic-diabetes.com/es/acerca-de-mcg

Medtronic. (4 de marzo de 2021). *Terapia con Bomba de Insulina*. Recuperado el 10 de diciembre de 2021, de Medtronic Diabetes: https://www.medtronicdiabetes.com/treatments/terapia-con-bomba-de-insulina

Nighscout Foundation. (2017). *What is Nightscout?* Recuperado el 20 de noviembre de 2021, de Nightscout: https://nightscout.github.io/

nightscout. (9 de noviembre de 2021). *nightscout/nightguard: iOS and WatchOS Client for the Nightscout CGM System*. Recuperado el 8 de marzo de 2022, de GitHub: https://github.com/nightscout/nightguard

Novalab. (2018). *Breve historia de la insulina: de su descubrimiento a la actualidad*. Recuperado el 15 de diciembre de 2021, de Making Diabetes Easier by Novalab - Haciendo la diabetes más fácil: https://www.makingdiabeteseasier.com/es/diabetes-

explicada/diabetes/breve-historia-de-la-insulina-de-su-descubrimiento-a-la-actualidad

Omer, T. (17 de agosto de 2016). Empowered citizen 'health hackers' who are not waiting. *BMC Medicine, 14*(1). doi:10.1186/s12916-016-0670-y

Organización Mundial de la Salud. (13 de abril de 2021). *Diabetes*. Recuperado el 10 de diciembre de 2021, de Organización Mundial de la Salud: https://www.who.int/es/news-room/fact-sheets/detail/diabetes

Rodriguez Herrero, A. (2010). *Propuesta de un algoritmo de control en lazo cerrado para la diabetes tipo 1*. Recuperado el 15 de diciembre de 2021, de http://oa.upm.es/3681/

Sáez de Fuente, J., Granja Berná, V., Valero Zanuy, M. A., Ferraqui Piquero, J., Herreros de Tejada, A., & Coterilla, L. (2008). Insulinoterapia en el medio hospitalario. *Nutrición Hospitalaria, 23*(2), 128-129.

Smith, R. (28 de mayo de 2020). *Dexcom*. Recuperado el 8 de marzo de 2022, de Can I View Dexcom G6 CGM data on an Apple Watch?: https://www.dexcom.com/faqs/can-i-view-dexcom-g6-cgm-data-on-apple-watch

Sommervile, I. (2005). *Ingeniería del Software.* Madrid: Pearson Education.

Suárez, L. (16 de marzo de 2017). *Guía sobre cómo utilizar el glucagón*. Recuperado el 23 de diciembre de 2021, de Guía Diabetes tipo 1: https://diabetes.sjdhospitalbarcelona.org/es/diabetes-tipo-1/consejos/guia-sobre-como-utilizar-glucagon#:~:text=El%20glucag%C3%B3n%20es%20un%20medicamento,con%20disolvente%20para%20soluci%C3%B3n%20inyectable.

Szász, B. (2021). *CareLinkJavaClient: Experimental Java library for retrieving data from Medtronic CareLink*. Recuperado el 23 de diciembre de 2021, de GitHub: https://github.com/benceszasz/CareLinkJavaClient

Tejedor, L. (24 de abril de 2018). *Nightscout, qué es y cómo montarlo*. Recuperado el 28 de noviembre de 2021, de Siendo célula beta: https://siendocelulabeta.com/2018/04/24/nightscout-que-es-y-como-montarlo/

The OpenAPS Foundation. (2 de diciembre de 2021). *OpenAPS Documentation Release 0.0.0*. Recuperado el 22 de diciembre de 2021, de Readthedocs.io: https://openaps.readthedocs.io/_/downloads/en/latest/pdf/

Thomas, A. (julio de 2009). Inference of Nanotechnology in modern Medicine - The Example of Diabetes Therapy. *Internet Electron. J. Nanoc. Moletrón., 7*(1), 1311-1322. Obtenido de https://www.researchgate.net/publication/266594807_Influence_of_nanotechnology_to_modern_medicine_-_as_an_example_of_diabetes_therapy

Young, R. R. (2004). *The requirements engineering handbook.* Boston: Artech House.

Índice de acrónimos

AHCL (Advanced Hybrid Closed Loop). Algoritmo avanzado de asa cerrada

ISCI. Infusión subcutánea continua de insulina.

MCG. Monitoreo continuo de glucosa.

JDRF (Juvenile Diabetes Research Foundation). Fundación de investigación de la diabetes juvenil.

APS. Sistema de Páncreas Artificial.

API (Application Programming Interface). Interfaz de programación de aplicaciones.

IDF (International Diabetes Foundation). Fundación internacional de la diabetes.

EARS (Easy Approach to Requirements Syntax). Plantilla de especificación de requisitos.

IDE (Integrated Development Environment). Entorno de desarrollo integrado.

WWDC (Worldwide Developers Conference) Conferencia Anual de tecnología hecha por Apple.