MÁSTER EN TECNOLOGÍAS Y APLICACIONES EN INGENIERÍA INFORMÁTICA

TRABAJO FIN DE MÁSTER

UNIVERSIDAD DE ALMERÍA

ESCUELA SUPERIOR DE INGENIERÍA

"Desarrollo y validación de un sistema interactivo de cajero automático utilizando Leap Motion, Java y Angular"

Curso 2020/2021

Alumno/a:
Víctor Manuel Pedraza

Director/es:
José Antonio Piedra

RESUMEN

Actualmente en el proceso de retiro de dinero a través de cajeros automáticos del sector bancario, se realiza interacción física. Esto causa que en repetidas ocasiones personas mal intencionadas intenten realizar fraudes y robos. Además, con la problemática que surgió a raíz del virus Covid19, se trató de evitar el contacto físico con superficies que pudieran ser potenciales focos de contagio. Esto hizo más evidente el hecho de que los teclados de los cajeros automáticos albergan una gran cantidad de parásitos y microbios que pueden causar innumerables problemas.

Por lo anteriormente expuesto, surgió la necesidad de buscar alternativas que permitieran la interacción no física con los cajeros automáticos. Este proyecto plantea el desarrollo de un sistema interactivo (a manos libres) permitiendo realizar las mismas operaciones básicas de cualquier cajero automático mediante leap motion. El backend se desarrolla en Java con Spring Boot y el frontend en Angular. El sistema permite guardar registro de las operaciones de retiro realizadas, así como consultar el saldo de cuentas existentes en una base de datos de MySQL.

En este trabajo se incluyen materiales, métodos y resultados obtenidos de pruebas reales realizadas con el sistema en las ciudades de Madrid y Almería (España). Se probó el sistema en poblaciones de diferentes edades, sexos y orígenes para así obtener una validación del sistema. Para ello, se realizaron encuestas posteriormente tabuladas y discutidas. Finalmente se llegó a concluir que el sistema como producto final es funcional y adecuado para su uso previsto con una aceptación de los usuarios de un 91,3%.

ABSTRACT

Currently the process of cash out through an ATM of a bank sector is with physical interaction. This causes in many situations malicious people try to make frauds and theft. In addition, with the difficulties of Covid 19, people try to avoid physical contact with surfaces which may be possible sources of contagion. This made more important fact that ATM keyboards host a lot of parasites and microbes which may cause countless problems.

In view of foregoing, the need to look for alternatives which letting no-physical interaction with ATMs were clear necessary. This project presents the development of an interactive system (hands-free) letting make the same basic operations of anyone ATM with leap motion. Backend is developed with Spring boot and Fronted with Angular. The system lets to save the register of all withdrawals made, also balance search of available accounts in MySQL database.

This paper contains materials, methods and results obtained in real testing with the system in Cities as Madrid and Almeria (Spain). The system was tested in different populations with different ages, genders and origins in order to obtain the validation of system. For that purpose, surveys were made subsequently tabulated and discussed. Finally, it was concluded that the system as the final product is functional and suitable for the expected use with user's acceptance of 91.3%.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

Actualmente en el proceso de retiro de dinero a través de cajeros automáticos del sector bancario, se realiza mucha interacción física. Esto causa que en repetidas ocasiones personas mal intencionadas intenten realizar fraudes, robos y clonaciones de tarjetas. Algunos de estos son, por ejemplo, clonaciones de tarjetas mediante un Skimmer, dispositivo que realiza una copia de la banda magnética de la tarjeta, y a través de una computadora pasa los datos a una tarjeta vacía; y el robo de contraseñas mediante cámaras puestas en teclado y software de captura de pulsaciones físicas.

Además, con la problemática que surgió a raíz del virus Covid19, con el cual, se trató de evitar el contacto físico con superficies que pudieran ser potenciales focos de contagio, se hizo más evidente el hecho de que los teclados de los cajeros automáticos albergan una gran cantidad de parásitos y microbios que pueden causar innumerables problemas como transmisión de enfermedades, afecciones de la piel e incluso transmitir enfermedades de transmisión sexual; como lo demuestran estudios realizados por el departamento de ciencias biomédicas y ambientales del centro chino de control y prevención de enfermedades en (Gholipour, y otros, 2020) y por el departamento de farmacología y toxicología de la universidad Taibah de Arabia Saudita en (Elbadawy, y otros, 2021).

Teniendo en cuenta los factores mencionados anteriormente, surgió la necesidad de buscar alternativas que permitieran la interacción no física con los cajeros automáticos. Por tanto, se plantea el desarrollo de un sistema interactivo que permita realizar las mismas operaciones básicas que permite uno real tales como retiros y consulta de saldos conectado a un backend desarrollado en Java con Spring framework que permita guardar registro de las operaciones de retiro realizadas, así como consultar el saldo de cuentas existentes en una base de datos de MySQL.

> El diseño de sistemas interactivos se ocupa de muchos diferentes tipos de productos. Se trata de diseñar sistemas de software que se ejecutarán en una computadora en el trabajo. Se trata de diseñar sitios web, juegos, productos interactivos como reproductores MP3, cámaras digitales y aplicaciones para tabletas (computadoras personales). Se trata de diseñar entornos completos en los que teléfonos, tabletas, computadoras portátiles, proyectores digitales y otros dispositivos y servicios se comuniquen entre sí y a través de los cuales las personas interactúen entre sí. Se trata de diseñar sistemas, productos y servicios interactivos para el hogar, el trabajo o para apoyar a las comunidades (Benyon, 2014).

A partir de ello, se realiza una búsqueda de softwares de sistemas interactivos implementados en el sector bancario, que tengan funcionalidades similares a la planteada con el desarrollo de este trabajo de fin de master; encontrándose así, que existen algunos desarrollados por la compañía ultraleap en (Ultraleap, 2021), con la diferencia de que no están conectados a un backend con base de datos.

Para realizar la implementación de sistemas interactivos, se busca conocer quién es el usuario final para centrarse en aspectos tales como la usabilidad, la experiencia de usuario, el diseño para un público objetivo y las interfaces gráficas. De manera que, se centra el desarrollo del sistema interactivo en personas que no tengan discapacidades físicas asumiendo que estas personas se encuentran acompañadas por un tutor a la hora de realizar operaciones de cajero automático y suponiendo, además, que la persona se ha identificado biométricamente y sin contacto ha confirmado la identidad.

Finalmente, una vez desarrollado el sistema interactivo, se realizará su implementación y se ejecutarán las pruebas de usuario que permitirán su posterior validación.

2. OBJETIVOS.

El objetivo del proyecto es el desarrollo y validación de un sistema de interacción de un cajero automático utilizando Leap Motion, Java y Angular.

Como objetivos específicos, se plantean:

- Crear el prototipo en 2D de las pantallas de un cajero automático con Angular.
- Configurar las interacciones Leap Motion con el framework para aplicaciones web Angular.
- Hacer un proyecto backend con Java haciendo uso de Spring donde se expongan las API's necesarias para realizar operaciones.
- Conectar el framework para aplicaciones web Angular a las API's necesarias expuestas por el backend.
- Conectar el backend con una base de datos MySQL.
- Realizar pruebas con usuarios reales del sistema.

3. PLANIFICACIÓN.

El proyecto tendrá una duración de 325 horas, como se aprecia en la Tabla 1, en donde se ubica por casilla la actividad y el número de horas trabajado por semana.

Tabla 1. Tabla de planificación de proyecto

Actividad	Julio				Agosto				Septiembre			
	1	2	3	4	1	2	3	4	1	2	3	5
Investigación Angular												
Búsqueda de fuentes académicas	10											
Familiarización con leap motion y Angular	10	10										
Ampliación de antecedentes	15											
Crear Prototipo en 2D Angular												
Búsqueda de prototipos bancarios		5										
Elección del más reciente de los prototipos bancarios		1										
Analizar que prototipo es el más viable para una interacción con gestos		7										
Diseño de prototipo 2D		15	10									
Implementación de prototipo en Angular		5	15									
Validación de prototipo con ayuda de tutor			1									
Configurar Leap Motion Angular												
Descarga e instalación del software controlador				0,5								
Prueba de funcionamiento del Leap motion				0,5								
Descarga e instalación de TouchFree				0,5								
Pruebas de funcionamiento de Touchfree				0,5								
Crear proyecto Java Backend												
Configurar Java				0,5								
Descargar e instalar Spring STS				0,5								
Descargar e instalar MySQL				0,5								
Análisis de arquitectura del proyecto				4								
Creación de proyecto Backend				10	15							
Creación de base de Datos				2	2							
Pruebas de Backend												
Prueba de funcionamiento de las apis					6							
Refactorización de código					8	5						
Prueba de nuevas funcionalidades					10	6						

Actividad	Julio				Agosto				Septiembre			
	1	2	3	4	1	2	3	4	1	2	3	5
Conectar Angular con Backend												
Investigar como conectar Angular con Spring						8						
Implementación de métodos de conexión de Angular con Spring						12						
Pruebas de funcionamiento							16					
Corrección de posibles bugs al backend							10					
Realizar Pruebas con Usuarios												
Búsqueda de usuarios de prueba								6				
Realización de pruebas								6				
Realización de encuestas de funcionamiento								2				
Análisis de datos obtenidos								10				
Creación de Documento TFM												
Creación de documento TFM		5			10	12	18	2				
Evaluación de avances por el tutor						2	3					
Correcciones de TFM						7	5					
Finalización de creación de TFM								10				
Presentación TFM												
Realización de presentación								3				
Evaluación final del tutor								1				
Correcciones finales									2			
Presentación del TFM									0,2			

La estimación de tiempo planteada en la Tabla 1, sufrió modificaciones durante la realización del proyecto. Inicialmente se había planteado el desarrollo y validación en un total de 325 horas y finalmente llego a ser de 332 horas. Lo cual surgió debido a demoras en la etapa de realización de pruebas con usuarios porque muchos de ellos en principio se negaban a escuchar la idea y en ocasiones a colaborar con el proyecto sin algún tipo de retribución.

Finalmente, teniendo en cuenta que un programador junior en España a 2021 gana 18000 euros al año[1], aproximadamente 9,37 euros la hora y de acuerdo al número de horas invertidas en el proyecto, se puede determinar, que la realización total del

[1] Tomado de https://es.indeed.com/career/programador-junior/salaries

proyecto tiene un costo de 3110 euros sin contar varias casuísticas como transportes para realización de pruebas y recursos materiales.

4. REVISIÓN BIBLIOGRÁFICA

De acuerdo a los objetivos de este trabajo, se realiza una búsqueda de softwares de sistemas interactivos implementados en el sector bancario, que tengan funcionalidades similares a la planteada con el desarrollo del mismo, encontrándose así, que existen algunos desarrollados por la compañía ultraleap en (Ultraleap, 2021).

La aplicación desarrollada por ultraleap se denomina Contactless atm, la cual, permite la interacción del usuario con un cajero automático sin necesidad de tener contacto físico con ningún dispositivo. Su pantalla de inicio se observa en la Ilustración 1. El sistema tiene dos opciones de ingreso realizados por medio de gestos, los cuales son la tarjeta y el móvil, como se observa en la Ilustración 2.

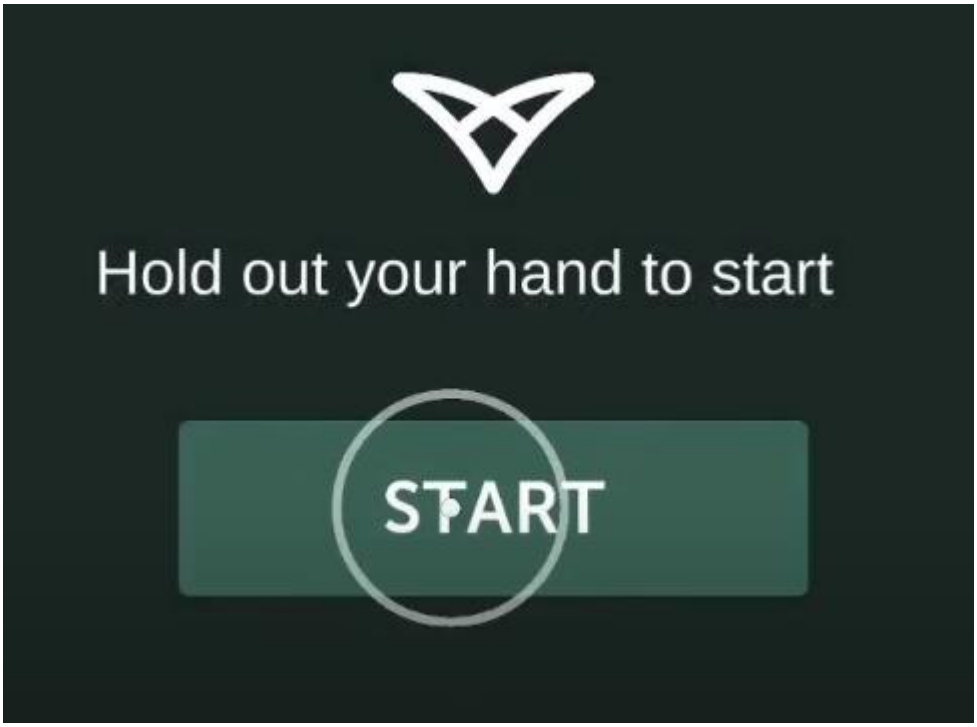

Ilustración 1. Pantalla inicial de Contactless atm

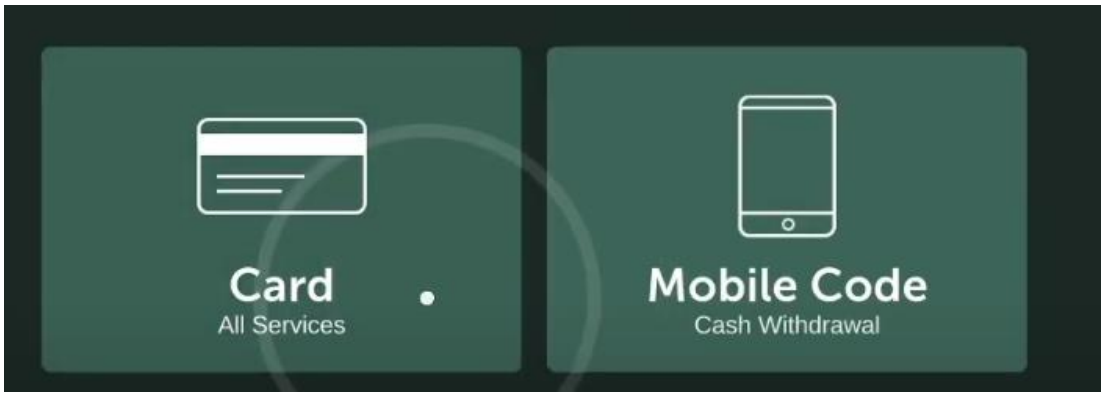

Ilustración 2. Método de ingreso al cajero

Al ingresar, la aplicación pide el ingreso del pin mediante gestos, Ilustración 3, es decir simulando que se está presionando el botón en un teclado.

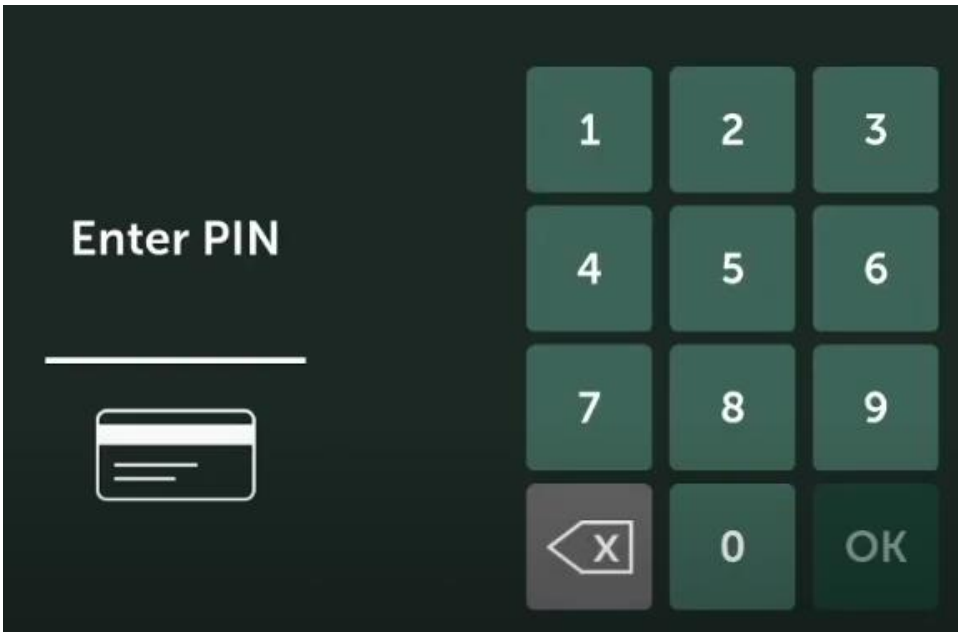

Ilustración 3. Ingreso del pin en Contactless-Atm

En este sistema los usuarios pueden retirar dinero, revisar su saldo y cambiar su contraseña como se observa en la Ilustración 4.

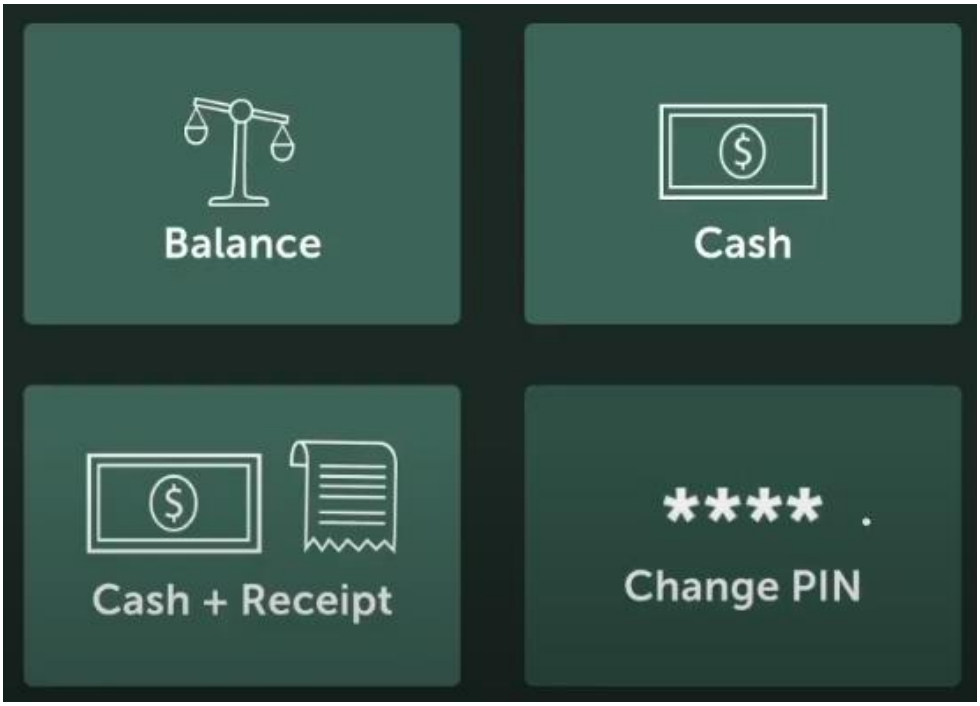

Ilustración 4. Opciones de operación en Contactless-Atm

Comparando la aplicación anteriormente descrita con el proyecto planteado, se encontró la diferencia de que de ultraleap no está conectada a un backend con base de datos, lo cual puede percibirse como una desventaja ya que los datos no se mantienen a través del tiempo sino únicamente en tiempo de ejecución. Además, el sistema de Ultraleap está desarrollado bajo Unity, tecnología que actualmente no es comúnmente aceptada por la mayoría de bancos.

Finalmente, no se encontraron más sistemas similares al planteado con el desarrollo de este proyecto, que permitieran profundizar en su arquitectura o funcionalidad con el fin de tener un punto de comparación.

5. MATERIALES Y MÉTODOS

En esta sección se encuentran explicados los materiales utilizados en la realización del proyecto, así como la arquitectura planteada, el diagrama de casos de usos, las historias de usuarios, la configuración de la base de datos en Google Cloud, una descripción del método seguido en el desarrollo del backend y el frontend, el despliegue en Heroku y por último la configuración del leap motion,

5.1 Materiales

Se describen a continuación tanto el hadware como el software utilizado en el desarrollo del proyecto.

5.1.1 Hardware

Leap Motion Controller

Leap Motion Controller es un dispositivo que se conecta a una computadora y permite al usuario interactuar con el sistema digital mediante movimientos de la mano, lo que permite la interacción simplemente haciendo gestos sobre el dispositivo, sin contacto físico. (Pavaloiu, 2017)

En la Tabla 2 se encuentran las ventajas y desventajas que posee este dispositivo.

Tabla 2. Ventajas y desventajas del leap motion

Ventajas	Desventajas
Permite capturar el movimiento de manos y dedos en su totalidad.	La compleja calibración y montaje que requiere para obtener medidas precisas.

Ventajas	Desventajas
Seguimiento de las manos, dedos, y objetos similares a un dedo situados dentro del espacio de trabajo con buena precisión.	Pérdida de naturalidad en los movimientos de la mano.
Soportado por los sistemas operativos Windows, Macintosh y Linux.	Elevado precio.
El dispositivo cuenta con un SDK muy flexible y con el que se pueden desarrollar aplicaciones muy diversas.	Para la puesta en marcha del dispositivo, es necesario descargar el "LeapDeveloperKit" para el correspondiente sistema operativo.
Al instalar los controladores permite cambiar el área de interacción del dispositivo, mecanismo de ahorro de energía, recalibrar el dispositivo, resolución o informar acerca de problemas, configuración de rastreo, actualizaciones, etc.	Presenta problemas de precisión cuando se expone a altas temperatura, luminosidad y polvo; además con el uso por parte del usuario de accesorios en dedos o muñecas.

Ordenador

Dispositivo electrónico capaz de procesar la información recibida, a través de unos dispositivos de entrada (input), y obtener resultados que serán mostrados haciendo uso de unos dispositivos de salida (output), gracias a la dirección de un programa escrito en el lenguaje de programación adecuado (Vera Rubio, 2016).

5.1.2 IDE

IntelliJ IDEA

Es un poderoso y ergonómico entorno de desarrollo integrado de Java para Enterprise Java, Scala y Kotlin, entre otros. ofrece una experiencia inteligente y ultrarrápida aportando sugerencias relevantes en cada contexto: finalización de código instantánea e inteligente, análisis del código al momento y herramientas de refactorización fiables. Integra herramientas esenciales como sistemas de control de versiones integrados y una amplia variedad de lenguajes compatibles y marcos de trabajo, sin necesidad de complicados complementos. También entiende y ofrece asistencia de codificación inteligente para una gran variedad de otros lenguajes como SQL, JPQL, HTML, JavaScript, etc., incluso si la expresión del lenguaje está inyectada en una cadena literal en su código Java (Jetbrains, s.f.).

5.1.3 Editor de texto

Visual studio code

Es un editor de código fuente ligero pero potente que se ejecuta en el escritorio y está disponible para Windows, macOS y Linux. Viene con soporte incorporado para JavaScript, TypeScript y Node.js y tiene un rico ecosistema de extensiones para otros lenguajes (como C ++, C #, Java, Python, PHP, Go) y tiempos de ejecución (como .NET y Unity) (Microsoft, 2021).

5.1.4 Lenguajes de programación

Typescript

TypeScript es un lenguaje de código abierto que se basa en JavaScript, una de las herramientas más utilizadas del mundo, al agregar definiciones de tipos estáticos. El tipado proporciona una forma de describir la forma de un objeto, proporcionando una mejor documentación y permitiendo que TypeScript valide que su código está funcionando correctamente. Los tipados de escritura pueden ser opcionales en TypeScript, porque la inferencia de tipos le permite obtener mucha potencia sin escribir código adicional (TypeScript, 2021).

Java

Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems. Java es rápido, seguro y fiable. Está diseñado para permitir el desarrollo de aplicaciones portátiles de elevado rendimiento para el más amplio rango de plataformas informáticas posible. (Java, s.f.).

Al contrario de los compiladores tradicionales, que convierten el código fuente en instrucciones a nivel de máquina, el compilador Java traduce el código fuente java en instrucciones que son interpretadas por la máquina virtual de java (JVM, Java Virtual Machine). A diferencia de C y C++ en los que está inspirado. Java es un lenguaje interpretado (Cruz Vílchez, 2012).

5.1.5 Framework

Angular

Angular es una plataforma de desarrollo, construida sobre Mecanografiado. Como plataforma incluye: Un marco basado en componentes para crear aplicaciones web

escalables, una colección de bibliotecas bien integradas que cubren una amplia variedad de características, que incluyen enrutamiento, administración de formularios, comunicación cliente-servidor y más, un conjunto de herramientas para desarrolladores que lo ayudarán a desarrollar, compilar, probar y actualizar su código (Angular, 2021).

Para este proyecto, se utilizó angular versión 9 para el desarrollo Frontend.

Spring boot

Es un framework Java basado en el Modelo Vista Controlador, mediante el cual gracias a los componentes y librerías que brinda hace fácil el desarrollo y despliegue de los servicios REST. Ha eliminado la necesidad de configurar la aplicación con el uso de archivos XML haciendo énfasis en el desarrollo de la misma (Toledano López, Vazquez Sánchez, & López del Castillo, 2018).

Para este proyecto, se utilizó Spring boot versión 2.4.5 para el desarrollo Backend.

Bootstrap

Bootstrap es un framework fornt-end, fue creado por Mark Otto y un grupo de desarrolladores como una solución interna a las inconsistencias y reducir el enorme trabajo que implicaba el mantenimiento de los proyectos, fue publicado en agosto de 2011 como proyecto de acercamiento a dispositivos móviles de código abierto basado en hojas de estilo CSS (Tituaña Maldonado, 2017).

5.1.6 Base de datos

MySQL

En términos generales MySQL es un sistema de gestión de bases de datos relacionales de código abierto. (B., 2020)

El software MySQL ofrece un servidor de base de datos SQL (Structured Query Language) muy rápido, multiproceso, multiusuario y robusto. MySQL Server está diseñado para sistemas de producción de carga pesada y de misión crítica, así como para integrarse en software de implementación masiva (Oracle Corporation, 2021).

En el desarrollo del proyecto se utilizó la versión 5.7.

5.1.7 Entorno de despliegue y BD

Google cloud

Es la infraestructura masiva y en la nube de Google, que opera el tráfico y trabajo de todos los usuarios de Google. Sus servicios están basados, principalmente, en la creación e implementación de aplicaciones y sitios webs con Google. Destaca por su amplia gama de servicios, como la Infraestructura como servicio (IaaS), Plataforma como servicio (PaaS) o el Software como servicio (SaaS). Cuenta con soluciones de Bases de Datos, almacenamiento en la nube y networking (UCloudStore, 2020).

Heroku

Es una plataforma en la nube que permite a las empresas construir, entregar, monitorear y escalar aplicaciones. Se enfoca sin descanso en las aplicaciones y la experiencia del desarrollador en torno a las aplicaciones. Hace que los procesos de implementación, configuración, escalado, ajuste y administración de aplicaciones sean lo más simples y directos posible (Heroku, 2021).

5.1.8 APIs - Rest

Es un conjunto de principios de arquitectura para la construcción de interfaces entre sistemas. El término REST proviene del acrónimo inglés Representational State

Transfer. El éxito de los protocolos REST se basa en cuatro principios clave: Un protocolo cliente/servidor sin estado, una sintaxis uniforme en la red para identificar los recursos, un conjunto de operaciones bien definidas que proporcionan un interfaz uniforme, el uso de hipermedios (hiperenlaces) (Ministerio de industria, energía y turismo de España).

5.1.9 Control de versiones

Git

Git es un sistema de control de versiones distribuido de código abierto y gratuito diseñado para manejar todo, desde proyectos pequeños a muy grandes, con velocidad y eficiencia. La característica de Git que realmente lo distingue de casi todos los demás SCM es su modelo de ramificación. Git permite y anima a tener múltiples sucursales locales que pueden ser completamente independientes entre sí (Chacon & Straub, 2021).

5.1.10 Teoría del color

Grupo de reglas básicas en la mezcla de colores para llegar al efecto deseado combinando colores de luz o pigmentos. También consiste en un grupo de reglas fundamentales sobre la combinación de los colores y sus efectos; el color es una sensación producida por el reflejo de la luz en la materia y trasmitida por el ojo al cerebro, la materia capta las longitudes de onda que componen la luz excepto las que corresponden al color que observamos y que son reflejadas (Cardona, Torres, Peña, Galeano, & Ardila, 2014).

Los colores para el aplicativo de este trabajo se escogieron basándose en que según (Parramón) disponiendo una superficie de color claro y otra de color oscuro o negro y pintando encima de ambas un mismo color, un limón, por ejemplo, se obtiene por ley

de contraste simultaneo la impresión óptica de que el amarillo del limón o un tomate rojo puestos sobre una superficie blanca son más oscuros que los mismos colores cuando están puestos sobre un fondo negro, como se observa en la Ilustración 5.

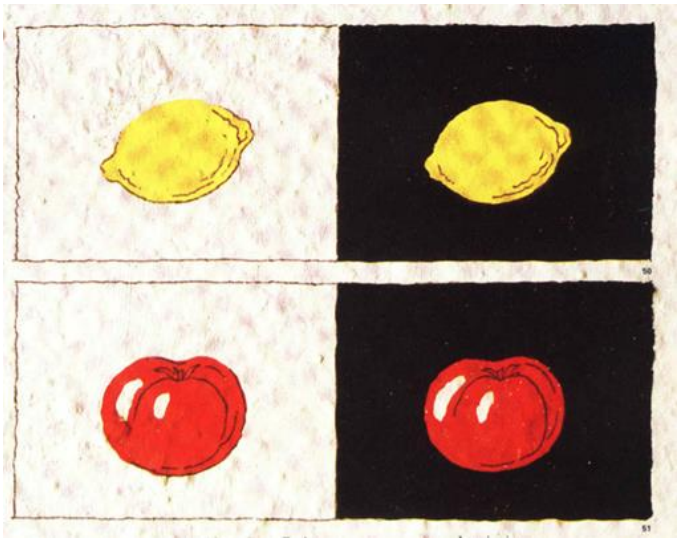

Ilustración 5. Ejemplo ley de contraste simultáneo

De acuerdo a la ley de contraste simultaneo el autor saca dos conclusiones:

- Un color cualquiera resulta más oscuro cuanto más claro es el color que lo rodea.
- El mismo color resulta más claro cuanto más oscuro es el color que lo rodea.

Por otro lado, el famoso físico de los colores Chevreul tiene una guía de colores para lograr contrastes máximos de color, como se muestra en la Ilustración 6.

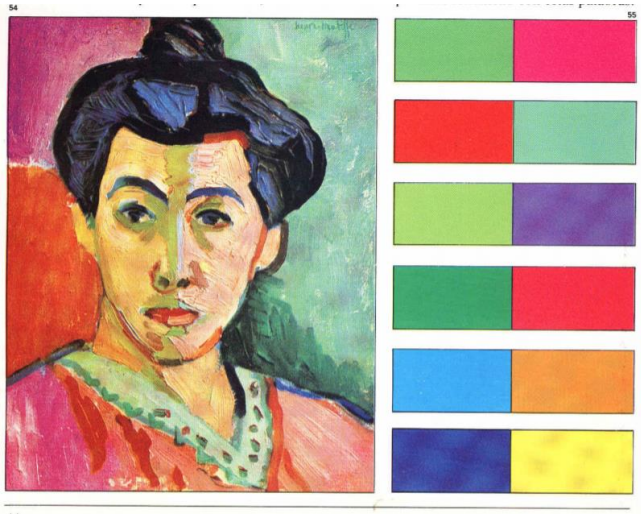

Ilustración 6. Guía de colores para lograr contrastes máximos de color

A continuación, se explica el significado de los colores utilizados en la interfaz gráfica de la aplicación:

- El color blanco es un color positivo, espiritual y lleno de vida. Es el color más puro y el que refleja frescura y calma. Este es el color que representa la limpieza y la perfección. El blanco atrae por su simplicidad, por su pureza y por su luminosidad.

El blanco combinado con amarillo, naranja o azul es enérgico y juvenil,

- El amarillo es un color enérgico que hace sentir bien e inspira el pensamiento original y la creatividad. Este es un color que repercute en el hemisferio cerebral izquierdo, estimula el trabajo intelectual, ayuda a hacer conexiones de ideas, elimina los bloqueos creativos, promueve el optimismo y la concentración.

- El color azul libera la ansiedad y el temor, desarrolla las habilidades mentales y ayuda a tomar decisiones.

- El color gris puro es el único color del espectro que no tiene propiedades psicológicas, sin embargo, algunos tonos de gris pueden generar una sensación

de opresión. El gris es un color moderado que se asocia a la calma y al autocontrol. Es el color de los ambientes zen; refleja espiritualidad, austeridad y sobriedad.

Para generar un espacio de calma y confianza, el gris puede combinarse con blanco y otros tonos neutros, sin olvidar los acentos de color para darle al espacio un poco de dinamismo.

5.1.11 Sistemas interactivos

Interfaz persona-computador compleja que utiliza simulación en tiempo real y canales multisensoriales de interacción. Estas modalidades sensoriales son visual, auditiva, táctil, olfativa y gustativa. (Burdea & Coiffet, 2003).

La dimensión de interacción indica el realismo con el que el sistema reacciona ante las acciones del usuario (Lozano Rodero, 2009). Los principales objetivos en la implementación de sistemas interactivos es saber quién es el usuario final para enfocarnos en diferentes aspectos como usabilidad, experiencia de usuario, interfaces gráficas y diseño principalmente para un público objetivo.

5.2 Métodos

En cuanto a metodología de desarrollo se opta por una metodología ágil Scrum adaptada. Scrum es un marco ligero que ayuda a las personas, equipos y organizaciones a generar valor a través de soluciones adaptables para problemas complejos. (Schwaber & Sutherland, 2020)

5.2.1 Arquitectura planteada

La arquitectura planteada, Ilustración 7, se describe a continuación.

Cada vez que se hace un push a GitHub desde la máquina local, Heroku actúa como plataforma como servicio, tomando los cambios en los repositorios para así desplegar las aplicaciones en su correspondiente sitio, Spring boot expone diferentes endpoints los cuales son consumidos por el front en Angular y este a su vez es capaz de mandar información al back, Spring boot se comunica con una base de datos MySQL, la cual, está alojada en Google Cloud.

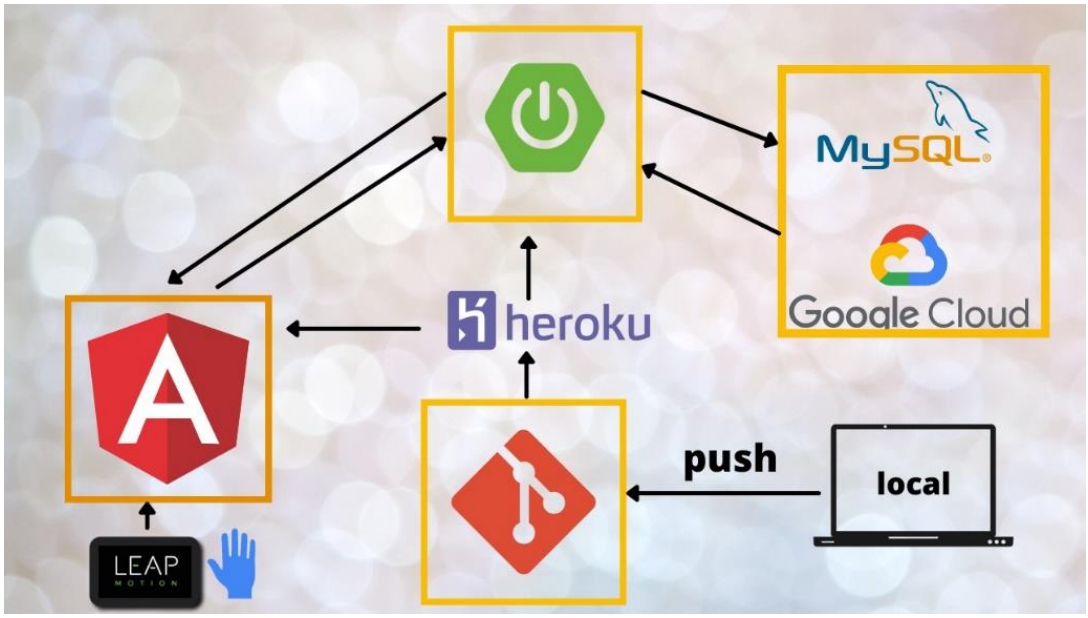

Ilustración 7. Arquitectura plantea

5.2.2 Diagrama de Casos de uso

Para el desarrollo de este proyecto se utiliza una metodología ágil, Ilustración 8, y no una metodología en cascada, por lo cual, no se aborda en este apartado. Sin embargo, en el apartado 5.2.3 se encuentran las historias de usuario escritas como usuario final, dando así una explicación detallada de lo que puede llegar a hacer en el sistema.

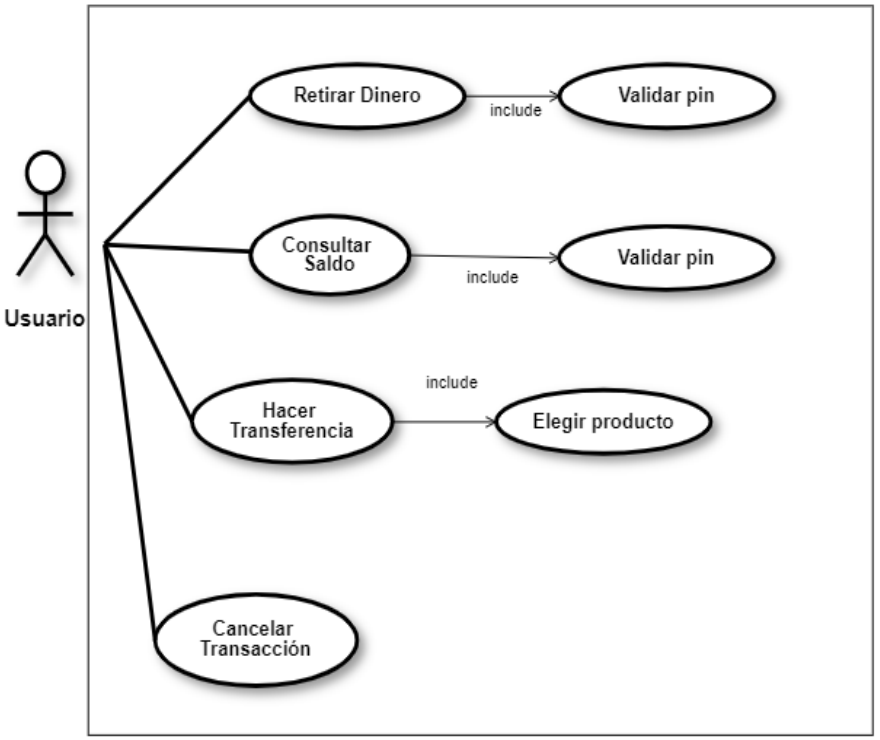

Ilustración 8. Diagrama de Casos de uso

5.2.3 Historias de usuario

Se cuenta con un tablero tipo scrum con diferentes historias de usuario, las cuales están concentradas en el backlog; hay una etapa de TODO en forma de Spring de una semana, así como la correspondiente etapa de DOING, que hace referencia a las historias de usuario que se están realizando en el sprint, la etapa de QA hace parte de la etapa de pruebas y DONE son las historias que se han terminado por completo.

El tablero de tareas se puede observar en la Ilustración 9 y su correspondiente enlace al tablero para verlo más a detalle en trello[2].

[2] https://trello.com/invite/b/FsYKxGYN/543f7dc8b3b685ce20b4d74bcc2b889d/tfm

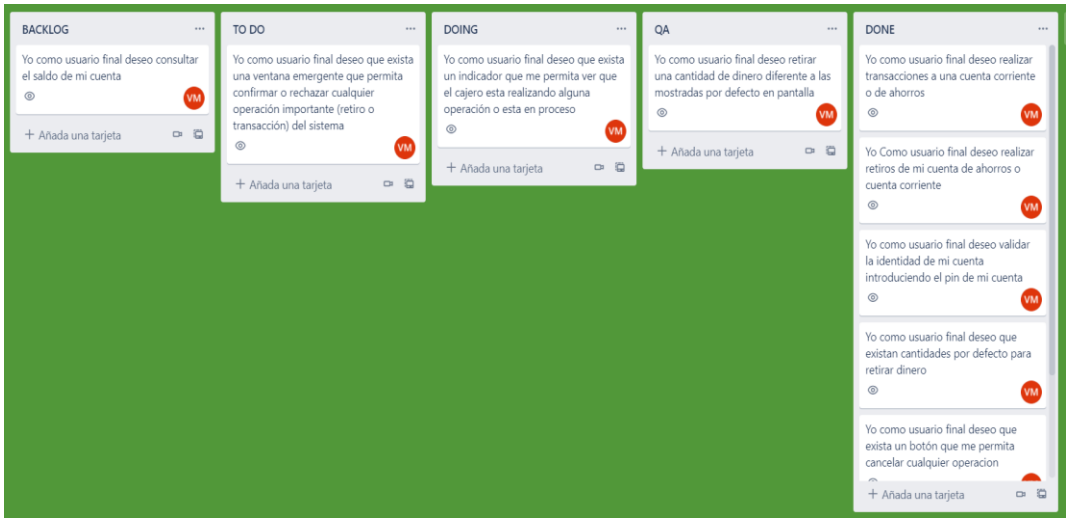

Ilustración 9. Tablero de tareas

5.2.4 Configurar base de datos en Google cloud

Primero se crearon instancias utilizando Cloud SQL, Ilustración 10.

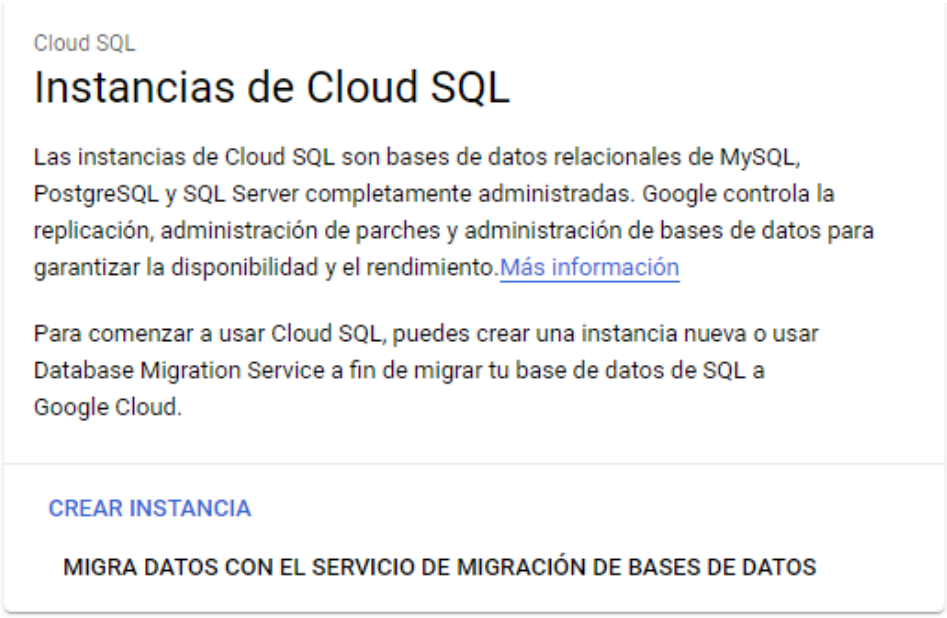

Ilustración 10. Creación de instancias cloud

Para ello, se selecciona MySQL como se observa en la Ilustración 11.

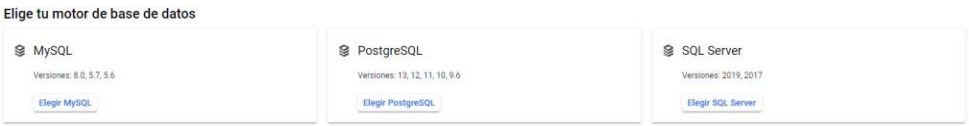

Ilustración 11. Selección de motor de BD en Cloud SQL

A continuación, se configura la instancia como se observa en Ilustración 12.

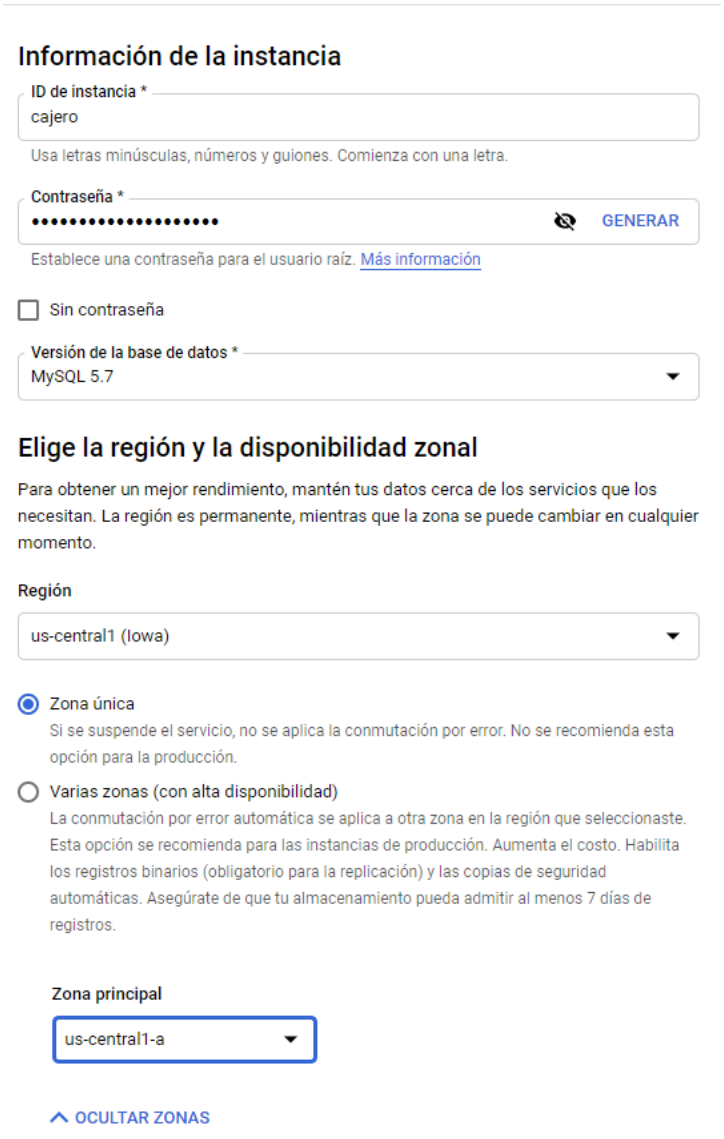

Ilustración 12. Información de la instancia de google cloud

Seguido a esto, se crea la base de datos, dando clic en Crear Base de datos, Ilustración 13.

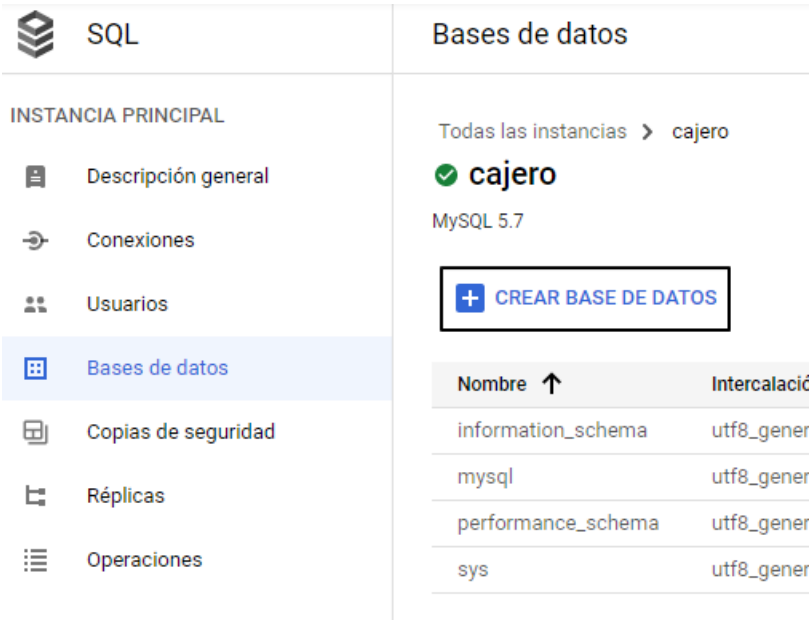

Ilustración 13. Creación de la base de datos en la instancia creada

Para ello, se hace uso de la configuración de la Ilustración 14.

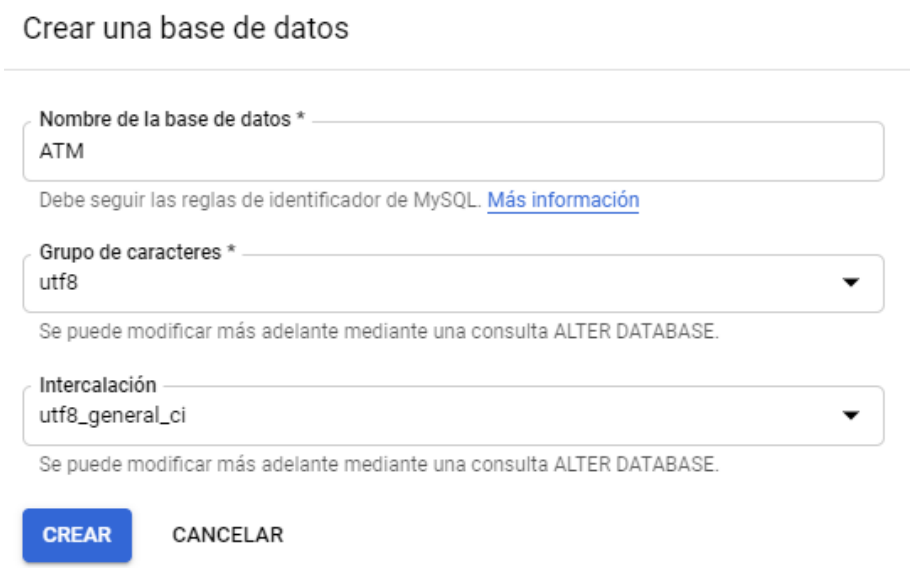

Ilustración 14. Información de la creación de base de datos

Se crea un usuario diferente a root para conectarse a la base datos, Ilustración 15.

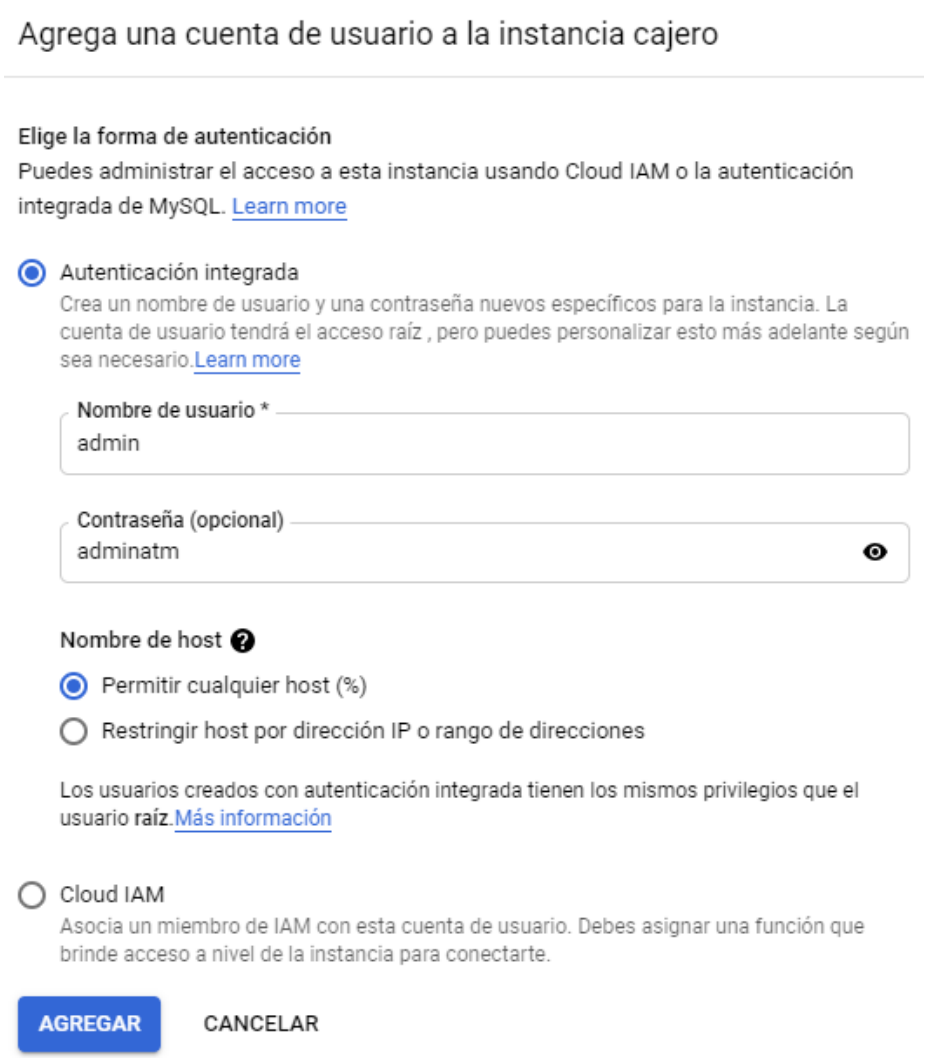

Ilustración 15. Creación de usuario para base datos

Para poder posteriormente conectar MySQL Workbench desde un equipo local se debe autorizar la red con la cual se accede, como se muestra en la Ilustración 16.

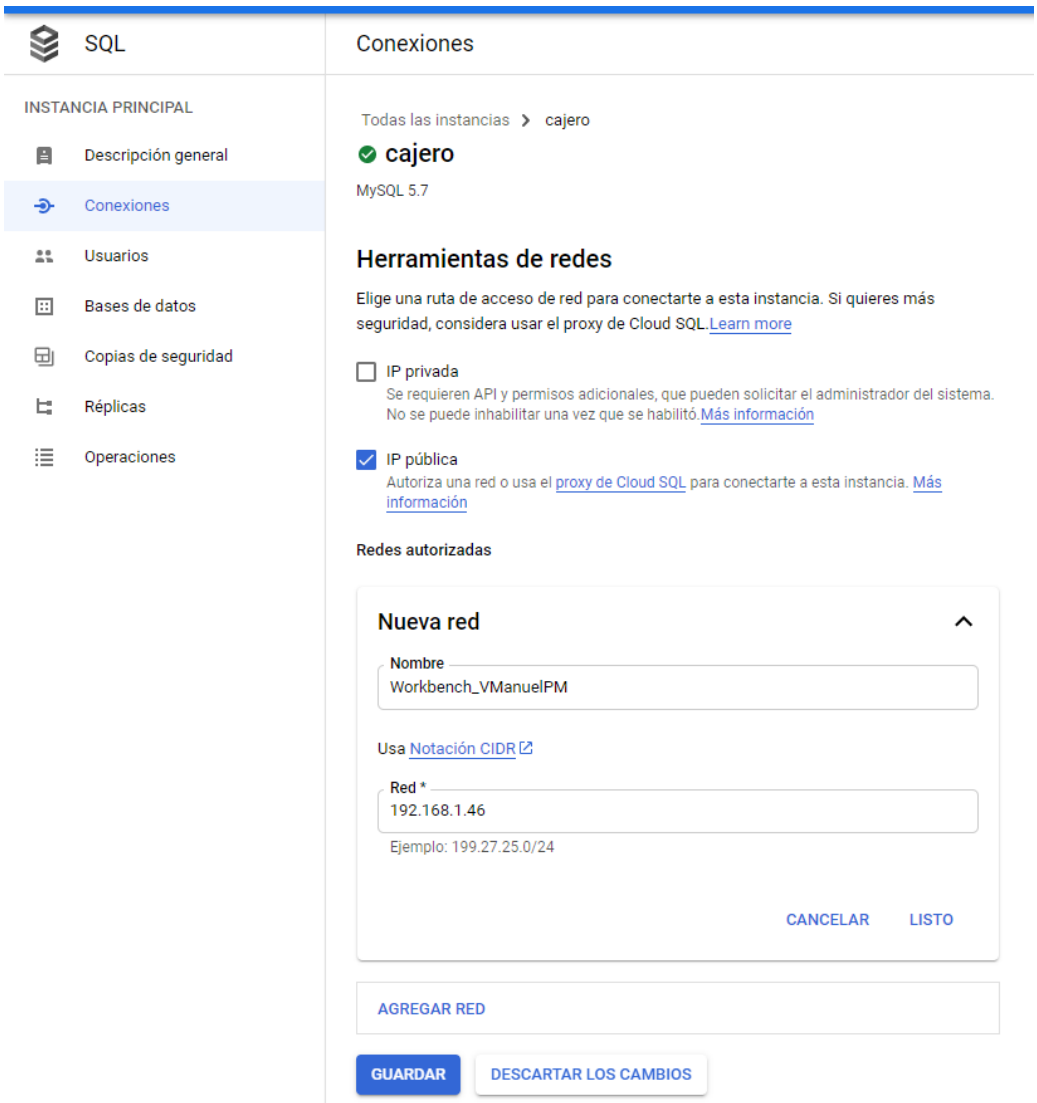

Ilustración 16. Permitir conexiones a instancia de SQL

Conexión a la base de datos usando MySQL Workbench

Una vez se abre MySQL Workbench se crea una nueva conexión a la base de datos que tenemos en Google Cloud, Ilustración 17.

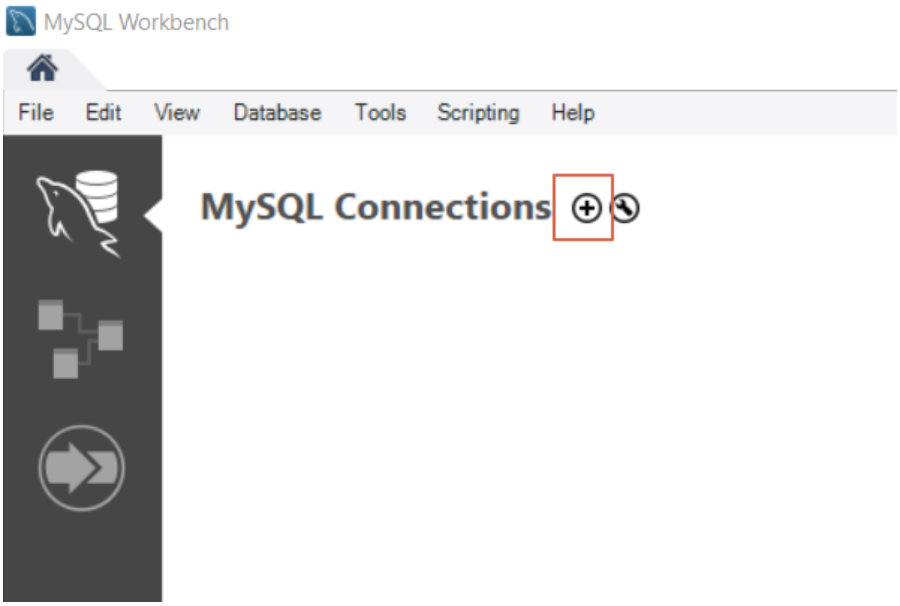

Ilustración 17. Creación de nueva conexión Workbench

Seguido a ello, se colocan las credenciales que proporciona la instancia de Google Cloud y se prueba la conexión, apareciendo el mensaje que se muestra en la Ilustración 18.

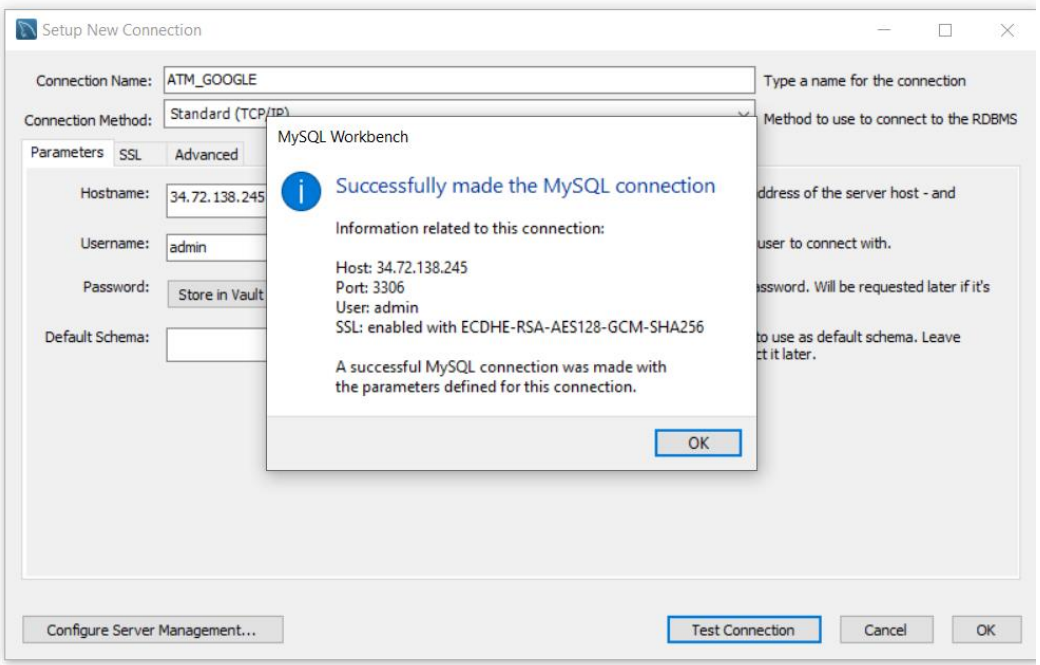

Ilustración 18. Conexión exitosa de Workbench a Instancia GCloud

Una vez se abre la conexión a Google cloud se observa en la pestaña de schemas las bases de datos que se tienen disponibles, Ilustración 19.

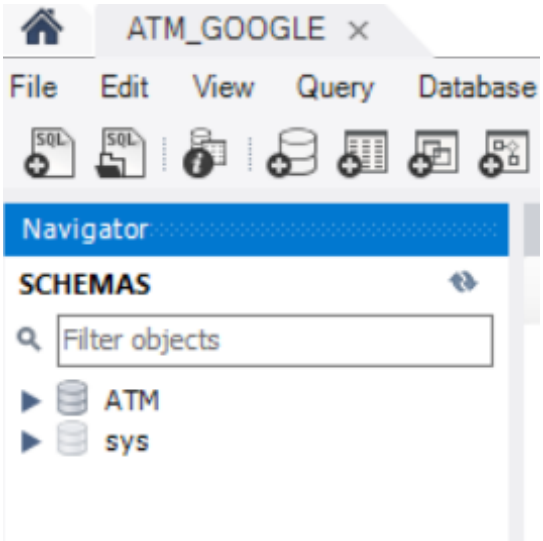

Ilustración 19. Bases de datos disponibles en instancia GCloud

Se cuenta con dos tablas, la primera tabla es para las cuentas disponibles y la segunda para guardar el historial de todas las transacciones realizadas como se observa en la Ilustración 20.

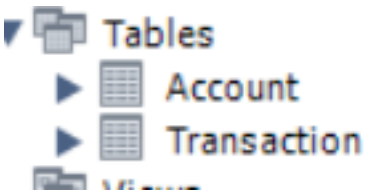

Ilustración 20 Tablas de base de datos ATM

5.2.5 Proyecto Java con Spring Boot

Para el proyecto de backend se utiliza el Framework de Java Spring Boot.

Para la creación del proyecto lo primero que se hace es ir a la página https://start.spring.io/ donde se crea un proyecto como se observa en la Ilustración 21.

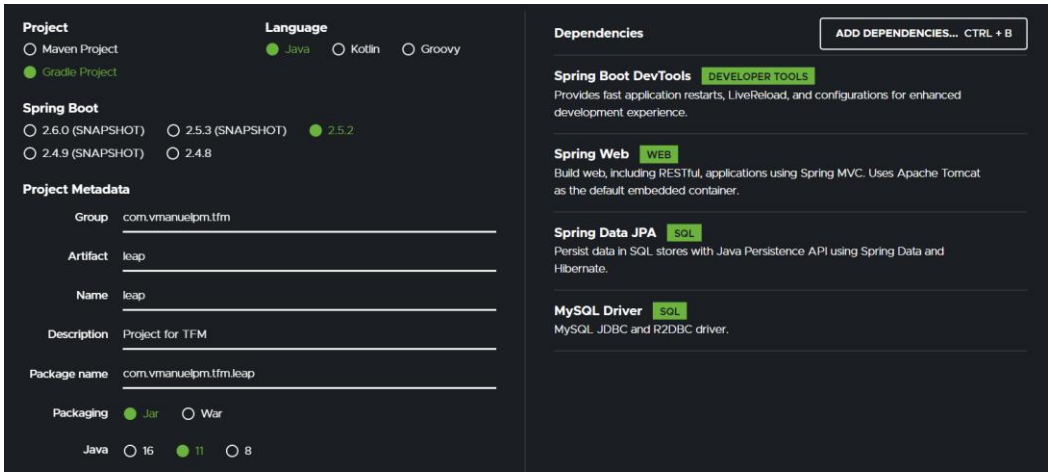

Ilustración 21. Creación de proyecto basico Spring boot

Las dependencias que se utilizan son las siguientes:

- Spring Web: Incluye las configuraciones necesarias para que el proyecto tenga un Tomcat que actué como servidor.
- MySQL Driver: Driver de conexión que permite entablar comunicación con la base de datos.
- Spring Data JPA: Dependencia que incluye las configuraciones para hacer la implementación ORM explicado anteriormente.
- Spring Boot DevTools: provee un servidor que está pendiente de cambios una vez este arriba el servidor, además de reinicios de servidor más rápido.

Se utiliza gradle como gestor de dependencias, por lo tanto al abrir el proyecto en cualquier IDE se tiene el archivo build.grade, Ilustración 22.

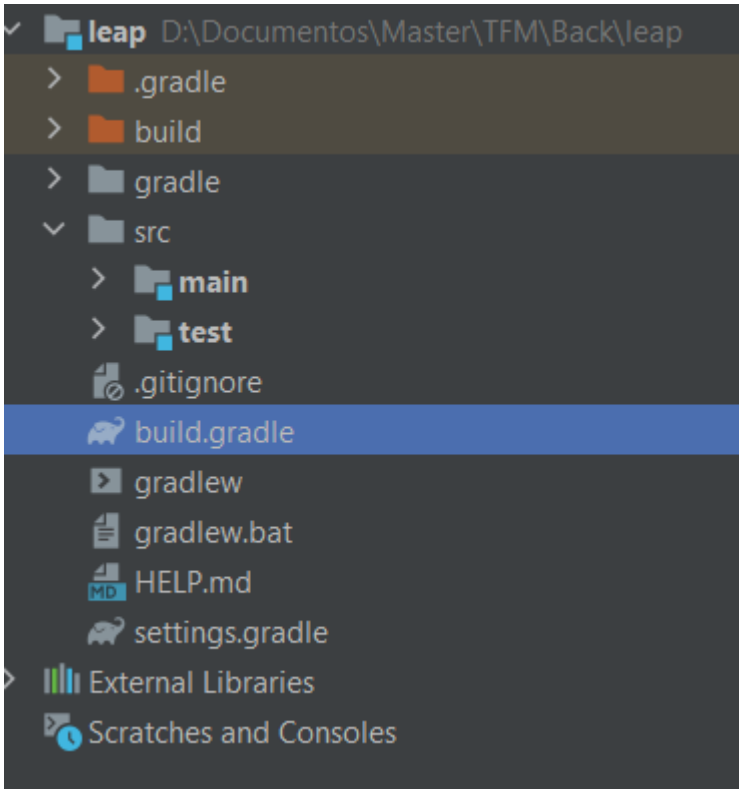

Ilustración 22. Archivo de dependencias

Principalmente se utilizan las dependencias mostradas en la Ilustración 23, las cuales se encuentran en build.gradle.

```
implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
implementation 'org.springframework.boot:spring-boot-starter-web'
implementation 'org.apache.commons:commons-lang3:3.6'
implementation 'org.springframework.boot:spring-boot-starter-validation:2.3.1.RELEASE'
implementation 'org.hibernate.validator:hibernate-validator:6.1.2.Final'
```

Ilustración 23. Dependencias utilizadas en proyecto de Spring Boot

- Apache Commons Lang: proporciona clases para manipular métodos String, métodos numéricos, reflexión de objetos, concurrencia, creación y serialización
- Hibernate Validator: Validaciones que utilizamos para verificar condiciones en algún campo especifico que requerimos persistir, por ejemplo, es muy común usar las etiquetas @NotBlank o @Min, con esta dependencia deja usar los decoradores mencionados.

5.2.6 Conexión a base de datos y Configuración de application.properties

La conexión a MySQL se hace con spring.datasource.url utilizando JDBC donde se utiliza el driver de conexión como se muestra en la Ilustración 24.

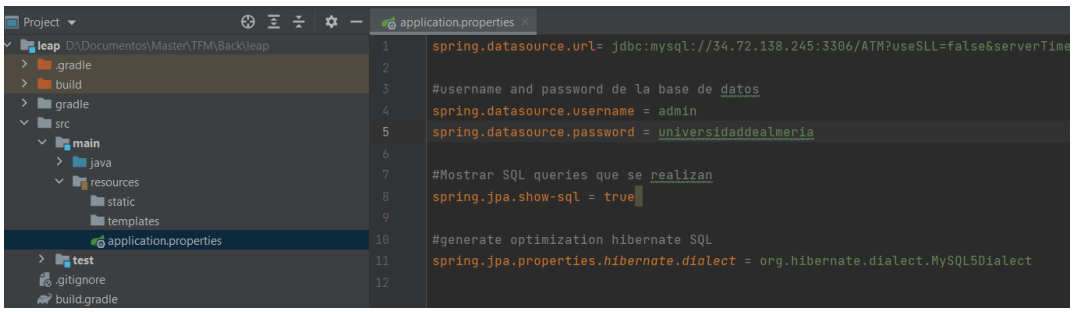

Ilustración 24. Archivo properties de configuración de proyecto

5.2.7 Estructura del proyecto

La estructura del proyecto se muestra en la Ilustración 25.

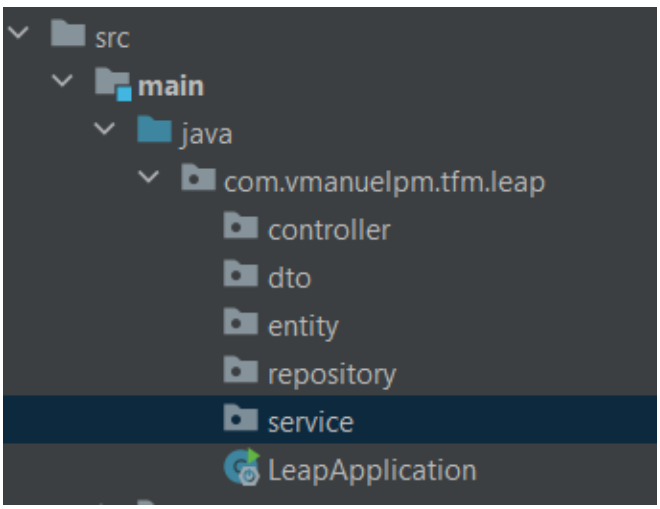

Ilustración 25. Estructura del proyecto

Se cuenta con los siguientes paquetes:

- Entity: Paquete donde se crea la clase que representan la tabla de la base de datos.
- Repository: Paquete en el cual se crea la clase que tiene como particularidad que extiende de JpaRepository el cual nos permite gestionar las operaciones CRUD fundamentales.
- Service: Paquete en donde se crea la clase que tiene como fin hacer la implementación de los métodos que se definan para la aplicación.
- Dto: Paquete donde se crea la clase que se limita a ser un objeto de transferencia entre el cliente y el servidor, recordemos el principio de responsabilidad única, donde la idea es que la entidad como únicamente el modelo de la tabla de la base de datos.
- Controller: Paquete donde se crea la clase que actúa como controlador Rest, es decir exponer las Apis que se definan.

5.2.8 Clases e Interfaces Java

Paquete de Entidades

Se realizan tres entidades las cuales corresponden a las respectivas tablas en base de datos, Ilustración 26. La única entidad que no se encuentra en base de datos es la entidad Audit, y es la entidad que va a permitir hacer la auditoria de la creación y modificación de nuevos campos. Cada entidad cuenta con su correspondiente construtor, getters y setters.

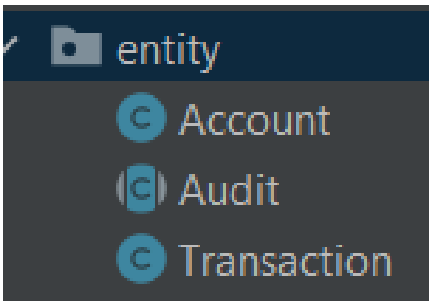

Ilustración 26. Entidades de proyecto en Spring Boot

A continuación, se muestra una parte de como luce la entidad Account, Ilustración 27.

```java
public class Account extends Audit implements Serializable {
    //Llave primaria de la tabla
    @Id
    //Con esto se especifica que el campo es un ID automatico
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_account")
    private int idAccount;
    @Column(name = "account_number")
    private String accountNumber;
    @Column(name = "balance")
    private Float balance;
    @Column(name = "password")
    private String password;

    public Account() {
    }
```

Ilustración 27. Entidad Account

La entidad de Transaction contiene una relación con la tabla Account por lo cual, la entidad luce de la manera mostrada en la Ilustración 28.

```java
@Entity
@Table(name = "Transaction")
public class Transaction extends Audit implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id_transaction")
    private int idTransaction;
    @Column(name = "transaction_value")
    private float transactionValue;
    @Column(name = "transaction_status")
    private String transactionStatus;
    @ManyToOne(fetch = FetchType.LAZY, optional = false)
    @JoinColumn(name = "fk_account")
    @JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})
    private Account fkAccount;

    public Transaction() {
    }
```

Ilustración 28. Entidad Transaction

Y para los campos de Auditoria en la entidad Audit se utilizan varias anotaciones que facilita Spring Boot, tal y como se muestra en la Ilustración 29.

```java
@MappedSuperclass
@EntityListeners(AuditingEntityListener.class)
@JsonIgnoreProperties(
        value = {"createdAt", "updatedAt"},
        allowGetters = true
)
public abstract class Audit implements Serializable {

    @Temporal(TemporalType.TIMESTAMP)
    @Column(name = "created_at", nullable = false, updatable = false)
    @CreatedDate
    private Date createdAt;

    @Temporal(TemporalType.TIMESTAMP)
    @Column(name = "updated_at", nullable = false)
    @LastModifiedDate
    private Date updatedAt;

    public Date getCreatedAt() { return createdAt; }

    public void setCreatedAt(Date createdAt) { this.createdAt = createdAt; }

    public Date getUpdatedAt() { return updatedAt; }

    public void setUpdatedAt(Date updatedAt) { this.updatedAt = updatedAt; }
}
```

Ilustración 29. Entidad Audit

Paquete de Repositorio

La interfaz del paquete repository del proyecto, se muestra en la Ilustración 30.

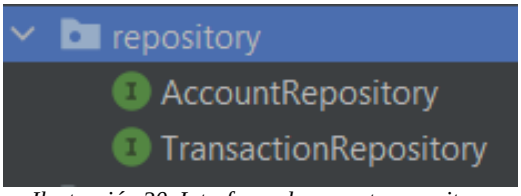

Ilustración 30. Interfaces de paquete repository

Se hace uso de JPA Repository como ORM (Object Relational Mapping) con la base de datos, en Spring boot basta con colocar @Repository para hacerle saber a Spring que la interfaz es un repositorio, Ilustración 31.

```
@Repository
public interface AccountRepository extends JpaRepository<Account, Integer> {
}
```

Ilustración 31. AccountRepository

De la misma manera se denota Transaction, Ilustración 32.

```
@Repository
public interface TransactionRepository extends JpaRepository<Transaction, Integer> {
}
```

Ilustración 32. TransactionRepository

Paquete de Servicios

En el paquete de servicio se hace la implementación de los métodos que contiene el repositorio, se tienen dos clases de servicio, una para Account y otra para Transaction, Ilustración 33.

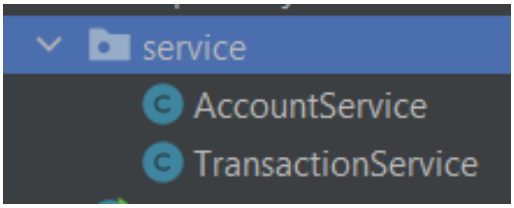

Ilustración 33. Paquete Service

En Spring boot se indica con @Service que una clase va a ser un servicio, además gracias a la inyección de dependencias se hace uso del repositorio para implementar la lógica de los métodos que este contiene, Ilustración 34.

```java
@Service
@Transactional
public class AccountService {

    @Autowired
    AccountRepository accountRepository;

    public List<Account> getAccounts() { return this.accountRepository.findAll(); }

    public Optional<Account> getAccountById(int idAccount) { return this.accountRepository.findById(idAccount); }

    public void saveAccount(Account account){
        accountRepository.save(account);
    }

    public void deleteAccount(int idAccount) { accountRepository.deleteById(idAccount); }

    public boolean existsAccountById(int idAccount) { return accountRepository.existsById(idAccount); }
}
```

Ilustración 34. AccountService

Paquete de Controladores

Se cuenta con dos controladores en la aplicación, Ilustración 35, básicamente contiene los métodos GET, POST, PUT y DELETE que se exponen en forma de API para que el frontend los pueda consumir.

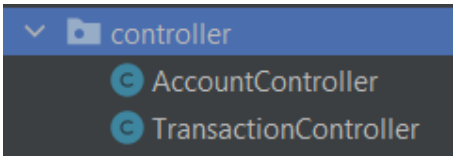

Ilustración 35. Paquete Controller

Para denotar en Spring que una clase es un controller se utiliza la etiqueta @Controller, pero se puede ser más estricto denotando que se utiliza un controlador con arquitectura REST y permitirá la interacción con los servicios web RESTful, Ilustración 36.

```java
//Especifica que es un controlador REST
@RestController
//Endpoint que se consume
@RequestMapping("/account")
//Activación de CORS
@CrossOrigin(origins = "*")
public class AccountController {

    //Inyección de dependencias
    @Autowired
    AccountService accountService;

    private ResponseEntity<?> responseEntity;
```

Ilustración 36. AccountController

En la Tabla 3 se muestran los endpoints que se pueden consumir, creados en AccountController asi como sus verbos de consumo:

Tabla 3. Endpoints creados en AccountController y verbos de consumo

Endpoint	Verbo
/account/ getAccounts	GET
/account/getAccount/{idAccount}	GET
/account/saveAccount	POST
/account/updateAccount/{idAccount}	PUT
/account//deleteAccount/{idAccount	DELETE

A continuación, Ilustración 37, se muestran algunos de los métodos de la clase AccountController.

```java
//Obtiene todas las cuentas
@GetMapping("/getAccounts")
public ResponseEntity<List<Account>> getAccounts() {
    List<Account> accounts = accountService.getAccounts();
    return new ResponseEntity<List<Account>>(accounts, HttpStatus.OK);
}

//Obtiene una cuenta por ID
@GetMapping("/getAccount/{idAccount}")
public ResponseEntity<Account> getAccountById(@PathVariable("idAccount") int idAccount) {
    if (!accountService.existsAccountById(idAccount))
        return new ResponseEntity(new MessageDTO("No existe la cuenta"), HttpStatus.NOT_FOUND);

    Account account = accountService.getAccountById(idAccount).get();
    return new ResponseEntity(account, HttpStatus.OK);
}
```

Ilustración 37. AccountController

5.2.9 Proyecto Angular

Creación de proyecto Angular

En la Ilustración 38 se pude observar la creación del proyecto angular.

```
D:\Documentos\Master\TFM\Front>ng new LeapFront
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? SCSS   [ https://sass-lang.com/documentation/syntax#scss
 ]
CREATE LeapFront/angular.json (3679 bytes)
CREATE LeapFront/package.json (1287 bytes)
```

Ilustración 38. Creación de proyecto Angular

Arquitectura de proyecto Angular

Angular por defecto trae su propia arquitectura definida, el único inconveniente de utilizar esta arquitectura es la escalabilidad, por lo cual, se opta por utilizar una arquitectura que permita a futuro realizar o implementar más módulos de manera más sencilla. Además de implementar Lazy load en el enrutamiento para retrasar la carga o inicialización de los objetos al momento de utilizarlos, lo que causa que la aplicación sea más rápida en cuanto a navegación y procesamiento se trata. A continuación, se muestra la arquitectura utilizada en la Ilustración 39.

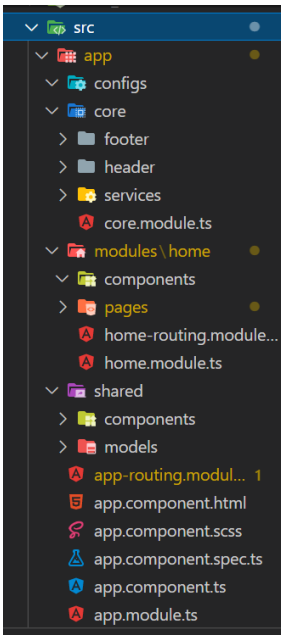

Ilustración 39. Arquitectura del proyecto angular

Modules

Un módulo está compuesto por páginas y componentes. La diferencia de una página a un componente radica en que la página es una colección de componentes, mientras que los componentes son pequeñas partes en común y específicas que se pueden utilizar para una página. A continuación, se muestra la composición de un módulo (html, css y ts) en Ilustración 40, Ilustración 41 y Ilustración 42 respectivamente. La mayoría de componentes tienen la misma estructura.

```
<div class="box">
  <app-header class="header"></app-header>
  <div class="wrapper">
    <div id="wrapper-left">
      <img src="../../../../../assets/img/card.svg" alt="card">
    </div>
    <div id="wrapper-right">
      <div id="text">
        <p>Escriba el valor que desea transferir</p>
      </div>
      <div id="form">
        <form>
          <div class="form-group">
            <input type="text" class="form-control" maxlength="3" minlength="2" value="{{finalValue}}">
          </div>
        </form>
      </div>
      <div>
        <app-keyboard (valueKeyBoard)="receiveMessage($event)"></app-keyboard>
      </div>
    </div>
  </div>
  <app-footer class="footer"></app-footer>
  <ngx-spinner></ngx-spinner>
</div>      You, a week ago • version 1.0 finish
```

Ilustración 40. Composición de un módulo HTML

```
.wrapper {
  display: flex;
  flex-direction: row;
  height: 80%;
  width: 70%;
  box-shadow: 0 3px 10px rgb(0 0 0 / 0.2);
}

#wrapper-left {
  display: flex;
  width: 40%;
  justify-content: center;
  align-items: center;
  background-color: rgb(255, 255, 255);
}

#wrapper-left img {
  width: 60%;
}

#wrapper-right {
  display: flex;
  flex-direction: column;
  width: 60%;
  background: rgb(255, 255, 255);
  justify-content: center;
  align-items: center;
}
```

Ilustración 41. Composición de un módulo CSS

```typescript
export class TransferValueComponent implements OnInit {
  valueKeyboard: any;
  valueaux: any;       You, a week ago • version 1.0 finish
  finalValue: any = "";
  accountNumber: string;

  constructor(
    private router: Router,
    private commonService: CommonService,
    private route: ActivatedRoute,
    private accountService: AccountService
  ) {
    this.accountNumber = this.route.snapshot.paramMap.get("account");
  }

  ngOnInit(): void {}
```

```typescript
receiveMessage($event) {
  this.valueKeyboard = $event;
  console.log("recibi", this.valueKeyboard);

  if (this.valueKeyboard === "ok") {
    if (this.finalValue.length == 3 || this.finalValue.length >= 2) {
      this.accountService
        .getAccountByNumber(this.accountNumber)
        .subscribe((res) => {
          this.commonService.showConfirmDialogTransfer(
            `Esta seguro que desea transferir $ ${this.finalValue} a la cuenta ${this.accountNumber}`,
            "Transacción exitosa",
            "Muchas gracias por utilizar nuestros cajeros. Recibirá un recibo de transacción al email registrado",
            "success",
            this.finalValue,
            res.idAccount
          );
        });
    } else {
      this.cleanValue();
      this.commonService.showAlertError(
        "Error",
        "Por favor ingrese la cantidad"
      );
    }
  } else if (this.valueKeyboard === "fix") {
    this.cleanValue();
  } else if (this.valueKeyboard === "cancel") {
    this.cleanValue();
  }
```

Ilustración 42. Composición de un módulo TS

Core

El paquete Core contiene servicios, componentes universales como el header o el

footer, lo que permite tener una sola instancia por aplicación de estos componentes. Adicionalmente los servicios, interceptors, guards, mocks y clases se colocan en esta carpeta.

Los servicios permiten conectar con los endpoints expuestos por el backend gracias al uso de HttpClient de Angular. En las siguientes ilustraciones, Ilustración 43 y Ilustración 44, se muestran los servicios implementados en la aplicación, así como el código de uno de estos.

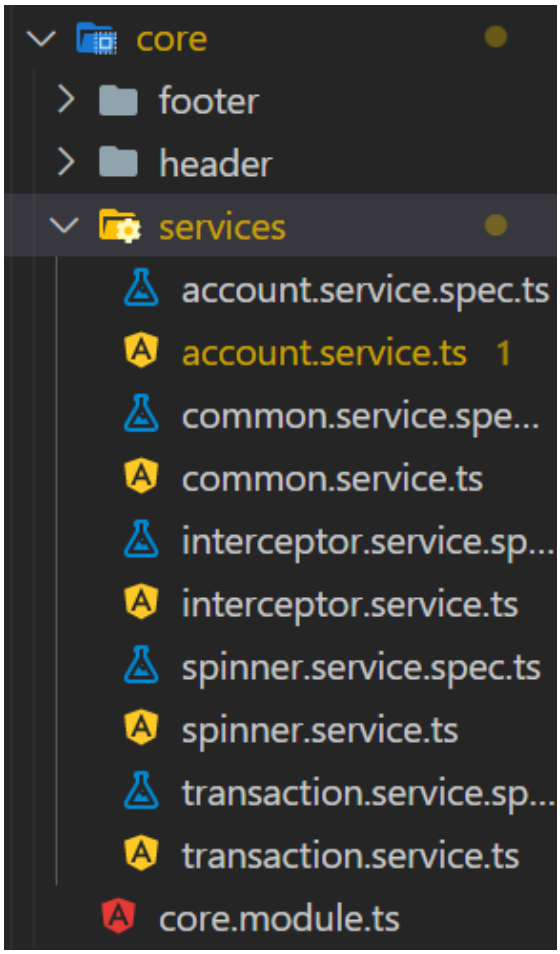

Ilustración 43. Composición de paquete Core

```
@Injectable({
  providedIn: "root",
})
export class AccountService {
  baseUrl = environment.baseUrl + "/account";

  constructor(private http: HttpClient) {}

  getAccount(idAccount: number) {
    return this.http.get<Account>(`${this.baseUrl}/getAccount/${idAccount}`);
  }

  validatePasswords(idAccount: number, password: string) {
    let jsonObject = {
      idAccount: idAccount,
      password: password,
    };
    return this.http.post<any>(`${this.baseUrl}/validatePassword`, jsonObject);
  }

  getAccountByNumber(account: string) {
    return this.http.get<Account>(
      `${this.baseUrl}/getAccountNumber/${account}`
    );
  }
}
```

Ilustración 44. Servicio Account

El resto de servicios están implementados de una manera muy similar.

Se encapsula la URL base de los endpoints por motivos de seguridad en el archivo de environments, como se muestra en la Ilustración 45.

```
export const environment = {
  production: true,
  baseUrl: "https://heroku-boot-leap.herokuapp.com",
};
```

Ilustración 45. Archivo Environment

Shared

En Shared se encuentran componentes compartidos que se utilizan en diferentes componentes de la aplicación. Al existir un SharedModule se puede importar en cualquier otro modulo para hacer uso de estos componentes. Al ser un módulo compartido no depende de ningún otro modulo.

En la aplicación se utiliza como componente compartido un teclado en pantalla el cual se implementa en varios sitios de la aplicación, Ilustración 46.

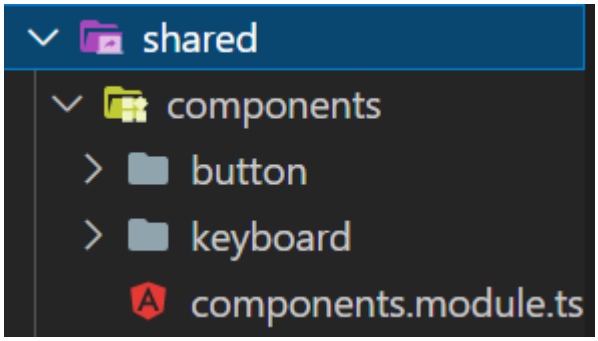

Ilustración 46. Estructura de carpeta shared

Dicho teclado se muestra en la siguiente Ilustración 47.

Ilustración 47. Keyboard compartido

Este teclado se construyó de la manera mostrada en la Ilustración 48 en la parte HTML.

```
<div class="parent">
  <div>
    <div>
      <button class="btn btn-secondary" (click)="getValue(1)">1</button>
    </div>
  </div>
  <div>
    <div>
      <button class="btn btn-secondary" (click)="getValue(2)">2</button>
    </div>
  </div>
  <div>
    <div>
      <button class="btn btn-secondary" (click)="getValue(3)">3</button>
    </div>
  </div>
  <div>
    <div>
    </div>
  </div>
```

Ilustración 48. Keyboard HTML

El trabajo de posicionamiento y la forma de columnas se logra gracias a utilizar Grid CSS, Ilustración 49.

```
button {
  width: 90px;
  height: 38px;
}

.grid-container {
  display: grid;
}

.parent {
  display: grid;
  grid-template-columns: repeat(3, 1fr) 0.2fr 1fr;
  grid-template-rows: repeat(4, 1fr);
  grid-column-gap: 12px;
  grid-row-gap: 18px;
}

.expand {
  grid-column-start: 5;
  grid-row: 3 / 4;
}
```

Ilustración 49. Keyboard CSS

A este teclado se le agrega un event emiiter para saber que botón se ha pulsado y de esta manera indicarle al componente padre el valor del botón, Ilustración 50.

```
export class KeyboardComponent implements OnInit {
  @Output() valueKeyBoard = new EventEmitter<any>();

  constructor() {}

  ngOnInit(): void {}

  getValue(value: any) {
    //console.log(value);
    this.valueKeyBoard.emit(value);
  }
}
```

Ilustración 50. Keyboard ts

Configs

En la carpeta configs se coloca la configuración de la aplicación en caso dado que sea necesario, scripts o archivos necesarios para el correcto funcionamiento de la aplicación.

5.2.10 Despliegue Backend Heroku

En la Ilustración 51 se pude observar la creación de un proyecto heroku por medio de consola de comandos.

```
D:\Documentos\Master\TFM\Back\leap>heroku create
Creating app... !
 !    Invalid credentials provided.
heroku: Press any key to open up the browser to login or q to exit:
Opening browser to https://cli-auth.heroku.com/auth/cli/browser/573df53c-a533-40c5-ba7d-dbb2f9a2b9bc?requestor=SFMyNT
2gDbQAAAA43OS4xNTcuMTA5LjIyNG4GAFhYeg17AWIAAVGA.w0xv8xF08Xu9tqO1JJLosDwIKJbe0oNCjyzamlsgl50
Logging in... done
Logged in as vpm570@inlumine.ual.es
Creating app... done, ⬚ salty-ocean-85032
https://salty-ocean-85032.herokuapp.com/ | https://git.heroku.com/salty-ocean-85032.git
```

Ilustración 51. Creación Proyecto Heroku

Se renombra el proyecto, Ilustración 52, ya que heroku por defecto asigna un nombre autogenerado por él.

```
D:\Documentos\Master\TFM\Back\leap>heroku apps:rename --app salty-ocean-85032 heroku-boot-leap
Renaming salty-ocean-85032 to heroku-boot-leap... done
https://heroku-boot-leap.herokuapp.com/ | https://git.heroku.com/heroku-boot-leap.git
 !    Don't forget to update git remotes for all other local checkouts of the app.
Git remote heroku updated
```

Ilustración 52. Renombramiento de proyecto Heroku

A continuación, se inicializa el repositorio Git y se añaden los cambios realizados en el proyecto, Ilustración 53.

```
D:\Documentos\Master\TFM\Back\leap>git init
Reinitialized existing Git repository in D:/Documentos/Master/TFM/Back/leap/.git/

D:\Documentos\Master\TFM\Back\leap>git add .
```

Ilustración 53. Inicialización del git en el proyecto

Con git remote -v se muestran los orígenes que existen el proyecto, donde se puede ver que se tiene una rama llamada heroku, Ilustración 54.

```
D:\Documentos\Master\TFM\Back\leap>git remote -v
heroku  https://git.heroku.com/heroku-boot-leap.git (fetch)
heroku  https://git.heroku.com/heroku-boot-leap.git (push)
origin  https://github.com/VManuelPM/LeapBackend.git (fetch)
origin  https://github.com/VManuelPM/LeapBackend.git (push)
```

Ilustración 54. Git Origin del proyecto

Se construye el proyecto, se comprime y se lanza la aplicación, Ilustración 55.

```
remote: -----> Compressing...
remote:        Done: 95M
remote: -----> Launching...
remote:        Released v3
remote:        https://heroku-boot-leap.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/heroku-boot-leap.git
 * [new branch]      main -> main
```

Ilustración 55. Push del proyecto a rama heroku

Gracias a Heroku automáticamente detecta que es una aplicación Java, se compila automáticamente y se despliega, Ilustración 56.

```
D:\Documentos\Master\TFM\Back\leap>git push heroku
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 8 threads.
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 523 bytes | 261.00 KiB/s, done.
Total 6 (delta 2), reused 0 (delta 0)
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Building on the Heroku-20 stack
remote: -----> Using buildpack: heroku/gradle
remote: -----> Gradle app detected
remote: -----> Spring Boot detected
remote: -----> Installing JDK 11... done
remote: -----> Building Gradle app...
remote: -----> executing ./gradlew build -x check
remote:        To honour the JVM settings for this build a single-use Daemon process will be forked. S
remote:        Daemon will be stopped at the end of the build
remote:
remote:        > Task :compileJava
remote:        Note: Some input files use unchecked or unsafe operations.
remote:        Note: Recompile with -Xlint:unchecked for details.
remote:
remote:        > Task :processResources
remote:        > Task :classes
remote:        > Task :bootJarMainClassName
remote:        > Task :bootJar
remote:        > Task :jar
remote:        > Task :assemble
remote:        > Task :build
remote:
remote:        BUILD SUCCESSFUL in 17s
remote:        5 actionable tasks: 5 executed
remote: -----> Discovering process types
remote:        Procfile declares types -> web
remote:
remote: -----> Compressing...
remote:        Done: 96M
remote: -----> Launching...
remote:        Released v15
remote:        https://heroku-boot-leap.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/heroku-boot-leap.git
   953d08d..5cef983  main -> main
```

Ilustración 56. Aplicación Java compilada y desplegada

Para verificar que se ha desplegado la aplicación basta con utilizan un cliente de API como Postman y hacer una petición a los diferentes endpoint que previamente se expusieron, tal y como se muestra en la Ilustración 57.

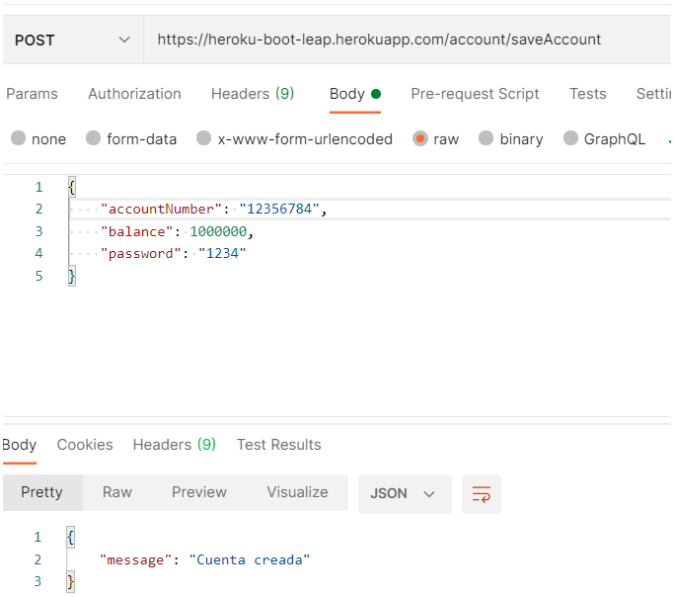

Ilustración 57. Probando despliegue en Heroku

Por ejemplo, si se utiliza el método de obtener los detalles de una cuenta por medio de ID, Ilustración 58.

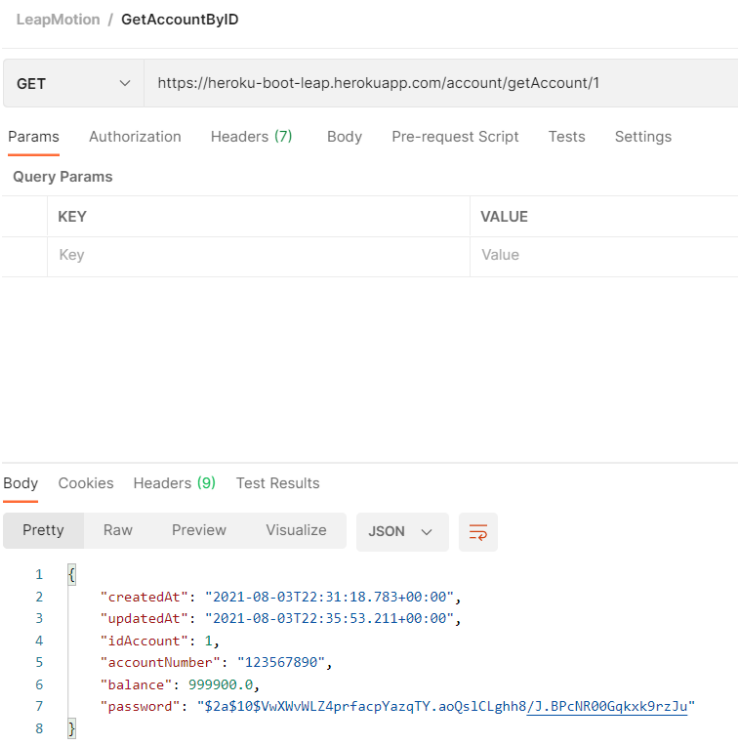

Ilustración 58. Despliegue en Heroku get Account

En heroku, gracias a que se activó hibernate show sql en el properties del proyecto java, se puede ver qué consultas hace a la base de datos por medio de los logs, Ilustración 59.

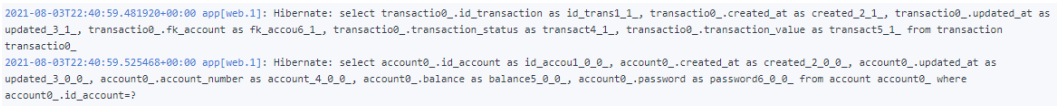

Ilustración 59. Logs de Heroku

5.2.11 Despliegue de Aplicación Angular en Heroku

Una vez creamos un proyecto en Heroku, se despliega la aplicación angular enlazando por medio de un repositorio de Git a heroku de la forma mostrada en la Ilustración 60.

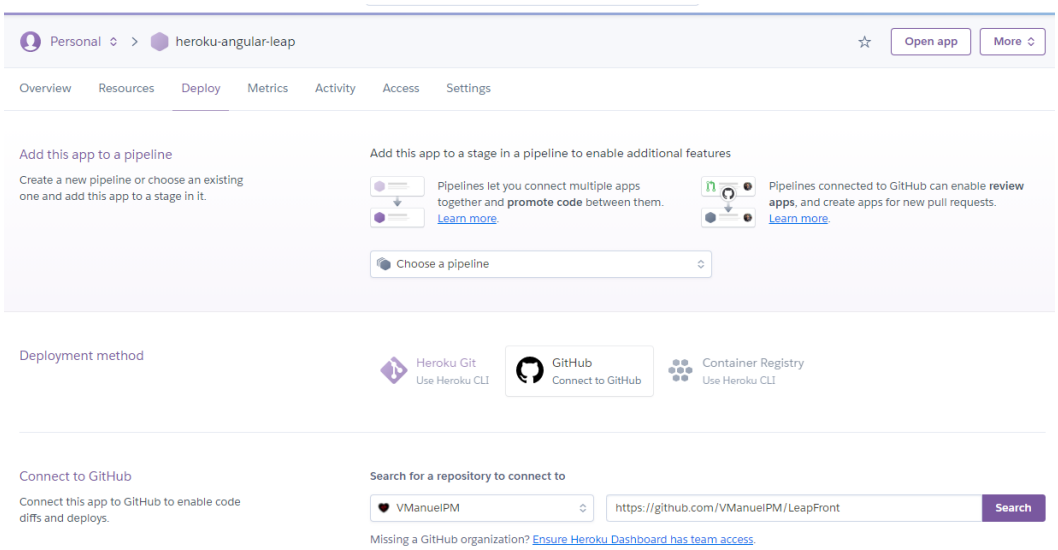

Ilustración 60. Conectando Heroku a Github

Se conecta al repositorio de Github, Ilustración 61.

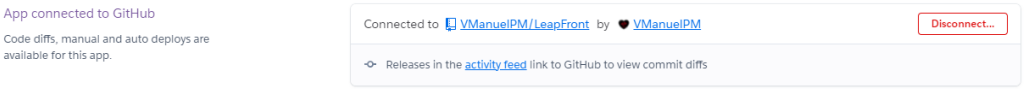

Ilustración 61. Conexión realizada a proyecto front en Git

Y se despliega el proyecto manualmente desde la rama main, Ilustración 62.

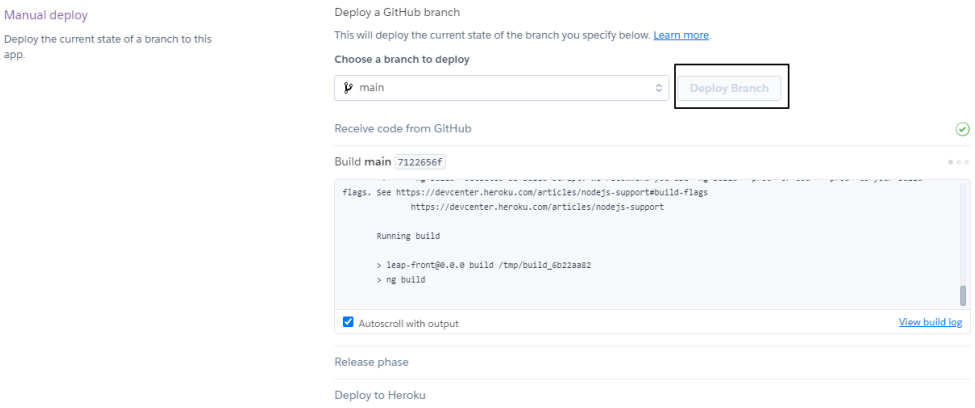

Ilustración 62. Primer despliegue de proyecto front desde heroku

Al desplegarlo de esta manera dará fallo ya que se debe primero agregar un paso en el package.json del proyecto, como se muestra en la Ilustración 63.

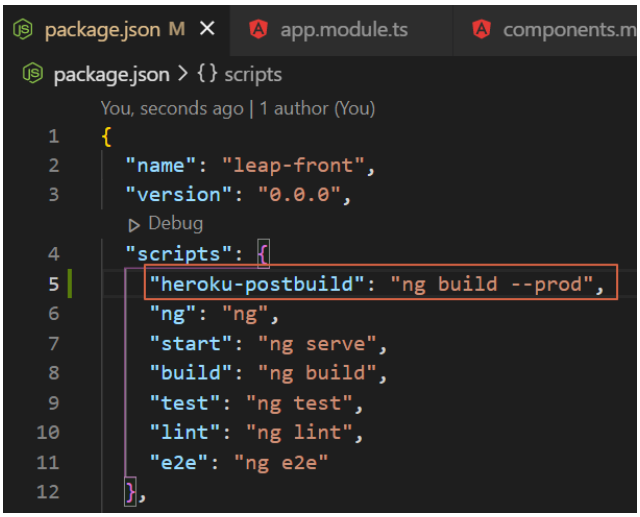

Ilustración 63. Configuración de Package.json

Se especifican las versiones de node y npm que se usaran para la compilación, Ilustración 64.

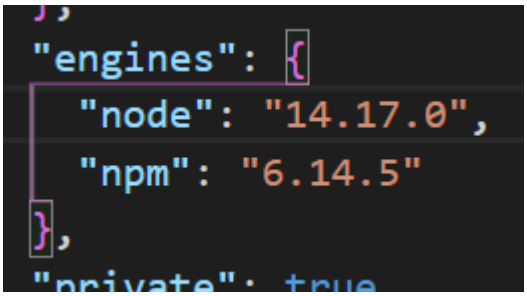

Ilustración 64. Especificación de versiones de node y npm

Se requiere también un servidor de Node.js para la aplicación de angular, se instala Node.js express module para crear un servidor simple, Ilustración 65.

Ilustración 65. Instalación en el proyecto de NodeJs y express

Se crea a nivel de raíz de proyecto un archivo .js llamado server en el cual se almacena la configuración básica del servidor, Ilustración 66.

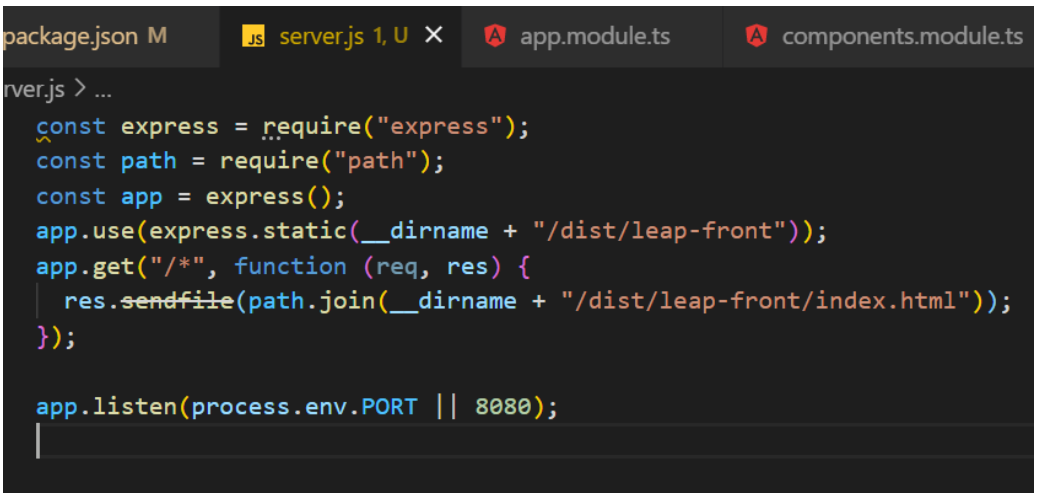

Ilustración 66. Configuración basica del servidor

Una vez realizado esto se debe cambiar en el package.json el inicio del proyecto para el servidor previamente configurado, Ilustración 67.

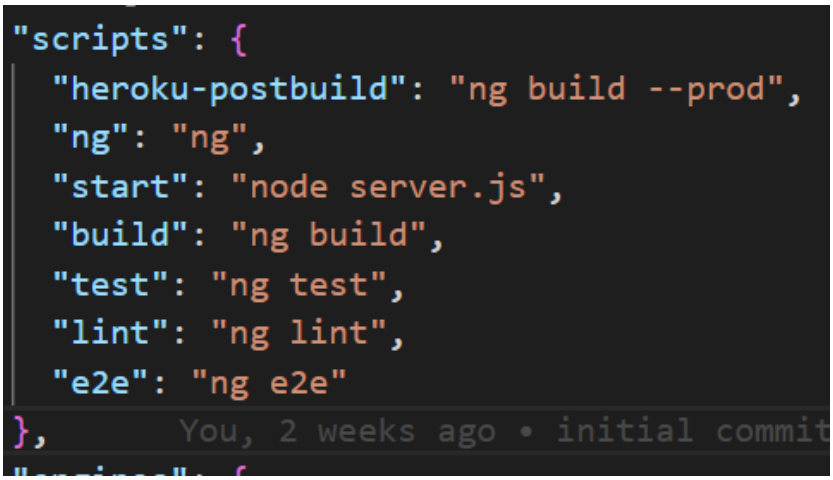

Ilustración 67. Cambio de Start package.json

Al realizar el push al repositorio de Git automáticamente se despliega la nueva versión subida, Heroku vuelve a lanzar la compilación y el despliegue, Ilustración 68.

```
-----> Build succeeded!
-----> Discovering process types
       Procfile declares types     -> (none)
       Default types for buildpack -> web
-----> Compressing...
       Done: 77.3M
-----> Launching...
       Released v5
       https://heroku-angular-leap.herokuapp.com/ deployed to Heroku
```

Ilustración 68. Log de Heroku desplegando

5.2.12 Configuración de Leap Motion:

Para configurar primero leap motion se debe descargar el sdk de leap motion que se encuentra en la página oficial[3], en este caso se utilizó Windows por lo cual se instala

[3] https://developer.leapmotion.com/sdk-leap-motion-controller

dando next en las ventanas emergentes, una vez instalado el sdk en la barra de tareas aparece el icono mostrado en la Ilustración 69.

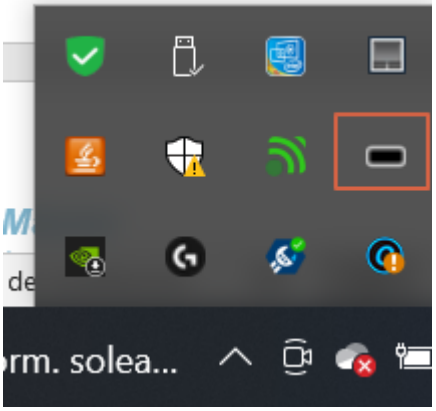

Ilustración 69. Icono de leap motion apagado

En la Ilustración 69 el leap motion aparece apagado ya que no se encuentra conectado, una vez se conecte el leap motion, el icono se mostrará como en la Ilustración 70.

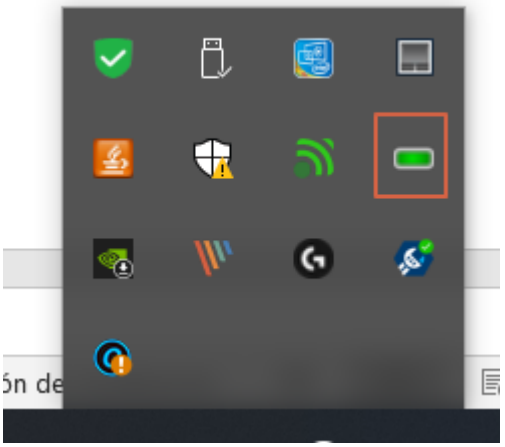

Ilustración 70. Leap Motion Conectado

El siguiente paso es calibrar el leap motion, para esto se da clic derecho en icono de leap motion de la barra de tareas y configuración, aparecerá una ventana como la de la Ilustración 71.

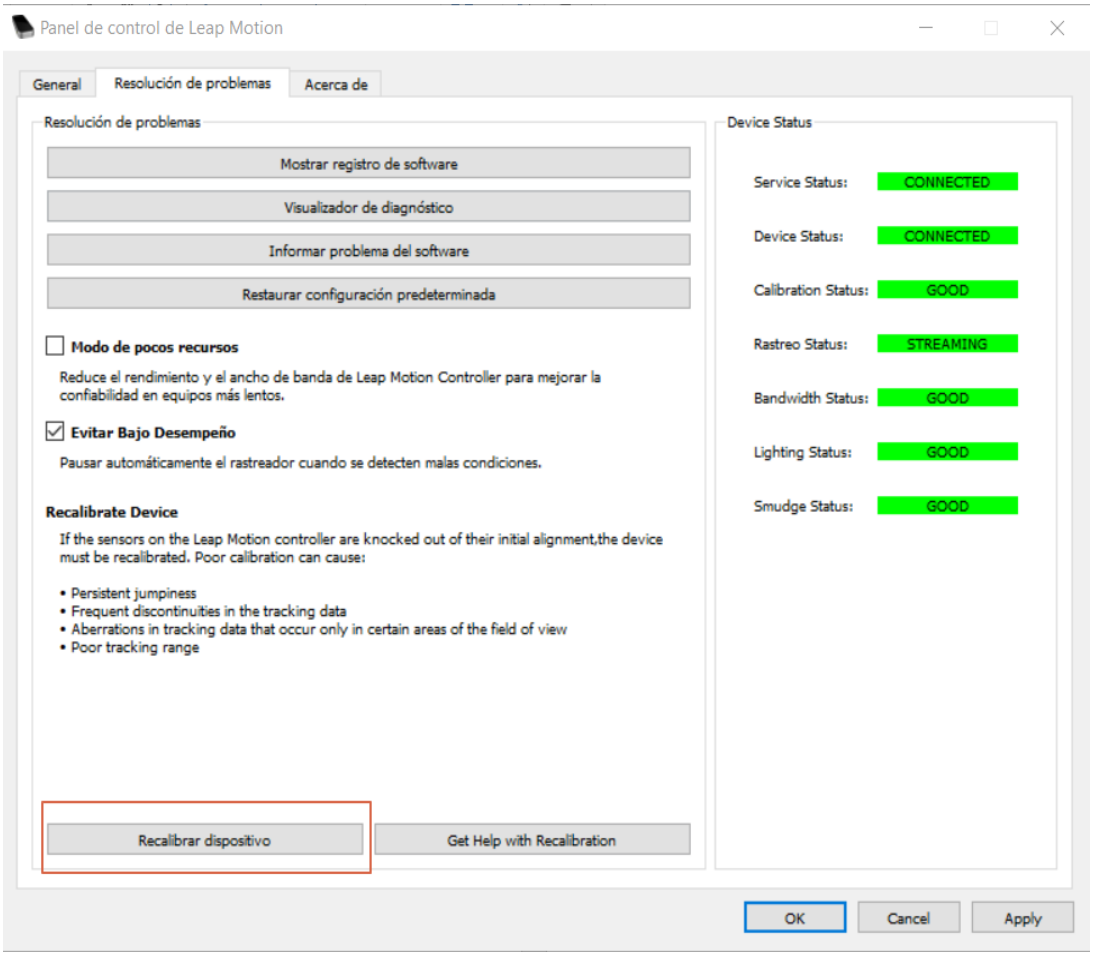

Ilustración 71. Ventana de configuración de leap motion

Luego de calibrarlo, nuevamente se pulsa clic derecho en el icono de leap motion y posteriormente se va a visualizador, para ver que está funcionando todo correctamente y el dispositivo es capaz de capturar las manos. A continuación, se presenta una figura en la cual se evidencia que el leap motion está observando y capturando las manos de manera correcta, Ilustración 72.

Ilustración 72. Visualizador de leap motion

Posteriormente se descarga la aplicación touch free de la página https://developer.leapmotion.com/touchfree, la cual permite simular un cursor en pantalla, o hacer el gesto de click. Al descargar la aplicación se descarga un .rar el cual al descomprimirlo contiene los archivos de la Ilustración 73.

MonoBleedingEdge	13/01/2021 12:38	Carpeta de archivos	
TouchFree_Data	13/01/2021 12:38	Carpeta de archivos	
LICENSE.txt	19/01/2021 8:51	Archivo TXT	1 KB
README.txt	13/01/2021 12:38	Archivo TXT	15 KB
ThirdPartyLicences.txt	13/01/2021 12:38	Archivo TXT	4 KB
TouchFree.exe	13/01/2021 12:38	Aplicación	644 KB
UnityCrashHandler64.exe	13/01/2021 12:38	Aplicación	1.069 KB
UnityPlayer.dll	13/01/2021 12:38	Extensión de la ap...	25.147 KB
version.txt	13/01/2021 12:38	Archivo TXT	1 KB

Ilustración 73. Archivos TouchFree

Al ejecutar el .exe de TouchFree se abre una aplicación la cual muestra que la cámara de leap motion está conectada y una opción para configurar la cámara,

Ilustración 74.

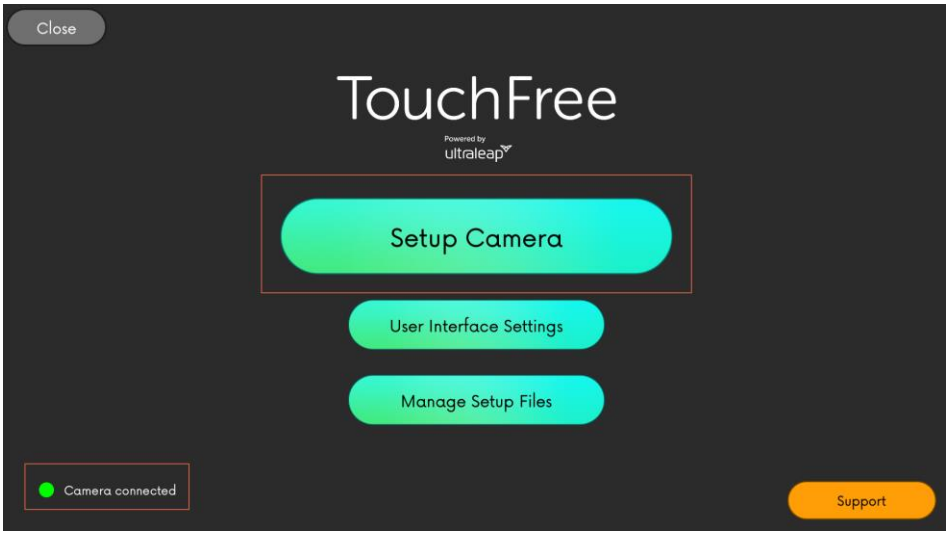

Ilustración 74. TouchFree primer contacto

Al dar click en la primera opción de configuración de cámara se despliega una ventana donde se indica que escoja la posición en la cual el leap motion esta, Ilustración 75.

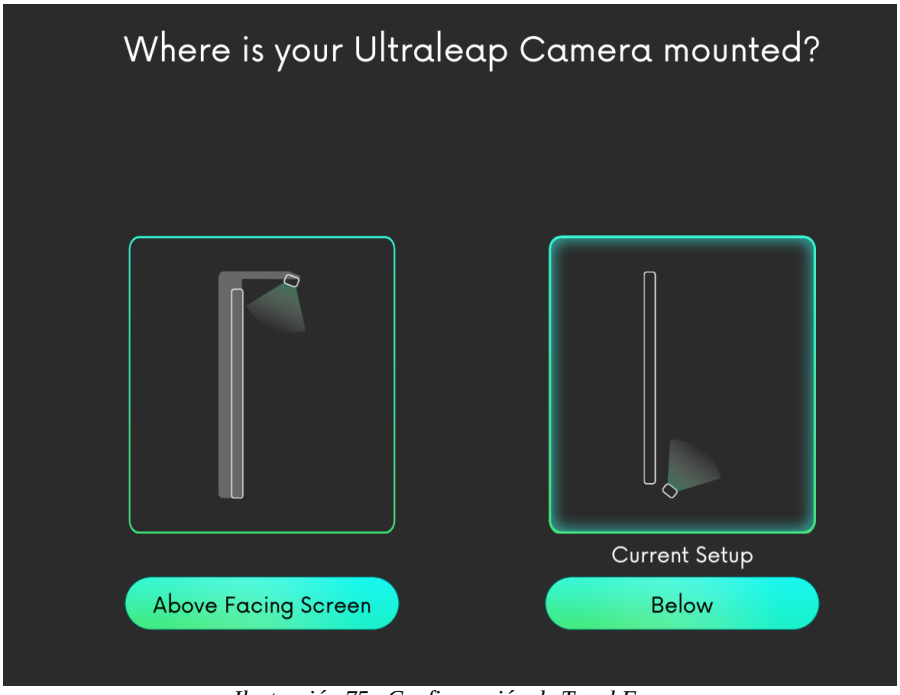

Ilustración 75. Configuración de TouchFree

Se despliegan dos opciones, manual o rápido, donde manual se utiliza cuando se va a colocar en un ambiente definitivo en el cual se conocen las dimensiones y obstáculos alrededor del leap, Ilustración 76.

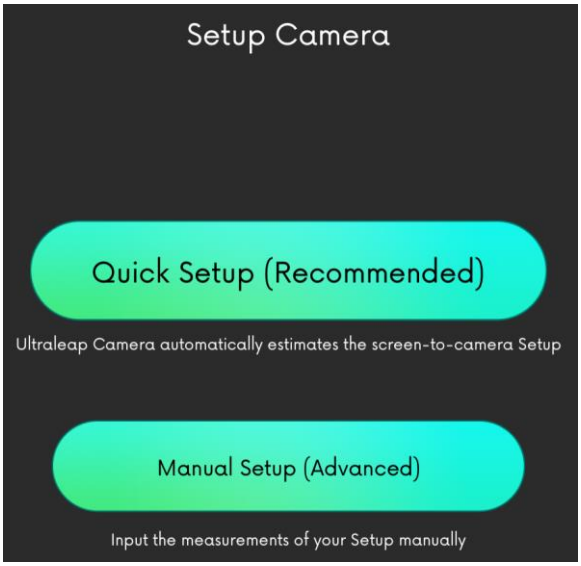

Ilustración 76. Configuración rapida o manual de TouchFree

Al seguir los pasos de configuración se vuelve al menú principal donde se va a observar un cursor de color oscuro en forma de circulo cuando leap motion detecta las manos, Ilustración 77.

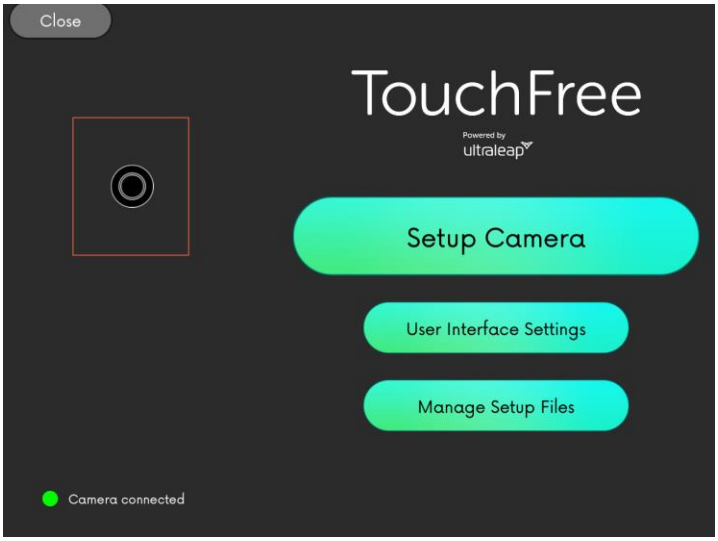

Ilustración 77. Cursor mostrado en pantalla

El cursor, tanto el tiempo de espera para click, como el color y el tipo de interacción es decir si se realiza el gesto de dar click o únicamente se mueve el cursor con la mano, Ilustración 78, se pueden cambiar entrando en la segunda opción User Interface Settings.

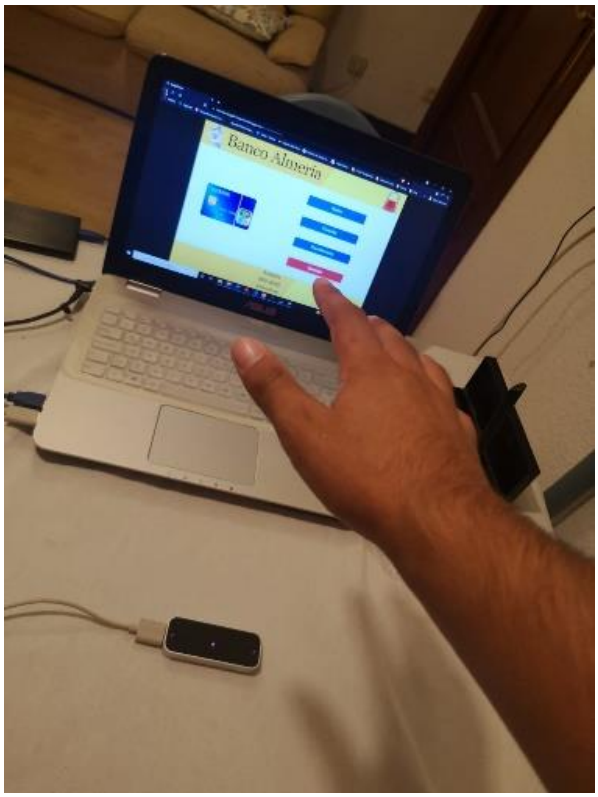

Ilustración 78. Interacción con leap motion

6. RESULTADOS Y DISCUSIÓN

6.1 Resultados

Como resultado la aplicación se encuentra desplegada y funcional[4].

En la Ilustración 79, se presenta la pantalla home del proyecto desplegado. Como se observa, basándose en la teoría del color, se realizó un contraste de colores que permitiera que la pantalla fuera llamativa sin perder la atención del usuario.

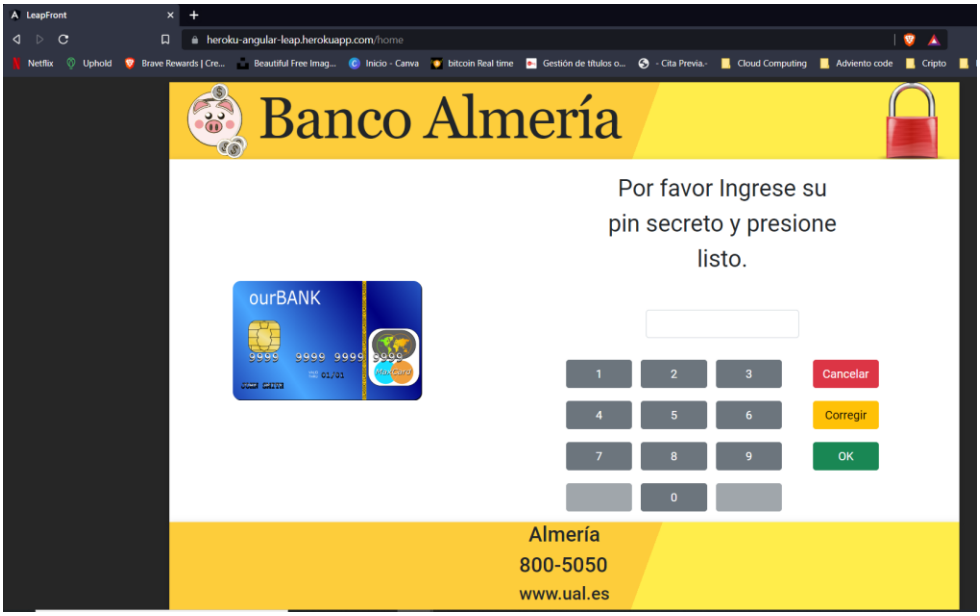

Ilustración 79. Pantalla Home

Una vez en la pantalla home, se requiere un password, el cual refleja la funcionalidad del cajero al momento de introducir el pin secreto de la cuenta, Ilustración 80. El password utilizado en el trabajo es 1234.

[4] https://heroku-angular-leap.herokuapp.com/home

Ilustración 80. Spinner de Carga del sistema

Una vez adentro, se pueden realizar tres operaciones básicas como se observa en la Ilustración 81, en donde, además, se aprecia que la operación se está realizando con el cursor provisto por el leap motion.

Ilustración 81. Cursor de leap motion en pantalla de retiro

En este caso, se pueden elegir dos tipos de cuenta estas indiferentes al tipo de selección, Ilustración 82.

Ilustración 82. Selección de tipo de cuenta

Por ejemplo, una vez seleccionada la operación retiro, cuenta de ahorros, permite la elección de diferentes cantidades tal como lo permite un cajero real, Ilustración 83.

Ilustración 83. Valor de retiros del sistema

Una vez seleccionado el valor de dinero a retirar en el ejemplo, aparece la confirmación de la operación, Ilustración 84 y posteriormente la ventana que indica el saldo restante, Ilustración 85.

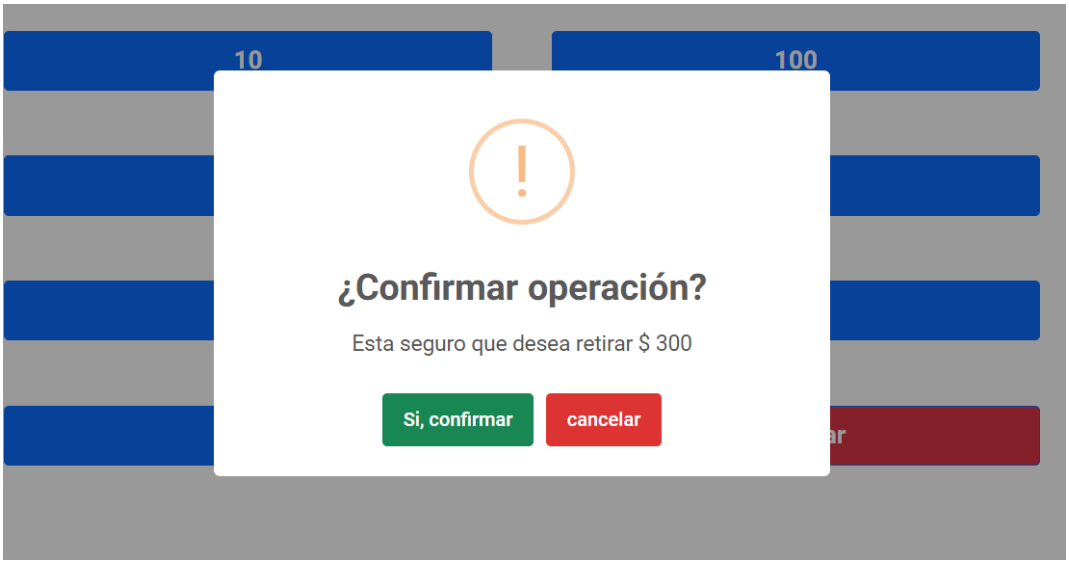

Ilustración 84. Ventana de confirmación de operación

Ilustración 85. Ventana de Saldo restante

6.1.1 Pruebas de aceptación

Las pruebas de aceptación se realizaron con postman y teniendo como base la URL de Heroku[5], la cual, facilitó heroku cuando se realizó el despliegue del proyecto Java. En la Ilustración 86 se muestra una prueba devolviendo un código de estado HTTP 200.

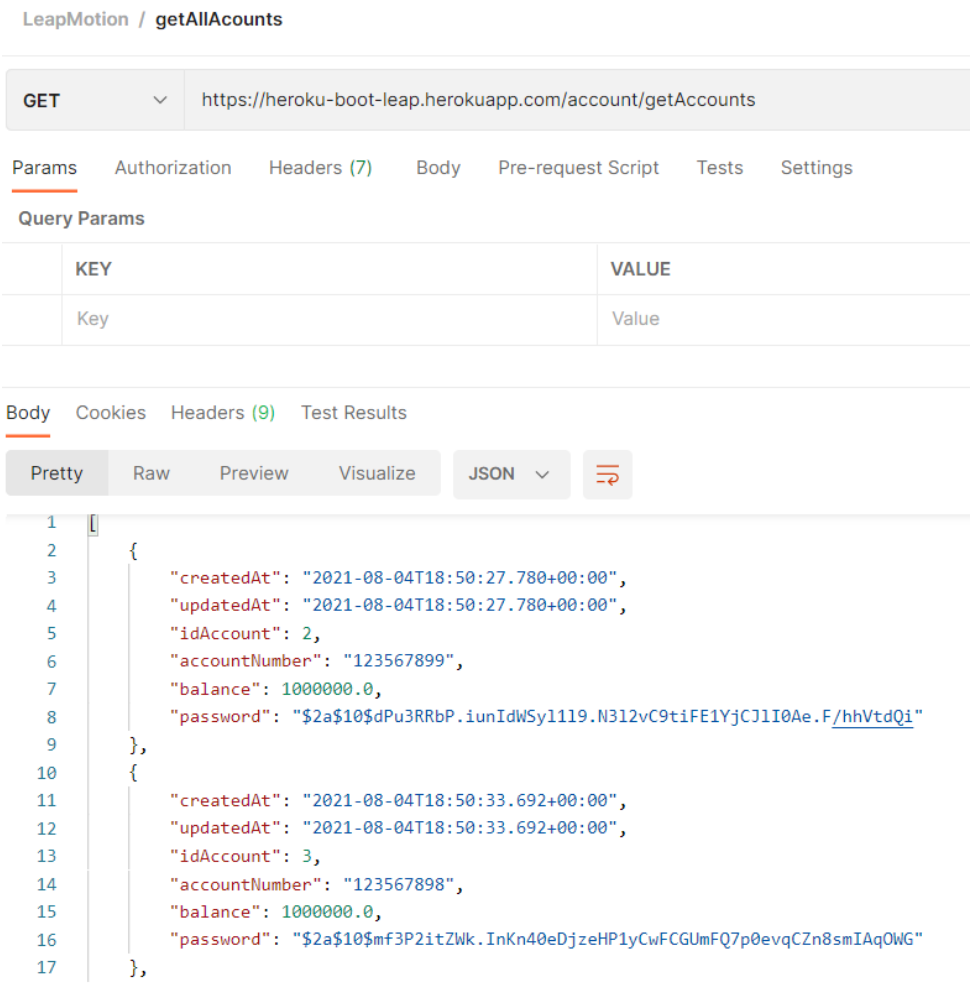

Ilustración 86. Pruebas de aceptación: Get Accounts

Se realizaron las pruebas tanto de error como pruebas funcionando correctamente. A continuación, se muestran las pruebas que se realizaron con postman en la Ilustración 87.

[5] https://heroku-boot-leap.herokuapp.com

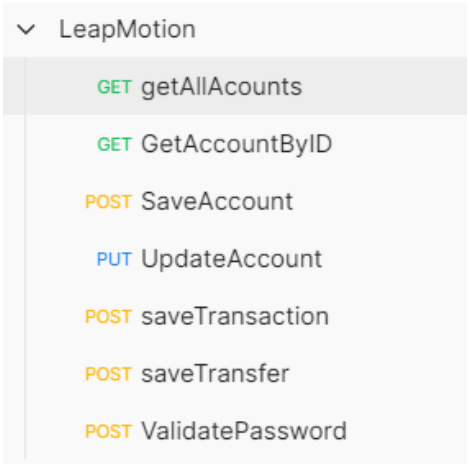

Ilustración 87. Pruebas de aceptación

6.1.2 Pruebas funcionales

Las pruebas funcionales se realizaron interactuando con cada una de las diferentes opciones que ofrece la aplicación (retiro, consulta de saldo y transferencia) teniendo en cuenta diferentes casuísticas. Además, se tomaron pruebas funcionales primeramente con 5 usuarios para posteriormente llevar el software a producción, Ilustración 88.

Ilustración 88. Pruebas funcionales

6.2 Discusión

Una vez realizadas las pruebas funcionales con éxito se procede a realizar pruebas de usuario con población de Madrid y Almería, de diferentes edades. La población se seleccionó aleatoriamente. Fotografías de las pruebas con usuarios se pueden observar en el Anexo 1.

Una vez probada la aplicación, se solicita a los usuarios la realización de una encuesta elaborada en google forms, la cual se puede observar en el Anexo 2; con el fin de obtener datos cuantitativos que permitieran determinar la validez del sistema, el planteamiento de la idea y la solución del problema.

El rango de edades de las personas encuestadas varía entre los 19 y los 69 años, como se observa en Figura 1.

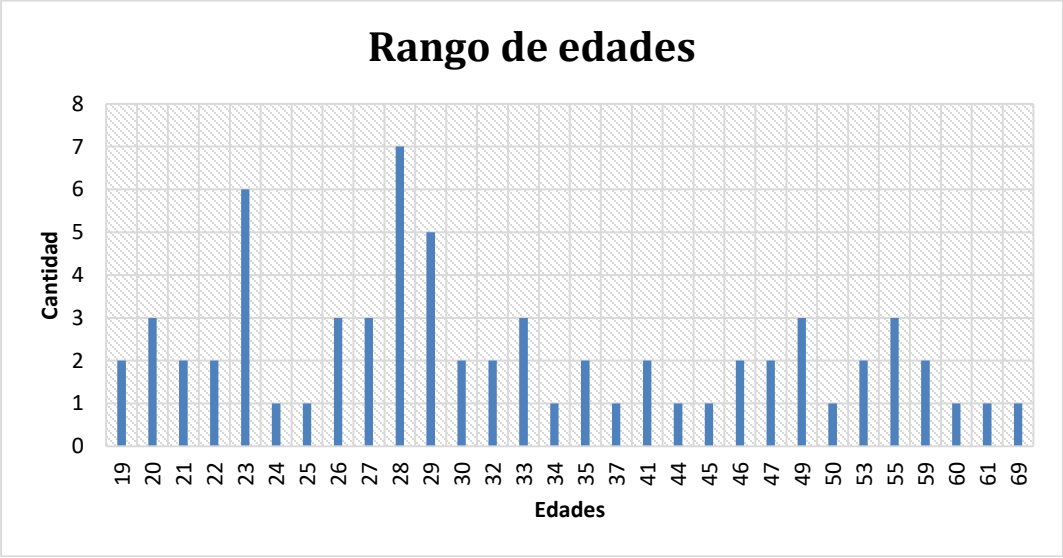

Figura 1. Rango de edades de los usuarios

De los datos obtenidos, se puede observar, Figura 2, que el grado de satisfacción de los usuarios con respecto a la dimensión de los botones y las cajas de texto es de un 92,8%.

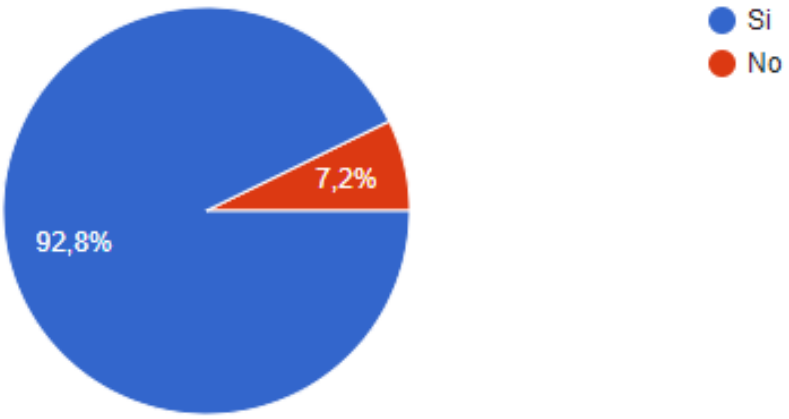

Figura 2. Grado de satisfacción de usuarios con dimensiones de botones y cajas.

En la Figura 3 se puede observar que el sistema para los usuarios es bastante familiar al de un cajero automático habitual, obteniendo un 92,8% que están de acuerdo en que lo es. Lo cual se logró tomando como base varios modelos de diseños de cajeros actuales implementados a lo largo del mundo.

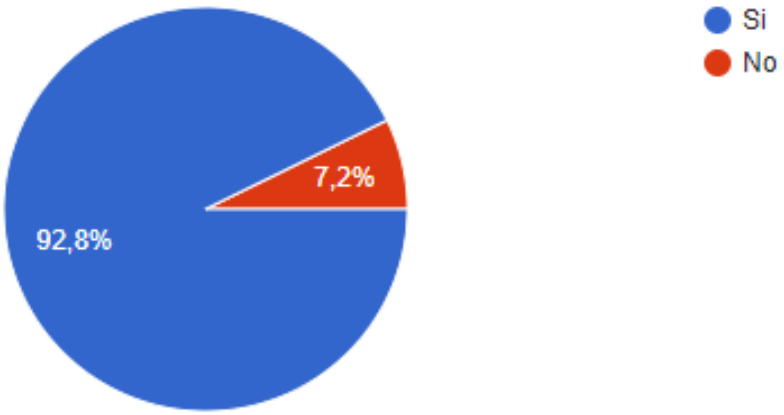

Figura 3. Familiaridad del proyecto con un cajero habitual

Se encontro también, que la navegación es fluida entre las ventanas ya que un 89,9% de los usuarios estuvieron de acuerdo en que lo era como se muestra en la Figura 4. Esto se logro en gran parte, por la arquitectura de alta escalabilidad implementada en el front y una arquitectura de back que permite persistir datos de manera eficaz y sencilla, sin quitar merito a Google Cloud y Heroku.

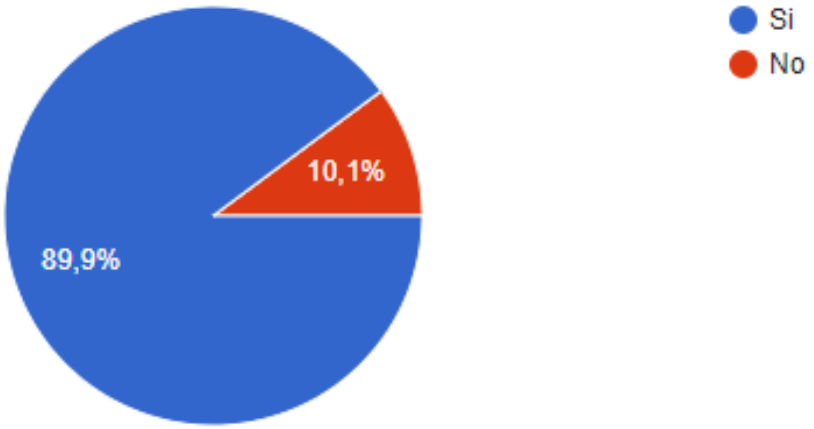

Figura 4. Navegación Fluida entre ventanas

Con respecto a la facilidad que tuvieron los usuarios para comprender a primera vista el sistema, se observa que en su mayoria, un 82,6%, indican que el sistema independientemente de la edad y el grado de familiaridad con la tecnología, es capaz brindar una experiencia de usuario sencilla y comprensible, según lo expuesto en la Figura 5.

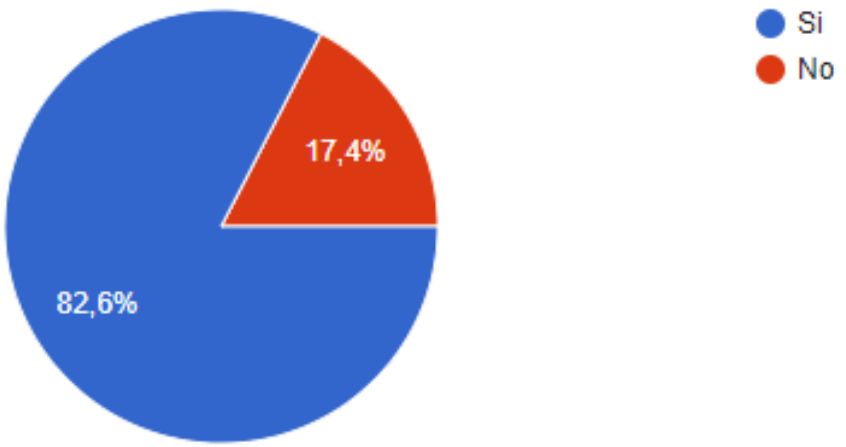

Figura 5. Comprensibilidad del sistema

Los colores escogidos para el sistema tienen una alta aceptación entre los usuarios, como se evidencia en la Figura 6, lo cual se consiguió, gracias a una correcta aplicación de la teoría del color, permitiendo contrastar los diferentes elementos de las pantallas

de tal forma que fuera llamativa sin llegar a sobrecargar al usuario.

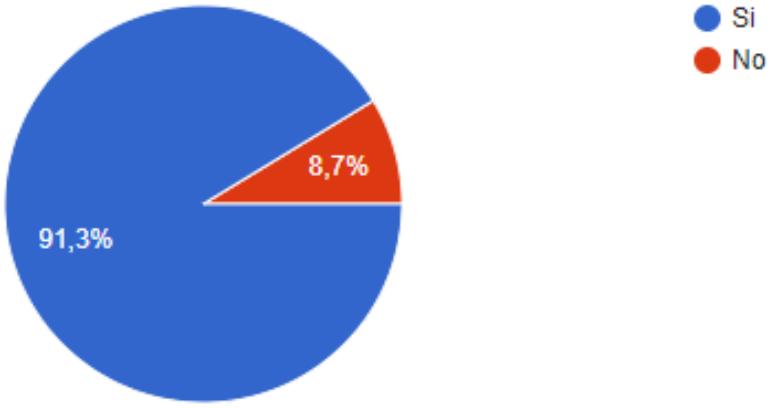

Figura 6. Aceptación de los Colores del sistema

La interacción del usuario con el aplicativo se tornó de forma fácil, Figura 7, gracias a que la interfaz creada es sencilla, fluida, agradable a la vista y con las dimensiones correctas de acuerdo a los datos obtenidos y explicados mencionados anteriormente. La utilización del leap motion, la cual hace parte de esta interacción con el aplicativo se entiende de igual forma intuitiva y sencilla.

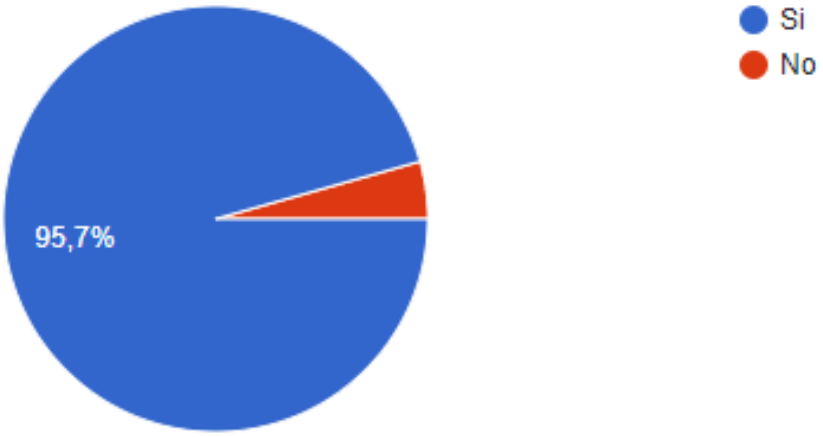

Figura 7. Facilidad de utilización del sistema

Los usuarios que utilizaron el sistema, Figura 8, lo encontraron de utilidad en el futuro mayoritariamente gracias a la situación vivida en el último año con el virus Covid19.

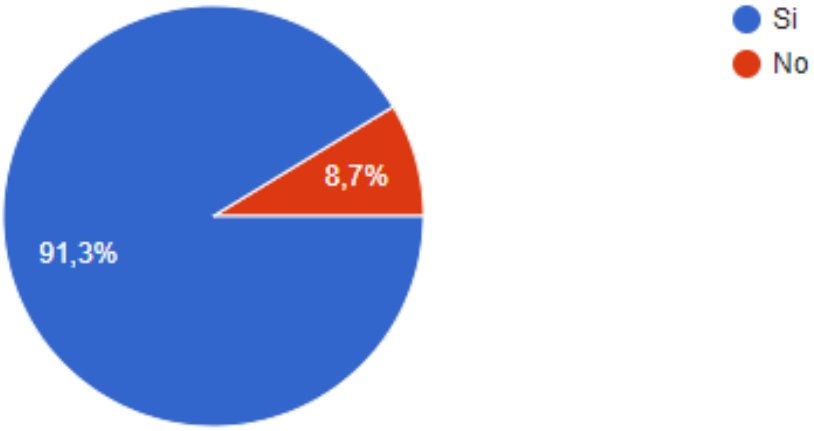

Figura 8. Utilidad del sistema en el futuro

Finalmente, en la Tabla 4, se encuentran tabulados todos los aspectos evaluados mediante la encuesta, discriminados por rangos de edades en intervalos de 10 años. En ella y en la Figura 9, se evidencia que la mayoría de las personas encuestadas se encuentran en un rango de edades entre los 21 y 30 años. Además, se puede observar que las personas comprendidas entre los 10 y los 60 años encontraron empatía y familiaridad en todos los aspectos evaluados del sistema mientras que para la opinión de las personas entre los 60 y 70 años, se encontraba dividida.

Tabla 4. Tabulación de encuestas discriminada en rangos de edades

	10-20		21-30		31-40		41-50		51-60		61-70		Total personas encuestadas
	Si	*No*	*Si*	*No*	*Si*	*No*	*Si*	*No*	*Si*	*No*	*Si*	*No*	
Dimensiones de botones y cajas de texto	5	0	29	3	8	1	12	0	8	0	1	1	68
Familiaridad con cajeros habituales	5	0	28	4	9	0	12	0	7	1	2	0	68
Fluidez de navegación	4	1	31	1	6	3	11	1	8	0	1	1	68
Facilidad de comprensión del sistema	4	1	28	4	7	2	9	3	7	1	1	1	68
Colores llamativos	5	0	31	1	9	0	8	4	7	1	2	0	68
Facilidad de utilización	5	0	32	0	8	1	12	0	7	1	1	1	68
Utilidad del sistema en el futuro	5	0	29	3	9	0	10	2	7	1	2	0	68

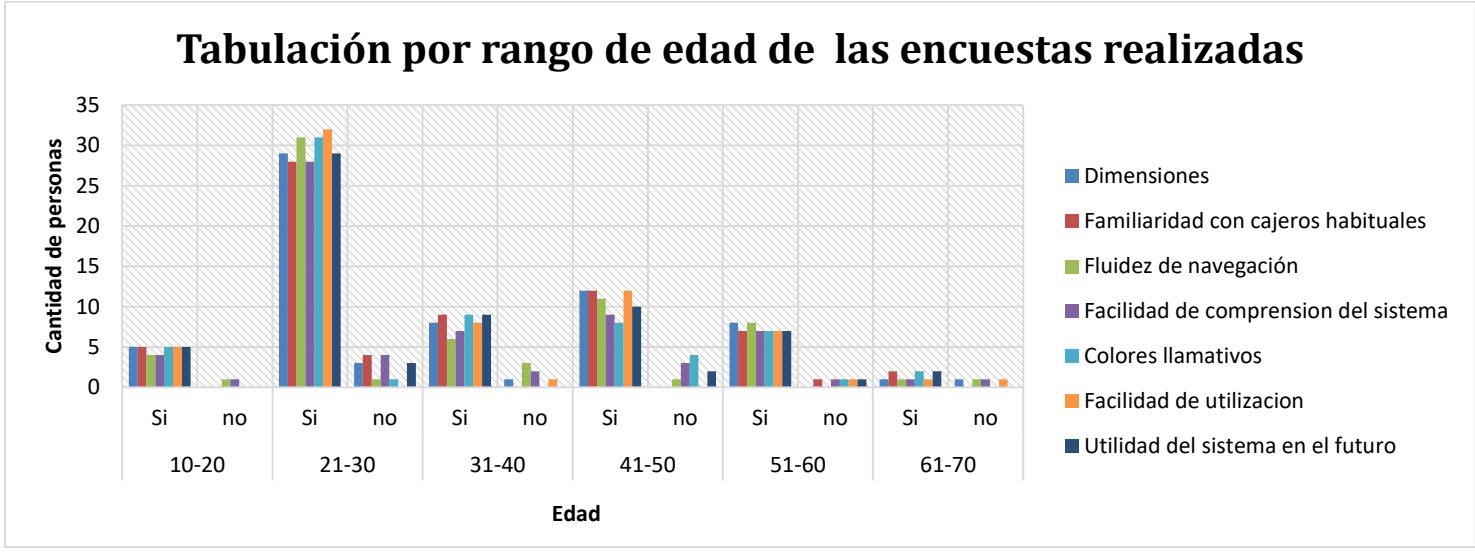

Figura 9. Tabulación de resultados por rango de edades

7. CONCLUSIONES Y TRABAJOS FUTUROS

7.1 Conclusiones

El grado de satisfacción de los usuarios con el sistema de forma general fue alto, con una aceptación de un 91,3%; cumpliéndose el objetivo de desarrollar un sistema de interacción de un cajero automático funcional y adecuado para su uso previsto.

La teoría del color es un aspecto importante en el diseño de interfaces, ya que, gracias a estos el usuario siente atracción, curiosidad y disposición por la interacción con un sistema. Además, el correcto uso de los colores sirve de guía para el usuario a través de las diferentes pantallas del sistema e incluso potencia y fortalece el impacto del mismo en los usuarios.

La experiencia de usuario es un aspecto importante porque con ella se mejora la usabilidad, accesibilidad y el placer de los usuarios al momento de utilizar el sistema, lo que permite incrementar su aceptación como producto en el mercado.

Las opiniones de los diferentes usuarios encuestados demostraron un alto interés en el desarrollo y puesta en marcha de proyectos de esta índole manifestando en varias ocasiones su desconocimiento por el uso de herramientas como el leap motion en aplicaciones diferentes al ámbito del entretenimiento.

De acuerdo a la Tabla 2, capítulo 5, el Leap motion presenta algunas desventajas como disminución de su precisión en presencia de mucha luz. Lo cual, fue corroborado en determinadas fases del proyecto en donde se tuvieron dificultades de calibración reduciendo la precisión de algunos movimientos de la persona a la hora de seleccionar una opción determinada. Sin embargo, si se toman algunas medidas en las condiciones de luminosidad presentes en el ambiente de trabajo del dispositivo como, por ejemplo,

la implementación de una cubierta de protección, puede llegar a reducir el grado de error.

El clima es otro factor que disminuye el rendimiento del leap motion. Esto se determinó en las pruebas llevadas a cabo en la ciudad de Madrid en donde se llegó a tener un clima de 42 grados centígrados, con el cual, el leap motion alcanzó una temperatura alrededor de los 52 grados centígrados haciendo que la precisión y captación de movimientos disminuyera considerablemente para finalmente desconectarse automáticamente como alternativa de seguridad.

Las plataformas como servicio (PaaS) son unas tecnologías muy robustas que agilizan el proceso de despliegue y desarrollo de un proyecto gracias a la integración continua que se puede realizar en estas.

Angular es un framework de desarrollo robusto gracias a que permite realizar estructuras modulares y consistencia del código. Además, al ser enfocado en componentes permite un mejor mantenimiento y adaptabilidad de nuevos componentes a futuro.

Spring Boot es un framework que acorta el tiempo de desarrollo en Java ya que elimina la necesidad de escribir códigos repetitivos. También permite que las aplicaciones sean flexibles y escalables a tal punto que el manteamiento e integración de módulos es más fácil de entender.

Spring boot promueve buenas prácticas de desarrollo haciendo que exista una alta cohesión y un bajo acoplamiento entre clases.

7.2 Trabajos futuros

Dentro de las líneas de investigación se plantea realizar investigación de mercado del producto con bancos y/o interesados, costos de puesta en marcha con volúmenes altos de usuarios, adaptabilidad del sistema para personas con algún grado de discapacidad y la integración de sistemas de IOT para lectura del chip de las tarjetas y seguridad ya sea biométrica o de otros tipos.

En cuanto a desarrollo, se encontró que a nivel de backend se puede realizar el aseguramiento de Endpoints, las pruebas unitarias con JUnite y Mockito, los test de integración con Mockito y Spring Boot Test y elaborar archivos de configuración de diferentes entornos como desarrollo, pre-producción y producción; a nivel de Frontend se puede realizar el aseguramiento de Endpoints, las pruebas unitarias con Karma y Jasmine asi como los Test E2E con protractor y realizar archivos de configuración de diferentes entornos como desarrollo, pre-producción y producción.

ANEXO 1. Imágenes de las pruebas del sistema con los usuarios

En este Anexo, se incluyen algunas imágenes de las pruebas de usuario realizadas en Madrid y Almería.

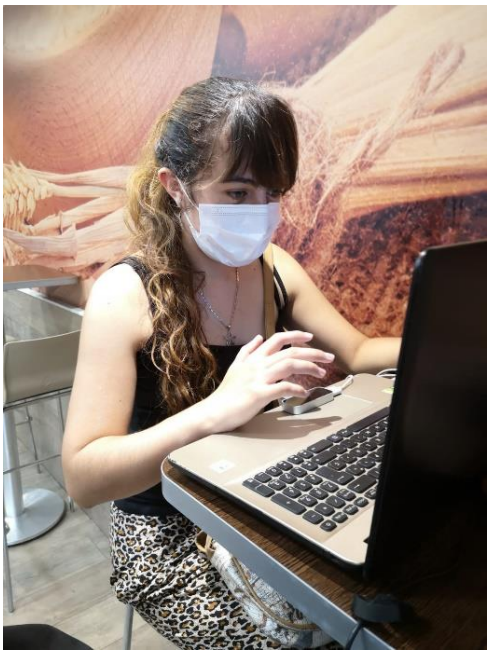

Ilustración 89. Prueba de usuario calibración de Leap Motion

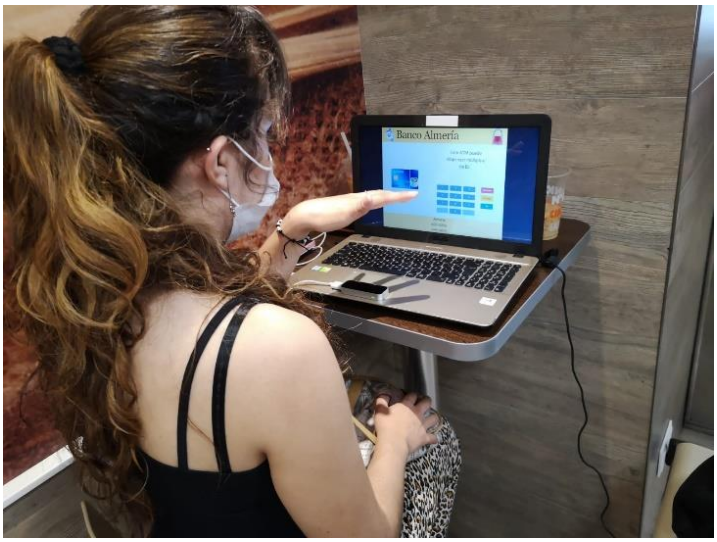

Ilustración 90. Prueba de usuario del sistema con Leap Motion

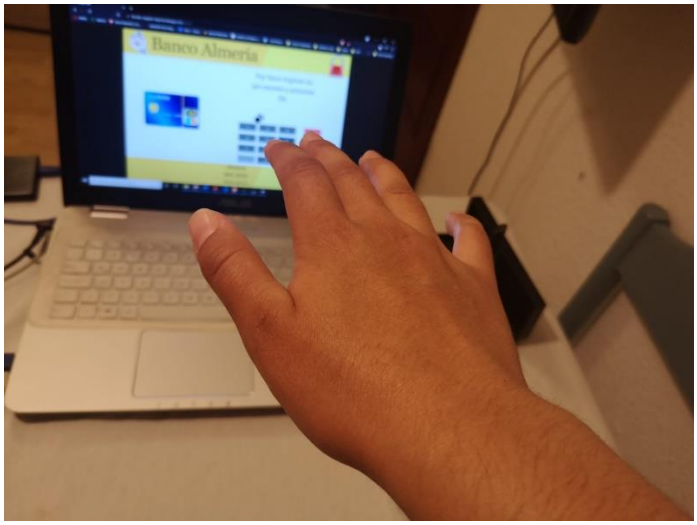

Ilustración 91. Prueba del sistema del usuario 2

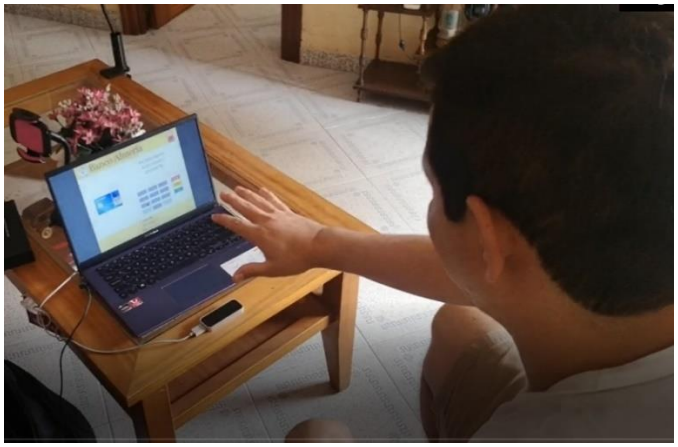

Ilustración 92. Prueba del sistema del usuario 3

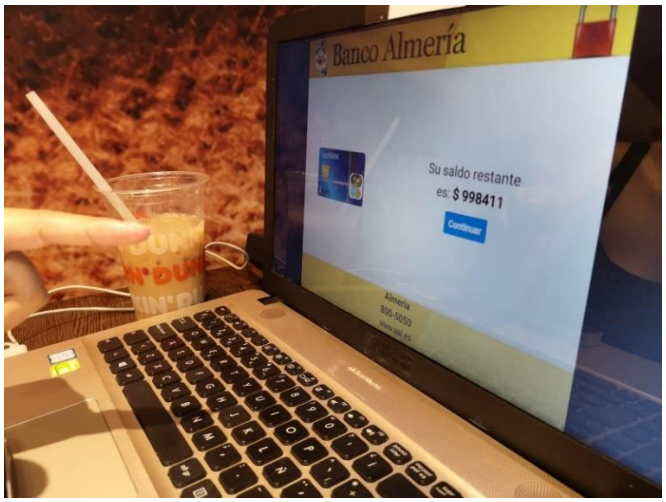

Ilustración 93. Prueba del sistema de usuario 4

Ilustración 94. Prueba del sistema de usuario 5

Ilustración 95. Prueba del sistema de usuario 6

Ilustración 96. Prueba del sistema de usuario 7

ANEXO 2. Test de evaluación del sistema

En este Anexo se incluye la encuesta presentada a los usuarios una vez se realizaron las pruebas con el sistema.

Sistema interactivo de cajero automático Leap Motion

*Obligatorio

1. Por favor escriba su nombre *

2. Por favor escriba su edad *

3. ¿El proyecto tiene las dimensiones de botones y cajas de texto acordes para visualizar la información de manera correcta? *

   Marca solo un óvalo.

   - Si
   - No

4. ¿El sistema es familiar a los cajeros habituales? *

   Marca solo un óvalo.

   - Si
   - No

5. ¿La navegación entre ventanas es fluida? *

Marca solo un óvalo.

- Si
- No

6. ¿Es fácil de comprender a primera vista es el sistema? *

Marca solo un óvalo.

- Si
- No

7. ¿Los colores del sistema son llamativos? *

Marca solo un óvalo.

- Si
- No

8. ¿Fue fácil utilizar el sistema? *

Marca solo un óvalo.

- Si
- No

9. ¿Piensa que puede llegar a ser de utilidad un sistema como este en el futuro? *

Marca solo un óvalo.

- Si
- No

BIBLIOGRAFÍA

Angular. (08 de Marzo de 2021). *Angular*. Obtenido de https://angular.io/guide/what-is-angular

B., G. (3 de Diciembre de 2020). *Hostinger*. Obtenido de https://www.hostinger.es/tutoriales/que-es-mysql#Que-es-MySQL

Benyon, D. (2014). *Designing Interactive Systems.* Harlow: Pearson Education Limited.

Burdea, G., & Coiffet, P. (2003). *Virtual Reality Technology.* Hoboken: J. Wiley-Interscience.

Cardona, a., Torres, a., Peña, y., Galeano, s., & Ardila, c. (2014). *Importancia de la teoria del color y caracteristicas. .*

Chacon, S., & Straub, B. (2021). *Pro Git.* Apress.

Cruz Vílchez, F. J. (2012). *Programación en Java.* Fundación Universitaria Andaluza Inca Garcilaso.

Elbadawy, H. M., Khattab, A., Alalawi, A., Aljohani, F. D., Sundogji, H., Mahmoud, A., . . . Suliman, B. (2021). The detection of SARS-CoV-2 in outpatient clinics and public facilities during the COVID-19 pandemic. *Journal of Medical Virology Wiley*, *93*(5), 2955-2961.

Garrote Núñez, A. (2016). *Funciones básicas para reconocimiento automático de lenguaje de signos utilizando Leap Motion.* Jaén: Universidad de Jaén .

Gholipour, S., Nikaeen, M., Manesh, R. M., Aboutalebian, S., Shamsizadeh, Z., Nasri, E., & Mirhendi, H. (2020). Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2) Contamination of High-touch Surfaces in Field Settings. *Biomed Environ Sc*, *33*(12), 925-929.

Heroku. (2021). *Heroku*. Obtenido de https://www.heroku.com/what

Java. (s.f.). *Java*. Obtenido de https://www.java.com/es/

Jetbrains. (s.f.). *Jetbrains*. Obtenido de IntelliJ Idea: https://www.jetbrains.com/es-es/idea/

Lozano Rodero, A. (2009). *Metodología de desarollo de Sistemas Interactivos inteligentes de ayuda al Aprendizaje de tareas procedimentales basados en*

realidad virtual y mixta. San sebastián : Universidad de Navarra.

Microsoft. (2021). *Visual studio code*. Obtenido de https://code.visualstudio.com/docs

Ministerio de industria, energía y turismo de España. (s.f.). *Buenas prácticas en el diseño de APIS y Linked Data.*

Oracle Corporation. (2021). *MySQL*. Obtenido de https://dev.mysql.com/doc/refman/8.0/en/introduction.html

Parramón, J. M. (s.f.). *Teoria y práctia del color.* Parramón Ediciones S.A.

Pavaloiu, B. (2017). *Leap Motion Technology In Learning.* Bucharest: Polytechnic University of Bucharest.

Schwaber, K., & Sutherland, J. (Noviembre de 2020). *Scrum Guides.* Obtenido de https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Spanish-European.pdf

Tierz López, J. (2013). *Reconocimiento e interpretación de gestos con dispositivo Leap.* Zaragoza: Universidad de Zaragoza.

Tituaña Maldonado, W. M. (2017). *Estudio de la integración de los framework bootstrap y primefaces para el desarrollo de aplicaciones web adaptativas con Java Server Faces.* Ibarra, Ecuador .

Toledano López, O. G., Vazquez Sánchez, A. A., & López del Castillo, D. C. (2018). Capa de Servicios para la plataforma de procesamiento de datos educativos masivos de la facultad 4 de la Universidad de las Ciencias Informáticas . *XVII Congreso Internacional de Informática en la Educación.* La Habana, Cuba.

TypeScript. (2021). *TypeScript*. Obtenido de https://www.typescriptlang.org/

UCloudStore. (2020). *UCloudStore*. Obtenido de https://ucloudstore.com/google-cloud-platform/

Ultraleap. (2021). *Leap Motion*. Obtenido de https://gallery.leapmotion.com/contactless-atm/

Vera Rubio, G. (2016). *Concepto de ordenador: Estructura Y Funcionamiento.* Huelva: Universidad de Huelva.