Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA
Mención en Ingeniería del Software

Blood Sugar Voice Assistant

Alumno:
Édgar Díez Alonso

Tutores:
Yania Crespo González-Carvajal (UVa)
Luis Vidal de la Rosa (HP-SCDS)

A mis padres, por estar siempre ahí cuando se les necesita

Agradecimientos

A mis padres, por ser un apoyo constante y siempre empujarme a aspirar a más.

A mis amigos de Pystacho, por continuar siendo mi segunda familia tras la carrera, aunque fuera de forma telemática en Discord.

A mi tutora Yania, por hacerme disfrutar de las asignaturas que impartía y dedicarme su apoyo y guía incondicionales durante la realización de este proyecto.

A mi tutor Jose Luis, por ayudarme con las dudas técnicas.

A Elena, Elías y Pablo por estar siempre ahí para echar una mano con las pruebas y demos.

Gracias a todos de corazón.

Resumen

Se puede observar cómo el número de diagnósticos de diabetes está aumentando considerablemente durante las últimas décadas. Esta enfermedad permite llevar un ritmo de vida normal siempre y cuando podamos controlar los niveles de glucemia de forma estricta.

El objetivo de este proyecto es desarrollar una skill para el asistente de voz Alexa, Blood Sugar Voice Assistant, que permita al usuario conocer su valor de glucemia de forma inmediata. Además, la aplicación debe gestionar un sistema de alertas que facilite al usuario realizar un seguimiento temporal de sus valores de glucemia.

El proyecto ha sido desarrollado en tres pilares, la skill de Alexa que permite usar comandos de voz para controlar los valores de glucosa en sangre, usando Python como lenguaje de programación, la API Rest que gestione las peticiones, programada en Golang y la página web de la aplicación que permita el registro de usuarios y manejo de los datos, desarrollada en Angular.

Abstract

It can be observed that the number of diabetes diagnoses has been increasing considerably over the last few decades. This disease lets us lead a normal life as long as we are able to control our glucose levels strictly.

The aim of this project is to develop a skill for the Alexa Voice Assistant, Blood Sugar Voice Assistant, which allows the user to know their blood glucose values immediately. In addition, the application must manage a system of alerts that lets the user monitor the blood glucose values over time.

The project has been developed on three pillars, the Alexa skill that uses voice commands to control blood glucose values, using Python as the programming language, the API Rest that manages the requests, programmed in Golang, and the web page of the application that allows user registration and data management, developed in Angular.

Índice general

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

1.1. Contexto

La diabetes mellitus ha sido definida como "la enfermedad del siglo XXI". Se puede observar como su prevalencia ha aumentado inexorablemente en las últimas décadas y se estima que va a afectar a un 10.2 % de la población mundial para 2030 [22].
Además, con la reciente aparición de la pandemia de Covid-19, numerosos estudios han determinado la posibilidad de que el virus de la Covid pueda estar relacionado con el desarrollo de diabetes mellitus, aunque aún hay mucha información que desconocemos al respecto [10].

Esta enfermedad permite llevar un estilo de vida normal, siempre que el paciente lleve un control muy estricto de sus niveles de glucemia en todo momento. Por ello, han surgido multitud de herramientas que facilitan la gestión del proceso de medición y registro de la glucemia de los pacientes, para que ellos mismos puedan controlar sus valores de forma autónoma. Por ejemplo, hay numerosas aplicaciones móviles que permiten estimar los niveles de azúcar consumidos en función de la dieta del usuario.

Sin embargo, el principal obstáculo para el paciente suele ser llevar a cabo el seguimiento de la glucemia con valores en tiempo real, ya que el procedimiento de medición es algo engorroso: el paciente debe llevar consigo en todo momento un glucómetro y pincharse en un dedo de forma periódica para obtener los valores de glucosa. Tras ello, el paciente debe registrar esos valores en un diario o similar.

Uno de los mayores avances en este campo ha sido la invención de los sistemas de medición flash continuos, como el sensor FreeStyle [1] que facilitan todo el proceso. El usuario se aplica un pequeño sensor en la parte trasera de su brazo, el cuál realiza mediciones de glucosa de forma periódica y se conecta por bluetooth a un dispositivo móvil. Esto permite al usuario olvidarse de realizar mediciones y registrar los valores obtenidos, ya que el sensor funciona de forma autónoma y envía alertas al usuario si los valores de glucemia se salen de los límites establecidos.

El proyecto **Blood Sugar Voice Assistant** surge con el objetivo de facilitar aún más el proceso, aprovechándose de la tecnología emergente de los asistentes de voz y permitiendo al usuario realizar un seguimiento de sus valores de glucemia mediante el uso de comandos de voz.

1.2. Motivación

Este Trabajo de Fin de Grado surgió, propuesto por HP SCDS, con la finalidad de facilitar la vida de los pacientes diagnosticados de diabetes. El empleo de sensores similares al de FreeStyle es cada vez más habitual, especialmente entre la población más joven que incorpora las nuevas tecnologías en su día a día.

Este tipo de usuario, suele disponer de dispositivos inteligentes o *wearables* que incluyen un asistente de voz. Por ejemplo, es habitual que los relojes inteligentes incorporen un asistente de voz tipo Alexa/Siri y los teléfonos inteligentes incorporan Google Assistant o Siri dependiendo de si son dispositivos Android o Apple.

Blood Sugar Voice Assistant pretende fusionar ambas tendencias, permitiendo el control de la información registrada por los sensores inteligentes, del tipo FreeStyle, mediante el empleo de comandos del asistente de voz de Amazon, **Alexa**.

1.3. Estudio de alternativas

Tras realizar un estudio de las aplicaciones móviles relacionadas con la gestión de la diabetes, se pueden clasificar en tres grandes categorías:

1. Apps en las que el usuario introduce su dieta y se realiza una estimación de los niveles de glucosa del usuario.
2. Apps en las que el usuario introduce sus valores de glucemia en un diario para realizar un seguimiento temporal.
3. Apps que usan asistentes de voz para facilitar la vida de los pacientes con diabetes.

Blood Sugar Voice Assistant se encuentra dentro de la tercera categoría, ya que emplea un asistente de voz para realizar un seguimiento de los valores de glucemia del usuario. En este bloque, la aplicación más representativa es **Sulli the Diabetes Guru** [13]. Esta aplicación funciona con Alexa y Google Assistant y responde a preguntas de carácter general sobre el seguimiento de la diabetes que pueda tener el usuario. Por ejemplo, ¿cuáles son valores elevados de azúcar?, ¿puedo hacer ejercicio antes de realizar un test de glucemia?, ...

Sin embargo, en ningún caso esta aplicación trabaja con datos de glucemia reales del usuario, por lo que Blood Sugar Voice Assistant es una aplicación única y original, que no tiene competencia en su nicho de mercado.

1.4. Nicho de mercado

El segmento de mercado que ocupa nuestra app es el de pacientes de diabetes que buscan un asistente de voz que les permita realizar un seguimiento de su enfermedad a través de un asistente de voz. Por tanto, el usuario objetivo de nuestra aplicación será de joven a mediana edad, entre 12-50 años, y será capaz de manejarse con soltura en el mundo de los dispositivos inteligentes y asistentes de voz.

1.5. Objetivos

El objetivo final que busca Blood Sugar Voice Assistant es la gestión del seguimiento en tiempo real y a lo largo del tiempo de los valores de glucemia de un usuario mediante comandos de voz del asistente de voz de Amazon, Alexa.

El usuario utilizará el comando de voz 'Alexa, dime mi nivel de glucemia' y Alexa le indicará:

1. Su nivel de glucosa en sangre.
2. Sugerencias para corregir su glucemia si el usuario tiene hipoglucemia o hiperglucemia. Por ejemplo ingerir azúcares o inyectarnos insulina, respectivamente.
3. Alexa programará alertas si el usuario tiene hipoglucemia o hiperglucemia para recordarle comprobar de nuevo su glucemia y reforzar el tratamiento si los valores no se han corregido. Las alertas pararán cuando se recupere un nivel de glucemia normal.

1.5.1. Objetivos personales

Además, al ser un Trabajo de Fin de Grado, este proyecto pretende lograr una serie de objetivos de formación personal:

1. Realizar un proyecto de desarrollo software de principio a fin, trabajando en todas las fases que abarcan el desarrollo de un proyecto en el ámbito profesional del desarrollo software
2. Realizar un proyecto de desarrollo software basándose en el framework propuesto por la metodología ágil Scrum.
3. Aprender a programar en el lenguaje de programación Go, cada vez más popular entre los lenguajes de programación orientados a backend.
4. Perfeccionar mis conocimientos de Python y Angular, muy relevantes actualmente.

1.6. Estructura de la memoria

El presente documento se estructura de la siguiente forma:

Capítulo 2. Requisitos y planificación: Describe el proceso realizado de preparación y planificación del proyecto, así como los requerimientos especificados para el mismo.

Capítulo 3. Tecnologías utilizadas: Se describen las herramientas y lenguajes de programación utilizados en la realización del proyecto.

Capítulo 4. Análisis: Se presentan los requisitos de información del proyecto, el diagrama de modelo de dominio inicial, la máquina de estados generada para gestionar los niveles de glucemia y el proceso de negocio de la aplicación.

Capítulo 5. Diseño: Se entra en detalle en la arquitectura de la plataforma, patrones de diseño aplicados, y el diseño de los distintos componentes de la plataforma: aplicación web, API Rest, Base de Datos y Skill. También se explica cómo se desarrollan skills para Alexa, el flujo de ejecución de la aplicación y el despliegue de los componentes.

Capítulo 6. Implementación y pruebas: En este capítulo se trata la estructura de directorios del repositorio y el contenido de cada uno de ellos, así como las pruebas realizadas, dificultades encontradas en el desarrollo y la licencia del proyecto.

Capítulo 7. Seguimiento del proyecto: Se explica cómo ha sido el desarrollo de cada uno de los Sprints del proyecto, y el feedback obtenido en cada una de las reuniones Sprint Review/Retrospective/Planning.

Capítulo 8. Conclusiones: Por último se entra en detalle sobre el cumplimiento de los requisitos especificados para el proyecto y posibles líneas de trabajo futuras.

Anexo A Manuales: Incluye manuales de instalación, despliegue, y de uso.

Anexo B Resumen de enlaces adicionales: Incluye el enlace al repositorio del proyecto con el código fuente y la documentación, y donde se realizó el seguimiento del proyecto con GitLab Issue Board.

Capítulo 2

Requisitos y Planificación

2.1. Scrum

Para la realización del proyecto Blood Sugar Voice Assistant se ha decidido seguir los planteamientos de la metodología ágil **Scrum**. Scrum [15] es un framework que ayuda a las personas, equipos de trabajo y empresas a generar valor a través de soluciones adaptativas a problemas complejos. La finalidad principal de Scrum es simplificar el proceso de desarrollo y, para ello, proporciona unas pautas que buscan guiar a sus usuarios a alcanzar metas y generar valor.

Scrum utiliza un enfoque iterativo e incremental para optimizar la predictabilidad y controlar riesgos. Para ello, se basa en tres pilares fundamentales:

- **Transparencia.** El trabajo debe ser visible tanto para los que lo realizan como los que lo reciben. Con Scrum, las decisiones a tomar se basan en el estado de sus tres **artefactos**. Los artefactos que tienen baja transparencia pueden llevar a los usuarios a tomar decisiones que disminuyen el valor del proyecto y aumentan los riesgos.
- **Inspección.** El progreso hacia los objetivos establecidos y los artefactos de Scrum deben ser inspeccionados a menudo y a fondo para poder detectar problemas indeseados. Para garantizar este pilar, Scrum establece cinco **eventos**.
- **Adaptación.** Este pilar establece que el proceso de desarrollo debe ser ajustado si cualquier aspecto de un proyecto se desvía de los límites establecidos o si el producto entregable no es aceptable. Este ajuste debe ser realizado lo antes posible para minimizar desviaciones futuras.

Estos pilares derivan en que Scrum sea el framework de desarrollo que más se adecúa a los proyectos con alta incertidumbre y que puedan sufrir cambios constantes [14]. Por ello, es perfecto para un proyecto innovador como Blood Sugar Voice Assistant, donde los requisitos

Figura 2.1: El framework Scrum. Imagen tomada de [25]

no están claramente establecidos, y puede haber muchos cambios durante el proceso de desarrollo.

2.1.1. Roles

La unidad fundamental de Scrum es el **equipo Scrum**, formado por un número reducido de personas [15]. Este equipo de trabajo se puede dividir en tres categorías: Scrum Master, Product Owner y Desarrolladores, como se puede ver en la Figura 2.2, centrados en un objetivo en cada momento temporal para generar un Product Goal, u objetivo del Product Backlog.

Los equipos Scrum deben ser **multifuncionales**, todos los miembros deben tener las skills necesarias para generar valor en cada sprint, y **auto-organizados**, los miembros del equipo deciden internamente quién, cómo y qué se hace en cada momento.

- **Scrum Master.** Es el responsable de garantizar que el proyecto siga los pilares establecidos por Scrum. Para ello instruye a los demás miembros del equipo para que implementen las prácticas impuestas por el framework Scrum.
- **Product Owner.** Su función es maximizar el valor del producto generado por el equipo de trabajo Scrum. Para ello establece el Product Goal y crea el Product Backlog. Además, representa las necesidades de los stakeholders en el proyecto.
- **Desarrolladores.** Son los miembros del equipo de trabajo Scrum cuya finalidad es desarrollar el producto generando un Incremento cada Sprint. Para ello crean un plan cada Sprint, el denominado Sprint Backlog.

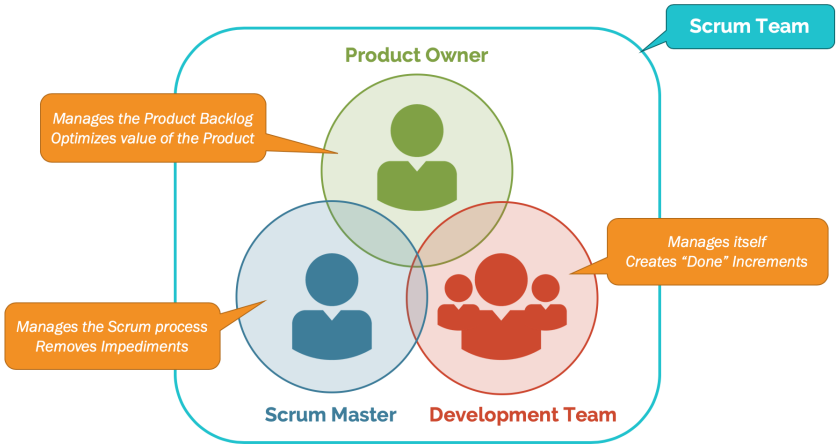


Figura 2.2: Roles de Scrum. Imagen tomada de [9]

2.1.2. Eventos

Scrum establece cinco eventos que permiten inspeccionar y adaptar los artefactos de Scrum. Estos eventos están diseñados especificamente para garantizar la Transparencia, crear regularidad y minimizar la necesidad de reuniones no planificadas. Idealmente, los eventos deben ser periódicos y tener una duración establecida.

Estos cinco eventos (Figura 4.1) son:

- **Sprint.** Son la pieza clave de Scrum, donde las ideas se convierten en valor. Los Sprints tiene duración fija, menor a un mes, y se suceden continuadamente en el tiempo hasta que finaliza el proyecto. Los requisitos de un Sprint son inmutables mientras dure el mismo, pero se pueden tomar decisiones que modifiquen el Sprint Backlog. Cada Sprint puede ser considerado como un projecto breve y la finalidad es obtener un Incremento, parte de producto producida en un Sprint que se caracteriza por estar terminada y operativa y que puede ser entregada al cliente [24]
- **Sprint Planning.** Reunión inicial que da el comienzo a cada Sprint y en donde se establece qué trabajo se va a llevar a cabo. Deben partipar todos los miembros del equipo Scrum y en ella se responden a las siguientes preguntas: ¿cuál es el valor del Sprint?, ¿qué se va a hacer este Sprint? y ¿cómo se va a hacer?.
- **Daily Scrum.** El objetivo principal es analizar el progreso realizado hacia el Sprint Goal y modificar el Product Backlog como se considere oportuno. Son reuniones de 15 minutos que tienen lugar a la misma hora cada día de trabajo en un Sprint y a las que acuden los desarrolladores. Ayudan a mejorar la comunicación, identificar impedimentos y proponer soluciones alternativas.
- **Sprint Review.** Su finalidad es analizar los resultados de un Sprint y reconocer si hace falta realizar modificaciones. El equipo Scrum presenta su trabajo a los stakeholders,

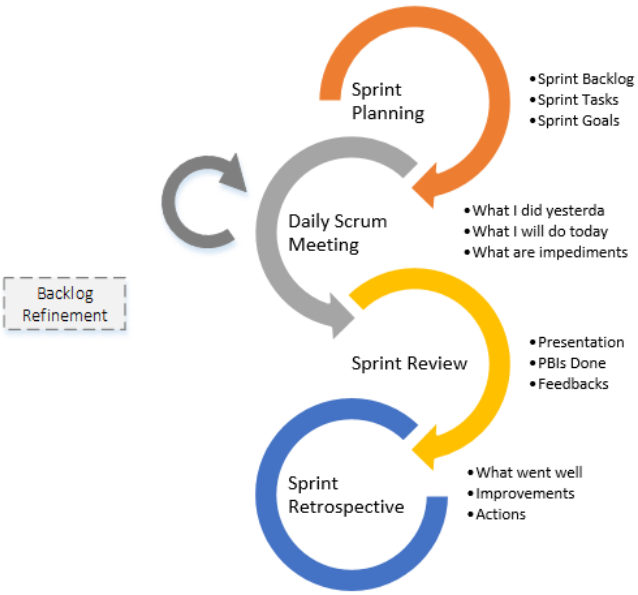


Figura 2.3: Eventos de Scrum. Imagen tomada de [19]

se discute sobre el progreso realizado hacia el Sprint Goal y se decide qué se va a hacer en el siguiente Sprint. Este evento puede durar un máximo de cuatro horas para un Sprint de un mes.

- **Sprint Retrospective.** El equipo Scrum realiza una autoevaluación entre la Sprint Review y el Sprint Planning e identifican qué cambios se pueden realizar para incrementar la efectividad de Sprints futuros. Esta reunión concluye el Sprint.

2.1.3. Artefactos

Los Artefactos de Scrum son generados a partir de los eventos o actividades de Scrum. Representan trabajo o valor y su finalidad principal es garantizar el pilar de la Transparencia de información. Son tres:

- **Product Backlog.** Es una lista ordenada de características y requisitos del producto, representadas en forma de historias de usuario. Su responsable es el Product Owner y a menudo se va refinando a menudo que se suceden los Sprints. Su objetivo es el *Product Goal*, objetivo del proyecto que debe ser realizado (o abandonado) antes de saltar al siguiente Product Goal.
- **Sprint Backlog.** Es un plan por y para los desarrolladores. Contiene el trabajo que los desarrolladores van a desarrollar en el Sprint actual, es decir, aquellas historias de usuario del Product Backlog que se van a realizar en un Sprint. Su finalidad es

alcanzar el *Sprint Goal*, u objetivo único del Sprint. El Sprint Goal se genera en el Sprint Planning y los desarrolladores lo deben tener en cuenta mientras dure el Sprint.

- **Incremento.** Es el producto que se ha generado a lo largo de un Sprint . Cada Incremento debe ser aditivo a los Incrementos de Sprints anteriores y varios pueden ser generados a lo largo de un único Sprint. Por tanto, su responsable es el equipo de desarrollo. Para poder ser empleado, debe cumplir la *Definición de Hecho*, descripción formal del estado del Incremento cuando cumple las características de calidad requeridas para el producto.

2.2. Adaptación de Scrum al proyecto Blood Sugar Voice Assistant

Como se ha explicado, Scrum ha sido el marco de desarrollo empleado para realizar Blood Sugar Voice Assistant. El principal motivo de escoger Scrum por encima de otras metodologías ágiles es la flexibilidad que demanda el proyecto, ya ue aunque desde HP SCDS se han establecido algunos requisitos del proyecto, la mayor parte de las decisiones de desarrollo se han dejado en manos del desarrollador.

Sin embargo, para poder aplicar el framework Scrum en Blood Sugar Voice Assistant se ha tenido que realizar una adaptación, al ser el proyecto un Trabajo de Fin de Grado y no un proyecto del ámbito laboral.

En primer lugar, se han adaptado los roles del Equipo Scrum. Los tutores, tanto por parte de HP SCDS como de la Universidad de Valladolid asumen el rol de *Scrum Master*, asegurándose de que el estudiante sigue las prácticas propuestas por Scrum y ayudándole en caso de dudas o bloqueos. Por otra parte, el estudiante ha asumido el rol de *Product Owner* y *Equipo de Desarrollo*, por lo que se encargará de generar las historias de usuario y de su desarrollo.

La adaptación de los eventos de Scrum se ha realizado de forma que los eventos *Sprint Planning*, *Sprint Review* y *Sprint Retrospective* tienen lugar el martes cada 3 semanas, al finalizar un Sprint y antes de comenzar el siguiente. En esa reunión se analizará el avance realizado en el Sprint, se hará una retrospectiva analizando si se ha cumplido el trabajo planificado y por último se planificará el trabajo a realizar en el siguiente Sprint. Por lo tanto, los Sprints han sido establecidos de una duración de 3 semanas. La *Daily Scrum* la realizará el estudiante antes de ponerse a trabajar, ya que es el Equipo de Desarrollo, y en ella se hará un autoanálisis de las tareas a realizar en el día.

Por último, todos los artefactos de Scrum, *Product Backlog*, *Sprint Backlog* y los *Incrementos* serán responsabilidad del estudiante, ya que asume los roles de Product Owner y Equipo de Desarrollo simultáneamente.

En la Sección 2.5 se explican las horas dedicadas a la semana que se usarán para estimar las horas de cada tarea del product backlog.

2.3. Análisis de Riesgos

Un riesgo [20] es un evento o condición incierta que, si sucede, tiene un efecto negativo en por lo menos uno de los objetivos del proyecto. Todos los riesgos tienen una (o varias) causa(s) y una (o varias) consecuencia(s) y, por tanto, además de intentar analizar qué riesgos se pueden producir en el desarrollo del proyecto debemos establecer un plan de mitigación (objetivo: minimizar la probabilidad de que se produzcan las causas que generan el riesgo) y otro de contingencia (objetivo: minimizar el impacto de las consecuencias cuando el riesgo se ha producido).

Para definir cada riesgo, tenemos:

- **Título.** Breve identificación del riesgo.
- **Descripción.** Breve explicación del riesgo.
- **Probabilidad.** Baja, media o alta. Probabilidad de que se produzca el riesgo.
- **Impacto.** Bajo, medio o alto. Nivel de importancia que tienen las consecuencias del riesgo en el proyecto si se produce el riesgo.
- **Plan de mitigación.** Medidas a llevar a cabo para reducir las probabilidades de que el riesgo se produzca.
- **Plan de contingencia.** Medidas a llevar a cabo para minimizar el impacto de las consecuencias si el riesgo se ha producido.

A continuación se presenta el análisis de riesgos. Se han dividido los riesgos en *Riesgos Generales*, aquellos que se pueden dar en cualquier proyecto de desarrollo de software (Tabla 2.1 - Tabla 2.5), y en *Riesgos Específicos*, aquellos particulares a la naturaleza de Blood Sugar Voice Assistant (Tabla 2.6 - Tabla 2.9).

2.4. Planificación de Presupuestos

Otro de los puntos claves de la planificación de un proyecto es el plan de presupuestos. El proyecto Blood Sugar Voice Assistant es un Trabajo de Fin de Grado y no está orientado al sector laboral, por lo que como tal no se le asigna un presupuesto 'real'.

Por ello, se ha decidido simular la presupuestación, generando un **presupuesto simulado**.

2.4.1. Presupuesto Simulado

Para la simulación del presupuesto se ha tenido en cuenta de que el proyecto va a realizarse durante 6 meses, empleando 15 horas semanales, por lo que en total simularemos prepuesto

Título	Estimación de tiempo y trabajo optimista
Descripción	Planificación temporal poco realista para una actividad o tarea que produce el incumplimiento de plazos durante un Sprint.
Probabilidad	Media
Impacto	Bajo
Plan de mitigación	▪ Establecer dos sprints de refuerzo al final del proyecto que proporcionen margen temporal al proyecto. ▪ Utilizar la información obtenida en los Sprints Retrospective para planificar mejor los Sprints sucesivos.
Plan de contingencia	Mover las tareas incompletas de un Sprint al siguiente y emplear el margen temporal proporcionado por los Sprints de refuerzo para finalizar el proyecto.

Tabla 2.1: Riesgo R01. Estimación de tiempo y trabajo optimista

Título	Enfermedad de algún miembro del Equipo Scrum
Descripción	El estudiante o los tutores contraen una enfermedad y no pueden trabajar en el proyecto temporalmente.
Probabilidad	Alta
Impacto	Media
Plan de mitigación	▪ Mantener higiene personal, lavado frecuente de manos. ▪ Establecer sprints de refuerzo para proporcionar margen temporal al proyecto.
Plan de contingencia	▪ Si la ausencia por enfermedad es breve, incrementar el número de horas empleadas en el proyecto los días sucesivos para recuperar el tiempo perdido. ▪ Si la ausencia tiene mayor duración, mover las tareas incompletas de un Sprint al siguiente y emplear el margen temporal proporcionado por los Sprints de refuerzo para finalizar el proyecto.

Tabla 2.2: Riesgo R02. Enfermedad de algún miembro del Equipo Scrum

Título	Cambios en los requisitos del proyecto
Descripción	Durante la realización del proyecto se decide realizar cambios en la funcionalidad de la aplicación, tanto por parte del equipo Scrum como del cliente.
Probabilidad	Alta
Impacto	Bajo
Plan de mitigación	▪ Tener una reunión de planificación del proyecto con el cliente donde realizar elicitación de requisitos, para garantizar un conocimiento completo del proyecto propuesto. ▪ Tener reuniones cada Sprint con los stakeholders para comprobar que los requisitos del cliente están siendo garantizados.
Plan de contingencia	Añadir las nuevas tareas al Product Backlog, analizar prioridades y planificar para realizar en el Sprint actual si las modificaciones son menores o en Sprints sucesivos si son cambios considerables.

Tabla 2.3: Riesgo R03. Cambios en los requisitos del proyecto

Título	Desconocimiento de las tecnologías a emplear en el proyecto
Descripción	Durante la realización del proyecto se trabaja con tecnologías o herramientas desconocidas por el equipo de desarrollo.
Probabilidad	Alta
Impacto	Medio
Plan de mitigación	Realizar un Sprint 0 de preparación, donde el estudiante aprenda a usar las tecnologías que desconozca y vayan a ser empleadas para la realización del proyecto.
Plan de contingencia	▪ Aprender a usar las herramientas desconocidas que haya que emplear durante un Sprint mediante documentación o videotutoriales. ▪ Solicitar ayuda a los tutores.

Tabla 2.4: Riesgo R04. Desconocimiento de las tecnologías a emplear en el proyecto

Título	Fallo o rotura en el equipo de trabajo del estudiante
Descripción	Durante la realización del proyecto se producen fallos técnicos temporales o definitivos en el equipo de trabajo del estudiante.
Probabilidad	Baja
Impacto	Alto
Plan de mitigación	▪ Reservar el ordenador personal únicamente para la realización del proyecto y emplear el ordenador de la empresa para todos los asuntos relacionados con el ámbito laboral. ▪ Trabajar en plataformas online, tipo Google Docs, Overleaf, ... y hostear el código en la plataformas de integración continua Gitlab.
Plan de contingencia	Reparar el ordenador o comprar uno nuevo si el dispositivo es irreparable.

Tabla 2.5: Riesgo R05. Fallo o rotura en el equipo de trabajo del estudiante

Título	Caídas o desconexiones a Internet
Descripción	Se producen problemas en el internet de la residencia del desarrollador (Eduroam) que detienen el desarrollo del proyecto.
Probabilidad	Media
Impacto	Alto
Plan de mitigación	Realizar un uso adecuado de los datos móviles para no agotar la cuota mensual.
Plan de contingencia	Emplear datos móviles para conectarse a Internet.

Tabla 2.6: Riesgo R06. Caídas o desconexiones a Internet

Título	Falta de dedicación de tiempo por motivos académicos
Descripción	El estudiante dedica menor tiempo del necesario a la realización del proyecto por motivos académicos. Principalmente debido a la realización de prácticas Erasmus + y eventos relacionados con la empresa.
Probabilidad	Media
Impacto	Bajo
Plan de mitigación	▪ Defender el proyecto en Septiembre, alargando así el proyecto temporalmente, lo que permite finalizarlo dedicando una menor cantidad de horas / semana. ▪ En cada Sprint, preguntar en la empresa qué eventos se avecinan en un futuro cercano. Comprobar periódicamente el calendario online de la empresa para ver si se añaden eventos nuevos.
Plan de contingencia	▪ Trasladar las tareas que no se hayan podido realizar al siguiente Sprint. ▪ Emplear los dos Sprints de refuerzo para finalizar las tareas incompletas al final del proyecto.

Tabla 2.7: Riesgo R07. Falta de dedicación de tiempo por motivos académicos

Título	Caída temportal de la máquina virtual de la UVa ofrecida al estudiante
Descripción	Se rompe la conexión a la máquina virtual de la UVa donde se aloja el servidor de la aplicación, por lo que se impide su ejecución.
Probabilidad	Baja
Impacto	Media
Plan de mitigación	Evitar realizar muchas peticiones en un corto periodo de tiempo para que no se añada la aplicación a una whitelist.
Plan de contingencia	Ponerse en contacto con los técnicos de la UVa para que reestablezcan la máquina virtual.

Tabla 2.8: Riesgo R08. Caída temportal de la máquina virtual de la UVa ofrecida al estudiante

Título	Problemas de conectividad entre estudiante y tutores.
Descripción	Se producen problemas que impiden la conexión telemática entre el estudiante y los tutores, lo que detiene el avance del proyecto.
Probabilidad	Baja
Impacto	Alto
Plan de mitigación	Establecer plataformas de comunicación alternativas.
Plan de contingencia	Comunicarse con una de las plataformas de comunicación telemáticas alternativas establecidas.

Tabla 2.9: Riesgo R09. Problemas de conectividad entre estudiante y tutores

para 390 horas si tenemos en cuenta que en medio año hay 26 semanas y no vamos a realizar vacaciones durante la realización del proyecto.

En primer lugar, tenemos en cuenta el salario del trabajador. No se han podido encontrar estimaciones de salarios específicos para desarrolladores de Skills Alexa. Sin embargo, si tenemos en cuenta de que para realizar una skill de Alexa hay que realizar tanto backend como la skill en sí, las habilidades que se buscan en el desarrollador son similares a las de un desarrollador de backend. El salario medio para un desarrollador de Backend en Francia según el portal *https://www.glassdoor.fr* es de 49.340€ brutos anuales. A mayores, la empresa debe cubrir un 36.1 % en la Seguridad Social del trabajador [11], lo que suma un total de 67151.74 € anuales para la empresa. En Francia hay un total de 252 días laborables al año de media, lo que corresponde a 50.4 semanas trabajadas si tenemos en cuenta que los sábados y domingos no se trabaja. Por último, en Francia se trabajan 35 horas a la semana, por lo que en un año se trabaja en total 1764 horas. A partir de todas estas cifras podemos calcular que la empresa de desarrollo de Blood Sugar Voice Assistant emplearía **14846.47 €** en el desarrollador.

Como equipo de trabajo, el desarrollador ha escogido su ordenador personal, un *MSI GS65 Stealth* valorado en 1750€, al cual se le asigna una vida útil de 6 años (72 meses). Por tanto el dispositivo tiene un coste de 24.31 €. El proyecto tiene una duración de 6 meses, por lo que el coste amortizado del dispositivo es de **145.86 €**. A mayores, en el proyecto se emplea una máquina virtual de la Universidad de Valladolid. El coste estimado para 6 meses de uso de una máquina de dichas características (2 cores, 2Gb de memoria RAM) [18] es de **196.56 €**.

Si analizamos costes de alojamiento y recursos (electricidad y agua) se ha estimado [16] que los costes medios de un espacio de coworking con Internet, agua y electricidad ascienden a 268 € mensuales, por lo que si contratamos 7 meses de alquiler el coste ascendería a 1608 €.

Por último, si tenemos en cuenta las licencias de software a emplear en el proyecto, son todas gratuitas excepto *Astah Professional* [3], que conlleva un gasto de **57.54 €** para los 6 meses de uso.

Si sumamos todos los costes explicados anteriormente, el coste total simulado de Blood Sugar Voice Assistant asciende a **16854.43 €**. Si dejamos un 10 % de margen para posibles contingencias el coste final simulado del proyecto es de **18539.87 €**. En la Tabla 2.10 se puede ver un resumen de los costes.

Título	**Coste estimado**
Salario del trabajador (incluye Seguridad Social)	14846.47€
Equipo del trabajador	145.86€
Espacio coworking	1608.00€
Máquina Virtual Linux	196.56€
Licencias de Software	57.54€
Total	16854.43€
Total (con margen de presupuestación del 10 %)	18539.87€

Tabla 2.10: Planificación de Presupuestos simulada

2.5. Planificación inicial de los Sprints

Como se explicó en la Sección 2.2, al adaptar el marco Scrum a Blood Sugar Voice Assistant se decidió establecer una duración de Sprint de 3 semanas.

La situación del estudiante es particular, ya que realiza el Trabajo de Fin de Grado al mismo tiempo que realiza Prácticas Erasmus+ en Francia con una duración de 6 meses, del 15 de Febrero al 15 de Agosto. Al no poder defender el TFG sin haber finalizado dichas prácticas, el estudiante se encuentra con que debe realizar la defensa en Septiembre, por lo que se decide iniciar el primer Sprint el día 1 de Febrero y finalizar el último Sprint el día 30 de Agosto, lo que permite realizar un total de **10 Sprints**.

A mayores se ha decidido incluir un **Sprint 0** de preparación, donde el estudiante se familiarice con las tecnologías a emplear en el proyecto. En nuestro caso: Golang y Alexa Skill Development Kit. Este Sprint 0 ha tenido lugar antes de comenzar el proyecto, en concreto del 1 de Enero hasta el 1 de Febrero. Los dos últimos Sprints se han reservado como **Sprints de refuerzo**, que serán empleados en caso de que el estudiante necesite un margen temporal para finalizar el proyecto.

Como el estudiante ha decidido emplear 15 horas semanales al Trabajo de Fin de Grado, en total va a dedicar **450 horas** al proyecto, sin contar las horas dedicadas al Sprint 0 de preparación.

En la Tabla 2.11 se puede observar la planificación resultante para cada Sprint. El evento designado como *Sprint Meeting* incluye los eventos Sprint Review, Retrospective y Planning.

2.6. Product Backlog Inicial

El Product Backlog ha sido generado por el Product Owner, en este caso el estudiante, como se ha explicado en la Sección 2.2. Para ello el estudiante se ha basado en el documento de HP SCDS donde se introducen las especificaciones necesarias para realizar la plataforma Blood Sugar Voice Assistant, añadiendo aportaciones propias y/o sugerencias de los tutores.

El Product Backlog se divide en **Épicas** [23], actividades demasiado extensas y que se

Sprint	**Fecha**	**Apuntes**
Sprint 0	01-01-22 - 01-02-22	Sprint de preparación
Sprint Meeting	01-02-22	
Sprint 1	01-02-22 - 22-02-22	
Sprint Meeting	22-02-22	
Sprint 2	22-02-22 - 15-03-22	
Sprint Meeting	15-03-22	
Sprint 3	15-03-22 - 05-04-22	
Sprint Meeting	05-04-22	
Sprint 4	05-04-22 - 26-04-22	
Sprint Meeting	26-04-22	
Sprint 5	26-04-22 - 17-05-22	
Sprint Meeting	17-05-22	
Sprint 6	17-05-22 -07-06-22	
Sprint Meeting	07-06-22	
Sprint 7	07-06-22 - 28-06-22	
Sprint Meeting	28-06-22	
Sprint 8	28-06-22 - 19-07-22	
Sprint Meeting	19-07-22	
Sprint 9	19-07-22 - 09-08-22	Sprint de refuerzo
Sprint Meeting	09-08-22	
Sprint 10	09-08-22 - 30-08-22	Sprint de refuerzo
Sprint Meeting	30-08-22	

Tabla 2.11: Planificación inicial de los Sprints

deben de disgregar y separar en otras más pequeñas, con valor propio y que se realicen dentro de un sprint, las **Historias de Usuario**.

Estos requisitos son enunciados siguiendo el esquema 'como *stakeholder* quiero *objetivo* para *valor obtenido*' en las Tablas 2.12 a 2.15.

Número	**Épica**
1	Como usuario quiero poder registrarme y hacer login en la aplicación Blood Sugar Voice Assistant para gestionar mi cuenta de usuario.
2	Como usuario registrado quiero poder conocer mi valor de glucemia con la skill de Alexa Blood Sugar Voice Assistant para poder realizar un control de mis valores de azúcar.
3	Como usuario registrado quiero poder acceder a todos mis datos registrados en la aplicación Blood Sugar Voice Assistant para poder realizar un control temporal de mi glucemia.

Tabla 2.12: Épicas del Product Backlog inicial

Número	**Historia de Usuario**
1	Como usuario sin registrar quiero poder crear una nueva cuenta en la aplicación Blood Sugar Voice Assistant para poder usar la plataforma.
2	Como usuario registrado quiero poder iniciar sesión con mis credenciales de usuario en la aplicación Blood Sugar Voice Assistant para ver mis valores de glucosa en los últimos siete días.
3	Como usuario registrado quiero poder recuperar mis credenciales de usuario en la aplicación Blood Sugar Voice Assistant para recuperar acceso a la aplicación.

Tabla 2.13: Historias de Usuario de la Épica 1 del Product Backlog inicial

2.7. Product Backlog Final

Durante la realización del proyecto, y siguiendo los principios planteados por Scrum, se han modificado algunas Historias de Usuario para adaptarse a los requisitos cambiantes de los stakeoholders. Estas modificaciones han supuesto el cambio, eliminación o edición de las Historias de Usuario ya existentes y el Product Backlog final se puede ver en las Tablas 2.16 - 2.19.

Por ejemplo se ha modificado la **US-15** del Product Backlog inicial para mostrar solo los valores de glucemia de los últimos siete días. Se ha eliminado la **US-3** del Product Backlog inicial por falta de tiempo, ya que el sistema de recuperación de credenciales no es algo prioritario para el producto. Por último se ha añadido la **US-12** del Product Backlog final para garantizar que la aplicación pueda ser llamada desde distintos dispositivos con soporte Alexa, como smartwatches o Amazon Echos.

Número	Historia de Usuario
4	Como usuario registrado quiero poder solicitar mi último valor de glucemia a la skill Alexa Blood Sugar Voice Assisntant para corregir mi glucemia si se encuentra en valores inadecuados
5	Como usuario registrado quiero que se registre una alerta de 15 minutos en Blood Sugar Voice Assistant si estoy en hipoglucemia y me recomiende ingerir hidratos de carbono de absorción rápida para poder corregir mi hipoglucemia.
6	Como usuario registrado quiero que en la alerta de hipoglucemia se programe una nueva alerta de hipoglucemia tras 15 minutos si sigo en estado de hipoglucemia para corregir mi hipoglucemia
7	Como usuario registrado quiero que la alerta de hipoglucemia me recomiende de nuevo ingerir hidratos de carbono de absorción rápida si el valor de glucemia es inferior al de la anterior medida solicitada para corregir mi hipoglucemia
8	Como usuario registrado quiero que la alerta de hipoglucemia me recomiende ingerir hidratos de carbono de absorción lenta si se ha corregido la hipoglucemia para mantener valores normales de glucemia
9	Como usuario registrado quiero que se registre una alerta de 30 minutos en Blood Sugar Voice Assistant si estoy en estado de hiperglucemia y me recomiende administrarme insulina para corregir mi hiperglucemia
10	Como usuario registrado quiero que tras la alerta de hiperglucemia se programe una nueva alerta de 30 minutos si sigo en estado de hiperglucemia y los niveles de azúcar han subido con respecto a la medida anterior para corregir mi hiperglucemia
11	Como usuario registrado quiero que tras la alerta de hiperglucemia se programe una nueva alerta de 1 hora si sigo en estado de hiperglucemia pero los niveles de azúcar han disminuido con respecto a la medida anterior para corregir mi hiperglucemia
12	Como usuario registrado quiero que las alertas se detengan cuando alcance un valor de glucemia normal (70-180) para no ser molestado por la aplicación si mis valores de azúcar son los adecuados
13	Como usuario registrado quiero que Blood Sugar Voice Assistant me recomiende vigilar mi glucemia si estoy entre 120-180 ya que me encuentro cerca de valores de hiperglucemia para que sea cuidadoso si me encuentro en el margen de hiperglucemia
14	Como usuario registrado quiero solicitar el valor de glucemia y si estoy en hipo/hiperglucemia no se anulen las alertas ya programadas y no se programen nuevas alertas para que no haya solapamiento de alertas

Tabla 2.14: Historias de Usuario de la Épica 2 del Product Backlog inicial

Número	Historia de Usuario
15	Como usuario registrado quiero acceder a todos los valores de glucemia almacenados en Blood Sugar Voice Assistant para realizar un seguimiento temporal de mi glucemia
16	Como usuario registrado quiero poder descargarme todos mis valores de glucemia en formato PDF para poder compartir mis resultados con personas externas como mi médico
17	Como usuario registrado quiero poder eliminar todos los datos de glucemia almacenados en Blood Sugar Voice Assistant para poder garantizar mi privacidad
18	Como usuario registrado quiero poder eliminar mi cuenta asociada a la aplicación Blood Sugar Voice Assistant para poder garantizar mi privacidad
19	Como usuario registrado quiero poder subir mis valores de glucosa a Blood Sugar Voice Assistant para poder realizar un seguimiento temporal de mi glucemia

Tabla 2.15: Historias de Usuario de la Épica 3 del Product Backlog inicial

Número	Épica
1	Como usuario quiero poder acceder a la aplicación Blood Sugar Voice Assistant para poder usar la plataforma.
2	Como usuario registrado quiero poder conocer mi valor de glucemia más reciente y obtener recomendaciones con la skill de Alexa Blood Sugar Voice Assistant para poder realizar un control de mis valores de azúcar.
3	Como usuario registrado quiero poder acceder a todos mis datos registrados en la aplicación Blood Sugar Voice Assistant para poder realizar un control temporal de mi glucemia y gestionar mi privacidad.

Tabla 2.16: Épicas del Product Backlog final

Número	Historia de Usuario
1	Como usuario sin registrar quiero poder crear una nueva cuenta en la aplicación Blood Sugar Voice Assistant para poder usar la aplicación.
2	Como usuario registrado quiero poder iniciar sesión con mis credenciales de usuario en la aplicación Blood Sugar Voice Assistant para ver mis valores de glucosa en los últimos siete días.

Tabla 2.17: Historias de Usuario de la Épica 1 del Product Backlog final

Número	Historia de Usuario
3	Como usuario registrado quiero poder solicitar mi último valor de glucemia a la skill Alexa Blood Sugar Voice Assisntant para corregir mi glucemia si se encuentra en valores inadecuados
4	Como usuario registrado quiero que se registre una alerta de 15 minutos en Blood Sugar Voice Assistant si estoy en hipoglucemia y me recomiende ingerir hidratos de carbono de absorción rápida para poder corregir mi hipoglucemia.
5	Como usuario registrado quiero que en la alerta de hipoglucemia se programe una nueva alerta de hipoglucemia tras 15 minutos si sigo en estado de hipoglucemia para corregir mi hipoglucemia
6	Como usuario registrado quiero que la alerta de hipoglucemia me recomiende de nuevo ingerir hidratos de carbono de absorción rápida si el valor de glucemia es inferior al de la anterior medida solicitada para corregir mi hipoglucemia
7	Como usuario registrado quiero que la alerta de hipoglucemia me recomiende ingerir hidratos de carbono de absorción lenta si se ha corregido la hipoglucemia para mantener valores normales de glucemia
8	Como usuario registrado quiero que se registre una alerta de 30 minutos en Blood Sugar Voice Assistant si estoy en estado de hiperglucemia y me recomiende administrarme insulina para corregir mi hiperglucemia
9	Como usuario registrado quiero que tras la alerta de hiperglucemia se programe una nueva alerta de 30 minutos si sigo en estado de hiperglucemia y los niveles de azúcar han subido con respecto a la medida anterior para corregir mi hiperglucemia
10	Como usuario registrado quiero que tras la alerta de hiperglucemia se programe una nueva alerta de 1 hora si sigo en estado de hiperglucemia pero los niveles de azúcar han disminuido con respecto a la medida anterior para corregir mi hiperglucemia
11	Como usuario registrado quiero que Blood Sugar Voice Assistant me recomiende vigilar mi glucemia si estoy entre 120-180 ya que me encuentro cerca de valores de hiperglucemia para que sea cuidadoso si me encuentro en el margen de hiperglucemia
12	Como usuario registrado quiero poder consultar mi último valor de glucosa desde cualquier dispositivo con soporte de Alexa para poder usar la aplicación en distintos dispositivos como los Echo o wearables como un smartwatch con salida tanto por voz como con texto.

Tabla 2.18: Historias de Usuario de la Épica 2 del Product Backlog final

Número	Historia de Usuario
13	Como usuario registrado quiero acceder a los valores de glucemia de los últimos 7 días almacenados en Blood Sugar Voice Assistant para realizar un seguimiento temporal de mi glucemia
14	Como usuario registrado quiero poder descargarme mis valores de glucemia en los últimos 7 días en formato PDF para poder compartir mis resultados con personas externas como mi médico
15	Como usuario registrado quiero poder eliminar todos los datos de glucemia almacenados en Blood Sugar Voice Assistant para poder garantizar mi privacidad
16	Como usuario registrado quiero poder eliminar mi cuenta asociada a la aplicación Blood Sugar Voice Assistant para poder garantizar mi privacidad
17	Como usuario registrado quiero poder subir mis valores de glucosa a Blood Sugar Voice Assistant para poder realizar un seguimiento temporal de mi glucemia
18	Como usuario registrado quiero poder cerrar mi sesión en la aplicación Blood Sugar Voice Assistant para garantizar la seguridad de mis datos

Tabla 2.19: Historias de Usuario de la Épica 3 del Product Backlog final

Capítulo 3

Tecnologías utilizadas

En este Capítulo se procede a desarrollar en profundidad las tecnologías utilizadas para la realización de Blood Sugar Voice Assistant. Para ello, se han organizado en tres bloques: tecnologías de organización, programación y comunicación.

Algunas de las tecnologías que se van a desarrollar han sido calificadas como *de uso obligatorio* en los requisitos del proyecto establecidos por HP SCDS o por la tutora de la Universidad de Valladolid; por ejemplo, el uso de GitLab como repositorio de gestión del proyecto. Otras se han dejado a *libre elección* del estudiante; son ejemplos el uso de Visual Studio Code como IDE y los lenguajes de programación elegidos para realizar el proyecto.

3.1. Tecnologías de organización

Como tecnologías de organización se incluyen aquellas que facilitan la gestión del proyecto y la realización de la memoria.

3.1.1. GitLab

GitLab [27] es un repositorio de código OpenSource basado en Git. A mayores proporciona una plataforma colaborativa de desarrollo de software que facilita las prácticas de *DevOps*.

En este proyecto GitLab posee 4 funciones: repositorio de almacenamiento del código, sistema de control de versiones, herramienta de gestión de proyecto (*GitLab Issue Board*) y herramienta de documentación del proyecto (*Gitlab Wiki*). A mayores se valoró la posibilidad de utilizar el sistema de integración contínua (*GitLab CI/CD*) pero debido a la naturaleza del proyecto, al ser una Skill de Alexa, no se consideró necesario.

Figura 3.1: Logo de GitLab. Imagen tomada de [29]

Git

Git [12] es un sistema de control de versiones Open Source que permite realizar el seguimiento de cambios en el código de un proyecto, lo que permite que muchos desarrolladores puedan trabajar simultáneamente en un proyecto.

Git basa su funcionamiento en los conceptos de repositorios y ramas. Un **repositorio** es un sistema de almacenamiento de directorios y archivos que guarda toda la historia de cambios realizados en forma de metadatos. Una **rama** es una línea de desarrollo independiente y se representa como un puntero a un estado del repositorio en un punto concreto del tiempo. Cuando un desarrollador desea añadir funcionalidad o corregir un bug, debe generar una nueva rama para encapsular los cambios. Cuando esta rama contenga la funcionalidad deseada y haya sido testeada, se realizará una operación de **merge** que añadirá esa funcionalidad a la rama de desarrollo principal, o *main*. Otras operaciones de Git son **add**, añade cambios al 'staging area', **commit**, guarda una instantánea de los cambios del repositorio local, **push**, sube los cambios commiteados del repositorio local al remoto, y **pull**, recupera cambios del repositorio remoto al local.

En Blood Sugar Voice Assistant se ha usado un flujo de trabajo *legacy* denominado **Gitflow** [5]. El proyecto contiene 3 tipos de ramas de desarrollo:

1. **Main:** es la rama principal del proyecto. También denominada *master*, contiene la evolución de las releases oficiales de un proyecto a lo largo del tiempo.

2. **Develop:** rama de integración de las features de un proyecto. También denominada *dev*, cuando esta rama contiene suficientes cambios para justificar una nueva versión del proyecto, y está suficientemente testeada para poder asegurar que es estable, se mergea a la rama main.

3. **Rama de 'Feature':** son ramas creadas a partir de dev, en las que se implementará una única nueva funcionalidad, denominada *feature*. Para este proyecto se ha seguido la plantilla *SPx-USy-Título*, donde 'SP' hace referencia al Sprint en el que se pretende implementar esa funcionalidad originalmente y 'US' a la historia de usuario que hace

referencia. Además, cada rama de feature incluye una pequeña descripción de la funcionalidad que se va a desarrollar junto a su *Acceptance Criteria*, que son las condiciones que la feature debe garantizar.

GitLab Issue Board

El tablero de Issues de GitLab es una herramienta de DevOps que permite planificar, organizar y visualizar el flujo de trabajo de un proyecto. Puede ser usada como un tablero KanBan o de Scrum.

Esta herramienta consiste en un tablón en el que hay columnas de tarjetas asociadas a una *label* o etiqueta. Cada tarjeta contiene una **Issue**, o tarea a realizar. En nuestro proyecto las columnas son:

1. **Open:** Contiene todas las tareas del Product Backlog que se van a implementar a lo largo del proyecto.
2. **Planned:** Contiene aquellas tareas que van a ser implementadas en el Sprint actual. Es un equivalente del Sprint Backlog.
3. **Doing:** Contiene las tareas que están siendo implementadas en un momento concreto del tiempo.
4. **Blocked:** Contiene las tareas que están siendo bloqueadas por otra tarea del Product Backlog. Cuando se finalize la tarea bloqueante se reanudará la tarea bloquada.
5. **Closed:** Contiene las tareas que ya han sido implementadas y revisadas.

Además, se han creado las labels **skill**, **front** y **server**, para ayudar a distinguir si las issues hacen referencia a la Skill Alexa, a la aplicación Frond-End, o al servidor Back-End que forman parte de Blood Sugar Voice Assistant.

Por último, de forma análoga a las ramas, para el nombre de las issues se ha seguido la plantilla *[SPx-USy] Título*, donde 'SP' hace referencia al Sprint en el que se pretende implementar dicha issue y 'US' a la historia de usuario que hace referencia.

Wiki

Por último, desde HP SCDS se ha solicitado utilizar la Wiki de GitLab para almacenar la documentación relativa al proyecto que pueda ser relevante a los usuarios de la aplicación Blood Sugar Voice Assistant. La Wiki de GitLab es un repositorio de Git independiente al repositorio del proyecto, que permite guardar la documentación junto al código del proyecto.

La Wiki de este proyecto contiene información relativa al análisis y diseño del mismo, junto a las instrucciones de compilación y despliegue de la aplicación.

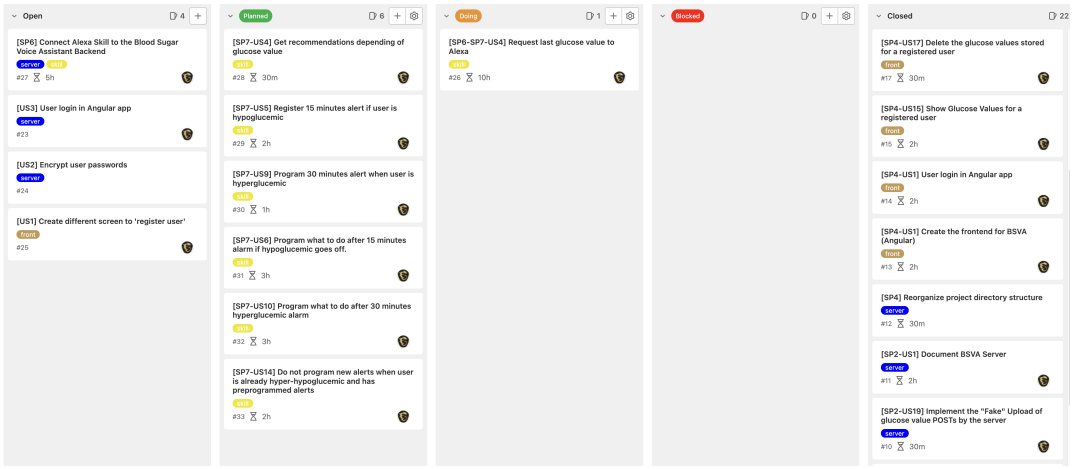


Figura 3.2: Estado del GitLab Issue Board durante el Sprint 7 del proyecto

3.1.2. Overleaf

Figura 3.3: Logo de Overleaf. Imagen tomada de [21]

Overleaf [33] es un editor de LaTex colaborativo en la nube, usado principalmente para escribir y editar publicaciones científicas. Se ha empleado para la realización de la memoria del Trabajo de Fin de Grado ya que permite almacenar las diferentes versiones del documento según se ha ido realizando. Además, este editor facilita el acceso simultáneo del documento por el estudiante y la tutora de la Universidad de Valladolid.

3.1.3. Astah Professional

Astah [4], es una herramienta de modelado de software que permite visualizar los distintos diagramas de análisis y diseño que se emplean en la realización de un proyecto de desarrollo de software. En este proyecto se ha empleado la versión *Astah Professional 8.5*, cuya licencia de pago es suministrada de forma gratuita a los estudiantes de la Universidad de Valladolid.

Figura 3.4: Logo de Astah. Imagen tomada de [4]

3.2. Tecnologías de Comunicación

Se entienden como tecnologías de comunicación aquellas que proporcionan formas de contacto telemático entre el estudiante y los tutores, tanto de forma escrita como mediante videollamadas.

3.2.1. Gmail

Como gestor de correo electrónico se ha escogido Gmail, ya que permite una fácil integración con el Calendario de Google y es uno de los gestores de correo electrónico más utilizados en todo el mundo. Este medio de comunicación se ha establecido para mantener un canal abierto entre el estudiante y el tutor de HP SCDS, donde poder responder dudas técnicas de forma urgente, que bloqueen el proyecto si se dejan sin respuesta hasta la siguiente reunión. Además, a través de Gmail se han compartido los enlaces de acceso a las reuniones de cada Sprint Review.

Figura 3.5: Logo de Gmail. Imagen tomada de [31]

3.2.2. Telegram

Telegram ha sido la herramienta de comunicación escogida entre el estudiante y la tutora de la Universidad de Valladolid. Ha sido utilizada para responder dudas de carácter administrativo, la memoria del proyecto y temas relativos a la universidad, por ejemplo, la administración de las máquinas virtuales de la universidad.

Figura 3.6: Logo de Telegram. Imagen tomada de [36]

3.2.3. Zoom

Zoom es una aplicación que permite realizar videollamadas en tiempo real, perfecta para reuniones de trabajo, webinars u otras reuniones colaborativas. Es una herramienta multiplataforma, ya que tiene versiones para todos los grandes sistemas operativos. Entre sus funciones destacan la posibilidad de compartir tu pantalla, un chat grupal y poder grabar la reunión de forma sencilla. En este proyecto se ha usado Zoom para realizar las Sprint Reviews. El tutor de HP SCDS, encargado de moderar dichas reuniones, posteriormente comparte la grabación de la reunión con los demás integrantes del proyecto.

Figura 3.7: Logo de Zoom. Imagen tomada de [38]

3.2.4. Google Meet

Google Meet es una aplicación de videollamadas, similar a Zoom. En este proyecto se ha empleado como opción de contingencia en caso de que la plataforma de Zoom no funcionara y para realizar reuniones repentinas, sin planificación previa, entre el estudiante y la tutora de la Universidad de Valladolid.

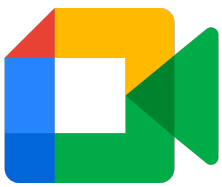


Figura 3.8: Logo de Google Meet. Imagen tomada de [30]

3.3. Tecnologías de Programación

Por último, se explican las tecnologías empleadas para realizar la programación del proyecto Blood Sugar Voice Assistant. Para ello se comentará a continuación qué lenguajes de programación se han utilizado en el proyecto, Bases de Datos, IDEs y suites de herramientas de terceros.

3.3.1. GoLang

Go es un lenguaje de programación desarrollado por los ingenieros informáticos de Google en 2007, y publicado en 2012. También se conoce como *GoLang* (the Google Language). Surgió para suplir las carencias que los desarrolladores encontraban en los lenguajes de programación, y por ello se buscaba que fuera un lenguaje tipado estáticamente, tuviera una gran eficiencia y rapidez, fuera fácilmente entendible y permitiera concurrencia. Al cumplir con todos estos objetivos se consiguió un lenguaje de programación expresivo, conciso, limpio y eficiente.

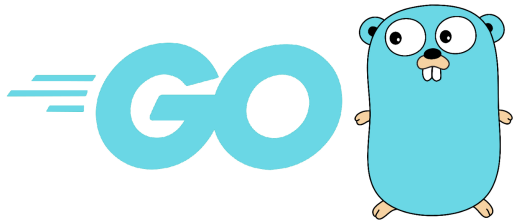


Figura 3.9: Logo de GoLang y su mascota. Imagen tomada de [17]

Go es popular en servidores y aplicaciones en la nube, herramientas de DevOps y utilidades de línea de comandos. En el proyecto Blood Sugar Voice Assistant se ha utilizado Golang para programar la lógica de negocio, es decir, la API REST que gestiona el backend de nuestro proyecto, respondiendo a las peticiones http que lancemos a nuestra aplicación.

Dentro de las librerías de terceros que se han usado para la realización del proyecto podemos destacar:

1. **Gorm:** Es una librería que facilita ORM (Object-Relational mapping) en Go. Para ello permite realizar conexiones con bases de datos de forma sencilla, realizar queries y realizar la conversión a Struct de GoLang (en Golang no existen las Clases).

2. **Gorilla/mux:** Este paquete implementa un Router de peticiones, que redirije las peticiones entrantes a su respectivo Handler.

3.3.2. Python

Python [34] es un lenguaje de programación de alto nivel, interpretado y de propósito general. Su filosofía destaca la legibilidad del código a través del indentado. Es un lenguaje multiparadigma, ya que se puede emplear para programación orientada a objetos, programación funcional o programación imperativa. Además es un lenguaje dinámico y multiplataforma. Python 3 es la última revisión principal del lenguaje y fue publicada en 2008. Es unos

Figura 3.10: Logo de Python. Imagen tomada de [34]

de los lenguajes de programación más utilizados ya que se puede emplear para desarrollar páginas web, tanto backend como frontend, scripts, testing, análisis y visualización de datos, etc. Además, se le considera un lenguaje sencillo de aprender.

En Blood Sugar Voice Assistant, utilizamos Python para crear la lógica de la Skill de Alexa, ya que el estudiante está muy familiarizado con el lenguaje y se considera que usar un lenguaje de programación conocido le ayudará a la hora de usar por primera vez el framework de Amazon para desarrollar skills.

3.3.3. Angular

Angular es un framework de desarrollo de aplicaciones web, basado en *Typescript*. Está mantenido por Google y su propósito principal es desarrollar Aplicaciones de Página Única (*single-page appplications*) [35]. Éstas son aplicaciones o páginas web que interaccionan con el usuario recargando el contenido dinámicamente con datos del backend, en vez del método por defecto, que consiste en recargar la página entera. Con ello se busca garantizar unas transiciones más rápidas, que hagan sentir que la página web funcione como una app nativa.

En Angular, los **componentes** son los pilares principales de la aplicación, por lo que es importante conocer su ciclo de vida. Tenemos 8 etapas en el ciclo de vida de los componentes y cada una de ellas se denomina *evento de enlace de ciclo de vida*. En total comprenden:

1. **Constructor:** No es un evento per-se. Como un componente es una clase de TypeScript debemos tener un método constructor. Aquí se deben inyectar las dependencias del componente.

Figura 3.11: Logo de Angular. Imagen tomada de [2]

2. **ngOnChanges:** Se llama cuando cambiar un valor de un *input control* dentro de un componente.
3. **ngOnInit:** Se ejecuta cuando se han desplegado las variables vinculadas a datos o un componente se inicializa.
4. **ngDoCheck:** Se llama cuando se verifica una condición lógica.
5. **ngAfterContentInit:** Se activa después de ngDoCheck para inicializar commponentes hijos.
6. **ngAfterViewInit:** Se llama cuando la vista del componente se ha inicializado totalmente.
7. **ngAfterViewChecked:** Se activa a continuación de ngAfterViewInit cuando la vista del componente verifique cambios.
8. **ngOnDestroy:** Se ejecuta antes de que Angular destruya un componente. Sirve para evitar fugas de memoria.

Para programar en Angular hay que conocer tres tecnologías: TypeScript, HTML y CSS.

TypeScript

TypeScript es un superset de JavaScript, por lo que contiene toda la funcionalidad de JavaScript y añade nuevas funciones. Por tanto, cualquier programa escrito en JavaScript puede ser ejecutado correctamente como TypeScript, ya que TypeScript al ser compilado se transforma en JavaScript.

TypeScript tiene *tipado estático*, se asigna una clase a las variables en la declaración, que se comprueba en tiempo de compilación. Si hay un error de tipos se muestra un mensaje de error pero el código se ejecuta igualmente. A mayores, TypeScript implementa *Interfaces*, que permiten asignar clases a los atributos de los objetos, y *Clases*, permitiendo crear nuevas clases o usar herencia.

HTML

HTML es un lenguaje de marcado de hipertexto, o *HyperText Markup Language*. Es el lenguaje estándar para mostrar documentos en los navegadores web. Los navegadores reciben documentos HTML de un servidor web y convierten el código HTML en páginas web multimedia. HTML describe la estructura de la página y puede ser complementado con CSS para añadir estilos.

CSS

CSS, *Cascading Style Sheets*, es el lenguaje que describe cómo los elementos HTML se muestran en el navegador web. Nos ayuda a resolver el problema original de HTML, ya que HTML nunca fue diseñado con la intención de contener etiquetas de estilo, sino que su función principal es describir el contenido. Por tanto CSS nos permite separar los documentos que describen el estilo de las páginas web en ficheros externos al HTML.

3.3.4. MySQL

MySQL [32], es un sistema de gestión de bases de datos relacionales, basado en SQL, *Structured Query Language*, con licencia open-source. Una base de datos relacional almacena datos en tablas, que tienen relaciones entre ellas que ayudan a estructurar los datos. Para crear, modificar, añadir permisos y obtener datos de la base de datos se usa el lenguaje SQL.

Figura 3.12: Logo de MySQL. Imagen tomada de [32]

Para representar las relaciones entre tablas se usan **claves**:

1. **Clave primaria:** Columna o conjunto de columnas que identifica de forma única a una fila de una tabla. No puede haber dos filas en una tabla con la misma clave primaria.

2. **Clave foránea:** Identifica una columna o conjunto de columnas de una tabla que hacen referencia a una columna o conjunto de columnas que son clave primaria en otra tabla. De esta manera se establecen las relaciones entre tablas en Bases de Datos relacionales.

Para gestionar la base de datos, se ha usado una extensión de Visual Studio Code llamada *MySQL* como se comenta en 3.3.5.

3.3.5. Visual Studio Code

Visual Studio Code [37], o vscode, es un editor de texto open-source creado por Microsoft. Es un editor multiplataforma, ya que tiene versiones para los tres grandes sistemas operativos. Además de ser un editor de texto, incluye funcionalidades como soporte para debugear, resaltado de sintaxis, refactorización de código, autocompletar de código, plugin integrado de Git, etc. A mayores, Visual Studio Code posee infinidad de extensiones que permiten añadir funcionalidad adicional, como soporte para otros lenguajes de programación, suites de testeo o gestión de bases de datos, entre otras.

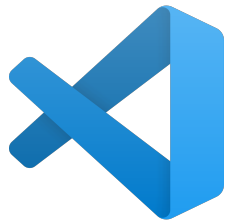


Figura 3.13: Logo de Visual Studio Code. Imagen tomada de [37]

El estudiante ha elegido Visual Studio Code, en vez de una IDE estándar, como editor de código ya este es un proyecto multilenguaje, con base de datos y con el framework de Amazon para desarrollo de skills de Alexa. Gracias a las extensiones de vscode, podemos programar en Go, Python, TypeScript, SQL, tener acceso a la plataforma Alexa Developer Console y sincronizar todo con GitLab sin cambiar de editor de texto.

A continuación se comentan las extensiones más útiles para la realización del proyecto.

Alexa Skill Toolkit

Esta extensión de vscode permite crear, testear y desplegar extensiones de Alexa desde el editor de texto. Para ello genera un espacio de trabajo similar a 3.3.6 dentro de Visual Studio Code. Para poder utilizar la extensión se necesita una cuenta de desarrollador de Amazon y Git.

Remote SSH

Esta extensión permite abrir un directorio remoto, máquina virtual o contenedor a través de una conexión SSH, por lo que obtenemos todas las ventajas de vscode a pesar de estar trabajando con una máquina remota. En este proyecto se ha empleado esta extensión para

desarrollar el servidor backend en GoLang, directamente en la máquina virtual proporcionada por la Universidad de Valladolid.

El código fuente se compila y ejecuta en la máquina remota y la extensión permite redirigir puertos a la máquina local para facilitar el proceso de testing y debug.

MySQL

Por último, la extensión MySQL permite tener un sistema de gestión de bases de datos dentro del editor de texto. Facilita crear conexiones a bases de datos, ejecutar código SQL y visualizar de forma sencilla la estructura y los datos de servidores SQL, bases de datos, tablas o consultas.

3.3.6. Alexa Developer Console

La consola de desarrollador de Alexa [7] es una plataforma de Amazon que permite crear, mantener y publicar skills para el asistente de voz Alexa. Para ello organiza el proceso en las categorías:

1. **Build:** 3.15 Engloba crear la skill, configurar el modelo de interacciones y establecer los endpoints del servicio.
2. **Code:** 3.16 Permite programar la aplicación directamente desde la consola. Usa el servicio Amazon Lambda y se conecta a él directamente. Tal y cómo hemos explicado en la Sección 3.3.2, la Skill de Alexa ha sido programada usando el lenguaje de programación Python.
3. **Test:** 3.17 y 3.18 Probar la skill con voz o texto, tanto en la fase de desarrollo como cuando la skill está desplegada.
4. **Distribution:** Comprobar de forma anticipada cómo aparecerá la skill en la tienda de skills de Alexa.
5. **Certification:** Permite validar la skill para poder distribuirla en la tienda de skills de Alexa.
6. **Analytics:** Proporciona métricas de la skill como número de descargas, intents invocados, etc.

Las skills de Alexa tienen tanto *modelo de interacción* como *lógica de negocio*. El usuario habla y Alexa procesa el mensaje en el contexto del modelo de interacción para determinar qué desea el usuario. Se envía la petición a la lógica de la aplicación, se ejecuta un *intent* y se envía la respuesta al usuario. La lógica de la aplicación puede ser un servicio proveído por Alexa, Amazon AWS o tu propio servidor web.

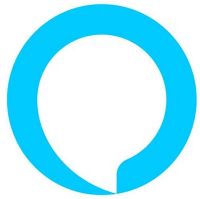


Figura 3.14: Logo de Amazon Alexa. Imagen tomada de [28]

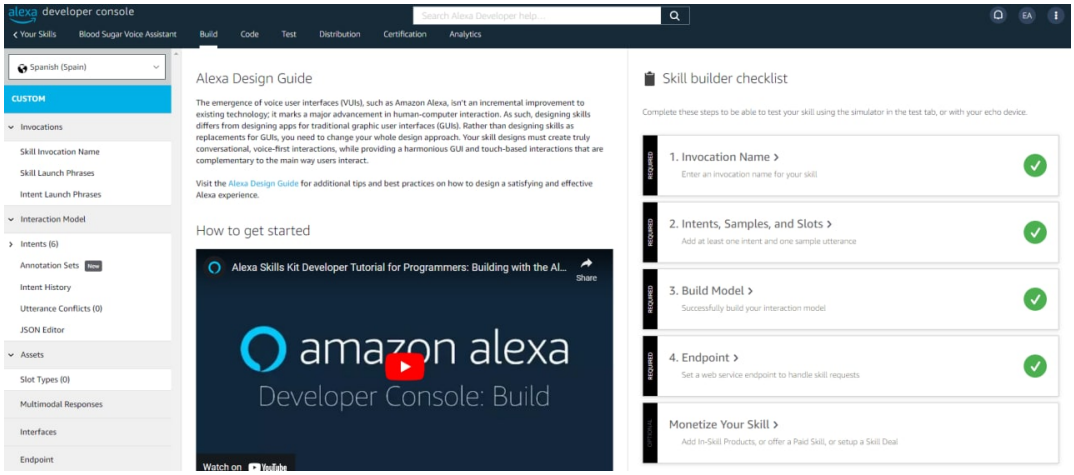


Figura 3.15: Pestaña de Build de Amazon Developer Console

3.3.7. AWS Lambda

Se pueden usar funciones Lambda para implementar servicios de una skill Alexa. El kit de desarrollo de Alexa provee la API, herramientas y documentación para crear skills, cuya lógica será controlada a través de funciones Lambda. Estas funciones se llamarán desde el endpoint de la skill Alexa, que se conecta a AWS Lambda a través de una URI ARN.

A través de las funciones Lambda la skill de Alexa puede enviar peticiones HTTP al servidor con la API REST de nuestra aplicación, hosteado en una máquina virtual de la Universidad de Valladolid.

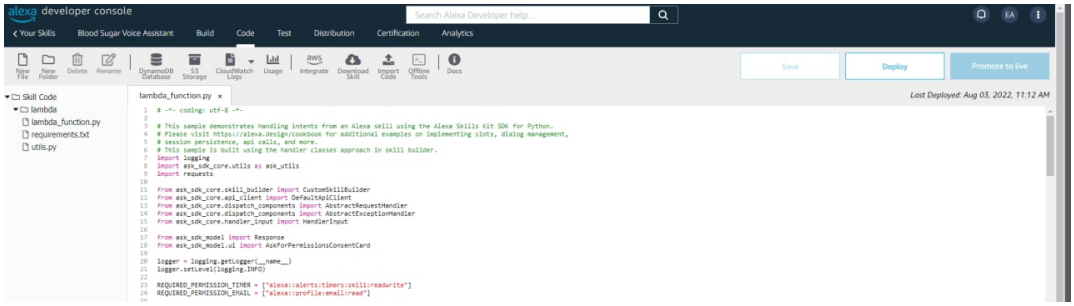


Figura 3.16: Pestaña de Code de Amazon Developer Console

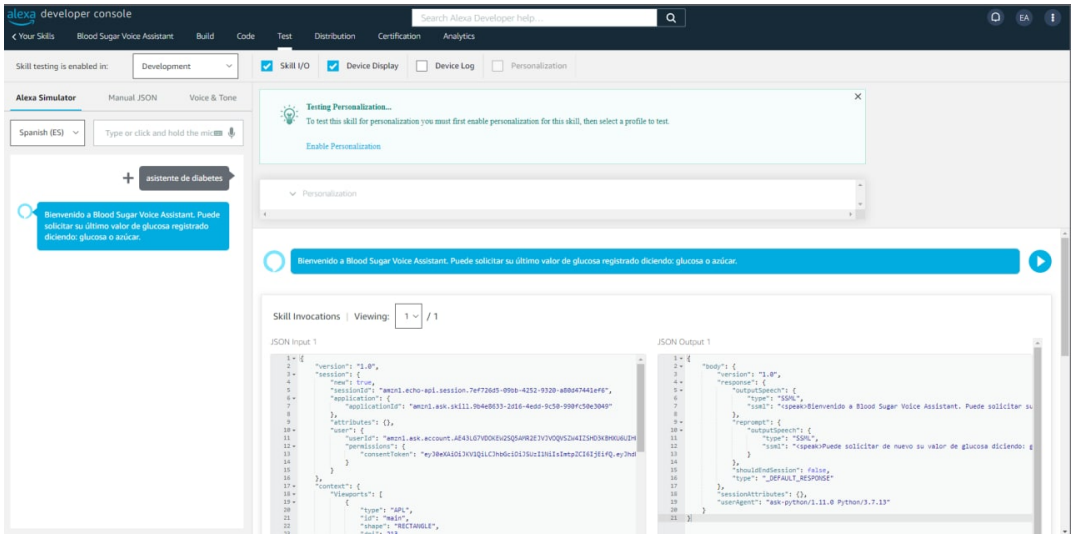


Figura 3.17: Pestaña de Test de Amazon Developer Console con ouput en formato JSON

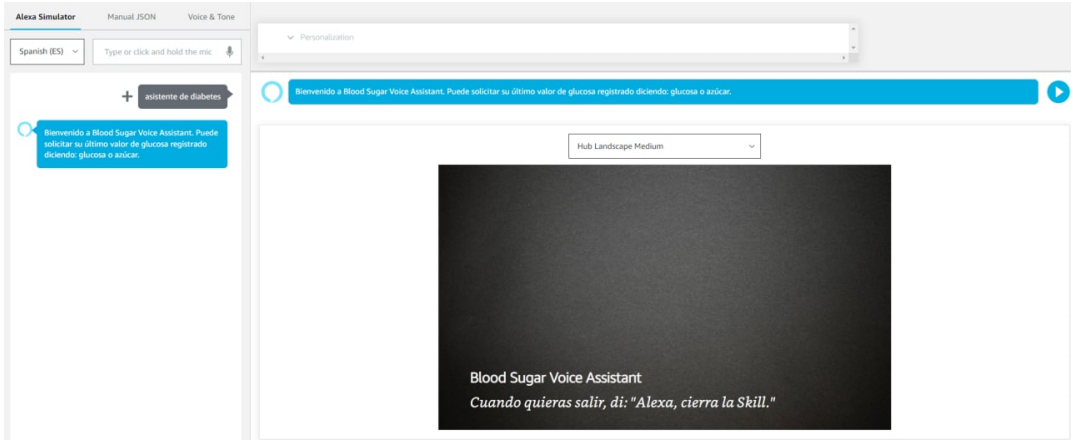


Figura 3.18: Pestaña de Test de Amazon Developer Console con output en pantalla

Capítulo 4

Análisis

Tras elegir las tecnologías a usar en el proyecto se continuó con el análisis del sistema. Este análisis se realizó principalmente en el Sprint 0 de preparación, pero también ha tenido lugar durante el comienzo de cada Sprint.

Este proceso de análisis incluye:

- **Elicitación de requisitos:** al usar Scrum en nuestro proyecto, los requisitos han sido definidos en forma de historias de usuario en las secciones 2.6 y 2.7. En este capítulo se definirán los requisitos de información, también como historias de usuario.
- **Modelo de dominio:** donde se definen las entidades del dominio a partir de los requisitos de información.
- **Máquina de estados:** la skill de Alexa funciona como una máquina de estados, ya que el usuario tiene 3 estados principales en función de su nivel de azúcar, con distintas transiciones entre ellos.
- **Proceso de negocio:** donde se define el proceso de interacción "correcto" entre el Usuario y la Skill. En el siguiente capítulo se modelarán todas las interacciones realizadas en diseño mediante un diagrama de flujo.

4.1. Requisitos de información como historias de usuario

Los requisitos de información pertenecen a los requisitos funcionales. Sin embargo, se ha decidido incluirlos en análisis ya que a partir de ellos se establecerán las entidades del Modelo de Dominio inicial. Se definen en la Tabla 4.1.

Requisitos de información como Historias de Usuario
Como desarrollador necesito almacenar email y contraseña de un **Usuario**.
Como desarrollador necesito almacenar valor, fecha de medida y usuario de una **Medida de Glucemia**.

Tabla 4.1: Requisitos de información como Historias de Usuario

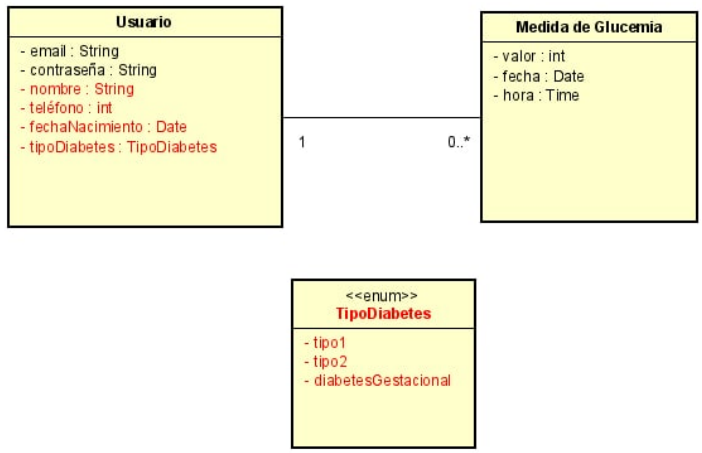


Figura 4.1: Modelo de dominio

4.2. Modelo de dominio

El Modelo de dominio se ha desarrollado a partir de los requisitos funcionales de información de la Tabla 4.1. Es un modelo sencillo, ya que la la aplicación maneja pocos datos. La complejidad reside en el desarrollo de la Skill Alexa, que contiene la lógica para actuar en función del valor de glucemia medido.

En color rojo se presentan atributos que se consideran útiles si la aplicación fuera a ser empleada en el mundo real, fuera del ámbito de un Trabajo de Fin de Grado. Por ejemplo, sería muy útil almacenar datos sanitarios del usuario, para poder proporcionar recomendaciones personalizadas a cada usuario, ya que no todos los diabéticos deben realizar el mismo tratamiento cuando tienen unos niveles de azúcar concretos.

4.3. Máquina de estados

Como ya se ha mencionado, la Skill de Alexa Blood Sugar Voice Assistant funciona como una máquina de estados. El usuario puede encontrarse en 3 estados, y las diferentes transiciones entre todos ellos definen las respuestas realizadas por la Skill. Estos 3 estados principales son:

- Hipoglucémico: valores de glucemia = 70 mg/dl.
- Normoglucémico: valores de glucemia 70 - 180 mg/dl.
- Hiperglucémico: valores de glucemia = 180 mg/dl.

La complejidad de la skill reside en la gestión de las transiciones de los distintos estados. De esto se encarga la Skill de Alexa, que procesa el último valor de glucemia, lo compara con el valor anterior, y en función del estado actual y el estado anterior establece una respuesta concreta. Destacar que puede haber dos transiciones a un mismo estado, en función de si el valor más reciente es superior o inferior al anterior o si se encuentra en un rango de valores. Por ejemplo, no es lo mismo tener hiperglucemia con 180 de azúcar que 250, y la skill gestiona estos rangos de valores que se pueden observar como transiciones dobles entre dos mismos estados en el diagrama.

La máquina de estados con sus transiciones se puede observar en el Diagrama UML 4.2.

4.4. Modelo de proceso de negocio

Para modelar la interacción del usuario con el asistente de voz Alexa se ha escogido un diagrama de actividades 4.3 en el que se representa el flujo de ejecución básico de la aplicación.

Se puede observar que en la interacción intervienen 3 actores: usuario, alexa y el medidor de glucosa. Usuario solicita a Alexa abrir la aplicación mediante la *frase de activación*. A continuación, Alexa solicita una acción y el Usuario solicita su último valor de glucosa mediante un *intent*. Finalmente, Alexa comprueba si el usuario se encuentra registrado en el agente Medidor de glucemia, y si no lo está, solicita su registro. Si está registrado, obtiene el último valor de glucosa para el usuario y ejecuta la acción correspondiente definida en el diagrama 4.2. Estos valores de glucosa son generados cada minuto de forma contínua por el Medidor de Glucemia.

Este diagrama de actividades se verá complementado en diseño con un diagrama de flujo en el que se mostrarán todos los flujos alternativos de la aplicación: sentencia del usuario no entendida por Alexa, usuario sin permisos, solicitar ayuda, salir de la aplicación, etc.

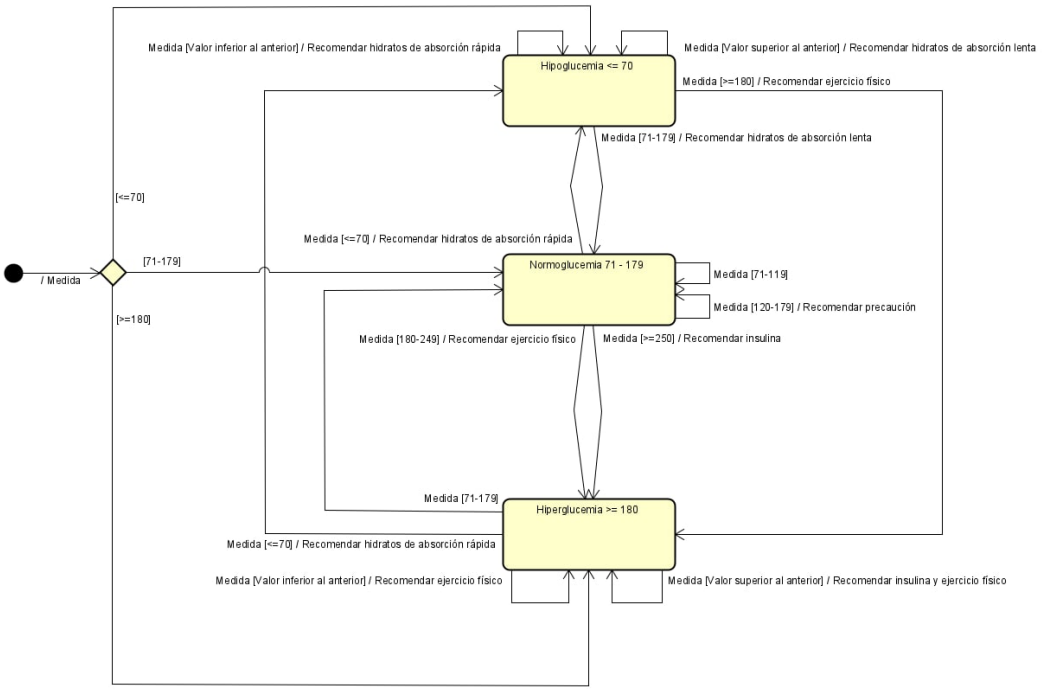


Figura 4.2: Máquina de estados

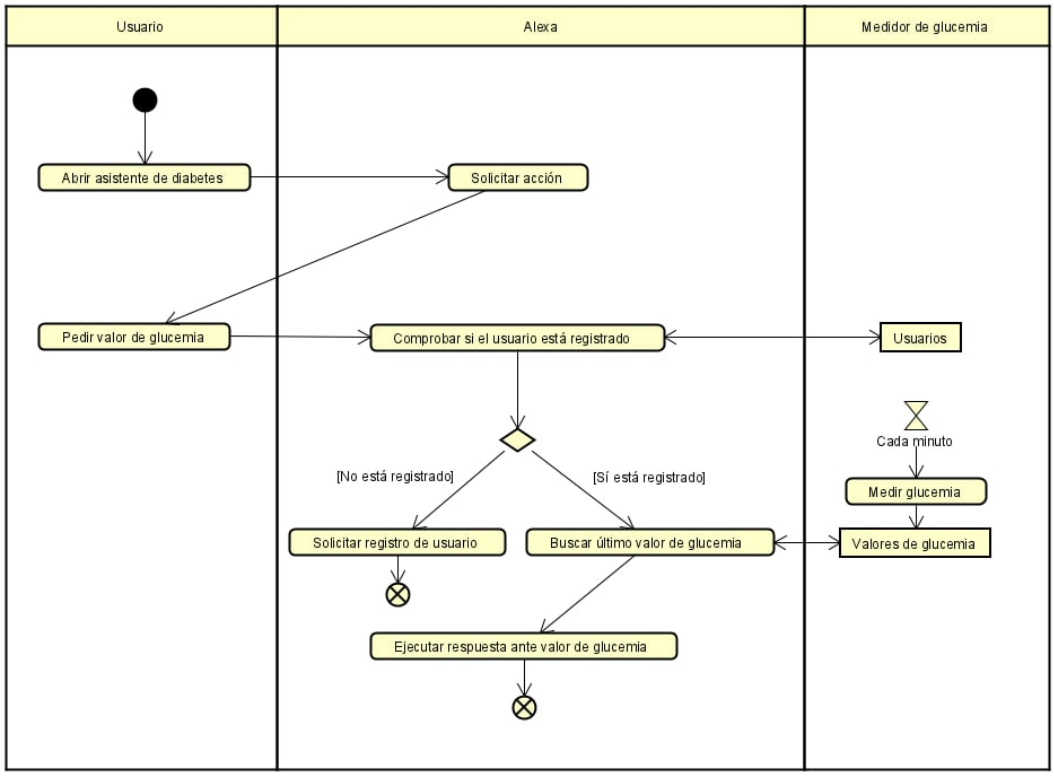


Figura 4.3: Diagrama de actividades del proceso de negocio **solicitar último valor de glucemia**.

Capítulo 5

Diseño

En este capítulo se explican las decisiones tomadas en el diseño del sistema, tras realizar el proceso de análisis. Estas decisiones incluyen la arquitectura general del sistema, elección de lenguajes de programación, diseño de los datos, patrones de diseño, aplicacion web, API Rest, desarrollo de skills de Alexa, flujo de ejecución y despliegue del sistema.

5.1. Arquitectura general del sistema

La primera decisión tomada en diseño fue establecer la arquitectura general del sistema 5.1. El documento de especificaciones del proyecto compartido por HP SCDS propone una arquitectura con tres componentes. En primer lugar se encuentra el componente **API REST** que procesa las peticiones http para crear y logear usuarios, o crear y retornar valores de glucosa. Esta API REST obtiene sus datos de un componente **Base de Datos**, que garantiza la persistencia del sistema. Por último tenemos el componente **Skill de Alexa**, que realiza peticiones http a la API REST para obtener el último valor de glucosa de un usuario concreto.

Sin embargo, en la fase de diseño el desarrollador consideró oportuno añadir un componente que no se contemplaba en las especificaciones. Para facilitar el uso de la aplicación por los usuarios se decidió añadir un componente **Aplicación Web**, que permitiera registrar y logear usuarios, mostrar los valores de glucosa de los últimos 7 días y exportar a PDF estos valores. Además, por motivos de privacidad, se decidió añadir como opciones eliminar todos los datos de la aplicación y borrar la cuenta de un usuario.

El diagrama 5.1 representa esta arquitectura. Posteriormente se entrará en detalle de cómo se ha desarrollado cada componente en el diagrama de despliegue.

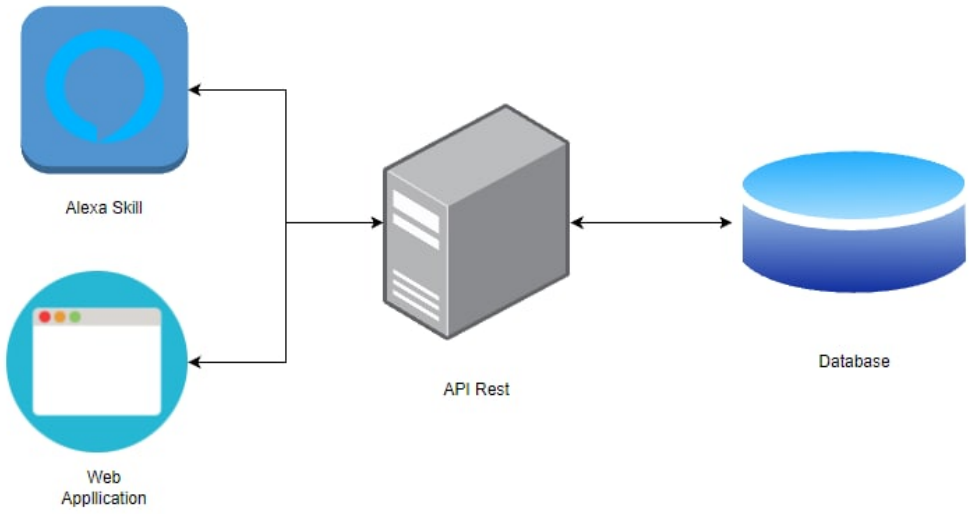


Figura 5.1: Esquema genérico de la arquitectura de la aplicación Blood Sugar Voice Assistant

5.2. Elección de lenguajes de programación

Una vez ya se tenía decidido cómo iba a ser la arquitectura de la aplicación, se continuó por decidir qué lenguajes de programación se iban a emplear para programar los distintos componentes.

En primer lugar, para la API Rest se empleó Golang como lenguaje de desarrollo, a propuesta de HP SCDS. Como se explicó en la Sección 3.3.1, Golang es un lenguaje perfecto para desarrollar Backend. Es un lenguaje sencillo, con sintaxis clara y concisa, y sobre todo, destaca su gran rendimiento y su facilidad para crear tareas concurrentes. El desarrollador no había programado nunca en Golang y se consideró que este proyecto era una oportunidad fantástica para conocer este lenguaje de programación de reciente aparición.

Para la aplicación Web, el desarrollador eligió Angular como framework de desarrollo, ya que lo había empleado anteriormente y al ser la aplicación web funcionalidad añadida, no quería emplear mucho tiempo en aprender una tecnología nueva.

Con respecto a la Base de Datos, se decidió emplear MySql (aunque no sea un lenguaje de programación se ha decidido incluir en este apartado por motivos de simplicidad) al ser esta una de las distribuciones de Bases de Datos más utilizadas en todo el mundo.

Por ultimo encontramos la Skill Alexa. La plataforma de desarrollo de Amazon para Skills de Alexa, Amazon Developer Console, permite el desarrollo de skills en numerosos lenguajes de programación, como Node, Python, Go, Java, ... Además, permite que los desarrolladores hosteen el código en AWS Lambda, o en sus propios servidores, por lo que virtualmente es

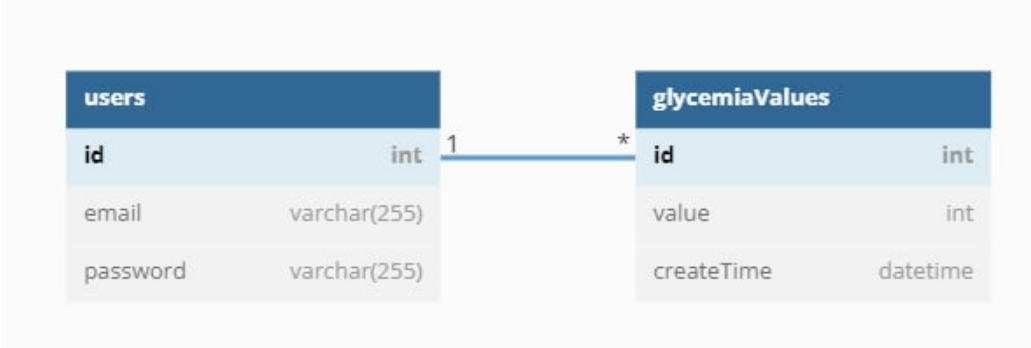


Figura 5.2: Diagrama de Diseño Lógico Relacional de la Base de Datos

posible desarrollar una skill con cualquier lenguaje de programación. Se eligió Python, ya que es el lenguaje de programación que más domina el desarrollador. También se escogió AWS Lambda como servidor backend de la skill, ya que facilita mucho el proceso de desarrollo. Esto permitió al desarrollador centrarse en la dificultad de aprender de cero cómo es el proceso de desarrollo de Skills de Alexa, y las APIS de Amazon para gestionar los servicios de las skills.

5.3. Diseño de los datos

A continuación se desarrolló el diseño de la Base de Datos MySql que proporciona persistencia a nuestro proyecto. El diseño final 5.2, muy sencillo, es el obtenido a partir del Modelo de Dominio inicial generado en análisis en 4.2. Como ya se ha explicado, solo se almacenan los datos necesarios para una proof of concept de nuestra Skill. La versión definitiva debería almacenar más datos de los usuarios, ya que cada individuo es particular a la hora de tratar la diabetes. Podríamos almacenar, por ejemplo, distintos umbrales de glucemia para cada usuario, tratamientos personalizados, etc.

5.4. Patrones de diseño

En el desarrollo de este proyecto se han empleado dos patrones de diseño, MVVM (Model-View-ViewModel) y DAO-DTO. El primero de ellos se pone de manifiesto en la arquitectura de la aplicación web desarrollada con Angular y el segundo se ha empleado en la API Rest para manejar la persistencia de los datos.

5.4.1. MVVM

El patrón de diseño Model-View-ViewModel [26] es un refinamiento del patrón de diseño Modelo-Vista-Controlador. Surgió con el objetivo de facilitar la separación del desarrollo de

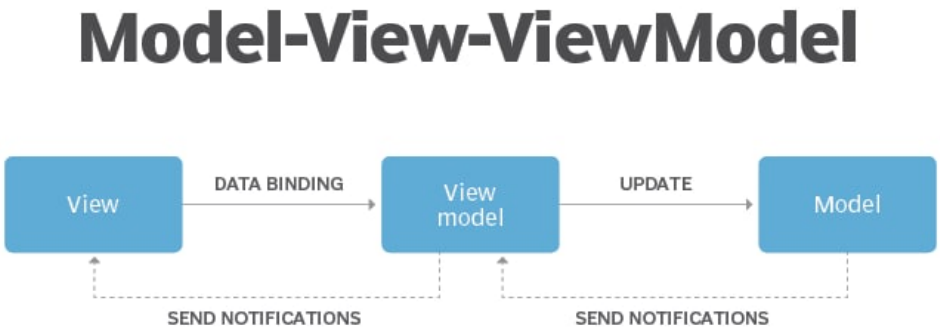


Figura 5.3: Patrón MVVM. Imagen tomada de [26]

la interfaz de usuario, también denominada como Capa de Presentación. Distinguimos tres componentes:

- **Model:** Contiene la lógica de negocio y el estado de la aplicación.
- **View:** Elementos visuales que representan el Modelo. También puede recibir inputs del usuario.
- **ViewModel:** Se encuentra a mitad de camino entren la capa de View y Model. Aquí se contienen los controles para interactuar con la View, que actualizan el Model.

En la figura 5.3 se pueden observar las relaciones entre los 3 componentes del patrón de diseño.

El framework Angular emplea una adaptación de MVVM en la que los distintintos compomentes de MVVM se representan como un tipo de ficheros distintos dentro del framework. Dentro de la Aplicación Web de este proyecto tenemos:

- **Model:** Se encuentra disperso en numersos elementos del framework. Por ejemplo en inyecciones de dependencias o en elementos del dominio. Por ejemplo *app.model.ts*.
- **View:** Se corresponde con los archivos .html de los componentes Angular. Por ejemplo *glucose-list.component.html*.
- **ViewModel:** Se corresponde con las clases Typescript de los componentes Angular que manejan los eventos generados en las Views. Por ejemplo *glucose-list.component.ts*.

5.4.2. DAO-DTO

El patrón de diseño DAO-DTO se puede subdividir en dos patrones. DAO es una abreviación de **Data Access Object** y encapsula la lógica de acceder, actualizar y guardar

datos en el almacenamiento persistente de una aplicación. DTO, por otro lado, significa **Data Transfer Object** y se utiliza para encapsular los datos que vamos a compartir entre procesos.

En nuestro proyecto se ha empleado DAO-DTO en la API Rest para gestionar de una forma sencilla el acceso a la Base de Datos y el manejo de los datos. En el paquete **database** se encuentran los ficheros Golang que permiten el acceso a la Base de Datos MySql desplegada para la aplicación (DAO). Además, en el paquete **entity** se encuentran los ficheros Golang que contienen las clases DTO que representan los objetos del dominio, en nuestro caso usuarios y medidas de glucemia.

5.5. Aplicación web

Como ya se ha explicado, se escogió Angular como framework de desarrollo para la aplicación web. En un principio no estaba contemplada en las especificaciones del proyecto, pero el desarrollador consideró oportuno añadirla para facilitar el proceso de registro y logeo de usuarios (imagen 5.4). Además, la aplicación de Blood Sugar Voice Assistant, permite que los usuarios accedan a sus datos de glucemia de los últimos 7 días 5.5), facilitando su exportación a un fichero PDF si desean imprirlos (imagen 5.6). Por último, la plataforma web también permite eliminar los datos almacenados en la misma o la cuenta de un usuario para garantizar la privacidad de los miembros de la plataforma.

Angular sigue el patrón de **Diseño basado en Componentes**, que se puede ver reflejado en los componentes *user-login* y *glucose-list* en nuestra aplicación, diagrama 5.7. El componente glucose-list es el responsable de gestionar los valores de glucemia del usuario, mientras que user-login se encarga de gestionar el login y registro de usuarios. El desarrollador no ha separado login y registro en dos componentes distintos, aunque esto sería lo ideal, ya que esta funcionalidad era secundaria y no quería emplear más tiempo del necesario en su desarrollo por falta de tiempo.

5.6. API Rest

La API Rest ha sido desarrollada en su totalidad en Golang. Es la encargada de escuchar las peticiones http de los clientes en nuestra aplicación Blood Sugar Voice Assistant. Dentro de nuestra arquitectura, estos clientes serían tanto la Aplicación Web como la Skill de Alexa.

Además, ya que no tenemos acceso a datos reales de usuarios diabéticos por motivos de privacidad, la API Rest se encarga también de introducir automáticamente datos sintéticos de glucemia cada minuto para todos los usuarios registrados. Esta tarea es muy sencilla de integrar en la API Rest ya que Golang facilita la programación de tareas concurrentes. Sin embargo, el desarrollador considera que hubiera sido más adecuado desde un punto de vista de Ingeniería del Software sacar esta funcionalidad de la API Rest y programar un pequeño script que se ejecutara cada minuto e introdujera datos sintéticos mediante una

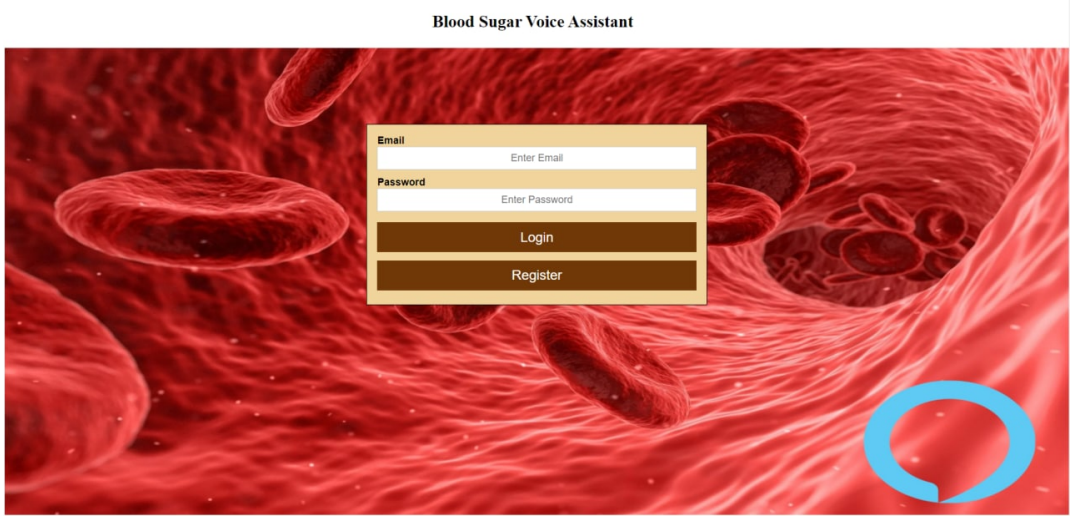


Figura 5.4: Pantalla inicial de la aplicación

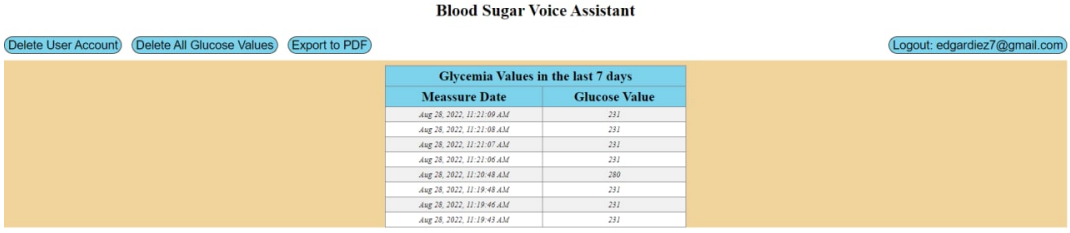
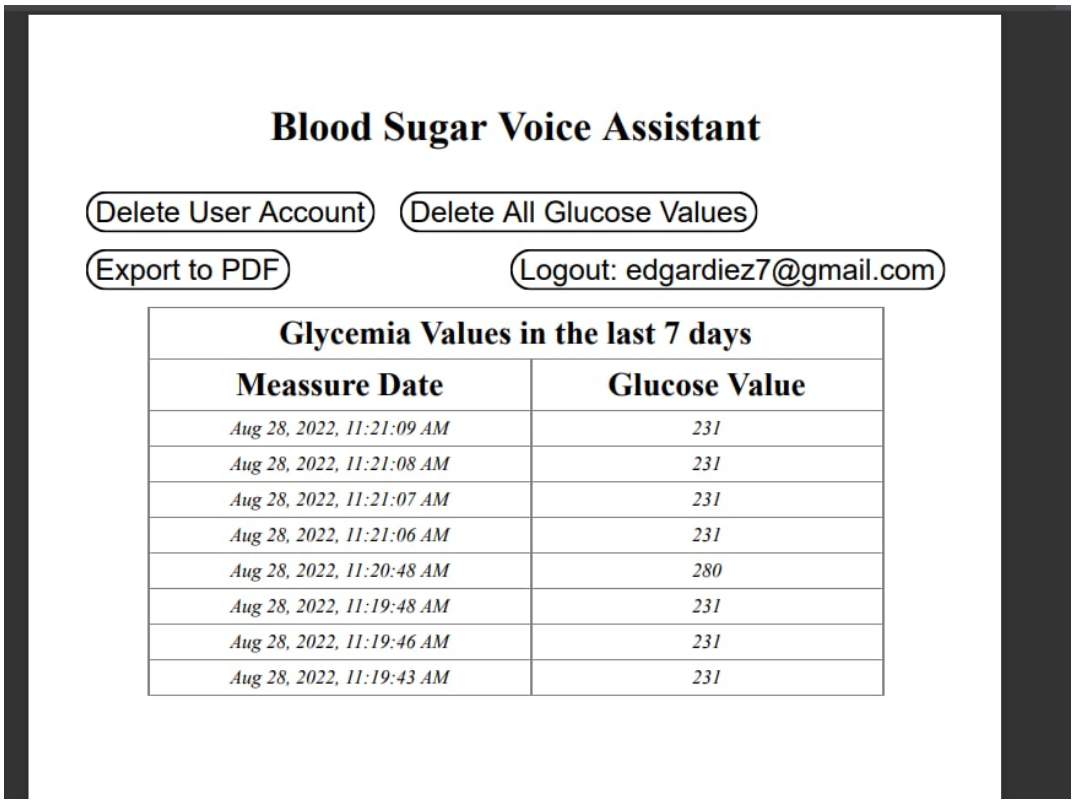


Figura 5.5: Vista con los datos de glucemia guardados en los últimos 7 días



Blood Sugar Voice Assistant

Delete User Account | Delete All Glucose Values

Export to PDF | Logout: edgardiez7@gmail.com

Glycemia Values in the last 7 days	
Meassure Date	**Glucose Value**
Aug 28, 2022, 11:21:09 AM	*231*
Aug 28, 2022, 11:21:08 AM	*231*
Aug 28, 2022, 11:21:07 AM	*231*
Aug 28, 2022, 11:21:06 AM	*231*
Aug 28, 2022, 11:20:48 AM	*280*
Aug 28, 2022, 11:19:48 AM	*231*
Aug 28, 2022, 11:19:46 AM	*231*
Aug 28, 2022, 11:19:43 AM	*231*

Figura 5.6: PDF exportado por la aplicación web con datos de glucemia de los últimos 7 días.

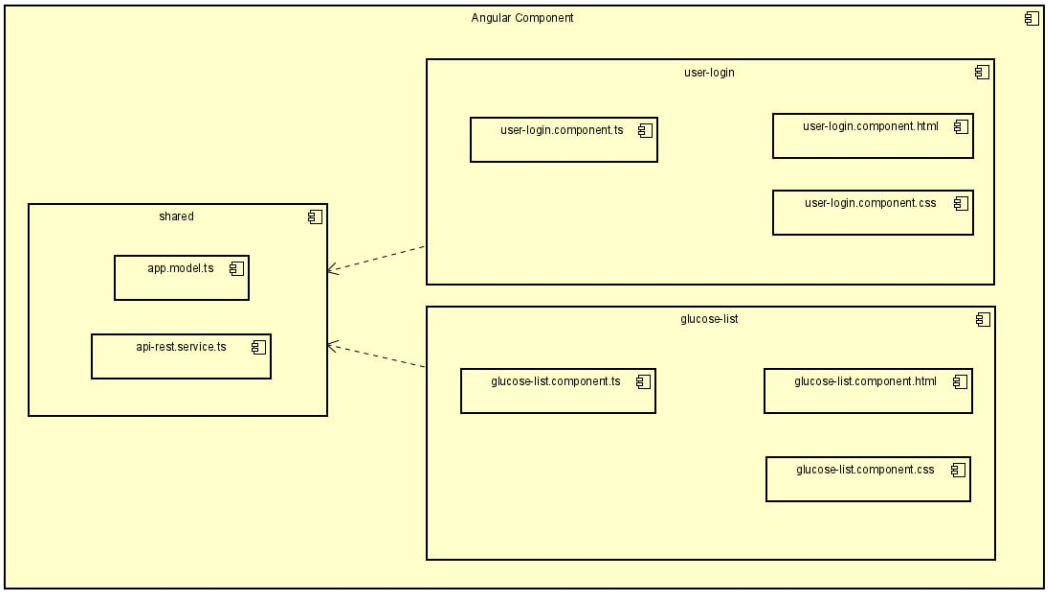


Figura 5.7: Diagrama de Componentes Angular de la Aplicación Web

petición POST a la API Rest. Esto no se ha realizado ya que el desarrollador debía seguir los principios establecidos en el documento de requisitos del proyecto generado por HP SCDS.

Se puede ver el resumen de todas las operaciones http implementadas en las tablas 5.8 - 5.12

5.7. Desarrollo de Skills de Alexa

El proceso de desarrollo de Skills Alexa ya ha sido explicado en los apartados 3.3.6 y 3.3.7. Como resumen, para crear una Skill de Alexa, el desarrollador debe registrarse en la plataforma Alexa Developer Console. Una vez ha iniciado sesión, debe asignar a su skill (figura 5.13):

- **Nombre único:** para identificar la skill en la tienda de skills de Amazon. Nuestra skill se llama Blood Sugar Voice Assistant.
- **Locale:** Idioma y localización para la skill. En nuestro caso es *Español (ES)*.
- **Modelo:** La plataforma permite elegir plantillas de Skills para facilitar el desarrollo de skills similares. Nuestra skill ha sido construida de cero, por lo que se le asignó el modelo *custom*.
- **Backend:** Dónde se va a hostear la lógica de la aplicación. Podemos elegir nuestro

Nombre	Login
Descripción	Devuelve el id del usuario con un email y una contraseña concretos
Método http	**POST**
Ruta	/login
Variables de la ruta	—
Parámetros de la petición	—
Cuerpo de la petición	• email • password
Respuesta	• 200: StatusOk + user ID. Si las credenciales coinciden con un usuario • 404: StatusNotFound. Si las credenciales no coinciden con ningún usuario

Nombre	GetAllUsers
Descripción	Devuelve todos los usuarios registrados
Método http	**GET**
Ruta	/user
Variables de la ruta	—
Parámetros de la petición	—
Cuerpo de la petición	—
Respuesta	• 200: StatusOk + users

Figura 5.8: Documentación de la API Rest (1/5)

Nombre	GetUserById
Descripción	Devuelve el usuario con un id determinado
Método http	**GET**
Ruta	/user/{id}
Variables de la ruta	id
Parámetros de la petición	—
Cuerpo de la petición	—
Respuesta	• 200: StatusOk + user ID. Si el id coincide con un usuario • 404: StatusNotFound. Si el id no coincide con ningún usuario

Nombre	AddNewUser
Descripción	Devuelve el id del usuario con un email y una contraseña concretos
Método http	**POST**
Ruta	/user
Variables de la ruta	—
Parámetros de la petición	—
Cuerpo de la petición	• email • password
Respuesta	• 200: StatusOk + user ID. Si las credenciales coinciden con un usuario • 404: StatusNotFound. Si las credenciales no coinciden con ningún usuario • 409: StatusConflict. Si ya hay un usuario registrado con ese email.

Figura 5.9: Documentación de la API Rest (2/5)

Nombre	DeleteUser
Descripción	Elimina un usuario y por consecuencia todos los valores de glucemia asociados a él
Método http	**DELETE**
Ruta	/user/{id}
Variables de la ruta	id
Parámetros de la petición	—
Cuerpo de la petición	—
Respuesta	• 200: StatusOk

Nombre	GetLastGlucoseValueByUserId
Descripción	Devuelve el id del usuario con un email y una contraseña concretos
Método http	**GET**
Ruta	/user/{id}/glucose/last
Variables de la ruta	id
Parámetros de la petición	—
Cuerpo de la petición	—
Respuesta	• 200: StatusOk + valor glucosa. Si el id pertenece a un usuario • 404: StatusNotFound. Si el id no pertenece a un usuario o no hay medidas de glucemia.

Figura 5.10: Documentación de la API Rest (3/5)

Nombre	GetSevenDaysGlucoseValueByUserId
Descripción	Devuelve el id del usuario con un email y una contraseña concretos
Método http	**GET**
Ruta	/user/{id}/glucose/seven_days
Variables de la ruta	id
Parámetros de la petición	—
Cuerpo de la petición	—
Respuesta	• 200: StatusOk + valores glucosa. Si el id pertenece a un usuario • 404: StatusNotFound. Si el id no pertenece a un usuario o no hay medidas de glucemia.

Nombre	AddGlucoseValueByUserId
Descripción	Añade una medida de glucemia para un usuario con un id determinado
Método http	**POST**
Ruta	/user/{id}/glucose
Variables de la ruta	id
Parámetros de la petición	—
Cuerpo de la petición	value
Respuesta	• 201: StatusCreated. Si el id pertenece a un usuario • 400: StatusBadRequest. Si el id no pertenece a un usuario o hay un error en el cuerpo de la petición

Figura 5.11: Documentación de la API Rest (4/5)

Nombre	DeleteGlucoseValuesByUserId
Descripción	Elimina los valores de glucemia registrados para un usuario
Método http	**DELETE**
Ruta	/user/{id}/glucose
Variables de la ruta	id
Parámetros de la petición	—
Cuerpo de la petición	—
Respuesta	• 202: StatusAccepted. Si el id pertenece a un usuario • 404: StatusNotFound. Si el id no pertenece a ningún usuario

Nombre	GetLastGlucoseValueByUserEmail
Descripción	Devuelve el último valor de glucemia de un usuario con email determinado
Método http	**GET**
Ruta	/user/{email}/glucose/last_value
Variables de la ruta	email
Parámetros de la petición	—
Cuerpo de la petición	—
Respuesta	• 200: StatusOk + valor glucosa. Si el id pertenece a un usuario y hay valores de glucemia • 404: StatusNotFound. Si el email no pertenece a un usuario o no hay medidas de glucemia.

Figura 5.12: Documentación de la API Rest (5/5)

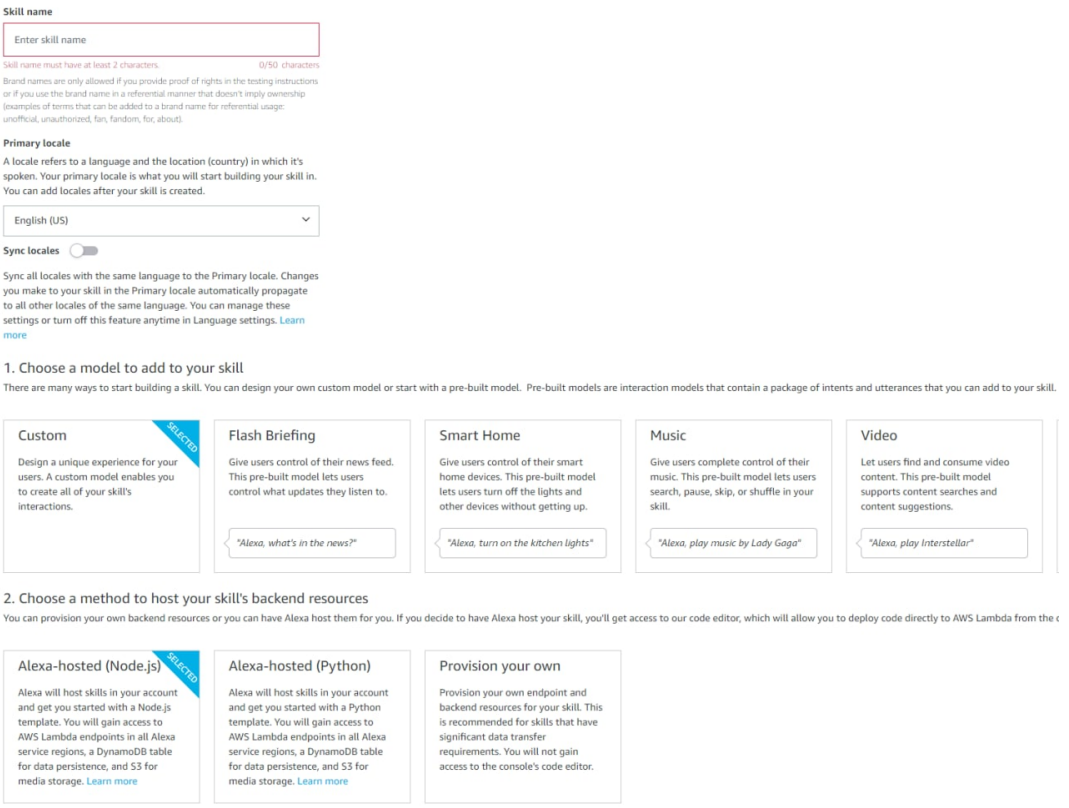


Figura 5.13: Formulario de creación de una Skill Alexa

propio servidor o un servidor AWS Lambda (opción "hosteado por Alexa"). En nuestra skill se ha elegido *hosteado por Alexa (Python)*.

Una vez se ha creado el esqueleto base de la Skill, el desarrollador debe asignar una **frase de activación** (*"Asistente de diabetes"*) a la Skill, esta frase activará la skill en el dispositivo Alexa cuando sea enunciada por el usuario. A continuación, se establecen los **intents**, que son las sentencias que, una vez abierta la skill, activarán las distintas acciones que pueda realizar la aplicación. Estos intents pueden ser activados con distintas frases, o **declaraciones**. Este paso se realiza en la pestaña de "build" de la plataforma, figura 3.15. En nuestra skill, los intents permitidos al usuario son:

- **Glucosa:** Se activa con numerosas declaraciones, como *"glucosa", "azúcar", "dime mi último valor de glucemia"*, etc. Cuando se lanza, se produce la acción asignada al último valor de glucosa medido, diagrama 4.2.
- **Ayuda:** Activado con con la declaracion del mismo nombre, la skill lee el mensaje de ayuda: *"Puede solicitar su último valor de glucosa registrado diciendo: glucosa o azúcar"*.

- **Salir:** Activado con con la declaracion del mismo nombre, cierra la skill Alexa.
- **Intent Desconocido:** Se ejecuta cuando ningún intent definido en la skill tiene una declaración que se corresponda con la frase del usuario. La skill lee el mismo mensaje que el intent de ayuda.

Para entender mejor los intents y sus declaraciones se presenta el contenido del fichero del Modelo de Interacción de la Skill Blood Sugar Voice Assisntant, 5.14.

Para desarrollar la lógica de la aplicación nos desplazamos a la pestaña de "code" de Alexa Developer Console, figura 3.16. Esta pestaña solo se habilita si el backend está hosteado en AWS Lambda, como es nuestro caso. Aquí es donde se le asigna código a cada intent, para que realice la opción correspondiente. Nuestra skill ha sido desarrollada totalmente en Python.

Por último, para testear la skill tenemos la pestaña de "test", figuras 3.17 y 3.18. Nos permite emular un dispositivo Alexa, usando comandos de voz o de texto y obteniendo output de voz y de texto. Esto permite testear la aplicación fácilmente para distintos dispositivos, ya que, por ejemplo, los dispositivos Amazon Echo Dot solo emiten audio, mientras que los Echo Show (o la mayoría de smartwatches con Alexa integrada) pueden mostrar texto por pantalla.

5.8. Flujo de ejecución de la aplicación

En el diagrama 5.15 se puede observar el flujo de la interacción entre el Usuario y la Skill del asistente Blood Sugar Voice Assistant.

El usuario abre la aplicación enunciando la frase de activación de Alexa: *"Alexa, abre mi asistente de diabetes"*. A continuación Alexa ejecuta la Skill Blood Sugar Voice Assistant con su frase de bienvenida: *"Bienvenido a Blood Sugar Voice Assistant. Puede solicitar su último valor de glucosa registrado diciendo: glucosa o azúcar"*. El Usuario ahora debe ejecutar uno de los intents de la aplicación:

- Si solicita *"Ayuda"* o la Skill no mapea el intent con ninguno de los registrados, la Skill ejecuta el mensaje de ayuda: *"Puede solicitar su último valor de glucosa registrado diciendo: glucosa o azúcar."*
- Si solicita *"Salir"*, la Skill se desactiva.
- Por último, si el Usuario ejecuta el intent "Glucosa" mediante una de sus numerosas frases de activación, por ejemplo *"Alexa, dime mi último valor de azúcar"* o simplemente *"Glucosa"*, la skill comprueba que el usuario se haya registrado en la aplicación con el correo electrónico asociado a su cuenta de Amazon (se recuerda que el registro en la aplicación se realiza en la plataforma web). Si el usuario no está registrado, la Skill le solicita que proceda a registrarse: *"El email de su cuenta de Amazon no está registrado en la aplicación Blood Sugar Voice Assistant. Por favor, acceda a la web de la aplicación para crearse una cuenta de usuario con el mismo correo electrónico que*

```
{
  "interactionModel": {
    "languageModel": {
      "invocationName": "asistente de diabetes",
      "intents": [
        {
          "name": "AMAZON.CancelIntent",
          "samples": [
            "Hasta luego",
            "Cancelar",
            "Cerrar",
            "Salir"
          ]
        },
        {
          "name": "AMAZON.HelpIntent",
          "samples": [
            "Ayuda por favor",
            "No sé qué hacer",
            "Necesito ayuda",
            "Duda",
            "Ayuda"
          ]
        },
        {
          "name": "AMAZON.StopIntent",
          "samples": []
        },
        {
          "name": "AMAZON.NavigateHomeIntent",
          "samples": []
        },
        {
          "name": "AMAZON.FallbackIntent",
          "samples": []
        },
        {
          "name": "LastGlucoseValue",
          "slots": [],
          "samples": [
            "Cuál es mi nivel de azúcar",
            "Cuál es mi nivel de glucosa",
            "Cuál es mi último valor de glucosa ",
            "Cuál es mi último valor de azúcar ",
            "Azúcar en sangre",
            "Azúcar",
            "Glucosa",
            "Di mi glucosa en sangre",
            "Di mi azúcar en sangre",
            "Último valor de glucosa",
            "Último valor de azúcar"
          ]
        }
      ],
      "types": []
    }
  },
  "version": "19"
}
```


Figura 5.14: Modelo de Interacción de la Skill Blood Sugar Voice Assistant

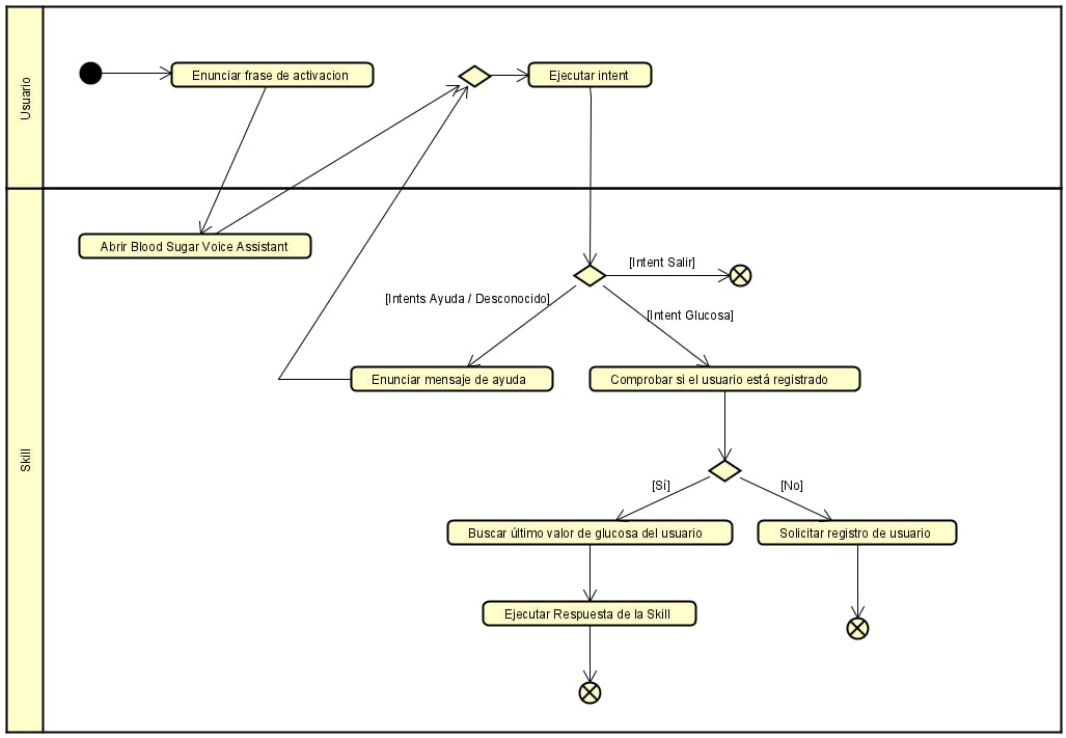


Figura 5.15: Diagrama de Actividad de la Skill Blood Sugar Voice Assistant

su cuenta de Amazon". Por el contrario, si lo está, consulta su último valor de glucosa y ejecuta la acción que se corresponda al valor de glucosa, figura 4.2. Por ejemplo, si el usuario tiene hiperglucemia y su valor ha sido superior a la anterior medida: *"Su último valor de glucosa registrado es xxx. Sigue con hiperglucemia y sus valores han empeorado. Por favor, realice ejercicio físico y corrija su valor de glucemia con insulina urgentemente. Se ha establecido un temporizador de 30 minutos. Por favor, compruebe de nuevo su nivel de azúcar en sangre al finalizar el temporizador."*

Las actividades *Comprobar si el usuario está registrado* 5.16 y *Buscar último valor de glucosa del usuario* 5.17 son actividades complejas, en las que interviene otro Agente, el Medidor de Glucosa de la fase de análisis, que se corresponde con el servidor Golang que contiene la API Rest.

5.9. Despliegue del sistema

Como ya se ha explicado, la aplicación Blood Sugar Voice Assistant está formada por 4 componentes: la Skill Alexa, la página web, la API Rest y la Base de Datos. Cada uno de estos componentes debe ser desplegado para que el sistema funcione en su totalidad. Sin embargo,

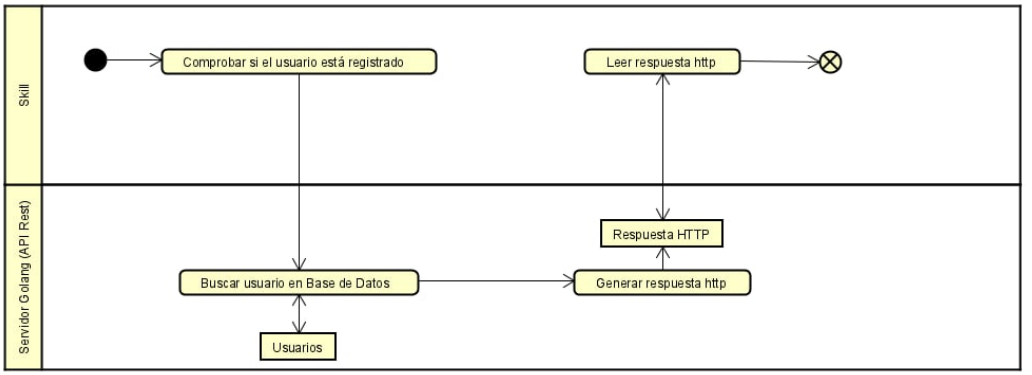


Figura 5.16: Diagrama de Actividad detallado de la Actividad Comprobar Usuario Registrado

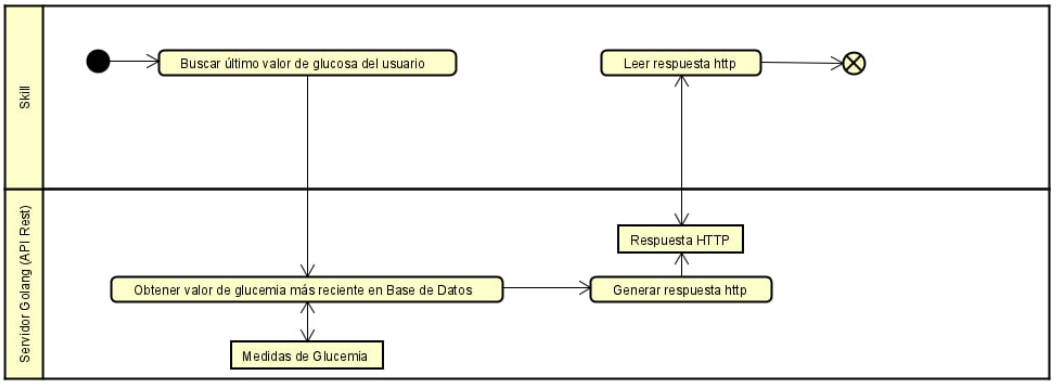


Figura 5.17: Diagrama de Actividad detallado de la Actividad Buscar último valor de glucemia

la aplicación web y la skill de Alexa se comunican con la API Rest pero no entre ellas, por lo que se podrían detener (o caer) los servicios de la primera sin afectar el funcionamiento de la segunda, o viceversa.

Como se puede ver en el Diagrama de Despliegue 5.18, los servidores de la aplicación web y la API Rest, y la Base de Datos se encuentran desplegados en la misma máquina. Se corresponde con una máquina virtual cedida por la Universidad de Valladolid con la finalidad única de realizar este proyecto, con la url **virtual.lab.inf.uva.es**. Por otra parte, la Skill de Alexa se encuentra desplegada en AWS Lambda, plataforma cloud que pertene a Amazon Web Services (AWS) y permite desplegar automáticamente Skills de Alexa. Su integración con Alexa Developer Console se realiza en el momento de crear la Skill, cuando se da la opción de hostear la lógica de la skill en un servidor "custom" o en AWS Lambda.

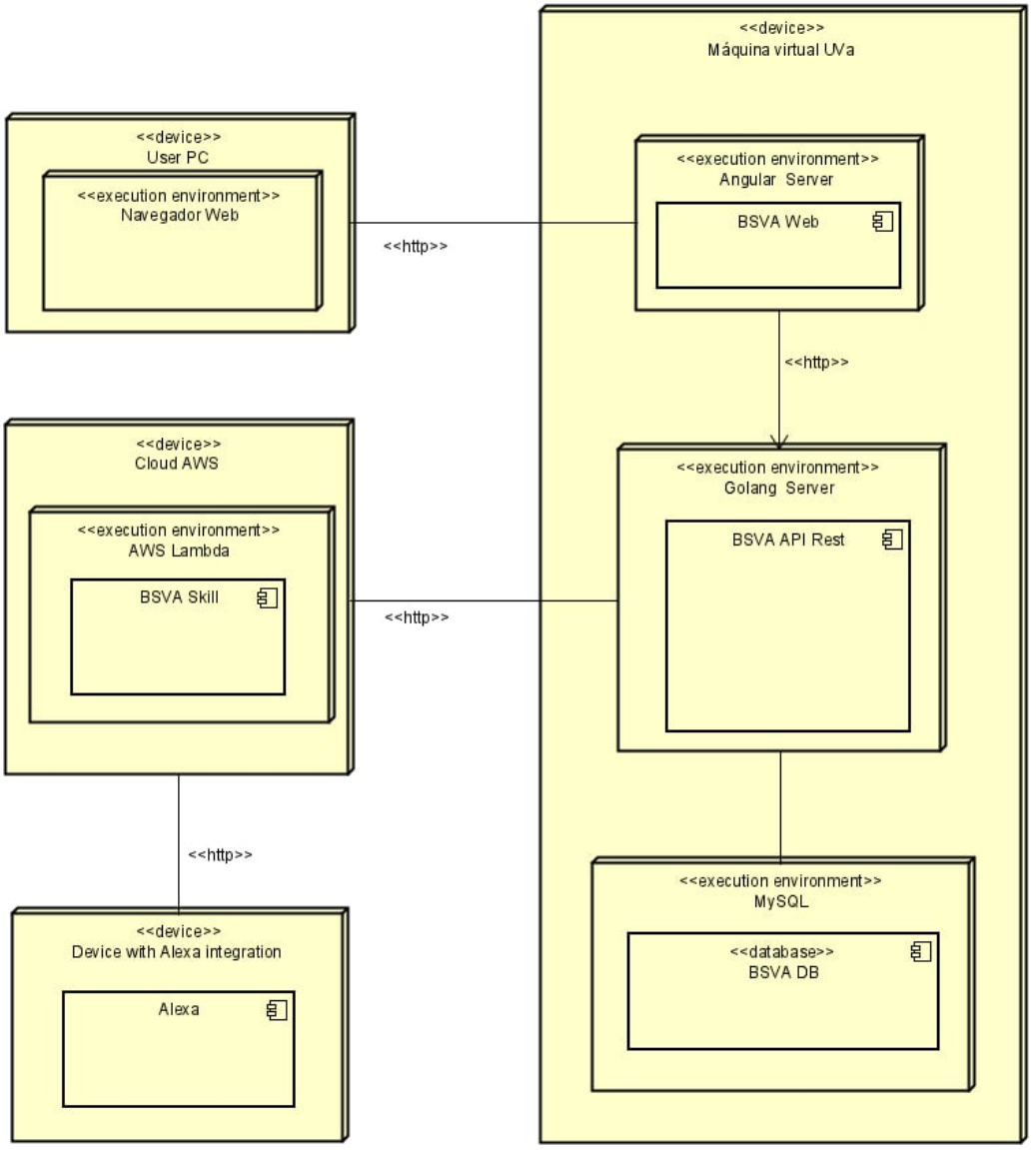


Figura 5.18: Diagrama de Despliegue de la Skill Blood Sugar Voice Assistant

Capítulo 6

Implementación y pruebas

En este capitulo se explica en detalle cómo se ha realizado la programación de la Aplicación Blood Sugar Voice Assistant. Para ello se explicará la estructura de directorios del sistema, el proceso de instalación y despliegue de la plataforma, tests realizados, dificultades encontradas en el desarrollo y licencia del proyecto.

6.1. Implementación

En esta sección se indica detalladamente cómo es la estructura de ficheros de la plataforma Blood Sugar Voice Assistant, así como el proceso de instalación y despliegue de los distintos componentes de la aplicación.

6.1.1. Estructura de directorios de la plataforma

En la figura 6.1 se representa la jerarquía de directorios del repositorio de la aplicación Blood Sugar Voice Assistant.

La carpeta **bsva-alexa-skill** contiene el código relativo a la skill de Alexa. En ella tenemos:

- **interactionModels/lamda:** Contiene el fichero JSON, *es-ES.json* con la configuración del modelo de la Skill, es decir, la estructura lógica de la misma.
- **lambda:** Contiene el código del backend de la Skill, *lambda-function.py*, además del fichero que especifica las dependencias, *requirements.txt*.

La carpeta **bsva-backend** contiene el código relativo al servidor Golang que ejecuta la API Rest de nuestra aplicación. En ella encontramos:

- **bin:** Contiene el fichero binario que levanta el servidor, *bsva-rest-api*.
- **cmd/bloodsugarvoiceassistant:** Donde se encuentra el fichero con la clase main de nuestra API Rest, *main.go*.
- **pkg:** Donde se encuentra el resto de clases del servidor Go:
  - **controller:** El fichero *bloodsugarvalues-controller.go* se encarga de gestionar las distintas peticiones http de la API Rest.
  - **database:** Como se explica en la Sección 5.4.2 aquí se encuentran las clases que implementan los DAOs de nuestra aplicación.
  - **entity:** Como se explica en la Sección 5.4.2 aquí se encuentran las clases que implementan los DTOs de nuestra aplicación.
- **utils:** Aquí está el fichero de utilidades , *DatabaseCreationScript.sql*, que facilita el despliegue de la Base de Datos.

La carpeta **bsva-frontend** contiene el código relativo al servidor Angular que levanta la plataforma web de nuestra aplicación. En ella encontramos los componentes:

- **src/app/glucose-list:** Responsable de mostrar los valores de glucosa del usuario para los últimos 7 días.
- **src/app/shared: :** Responsable de gestionar la comunicación entre el servidor web y la API Rest mediante peticiones http.
- **src/app/user-login: :** Responsable de gestionar el inicio de sesión y registro de usuarios.

6.1.2. Instalación y despliegue de la plataforma

Como ya se explicó en la Sección 5.9, el sistema se ha desplegado en la nube de Amazon AWS y en una Máquina Virtual ofrecida por la Universidad de Valladolid 5.18.

Base de Datos

Se requiere de la instalación de mysql en la máquina host. Para desplegar la Base de Datos se ha facilitado un script sql, */bsva-backend/utils/DatabaseCreationScript.sql*.

La base de datos escucha en el puerto 3306 (puerto por defecto para mysql).

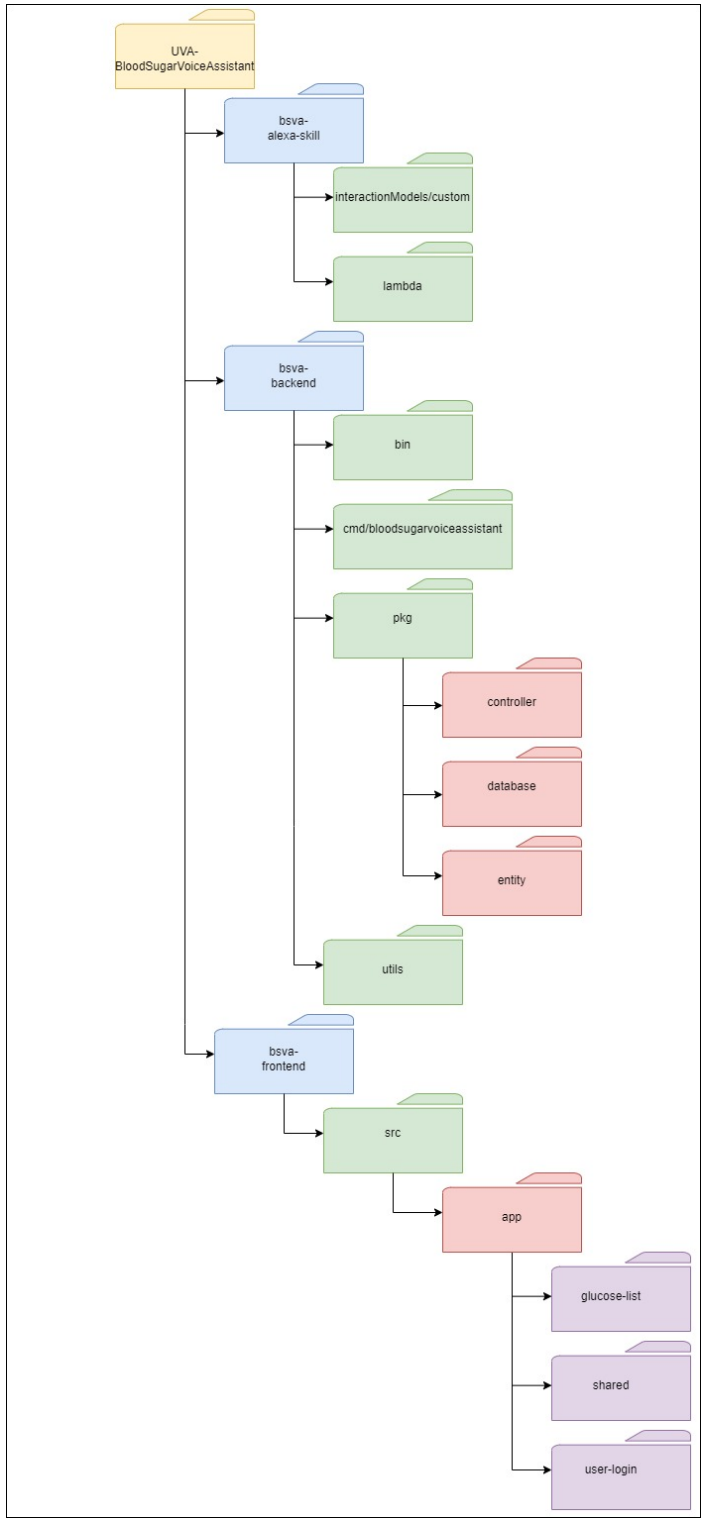


Figura 6.1: Diagrama de directorios de la plataforma Blood Sugar Voice Assistant

API Rest

El servidor que hostea la API Rest se ha desarrollado en Golang, por lo que debemos instalar este lenguaje de programación en la máquina host. Además, se deben instalar las dependencias necesarias para levantar el servidor, que se especifican en el fichero */bsva-backend/go.mod*. Golang se encarga automáticamente de instalar las dependencias especificadas en este fichero.

Para acceder a la API Rest, se despliega el servidor web programado en Golang, mediante la ejecución de su binario */bsva-backend/bin/bsva-rest-api*. Además, se ha redirigido el puerto local 8090 (puerto local del servidor asignado en el código) al puerto exterior 20212, por lo que la url raíz para enviar peticiones http a la API Rest es *www.virtual.lab.inf.uva.es:20212*.

Aplicación Web

Para levantar este componente, necesitamos instalar tanto Node como Angular en la máquina huésped. Para instalar las dependencias del servidor Angular debemos ejecutar el comando *npm install* en la dirección raíz de la aplicación Angular.

Por último, para acceder a la página web de la aplicación, se despliega el servidor mediante el comando *ng serve*, que levanta el servidor de desarrollo de Angular. Se ha redirigido el puerto local 4200 (puerto por defecto de Angular) al puerto exterior 20213, por lo que la url raíz para acceder a la aplicación web de Blood Sugar Voice Assistant es *www.virtual.lab.inf.uva.es:20213*.

Skill Alexa

Para ejecutar la Skill de Alexa no es necesario instalar ninguna dependencia, ya que es Amazon Lambda quién se encarga de su gestión a partir del fichero */bsvaalexaskill/lambda/-requirements.txt*. Para desplegar la Skill en nuestra cuenta de Amazon debemos ir a la web *www.developer.amazon.com/alexa/console/ask* y crear una skill por defecto. Una vez creada la Skill, importaremos el código de la Skill Blood Sugar Voice Assistant tal y como se explica en la documentación de Alexa [8].

Para desplegar la Skill, solo debemos clickar en la pestaña *Deploy* dentro del apartado *Code* de Alexa Developer Console y la Skill podrá ser accedida automáticamente desde todos nuestros dispositivos Alexa. Este procedimiento es ideal para desarrollo y testeo. Sin embargo, para comercializar la Skill deberíamos publicarla en la tienda de Skills de Alexa para que pudiera ser accesible gratuitamente para cualquier usuario de Alexa.

6.2. Pruebas

Como ya se ha comentado en apartados anteriores, la principal dificultad del diseño de la plataforma Blood Sugar Voice Assistant, reside en la integración de los distintos componentes de la misma. Debido a este motivo, se han ido realizando pruebas de forma continua durante el desarrollo del proyecto, comprobando el funcionamiento de los componentes de forma individual. Una vez implementados todos ellos, se procedió a realizar tests de la plataforma en sistema. A continuación se procede a explicar los tests realizados para la plataforma.

6.2.1. Tests unitarios

Estos tests pretenden comprobar el funcionamiento aislado de cada componente del sistema. En el caso de nuestra aplicación, estos tests no han sido automatizados por falta de tiempo y se realizaron para la **API Rest**. El servidor Golang se testeó realizando todas las peticiones http implementadas, 5.6, desde el cliente Postman, comprobando todas las respuestas generadas posibles, de forma similar a los tests de caja blanca.

6.2.2. Tests de integración

Los tests de integración se emplean para comprobar que la comunicación entre distintos componentes se realiza de forma correcta. En nuestro caso, se realizaron tanto para la Aplicación Web como para la Skill Alexa, ya que ambos componentes deben comunicarse con la API Rest de nuestra aplicación.

Los tests de integración de la **Skill Alexa** han sido desarrollados de forma manual, ya que al tratarse de una aplicación por comandos de voz, la forma más sencilla de testear la skill es enunciar dichos comandos. Para ello, se realizaron distintos tests de integración:

- Primero se comprobó que la Skill se abría correctamente mediante la frase de activación y que los diversos intents se activaban con todas las frases de declaración establecidas en el Modelo de la Skill.
- A continuación el desarrollador introdujo datos en la base de datos de forma manual, para comprobar que todas las posibles transiciones entre estados, explicadas en 4.3, se producían adecuadamente al solicitar el último valor de glucosa. Se han aplicado los conocimientos adquiridos en el Grado para diseñar una prueba de secuencia en la que se pasa por todos los estados de la Máquina de Estados y se activan todas las transiciones posibles (árbol de transiciones). Estas pruebas se pueden ver en la tabla 6.1.
- También se testeó la solicitud de permisos de privacidad de la skill, cuando se instala por primera vez en un dispositivo.
- Además, se ejecutó la aplicación desde dos cuentas de Amazon distintas, una registrada en la plataforma Blood Sugar Voice Assistant, y otra no, para comprobar que la aplicación gestionaba correctamente el logeo de usuarios.

- Por último, se tuvo que comprobar que las alertas por hipoglucemia o hiperglucemia se establecían y lanzaban correctamente cuando los valores de glucemia se encontraban en el rango correspondiente.

Con respecto a la **Aplicación Web**, se realizaron **tests de usabilidad**, de forma conjunta a los tests de integración, ya que solicité ayuda a mis compañeros de carrera para obtener sugerencias de mejora de la aplicación web, a la vez que comprobaba que la integración con la API Rest se producía correctamente en las distintas peticiones.

Se realizaron tests de encuesta abierta con 3 usuarios, a los que se pidió que usaran la aplicación web sin recibir indicaciones externas. Al finalizar el testeo se recogió feedback tanto de la interfaz de uso como de la funcionalidad de la plataforma. Por ejemplo, se recomendó cambiar los botones que permiten eliminar datos de la cuenta o hacer logout a esquinas redondeadas o ordenar los valores de glucosa en la tabla con los más recientes al comienzo de la misma.

6.3. Dificultades en la implementación

La principal dificultad de desarrollo de este proyecto consistió en el desarrollo de la Skill Alexa, ya que esto implicó conocer una plataforma nueva, **Alexa Developer Console**, e interactuar con las distintas **APIs de desarrollo de Alexa**, por ejemplo para gestionar la privacidad de la skill o establecer alertas en la misma.

En un primer momento, pese a haber realizado varios tutoriales en el Sprint de preparación, el desarrollador se vió desbordado por la plataforma de desarrollo Alexa Developer Console, ya que debido a un conocimiento mal aplicado, consideraba que el backend de la Skill debía ser hosteado en un servidor propio, por lo que la pestaña *Code* de la plataforma aparecía bloqueada, y el desarrollador no podía añadir lógica a sus intents. Sin embargo, una vez aclarado este concepto gracias a la ayuda de sus tutores, el desarrollador hosteó la lógica de negocio de la Skill en Amazon Lambda (AWS) y procedió a su desarrollo.

Con respecto al uso de APIs de Amazon para el desarrollo de Skills Alexa, la principal dificultad residió en usar la API de creación de temporizadores [6], ya que la documentación oficial viene explicado cómo realizar el proceso en Node, y nuestra Skill está desarrollada en Python. Sin embargo, esta dificultad fue superada tras muchas horas de esfuerzo por parte del desarrollador integrando conocimientos adquiridos en diferentes tutoriales online.

6.4. Licencia empleada

Para este proyecto se ha asignado, desde HP SCDS, una licencia de código **MIT**. Esta licencia es de código libre permisiva, es decir, permite realizar copias del código, modificarlo o distribuirlo, siempre que se acredite la licencia original del repositorio. Esta licencia se puede leer en el fichero *LICENSE* del repositorio.

Estado inicial	**Medida de glucemia**	**Estado final esperado**	**Recomendación**
Hipoglucemia	Mayor o igual que 70 y valor inferior al anterior	Hipoglucemia	Recomendar hidratos absorción rápida
Hipoglucemia	Menor o igual que 70 y valor superior al anterior	Hipoglucemia	Recomendar hidratos absorción lenta
Hipoglucemia	71 - 179	Normoglucemia	Recomendar hidratos absorción lenta
Hipoglucemia	Mayor o igual que 180	Hiperglucemia	Recomendar ejercicio físico
Normoglucemia	0 - 70	Hipoglucemia	Recomendar hidratos absorción rápida
Normoglucemia	71 - 119	Normoglucemia	No recomendar ninguna acción
Normoglucemia	120 - 179	Normoglucemia	Recomendar precaución
Normoglucemia	180 - 249	Hiperglucemia	Recomendar ejercicio físico
Normoglucemia	Mayor o igual que 250	Hiperglucemia	Recomendar insulina
Hiperglucemia	Menor o igual que 70	Hipoglucemia	Recomendar hidratos absorción rápida
Hiperglucemia	71 - 179	Normoglucemia	No recomendar ninguna acción
Hiperglucemia	Mayor o igual que 180 y valor inferior al anterior	Hiperglucemia	Recomendar ejercicio físico
Hiperglucemia	Mayor o igual que 180 y valor superior al anterior	Hiperglucemia	Recomendar insulina y ejercicio físico

Tabla 6.1: Pruebas unitarias de la Máquina de Estados de la Skill

Capítulo 7

Seguimiento del proyecto

En este capítulo se explicará el desarrollo del proyecto realizado para el Trabajo de Fin de Grado. Como ya se ha mencionado, para la realización del mismo se ha empleado la metodología de desarrollo Scrum, por lo que se explicará el desarrollo Sprint a Sprint.

Para realizar el seguimiento del proyecto se ha usado GitLab Issue Tracker. Esta herramienta ha facilitado el proceso ya que nos permite clasificar las tareas. Para describir cada Sprint se proporcionará una tabla con los siguientes apartados:

- **Categoría:** diferencia entre tareas vinculadas a la API Rest (**server**), Aplicación Web (**front**), o a la Skill Alexa (**skill**). En GitLab Issue Tracker se describe en forma de etiquetas (tags o labels) para cada issue.
- **Historia de usuario (U.S.):** historia de usuario vinculada a la issue. Se puede observar la historia de usuario relacionada con cada issue en el título de la misma.
- **Issue:** nombre que describe la tarea realizada. Se corresponde con el título de cada Issue en el Gitlab Issue Tracker. Las Issues se han descrito en inglés ya que es el lenguaje especificado para usar en el Issue Tracker por HP SCDS.
- **Tiempo estimado:** tiempo que se estimó iba a durar la issue en la preparación del Sprint.
- **Tiempo empleado:** tiempo que se empleó realmente durante el Sprint.
- **Estado:** estado de la tarea al finalizar el Sprint. Puede ser open (issue del product backlog), planned (issue a realizar en el Sprint actual), doing (issue que se está realizando en ese momento), blocked (issue bloqueada que debe esperar a la finalización de otra issue) o closed (issue resuelta).

7.1. Sprint 0: 01/01/2022 - 01/02/2022

Como ya se explicó en el Capítulo 2 de Planificación, el primer Sprint se dedicó a que el estudiante se familiarizara con las tecnologías de desarrollo, desconocidas para él, como son Golang y Alexa Developer Console. Para ello se llevaron a cabo diversos videotutoriales que permitieron obtener soltura al desarrollar con estas tecnologías.

En este Sprint también se analizaron los requisitos del proyecto, especificados por HP SCDS en un documento de especificaciones técnicas, que podía ser modificado por el estudiante si todas las partes se encontraban de acuerdo. Fue en esta fase del proyecto cuando el estudiante decidió añadir una pequeña aplicación web a la plataforma Blood Sugar Voice Assistant, que facilitara el registro de usuarios en el sistema.

También se instalaron las herramientas necesarias para el proyecto, como son el lenguaje de programación GoLang o el framework Angular. En una primera fase se decidió que el proyecto se iba a desarrollar en local, en la máquina del estudiante, y el estudiante realizó una API Rest muy sencilla para mostrar en la reunión inicial del Sprint 1

Por último, para preparar la reunión de planificación del Sprint 1, el estudiante preparó una máquina de estados inicial, con todas las transiciones posibles de los niveles de glucosa de un usuario y los tratamientos recomendados, un product backlog inicial, qué fue mejorado en la reunión y un boceto del diagrama de despliegue, mostrando los componentes del sistema propuestos por HP SCDS más el componente Aplicación Web.

En este Sprint no se realizó seguimiento del tiempo empleado ni se realizaron tareas relacionadas con las Historias de Usuario, por lo que no se incluyó en el GitLab Issue Tracker.

7.1.1. Reunión Sprint Review/Retrospective/Planning

Como se explicó en el apartado Adaptación a Scrum 2.2, los tres eventos de Scrum tienen lugar de forma simultánea en nuestro proyecto, los martes cada tres semanas, por lo que en esta reunión se analiza cómo ha transcurrido el Sprint anterior y cuál es el trabajo a desarrollar en el Sprint sucesivo.

Por parte del estudiante, se presentó el Product Backlog inicial, el cuál fue modificado para mejorar las sugerencias en el tratamiento de hipoglucemia e hiperglucemia. Además, se mostraron los bocetos de la máquina de estados y del diagrama de despliegue. Por último, se mostró una pequeña demo de la API Rest “mock” en funcionamiento. El tutor de HP SCDS aprobó la modificación de las especificaciones para añadir la Aplicación Web al proyecto.

Con respecto a la planificación del primer Sprint, se decidió que en estas tres semanas el estudiante mejoraría la API Rest para que gestionara todas las peticiones http que deben ser atendidas por la aplicación.

7.2. Sprint 1: 01/02/2022 - 22/02/2022

El objetivo del primer Sprint es desarrollar la API Rest a partir del servidor "mock" que se implementó en el Sprint de preparación para aprender a programar en GoLang. Para ello se creó la Base de Datos MySQL, se levantó un servidor escuchando en un puerto local y se implementaron los distintos handlers que debe manejar la API Rest para responder a las peticiones http por parte de la Skill como se puede ver en la tabla 7.1

Categoría	**U.S.**	**Issue**	**Tiempo estimado**	**Tiempo empleado**	**Estado**
Server	US1	Create DataBase and connect to DB from Rest API	2h	1h57m	Closed
Server	US1	Create Http Server and GET handler for all users	2h	1h30m	Closed
Server	US2	GET Handler user with id	30m	32m	Closed
Server	US1	POST Handler new user	30m	1h	Closed
Server	US2	DELETE Handler for user with id	30m	1h	Closed
Server	US4	GET Handler last glucose value of user with id	30m	30m	Closed
Server	US4	GET Handler last 7 days of glycemia values of user with id	15m	15m	Closed
Server	US4	POST Handler for a glucose value of user with id	15m	20m	Closed
Server	US4	DELETE Handler of glucose values of user with id	10m	10m	Closed

Tabla 7.1: Sprint Backlog del Sprint 1

7.2.1. Reunión Sprint Review/Retrospective/Planning

En la reunión de fin de Sprint se realizó una demo de la API Rest mediante el uso del cliente Postman para simular peticiones http al servidor. Se comprobó que todas las peticiones funcionaban correctamente y se gestionaban correctamente los códigos de error. El tutor de HP sugirió, como mejora, modificar el código, ya que se había creado un handler para subir valores de glucosa a la base de datos tras leer una petición http. Esto se modificó para que el propio servidor metiera los valores en la base de datos directamente de forma concurrente a responder a las peticiones http. Como ya sabemos, GoLang facilita la concurrencia en servidores, por lo que esta tarea se implementó en el Sprint 2.

Para el siguiente Sprint se decidió dejar como tareas modificar cómo añadir los valores de glucosa a la base de datos, explicado en el párrafo anterior, y documentar el código del servidor de la API Rest.

7.3. Sprint 2: 22/02/2022 - 15/03/2022

En este Sprint, al cuál se dedicaron pocas horas de trabajo, se modificó el código de la API Rest para que añadiera valores de glucosa a la base de datos de forma concurrente al servidor http. Además, se realizó toda la documentación interna del servidor. Estas tareas se pueden ver en la tabla 7.2

Categoría	**U.S.**	**Issue**	**Tiempo estimado**	**Tiempo empleado**	**Estado**
Server	US19	Implement Upload of Random Glucose Values	30m	25m	Closed
Server	US1	Document BSVA Server	2h	3h	Closed

Tabla 7.2: Sprint Backlog del Sprint 2

7.3.1. Reunión Sprint Review/Retrospective/Planning

La reunión de este Sprint fue muy breve. Se mostró cómo se implementó en código la funcionalidad para añadir valores de glucosa en la base de datos cada minuto y se enseñó su funcionamiento en una pequeña demo.

Para el Sprint 3 se decidió realizar la Aplicación Web en Angular, que permitiera el registro-login de usuarios, acceder a sus valores de glucosa de los últimos 7 días y exportar estos valores en un PDF. Además se tomó la decisión de empezar con la memoria del Trabajo de Fin de Grado, realizando en el siguiente Sprint el Capítulo 1: Introducción.

7.4. Sprint 3: 15/03/2022 - 05/04/2022

En este Sprint no se pudo continuar con el desarrollo del proyecto, ya que se manifestó el Riesgo 2, Enfermedad de algún miembro del equipo de trabajo Scrum, tabla 2.2 y el Riesgo 7, Falta de dedicación de tiempo por motivos académicos 2.7. El estudiante no pudo trabajar un fin de semana por enfermedad y otro por eventos relacionados con la empresa donde realizaba sus prácticas de empresa, por lo que tras comunicar lo sucedido a sus tutores se decidió anular el Sprint 3 y continuar con el desarrollo planeado en la reunión final del Sprint 2 en el Sprint 4.

7.5. Sprint 4: 05/04/2022 - 26/04/2022

En este Sprint se realizó el desarrollo de la Aplicación Web, tras la pausa de 3 semanas del Sprint anterior. Para ello se creó una aplicación usando Angular, tomando como modelo

el código desarrollado en la asignatura Diseño Basado en Componentes y Servicios. La funcionalidad que se implementó consiste en el registro y login de usuarios, mostrar los valores de glucosa de los últimos 7 días para un usuario registrado, permitir la eliminación de los datos registrados en una cuenta, eliminar una cuenta completamente, y exportar los valores de glucosa en un PDF. Además, se dedicaron 10 horas a la memoria del proyecto, escribiendo el Capítulo introductorio de la misma.

Cabe destacar también que en este Sprint se clonó el repositorio en una Máquina Virtual proporcionada por la Universidad de Valladolid para facilitar el despligue de la plataforma. Para ello, se solicitó a los técnicos de la Universidad que abrieran dos puertos externos, para poder acceder a la Aplicación Web y la API Rest a través de Internet.

Categoría	**U.S.**	**Issue**	**Tiempo estimado**	**Tiempo empleado**	**Estado**
Server	X	Reorganize project directory structure	30m	30m	Closed
Front	US1	Create the frontend for BSVA	2h	1h45m	Closed
Front	US1	User login in Angular app	2h	8h	Closed
Front	US15	Show glucose values for a registered user	2h	1h30m	Closed
Front	US17	Delete the glucose values stored for a registered user	30m	20m	Closed
Front	US18	Delete a user account	20m	45m	Closed
Front	US16	Export Glucose data in the last 7 days as pdf	30m	1h	Closed
X	X	Ellaborate Project Documentation for thesis	10h	10h	Closed

Tabla 7.3: Sprint Backlog del Sprint 4

7.5.1. Reunión Sprint Review/Retrospective/Planning

En la reunión de final de Sprint se realizó una demo con los tutores, en la que se mostró la funcionalidad añadida en el Sprint explicada anteriormente, y la interconexión entre la Aplicación Web, la API Rest y la Base de Datos. En la demo se crearon distintos usuarios en la web, y se observó cómo se añadían los valores de glucosa automáticamente cada minuto. También se revisó la documentación realizada en el Sprint con la tutora de la Universidad de Valladolid.

El siguiente Sprint se decidió reservar para continuar con la memoria del proyecto, ya que el estudiante se encontraba mucho más adelantado en el código que en la memoria. A mayores, se decidió que se crearía el esqueleto básico de la Skill Alexa, en la forma de una Proof of Concept.

7.6. Sprint 5: 26/04/2022 - 17/05/2022

En la tabla 7.4 se ven las dos tareas realizadas en este Sprint. La mayor parte del esfuerzo de estas tres semanas residió en continuar con la documentación, ya que en este Sprint se dedicaron 30h a finalizar el Capítulo 2: Requisitos y Planificación 2.7. A mayores, el estudiante dedicó un fin de semana a crear una Skill "mock" dentro de Visual Studio Code, usando un plugin que permite su integración. Sin embargo, el estudiante pronto descubrió que se encontraba más cómodo desarrollando la Skill directamente en la plataforma de Alexa Skill Developer, por lo que procedió a desarrollar la Proof of Concept directamente en la plataforma. Para ello se creó la Skill y se generaron la frase de activación y los intents, aunque no se programó la lógica de los mismos.

Categoría	**U.S.**	**Issue**	**Tiempo estimado**	**Tiempo empleado**	**Estado**
X	X	Ellaborate project documentation for thesis	30h	30h	Closed
Skill	XX	Proof of concept: Alexa Skill development and integration inside VsCode	10h	2h	Closed

Tabla 7.4: Sprint Backlog del Sprint 5

7.6.1. Reunión Sprint Review/Retrospective/Planning

En esta reunión el estudiante revisó el Capítulo 2 de la memoria con la tutora de la Universidad de Valladolid. A continuación realizó una demo de la Proof of Concept. Para ello, utilizó la pestaña de tests de la plataforma Alexa Developer Console para demostrar cómo se podía abrir la skill con la frase de activación.

Para el planning del siguiente Sprint, se decidió que el estudiante se centrara en desarrollar los Capítulos 3 y 4 de la memoria, Tecnologías utilizadas y Análisis. Además, para no dejar olvidada la skill, se estableció también que se debería desarrollar funcionalidad en la Skill para solicitar el último valor de glucosa, aunque todavía estamos hablando de un número aleatorio, ya que la skill no es capaz aún de comunicarse con la API Rest.

7.7. Sprint 6: 17/05/2022 - 07/06/2022

Se desarrollaron los Capítulos 3 y 4 de la memoria del proyecto, Tecnologías utilizadas y Análisis.

Además, en este Sprint del desarrollo del proyecto se manifestó el Riesgo 4: Desconocimiento de las tecnologías a emplear en el proyecto 2.4. El estudiante no encontraba la manera

de asignar código a los intents de la aplicación, para programar la lógica de negocio. Esto se debía a que al crear la Skill había seleccionado la opción "servidor propio" para hostear la aplicación, en vez de hosteado en Amazon Lambda, por lo que la pestaña Code aparecía sombreada. Este concepto de hosteo de la Skill fue explicado en la reunión final del Sprint, al informar del problema al tutor de HP SCDS.

Categoría	**U.S.**	**Issue**	**Tiempo estimado**	**Tiempo empleado**	**Estado**
X	X	Ellaborate project documentation for thesis	30h	15h	Closed
Skill	US4	Request last glucose value to Alexa	10h	5h	Doing

Tabla 7.5: Sprint Backlog del Sprint 6

7.7.1. Reunión Sprint Review/Retrospective/Planning

En la revisión de este Sprint, con respecto a la memoria, la tutora de la Universidad de Valladolid recomendó al estudiante que citara la fuente de las imágenes que este había obtenido de internet. Con respecto a la Skill, como ya se ha mencionado el tutor de HP SCDS explicó cómo funcionan los distintos tipos de hosteo de Skills Alexa, para que el estudiante pudiera programar la lógica de los intents en el Sprint 7.

Se decidió que para los Sprints sucesivos el estudiante enfocaría totalmente su tiempo en finalizar la Skill de Alexa, ya que la tutora de la Universidad de Valladolid se iba a encontrar muy ocupada en esas fechas debido a las defensas de los TFGs a final de curso. Por lo tanto, se estableció que para el siguiente Sprint el estudiante desarrollaría todos los intents de la Skill, dejando las alertas y el enlace a la API Rest para los Sprints finales.

7.8. Sprint 7: 07/06/2022 - 28/06/2022

En este Sprint, el estudiante dedicó todo su esfuerzo a añadir la lógica a todos los intents que conforman la Skill: pedir glucosa, ayuda, salir e intent por defecto. Para ello, tuvo que crear una clase Python para cada intent, que actúa como handler y cuyo código es ejecutado al hacer match la sentencia del usuario con la frase de intent de la skill. Las tareas desarrolladas se pueden ver en la tabla 7.6. En las issues relativas a las alertas se desarrolló la lógica de la Skill, pero la generación de las alertas por la Skill se dejó para el Sprint 8 debido a su complejidad. Cabe destacar que en este Sprint se implementó la funcionalidad para gestionar el login de usuarios en la plataforma, ya que para poder pedir su último valor de glucosa estos deben registrarse en la Aplicación Web de Blood Sugar Voice Assistant con el mismo correo electrónico que el vinculado a su cuenta de Amazon.

Categoría	U.S.	Issue	Tiempo estimado	Tiempo empleado	Estado
Skill	US4	Request last glucose value to Alexa	10h	5h	Closed
Skill	US4	Get recommendations depen-dending on glucose value	30m	25m	Closed
Skill	US5	Register 15m alert if user is hypoglucemic	2h	1h	Closed
Skill	US9	Program 30m alert when user is hyperglucemic	1h	30m	Closed
Skill	US6	Program what to do after 15 mins alarm if hypoglucemic goes off	3h	30m	Closed
Skill	US5	Get the Amazon Account email from the Alexa Skill	5h	7h	Closed

Tabla 7.6: Sprint Backlog del Sprint 7

7.8.1. Reunión Sprint Review/Retrospective/Planning

Esta reunión fue únicamente con el tutor de HP SCDS por los motivos explicados en el Sprint anterior. En ella se hizo una demo en la que el estudiante solicitaba el último valor de glucosa, con distintas combinaciones de glucemia, para demostrar cómo la skill emite recomendaciones personalizadas para los distintos valores de glucemia. Además, se mostró cómo funciona el registro en la plataforma.

La evaluación fue positiva y se dejó para el Sprint 8 la funcionalidad restante de la Skill: establecer las alertas y conectar la Skill con la API Rest para obtener los valores de glucosa de la Base de Datos.

7.9. Sprint 8: 28/06/2022 - 19/07/2022

En este Sprint, último Sprint de desarrollo de la plataforma Blood Sugar Voice Assistant, se realizaron solo 2 tareas, pero de gran importancia.

La primera de ellas fue el lanzamiento de alertas similares a un temporizador por la Skill, cuando el usuario se encontraba en hipoglucemia o hiperglucemia. Para implementar esta funcionalidad el estudiante tuvo que leer la documentación de la API de temporizadores de Alexa, enfocada al desarrollo en Node, y adaptarlo a Python. Con la ayuda de diversos tutoriales online fue capaz de solventar esta issue pero con gran dificultad.

La segunda parte consistió en el envío de peticiones http desde la Skill a la API Rest, para gestionar tanto el login de usuarios como para obtener su último valor de glucemia. Esto resulto mucho más fácil que la primera tarea del Sprint, ya que desde Python enviar peticiones http resulta muy sencillo.

Las issues que se incluyen en este Sprint son las de la tabla 7.7.

Categoría	U.S.	Issue	Tiempo estimado	Tiempo empleado	Estado
Skill	US5	Set an Alexa timer	10h	20h	Closed
Skill	X	Connect Alexa SKill to the BS-VA API Rest	5h	10h	Closed

Tabla 7.7: Sprint Backlog del Sprint 8

7.9.1. Reunión Sprint Review/Retrospective/Planning

Para finalizar este Sprint se realizó una demo en vivo con el tutor de HP SCDS, el representante de innovación de HP SCDS en Castilla y León y un médico del Hospital Universitario de León. En ella el estudiante preparó una pequeña presentación PowerPoint para explicar el funcionamiento y la arquitectura general de la plataforma Blood Sugar Voice Assistant. A continuación, se realizó una demo en la que el estudiante creó un usuario, mostró en vivo el contenido de la Base de Datos para ver cómo se añadían los valores de glucosa cada minuto, solicitó su último valor de glucosa a la Skill y mostró cómo las recomendaciones de la Skill varían en función de la evolución temporal de la glucemia. Por último demostró el funcionamiento de las alertas programadas por la Skill y el uso de la Skill en diferentes wearables como smartwatches o un dispositivo móvil.

Se dió por finalizada la parte del proyecto relativa a HP SCDS y lo único que resta es finalizar la memoria del proyecto y depositar el TFG.

7.10. Sprint 9: 19/07/2022 - 09/08/2022 (Refuerzo)

En este Sprint del desarrollo, el estudiante realizó los apartados de Diseño e Implementación de la memoria. Para ello realizó los distintos diagramas UML con la herramienta Astah.

Con respecto a el Diseño se explicó en detalle en la arquitectura de la plataforma, los patrones de diseños empleados y el diseño de los distintos componentes.

Por otro lado, para el apartado de Implementación se explicó el contenido de los distintos directorios del proyecto, las pruebas realizadas y las dificultades encontradas en el desarrollo.

7.10.1. Reunión Sprint Review/Retrospective/Planning

En la reunión de este Sprint se revisó la memoria con la tutora de la Universidad de Valladolid, en la cuál se recomendó el uso de diagramas UML para todos los modelados, ya

que el estudiante había realizado el diagrama de la Máquina de Estados con una herramienta que no era Astah, por lo que la sintaxis no era del todo la correcta. También se sugerió al estudiante que mejorara la documentación de los casos de prueba de la Skill.

7.11. Sprint 10: 09/08/2022 - 30/08/2022 (Refuerzo)

En el último Sprint del proyecto se escribieron los capítulos de Seguimiento, explicando cómo fue el desarrollo del proyecto a lo largo de los Sprints, y Conclusiones, donde se explican cuáles son los requimientos del proyecto que se han cumplido y los que no, además de lo aprendido en el mismo. También se aplicaron las sugerencias emitidas en la reunión del Sprint anterior.

7.11.1. Reunión Sprint Review/Retrospective/Planning

La conclusión final de la reunión del último Sprint fue que había que enviar un correo electrónico a los responsables de HP SCDS para establecer la visibilidad del repositorio como público, ya que se debe permitir el acceso al respositorio a los miembros del tribunal. Este acceso fue proporcionado al día siguiente, por lo que el repositorio con el código del proyecto, el Issue Tracker y la memoria se encuentran actualmente abiertos al público bajo licencia MIT.

7.12. Conclusión de la Planificación Temporal del Proyecto

En la Sección Planificación inicial de los Sprints 2.5, se explica que se preveyó emplear un total de 450 horas al proyecto, reservando 15 horas semanales para la realización del mismo.

Si sumamos todos los tiempos estimados y empleados en las Issues del GitLab Issue Tracker obtenemos que el estudiante estimó un total de 143 horas y 30 minutos para realizar todas las tareas (en este tiempo recordamos que no se incluyen el Sprint 0 de preparación ni los Sprints 9 y 10 de realización de memoria). Por otro lado, el tiempo realmente empleado para resolver las Issues ha sido de casi 129 horas.

Si a las 129 horas empleadas se le suma el trabajo realizado en los Sprints que no han sido trackeados, estimando 60 horas para el Sprint 0 (15 horas semanales durante un mes) y 45h para los dos Sprints finales (cada Sprint de 3 semanas y 15 horas semanales), se puede calcular el tiempo real dedicado al Trabajo de Fin de Carrera en torno a las 279 horas.

Capítulo 8

Conclusiones

8.1. Conclusiones

Una vez expuestos todos los apartados anteriores, resta solo evaluar cómo ha sido el desarrollo de este proyecto y qué ha supuesto en el estudiante.

El proceso de desarrollo de **Blood Sugar Voice Assistant** ha sido un proceso arduo, ya que la realización de este Trabajo de Fin de Grado ha sido compaginada con prácticas de empresa a jornada completa en su total duración. Sin embargo, ha sido una experiencia muy gratificante tanto a nivel de desarrollador como personal, ya que he aprendido de cero el lenguaje de programación Golang y la plataforma de desarrollo Alexa Developer Console, para Skills de Alexa. No descarto en un futuro aplicar los conocimientos obtenidos para desarrollar una skill que me facilite mis tareas del día a día. Por otra parte, a nivel personal, me ha gustado especialmente poder aplicar lo aprendido en el Grado en Ingeniería Informática en mi primer "gran proyecto", por ejemplo he empleado mis conocimientos en diseño de Software, diseño de aplicaciones Web o Bases de Datos.

Considero que se han cumplido los requisitos establecidos en el documento de especificaciones propuesto por HP SCDS. El objetivo final del proyecto era desarrollar un asistente que permitiera a los usuarios interaccionar con los sistemas flash de medición de glucemia mediantes comandos de voz, dejando a libre elección el asistente de voz. Se establecieron tres requisitos:

- Solicitar valor de glucosa mediante comando de voz.
- El asistente devolverá el último valor de glucosa y en función del nivel de glucemia dará recomendaciones.
- El asistente programará alertas que recuerden al usuario medir su glucemia si se encuentra en valores considerados de riesgo.

Todos estos requisitos han sido cumplidos en este Trabajo de Fin de Grado y a mayores el estudiante ha considerado oportuno desarrollar una Aplicación Web para la plataforma, que permite el registro de nuevos usuarios, y el acceso de los mismos a los datos registrados en Blood Sugar Voice Assistant.

8.2. Líneas de trabajo futuras

Para concluir este capítulo, se aporta una serie de propuestas con las que se podría continuar trabajando en el proyecto. Las cuatro primeras sugerencias serían muy sencillas de implementar; todas ellas pertenecen al ámbito de la Aplicación Web, que era una tarea optativa, y no se han desarrollado por falta de tiempo. En total se consideran como propuestas:

- **Recuperación de credenciales de usuario** en la aplicación web: se podría desarrollar un sistema de recuperación de contraseñas para los usuarios de la aplicación utilizando un cliente smtp como Gmail.
- **Encriptación de las credenciales** de los usuarios: se podría implementar un sistema de encriptación/desencriptación de las contraseñas de los usuarios de la aplicación para garantizar una comunicación segura.
- **Encriptación de la sesión** de los usuarios: en esta aplicación se trabaja con datos sensibles, del ámbito sanitario, por lo que sería conveniente que se garantizara un manejo seguro de los mismos.
- Generar un **componente Angular para el registro** de los usuarios: sería más adecuado separar el componente Angular para el registro de usuarios del componente encargado del login de usuarios.
- **No generar nuevas alertas en la Skill cuando ya haya una alerta programada**: en el caso de que un usuario haya tenido una medida de riesgo y la skill haya programado una alerta, sería ideal que no se programaran nuevas alertas si el usuario solicita medir de nuevo su glucemia y vuelve a estar en una zona de riesgo.
- **Integrar una base de datos real** con la API Rest: en este proyecto se ha empleado una Base de Datos con valores de glucemia generados aleatoriamente cada minuto por el servidor para cada usuario. Para el despliegue de la aplicación sería necesario sustituir esta Base de Datos por una con datos auténticos. Para esto sería necesario llegar a un acuerdo con una empresa que venda sistemas de medición flash de glucosa, por ejemplo la empresa Abbot. Sin embargo, esta empresa ha mostrado su rechazo a compartir estos datos por motivos de privacidad cuando el tutor de HP se ha puesto en contacto con ellos.
- **Añadir más información a la Base de Datos que permita generar recomendaciones personalizadas** para cada usuario: sería interesante añadir columnas en la base de datos que faciliten una manera de establecer distintos umbrales de glucemia e incluso distintos tratamientos, ya que cada paciente de diabetes es individual y necesita un tratamiento personalizado. Habría. también, que modificar la lógica de negocio de

la Skill para contemplar esta nueva información y mejorar la asistencia por comandos de voz.

- **Internacionalización:** Sería muy interesante añadir Locales para más regiones e idiomas en la Skill de Alexa, permitiendo así su uso en otros idiomas y facilitando su despliegue a nivel mundial.

Bibliografía

[1] Abbott. Revolutionizing cgm with freestyle libre. https://www.abbott.com/corpnewsroom/diabetes-care/revolutionizing-cgm-with-freestyle-libre.html. Accedido el 25/04/2022.

[2] Angular.io. The modern web developer's platform. https://angular.io/start. Accedido el 30/05/2022.

[3] Astah. Individual licensing options. https://astah.net/pricing/individual/. Accedido el 16/05/2022.

[4] Astah. The power of software modelling. https://astah.net/. Accedido el 23/05/2022.

[5] Atlassian. Gitflow workflow. https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow#:~:text=Gitflow%20is%20a%20legacy%20Git,software%20development%20and%20DevOps%20practices. Accedido el 23/05/2022.

[6] Alexa Developer Documentation. Alexa timers api reference). https://developer.amazon.com/en-US/docs/alexa/smapi/alexa-timers-api-reference.html). Accedido el 04/09/2022.

[7] Alexa Developer Documentation. Create and manage skills in the developer console). https://developer.amazon.com/en-US/docs/alexa/devconsole/about-the-developer-console.html). Accedido el 07/07/2022.

[8] Alexa Developer Documentation. Import an alexa skill from git). https://developer.amazon.com/en-US/docs/alexa/hosted-skills/alexa-hosted-skills-git-import.html). Accedido el 04/09/2022.

[9] Hiren Doshi. How do the 3 scrum roles promote self-organization. https://www.scrum.org/resources/blog/how-do-3-scrum-roles-promote-self-organization. Accedido el 30/05/2022.

[10] Faye Riley PhD. Exploring research: can coronavirus cause diabetes, or make it worse? https://www.diabetes.org.uk/about_us/news/new-worse-cases-coronavirus. Accedido el 25/04/2022.

[11] Finanzas Claras. Impuestos en francia. https://www.finanzasclaras.es/impuestos-francia/. Accedido el 16/05/2022.

[12] Git. What is git. https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F. Accedido el 23/05/2022.

[13] Healthline. Talking diabetes tech: Hey, alexa, what's my blood sugar? https://www.healthline.com/diabetesmine/talking-diabetes-tech-alexa. Accedido el 25/04/2022.

[14] Jasmine Lee. Agile vs. scrum: A detailed comparison (+when to use each). https://www.g2.com/articles/agile-vs-scrum. Accedido el 03/05/2022.

[15] Ken Schwaber, Jeff Sutherland. The 2020 scrum guide. https://scrumguides.org/scrum-guide.html. Accedido el 03/05/2022.

[16] Bureaux Locaux. Espacios de coworking en lille). https://www.bureauxlocaux.com/prix-marche/coworking/lille). Accedido el 20/08/2022.

[17] Medium. Golang - child of a necessity. https://medium.com/csirait-decrypt/golang-child-of-a-necessity-e4c99ee4e8ed. Accedido el 30/05/2022.

[18] Microsoft Azure. Linux virtual machines pricing. https://azure.microsoft.com/en-us/pricing/details/virtual-machines/linux/#pricing. Accedido el 16/05/2022.

[19] Magda Miu. Events in scrum. https://magdamiu.com/2015/12/17/events-in-scrum/. Accedido el 30/05/2022.

[20] Northware. Errores comunes en proyectos de desarrollo de software. https://www.northware.mx/blog/7-errores-comunes-en-proyectos-de-desarrollo-de-software/#:~:text=%C2%BFQu%C3%A9%20es%20un%20riesgo%20en,de%20los%20objetivos%20del%20proyecto. Accedido el 15/05/2022.

[21] Overleaf. Overleaf official logos. https://fr.overleaf.com/for/partners/logos. Accedido el 23/05/2022.

[22] Pouya Saeedi, Inga Petersohn ... Global and regional diabetes prevalence estimates for 2019 and projections for 2030 and 2045: Results from the international diabetes federation diabetes atlas. https://pubmed.ncbi.nlm.nih.gov/31518657/#:~:text=Results%3A%20The%20global%20diabetes%20prevalence,(700%20million)%20by%202045. Accedido el 20/04/2022.

[23] QALovers. Épicas e historias de usuario en proyectos ágiles. https://www.qalovers.com/2018/04/historias-de-usuario.html. Accedido el 16/05/2022.

[24] ScrumManager. Incremento. https://www.scrummanager.net/bok/index.php?title=Incremento. Accedido el 06/05/2022.

[25] StackTrace. Learn the fundamentals of the agile scrum method. https://stacktraceback.com/apprendre-les-fondamentaux-de-la-methode-agile-scrum/. Accedido el 30/05/2022.

[26] Techtarget. Mvvm). https://www.techtarget.com/whatis/definition/Model-View-ViewModel#:~:text=Model%2DView%2DViewModel%20(MVVM)%20is%20a%20software%20design,Ken%20Cooper%20and%20John%20Gossman.). Accedido el 28/08/2022.

[27] Techtarget. What is gitlab. https://www.techtarget.com/whatis/definition/GitLab. Accedido el 23/05/2022.

[28] Wikipedia. Amazon alexa). https://en.wikipedia.org/wiki/Amazon_Alexa). Accedido el 07/07/2022.

[29] Wikipedia. Gitlab. https://commons.wikimedia.org/wiki/File:GitLab_logo.svg. Accedido el 23/05/2022.

[30] Wikipedia. Google meet. https://fr.wikipedia.org/wiki/Google_Meet. Accedido el 30/05/2022.

[31] Wikipedia. History of gmail. https://en.wikipedia.org/wiki/History_of_Gmail. Accedido el 30/05/2022.

[32] Wikipedia. Mysql. https://en.wikipedia.org/wiki/MySQL. Accedido el 07/07/2022.

[33] Wikipedia. Overleaf. https://en.wikipedia.org/wiki/Overleaf. Accedido el 23/05/2022.

[34] Wikipedia. Python (programming language). https://en.wikipedia.org/wiki/Python_(programming_language). Accedido el 07/07/2022.

[35] Wikipedia. Single-page application. https://en.wikipedia.org/wiki/Single-page_application. Accedido el 31/05/2022.

[36] Wikipedia. Telegram (application). https://fr.wikipedia.org/wiki/Telegram_(application). Accedido el 30/05/2022.

[37] Wikipedia. Visual studio code). https://en.wikipedia.org/wiki/Visual_Studio_Code). Accedido el 07/07/2022.

[38] Zoom. Zoom for you. https://zoom.us/. Accedido el 30/05/2022.

Apéndice A

Manuales

A.1. Manual de despliegue e instalación

Los requerimientos y pasos necesarios para levantar la plataforma Blood Sugar Voice Assistant han sido explicados en la Sección 6.1.2.

A.2. Manual de usuario

A continuación se explica el procedimiento que debería realizar el usuario final para utilizar la plataforma Blood Sugar Voice Assisntant, si esta estuviera integrada con un sistema flash de medición de glucosa y publicada en la tienda de skills de Alexa.

Requerimientos iniciales

- El usuario debe poseer un sistema flash de medición de glucosa, por ejemplo FreeStyle Libre de Abbot, que tome medidas de glucemia y las suba a la base de datos de la empresa fabricante.
- El usuario debe tener una cuenta de Amazon. Si no la tuviera, tendría que registrarse en la plataforma.
- El usuario debe ser propietario de un dispositivo compatible con Alexa, por ejemplo smartphones con la aplicación Alexa, Amazon Echo o smartwatches compatibles con el asistente.
- El usuario debe tener acceso a Internet para registrarse en la aplicación Blood Sugar Voice Assistant.

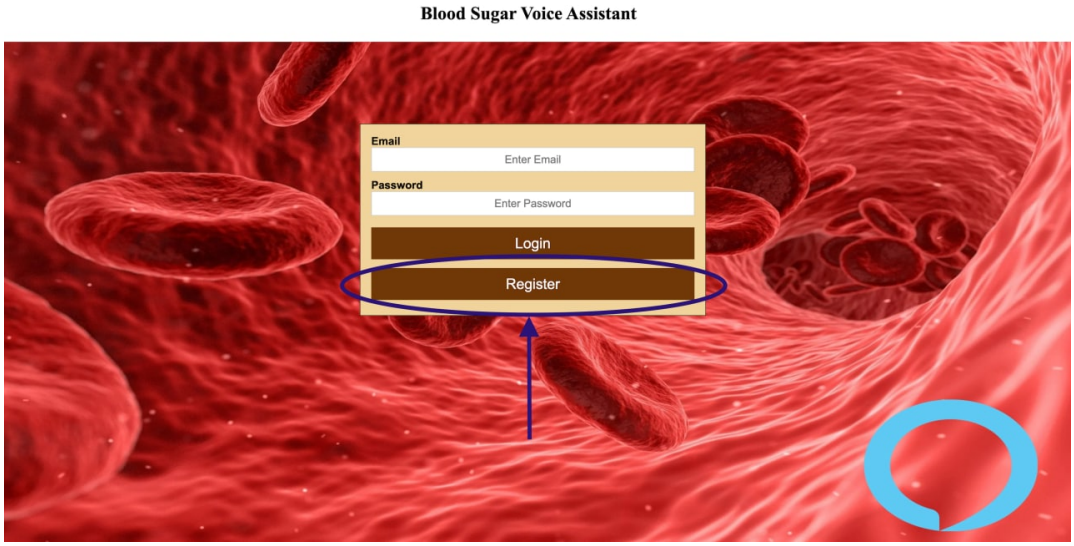


Figura A.1: Página web de registro en la aplicación Blood Sugar Voice Assistant

Uso de la Skill Alexa

- El usuario debe registrarse en la plataforma Blood Sugar Voice Assistance en la url *www.virtual.lab.inf.uva.es:20213/login* con el mismo email asociado a su cuenta de Amazon. Este detalle es muy importante, ya que si se registra con un email diferente la Skill no será capaz de acceder a los valores de glucemia asociados a su cuenta. Marcado con una elipse azul en la figura A.1.

- El usuario debe instalar la Skill de Alexa para la aplicación Blood Sugar Voice Assisn-tan. La instalación de la skill puede ser realizada en la tienda de skills de Alexa (url *https://www.amazon.es/b?ie=UTF8node=13944662031*, figura A.2) o en la aplicación de Alexa para dispositivos móviles.

- Una vez instalada la Skill, el usuario puede activar la aplicación mediante el comando de voz *"Alexa, abre mi asistente de diabetes"* y solitar la última medición de su sistema flash mediante diferentes comandos de voz como *"Glucosa"* o *"Cuál es mi nivel de glucosa"*.

- La skill responderá con el último valor de glucemia registrado para el usuario, enunciará recomendaciones específicas para el nivel medido y programará una alerta si la medida se encuentra dentro de las zonas de riesgo.

- Si el usuario desconoce cómo solicitar su último valor de glucosa, puede enunciar el comando de voz *"Ayuda"* y si desea cerrar la skill de Alexa enunciará *"Salir"*.

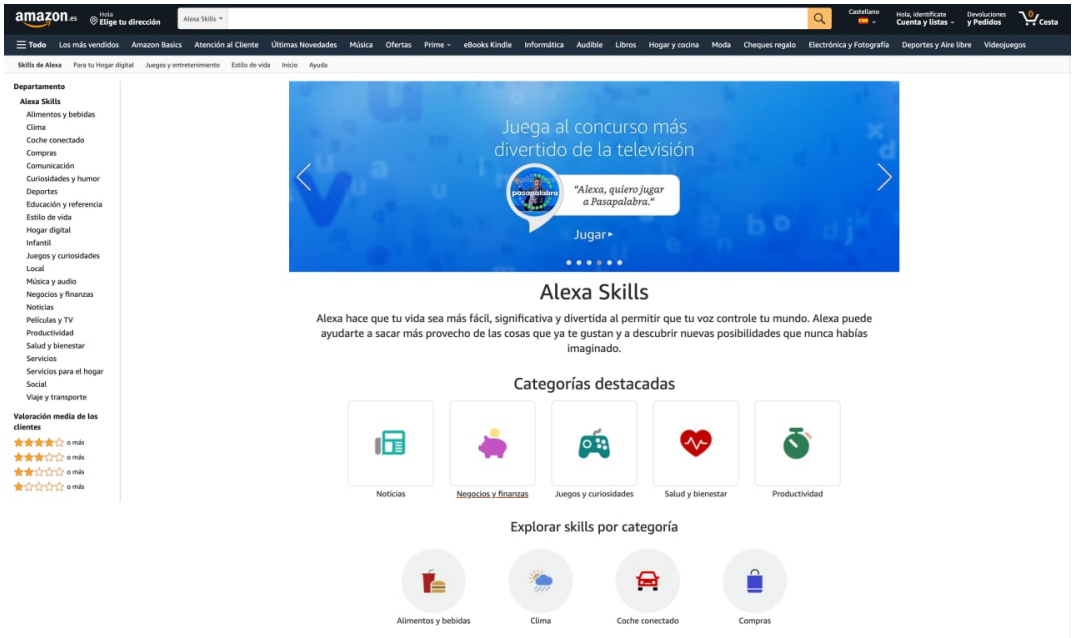
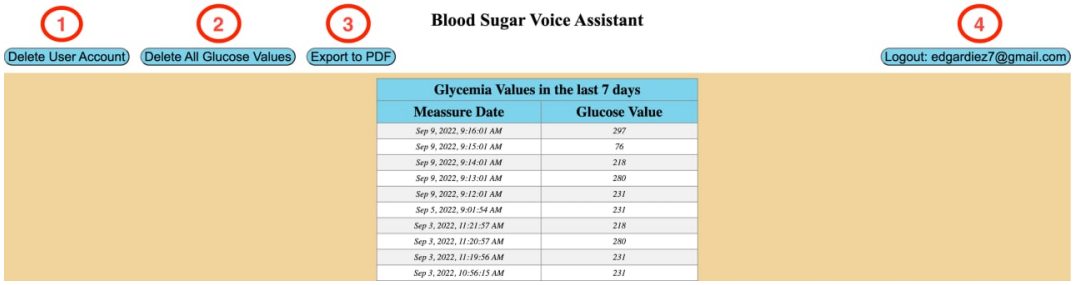


Figura A.2: Tienda de skills de Alexa de Amazon



Glycemia Values in the last 7 days

Meassure Date	Glucose Value
Sep 9, 2022, 9:16:01 AM	297
Sep 9, 2022, 9:15:01 AM	76
Sep 9, 2022, 9:14:01 AM	218
Sep 9, 2022, 9:13:01 AM	280
Sep 9, 2022, 9:12:01 AM	231
Sep 5, 2022, 9:01:54 AM	231
Sep 3, 2022, 11:21:57 AM	218
Sep 3, 2022, 11:20:57 AM	280
Sep 3, 2022, 11:19:56 AM	231
Sep 3, 2022, 10:56:15 AM	231

Figura A.3: Panel de usuario de la página web de Blood Sugar Voice Assistant

Uso de la Aplicación Web

- El usuario, que ya debe estar registrado en la plataforma Blood Sugar Voice Assistant, debe iniciar sesión con sus credenciales en la url *www.virtual.lab.inf.uva.es:20213/login*.
- Una vez iniciada la sesión el usuario puede observar una tabla con sus mediciones de glucemia de los últimos 7 días.
- Si desea exportar esta tabla en un PDF, podrá hacerlo clickando en el botón *"Export to PDF"*, situado en la parte superior de la pantalla. Marcado con un "3" en rojo en la figura A.3.
- Si el usuario quiere eliminar todas las medidas registradas en su cuenta, deberá clickar en el botón *"Delete All Glucose Values"*, situado en la parte superior de la pantalla. Marcado con un "2" en rojo en la figura A.3.
- Si el usuario quiere eliminar su cuenta registrada en la plataforma, y con ella todas sus mediciones de glucemia, deberá clickar en el botón *"Delete User Account"*, situado en la parte superior de la pantalla. Marcado con un "1" en rojo en la figura A.3.
- Por último, si el usuario desea cerrar la sesión en la aplicación web, deberá clickar en el botón *"Logout"*, situado en la parte superior de la pantalla. Marcado con un "4" en rojo en la figura A.3.

Apéndice B

Resumen de enlaces adicionales

Repositorio con el código y la documentación del proyecto:
`https://gitlab.com/HP-SCDS/public/observatorio/uva-bloodsugarvoiceassistant`