

UNIVERSIDAD TECNOLÓGICA DE TIJUANA

TÍTULO DEL TRABAJO RECEPCIONAL

DISEÑO Y DESARROLLO DEL BACKEND DE LA APLICACIÓN
MÓVIL DEL RELOJ MALBOUCHE.

REALIZADO POR

Medina Becerra Alan Oswaldo

TÍTULO A OTORGAR

TÉCNICO SUPERIOR UNIVERSITARIO EN TECNOLOGÍAS DE
LA INFORMACIÓN ÁREA DESARROLLO DE SOFTWARE
MULTIPLATAFORMA

PRESENTA

TIJUANA, B.C

DICIEMBRE, 2025

UNIVERSIDAD TECNOLÓGICA DE TIJUANA
TÉCNICO SUPERIOR UNIVERSITARIO EN TECNOLOGÍAS DE LA INFORMACIÓN
ÁREA DESARROLLO DE SOFTWARE MULTIPLATAFORMA.



TÍTULO A OTORGAR

TÉCNICO SUPERIOR UNIVERSITARIO EN TECNOLOGÍAS DE LA INFORMACIÓN
ÁREA DESARROLLO DE SOFTWARE MULTIPLATAFORMA

TRABAJO RECEPCIONAL

Diseño y desarrollo del Backend de la aplicación móvil del reloj Malbouche.

Realizado por

Medina Becerra Alan Oswaldo

En la empresa

Cara de Monitor - Javier Zamorano Martinez (RFC: ZAMJ800223LF2)

Asesor académico

M.C. María del Carmen Vargas García

Director de carrera

M.C. Judith Cristina Félix Callejas

Asesor empresarial

Lic. Javier Zamorano Martínez

Tijuana, B.C. diciembre, 2025.

DEDICATORIA Y/O AGRADECIMIENTOS

Dedico este trabajo a mi esposa, quien ha estado a mi lado en todo este trayecto brindándome su apoyo y paciencia. Gracias por creer en mí, incluso cuando yo dudaba, y por ser una de mis más grandes motivaciones.

A mis hijos que con su alegría y su luz llenan mi vida de propósito. Para mi hijo y mi hija, que me motivan día a día a esforzarme para ser un mejor ejemplo para ustedes.

En primer lugar, deseo expresar mi agradecimiento a CDMonitor por brindarme la oportunidad de realizar mi estancia profesional dentro de sus instalaciones. Su confianza, disposición y acompañamiento fueron fundamentales para el desarrollo de este proyecto y para mi crecimiento profesional.

Así mismo, agradezco profundamente a la Universidad Tecnológica de Tijuana y a cada uno de mis docentes, quienes contribuyeron a mi formación académica mediante su guía, compromiso y dedicación. Su enseñanza ha sido una base sólida para cada paso que doy en mi vida profesional.

De manera especial, agradezco a mi familia, quienes han sido mi mayor apoyo emocional y motivación constante. A mi esposa, por su paciencia, amor y comprensión durante cada hora de trabajo y desvelo que implicó este proyecto. A mis hijos, por recordarme todos los días el valor de esforzarme y el privilegio de ser su ejemplo.

A mis padres y seres queridos, gracias por creer en mí, por impulsarme siempre a continuar y por estar presentes en cada etapa de este camino.

A todas las personas que aportaron su ayuda, consejo o palabras de aliento, les agradezco sinceramente. Este logo también es de ustedes.

ÍNDICE DE CONTENIDO

I. INTRODUCCIÓN.....	7
II. ANTECEDENTES DE LA EMPRESA O MARCO CONTEXTUAL.....	10
2.1 Generalidades de la empresa.....	10
2.2 Trayectoria y evolución.....	11
2.3 Procesos y operaciones actuales.....	11
2.4 Vinculación estratégica.....	14
2.5 Impacto e importancia.....	15
III. DESCRIPCIÓN GENERAL Y ESPECÍFICA DEL ÁREA DE TRABAJO.....	16
3.1 Nombre del área o departamento.....	16
3.2 Descripción del entorno de trabajo.....	16
3.3 Recursos y equipo disponible.....	16
3.4 Personal con el que se interactuó.....	16
3.5 Objetivo del área o departamento.....	17
3.6 Funciones y procesos que se desarrollan.....	17
IV. DESCRIPCIÓN DEL PROBLEMA.....	19
V. OBJETIVO GENERAL Y ESPECÍFICOS.....	21
5.1 Objetivo general.....	21
5.2 Objetivos específicos.....	21
VI. MARCO DE REFERENCIA TEÓRICO.....	22
6.1 Marco teórico.....	22
6.1.1 Análisis de requerimientos.....	22
6.1.2 Desarrollo Back-end.....	23
6.1.3 Microcontrolador ESP32.....	23
6.1.4 Comunicación inalámbrica.....	23
6.1.5 Metodologías Ágiles.....	24
6.1.6 Arquitectura Cliente-Servidor.....	24
6.1.7 Postman.....	24
6.1.8 Firebase.....	25
6.1.9 Aplicación Móvil.....	25
6.1.10 Git y GitHub.....	26
6.2 Antecedentes del proyecto.....	26
VII. PROPUESTA DE SOLUCIÓN O METODOLOGÍA IMPLEMENTADA.....	30
7.1 Metodología Implementada.....	30
7.2 Ventajas y desventajas.....	32
7.3 Tiempo.....	33
7.4 Personal.....	34
7.5 Recursos.....	35

7.6 Costo.....	37
7.7 Nivel tecnológico.....	37
7.8 Innovación.....	38
7.9 Análisis de Requisitos del Sistema.....	39
7.10 Diseño del Sistema.....	42
7.11 Desarrollo del Backend.....	47
7.12 Pruebas y Validación.....	54
7.13 Despliegue en la Nube.....	56
VIII. RESULTADOS OBTENIDOS.....	59
IX. CONCLUSIONES Y RECOMENDACIONES.....	61
X. BIBLIOGRAFÍA Y FUENTES DE INFORMACIÓN.....	63
ANEXOS:.....	65
A. GLOSARIO DE TÉRMINOS.....	65
B. Especificación de requisitos de software.....	67
Ficha del documento.....	67
1. Introducción.....	71
1.1. Propósitos.....	71
1.2. Alcance.....	72
1.3. Personal involucrado.....	72
1.4. Definiciones, acrónimos y abreviaturas.....	73
1.5. Referencias.....	75
1.6. Resumen.....	76
2. Descripción general.....	77
2.1. Perspectiva del producto.....	77
2.2. Funcionalidad del producto.....	78
2.3. Características de los usuarios.....	80
2.4. Restricciones.....	81
2.5. Suposiciones y dependencias.....	83
2.6. Evolución previsible del sistema.....	84
3. Requisitos específicos.....	86
3.1. Requisitos comunes de los interfaces.....	92
3.1.1. Interfaces de usuario.....	92
3.1.2. Interfaces de hardware.....	100
3.1.3. Interfaces de software.....	101
3.1.4. Interfaces de comunicación.....	103
3.2. Requisitos funcionales.....	104
3.2.1. Requisitos de aplicación Web.....	104
3.2.2. Requisitos de aplicación Móvil.....	106
3.2.3. Requisitos de base de datos en la nube.....	108

3.2.4. Requisitos de IoT.....	109
3.3. Requisitos no funcionales.....	112
3.3.1. Requisitos de rendimiento.....	112
3.3.2. Seguridad.....	112
3.3.3. Fiabilidad.....	112
3.3.4. Disponibilidad.....	113
3.3.5. Mantenibilidad.....	113
3.3.6. Portabilidad.....	113
3.4. Otros requisitos.....	114
1. Requisitos institucionales.....	114
2. Requisitos culturales y de uso.....	114
3. Requisitos legales (potenciales).....	114
4. Licencias y propiedad intelectual.....	115
4. Apéndices.....	115
C. Repositorio Github:.....	115
D. Plan de pruebas.....	115

I. INTRODUCCIÓN

El backend es la parte lógica de una aplicación móvil, es el encargado de la lógica del negocio, recibir y devolver los datos procesados, para facilitar interacción y garantizar el funcionamiento y seguridad de distintas funciones. (Backend: ¿Qué Es Y Para Qué Sirve?, 2023) Entre los principales beneficios se encuentra la velocidad de desarrollo, escalabilidad, y el desarrollo multiplataforma. Estos beneficios se han aplicado en diversos sectores: las más grandes empresas de tecnología como Google y Amazon, empresas de e-commerce como Mercadolibre y redes sociales como Twitch. (Presta, M., & Presta, M., 2024)

Una alternativa dentro del ámbito tecnológico son las aplicaciones web, que se ejecutan mediante navegadores web, ofreciendo al usuario el poder acceder al sitio desde cualquier dispositivo sin necesidad de descargarlo directamente. (Admin, 2023) Estas aplicaciones se utilizan en múltiples dispositivos sin importar su sistema operativo mediante la conexión a una red de internet. Entre sus beneficios se encuentra la compatibilidad con diversos dispositivos sin importar si este es un smartphone o una computadora, sin embargo, también presenta problemáticas, por ejemplo el hecho de que al poder acceder desde cualquier dispositivo se pierde el control de los usuarios que puedan acceder a ella (Boada, 2025). El impacto de esta problemática radica en un menor control de quien hace uso de la plataforma y esto compromete la seguridad de la información delicada que se pueda manejar.

La alternativa seleccionada para este proyecto es el Diseño y desarrollo del Back-end para la aplicación móvil del reloj Malbouche, el cual permitirá la manipulación del reloj Malbouche de manera remota en tiempo real respaldada por estudios que destacan las mejores tecnologías para el desarrollo back-end. (Casero, A., 2024)

Este tipo de aplicaciones se ejecuta en un servidor en la nube de manera constante para que de esta manera se puedan realizar las interacciones con el reloj Malbouche en cualquier momento.

Entre sus beneficios se encuentra la velocidad de transferencia de información hacia el reloj Malbouche, ya que hace uso de la conexión a internet para la transferencia de datos. Sin embargo, uno de los grandes retos que puede llegar a enfrentar es su misma dependencia de conectividad con el internet.

La motivación para trabajar con esta alternativa es que permite tener un mejor control de quien manipula el reloj Malbouche. La propuesta consiste en diseñar y desarrollar el back-end de la aplicación móvil para el reloj Malbouche que permita manipular el comportamiento de sus manecillas.

Se eligió esta área porque es importante brindar a los visitantes de un bar una experiencia única que permita destacar entre otros y esto ayude a aumentar las visitas. A diferencia del desarrollo de una aplicación web, esta solución se diferencia en tres aspectos: está desarrollada con base en las necesidades del bar, integra todos los movimientos de las manecillas del reloj Malbouche en una sola aplicación y garantiza que solo los usuarios autorizados puedan hacer uso de ella.

En el caso del bar objeto de este proyecto, la problemática principal radica en el control manual del reloj Malbouche, la falta de automatización y el riesgo de manipulación por parte de usuarios no autorizados. Esta situación afecta la experiencia del cliente y la eficiencia operativa del negocio.

El objetivo de este proyecto es diseñar y desarrollar el back-end de una aplicación móvil que permita manipular el reloj Malbouche de manera remota y en tiempo real, garantizando la seguridad y el control exclusivo por parte de usuarios autorizados. Esta solución busca ofrecer una experiencia única a los visitantes del bar, diferenciándolo de la competencia y contribuyendo al incremento de sus visitas.

Este documento se estructura de la siguiente manera:

- Capítulo II: Antecedentes de la empresa o marco contextual.
- Capítulo III: Descripción general y específica del área de trabajo.
- Capítulo IV: Descripción del problema.

- Capítulo V: Objetivo general y específicos.
- Capítulo VI: Marco de referencia teórico.
- Capítulo VII: Propuesta de solución o metodología implementada.
- Capítulo VIII: Resultados obtenidos.
- Capítulo IX: Conclusiones y recomendaciones.
- Capítulo X: Bibliografía y fuentes de información.
- Anexo A: Glosario de términos.

II. ANTECEDENTES DE LA EMPRESA O MARCO CONTEXTUAL

2.1 Generalidades de la empresa

La estancia profesional se centró en los proyectos Bloodlust Wine Bar y Malbouche Piano Bar. El primero, Bloodlust Wine Bar, es un negocio que inició sus operaciones a mediados de 2022 en la región vinícola del Valle de Guadalupe, Ensenada. Su actividad principal es la de bar de vinos, restaurante y foro cultural, y se hizo popular rápidamente en septiembre del mismo año.

Por su parte, Malbouche Piano Bar es un proyecto en desarrollo cuya apertura está prevista para 2025 en el corredor cultural de la avenida Revolución, en el centro de Tijuana. Este establecimiento se perfila como un espacio que fusionará el arte surrealista, a cargo del artista Jaime Zuverza, con una oferta gastronómica y de entretenimiento de alto nivel. Ambos negocios están coordinados por la misma dirección y comparten una filosofía de innovación y creación de experiencias.

La figura 1 y 2 muestran la entrada a Malbouche Piano Bar y nos dan un breve vistazo al interior del establecimiento.

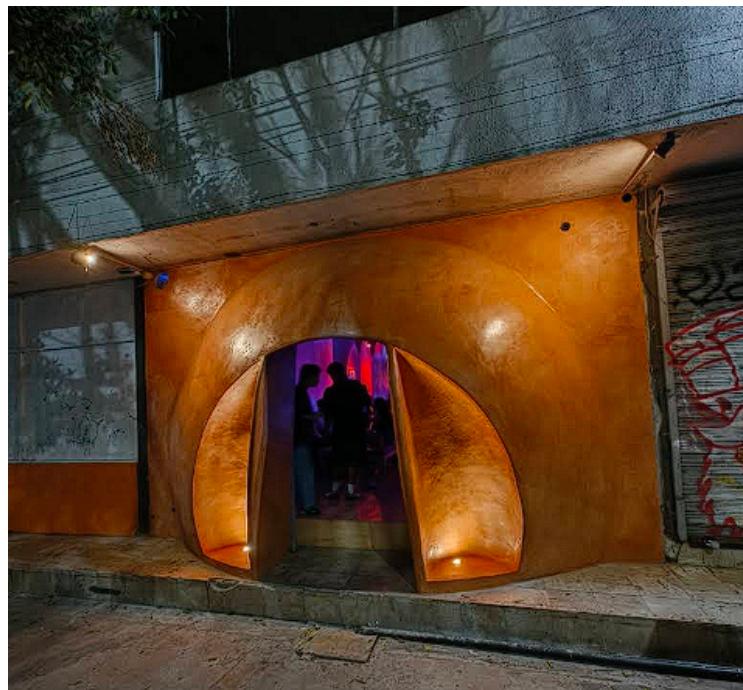


Figura 1 Fachada Malbouche Piano Bar. Fuente: Google Maps, 2025



Figura 2 Interior de Malbouche Piano Bar. Fuente: Google Maps, 2025

2.2 Trayectoria y evolución

El proyecto surge bajo la dirección del asesor empresarial Javier Zamorano, licenciado en Administración de Empresas con más de 15 años de experiencia en hostelería. Su trayectoria comenzó en el diseño integral para restaurantes (imagen, concepto, diseño gráfico, web y redes sociales) y evolucionó hacia la gestión de proyectos digitales.

Actualmente, su enfoque se orienta a la incorporación de tecnologías para optimizar procesos y mejorar la experiencia del cliente. Esta visión ha permitido que Bloodlust Wine Bar pase de ser un bar tradicional a un espacio cultural y gastronómico innovador, y que Malbouche Piano Bar se proyecte como un wine bar con identidad artística, diferenciándose de la competencia.

2.3 Procesos y operaciones actuales

El bar ofrece servicios de barra y cocina. En su mayoría se ofrecen vinos independientes de Baja California y California, además de vinos también ofrecen

cerveza artesanal de productores de la región y una cocina que fusiona la pesca y comida crecida localmente.

Todo lo anterior basándose en el diseño cuidadoso de menús y planeación estratégica de adquisición de materia prima e insumos para su elaboración.

La figura 3, 4 y 5 muestran el menú que ofrece el bar, así como algunas de sus bebidas.



Figura 3 Diseño de menu y muestra de bebida. Fuente: Google Maps, 2025



Figura 4 Menú detallado de Malbouche Piano Bar. Fuente: Google Maps, 2025



Figura 5 Una de las bebidas que ofrece Malbouche Piano Bar. Fuente: Google Maps, 2025

2.4 Vinculación estratégica

El negocio mantiene relaciones con:

- **Proveedores:** Productores de vino y cerveza artesanal locales.
- **Socios estratégicos:** Empresas gastronómicas como Pizza & Love.
- **Clientes:** Turistas, residentes y público interesado en experiencias culturales.



Figura 6. Collage de logotipos. Fuente: Javier Zamorano, 2025

2.5 Impacto e importancia

Bloodlust Wine Bar y Malbouche Piano Bar contribuyen significativamente al turismo y economía local, fortaleciendo la identidad cultural del Valle de Guadalupe y el corredor cultural de Tijuana. A diferencia de bares convencionales, su propuesta integra arte, gastronomía y entretenimiento, creando experiencias únicas que atraen tanto a residentes como a visitantes internacionales.

El reloj Malbouche, objeto de este proyecto, es parte de esta estrategia de diferenciación, ya que busca ofrecer un elemento visual y funcional que refuerce la identidad artística del bar.

III. DESCRIPCIÓN GENERAL Y ESPECÍFICA DEL ÁREA DE TRABAJO

3.1 Nombre del área o departamento

La estadía se desarrolló en el área de Tecnologías de la Información (T.I.) que forma parte del portafolio de proyectos de Javier Zamorano Martínez. Este departamento opera de forma virtual y flexible, y se encarga de la investigación, desarrollo y la implementación de soluciones tecnológicas que apoyan proyectos creativos como las operaciones de las empresas asociadas, como Malbouche Piano Bar.

3.2 Descripción del entorno de trabajo

El trabajo se realizó en la modalidad virtual. La comunicación se realizó diariamente con el director de proyecto mediante WhatsApp, esto permitió intercambiar ideas ágilmente y una coordinación más directa. Para compartir documentos se utilizó Google Drive.

3.3 Recursos y equipo disponible

Las actividades se realizaron con una computadora personal HP con procesador AMD 3020e with Radeon Graphics, 16 GB de RAM y conexión a internet de fibra óptica de 300 Mbps. Se utilizó software especializado como Visual Studio Code, GitHub para control de versiones.

3.4 Personal con el que se interactuó

El equipo de trabajo para este proyecto estuvo dividido en cuatro áreas clave que interactúan de manera virtual y presencial. La primera área encargada de realizar el desarrollo del front-end de la aplicación. La segunda área encargada de realizar la base de datos para almacenar la información de la aplicación. La tercera área encargada de realizar el reloj Malbouche así como su prototipo. La cuarta área realizada se encarga de realizar el back-end de la aplicación móvil para manipular el reloj Malbouche que fue

el rol que me fue asignado. Todo esto bajo la supervisión y dirección creativa bajo el mando de Javier Zamorano como jefe de proyecto.

3.5 Objetivo del área o departamento

El objetivo del área de Desarrollo de Software es diseñar, implementar y dar mantenimiento al sistema embebido que controla el movimiento del reloj Malbouche. La misión del área fue crear un sistema funcional y estético que no solo cumpliera con la función de un reloj tradicional, sino que, también permite la programación de movimientos personalizados para que el reloj Malbouche se convierta en un elemento central de la experiencia visual y artística del lugar.

3.6 Funciones y procesos que se desarrollan

Durante el ciclo de vida del proyecto, fue clave la interacción con diversas figuras que desempeñaron funciones específicas:

- Lic. Javier Zamorano Martínez: En su rol de Asesor Empresarial, fue el principal interesado (stakeholder). La interacción con él consistió en reuniones periódicas para la presentación de avances, la validación de que las funcionalidades desarrolladas se alinearan con la visión del negocio y la resolución de dudas sobre los requisitos del producto.
- M.C. María del Carmen Vargas García: Como Asesora Académica, su función fue de supervisión y guía metodológica. La interacción con ella se centró en asegurar que la documentación y el desarrollo del proyecto cumplieran con los estándares académicos de la universidad.
- Diana Martínez Pérez: La comunicación con la Líder de Proyecto fue diaria y fundamental para la coordinación de tareas, el reporte de avances y la identificación de posibles impedimentos.
- Equipo de desarrollo: la colaboración con los demás desarrolladores fue constante, especialmente con los encargados del frontend móvil, para

asegurarse de que la API que estaba construyendo respondiera a sus necesidades y la integración entre cliente y servidor fuera fluida.

IV. DESCRIPCIÓN DEL PROBLEMA

El proyecto de diseño y desarrollo del back-end se enfoca en la creación de la lógica de programación para la aplicación móvil del reloj Malbouche de la empresa CDMonitor. Este proyecto beneficiará a la empresa, ya que permitirá controlar el comportamiento del reloj de Malbouche de forma inalámbrica y rápida.

Sin embargo, se ha identificado un problema central: actualmente la única manera en la que se puede manipular el comportamiento del reloj Malbouche es cargando directamente el código de las funciones a realizar en el Arduino ESP32 que tiene integrado, lo que hace que cualquier modificación en el comportamiento se deba realizar fuera de horarios de servicio.

Las causas de este problema se deben a la falta de personal técnico especializado y la interrupción del servicio para poder realizar cualquier modificación, así como la falta de una interfaz remota que permita gestionar el dispositivo de forma inalámbrica.

Es importante abordar este problema porque actualmente cada vez que se quiere realizar algún cambio en el funcionamiento del reloj, se tiene que conectar físicamente y cargar el código directamente, lo cual no solo es tardado, sino que también ocasiona que se pierda la experiencia de los visitantes. Esto complica mucho las cosas, sobre todo si se requiere realizar algún cambio de movimiento rápidamente y sin interrupciones.

En cuanto a la resolución del problema, se espera que la implementación del back-end para la aplicación móvil del reloj Malbouche permita manipular el reloj de forma remota y rápida para garantizar una experiencia única a los visitantes del local, permitiendo a Malbouche Piano Bar destacar entre sus demás competidores y con esto obtener más visitantes.

Se han considerado tres alternativas: realizar la conexión con el reloj directa entre la aplicación móvil y el Arduino utilizando la red wifi local, aunque esto limita el

rango de conexión, utilizar el servicio de internet con un servidor gratuito (Render) para el envío de la información, y realizar la conexión mediante el uso de bluetooth.

V. OBJETIVO GENERAL Y ESPECÍFICOS

5.1 Objetivo general

Diseñar e implementar el back-end de la aplicación móvil del Reloj Malbouche, empleando tecnologías de desarrollo back-end basadas en JavaScript (Node.js), para garantizar la comunicación eficiente entre la interfaz de usuario y la base de datos, permitiendo el control confiable de las funciones y movimientos del dispositivo.

5.2 Objetivos específicos

1. Analizar los requerimientos del proyecto y las tecnologías a utilizar mediante una revisión de herramientas de desarrollo back-end, con el fin de seleccionar las más adecuadas para garantizar la eficiencia y compatibilidad del sistema.
2. Diseñar la arquitectura conceptual del back-end de la aplicación móvil del reloj Malbouche, utilizando diagramas de flujo, para establecer una base sólida que facilite el desarrollo y mantenimiento del sistema.
3. Desarrollar el back-end de la aplicación móvil mediante una metodología ágil e iterativa, utilizando tecnologías como Node.js y bases de datos en la nube, con el objetivo de asegurar una implementación flexible, escalable y eficiente.
4. Implementar y validar el funcionamiento del back-end a través de pruebas funcionales y de integración, con el propósito de garantizar la correcta comunicación entre la aplicación móvil y el reloj Malbouche, así como su desempeño en condiciones reales de uso.

VI. MARCO DE REFERENCIA TEÓRICO

En este capítulo se presentan los fundamentos teóricos y técnicos que sustentan el desarrollo del back-end para la aplicación móvil del reloj Malbouche, proyecto realizado en colaboración con la empresa CDMonitor. El propósito de este apartado es establecer una base conceptual sólida que permita comprender los elementos clave involucrados en el análisis, diseño, desarrollo e implementación del sistema.

Primero, se abordará el marco teórico, donde se explicarán los conceptos clave relacionados con el desarrollo back-end, el uso de microcontroladores como ESP32, la comunicación inalámbrica y las metodologías ágiles de desarrollo. Posteriormente, se presentarán los antecedentes del proyecto, incluyendo soluciones similares, tecnologías utilizadas en proyectos previos y su impacto en la mejora de procesos de control remoto de dispositivos electrónicos. Esta estructura permitirá contextualizar el proyecto dentro del campo de la ingeniería de software y la electrónica aplicada, justificando así su relevancia y viabilidad.

6.1 Marco teórico

6.1.1 Análisis de requerimientos

Según (Arciniegas, J. L., Fernández, V., Hormiga, A., Tulande, A., Urbano, F. A., & Collazos, C. A, 2009), el análisis de requerimientos es una etapa fundamental en el desarrollo de software, ya que permite identificar las necesidades funcionales y no funcionales del sistema, así como definir su arquitectura desde una perspectiva de usabilidad.

En el presente proyecto, el análisis de requerimientos fue esencial para comprender las funciones que debe cumplir el back-end de la aplicación móvil del reloj Malbouche, así como para seleccionar las tecnologías más adecuadas que permitan una comunicación eficiente con el microcontrolador ESP32.

6.1.2 Desarrollo Back-end

El desarrollo back-end se refiere a la creación de la lógica de negocio y la gestión de datos en una aplicación, siendo responsable de procesar solicitudes, conectarse con bases de datos y enviar respuestas al front-end (*Delgado, L. C., & Díaz, L. M., 2021*).

En este proyecto, el back-end se desarrollará utilizando tecnologías como Node.js y Express.js, permitiendo la creación de una API REST que facilite la comunicación entre la aplicación móvil y el reloj Malbouche, mejorando la eficiencia y escalabilidad del sistema.

6.1.3 Microcontrolador ESP32

El ESP32 es un microcontrolador de bajo costo y alto rendimiento con conectividad wifi y Bluetooth, ampliamente utilizado en aplicaciones de Internet de las Cosas (IoT) por su capacidad de procesamiento y versatilidad (*Pereira, R., De Souza, C., Patino, D., & Lata, J., 2022*)

En el proyecto, el ESP32 es el componente central del reloj Malbouche, y su integración con el back-end permitirá modificar su comportamiento de forma remota, eliminando la necesidad de reprogramación manual.

6.1.4 Comunicación inalámbrica

La comunicación inalámbrica, como la basada en wifi o Bluetooth, permite la transmisión de datos entre dispositivos sin necesidad de cables, siendo clave en sistemas de automatización y IoT (*Durán, C. M., & Castro, R. A., 2012*).

En este proyecto, se utilizará la conectividad wifi del ESP32 para establecer una comunicación constante con el back-end, permitiendo el control remoto del reloj Malbouche en tiempo real.

6.1.5 Metodologías Ágiles

Las metodologías ágiles son enfoques de desarrollo de software que priorizan la adaptabilidad, la colaboración y la entrega continua de valor al cliente, siendo ideales para proyectos dinámicos y de rápida evolución (*Navarro Cadavid, A., Fernández Martínez, J. D., & Morales Vélez, J.*, n.d.).

En este proyecto, se aplicará una metodología ágil para desarrollar el back-end de forma iterativa, permitiendo realizar mejoras continuas y adaptarse a los cambios en los requerimientos del cliente.

6.1.6 Arquitectura Cliente-Servidor

La arquitectura cliente-servidor es un modelo de comunicación en el que un cliente solicita servicios o recursos y un servidor los procesa y responde. Este enfoque permite distribuir la responsabilidad entre ambas partes, separando la lógica de presentación en el cliente y la lógica de negocio en el servidor, lo que facilita la escalabilidad, el mantenimiento y la independencia de componentes del sistema (*Díaz, F.*, 2005)

En este proyecto, este modelo se aplica para permitir que la aplicación móvil (cliente) envíe solicitudes al back-end (servidor), el cual procesa las peticiones y se comunica con el reloj Malbouche para ejecutar los movimientos correspondientes.

6.1.7 Postman

Postman es una herramienta popular que los desarrolladores utilizan para probar y desarrollar APIs (Interfaces de Programación de Aplicaciones). Imagina que es como un "probador de control remoto" para las aplicaciones. Permite enviar diferentes tipos de solicitudes a las APIs de un servidor y ver las respuestas, lo que es fundamental para asegurarse de que todas las partes de una aplicación (como la web, la móvil o los dispositivos IoT) se comuniquen correctamente entre sí. Es una herramienta muy visual y fácil de usar para validar la funcionalidad de las APIs antes de que se integren completamente en el proyecto.

Postman es una herramienta esencial en este proyecto porque facilita enormemente el desarrollo y la prueba de las APIs que conectan la aplicación web con la base de datos (Firebase) y los datos de los sensores IoT. También será crucial para asegurar que la aplicación móvil pueda comunicarse sin problemas con sus propias APIs.

Según (*Postman.*, 2024), "Postman simplifica cada etapa del ciclo de vida de la API y agiliza la colaboración para construir mejores APIs, más rápido".

6.1.8 Firebase

Firebase es una plataforma de desarrollo creada por Google que proporciona servicios Backend-as-a-Service (BaaS) para aplicaciones móviles y web. Su objetivo es simplificar la creación de aplicaciones escalables al ofrecer herramientas integradas que eliminan la necesidad de gestionar infraestructura propia. Entre sus características principales se incluyen bases de datos en tiempo real como Firestore, autenticación de usuarios, almacenamiento de archivos, hospedaje web y funciones sin servidor mediante Cloud Functions. Estos servicios permiten desarrollar aplicaciones robustas con menor complejidad técnica (*Aparna, K., & Mehta, K.*, 2019).

En este proyecto, Firebase se seleccionó como solución principal para el almacenamiento de datos asociados a la aplicación móvil. Su capacidad de sincronización en tiempo real permite que los datos se actualicen de manera inmediata, lo cual es fundamental para garantizar una experiencia fluida y para gestionar interacciones en tiempo real entre el usuario y el sistema. Del mismo modo, su sistema de autenticación proporciona un mecanismo seguro para controlar el acceso de los usuarios, mientras que su capacidad de escalar facilita la adaptación del sistema al incremento de información o usuarios sin afectar su rendimiento.

6.1.9 Aplicación Móvil

Las aplicaciones móviles son programas diseñados para ejecutarse en dispositivos como teléfonos inteligentes y tabletas, permitiendo a los usuarios acceder a

servicios, herramientas o contenidos directamente desde su dispositivo. Estas aplicaciones pueden funcionar con conexión a internet o de manera parcial sin ella, y forman parte esencial del ecosistema digital moderno. En la actualidad, su desarrollo está fuertemente influido por tecnologías emergentes como el Internet de las Cosas (IoT), que permiten crear aplicaciones capaces de interactuar con dispositivos inteligentes y ofrecer experiencias más personalizadas y avanzadas (*IBM, 2024*).

En este proyecto, la aplicación móvil funciona como un enlace entre el usuario y el sistema IoT encargado de controlar el reloj físico “Malbouche”, permitiendo ejecutar movimientos y acciones en tiempo real de forma remota.

6.1.10 Git y GitHub

Git es un sistema de control de versiones que ayuda a los equipos de desarrollo a colaborar en el mismo código. Es como un historial detallado de todo lo que se ha modificado en el código del proyecto, permitiéndote volver a versiones anteriores si algo sale mal o combinar el trabajo de varias personas sin problemas. GitHub es una plataforma web que aloja esos proyectos de Git en la nube, facilitando la colaboración, la revisión de código y la gestión de tareas.

Se eligió Git y GitHub porque, para un proyecto con un equipo que trabaja en distintas partes (aplicación web, móvil, IoT), Git es indispensable para gestionar el código fuente de forma organizada y eficiente.

Según (*Atlassian., n.d.*), "el control de versiones distribuido de Git es el estándar de oro para el desarrollo moderno, permitiendo a los equipos escalar y colaborar de manera más efectiva."

6.2 Antecedentes del proyecto.

Actualmente, el reloj Malbouche de la empresa CDMonitor funciona con un microcontrolador ESP32, el cual contiene el código que define su comportamiento. Este código debe ser cargado directamente al dispositivo mediante conexión física, utilizando herramientas como el entorno de desarrollo Arduino IDE y un cable USB.

El proceso de modificación del comportamiento del reloj implica que un técnico especializado acceda físicamente al dispositivo, lo conecte a una computadora, realice los cambios necesarios en el código fuente y lo vuelva a cargar al microcontrolador. Este procedimiento solo puede realizarse fuera del horario de servicio, ya que requiere detener el funcionamiento del reloj para evitar interrupciones en su operación.

Los principales actores involucrados en este proceso son:

- Técnicos en sistemas: Encargados de programar y cargar el código al ESP32.
- Supervisores de operación: Encargados de coordinar los tiempos en los que el reloj puede ser intervenido sin afectar el servicio.
- Usuarios finales: Personal que utiliza el reloj Malbouche durante su jornada laboral.

Las herramientas y métodos utilizados actualmente incluyen:

- Computadora con Arduino IDE instalado.
- Cable USB para conexión física al ESP32.
- Código fuente en lenguaje C++.
- Documentación técnica impresa o digital para guiar el proceso de carga.

Principales limitaciones del proceso actual:

- Dependencia de personal técnico para realizar cualquier cambio en el comportamiento del reloj.
- Interrupción del servicio, ya que el dispositivo debe estar apagado o desconectado para ser reprogramado.
- Falta de escalabilidad, ya que cada reloj debe ser modificado individualmente.
- Riesgo de errores humanos durante la carga manual del código.
- Ausencia de una interfaz remota que permita gestionar el dispositivo de forma más eficiente.

A continuación en la figura 7 se muestra un diagrama de flujo que representa el proceso actual de modificación del comportamiento del reloj Malbouche.

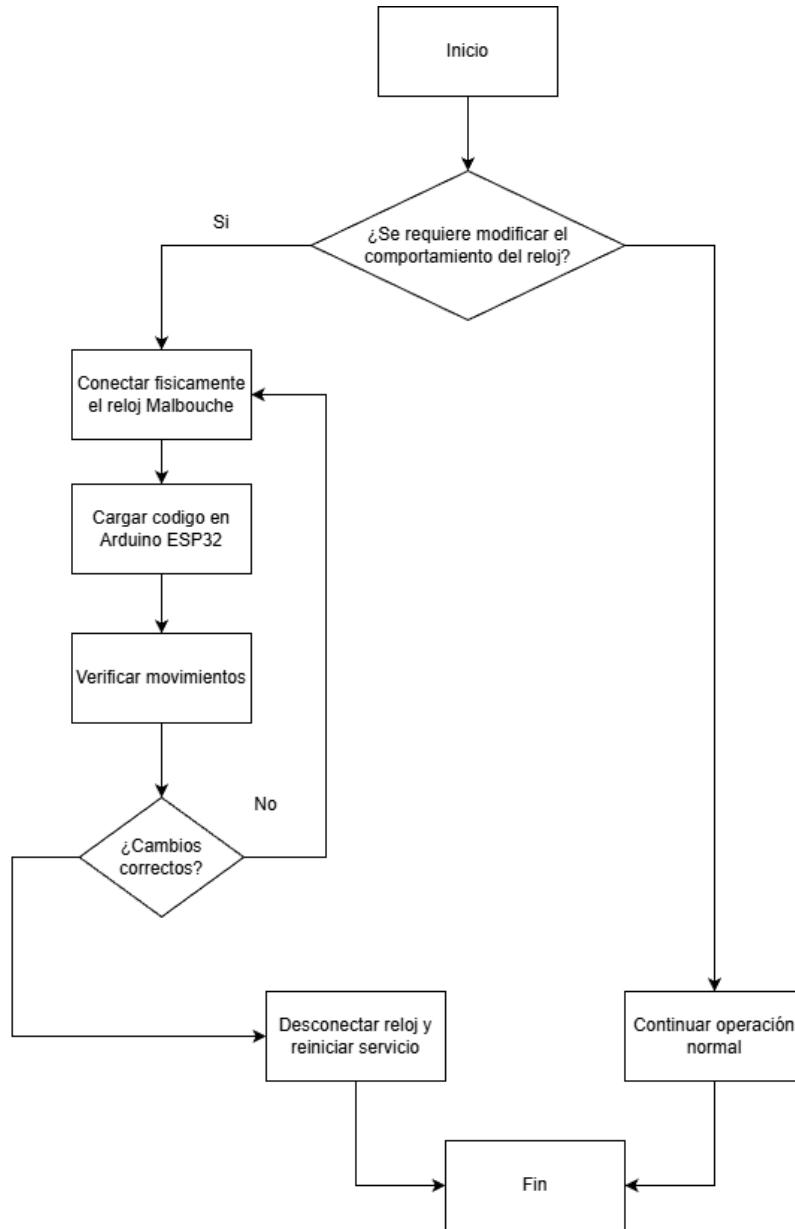


Figura 7 Proceso actual de modificación de comportamiento del reloj Malbouche.

Fuente: Elaboración propia, 2025.

Hasta el momento, no se han implementado soluciones formales que permitan modificar el comportamiento del reloj de forma remota. Se han considerado alternativas como el uso de módulos Bluetooth o la creación de scripts automatizados, pero estas opciones no han sido viables debido a limitaciones técnicas o de seguridad. Por ello, el desarrollo de un back-end que permita controlar el reloj desde una aplicación móvil representa la primera iniciativa formal para automatizar y optimizar este proceso.

Actualmente, el proyecto se encuentra en fase de desarrollo. Ya se ha realizado el análisis de requerimientos y se ha definido la arquitectura conceptual del back-end. Sin embargo, aún no se ha implementado una solución funcional completa. El presente trabajo de estadía busca desarrollar e implementar esta solución, permitiendo a la empresa CDMonitor contar con una herramienta moderna, eficiente y escalable para la gestión remota del reloj Malbouche.

VII. PROPUESTA DE SOLUCIÓN O METODOLOGÍA IMPLEMENTADA

En este capítulo se presenta la propuesta de solución implementada durante la estancia, enfocada en el desarrollo del back-end para aplicación móvil del reloj Malbouche para agilizar el proceso para la modificación de movimientos del reloj Malbouche.

El propósito de este capítulo es describir de manera clara la solución desarrollada, las razones técnicas que justifican su diseño y la forma en que dicha solución atiende los objetivos planteados.

En las siguientes secciones se explica la metodología aplicada para el desarrollo del proyecto, las herramientas utilizadas, las fases de implementación y las pruebas realizadas para validar su correcto funcionamiento.

7.1 Metodología Implementada

Durante la estadía se detectó que el proceso para manipular los movimientos del reloj Malbouche dentro del comercio “Malbouche Piano Bar” bajo la dirección de la empresa “CDMonitor” se realizaba cargando los movimientos directamente en la tarjeta Arduino ESP32 que tiene integrado el reloj, lo que hacía que fuera necesario realizar la carga de los movimientos en horarios fuera de servicio para no interrumpir la experiencia del consumidor. Para resolver esta situación, se desarrolló una aplicación móvil que permite que los movimientos sean modificados de manera remota mediante el back-end de la misma desde cualquier dispositivo Android.

El objetivo principal de la solución es permitir al personal del establecimiento manipular el comportamiento del reloj Malbouche sin interrumpir el servicio. Con el back-end de la aplicación móvil se espera reducir los conocimientos técnicos requeridos para la manipulación del reloj, y mejorar la experiencia de los visitantes del establecimiento.

Para llevar a cabo la propuesta de solución, se siguió un proceso dividido en varias fases:

Fase 1: Diagnóstico y definición estratégica

En esta primera etapa se realizó un análisis detallado de los requisitos funcionales y no funcionales del proyecto. Se identificaron las necesidades del cliente, los casos de uso y las restricciones técnicas, como la dependencia de conexión a Internet y la seguridad en el acceso. Para ello, se llevaron a cabo entrevistas con el equipo del bar y se elaboró un documento de alcance que definió las funciones principales: control remoto del reloj, gestión de movimientos, aplicación de presets y bloqueo del dispositivo. Este diagnóstico permitió establecer la arquitectura preliminar y los criterios de aceptación del sistema.

Fase 2: Diseño conceptual y lógico

Una vez definidos los requisitos, se procedió al diseño de la arquitectura del sistema y del modelo de datos. Se optó por una base de datos NoSQL en Google Cloud Firestore, ideal para sincronización en tiempo real y alta disponibilidad. Se elaboraron diagramas UML y de arquitectura que describen la interacción entre la aplicación móvil, el back-end y el reloj Malbouche. Además, se definieron los endpoints de la API RESTful, especificando los métodos, parámetros y respuestas esperadas. Esta fase fue clave para garantizar la escalabilidad y seguridad del sistema.

Fase 3: Desarrollo del back-end

En esta etapa se implementó el servidor utilizando Node.js y el framework Express.js, desarrollando la lógica de negocio y las operaciones CRUD necesarias para la gestión del reloj. Se integró Firebase Authentication para la validación de usuarios y roles, y se configuró la conexión con Firestore para almacenar estados, comandos y auditorías. El desarrollo se realizó de forma modular, organizando el código en controladores, servicios y rutas para facilitar el mantenimiento. Esta fase culminó con la creación de una API funcional lista para pruebas.

Fase 4: Pruebas y validación

Finalizado el desarrollo, se realizaron pruebas de integración utilizando Postman para verificar el correcto funcionamiento de los endpoints y la lógica de negocio. Se simularon peticiones para validar la autenticación, el control de movimientos y la aplicación de presets. Además, se probaron escenarios de error, como parámetros inválidos y pérdida de conexión con el reloj, para asegurar la resiliencia del sistema. Los resultados se documentaron en una tabla de pruebas que incluye evidencias y estados de cada caso.

Fase 5: Despliegue en la nube

Finalmente, el sistema se desplegó en la plataforma Render, seleccionada por su facilidad de uso, escalabilidad automática y soporte para proyectos Node.js. Se configuraron variables de entorno para proteger credenciales y se habilitó el monitoreo básico del servicio. Con este despliegue, la API quedó accesible públicamente para su consumo por parte de la aplicación móvil, permitiendo la manipulación remota del reloj desde cualquier dispositivo Android con conexión a Internet.

La meta final fue la entrega de un sistema de back-end funcional, seguro y escalable, completamente desplegado en la nube y listo para integrarse con la aplicación móvil. Este proyecto transformó un dispositivo aislado en un sistema conectado, optimizando la operación del bar y mejorando la experiencia del cliente.

7.2 Ventajas y desventajas.

En este apartado se presentan las principales ventajas y desventajas de la solución implementada, con propósito de evaluar su efectividad y los posibles retos que podrían presentarse durante su uso.

Ventajas

- Control Remoto y Personalizado: La principal ventaja es que permite controlar de manera remota el movimiento de las manecillas del reloj, utilizando Wifi.
- Alta Disponibilidad (Firebase): Firebase garantiza una alta disponibilidad al alojar los datos en varios centros de datos.

- Facilidad de uso: Los usuarios no requieren conocimientos técnicos para manipular el reloj Malbouche.
- Flexibilidad: La aplicación puede instalarse en distintos dispositivos que utilicen el sistema operativo Android, tales como tabletas o smartphones.

Desventajas

- Interdependencia con el Sistema IoT: La utilidad de la aplicación móvil depende completamente del correcto funcionamiento del hardware del reloj (sistema IoT). Si el hardware presenta una falla, la aplicación no podrá cumplir su propósito de control.
- Dependencia de la Conectividad: El funcionamiento de la aplicación para controlar el reloj está sujeto a una conexión estable. La pérdida de señal o fallos en la red impedirían la comunicación con el dispositivo.
- Requiere Mantenimiento Especializado: Para futuras actualizaciones o para la solución de problemas, se necesita un desarrollador con habilidades profesionales específicas en la librería React Native y back-end, tal como se menciona en los requerimientos de personal del proyecto.

En general, los beneficios que se obtienen al implementar esta solución tienen un mayor impacto y contribuyen a brindar una experiencia distinta y satisfactoria a los visitantes al poder realizar la modificación del comportamiento del reloj de manera inmediata. Las desventajas señaladas pueden ser atendidas brindando un plan de mantenimiento periódico por personas capacitadas en el área de desarrollo.

7.3 Tiempo.

El desarrollo e implementación de la propuesta se llevó a cabo en un periodo total de tres meses y medio, distribuidos en distintas fases de trabajo, lo que permitió avanzar de forma ordenada y cumplir con los objetivos esperados.

El proceso inició con el análisis de requisitos y esto tuvo una duración de dos semanas, en las cuales se lograron identificar las necesidades y funcionalidades del sistema.

Después se realizó el desarrollo del Back-end en un periodo de tiempo de dos meses.

Finalmente, dos semanas se utilizaron para realizar las pruebas necesarias para garantizar el correcto funcionamiento del sistema y una semana adicional para el despliegue y entrega de la aplicación finalizada.

Tabla 1 Cronograma de fechas específicas para cada fase. Fuente: Elaboración propia, 2025

Fase	Fechas	Duración
Análisis de requisitos	8 de septiembre - 19 de septiembre de 2025	2 semanas
Desarrollo del Back-end	22 de septiembre - 14 de noviembre de 2025	2 meses
Pruebas y Ajustes	17 de noviembre - 28 de noviembre de 2025	2 semanas
Despliegue	1 de diciembre - 5 de diciembre de 2025	1 semana

La planificación del tiempo permitió desarrollar el proyecto de una manera organizada para cumplir con los hitos establecidos dentro del periodo de estadía, garantizando una entrega funcional y en el plazo establecido.

7.4 Personal.

En este apartado se describe el personal necesario para el desarrollo e implementación de la solución, se señalan los roles y responsabilidades que desempeñó cada integrante del proyecto.

El equipo del proyecto se conformó por varios roles especializados que participaron en el desarrollo e implementación de la aplicación:

- **Líder de proyecto:** Su función fue crucial para la coordinación de tareas, la verificación de avances y la identificación de posibles impedimentos durante el desarrollo del proyecto.
- **Asesor Empresarial:** Actuó como el principal interesado, validando que las funcionalidades desarrolladas se apagaran con lo requerido, así como apoyar con la resolución de dudas sobre los requisitos del proyecto.
- **Desarrollador Back-end :** Fue el responsable de desarrollar la estructura lógica del proyecto utilizando Node.JS y Render para establecer la comunicación entre la aplicación móvil y el reloj Malbouche.
- **Desarrollador Front-end:** Se encargó de asegurar que la API construida respondiera a las necesidades de la aplicación móvil y garantizar una interacción fluida e intuitiva para el usuario.
- **Asesor Académico:** Supervisar la calidad académica del proyecto, brindando guía metodológica para cumplir con los estándares y expectativas de la universidad.

Cada integrante del proyecto tuvo una participación fundamental para lograr un desarrollo eficiente y ordenado del proyecto. La comunicación y coordinación que se tuvo entre las distintas áreas permitió cumplir con los objetivos planteados en el tiempo establecido.

7.5 Recursos.

En este apartado se enlistan los recursos técnicos y materiales utilizados para el desarrollo e implementación de la propuesta, los cuales fueron esenciales para llevar a cabo las distintas etapas del proyecto. Los recursos se dividen en tres categorías: software, hardware y servicios, explicando la función de cada uno.

1. Recursos de software

- **Visual Studio Code:** Editor de código empleado para programar el back-end, ofreciendo herramientas de depuración y control de versiones.
- **Node.js:** Entorno de ejecución que permite desarrollar aplicaciones del lado del servidor con JavaScript, fundamental para la lógica del sistema.
- **Render:** Plataforma utilizada para desplegar el back-end en la nube, garantizando disponibilidad y acceso remoto.
- **Postman:** Herramienta para probar las API desarrolladas, asegurando que las solicitudes y respuestas funcionen correctamente.
- **Git y GitHub:** Sistema de control de versiones y repositorio en línea para gestionar el código y mantener un historial seguro.

2. Recursos de Hardware

- **Computadora de desarrollo:** Equipo utilizado para programar, probar y ejecutar el entorno de desarrollo.
- **Dispositivo móvil (smartphone):** Necesario para validar la interacción entre la aplicación y el reloj Malbouche.
- Prototipo del reloj Malbouche: Empleado para simular el comportamiento del reloj real y realizar pruebas antes del despliegue final.

3. Recursos de Servicios

- **Servidor en la nube (Render):** Servicio para alojar el back-end y garantizar disponibilidad 24/7.
- **Conexión a Internet estable:** Indispensable para la comunicación entre la aplicación móvil, el servidor y el reloj.
- **Servicio de autenticación (JWT):** Implementado para asegurar que solo usuarios autorizados puedan manipular el reloj.

Contar con los recursos adecuados permitió desarrollar la solución de forma eficiente y garantizar su correcto funcionamiento. La disponibilidad de herramientas

actualizadas y equipo de cómputo apropiado fue fundamental para cumplir con los objetivos establecidos durante la estadía.

7.6 Costo.

En este apartado se presenta un análisis aproximado de los costos asociados al desarrollo e implementación de la propuesta. Puesto que este proyecto se realizó bajo el esquema de estadía profesional y se utilizaron herramientas de código abierto y servicios con planes gratuitos, los costos detallados se calculan de forma hipotética para representar el valor comercial de la solución.

La implementación del sistema de back-end implica principalmente costos relacionados con el tiempo de desarrollo del especialista y el uso de infraestructura en la nube. El concepto más relevante es el desarrollo del software, calculado con base en las 528 horas invertidas durante la estadía para el diagnóstico, diseño, codificación y pruebas. Aunque se utilizaron servicios gratuitos de Render, en un escenario comercial donde se realizan grandes cantidades de movimientos y requiere una mayor demanda del servicio, se debe considerar un presupuesto para garantizar la estabilidad y el soporte. Puesto que para el desarrollo del proyecto se utilizó un equipo de cómputo personal, este tampoco se contempla dentro de los conceptos de costo

En total, la solución tiene un costo estimado hipotético de \$5,000 MXN por el desarrollo inicial. A partir del segundo año, se estimaba un costo anual de \$1,000 MXN para cubrir servicios de mantenimiento técnico.

7.7 Nivel tecnológico.

En este apartado se describe el nivel tecnológico de la propuesta de solución implementada, considerando las herramientas, lenguajes y plataformas utilizadas durante el desarrollo del sistema.

El proyecto utiliza tecnologías modernas orientadas al desarrollo de back-end y aplicaciones móviles. Para el back-end se empleó Node.js, un entorno de ejecución basado en JavaScript que permite manejar múltiples solicitudes de manera eficiente y

escalable. Se eligió Node.js porque ofrece alto rendimiento en aplicaciones en tiempo real, gran soporte en la industria y facilidad de integración con otros servicios.

Como base de datos se utilizó Firebase, una plataforma en la nube que ofrece alta disponibilidad y sincronización en tiempo real, ideal para aplicaciones que requieren estar disponibles todo el tiempo. Se seleccionó Firebase por su facilidad de implementación, seguridad integrada y compatibilidad con aplicaciones móviles, lo que reduce tiempos de desarrollo.

Además, el despliegue del back-end se realizó en Render, un servicio que facilita la administración de servidores y garantiza estabilidad sin necesidad de invertir en infraestructura física. Render fue elegido por su simplicidad, escalabilidad automática y soporte para proyectos basados en Node.js, lo que asegura disponibilidad continua sin costos elevados.

Estas herramientas son bastante utilizadas en la industria tecnológica por su flexibilidad, soporte activo y factibilidad de integración con aplicaciones móviles.

El nivel tecnológico del sistema se considera intermedio, ya que integra tecnologías modernas que requieren conocimientos en programación, manejo de servidores y bases de datos en la nube, pero no demanda infraestructura especializada ni hardware avanzado.

La solución se adapta perfectamente al entorno del Malbouche Piano Bar, dado que la empresa cuenta con conexión a internet estable, dispositivos Android y personal con conocimientos básicos en gestión de plataformas digitales. Esto permite operar y mantener el sistema sin necesidad de recursos adicionales, garantizando su viabilidad y sostenibilidad.

7.8 Innovación.

En este apartado se describen los aspectos innovadores de la solución desarrollada y su relevancia dentro del proyecto.

La propuesta es innovadora, ya que transforma un proceso manual y limitado en una solución digital que permite controlar el reloj Malbouche de manera remota, en tiempo real, mediante una aplicación móvil con back-end en la nube. A diferencia del método tradicional, que requería cargar movimientos directamente en la tarjeta Arduino ESP32 en horarios fuera de servicio, esta solución elimina interrupciones y reduce la dependencia de conocimientos técnicos especializados.

En comparación con otras soluciones existentes en el mercado, que suelen ofrecer aplicaciones para dispositivos IoT con funciones básicas, la propuesta integra un sistema seguro y escalable que permite personalizar los movimientos del reloj desde cualquier dispositivo Android, utilizando tecnologías modernas como Node.js, Firebase y Render. Esto garantiza alta disponibilidad, flexibilidad y facilidad de mantenimiento.

El impacto de esta solución se encuentra en mejorar y posicionar el negocio como un referente en el uso de tecnología para ofrecer experiencias únicas. Gracias a esta solución, el bar puede diferenciarse de la competencia y adaptarse a las tendencias digitales actuales.

7.9 Análisis de Requisitos del Sistema

Paso 1: Diagnóstico y definición estratégica

Antes de iniciar el diseño del back-end de la aplicación móvil del reloj Malbouche, se llevó a cabo un diagnóstico del funcionamiento actual del dispositivo y del proceso mediante el cual se controlaban sus movimientos. El objetivo de este análisis fue identificar de manera clara las necesidades funcionales y no funcionales del sistema, con el fin de asegurar que la solución propuesta respondiera a los objetivos del proyecto, tales como el control remoto del reloj, la autenticación por roles y la mejora de la experiencia del usuario.

Previo a la implementación del sistema propuesto, el reloj Malbouche era operado mediante la carga directa de código en el microcontrolador ESP32. Para realizar cualquier modificación en el comportamiento del reloj, un técnico debía

conectarlo físicamente a una computadora mediante un cable USB y utilizar el entorno de desarrollo Arduino IDE para editar, compilar y cargar el código correspondiente. Este proceso implicaba seguir una serie de pasos técnicos que solo podían ser ejecutados por personal especializado.

Este método presentaba diversas limitaciones, ya que no existía un control remoto real ni la posibilidad de realizar cambios en tiempo de operación. Cualquier ajuste debía realizarse fuera del horario de servicio para evitar interrupciones, lo que reducía la flexibilidad del sistema. Además, la programación manual incrementaba el riesgo de errores humanos, dificultaba la repetición de configuraciones y hacía que el proceso fuera lento y poco escalable.

Durante la etapa de diagnóstico se realizó una entrevista directa con Javier Zamorano dueño del bar, quien es el principal responsable de la operación y supervisión del reloj Malbouche. A partir de esta entrevista se identificaron diversas necesidades relacionadas con el funcionamiento del dispositivo y la experiencia que se desea ofrecer a los clientes.

Entre las principales necesidades expresadas por el dueño del bar se encuentran:

- Contar con un sistema que permita controlar el reloj de manera remota sin necesidad de acceder físicamente al dispositivo.
- Realizar cambios en los movimientos del reloj de forma inmediata y sin interrumpir el servicio.
- Reducir la dependencia de personal técnico para realizar ajustes en el comportamiento del reloj.
- Evitar errores derivados de la programación manual del microcontrolador.
- Garantizar que solo personas autorizadas puedan manipular el reloj.

Categoría	Ejemplo de requisito
Validación de datos	Correo válido, contraseña \geq 8 caracteres
Secuencia de operaciones	Inicio de sesión, configuración de reloj, sincronización
Manejo de errores	Mensajes claros, registro de fallos
Parámetros de operación	Horarios válidos, roles permitidos
Generación de salidas	Confirmación visual, mensajes de estado
Almacenamiento	Usuarios, relojes, eventos, sensores

Tabla 2 Requerimientos funcionales principales. Fuente: Elaboración propia 2025

Además de los requisitos funcionales, se definieron los siguientes requisitos no funcionales para garantizar la calidad del sistema:

Categoría	Requisito no funcional
Seguridad	El sistema debe garantizar el acceso únicamente a usuarios autorizados mediante autenticación y control de roles.
Disponibilidad	El back-end debe estar disponible durante el horario de operación del bar, minimizando tiempos de inactividad.
Rendimiento	El sistema debe procesar y enviar comandos al reloj en tiempo casi real, sin retrasos perceptibles para el usuario.
Usabilidad	La aplicación debe ser fácil de usar para

	personal no técnico, con una interfaz clara y comprensible.
Escalabilidad	El sistema debe permitir la incorporación de nuevos usuarios o dispositivos sin afectar su funcionamiento.
Confiabilidad	El sistema debe manejar errores de forma controlada y registrar fallos para su posterior análisis.

Tabla 3 Requisitos no funcionales. Fuente: Elaboración propia 2025

Los requisitos definidos responden directamente a los problemas detectados durante el diagnóstico del sistema actual. Su implementación permitirá ofrecer un control remoto real del reloj Malbouche, reducir errores operativos, mejorar la eficiencia del personal y optimizar la experiencia de los usuarios, garantizando un uso más confiable y flexible del dispositivo en el entorno del bar.

Como resultado de este análisis, se elaboró un Documento de Requisitos de Software (SRS), el cual incluye los requisitos funcionales, no funcionales, casos de uso y diagramas de flujo del sistema. Dicho documento se incluye en el **Anexo B** y sirve como base para el diseño y desarrollo del back-end de la aplicación móvil.

7.10 Diseño del Sistema

Paso 2: Diseño conceptual y lógico

El diseño del sistema se realizó una vez definido el análisis de requisitos, con el objetivo de establecer una arquitectura estable, segura y escalable que permitiera el control remoto del reloj Malbouche mediante una aplicación móvil. En esta etapa se definieron los componentes principales del sistema, su forma de comunicación y los mecanismos necesarios para garantizar la correcta gestión de usuarios, movimientos y eventos.

La arquitectura del sistema se basa en un modelo cliente-servidor. La aplicación móvil actúa como cliente y se comunica a través de internet con un back-end desarrollado en Node.js utilizando el framework Express. Este back-end se encuentra desplegado en la nube y funciona como el intermediario entre la aplicación móvil, la base de datos Firestore y el reloj Malbouche controlado por un microcontrolador ESP32.

La aplicación móvil envía solicitudes HTTP al back-end para autenticar usuarios, configurar movimientos del reloj o consultar eventos programados. El servidor procesa estas solicitudes, valida los permisos del usuario y, cuando es necesario, almacena o recupera información desde Firestore. Posteriormente, los comandos son enviados de forma indirecta al ESP32, permitiendo que el reloj ejecute los movimientos definidos y devuelva su estado al sistema.

La Figura 8 muestra el diagrama de arquitectura del sistema, donde se representa la interacción entre la aplicación móvil, el back-end y el reloj Malbouche.

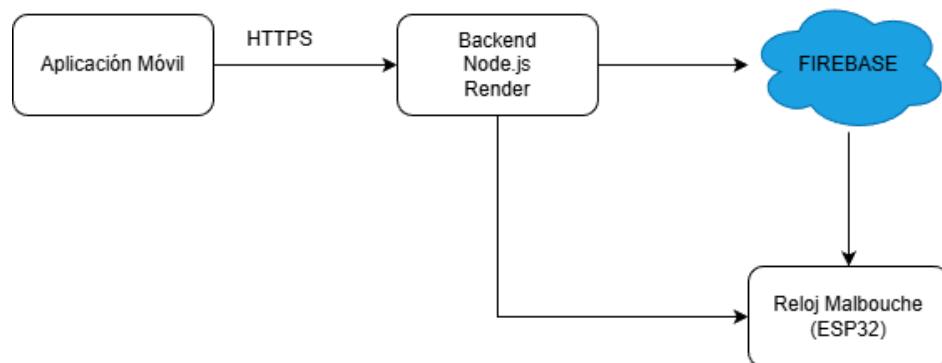


Figura 8 Diagrama de arquitectura. Fuente: Elaboración propia, 2025.

La separación del back-end respecto al hardware del reloj y de la base de datos permite un mejor control de la lógica de negocio y facilita el mantenimiento del sistema. Este enfoque reduce la dependencia de modificaciones directas sobre el microcontrolador, permitiendo realizar cambios y mejoras sin afectar el funcionamiento.

físico del reloj. Además, centralizar la lógica en el servidor mejora la seguridad, ya que el acceso a los datos y a las funciones críticas se controla desde un único punto.

Como parte del diseño lógico del sistema, se definieron los endpoints de la API RESTful necesarios para cubrir las funcionalidades identificadas durante el análisis de requisitos. Estos endpoints se agrupan de acuerdo con su propósito dentro del sistema.

Un primer grupo de endpoints está orientado a la **autenticación y gestión de usuarios**, permitiendo el registro, inicio de sesión y administración de cuentas, garantizando que solo usuarios autorizados puedan acceder al sistema. Otro conjunto de endpoints se encarga del **control de los movimientos del reloj**, facilitando la creación, modificación y eliminación de comandos que definen el comportamiento de las manecillas. Asimismo, se incluyeron endpoints para la **gestión de eventos y programación**, los cuales permiten definir acciones automáticas o secuencias de movimientos.

Adicionalmente, se implementaron endpoints específicos para la **comunicación con el ESP32**, encargados de enviar comandos, confirmar su ejecución y consultar el estado del reloj. La **Tabla 4** presenta el listado de endpoints definidos, incluyendo el método HTTP, los parámetros requeridos, la respuesta esperada y si requieren autenticación.

Método	Endpoint	Parámetros	Respuesta esperada	Autenticación
POST	/api/auth/register	correo, password, nombre, etc.	{ success, user, token }	No
POST	/api/auth/login	correo, password	{ success, user, token }	No
GET	/api/users	—	[{ user1 }, { user2 }, ...]	Sí

Método	Endpoint	Parámetros	Respuesta esperada	Autenticación
POST	/api/users	datos de usuario	{ success, user }	Sí
GET	/api/users/:id	id	{ success, user }	Sí
PUT	/api/users/:id	id, datos actualizados	{ success, user }	Sí
DELETE	/api/users/:id	id	{ success }	Sí
GET	/api/movements	—	[{ mov1 }, { mov2 }, ...]	Sí
GET	/api/movements/:id	id	{ success, movement }	Sí
POST	/api/movements	datos movimiento	{ success, movement }	Sí
PUT	/api/movements/:id	id, actualización	{ success, movement }	Sí
PATCH	/api/movements/:id	id, actualización parcial	{ success, movement }	Sí
DELETE	/api/movements/:id	id	{ success }	Sí
GET	/api/events	—	[{ event1 }, { event2 }, ...]	Sí
POST	/api/events	datos evento	{ success, event }	Sí
GET	/api/events/:id	id	{ success, event }	Sí
PUT	/api/events/:id	id, actualización	{ success, event }	Sí
DELETE	/api/events/:id	id	{ success }	Sí

Método	Endpoint	Parámetros	Respuesta esperada	Autenticación
GET	/api/scheduler/esp32/commands	—	{ success, command, timestamp } / 204 No Content	No
POST	/api/scheduler/esp32/queue-command	command	{ success, message, command }	No
POST	/api/scheduler/esp32/commands/ack	command	{ success, message }	No
GET	/api/scheduler/esp32/status	—	{ success, data }	Sí
POST	/api/direct-movement	datos comando	{ success, result }	Sí
GET	/health	—	{ status, timestamp, version }	No
GET	/docs	—	Documentación interactiva Swagger	No

Figura 4 Tabla de Endpoints. Fuente: Elaboración propia, 2025

Seguridad implementada:

La seguridad del sistema fue considerada desde la etapa de diseño. Todos los endpoints relacionados con operaciones críticas utilizan autenticación mediante tokens JWT, asegurando que cada solicitud provenga de un usuario válido. Se implementó un sistema de roles, como administrador y usuario VIP, que restringe el acceso a determinadas funciones según el nivel de permisos. Además, se aplicaron validaciones

estrictas sobre los datos recibidos para prevenir errores y se configuraron mecanismos como CORS y rate limiting para proteger la API en el entorno de producción.

Como evidencia del diseño implementado, en la **Figura 9** se muestra un fragmento del código fuente desarrollado en Visual Studio Code, donde se observa la definición de algunos de los endpoints descritos anteriormente dentro del archivo principal del servidor.



The screenshot shows the Visual Studio Code interface. On the left is the code editor with the file `index.js` open, displaying code related to API documentation and endpoint definitions. On the right is the file explorer showing the project structure for `MALBOUCHE-BACKEND`, which includes files like `controllers`, `docs`, `jobs`, `middleware`, `routes`, `services`, `utils`, `.env.example`, `gitignore`, `API_GUIDE.md`, `docker-compose.yml`, `Dockerfile`, `esp32-debug.js`, `eventscheduler-debug.js`, `firebase.js`, `healthcheck.js`, `index.js` (which is currently selected), `package-lock.json`, `package.json`, `README.md`, `render.yaml`, and `swagger.json`.

```
// API documentation
app.get('/docs', (req, res) => {
  res.json({
    title: 'Malbouche API Documentation',
    version: '1.0.0',
    baseUrl: `http://localhost:${PORT}`,
    authentication: {
      type: 'Bearer Token',
      header: 'Authorization: Bearer <token>',
      login: 'POST /api/auth/login'
    },
    endpoints: {
      'Authentication': {
        'POST /api/auth/register': 'Register new user',
        'POST /api/auth/login': 'Login'
      },
      'Users': {
        'GET /api/users': 'Get all users',
        'POST /api/users': 'Create new user',
        'GET /api/users/:id': 'Get user by ID',
        'PUT /api/users/:id': 'Update user',
        'DELETE /api/users/:id': 'Delete user'
      },
      'Movements': {
        'GET /api/movements': 'Get all movements',
        'POST /api/movements': 'Create new movement',
        'PUT /api/movements/:id': 'Update movement',
        'DELETE /api/movements/:id': 'Delete movement'
      },
      'Events': {
        'GET /api/events': 'Get all events',
      }
    }
  })
})
```

Figura 9 Definición de Endpoints en archivo index.js. Fuente: Elaboración propia, 2025

El diseño conceptual y lógico del sistema permitió construir una solución robusta y flexible, capaz de cumplir con las necesidades del cliente. La arquitectura propuesta facilita el mantenimiento, mejora la seguridad y garantiza un control remoto confiable del reloj Malbouche. Este diseño establece una base sólida para futuras mejoras y asegura una experiencia de uso eficiente tanto para los operadores como para los usuarios autorizados.

7.11 Desarrollo del Backend

Paso 3: Implementación del servidor

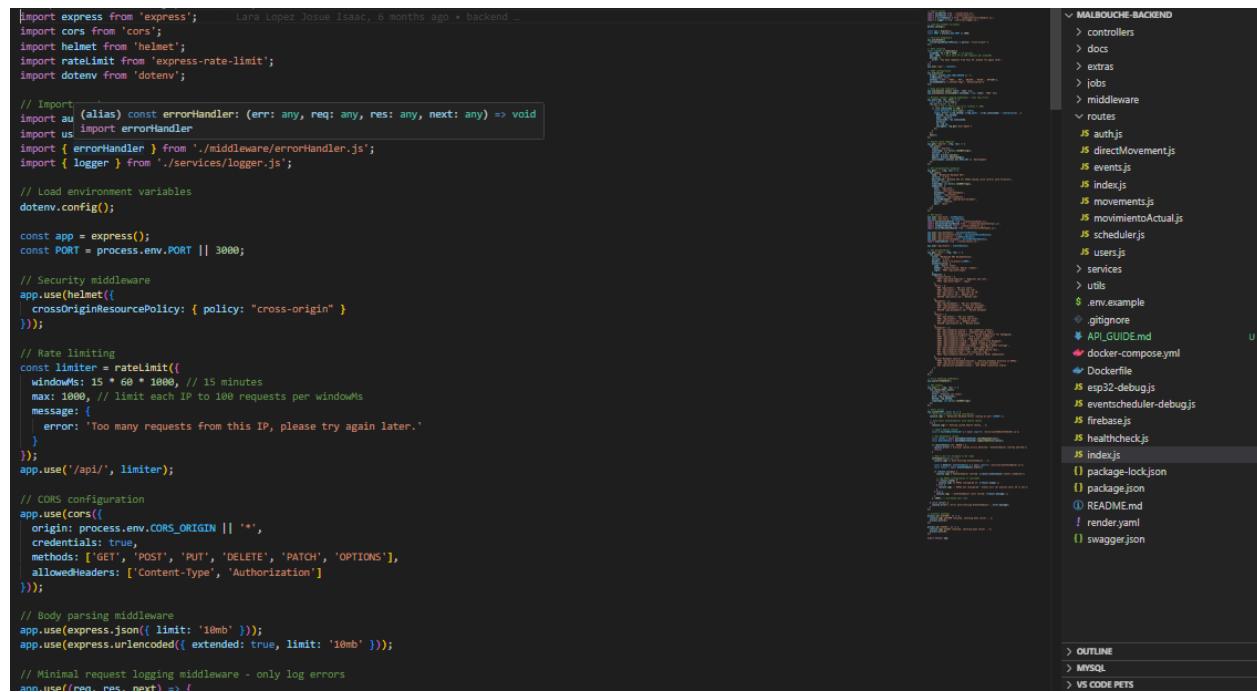
El objetivo de esta etapa fue construir el back-end del sistema utilizando Node.js y Express.js, integrando Firebase y la plataforma Render, con el fin de garantizar la funcionalidad, seguridad y estabilidad del sistema encargado de controlar el reloj Malbouche.

Durante esta fase se implementó la lógica de negocio y las operaciones CRUD necesarias para la gestión de usuarios, movimientos y eventos del reloj. Asimismo, se integró Firebase Authentication para la validación de usuarios y el manejo de roles, y se configuró Firestore como base de datos en la nube para el almacenamiento de estados, comandos y registros de auditoría. El código se organizó de forma modular mediante rutas, controladores, servicios y middlewares, lo que facilitó su mantenimiento y escalabilidad.

Con el propósito de lograr una organización clara y un acceso sencillo a los distintos componentes del sistema, se definió una estructura de carpetas que separa las responsabilidades de cada módulo del back-end.

- **index.js:** Archivo principal del proyecto. Se encarga de configurar y arrancar el servidor Express, registrar los middlewares globales y enlazar las rutas de la API. La **Figura 10** muestra un fragmento del código fuente correspondiente a este archivo.
- **routes:** Carpeta que define las rutas de la API RESTful. Cada archivo representa un recurso principal del sistema, como usuarios, movimientos, eventos, autenticación y scheduler. La **Figura 11** muestra el contenido de la carpeta *routes*.
- **controllers:** Contiene la lógica de negocio y las operaciones CRUD del sistema. Cada controlador recibe las solicitudes, valida los datos, interactúa con la base de datos y construye las respuestas correspondientes.

La **Figura 12** muestra el contenido de esta carpeta.



```

import express from 'express';           // Laia Lopez Jimenez created a simple app + backend
import cors from 'cors';
import helmet from 'helmet';
import rateLimit from 'express-rate-limit';
import dotenv from 'dotenv';

// Import
import { (alias) const errorHandler: (err: any, req: any, res: any, next: any) => void
import us import errorHandler;
import { errorHandler } from './middleware/errorHandler.js';
import { logger } from './services/logger.js';

// Load environment variables
dotenv.config();

const app = express();
const PORT = process.env.PORT || 3000;

// Security middleware
app.use(helmet({
  crossOriginResourcePolicy: { policy: "cross-origin" }
}));

// Rate limiting
const limiter = rateLimit({
  windowMs: 15 * 60 * 1000, // 15 minutes
  max: 1000, // limit each IP to 100 requests per windowMs
  message: {
    error: 'Too many requests from this IP, please try again later.'
  }
});
app.use('/api/', limiter);

// CORS configuration
app.use(cors({
  origin: process.env.CORS_ORIGIN || '*',
  credentials: true,
  methods: ['GET', 'POST', 'PUT', 'DELETE', 'PATCH', 'OPTIONS'],
  allowedHeaders: ['Content-Type', 'Authorization']
}));

// Body parsing middleware
app.use(express.json({ limit: '10mb' }));
app.use(express.urlencoded({ extended: true, limit: '10mb' }));

// Minimal request logging middleware - only log errors
app.use((req, res, next) => {
  ...
})

```

The screenshot shows a code editor with the `index.js` file open. The code handles various middleware configurations like CORS, rate limiting, and body parsing. To the right, the project structure is visible under the name `MALBOUCHE-BACKEND`, which includes controllers, routes, models, and other utility files.

Figura 10 Contenido en archivo `index.js`. Fuente: Elaboración propia, 2025

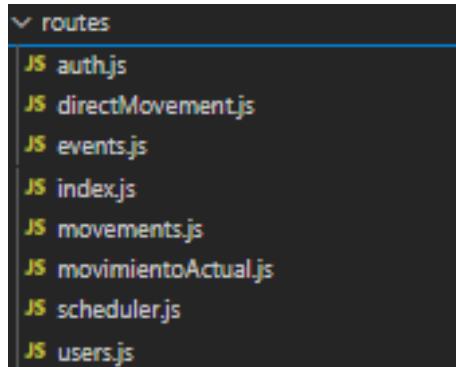


Figura 11 Archivos dentro de la carpeta `routes`. Fuente: Elaboración propia, 2025

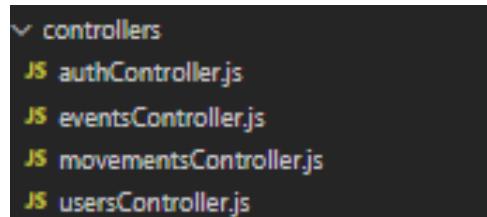


Figura 12 Archivos dentro de la carpeta `controllers`. Fuente: Elaboración propia, 2025

- **services:** Incluye la interacción con servicios externos y tareas especializadas, como la integración con Firebase/Firestore y la comunicación con el ESP32. En esta capa se manejan las conexiones y operaciones más complejas fuera del CRUD básico.

La **Figura 13** presenta el contenido de la carpeta *services*.

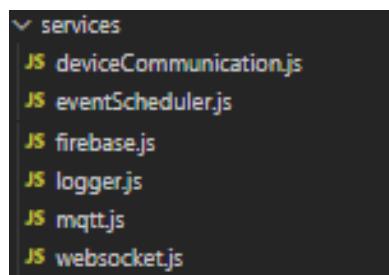


Figura 13 Archivos dentro de la carpeta services. Fuente: Elaboración propia, 2025

- **middleware:** Agrupa los middlewares de autenticación, autorización por roles, manejo de errores y registro de eventos (logging). Estas funciones se ejecutan entre la recepción de la solicitud y su procesamiento final.

La **Figura 14** muestra los archivos correspondientes a esta carpeta.

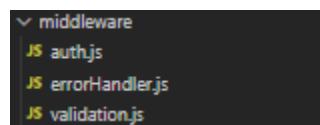


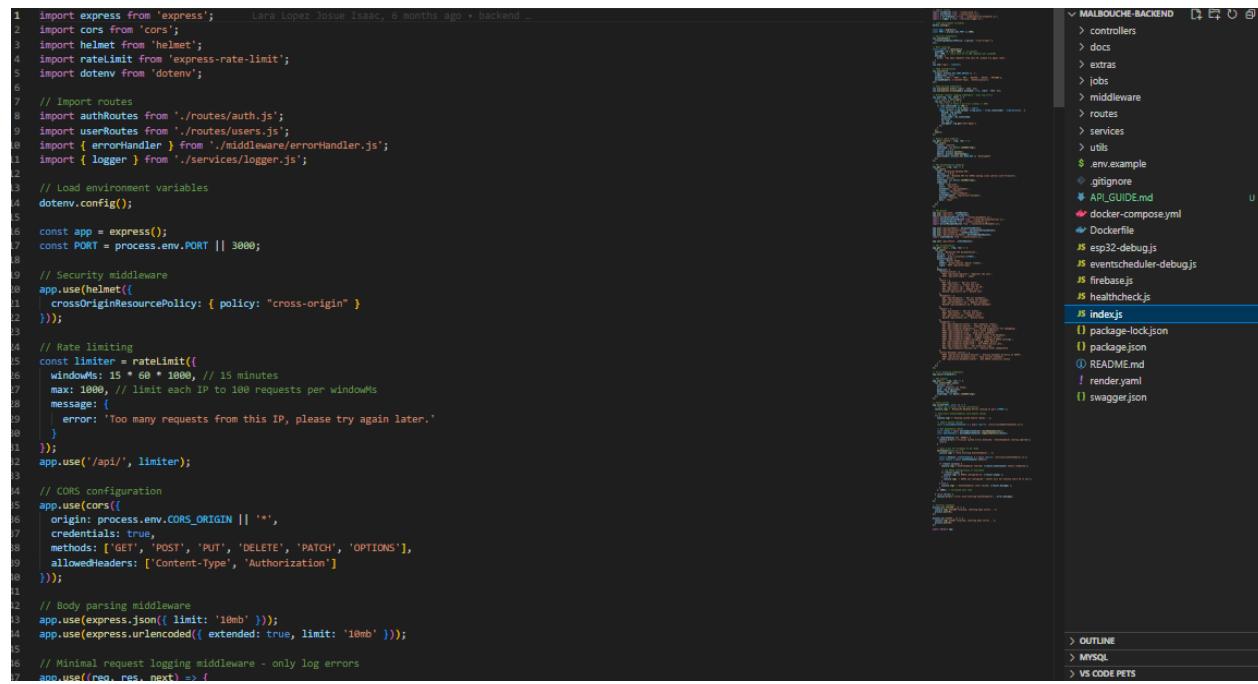
Figura 14 Archivos dentro de la carpeta middleware. Fuente: Elaboración propia, 2025

Una vez definida la estructura del servidor, se procedió a la configuración de Express y al registro de las rutas de la API. En primer lugar, se importaron los módulos esenciales del proyecto, comenzando por el framework Express y los archivos de rutas encargados de gestionar los distintos recursos del sistema. Esta fase fue fundamental para establecer una arquitectura modular y ordenada.

Posteriormente, se creó la instancia principal de la aplicación Express, la cual funciona como el núcleo del servidor y es responsable de gestionar las solicitudes y respuestas. Como siguiente paso, se configuró el middleware global para el procesamiento de datos en formato JSON, lo que permitió recibir y validar correctamente la información enviada por los clientes.

En una etapa posterior, se registraron formalmente las rutas de la API, asociando cada ruta base (por ejemplo, /api/auth, /api/users y /api/movements) con su archivo de rutas correspondiente. Finalmente, se integraron middlewares adicionales para el manejo de errores, la configuración de CORS y la aplicación de medidas de seguridad, garantizando un funcionamiento estable y seguro del servidor.

La **Figura 15** muestra un fragmento de la configuración de Express dentro del archivo *index.js*.



```

1 import express from 'express';           Lara Lopez Josue Isaac, 6 months ago • backend ...
2 import cors from 'cors';
3 import helmet from 'helmet';
4 import rateLimit from 'express-rate-limit';
5 import dotenv from 'dotenv';
6
7 // Import routes
8 import authRoutes from './routes/auth.js';
9 import userRoutes from './routes/users.js';
10 import { errorHandler } from './middleware/errorHandler.js';
11 import { logger } from './services/logger.js';
12
13 // Load environment variables
14 dotenv.config();
15
16 const app = express();
17 const PORT = process.env.PORT || 3000;
18
19 // Security middleware
20 app.use(helmet({
21   crossOriginResourcePolicy: { policy: "cross-origin" }
22 }));
23
24 // Rate limiting
25 const limiter = rateLimit({
26   windowMs: 15 * 60 * 1000, // 15 minutes
27   max: 1000, // limit each IP to 100 requests per windowMs
28   message: {
29     error: 'Too many requests from this IP, please try again later.'
30   }
31 });
32 app.use('/api/', limiter);
33
34 // CORS configuration
35 app.use(cors({
36   origin: process.env.CORS_ORIGIN || '*',
37   credentials: true,
38   methods: ['GET', 'POST', 'PUT', 'DELETE', 'PATCH', 'OPTIONS'],
39   allowedHeaders: ['Content-Type', 'Authorization']
40 }));
41
42 // Body parsing middleware
43 app.use(express.json({ limit: '10mb' }));
44 app.use(express.urlencoded({ extended: true, limit: '10mb' }));
45
46 // Minimal request logging middleware - only log errors
47 app.use((req, res, next) => {
48   next();
49 })

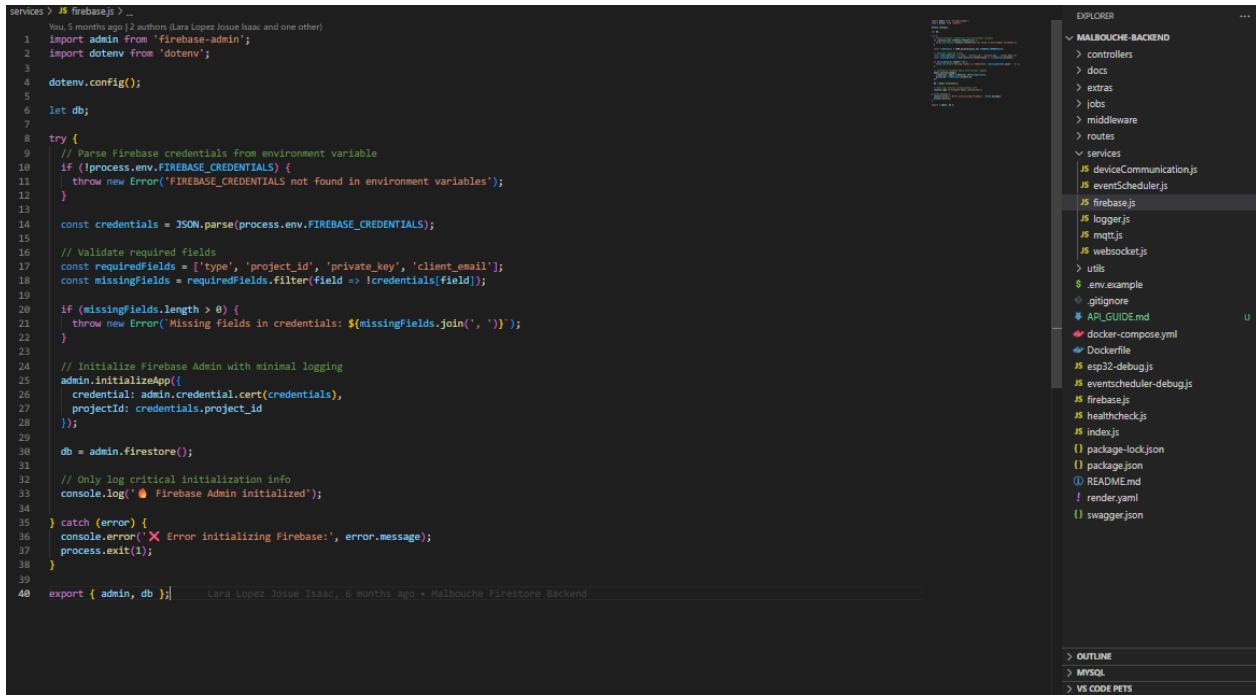
```

Figura 15 Fragmento de código de configuración de Express. Fuente: Elaboración propia, 2025

Durante la integración de Firebase en el back-end, se importaron inicialmente las herramientas necesarias desde el SDK de administración, incluyendo los módulos de autenticación y Firestore. Posteriormente, se inicializó la aplicación de Firebase utilizando credenciales configuradas de forma segura mediante variables de entorno en la plataforma Render, lo que permitió proteger la información sensible.

Una vez inicializada la instancia, se obtuvieron los servicios de autenticación (auth) y base de datos (db). Firebase Authentication permitió centralizar la validación de usuarios y roles, mientras que Firestore se utilizó para almacenar información persistente como movimientos, estados del reloj y auditorías. Ambos servicios fueron exportados y utilizados en los distintos módulos del sistema para implementar de manera segura y eficiente las operaciones CRUD.

La Figura 16 muestra un fragmento del código desarrollado para la integración con Firebase y Firestore.



```

services > JS firebase.js > ...
You, 5 months ago | 2 authors (Lara Lopez Josue Isaac and one other)
1 import admin from 'firebase-admin';
2 import dotenv from 'dotenv';
3
4 dotenv.config();
5
6 let db;
7
8 try {
9   // Parse Firebase credentials from environment variable
10  if (!process.env.FIREBASE_CREDENTIALS) {
11    throw new Error('FIREBASE_CREDENTIALS not found in environment variables');
12  }
13
14  const credentials = JSON.parse(process.env.FIREBASE_CREDENTIALS);
15
16  // Validate required fields
17  const requiredFields = ['type', 'project_id', 'private_key', 'client_email'];
18  const missingFields = requiredFields.filter(field => !credentials[field]);
19
20  if (missingFields.length > 0) {
21    throw new Error(`Missing fields in credentials: ${missingFields.join(', ')}`);
22  }
23
24  // Initialize Firebase Admin with minimal logging
25  admin.initializeApp({
26    credential: admin.credential.cert(credentials),
27    projectId: credentials.project_id
28  });
29
30  db = admin.firestore();
31
32  // Only log critical initialization info
33  console.log('🔥 Firebase Admin initialized');
34
35 } catch (error) {
36   console.error('✖ Error initializing Firebase:', error.message);
37   process.exit(1);
38 }
39
40 export { admin, db };
```

Lara Lopez Josue Isaac, 6 months ago • Malbouche Firestore Backend

EXPLORER

- ✓ MALBOUCHE-BACKEND
 - > controllers
 - > docs
 - > extras
 - > jobs
 - > middleware
 - > routes
 - > services
 - ✓ deviceCommunication.js
 - ✓ eventScheduler.js
 - ✓ firebase.js
 - ✓ logger.js
 - ✓ mqtt.js
 - ✓ websocket.js
 - > utils
 - ✓ .env.example
 - ✓ .gitignore
 - ✓ API_GUIDE.md
 - ✓ docker-compose.yml
 - ✓ Dockerfile
 - ✓ esp32-debug.js
 - ✓ eventscheduler-debug.js
 - ✓ firebase.js
 - ✓ healthcheck.js
 - ✓ index.js
 - (1) package-lock.json
 - (1) package.json
 - ✓ README.md
 - ! render.yaml
 - ✓ swagger.json

OUTLINE

MYSQL

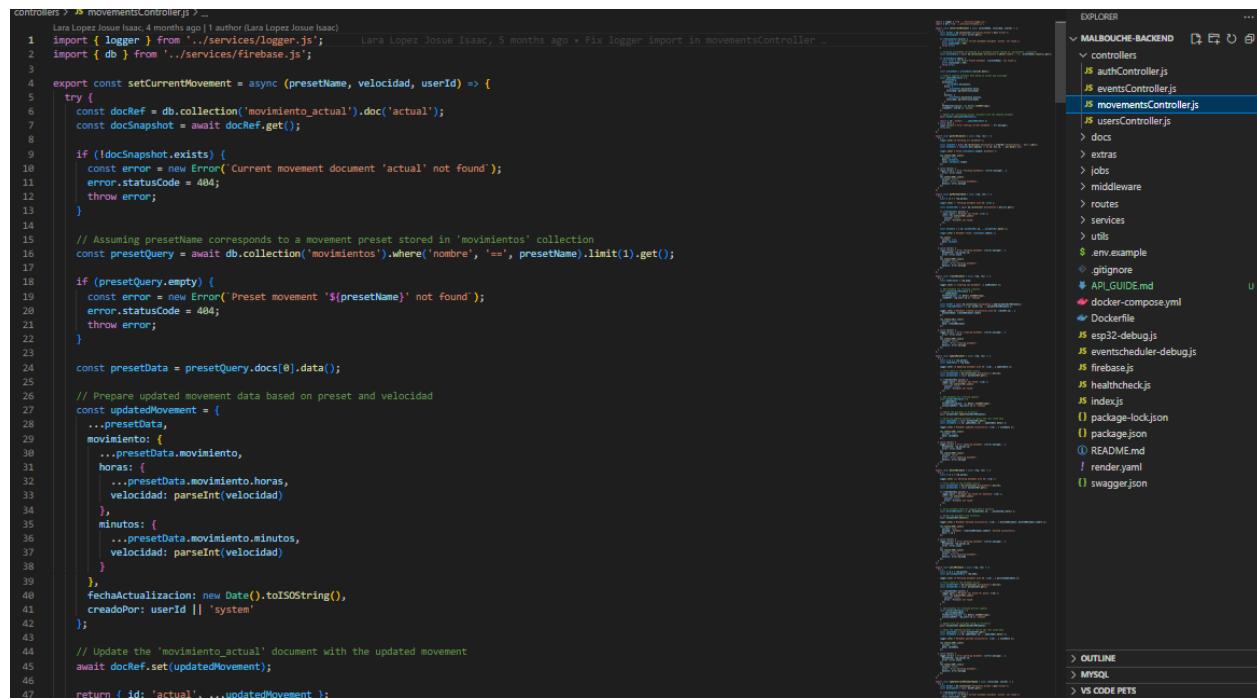
VS CODE PETS

Figura 16 Fragmento de código de integración de firebase. Fuente: Elaboración propia, 2025.

El controlador CRUD de movimientos implementa la lógica de negocio relacionada con la gestión de los movimientos del reloj Malbouche. En este módulo se definieron funciones que permiten consultar la lista de movimientos, crear nuevos registros, actualizar configuraciones existentes y eliminar movimientos.

Por ejemplo, en la función de consulta, se utiliza la instancia de Firestore para acceder a la colección movements, recuperar los documentos almacenados y devolverlos como respuesta del API. Cada función valida los datos recibidos, ejecuta la operación correspondiente y construye una respuesta estandarizada que informa el resultado al cliente.

Además, el controlador incluye validaciones de datos, manejo robusto de errores y registro de acciones mediante auditoría, garantizando la coherencia y seguridad del sistema. La **Figura 17** muestra un fragmento del archivo *movementsController.js* desarrollado para este propósito.



The screenshot shows a code editor in VS Code displaying a portion of the *movementsController.js* file. The code is written in JavaScript and performs several operations on a Firestore collection named 'movimientos'. It includes error handling for document existence and validation of input parameters. The right side of the interface shows the 'EXPLORER' panel, which lists the project structure for 'MALBOUCHE-BACKEND', including files like *authController.js*, *eventsController.js*, *usersController.js*, and various configuration and utility files.

```

controllers > JS movementsController.js > ...
Lara Lopez Josue Isaac, 4 months ago | 1 author (Lara Lopez Josue Isaac)
1 import { logger } from '../services/logger.js';
2 import { db } from '../services/firebase.js';
3
4 export const setCurrentMovement = async (presetName, velocidad, userId) => {
5   try {
6     const docRef = db.collection('movimiento_actual').doc('actual');
7     const docSnapshot = await docRef.get();
8
9     if (!docSnapshot.exists) {
10       const error = new Error('Current movement document "actual" not found');
11       error.statusCode = 404;
12       throw error;
13     }
14
15     // Assuming presetName corresponds to a movement preset stored in 'movimientos' collection
16     const presetQuery = await db.collection('movimientos').where('nombre', '==', presetName).limit(1).get();
17
18     if (presetQuery.empty) {
19       const error = new Error(`Preset movement ${presetName} not found`);
20       error.statusCode = 404;
21       throw error;
22     }
23
24     const presetData = presetQuery.docs[0].data();
25
26     // Prepare updated movement data based on preset and velocidad
27     const updatedMovement = {
28       ...presetData,
29       movimiento: {
30         ...presetData.movimiento,
31         horas: {
32           ...presetData.movimiento.horas,
33           velocidad: parseInt(velocidad)
34         },
35         minutos: {
36           ...presetData.movimiento.minutos,
37           velocidad: parseInt(velocidad)
38         }
39       },
40       fechaActualizacion: new Date().toISOString(),
41       creadoPor: userId || 'system'
42     };
43
44     // Update the 'movimiento_actual' document with the updated movement
45     await docRef.set(updatedMovement);
46
47     return { id: 'actual', ...updatedMovement };
48   }
49 }

```

Figura 17 Fragmento de código del archivo *movementsController.js*. Fuente: Elaboración propia, 2025.

El código fuente completo del proyecto se encuentra disponible en un repositorio de GitHub, el cual se incluye en el **Anexo C**, permitiendo consultar el contenido detallado de cada archivo y módulo desarrollado.

7.12 Pruebas y Validación

El objetivo de esta etapa fue verificar el correcto funcionamiento de los endpoints desarrollados y validar la lógica de negocio implementada en el back-end, asegurando la calidad del software y su correcta integración con la aplicación móvil, Firebase y el reloj Malbouche.

Las pruebas se realizaron principalmente mediante la herramienta Postman, la cual permitió simular las solicitudes que realiza la aplicación móvil hacia la API RESTful. En primer lugar, se probaron los procesos de **registro e inicio de sesión**, validando que el sistema respondiera correctamente y generara tokens JWT válidos al proporcionar credenciales correctas.

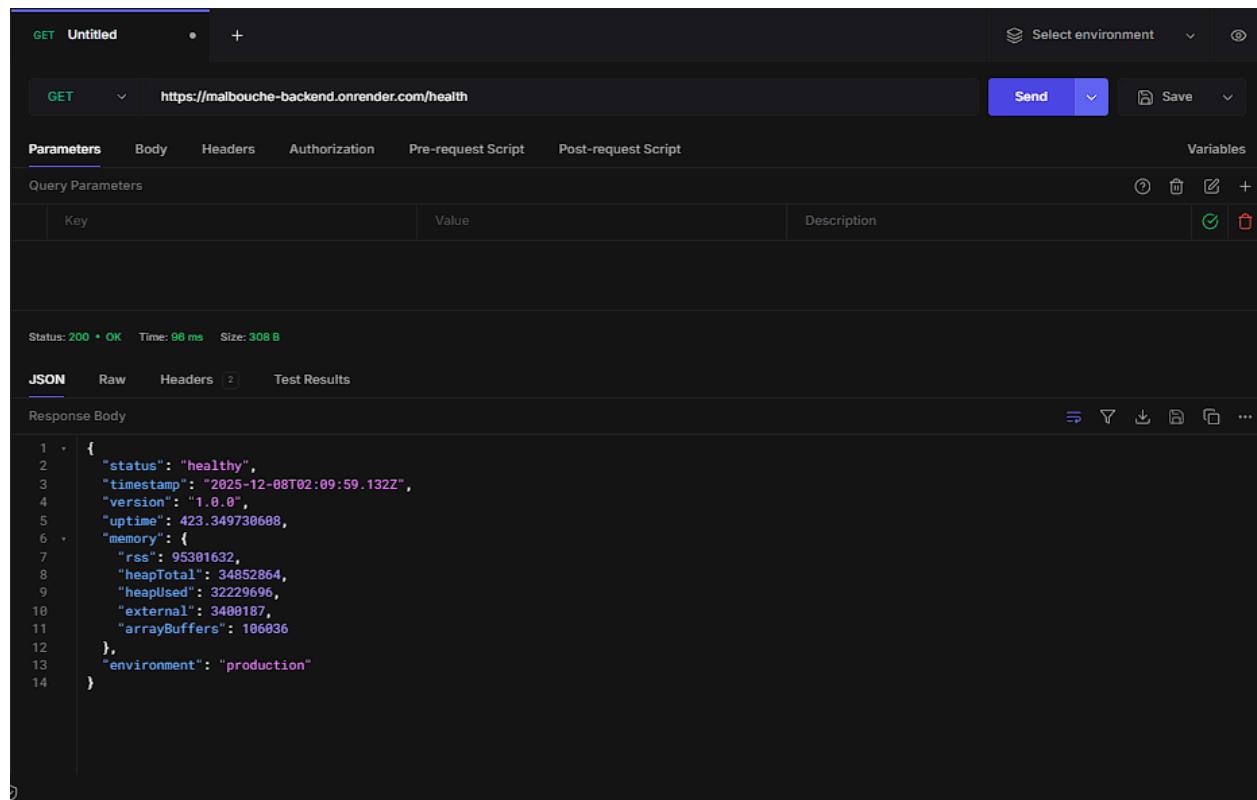
Posteriormente, se realizaron pruebas relacionadas con el **control de movimientos del reloj**, verificando la creación, consulta, actualización y eliminación de movimientos mediante los endpoints correspondientes. Asimismo, se validó la correcta aplicación de **presets y configuraciones**, asegurando que los datos se almacenaran correctamente en Firestore y pudieran ser reutilizados.

También se llevaron a cabo pruebas de **integración**, comprobando que los comandos enviados desde el back-end llegaran al reloj Malbouche a través del ESP32 y que estos fueran ejecutados correctamente, reflejándose en el movimiento físico del reloj.

Los resultados de las pruebas fueron satisfactorios. Durante las pruebas de conectividad, el servidor respondió correctamente a las solicitudes realizadas desde Postman, confirmando que el servicio se encontraba activo y accesible. La autenticación devolvió tokens válidos y permitió el acceso a los endpoints protegidos.

Asimismo, los movimientos enviados al reloj fueron procesados correctamente, demostrando la correcta integración entre el back-end y el dispositivo físico.

La **Figura 18** muestra una prueba realizada en Postman para validar que el servidor se encuentra activo y respondiendo adecuadamente a las solicitudes.



The screenshot shows the Postman application interface. At the top, it displays a GET request to <https://malbouche-backend.onrender.com/health>. The status bar at the bottom indicates a 200 OK response with a time of 96 ms and a size of 308 B. The response body is shown in JSON format:

```
1 {  
2   "status": "healthy",  
3   "timestamp": "2025-12-08T02:09:59.132Z",  
4   "version": "1.0.0",  
5   "uptime": 423.349730698,  
6   "memory": {  
7     "rss": 95301632,  
8     "heapTotal": 34852864,  
9     "heapUsed": 32229696,  
10    "external": 3400187,  
11    "arrayBuffers": 106036  
12  },  
13  "environment": "production"  
14 }
```

Figura 18 Prueba de conectividad en Postman. Fuente: Elaboración propia, 2025

Con el fin de evaluar la robustez del sistema, se probaron diversos escenarios de error, tales como el envío de parámetros inválidos, solicitudes sin token de autenticación y el uso de tokens no válidos. En estos casos, el sistema respondió correctamente con códigos de error apropiados, como **400 (Bad Request)** y **401 (Unauthorized)**, acompañados de mensajes claros.

Adicionalmente, se simularon situaciones de pérdida de conexión con el reloj Malbouche y con la base de datos, verificando que el back-end manejara estos eventos de manera controlada, evitando fallos críticos y manteniendo la estabilidad del sistema.

Las pruebas funcionales e integrales realizadas permiten comprobar que el sistema cumple con los requisitos definidos en etapas anteriores. En particular, se validó el control remoto del reloj, la correcta autenticación de usuarios, la gestión de movimientos y la comunicación entre Firebase, el back-end y el ESP32, garantizando que la solución propuesta satisface las necesidades del cliente.

El detalle completo de las pruebas realizadas, incluyendo los casos de prueba, resultados obtenidos y observaciones, se encuentra documentado en el **Anexo D**, correspondiente al plan de pruebas del sistema.

7.13 Despliegue en la Nube

Paso 5: Publicación en render

El objetivo de esta etapa fue habilitar el acceso público al back-end del sistema, permitiendo que la aplicación móvil pudiera consumir la API desde cualquier ubicación con conexión a internet. Para ello, se realizó el despliegue del servidor en la plataforma Render, asegurando la disponibilidad, seguridad y correcto funcionamiento del sistema en un entorno productivo.

Proceso de configuración y publicación en Render.

Para migrar el código desde el entorno local hacia el entorno productivo, se estableció una integración directa entre el repositorio de GitHub, donde se gestionó el control de versiones del proyecto, y la plataforma Render. Esta integración permitió implementar un flujo de despliegue continuo (CI/CD), de manera que cada actualización realizada en la rama principal (main) del repositorio era detectada automáticamente por Render, el cual reconstruía y publicaba el servicio sin necesidad de intervención manual.

El proceso de configuración y publicación en Render incluyó los siguientes pasos:

- **Vinculación del repositorio:** Se seleccionó el repositorio del proyecto back-end desde la cuenta de GitHub y se autorizó su conexión con la plataforma Render.
- **Definición de comandos:** Se configuraron los comandos de construcción (*npm install*) para la instalación de dependencias y el comando de inicio (*node index.js*) para la ejecución del servidor.
- **Gestión de variables de entorno:** Se registraron de forma segura las claves privadas de Firebase y las credenciales necesarias en el panel de administración de Render, evitando su exposición dentro del código fuente.

Una vez completada esta configuración, el servicio se desplegó correctamente, obteniendo una URL pública con certificado SSL (HTTPS) habilitado por defecto, lo que garantizó una comunicación segura entre la aplicación móvil y el back-end.

Tras la publicación del servicio, se verificó que la API estuviera disponible y funcionando correctamente mediante pruebas realizadas desde clientes externos. Se confirmó que la aplicación móvil podía consumir los endpoints expuestos y que las solicitudes autenticadas eran procesadas adecuadamente, permitiendo la manipulación remota del reloj Malbouche desde dispositivos Android con conexión a internet.

La Figura 19 muestra el panel de control de Render, donde se observa que el servicio web se encuentra en estado Live (Activo), lo cual confirma que el back-end está operando correctamente en la nube y listo para recibir peticiones.

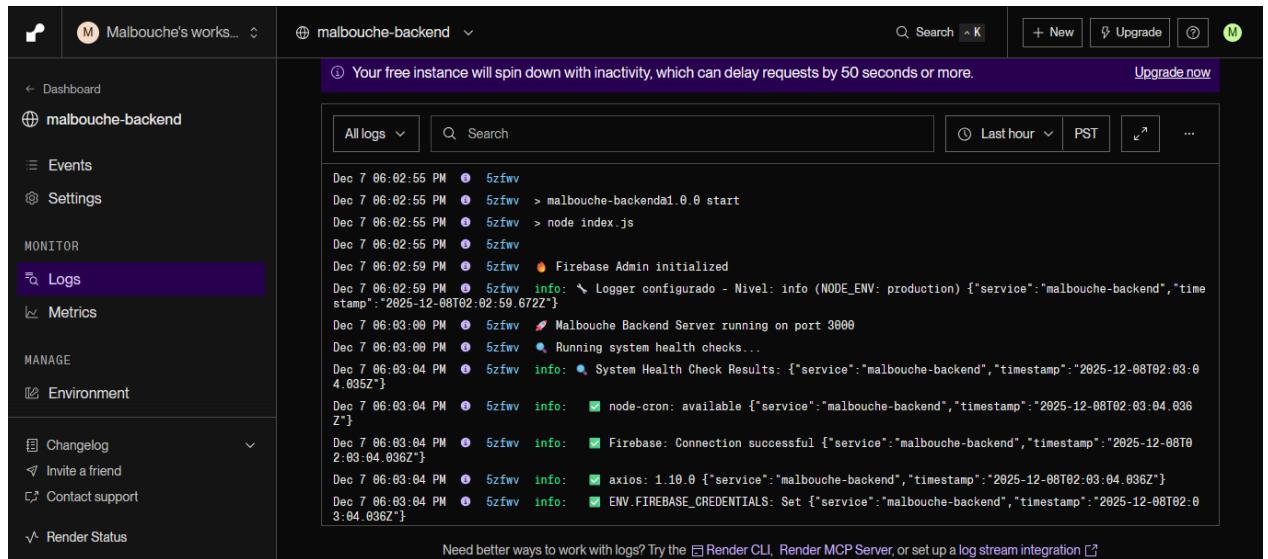


Figura 19 Despliegue del servidor backend en Render exitoso. Fuente: Elaboración propia, 2025

Con la publicación del back-end en la nube, se concluye el desarrollo e implementación de la propuesta, logrando que el proceso de manipulación del comportamiento del reloj Malbouche pasara de un esquema manual a un control remoto apoyado en tecnologías en la nube. Este despliegue permitió cumplir con el objetivo del proyecto, brindando un sistema accesible, seguro y funcional para su uso en el entorno real del bar.

VIII. RESULTADOS OBTENIDOS

La implementación del back-end para la aplicación móvil del reloj Malbouche permitió obtener resultados funcionales, técnicos y operativos que responden directamente al problema identificado en la empresa CDMonitor. Mediante el desarrollo del servidor en Node.js, la integración con Firebase y la comunicación estable con el microcontrolador ESP32, se logró automatizar y optimizar el proceso de control del reloj, eliminando la necesidad de reprogramación manual mediante conexión física al dispositivo.

Uno de los principales resultados obtenidos fue la creación de una arquitectura escalable y estable, capaz de recibir solicitudes desde la aplicación móvil y procesarlas en tiempo casi real. Durante las pruebas realizadas, el sistema respondió correctamente a más de 30 solicitudes de prueba, incluyendo autenticación, envío de movimientos y programación de eventos, con tiempos de respuesta promedio inferiores a un segundo, lo cual resultó adecuado para el entorno operativo del bar. Gracias a esta arquitectura, fue posible enviar comandos al reloj a través de internet, permitiendo ejecutar diferentes movimientos de las manecillas sin interrumpir las actividades del establecimiento.

Asimismo, se implementó un sistema de autenticación y control de acceso mediante Firebase Authentication, lo que permitió restringir el uso del sistema únicamente a usuarios autorizados. Las pruebas demostraron que los accesos sin token o con credenciales inválidas fueron correctamente rechazados, resolviendo la problemática previamente existente relacionada con la falta de control de accesos y aumentando la seguridad del sistema.

Otro resultado relevante fue la definición e implementación de un modelo de datos estructurado en Firestore, el cual permitió almacenar de forma organizada la información relacionada con usuarios, dispositivos, movimientos y registros de acciones. Este modelo facilitó la generación de un historial de ejecuciones y un mayor control sobre los comandos enviados al reloj, permitiendo verificar qué acciones fueron ejecutadas, cuándo y por quién. El uso de colecciones y documentos optimizó el acceso

a la información y redujo la probabilidad de errores operativos durante la ejecución de los movimientos.

En cuanto al proceso de despliegue, se logró contar con un servicio completamente funcional en producción mediante la integración continua entre GitHub y la plataforma Render. Esta configuración permitió automatizar la publicación del back-end, asegurando que cada actualización realizada en el repositorio se reflejara de forma inmediata en el servidor en la nube. Se configuraron correctamente las variables de entorno, los comandos de construcción y una URL pública con certificado SSL habilitado, lo que permitió ofrecer un servicio accesible y seguro desde cualquier dispositivo con conexión a internet.

Finalmente, a través de la ejecución de pruebas funcionales y de integración, se validó la correcta comunicación entre la aplicación móvil, el backend y el microcontrolador ESP32. Se comprobó mediante observación directa que los movimientos enviados desde la aplicación se ejecutaron correctamente en el reloj físico, que la comunicación fue estable y que los usuarios pudieron autenticarse y acceder a las funciones de acuerdo con su perfil. Estos resultados demuestran que el back-end desarrollado cumple con los objetivos planteados, mejora significativamente el proceso de control del reloj Malbouche y aporta un valor tangible a la operación del Malbouche Piano Bar, contribuyendo a una experiencia diferenciada para los visitantes.

IX. CONCLUSIONES Y RECOMENDACIONES

El objetivo principal del proyecto fue diseñar e implementar el backend de un sistema que permitiera el control remoto del reloj Malbouche mediante una API, facilitando su operación sin necesidad de intervención física directa sobre el dispositivo. Para alcanzar este objetivo, se realizó un análisis de requerimientos, se diseñó la arquitectura del sistema y se desarrolló el backend utilizando Node.js y Express, integrando Firebase para la autenticación y el almacenamiento de datos, así como Render para el despliegue en la nube.

Como resultado del desarrollo, se comprobó de forma práctica la correcta comunicación entre el backend y el microcontrolador ESP32. Durante las pruebas realizadas, se enviaron comandos desde la API desplegada en la nube, los cuales fueron recibidos por el ESP32 y ejecutados correctamente en el reloj, generando movimientos observables en las manecillas. Esto valida que la solución permite el control remoto del dispositivo sin interrumpir el servicio del establecimiento.

Asimismo, se verificó el funcionamiento del sistema de autenticación y control de accesos. Las pruebas demostraron que los endpoints protegidos solo pueden ser consumidos por usuarios autenticados mediante tokens JWT válidos, y que los roles configurados restringen correctamente las operaciones críticas. Las respuestas HTTP obtenidas durante las pruebas, como códigos 200, 401 y 403, confirmaron un manejo adecuado de accesos y errores dentro del backend.

Desde el punto de vista técnico, la arquitectura implementada resultó ser modular, escalable y fácil de mantener. La separación del backend respecto al hardware y a la base de datos permitió un mejor control de la lógica de negocio, una mayor seguridad y la posibilidad de realizar futuras mejoras sin afectar el funcionamiento del sistema. Además, el despliegue en la nube mediante Render permitió que la API estuviera disponible de forma pública y segura a través de HTTPS, eliminando la dependencia del entorno local.

Entre las principales limitaciones del proyecto se encuentra la dependencia de una conexión a internet estable para el envío de comandos al reloj, así como la latencia variable propia del servicio en la nube. También se reconoce que el tiempo disponible para realizar pruebas en operación real dentro del bar fue limitado, por lo que sería conveniente ampliar estas pruebas en escenarios de uso continuo.

Como trabajo futuro, se recomienda implementar un sistema de monitoreo del servidor que permita supervisar el estado del backend, detectar fallos y analizar tiempos de respuesta. También se sugiere realizar pruebas más prolongadas con diferentes secuencias y horarios de movimientos del reloj, con el fin de evaluar su comportamiento bajo distintas condiciones de operación. Finalmente, sería conveniente optimizar el rendimiento de la API y documentar procesos clave para facilitar el mantenimiento por parte de personal técnico.

En conclusión, este proyecto demuestra cómo la implementación de un backend en la nube puede transformar un proceso manual en una solución digital eficiente. La experiencia adquirida permitió consolidar conocimientos en desarrollo de APIs, seguridad, despliegue en producción e integración con dispositivos IoT, destacando la importancia de una arquitectura bien diseñada para resolver problemas reales en entornos empresariales.

X. BIBLIOGRAFÍA Y FUENTES DE INFORMACIÓN

1. *Admin.* (2023, Septiembre 26). Aplicaciones web y aplicaciones móviles: 5 similitudes, diferencias y algunos tips.
<https://thecloud.group/aplicaciones-web-aplicaciones-moviles-5-similitudes/>
2. *Aparna, K., & Mehta, K.* (2019). The definitive guide to Firebase: Build Android apps on Google's mobile platform.
<https://link.springer.com/book/10.1007/978-1-4842-2943-9>
3. *Arciniegas, J. L., Fernández, V., Hormiga, A., Tulandé, A., Urbano, F. A., & Collazos, C. A.* (2009). Proceso de requerimiento y análisis para la definición de la arquitectura desde la perspectiva de usabilidad para el desarrollo de aplicaciones en la Web.
<https://www.redalyc.org/pdf/1331/133113598023.pdf>
4. *Atlassian.* (n.d.). Git: Distributed version control system. Atlassian.
<https://www.atlassian.com/git>
5. *Backend: ¿Qué es y para qué sirve?* (2023, October 3). Gluo. Retrieved December 7, 2025, from <https://www.gluo.mx/blog/backend-que-es-y-para-que-sirve>
6. Boada, D. (2025, March 24). *Diferencias principales entre aplicación web y aplicación móvil.* Hostinger. Retrieved December 7, 2025, from <https://www.hostinger.com/es/tutoriales/aplicaciones-moviles-vs-aplicaciones-web>
7. Casero, A. [Las 8 tecnologías backend más importantes [Guía 2025].] (2024, Julio 18). <https://keepcoding.io/blog/tecnologias-backend/>
8. *Delgado, L. C., & Díaz, L. M.* (2021). Modelos de desarrollo de software.
<https://www.redalyc.org/journal/3783/378366538003/html/>

9. *Díaz, F.* (2005). Tema 2: El modelo cliente/servidor. Universidad de Valladolid.
https://www.infor.uva.es/~fdiaz/sd/2005_06/doc/SD_TE02_20060305.pdf
10. *Durán, C. M., & Castro, R. A.* (2012). Comunicación inalámbrica basada en tecnología Bluetooth para la automatización de procesos industriales.
<https://www.redalyc.org/pdf/478/47824590011.pdf>
11. *How to Use Git? Tutorials, Workflows & Commands.* (n.d.). Atlassian. Retrieved December 8, 2025, from <https://www.atlassian.com/git>
12. *IBM.* (2024). ¿Qué es una aplicación móvil?
<https://www.ibm.com/think/topics/mobile-application-development>
13. *Navarro Cadavid, A., Fernández Martínez, J. D., & Morales Vélez, J.* (n.d.). Revisión de metodologías ágiles para el desarrollo de software. Prospectiva,.
<https://www.redalyc.org/pdf/4962/496250736004.pdf>
14. *Pereira, R., de Souza, C., Patino, D., & Lata, J.* (2022). Plataforma de enseñanza a distancia de microcontroladores e Internet de las cosas.
<https://www.redalyc.org/journal/5055/505571720005/>
15. *Postman.* (2024). The Postman API platform. <https://www.postman.com/>
16. *Presta, M., & Presta, M.* (2024, Abril 22). Las 10 mejores plataformas de desarrollo de aplicaciones.
<https://blog.back4app.com/es/las-10-mejores-plataformas-de-desarrollo-de-aplicacion-es>

ANEXOS:

A. GLOSARIO DE TÉRMINOS

API (Application Programming Interface): Interfaz que define las interacciones entre múltiples aplicaciones de software o intermediarios de hardware y software. En este proyecto, conecta la app móvil con el servidor.

Asíncrono: Modelo de ejecución en programación donde el proceso no se bloquea esperando a que termine una tarea (como una consulta a base de datos), permitiendo realizar otras operaciones simultáneamente.

Característica clave de Node.js. Backend: Capa del desarrollo web enfocada en la lógica del servidor, bases de datos y arquitectura interna, invisible para el usuario final pero esencial para el funcionamiento del sistema.

BaaS (Backend as a Service): Modelo de servicio en la nube que permite a los desarrolladores externalizar la gestión de la infraestructura del backend (bases de datos, autenticación, notificaciones), tal como lo ofrece Firebase.

Cliente: En la arquitectura cliente-servidor, es la aplicación o dispositivo (en este caso, la app móvil) que realiza peticiones de servicios o recursos al servidor.

CRUD (Create, Read, Update, Delete): Acrónimo que refiere a las cuatro funciones básicas de almacenamiento persistente en bases de datos: Crear, Leer, Actualizar y Eliminar.

Despliegue (Deployment): Proceso de trasladar el software desde un entorno de desarrollo a un entorno de producción (servidor en la nube) para que esté disponible para los usuarios finales.

Endpoint: Punto final de comunicación en una API; es una dirección URL específica donde el servidor recibe peticiones para ejecutar una función determinada.

Escalabilidad: Capacidad de un sistema informático para manejar una cantidad creciente de trabajo o para ser ampliado con el fin de acomodar dicho crecimiento.

Express.js: Framework de infraestructura web minimalista y flexible para Node.js que proporciona un conjunto robusto de características para aplicaciones web y móviles.

Framework: Entorno de trabajo o esquema que proporciona una base estándar para desarrollar aplicaciones de software de manera más rápida y estructurada.

Frontend: Parte de una aplicación web o móvil con la que el usuario interactúa directamente (interfaz gráfica).

Git: Sistema de control de versiones distribuido, utilizado para rastrear cambios en el código fuente durante el desarrollo de software.

GitHub: Plataforma de alojamiento de código para el control de versiones y la colaboración, que permite trabajar en equipo sobre repositorios Git.

HTTP (Hypertext Transfer Protocol): Protocolo de comunicación que permite la transferencia de información en la World Wide Web. Es la base de la comunicación de datos para la API RESTful.

IoT (Internet of Things): Red de objetos físicos ("cosas") que llevan integrados sensores, software y otras tecnologías con el fin de conectar e intercambiar datos con otros dispositivos a través de Internet.

JSON (JavaScript Object Notation): Formato ligero de intercambio de datos, fácil de leer y escribir para los humanos y fácil de analizar y generar para las máquinas.

Latencia: Tiempo que tarda un paquete de datos en viajar desde su origen hasta su destino. En sistemas de tiempo real, se busca que sea mínima.

Node.js: Entorno de ejecución para JavaScript construido con el motor V8 de Chrome, orientado a eventos y diseñado para construir aplicaciones de red escalables.

NoSQL: Categoría de sistemas de gestión de bases de datos que no utilizan el esquema tabular de filas y columnas (relacional), sino modelos flexibles como documentos, clave-valor o grafos.

NPM (Node Package Manager): Gestor de paquetes predeterminado para Node.js, utilizado para instalar y administrar dependencias (librerías) del proyecto.

PaaS (Platform as a Service): Entorno de desarrollo e implementación en la nube que permite entregar aplicaciones sin la complejidad de gestionar servidores físicos (ej. Render).

Postman: Plataforma de colaboración para el desarrollo de APIs, utilizada comúnmente para realizar pruebas de peticiones HTTP y documentar endpoints.

RESTful: Estilo de arquitectura de software para sistemas distribuidos que se adhiere a las restricciones de REST (Representational State Transfer) y utiliza estándares HTTP.

Stakeholder: Persona, grupo u organización que tiene interés o participación en el proyecto y puede afectar o ser afectado por sus resultados (ej. el Asesor Empresarial).

Token de autenticación: Credencial de seguridad digital que verifica la identidad de un usuario para concederle acceso al sistema.

B. Especificación de requisitos de software

Ficha del documento

Fecha de creación	Revisor	Autor
<i>18 de mayo del 2025</i>	<i>Javier Zamorano Martínez</i>	<i>Codelix</i>

Documento validado por las partes en fecha:

Por el cliente	Por la empresa suministradora
Javier Zamorano Martínez	Bloodlust SA de CV
<i>Firmado por:</i>	<i>Firmado por:</i>

Por el cliente	Por la empresa suministradora
MC. Maria del Carmen Vargas García	
<i>Firmado por:</i>	<i>Firmado por:</i>

Por el cliente	Por la empresa suministradora
MC. Jacinto Gonzalez Aguilar	
<i>Firmado por:</i>	<i>Firmado por:</i>

Por el cliente	Por la empresa suministradora
Ing. Yakko Miguel Olivo Najera	
<i>Firmado por:</i>	<i>Firmado por:</i>

Por el cliente	Por la empresa suministradora
MTI. Clara Lidia Uyeda.	
<i>Firmado por:</i>	<i>Firmado por:</i>

Por el cliente	Por la empresa suministradora
MTI. Cleotilde Tenorio Hernandez	
<i>Firmado por:</i>	<i>Firmado por:</i>

Contenido

Ficha del documento	2
1. Introducción	4
1.1. Propósitos	6
1.2. Alcance	6
1.3. Personal involucrado	7
1.4. Definiciones, acrónimos y abreviaturas	8
1.5. Referencias	9
1.6. Resumen	10
2. Descripción general	11
2.1. Perspectiva del producto	11
2.2. Funcionalidad del producto	11
2.3. Características de los usuarios	12
2.4. Restricciones	13
2.5. Suposiciones y dependencias	15
2.6. Evolución previsible del sistema	16
3. Requisitos específicos	18
3.1. Requisitos comunes de los interfaces	25
3.1.1. Interfaces de usuario	25
3.1.2. Interfaces de hardware	33
3.1.3. Interfaces de software	34
3.1.4. Interfaces de comunicación	35
3.2. Requisitos funcionales	36
3.2.1. Requisitos de aplicación Web	36
3.2.2. Requisitos de aplicación Móvil	38
3.2.3. Requisitos de base de datos en la nube	40
3.2.4. Requisitos de IoT	42
3.3. Requisitos no funcionales	45
3.3.1. Requisitos de rendimiento	45
3.3.2. Seguridad	45
3.3.3. Fiabilidad	45
3.3.4. Disponibilidad.	46
3.3.5. Mantenibilidad	46
3.3.6. Portabilidad	46
3.4. Otros requisitos	47
4. Apéndices	48

1. Introducción

En el presente documento se especifican los requisitos del software para un sistema integrado combinando tecnologías móviles, web e IoT.

El objetivo principal de este proyecto es desarrollar un sistema multiplataforma compuesto por una aplicación móvil, una aplicación web y un sistema IoT que permita controlar y monitorear en tiempo real el reloj análogo “**Malbouche**”, utilizando tecnologías de desarrollo móvil, bases de datos en la nube y sensores, con el fin de resolver una necesidad real en el área de automatización remota de dispositivos físicos durante un periodo de 3 meses.

El sistema está compuesto por cuatro módulos fundamentales: una aplicación móvil, una aplicación web, base de datos en la nube y una plataforma IoT (Internet of Things, Internet de las cosas). La aplicación móvil permitirá a los usuarios iniciar sesión, controlar el movimiento de las manecillas del reloj y administrar eventos (movimientos programados para ejecutarse a cierta hora y cierto día/s). El desarrollo de la aplicación móvil no solo forma parte del proyecto académico, sino que también tiene aplicación en un contexto externo. Como parte de su implementación, la aplicación será entregada a Javier Zamorano Martínez, quien hará uso de ella fuera del entorno académico.

Por otro lado, se desarrollará una aplicación web enfocada en un bar, cuyo propósito principal será la visualización de datos recolectados por sensores en tiempo real, almacenados en una base de datos NoSQL (base de datos no relacional) para facilitar su análisis y representación gráfica. Además, esta plataforma web contará con módulos funcionales como gestión de menús (bebidas y alimentos), eventos, reservaciones y usuarios. Los usuarios con permisos podrán iniciar sesión y administrar el contenido del sitio. El objetivo es que, desde el establecimiento, tanto el personal como los clientes puedan acceder al sitio web para consultar la información ofrecida por estos módulos de forma práctica e intuitiva.

El componente IoT está conformado por una estructura mecánica que incluye motores paso a paso para controlar las agujas del reloj, así como sensores de movimiento y otros elementos electrónicos como Arduino UNO, ESP32, y módulos Bluetooth, que permitirán la comunicación entre los dispositivos físicos y las aplicaciones desarrolladas. Cabe destacar que el reloj físico está siendo diseñado y construido en el área de Mecatrónica, integrando el sistema mecánico, electrónico y de control para su funcionamiento real.

1.1. Propósitos

Este documento de Especificación de requisitos de software (ERS) detalla los requerimientos necesarios para el proyecto Sistema de control y gestión de reloj análogo “**Malbouche**”, aquí se

detalla todo lo necesario para el desarrollo de una página web, una aplicación móvil y un proyecto de IoT el cual será un reloj analógico cuyas manecillas serán controladas desde la aplicación móvil. Se explicará la funcionalidad que tendrá cada módulo de la aplicación móvil, así como la manera en la que interactúa la misma con el reloj analógico, por parte de la página web explica su utilidad, módulos dinámicos y graficación de los datos obtenidos por los sensores (dashboards).

Este documento está creado para brindar información detallada acerca del desarrollo del proyecto a cada uno de los maestros de la carrera y para brindar un panorama más amplio acerca de las funcionalidades que tendrá.

1.2. *Alcance*

Sistema de control y monitoreo de reloj análogo “Malbouche”.

Este proyecto consiste en el desarrollo de un sistema de control y monitoreo para el reloj análogo “**Malbouche**”, utilizando una aplicación móvil y una plataforma web. El reloj formará parte de un sistema IoT (Internet de las Cosas), integrando sensores que recabarán información, por ejemplo, los movimientos del entorno, que tanta proximidad o interacción hubo con el reloj físico.

La aplicación móvil consiste en enviar movimientos vía WiFi a los componentes de IoT para realizar movimientos en las manecillas del reloj físico.

El sitio web consiste en mantener la interacción dinámica con los distintos módulos como menú, eventos, promociones activas y reservaciones. Para el caso de usuarios con acceso (empleados del bar), podrán realizar cambios en la información de la página, consultar dashboards y registrar usuarios con permisos.

La base de datos en la nube consiste en almacenar toda la información del sitio web, a su vez, recibirá datos generados por los sensores y dichos datos serán obtenidos por el sitio web y serán mostrados en un dashboard.

1.3. *Personal involucrado*

Nombre	Diaz Cervantes Amieva Alejandro
Rol	Programador
Categoría profesional	Estudiante
Responsabilidades	Base de Datos, IoT

Información de contacto	0323105898@ut-tijuana.edu.mx
Aprobación	Aceptado. Diaz Cervantes Amieva Alejandro

Nombre	Lara López Josue Isaac
Rol	Programador
Categoría profesional	Estudiante
Responsabilidades	App Móvil
Información de contacto	0323105937@ut-tijuana.edu.mx
Aprobación	Aceptado. Lara López Josue Isaac

Nombre	Martinez Pérez Diana
Rol	Programadora/Diseñadora
Categoría profesional	Estudiante
Responsabilidades	Líder ,App Móvil, Sitio Web
Información de contacto	0323105965@ut-tijuana.edu.mx
Aprobación	Aceptado. Martinez Pérez Diana

Nombre	Medina Becerra Alan Oswaldo
Rol	Programador
Categoría profesional	Estudiante
Responsabilidades	App Móvil, Sitio Web
Información de contacto	0323106021@ut-tijuana.edu.mx
Aprobación	Aceptado. Medina Becerra Alan Oswaldo

Nombre	Ramirez Navarro Marcos David
Rol	Programador
Categoría profesional	Estudiante
Responsabilidades	Base De Datos, Sitio Web, App Móvil
Información de contacto	0323105847@ut-tijuana.edu.mx
Aprobación	Aceptado. Ramirez Navarro Marcos David

1.4. Definiciones, acrónimos y abreviaturas

API (Interfaz de Programación de Aplicaciones): Conjunto de funciones que permite a diferentes sistemas comunicarse entre sí.

App: Abreviatura de "aplicación", es un programa diseñado para realizar una tarea específica en un dispositivo.

Back-end: Parte del software que maneja la lógica, el procesamiento de datos y la conexión con bases de datos; no es visible para el usuario.

Base de datos NoSQL: Tipo de base de datos que almacena datos de forma no estructurada, ideal para manejar grandes volúmenes de información de manera flexible.

Cloud (Nube): Entorno remoto donde se almacenan y procesan datos a través de internet, sin necesidad de almacenamiento local.

Confiabilidad: Capacidad del sistema para funcionar de forma predecible, fácil de entender y usar.

Dispositivo IoT (Internet de las Cosas): Objeto físico con conexión a internet que puede recopilar y enviar datos (como sensores o actuadores).

Escalabilidad: Facilidad con la que un sistema puede adaptarse al aumento de usuarios o datos sin perder rendimiento.

Front-end: Parte visual del software con la que interactúa el usuario, como pantallas o botones.

Log: Registro automático de eventos, errores o actividades realizadas por un sistema, utilizado para monitoreo, diagnóstico y auditoría.

Optimización: Mejora del sistema para que consuma menos recursos y funcione más rápido.

Performance (Rendimiento): Eficiencia con la que el software utiliza los recursos del sistema.

Portabilidad: Capacidad de un software para funcionar en diferentes dispositivos o sistemas operativos.

React Native: Marco de desarrollo que permite crear aplicaciones móviles para Android y iOS usando JavaScript.

Seguridad: Protección de datos y usuarios frente a accesos no autorizados o ataques.

Software: Conjunto de programas que permiten ejecutar tareas en un computador o dispositivo.

Usuario: Persona que interactúa con el sistema para cumplir un objetivo.

Web App (Aplicación Web): Programa accesible desde un navegador que no requiere instalación en el dispositivo.

Input: Entrada de información en un sistema informático.

1.5. *Referencias*

<i>Referencia</i>	<i>Título</i>	<i>Ruta</i>	<i>Fecha</i>	<i>Autor</i>
Sitio Web	Introducción a React Native	Introduction - React Native	14 de abril del 2025	Meta platforms, Inc.
Sitio Web	Documentación oficial de MongoDB	MongoDB Documentation	No especificado	MongoDB Inc.

1.6. Resumen

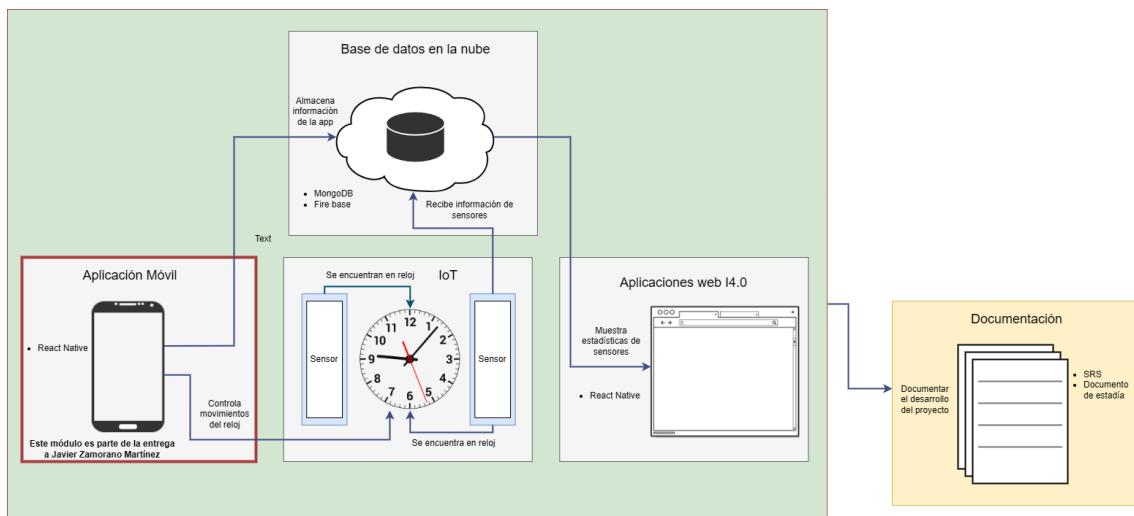
Este documento está organizado en las siguientes secciones, con el fin de presentar de manera clara y estructurada toda la información necesaria sobre el proyecto “**Malbouche**”:

- **Introducción:** Presenta el propósito del documento, el alcance del sistema, definiciones importantes, acrónimos y referencias necesarias para su comprensión. Esta sección proporciona el contexto general del proyecto.
- **Descripción General:** Ofrece una visión amplia del sistema, incluyendo las funciones principales, características generales, usuarios involucrados, restricciones, suposiciones y dependencias relevantes para el desarrollo.
- **Requisitos Específicos:** Resume los requerimientos del sistema de manera general antes de dividirlos en funcionales y no funcionales. Esta sección introduce la estructura detallada de los requisitos.
- **Requisitos Funcionales:** Detalla las funcionalidades específicas que debe cumplir el sistema. Se describen los comportamientos esperados, acciones del usuario y respuestas del sistema ante diferentes situaciones.
- **Requisitos No Funcionales:** Define las características de calidad del sistema, como el rendimiento, la seguridad, la usabilidad, la escalabilidad y otros aspectos técnicos que no están relacionados directamente con funciones específicas.
- **Apéndices:** Contiene información complementaria o de apoyo que facilita la comprensión del documento, como glosarios, referencias, diagramas, tablas u otros datos relevantes.

2.Descripción general

2.1. Perspectiva del producto

El sistema **Malbouche** está conformado por varios sistemas, los cuales estarán interactuando entre sí continuamente.



2.2. Funcionalidad del producto

Se detalla de forma escrita el diagrama creado en la perspectiva del producto.

El sistema está compuesto por distintos módulos interconectados que permiten el control del reloj análogo “**Malbouche**” y la visualización de datos recolectados por sensores. Todo esto incluye el desarrollo de la aplicación web, móvil y el sistema IoT. Cabe aclarar que la aplicación móvil se desarrollará para el cliente Javier Zamorano Martínez, esto quiere decir que no solo formará parte como proyecto escolar, sino como proyecto real.

Aplicación móvil: Esta será para cumplir una función decorativa. La aplicación móvil estará desarrollada con React Native. Su función principal será el control de las manecillas del reloj, este enviará las indicaciones a una base de datos para que sean leídos por los componentes del reloj y se deberán reflejar en este. Contará con diferentes módulos que no solo tomarán la función de controlar los movimientos del reloj, sino también se podrán crear nuevos movimientos de manera personalizada para que el usuario pueda tener diferentes opciones. También contará con el apartado de eventos que ayudará al usuario a asignarle una hora y días para que dicho movimiento se reproduzca.

Aplicación Web I4.0: Su función consta en crear el sitio web de un bar (negocio de nuestro cliente real) accesible por los clientes y personal autorizado. Este sitio web cuenta con módulos que permiten al usuario acceder a las pantallas de Home (Información del bar, promociones y galería de imágenes del bar), Menú (para visualizar los alimentos y bebidas que ofrece el bar), Eventos (eventos planeados por el establecimiento como temáticas, bandas, días especiales, etc), Reservaciones (muestra un formulario donde el cliente puede hacer la reservación de una mesa para asistir al bar a una hora específica y de forma directa sin espera), Contacto (muestra información de contacto del local, ubicación, horarios de apertura/cierre y un formulario para recibir comentarios de los clientes), Log in y Dashboard (datos obtenidos de sensores de IoT para obtener datos de proximidad e interacción). Los usuarios con accesos concedidos mantendrán la administración del sitio web.

Sistema IoT: Este, al igual que la aplicación móvil, su principal función será meramente decorativa. En este apartado se estarán integrando y ejecutando los movimientos del reloj según las órdenes enviadas desde la aplicación móvil. El uso de sensores permite de forma dinámica, la interacción con este. El almacenamiento de los datos recabados por

los sensores nos permite comprender las interacciones, acercamientos e incluso, la aceptación de los clientes por el producto artístico (reloj análogo).

Base de datos en la nube: Su principal función será almacenar toda la información que se registre en la aplicación web. Almacenará información como el menú, promociones, reservaciones, usuarios, datos para el dashboard.

2.3. Características de los usuarios

2.3.1. Aplicación móvil

Tipo de usuario	Empleados del lugar
Formación	Preparatoria
Habilidades	Conocimiento básico del uso de dispositivos móviles. Interacción simple con aplicaciones.
Actividades	Manejo de los movimientos del reloj

Tipo de usuario	Cliente
Formación	Educación básica o media
Habilidades	Uso básico de apps móviles, navegación táctil, lectura de información
Actividades	Visualizar hora, activar modo loco.

2.3.2. Aplicaciones web

Tipo de usuario	Empleados del lugar
Formación	Capacitación interna técnica.
Habilidades	Interacción simple con apps y dispositivos IoT.
Actividades	Manejo físico del reloj, monitoreo de funcionamiento y comportamiento visual.

Tipo de usuario	Cliente
Formación	Educación media o superior.
Habilidades	Navegación web, interacción con formularios.
Actividades	Cambiar configuraciones desde el portal web, visualizar menú.

2.3.3. IoT

Tipo de usuario	Técnico en IoT
Formación	Ingeniería/TSU en Electrónica, Mecatrónica o técnica.
Habilidades	Programación de microcontroladores, Bluetooth/WiFi, sensores.
Actividades	Dar mantenimiento al reloj físico, pruebas de hardware y comunicación.

2.3.4. Base de datos

Tipo de usuario	Administrador de Base de Datos
Formación	Ingeniería/TSU en Sistemas o Ciencias de la Computación.
Habilidades	SQL, NoSQL, modelado de datos, backups, control de accesos.

Actividades	Optimizar el rendimiento de la base de datos, mantener seguridad de datos.
-------------	--

2.4. Restricciones

Durante el desarrollo del sistema de control y gestión del reloj análogo “Malbouche”, se han identificado una serie de restricciones que deben ser consideradas cuidadosamente para asegurar la viabilidad, estabilidad y funcionalidad del sistema. Estas restricciones abarcan tanto aspectos técnicos como operativos, e impactan directamente en las decisiones de diseño, implementación y despliegue del software. Es importante señalar que algunas de estas limitaciones están definidas por la tecnología disponible, otras por los lineamientos académicos del proyecto, y otras más por el entorno de desarrollo específico elegido por el equipo. A continuación, se detallan dichas restricciones que condicionan tanto el desarrollo de la aplicación móvil, la plataforma web y el sistema IoT asociado al reloj físico:

1. Compatibilidad con sistemas operativos móviles:

La aplicación móvil debe ser completamente funcional en dispositivos Android e iOS, por lo cual el desarrollo se está realizando con tecnologías multiplataforma como React Native y Expo.

2. Compatibilidad del hardware IoT:

El sistema IoT ha sido diseñado exclusivamente para funcionar con microcontroladores ESP32 No se considera compatible con otras plataformas de hardware.

3. Tipo de conexión del reloj físico:

Aunque aún no se define de manera definitiva, se planea que el reloj físico utilice conexión **Bluetooth** para interactuar con la aplicación móvil. Esto implica la necesidad de soporte Bluetooth activo en los dispositivos móviles.

4. Restricciones de prueba en navegadores web:

Aunque la aplicación web no tiene un navegador obligatorio, las pruebas y validaciones del sistema se han realizado únicamente en Google Chrome y Mozilla Firefox.

5. Limitaciones físicas del dispositivo IoT:

Las posibles restricciones de espacio, energía o diseño del reloj físico no serán consideradas dentro del alcance de este documento, ya que corresponden a un equipo externo al área de desarrollo de software.

6. Normativas y estándares aplicables:

El proyecto debe seguir el estándar IEEE 830-1998 para la especificación de requisitos de software, como parte de los lineamientos escolares establecidos para la documentación.

7. Seguridad:

Aunque no se han definido normas específicas, se tiene previsto implementar un nivel de seguridad básico, que cumpla con buenas prácticas en el manejo de usuarios y transmisión de datos.

8. Límites temporales:

El proyecto cuenta con un tiempo de desarrollo estimado de cuatro meses, lo cual establece una restricción importante en la planificación y entrega de funcionalidades.

2.5. *Suposiciones y dependencias*

El desarrollo exitoso de la plataforma depende de varios factores críticos que, de modificarse o no atenderse, podrían provocar fallos en el sistema:

- **Acceso de los usuarios a los dispositivos:** Se asume que todos los usuarios tendrán acceso a dispositivos con conexión a internet, como teléfonos inteligentes, tabletas o computadoras, para interactuar con el sistema.
- **Conectividad a internet fiable:** Para la correcta funcionalidad del sistema, como gestión de eventos, reservaciones, promociones y el sistema de base de datos en la nube, se presupone que los usuarios tendrán un acceso estable a internet.
- **Sistema Operativo actualizado:** Se asume que el sistema operativo de los dispositivos que utilizarán el sistema estarán en la última versión para prevenir errores de incompatibilidad.
- **Funcionamiento de sensores:** Se asume que los sensores y dispositivos que forman parte del sistema de IoT están correctamente calibrados, alimentados y conectados durante el uso de estos.
- **Instalación del sistema:** Se asume que el usuario final hará una correcta instalación del sistema sin alterar el funcionamiento de este.

2.6. *Evolución previsible del sistema*

El sistema de control y gestión del reloj análogo “Malbouche” ha sido diseñado con una arquitectura modular y escalable que permite su expansión y mejora a mediano y largo plazo. A continuación, se describen algunas de las posibles líneas de evolución que podrían implementarse una vez finalizada la versión inicial del sistema:

1. Ampliación de funcionalidades

- **Programación avanzada de eventos:** Se contempla la posibilidad de incorporar una interfaz visual tipo calendario para programar eventos de manera más intuitiva, incluyendo repeticiones, condiciones lógicas y acciones encadenadas.
- **Notificaciones al usuario:** Podría añadirse un sistema de notificaciones push (en la app móvil) o alertas por correo, informando al usuario sobre el estado del reloj, ejecución de eventos o problemas técnicos.
- **Control por voz:** A futuro, podría integrarse compatibilidad con asistentes virtuales como Google Assistant o Alexa para controlar el reloj mediante comandos de voz.

2. Mejora del sistema IoT

- **Comunicación bidireccional con el reloj:** Actualmente, el sistema actúa de forma unidireccional (comandos desde app al reloj). En futuras versiones se podría implementar un canal de respuesta donde el ESP32 envíe retroalimentación al backend (estado actual, errores, confirmación de ejecución, etc.).
- **Soporte para múltiples relojes:** Se podría extender la plataforma para permitir el control de varios relojes conectados a una misma cuenta o desde una misma app/web, útil para instalaciones con múltiples dispositivos.
- **Modo offline o sincronización local:** Si bien el sistema depende actualmente de conexión a internet, se podría incorporar un modo offline limitado, o bien una funcionalidad de sincronización local directa mediante Bluetooth sin necesidad de conexión al back-end.

3. Expansión tecnológica

- **Migración del back-end a la nube:** Una vez superada la etapa de pruebas y desarrollo local, se planea desplegar el back-end en servicios externos como AWS, Heroku o Vercel, para mejorar la disponibilidad, seguridad y escalabilidad del sistema.
- **Internacionalización:** La interfaz podría adaptarse en versiones futuras para soportar múltiples idiomas, facilitando su uso por usuarios de diferentes regiones.

4. Interfaz y experiencia de usuario

- **Mejora del diseño UI/UX:** Aunque el sistema actual cuenta con una interfaz funcional, se plantea una mejora progresiva en términos de experiencia de usuario, navegación más intuitiva, temas personalizables y animaciones para representar visualmente el estado del reloj.
- **Panel administrativo avanzado:** Para la versión web, se podría integrar un panel de administración más robusto, con gráficos en tiempo real, filtros avanzados, exportación de datos y gestión de roles de usuarios.

3. Requisitos específicos

Aplicación Móvil

Número de requisitos	RF 1		
Nombre de requisito	Inicio de sesión		
Tipo	<input checked="" type="checkbox"/> Requisito	<input type="checkbox"/> Restricción	
Fuente del requisito			
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial	<input type="checkbox"/> Media/Deseado	<input type="checkbox"/> Baja/ Opcional

Al iniciar la aplicación, se mostrará una pantalla de inicio de sesión donde el usuario podrá ingresar sus datos personales para acceder a todas las funcionalidades disponibles. También se ofrecerá la opción de ingresar sin una cuenta; sin embargo, el acceso será limitado y solo se podrán utilizar ciertas funciones básicas.

Número de requisitos	RF 1.1		
Nombre de requisito	Correo y contraseña		
Tipo	<input checked="" type="checkbox"/> Requisito	<input type="checkbox"/> Restricción	
Fuente del requisito			
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial	<input type="checkbox"/> Media/Deseado	<input type="checkbox"/> Baja/ Opcional

Para acceder a todas las funcionalidades de la aplicación, el usuario deberá contar con una cuenta registrada, la cual requiere un correo electrónico y una contraseña. Estos datos deberán ingresarse en los campos correspondientes durante el inicio de sesión.

Número de requisito	RF 2		
Nombre de requisito	Módulo de control de reloj		
Tipo	<input checked="" type="checkbox"/> Requisito	<input type="checkbox"/> Restricción	
Fuente del requisito			
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial	<input type="checkbox"/> Media/Deseado	<input type="checkbox"/> Baja/ Opcional

En este apartado, la aplicación permitirá controlar manualmente los movimientos del reloj, ofreciendo seis modos predefinidos:

- Derecha: Las manecillas giran en dirección horaria.
- Izquierda: Las manecillas giran en dirección antihoraria.

- Loco: Las manecillas giran en distintas direcciones a alta velocidad, generando un movimiento errático.
- Personalizado: El usuario puede controlar individualmente cada manecilla, definiendo su dirección de giro.
- Normal: Las manecillas se sincronizan con la hora actual, funcionando como un reloj convencional.

Además, desde esta sección también será posible encender o apagar el reloj según se requiera.

Número de requisito	RF 2.1		
Nombre de requisito	Módulo de gestión de movimientos de reloj		
Tipo	<input checked="" type="checkbox"/> Requisito	<input type="checkbox"/> Restricción	
Fuente del requisito			
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Essencial	<input type="checkbox"/> Media/Deseado	<input type="checkbox"/> Baja/ Opcional

La aplicación permitirá eliminar movimientos previamente registrados, así como añadir nuevos o actualizar los existentes. Para cada movimiento del reloj, se podrá configurar la siguiente información:

- Nombre: Identificador del movimiento.
- Sentido de las manecillas: Selección entre derecha o izquierda.
- Tiempo: Duración del movimiento.
- Velocidad: Intensidad o rapidez con la que se ejecuta el movimiento.

Esta funcionalidad brinda flexibilidad para personalizar el comportamiento del reloj según las necesidades del usuario.

Número de requisito	RF 3		
Nombre de requisito	Módulo de eventos		
Tipo	<input checked="" type="checkbox"/> Requisito	<input type="checkbox"/> Restricción	
Fuente del requisito			
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Essencial	<input type="checkbox"/> Media/Deseado	<input type="checkbox"/> Baja/ Opcional

Los eventos representan movimientos que el reloj debe ejecutar en horarios previamente definidos. En este módulo, los usuarios podrán gestionar todos los eventos programados, con la posibilidad de:

- Visualizar los eventos existentes.

- Activar o desactivar eventos en cualquier momento.
- Añadir nuevos eventos.
- Actualizar la información de eventos existentes.
- Eliminar eventos que ya no se requieran.

Cada evento incluirá la siguiente información:

- Nombre del evento.
- Hora de inicio.
- Hora de finalización.
- Días de la semana en los que se ejecutará el evento.
- Tipo de movimiento a realizar.

Número de requisito	RF 4		
Nombre de requisito	Módulo de usuarios		
Tipo	<input checked="" type="checkbox"/> Requisito	<input type="checkbox"/> Restricción	
Fuente del requisito			
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial	<input type="checkbox"/> Media/Deseado	<input type="checkbox"/> Baja/ Opcional

Este módulo permite visualizar, registrar, editar y eliminar usuarios. Cada usuario contará con la siguiente información:

- Nombre
- Apellidos
- Correo electrónico
- Rol

Número de requisito	RF 5		
Nombre de requisito	Base de datos en la nube		
Tipo	<input type="checkbox"/> Requisito	<input type="checkbox"/> Restricción	
Fuente del requisito			
Prioridad del requisito	<input type="checkbox"/> Alta/Esencial	<input type="checkbox"/> Media/Deseado	<input type="checkbox"/> Baja/ Opcional

La base de datos se encargará de almacenar de forma segura toda la información registrada en la aplicación, incluyendo usuarios, eventos programados, movimientos del reloj, entre otros datos relevantes para el funcionamiento de la aplicación.

Aplicación Web

Número de requisito	RF 6		
Nombre de requisito	Módulo de estadísticas de sensores		
Tipo	<input checked="" type="checkbox"/> Requisito	<input type="checkbox"/> Restricción	
Fuente del requisito			
Prioridad del requisito	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> Baja/ Media/Deseado <input type="checkbox"/> Opcional
	Alta/Esencial	Media/Deseado	Opcional

Este módulo permite visualizar y graficar la información provenientes de los sensores que se encuentran colocados en el reloj físico. A su vez será posible realizar un pdf de las estadísticas actuales.

Número de requisito	RF 7		
Nombre de requisito	Módulo de menú		
Tipo	<input checked="" type="checkbox"/> Requisito	<input type="checkbox"/> Restricción	
Fuente del requisito			
Prioridad del requisito	<input checked="" type="checkbox"/>	<input type="checkbox"/> Alta/Esencial	<input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/ Opcional

Este módulo permitirá visualizar la información de las bebidas y platillos disponibles actualmente en el establecimiento. Además, ofrece la posibilidad de:

- Registrar nuevas bebidas o platillos.
- Editar la información de los ya existentes.
- Eliminar aquellos que ya no estén disponibles.

Número de requisito	RF 8		
Nombre de requisito	Módulo de eventos y promociones		
Tipo	<input checked="" type="checkbox"/> Requisito	<input type="checkbox"/> Restricción	
Fuente del requisito			
Prioridad del requisito	<input checked="" type="checkbox"/>	<input type="checkbox"/> Alta/Esencial	<input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/ Opcional

Este módulo permitirá visualizar la información de los eventos y promociones disponibles actualmente en el establecimiento. Además, ofrece la posibilidad de:

- Registrar nuevos eventos y promociones.
- Editar la información de los ya existentes.

- Eliminar aquellos que ya no estén disponibles.

Número de requisito	RF 9		
Nombre de requisito	Módulo de usuario		
Tipo	<input checked="" type="checkbox"/> Requisito	<input type="checkbox"/> Restricción	
Fuente del requisito			
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial	<input type="checkbox"/> Media/Deseado	<input type="checkbox"/> Baja/ Opcional

Este módulo permitirá visualizar, registrar, editar y eliminar usuarios. Cada usuario contará con la siguiente información:

- Nombre
- Apellidos
- Correo electrónico
- Puesto

Número de requisito	RF 10		
Nombre de requisito	Módulo de reservaciones		
Tipo	<input checked="" type="checkbox"/> Requisito	<input type="checkbox"/> Restricción	
Fuente del requisito			
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial	<input type="checkbox"/> Media/Deseado	<input type="checkbox"/> Baja/ Opcional

Este módulo permitirá a los clientes realizar reservaciones en el local, para realizar estas reservaciones se les pedirá lo siguiente:

- Nombre completo de la persona responsable.
- Número de teléfono de la persona responsable.
- Correo electrónico de la persona responsable.
- Día de la reservación.
- Horario en que se realizará la reservación.
- Cantidad de personas.

En este módulo los usuarios con rol de administrador podrán visualizar todas las reservaciones al igual que confirmarlas, cancelarlas o editarlas.

Sistema IoT

Número de requisito	RF 11		
Nombre de requisito	Reloj físico análogo		
Tipo	<input checked="" type="checkbox"/> Requisito	<input type="checkbox"/> Restricción	
Fuente del requisito			
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial	<input type="checkbox"/> Media/Deseado	<input type="checkbox"/> Baja/ Opcional

El reloj deberá estar diseñado para conectarse con la aplicación móvil, permitiendo su control remoto desde esta. Todos los movimientos seleccionados en la aplicación deberán reflejarse de forma precisa en el funcionamiento del reloj físico.

Número de requisito	RF 12		
Nombre de requisito	Sensores en el reloj físico		
Tipo	<input checked="" type="checkbox"/> Requisito	<input type="checkbox"/> Restricción	
Fuente del requisito			
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial	<input type="checkbox"/> Media/Deseado	<input type="checkbox"/> Baja/ Opcional

Los sensores integrados en el reloj físico deberán enviar información directamente a la base de datos, la cual será posteriormente representada en la aplicación web. Además, estos sensores podrán ser controlados desde la propia aplicación web, permitiendo una gestión centralizada e interactiva del sistema.

Base de datos en la nube

Número de requisito	RF 13		
Nombre de requisito	Almacenamiento de datos en tiempo real		
Tipo	<input checked="" type="checkbox"/> Requisito	<input type="checkbox"/> Restricción	
Fuente del requisito			
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Esencial	<input type="checkbox"/> Media/Deseado	<input type="checkbox"/> Baja/ Opcional

La base de datos deberá ser capaz de sincronizar en tiempo real los datos generados por los sensores integrados en el reloj físico. Esto garantizará que la información esté siempre actualizada y para su visualización en la aplicación web.

Número de requisito	RF 14		
Nombre de requisito	Seguridad en el almacenamiento de los datos		
Tipo	<input checked="" type="checkbox"/> Requisito	<input type="checkbox"/> Restricción	

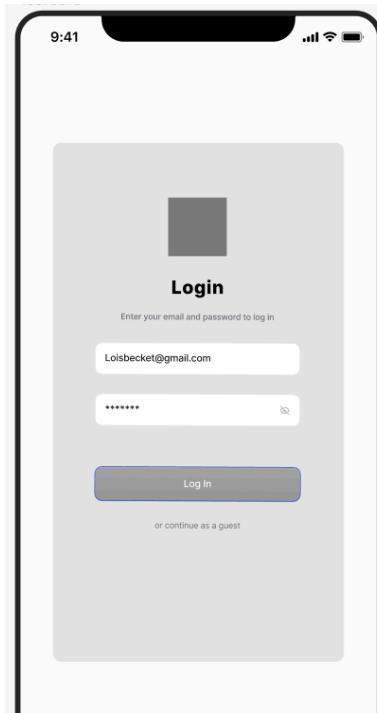
Fuente del requisito	
Prioridad del requisito	<input checked="" type="checkbox"/> Alta/Essencial <input type="checkbox"/> Media/Deseado <input type="checkbox"/> Baja/ Opcional

La base de datos se encargará de almacenar de forma segura toda la información registrada en la aplicación, incluyendo usuarios, eventos programados, platillos, bebidas, entre otros datos relevantes para el funcionamiento de la aplicación. Todos los datos ingresados por el usuario o recibidos desde fuentes externas deben ser validados antes de ser procesados o almacenados.

3.1. *Requisitos comunes de los interfaces*

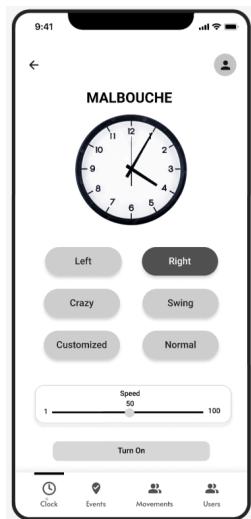
3.1.1. *Interfaces de usuario*

- Interfaz de usuario Móvil:
 - Inicio de sesión



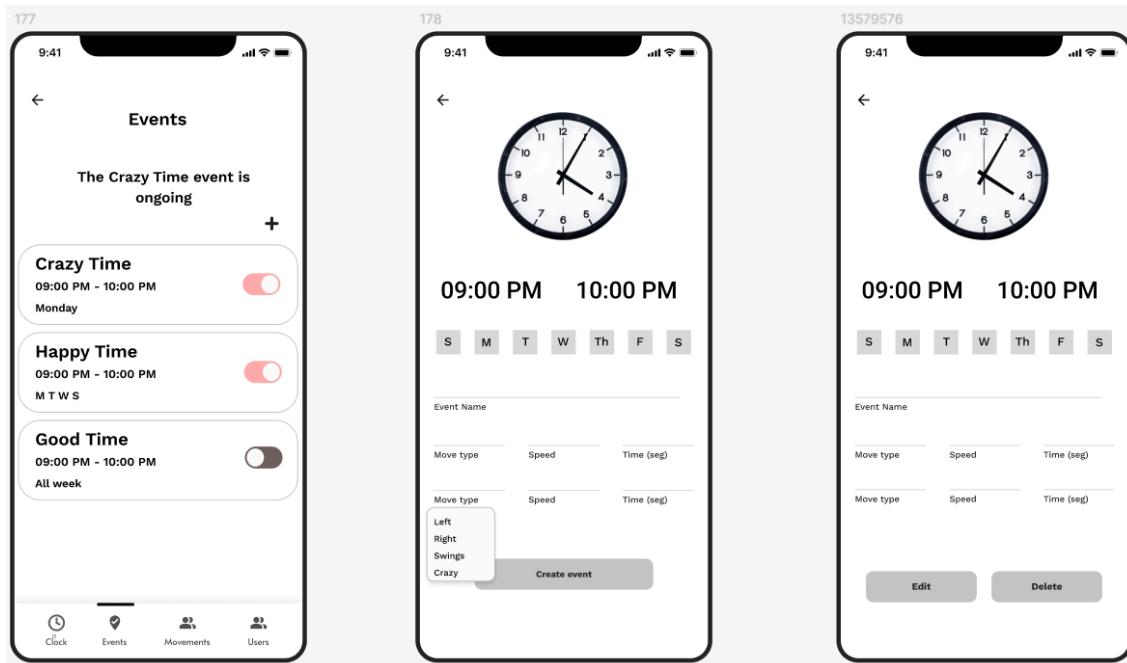
- En los inputs se ingresará la información personal del usuario (correo electrónico y contraseña).
- El botón permitirá iniciar sesión en la aplicación.

- Módulo de control de reloj

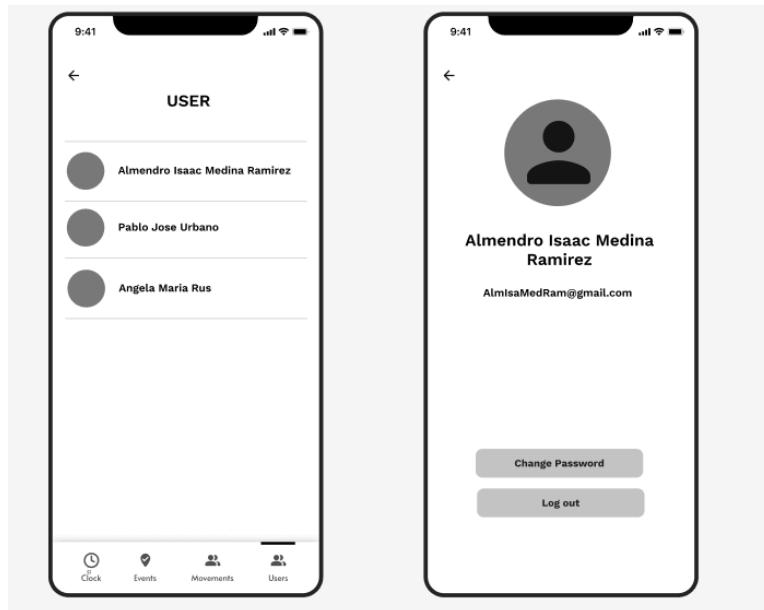


- Los botones permitirán cambiar el sentido de las manecillas.
- La barra inferior permitirá controlar la velocidad.
- El botón que se encuentra al final permitirá encender o apagar el reloj.

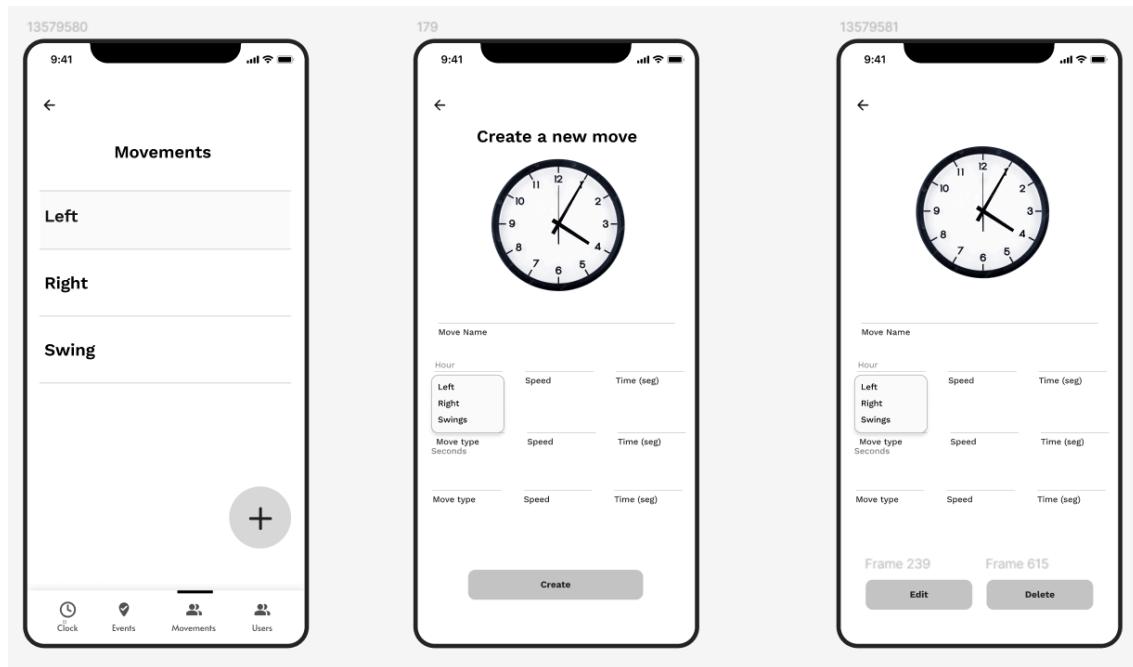
- Módulo de eventos



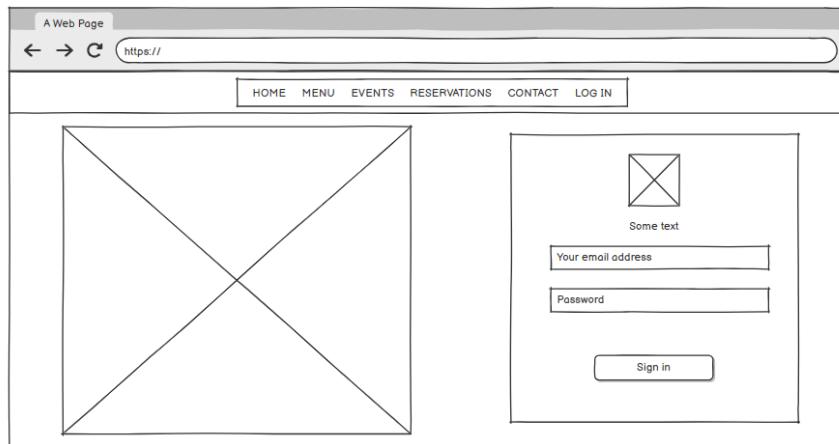
- El signo de “+” redirigirá al usuario al apartado de creación de un nuevo evento.
 - Los botones de encendido y apagado permitirán activar o desactivar el evento que se deseé.
 - Si se selecciona un evento se redirigirá a la información del mismo y ahí se podrá editar o eliminar tal movimiento.
- Módulo de usuarios



- Se muestra la información de los usuarios que están registrados en la aplicación móvil.
 - Se muestra la información del usuario que está en la sesión actual.
- Módulo de movimientos



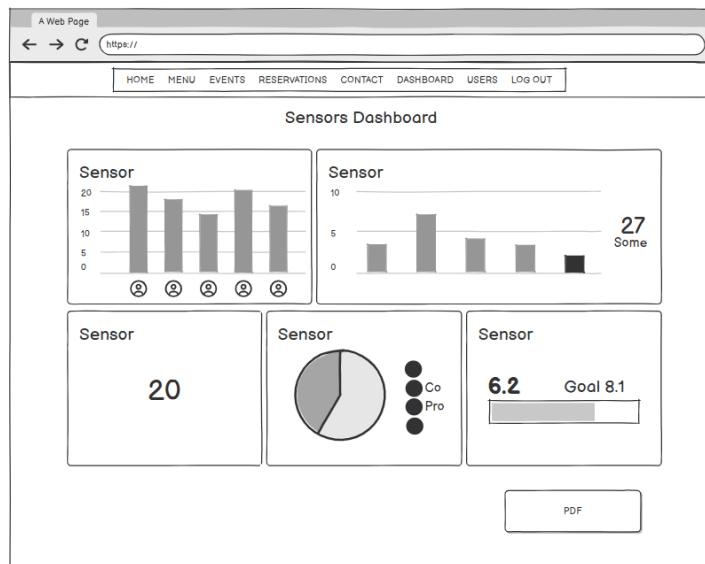
- Se muestran todos los movimientos registrados en la aplicación.
- En el signo de “+” redirige al apartado de creación de un nuevo movimiento.
- Si se selecciona un movimiento se redirigirá a la información del mismo y ahí se podrá editar o eliminar tal movimiento.
- Interfaz de usuario web:
 - Inicio de sesión



- En los inputs se ingresará la información personal del usuario (correo electrónico y contraseña).

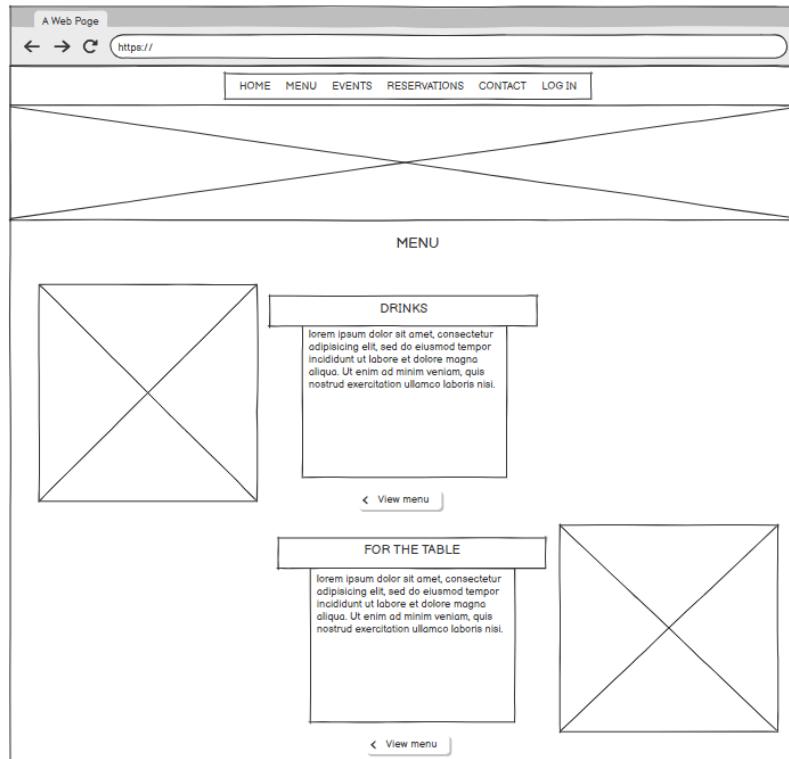
- El botón permitirá iniciar sesión en la aplicación web.

- Módulo de gráficas de sensores

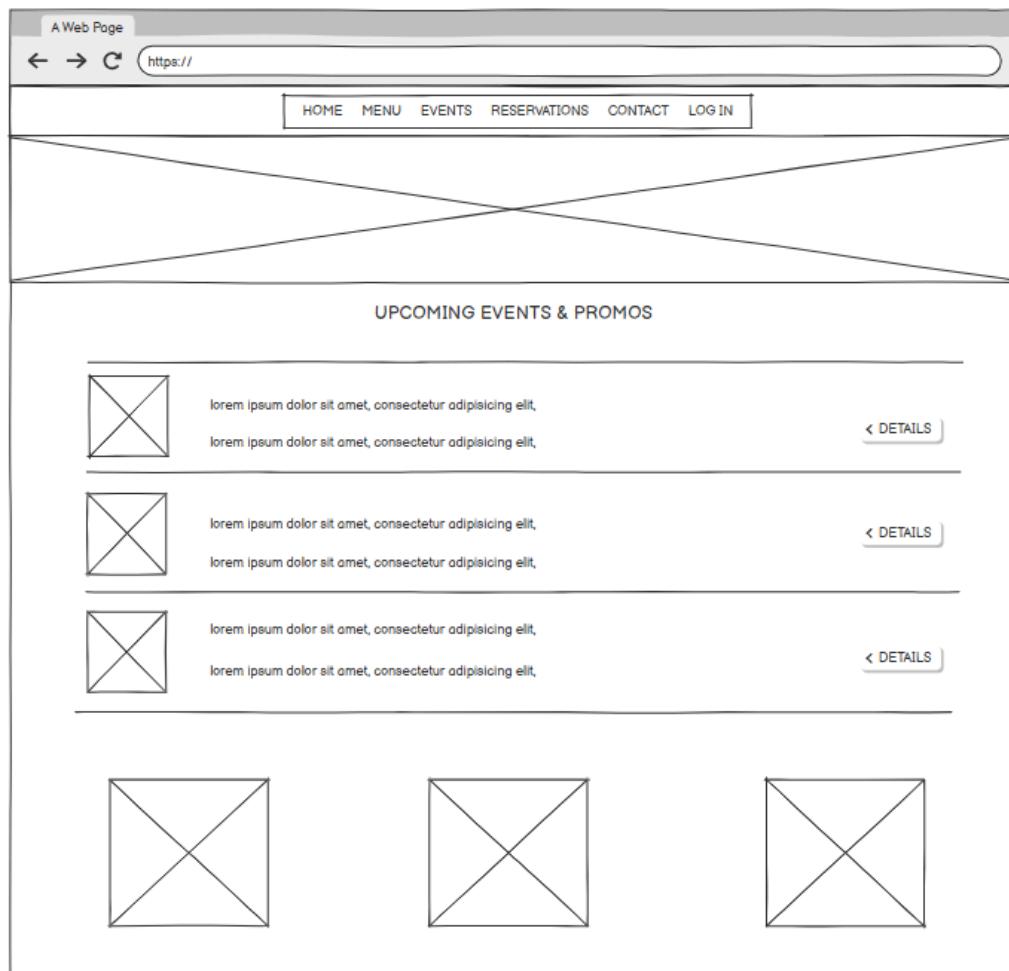


- Se muestran de manera gráfica la información de los sensores integrados en el reloj.
- El botón permitirá generar un pdf de los gráficos.

- Módulo de menú



- En los recuadro se pretende agregar fotografías de la bebida y platillo especial del establecimiento, junto a una descripción sobre las bebidas o platillos en general.
- Los botones redirigirán a un documento con el menú actual del establecimiento.
 - Módulo de eventos y promociones



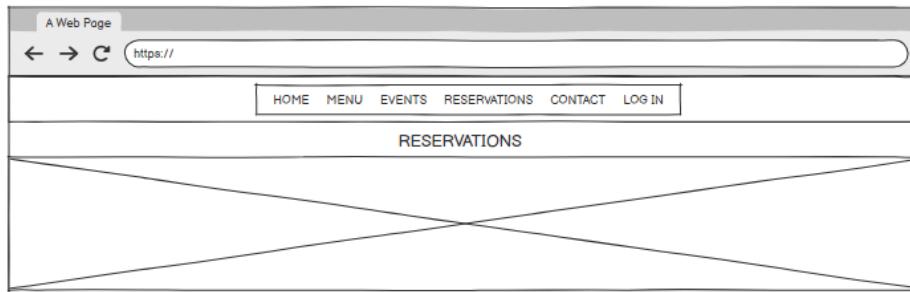
- De manera enlistada estarán los eventos disponibles, estos contarán con una descripción general.
- En la parte inferior se encontrarán las promociones actuales del establecimiento.
 - Módulo de reservaciones

A Web Page

https://

HOME MENU EVENTS RESERVATIONS CONTACT LOGIN

RESERVATIONS



SCHEDULE

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea

Fill the reservation form

Gests	Date	Time
-------	------	------

Availability

21:00	21:30	22:00	22:30
-------	-------	-------	-------

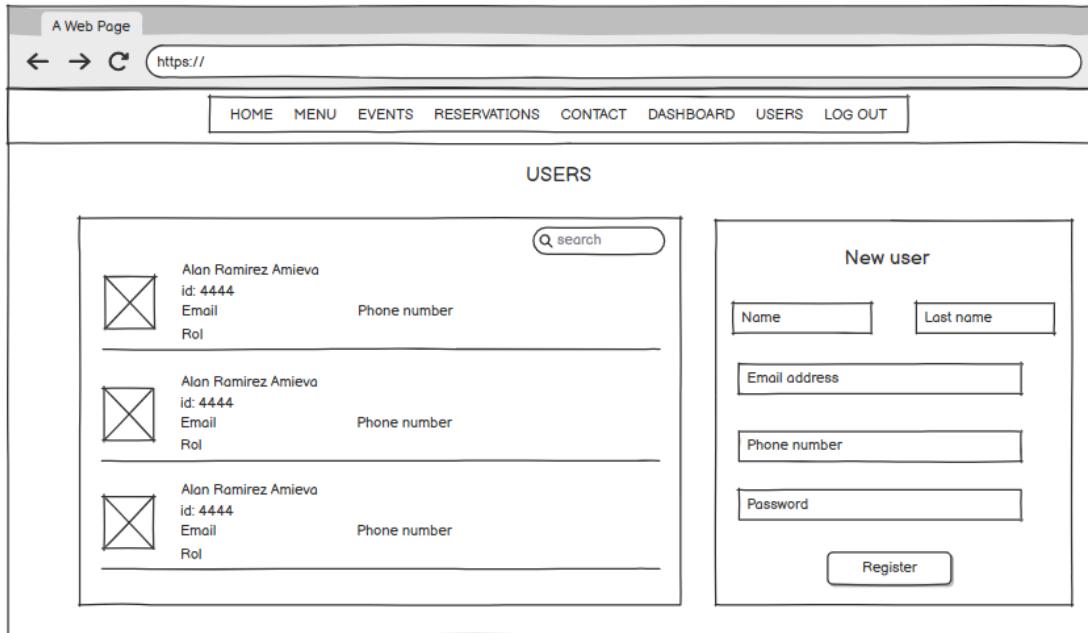
Customer information

Name	Last name	Num tel	Email
------	-----------	---------	-------

- Se muestra el horario del establecimiento.
- Se presenta un formulario con los campos necesarios para obtener la información requerida para realizar una reservación.
- Módulo de visualización de reloj



- Se muestra el movimiento actual que está teniendo el reloj físico.
- Módulo de usuarios



- Se muestra el listado de los usuarios registrados en el sistema.
- Se presenta un formulario para poder registrar nuevos usuarios.

3.1.2. *Interfaces de hardware*

El reloj está basado en un **ESP32**, el cual se encarga de controlar el movimiento independiente de las manecillas del reloj mediante motores, paso a paso o servos, y puede sincronizarse para mostrar la hora real. Este apartado describe cómo el sistema se comunica con los componentes físicos.

- Motores Paso a Paso (x2) + Drivers ULN2003 o A4988

Cada motor se conecta a un conjunto de 4 GPIOs del ESP32 (aún no están definidos los pines).

Drivers alimentados con 5V.

- Sensor Ultrasónico (HC-SR04)

El reloj puede reaccionar al entorno. Mediante un sensor ultrasónico montado en la estructura, el sistema detecta la proximidad de una persona. Esto permite activar modos especiales o animaciones de las manecillas cuando alguien se acerca (por ejemplo, que las manecillas giren en espiral).

El ESP32 se comunica con el sensor mediante dos GPIOs: uno como salida (Trigger) y otro como entrada (Echo). La distancia se calcula en tiempo real midiendo el tiempo de retorno del pulso ultrasónico. El sensor opera con alimentación de 5 V.

- Tira de LED (WS2812B o similar)

El reloj también cuenta con una tira de LEDs direccionables, que proporciona retroiluminación y efectos visuales sincronizados con el comportamiento de las manecillas. Por ejemplo, puede iluminarse al detectar presencia, cambiar de color con el paso del tiempo, o parpadear en patrones al marcar cada hora.

La tira se controla con un solo GPIO del ESP32 usando una señal digital con temporización precisa. Cada LED puede ser programado de forma individual, lo que permite crear efectos visuales complejos. La tira se alimenta con 5 V y debe compartir tierra con el ESP32.

- Sensor infrarrojo

En caso de que los clientes se acerquen mucho al reloj el sensor infrarrojo detectara su presencia cambiando de modo al reloj, lo que activará un evento diferente de colores en la tira de led.

3.1.3. *Interfaces de software*

Nombre de la aplicación/programa:

Node.js (backend), MongoDB Atlas (base de datos), API RESTful, React Native (móvil), React (web).

Para qué sirve la conexión:

Para procesar comandos del usuario, gestionar la lógica del reloj, almacenar información (usuarios, configuraciones) y enviar instrucciones al ESP32.

Cómo se va a conectar:

Mediante protocolo HTTP/HTTPS usando endpoints RESTful. La app móvil y la web se conectan al mismo backend. El backend envía instrucciones al reloj físico por Bluetooth o conexión serial (según implementación final).

Tipos de datos intercambiados:

- Texto (usuarios, configuraciones, modos del reloj).
Fechas y horas (eventos programados).
- Comandos (mover manecillas, cambiar dirección o modo).
Credenciales (inicio de sesión).
- Respuestas de estado (modo actual del reloj, confirmación de ejecución).

3.1.4. *Interfaces de comunicación*

- **Tipo de comunicación:** Wifi, Bluetooth
- **Protocolo de Comunicación:** MQTT,
- **Datos que se envían o reciben:** Los datos que serán enviados mediante bluetooth son las instrucciones que deberá realizar el reloj desde la aplicación móvil y mediante wifi se enviará información recabada de los sensores integrados en el reloj al sitio web para mostrar estadísticas del funcionamiento del mismo.
- **Seguridad:** Se integrará un manejo de sesiones para que personas ajenas al establecimiento solo tenga acceso a una versión de prueba y no pueda manipular información crítica.

3.2. Requisitos funcionales

3.2.1. Requisitos de aplicación Web

3.2.1.1. Validación de Datos

- Inicio de sesión: Válida que el correo electrónico y la contraseña cumplan con el formato solicitado y que los campos no estén vacíos.
- Gráficas de sensores: Verifica que los datos mostrados se encuentren en rangos válidos antes de graficarlos.
- Menú, eventos, promociones, reservaciones y usuarios): Valida que campos como nombres, fechas, precios y horarios estén completos y cumplan con el formato solicitado.

3.2.1.2. Secuencia de Operaciones

- Inicio de sesión: Primero el usuario ingresa correo y contraseña, después el sistema verifica que sea correcto y por último el sistema da acceso.
- Gráficas de sensores: Primero mide un sensor, luego envía el dato, después lo guarda en memoria y por último se muestran en la página en formato de gráfica.
- Registros: Primero el usuario llena los campos que se solicitan de manera obligatoria, luego el usuario confirma el registro, después el sistema verifica que la información esté dentro de los criterios y por último el sistema guarda la información en la base de datos.
- Ediciones: Primero el usuario edita los campos que estén permitidos, luego el usuario confirma los cambios, después el sistema verifica que la información esté dentro de los criterios y por último el sistema guarda la información en la base de datos.
- Eliminaciones: Primero el usuario selecciona un registro que desee eliminar, después confirma la eliminación y por último el sistema elimina el registro del sistema.

3.2.1.3. Manejo de Errores

- Si un campo obligatorio está vacío, el sistema no permitirá continuar con el flujo de la acción que se esté realizando, hasta que no se haya llenado tal campo.
- Si un campo no cumple con los criterios establecidos el sistema no permitirá continuar con el flujo de la acción que se esté realizando, hasta que no se haya llenado tal campo.

- Si ocurre un error al registrar, editar o eliminar, se muestra un mensaje de error sin perder los datos ingresados.
- Si no se puede conectar con el servidor, se muestra un mensaje de error.

3.2.1.4. Parámetros de Operación

- Longitud mínima de contraseña: 8 caracteres.
- Formato de correo: debe incluir "@" y dominio válido.
- Rango de fechas válidas: desde el día actual hasta 1 año adelante.
- Intervalo de actualización de gráficas: cada 60 segundos.
- Campos obligatorios: nombre, correo, contraseña, fecha, precio y descripción según módulo.
- Límite de registros por página: 10 a 50 según sección.

3.2.1.5. Generación de salidas

- Mensaje de bienvenida al iniciar sesión.
- Confirmación visual y textual al registrar, editar o eliminar.
Alertas en color rojo para errores de validación.
- Visualización de datos de sensores en gráficas interactivas.
- Notificaciones para cambios guardados exitosamente.
- Actualización automática de la tabla de datos tras cada operación.

3.2.1.6. Relación entre Entradas y Salidas

- Entrada: correo y contraseña válidos → Salida: acceso al sistema.
- Entrada: datos del sensor válidos → Salida: actualización de gráficas.
- Entrada: datos de formulario → Salida: registro guardado y mensaje de confirmación.
- Entrada: clic en eliminar → Salida: confirmación visual y eliminación del registro.
- Entrada: error de conexión → Salida: mensaje de error y opción de reintento.

3.2.1.7. Almacenamiento de Base de datos:

- Se almacenan: usuarios, promociones, eventos, gráficas de sensores.
- Formato: registros con campos estructurados (JSON o SQL).
- Acceso: lectura y escritura según permisos del usuario.
- Respaldo: se realiza copia de seguridad periódica.

- Persistencia: los datos se conservan aún tras cierre de sesión o error.

3.2.2. Requisitos de aplicación Móvil

3.2.2.1.

Validación de Datos:

Antes de procesar cualquier acción en la aplicación, el software validará que los datos recibidos sean correctos y estén en el formato adecuado. Por ejemplo:

- El correo electrónico debe tener un formato válido (ej. usuario@dominio.com).
- Las contraseñas deben tener una longitud mínima de 8 caracteres.
- La hora ingresada para configurar el reloj debe estar en formato HH:MM y no debe duplicarse.
- Las fechas de eventos no pueden estar en el pasado.
- Los datos de usuarios nuevos deben incluir un correo único y un rol permitido.

3.2.2.2.

Secuencia de Operaciones:

El sistema sigue una serie de pasos definidos para ejecutar las acciones de forma correcta y ordenada. Algunos ejemplos de flujos típicos:

- Inicio de sesión: El usuario introduce credenciales → Se validan → Si son correctas, accede; si no, se muestra un error.
- Configuración de reloj/eventos/usuarios: El usuario realiza una acción (crear, editar, eliminar) → Se valida → Se actualiza la base de datos.
- Control de sensores: El usuario solicita encender/apagar → Se envía el comando → Se espera respuesta del dispositivo.
- Sincronización: La app detecta el reloj → Compara datos → Actualiza los necesarios.

3.2.2.3.

Manejo de Errores:

El sistema debe gestionar adecuadamente las fallas para mantener la estabilidad. Ejemplos:

- Si se ingresan credenciales incorrectas, se notifica al usuario sin revelar información sensible.

- Si no se puede sincronizar con el reloj, se informa del fallo y se registra el error
- Si se intenta registrar un usuario con un correo ya existente, se bloquea la operación.

3.2.2.4. Parámetros de Operación:

Para el correcto funcionamiento del sistema, se deben configurar ciertos parámetros clave:

- Rango de horarios válidos para reloj y eventos: entre 00:00 y 23:59.
- Roles permitidos para usuarios: administrador, usuario y visitante.
- Tiempo de espera para respuesta de sensores: máximo 5 segundos.
- Máximo de 10 eventos configurables por día.

3.2.2.5.

Generación de salidas:
Luego de procesar los datos, el sistema genera salidas según la acción realizada:

- Si el inicio de sesión es exitoso, el usuario es redirigido a la pantalla principal.
- Al guardar un reloj, evento o usuario, se muestra una confirmación visual.
- Si se activa un sensor, se muestra un mensaje como “Sensor encendido”.
- En caso de error, se muestran mensajes claros y específicos.

3.2.2.6.

Relación entre Entradas y Salidas:

Las entradas que recibe el sistema generan respuestas específicas:

- Al introducir correo y contraseña (entrada), el sistema verifica y otorga acceso (salida).
- Si se introduce una hora de reloj o un evento (entrada), se actualiza la configuración (salida).
- Si se presiona “encender sensor” (entrada), el sistema activa el sensor y muestra estado (salida).

3.2.2.7.

Almacenamiento de Base de datos:

El sistema guarda información crítica de forma estructurada y segura para su posterior análisis o consulta:

- Usuarios: ID, nombre, correo, rol, fecha de creación/modificación.

- Relojes: ID, hora configurada, nombre, fecha de última edición.
- Eventos: Título, descripción, fecha y hora, estado.
- Sensores: Historial de activaciones con fecha y hora.
- Registros de sincronización: hora de última sincronización, estado.

3.2.3. Requisitos de base de datos en la nube

3.2.3.1. Validación de Datos:

- Todos los datos ingresados por el usuario o recibidos desde fuentes externas deben ser validados antes de ser procesados o almacenados. Esto incluye:
 - Verificación de campos obligatorios.
 - Validación de formatos (por ejemplo, correo electrónico, fechas, números).

3.2.3.2. Secuencia de Operaciones

- El software sigue una serie de pasos para procesar los datos de forma correcta y ordenada.

Ejemplo general de flujo:

1. Inicio de sesión del usuario (Login): Verifica credenciales y autentica al usuario.
2. Lectura de sensores: Se activan los sensores y se toman medidas.
3. Validación de datos: Se validan los datos recibidos de los sensores.
4. Procesamiento: Se aplican reglas o se ejecutan cálculos necesarios.
5. Almacenamiento: Los datos se guardan en la base de datos.
6. Generación de salidas: Se envían datos al usuario, a un actuador o a un sistema externo.
7. Registro de logs y eventos.

3.2.3.3. Manejo de Errores

- El sistema debe manejar errores de forma controlada para evitar fallos y pérdida de datos. Esto incluye:
 - Mostrar mensajes claros y comprensibles al usuario.
 - Registrar errores técnicos en un log para revisión administrativa.

- Permitir recuperación automática o manual ante fallos críticos.
- Manejar errores de conexión con la base de datos o servicios externos.

3.2.3.4. Parámetros de Operación

- Intervalo de envío de datos: cada 5 minutos o al detectar evento.
- Formato de datos: estructura con hora, estado de manecillas, animación, presencia y distancia.
- Servicio en la nube: Firebase o MongoDB (según configuración).
- Autenticación: clave API o token seguro.
- Reintentos por fallo de red: hasta 5 veces.
- Almacenamiento local temporal: habilitado si no hay conexión.
- Tiempo de espera por respuesta del servidor: 3 segundos.

3.2.3.5. Generación de salidas

- Formatos de salida: JSON, CSV, XML o visualizaciones dinámicas según el tipo de consumidor (API, usuario final, administrador).
- Consultas personalizadas: Soporte para filtros, paginación y ordenamiento.
- Reportes: Generación automática y programada de reportes a partir de la base de datos (ej. informes mensuales, KPIs, etc.).

3.2.3.6. Relación entre Entradas y Salidas

- Entrada: Usuario selecciona nueva hora en la app.
→ Salida: Se mueven las manecillas del reloj para reflejar ese horario.
- Entrada: Usuario sincroniza el reloj con la hora actual del móvil.
→ Salida: El reloj se actualiza automáticamente.
- Entrada: Usuario cambia la zona horaria.
→ Salida: El reloj ajusta su hora según la nueva zona.

3.2.3.7. Almacenamiento de Base de datos.

- Fecha y hora del ajuste o evento.
- Hora configurada en el reloj.
- ID de usuario que realizó el ajuste (si aplica).
- Resultado del ajuste (éxito o error).
- Mensajes de error si corresponde.

3.2.4. **Requisitos de IoT**

3.2.4.1. Validación de Datos:

- Que la hora obtenida (por sincronización externa o RTC) esté en formato válido (por ejemplo, HH:MM, con horas entre 0–23 y minutos entre 0–59).
- Que los valores del sensor ultrasónico estén dentro de un rango útil (por ejemplo, de 20 cm a 200 cm). Lecturas fuera de este rango se descartan.
- Que la entrada del usuario o modo aleatorio tenga un valor definido para velocidad o patrón de movimiento (Lento, Normal, Rápido, Aleatorio).
- Que la comunicación con los motores y la tira LED esté activa y estable para evitar acciones imprevistas.

3.2.4.2. Secuencia de Operaciones

1. Inicialización de componentes (ESP32, motores, sensor ultrasónico, tira LED).
2. Sincronización de la hora (ya sea por red o desde un reloj interno).
3. Cálculo de posiciones de las manecillas basado en la hora actual.
4. Movimiento de manecillas usando los motores paso a paso.
5. Lectura del sensor ultrasónico periódicamente.
6. Si se detecta presencia:
 - Cambiar modo de movimiento (por ejemplo, animación en espiral o huida).
 - Activar patrones en la tira LED.
7. Guardar datos si es necesario (detecciones, cambios de estado, hora marcada, etc.).

3.2.4.3. Manejo de Errores

- Fallo de red o RTC: Mostrar hora predeterminada o dejar reloj detenido con LED indicador.
- Fallo en motores (sin respuesta o atasco): Reintentar movimiento; si persiste, emitir alerta visual.
- Fallo en sensor ultrasónico: Si se obtienen datos erróneos de forma consecutiva, desactivar animaciones dependientes de proximidad.

- Desincronización de hora: Intentar nueva sincronización cada cierto tiempo.
- Desconexión o error en tira LED: Usar modo seguro (apagado o patrón básico de alerta)

3.2.4.4. Parámetros de Operación

- Velocidades de movimiento: Lento, Normal, Rápido, Aleatorio.
- Frecuencia de lectura del sensor ultrasónico (por ejemplo, cada 1 segundo).
- Modos especiales de animación: Espiral, reversa, huida.
- Intervalo de sincronización horaria (si aplica).
- Rangos válidos para detección de presencia (por ejemplo, entre 20 y 200 cm).

3.2.4.5. Generación de salidas

- Movimiento de manecillas según hora o animación.
- Activación de LEDs con patrones según modo o eventos (por ejemplo, cambio de color cada hora, parpadeo al detectar presencia).
- Mensajes o logs (si hay pantalla o interfaz de comunicación).

3.2.4.6. Relación entre Entradas y Salidas

- Entrada: Hora actual → Salida: Movimiento de manecillas.
- Entrada: Presencia detectada (sensor ultrasónico) → Salida: Movimiento especial (espiral, huida)/Activación de LEDs en color o patrón especial.
- Entrada: Parámetros configurados o aleatorios → Salida: Determina velocidad y estilo de movimiento.
- Entrada: Falla en componente → Salida: Señal visual de error o intento de recuperación.

3.2.4.7. Almacenamiento de Base de datos:

- Guardar: Hora marcada en cada ciclo, Eventos especiales (detección de presencia, animaciones activadas) y Errores detectados.

- Formato: JSON o estructuras simples (fecha, hora, tipo de evento).
- Uso: Para diagnóstico, estadísticas o auditoría del comportamiento del reloj.

3.3. Requisitos no funcionales

Los requisitos no funcionales definen características de calidad que el sistema “Malbouche” debe cumplir, más allá de sus funciones básicas. Estos requisitos garantizan que el sistema sea eficiente, seguro, confiable, fácil de mantener y adaptable a diferentes plataformas.

3.3.1. Requisitos de rendimiento

- El sistema debe permitir que los usuarios interactúen de forma simultánea con la aplicación móvil y la plataforma web sin presentar caídas ni ralentizaciones significativas.
- El backend deberá procesar comandos enviados al reloj con una latencia máxima de 1 segundo bajo condiciones normales de red.
- La app móvil deberá mantener una tasa de actualización del estado del reloj en tiempo real, sincronizándose cada 10 segundos con el backend.

3.3.2. Seguridad

- Las credenciales de usuario (correo y contraseña) deben ser almacenadas en la base de datos con algoritmos de hash seguros, como bcrypt.
- Toda la comunicación entre cliente y servidor se deberá realizar a través de conexiones HTTPS, una vez el sistema se despliegue en la nube.
- Se establecerán restricciones de acceso por roles (administrador, usuario) para limitar acciones dentro del sistema.
- Se recomienda mantener registros de actividad en el backend (logs) para futuras auditorías o depuración.

3.3.3. Fiabilidad

- El sistema debe poder operar de forma estable durante al menos 8 horas continuas sin reinicios ni pérdida de funcionalidad.

- En caso de error en la ejecución de un comando al reloj, el backend deberá emitir una notificación al cliente y permitir el reintento.

3.3.4. *Disponibilidad.*

- En su etapa inicial (alojado localmente), el sistema estará disponible durante el horario de pruebas definido por el equipo.
- En un futuro despliegue en la nube, se busca garantizar una disponibilidad del 99% mensual, asegurando acceso casi continuo a la plataforma.
- Las interrupciones por mantenimiento deberán ser notificadas con al menos 24 horas de anticipación a los usuarios.

3.3.5. *Mantenibilidad*

- El sistema deberá contar con documentación técnica clara para facilitar el mantenimiento por parte de futuros desarrolladores.
- Se recomienda que el código fuente siga buenas prácticas (uso de Git, control de versiones, comentarios, modularidad).
- Las actualizaciones del sistema deberán realizarse sin necesidad de reinstalar completamente la app móvil o la plataforma web, utilizando herramientas como OTA (Over-the-Air updates) si se habilitan con Expo.

3.3.6. *Portabilidad*

- Gracias al uso de React Native, la app móvil podrá ejecutarse tanto en dispositivos Android como iOS sin necesidad de versiones separadas.
- La plataforma web debe ser accesible desde los navegadores Google Chrome y Mozilla Firefox, que han sido usados durante las pruebas.
- La arquitectura modular del backend en Node.js y el uso de MongoDB Atlas permiten su fácil migración a un entorno en la nube (como AWS, Heroku o Vercel) sin necesidad de reescribir componentes clave.

3.4. *Otros requisitos*

Además de los requisitos funcionales y no funcionales previamente definidos, el sistema “Malbouche” debe cumplir con algunas condiciones adicionales derivadas del contexto académico en el que se desarrolla, así como otras consideraciones que podrían influir en su uso o evolución futura:

1. *Requisitos institucionales*

- El proyecto debe ajustarse a los lineamientos y estructura del estándar IEEE 830-1998, requerido por la institución educativa como formato oficial para la documentación de especificaciones de requisitos de software.
- Todos los entregables deben cumplir con los plazos establecidos dentro del calendario académico, lo cual influye directamente en el alcance funcional que puede ser implementado en la primera versión del sistema.

2. *Requisitos culturales y de uso*

- La interfaz del sistema está diseñada inicialmente para un entorno de habla hispana, considerando que los usuarios objetivos se encuentran en un contexto cultural y lingüístico hispanohablante. Esto afecta la terminología utilizada en la app y en la web.
- La organización visual del contenido en la plataforma debe seguir convenciones familiares para los usuarios, como formatos de fecha, horas en formato de 24 horas, y menús desplegables estándar.

3. *Requisitos legales (potenciales)*

- Aunque el sistema no gestiona información sensible de alto riesgo, se planea que en el futuro se adopten prácticas de seguridad mínimas para proteger datos de usuarios, conforme a las buenas prácticas generales de privacidad (por ejemplo, evitar almacenamiento de contraseñas en texto plano, cifrado en tránsito de datos, etc.).
- En caso de evolucionar a un entorno comercial, será necesario evaluar el cumplimiento de leyes de protección de datos personales, especialmente si se recopilan estadísticas, datos

de comportamiento o historial de acciones de los usuarios.

4. Licencias y propiedad intelectual

- El sistema, incluyendo su código fuente, documentación y diseño, es de propiedad del equipo de desarrollo, en el contexto de un proyecto académico. Cualquier uso posterior deberá contar con autorización expresa del equipo responsable.
- Las tecnologías utilizadas (React Native, Node.js, MongoDB Atlas, etc.) son de código abierto o con licencias compatibles con su uso académico y no comercial.

4. Apéndices

Dado que la aplicación móvil estará conectada directamente al reloj físico, a continuación se presenta el modelo correspondiente del dispositivo. Este fue proporcionado por el equipo encargado de su fabricación, específicamente por el área de Mecatrónica.

https://drive.google.com/file/d/1ikt4WR786VOpHC-gRBSj1_6eA-t0NEF2/view?usp=sharing

C. Repositorio Github:

<https://github.com/jl-lara/Malbouche-backend>

D. Plan de pruebas

I. Planificación de pruebas.....	4
A. Alcance.....	4
B. Objetivos.....	4
C. Tipos de pruebas a realizar.....	5
D. Recursos necesarios.....	6
E. Roles y responsabilidades.....	7
F. Criterios de entrada y salida.....	8
G. Entregables.....	9
II. Análisis y diseño de pruebas.....	10
A. Casos de prueba.....	10
B. Estrategias.....	12
C. Datos de prueba.....	12
III. Implementación y ejecución de pruebas:.....	13
IV. Evaluación de los resultados y cierre.....	21
A. Resumen de las pruebas de aplicación Móvil.....	21

B. Fallos detectados.....	21
C. Conclusiones y recomendaciones.....	21
D. Resumen de las pruebas del Sitio Web.....	22
E. Fallos detectados.....	22
F. Conclusiones y recomendaciones.....	23
V. Gestión de defectos.....	24
A. Registro de Defectos y Pruebas Pendientes.....	24
B. Acciones Inmediatas y Pasos a Seguir.....	26
C. Acciones de solución implementadas.....	27
VI. Evaluación de salida y cierre.....	29
A. Cierre formal del proceso de pruebas.....	29

I. Planificación de pruebas

A. Alcance

El alcance de este plan de pruebas abarca la verificación y validación del sistema Malbouche, asegurando que cumpla con los requisitos funcionales y no funcionales definidos, así como con los estándares de calidad establecidos. Se incluirán pruebas de componentes individuales, integración entre módulos, y el sistema completo, con un enfoque en funcionalidades críticas, rendimiento, seguridad, entre otros.

B. Objetivos

Los principales objetivos de esta planificación de pruebas son:

- ❖ **Garantizar la calidad del software:** Identificar y corregir defectos para asegurar que el sistema sea estable, fiable y funcione según lo esperado.
- ❖ **Verificar el cumplimiento de los requisitos:** Confirmar que todas las funcionalidades implementadas satisfacen las especificaciones definidas en los documentos de requisitos.
- ❖ **Asegurar la usabilidad:** Evaluar si el sistema es intuitivo y fácil de usar para los usuarios finales.
- ❖ **Minimizar riesgos:** Reducir la probabilidad de fallos en producción y sus posibles impactos negativos.

C. Tipos de pruebas a realizar

Se llevarán a cabo los siguientes tipos de pruebas:

Pruebas Unitarias: Realizadas por los desarrolladores para validar unidades individuales de código (funciones, métodos, etc).

Pruebas de Integración: Verifican la interacción entre diferentes módulos o componentes del sistema.

Pruebas Funcionales: Confirman que cada función del sistema opera de acuerdo con las especificaciones.

Pruebas de Regresión: Aseguran que los cambios o adiciones recientes al código no han introducido nuevos defectos en funcionalidades existentes.

Pruebas de Seguridad: Identifican vulnerabilidades y aseguran que el sistema está protegido contra accesos no autorizados.

Pruebas de Usabilidad: Evalúan la facilidad de uso y la experiencia del usuario con el sistema.

Pruebas de Compatibilidad: Verificar el funcionamiento del sistema en diferentes entornos (navegadores y dispositivos).

D. Recursos necesarios

Los recursos clave para la ejecución de este plan de pruebas incluyen:

Recursos Humanos:

- Desarrolladores para soporte y corrección de defectos.
- Usuarios finales.
- Líder de pruebas.

Software:

Entornos de pruebas como sistemas operativos, bases de datos, navegadores web.

Hardware:

- Estaciones de trabajo para el equipo de pruebas.
- Dispositivos móviles.

Documentación:

- Documentos de requisitos (funcionales y no funcionales).
- Plan de pruebas.

E. Roles y responsabilidades:

Líder de Pruebas:

- ❖ Define la estrategia y el plan de pruebas.
- ❖ Asigna tareas y gestiona al equipo de pruebas.
- ❖ Supervisa el progreso y el estado de las pruebas.

QA Analysts:

- ❖ Diseñan, desarrollan y ejecutan casos de prueba.
- ❖ Identifican, documentan y dan seguimiento a los defectos.
- ❖ Colaboran con los desarrolladores para resolver problemas.

Desarrolladores:

- ❖ Escriben código y realizan pruebas unitarias.
- ❖ Corrigen los defectos reportados por el equipo de QA.
- ❖ Colaboran con el equipo de pruebas para reproducir y resolver problemas.

F. Criterios de entrada y salida

Criterios de Entrada (Cuando las pruebas pueden empezar)

- ❖ Todos los requisitos funcionales y no funcionales para la aplicación móvil, web, IoT y base de datos están definidos y documentados.
- ❖ Los módulos principales de cada componente (móvil, web, IoT) han completado sus pruebas unitarias iniciales.
- ❖ Los entornos de prueba (servidores web, base de datos en la nube, dispositivos IoT, emuladores móviles) están configurados y accesibles.
- ❖ Los recursos humanos y materiales necesarios están asignados.

Criterios de Salida (Cuando las pruebas pueden considerarse completas)

- ❖ Todos los casos de prueba críticos y de alta prioridad para la aplicación móvil, web, IoT y sus integraciones han sido ejecutados con éxito.
- ❖ Todos los defectos de severidad crítica y alta prioridad han sido corregidos, verificados y cerrados.
- ❖ Los defectos restantes de menor prioridad han sido registrados y evaluados.
- ❖ La cobertura de las pruebas (funcional y de requisitos) ha alcanzado el nivel deseado para cada componente.
- ❖ Los resultados de las pruebas de rendimiento cumplen con los requisitos definidos para la aplicación web y el sistema IoT.

G. Entregables

Los entregables de este proceso de planificación y ejecución de pruebas son:

Plan de Pruebas del Sistema ERS "Malbouche": Este documento detallado.

Casos de Prueba:

- Casos de Prueba para la Aplicación Móvil (control del reloj).
- Casos de Prueba para la Aplicación Web I4.0 (funcionalidades del bar, dashboard, administración).

Informes de Defectos (Bug Reports): Documentación de todos los defectos encontrados en la aplicación móvil, web, IoT y base de datos.

II. Análisis y diseño de pruebas

A. Casos de prueba

Casos de prueba de aplicación móvil		
ID de la prueba	Módulo	Descripción
TST01	Login	Inicio de sesión con datos correctos
TST02	Login	Inicio de sesión con datos incorrectos
TST03	Usuarios	Registrar usuarios con datos válidos
TST04	Usuarios	Registrar usuario con datos inválidos
TST05	Usuarios	Editar campos de un usuario
TST06	Reloj	Mover manecillas hacia la derecha
TST07	Reloj	Mover manecillas hacia la izquierda
TST08	Reloj	Mover manecillas en direcciones opuestas
TST09	Reloj	Balancear las manecillas.
TST10	Reloj	Poner las manecillas en la hora actual.
TST11	Eventos	Registrar un nuevo evento.
TST12	Eventos	Editar un evento.
TST13	Eventos	Reproducir movimiento de eventos en el reloj.
TST14	Movimientos	Registrar un nuevo movimiento.
TST15	Movimientos	Editar un movimiento.
TST16	Movimientos	Reproducir

Casos de prueba de aplicación web		
ID de la prueba	Módulo	Descripción
TST01	Menú (básico)	Visualizar menús provenientes de la base de datos.
TST02	Reservaciones (admin)	Realizar reservación con valores correctos.
TST03	Reservaciones (admin)	Realizar reservaciones con valores no válidos.
TST04	Eventos	Visualizar eventos provenientes de la base de datos.
TST05	Login	Iniciar sesión con datos correctos
TST06	Login	Iniciar sesión con datos incorrectos
TST07	Usuarios	Visualizar usuarios registrados
TST08	Usuarios	Registrar usuarios
TST09	Menú	Registrar un menú
TST10	Reservaciones (admin)	Visualizar reservaciones
TST11	Eventos (admin)	Registrar un nuevo evento
TST12	Dashboard	Recibe información en tiempo real de los sensores.

B. Estrategias

Para la realización de pruebas se estarán utilizando las pruebas manuales, esto ayudará a facilitar la aproximación entre el tester y el usuario final en beneficio de la experiencia del cliente. Se pretende testear módulo por módulo de manera individual, para después realizar el testeo de integración para verificar que se estén enviando, recibiendo y mostrando los datos de manera correcta.

C. Datos de prueba

Los datos que serán necesarios para la realización de las pruebas son los siguientes:

- ❖ Información personal del usuario final:
 - Nombre completo
 - Correo electrónico
 - Contraseña
- ❖ Información del reloj
 - Sentido de la manecillas
 - Velocidad de las manecillas
- ❖ Información de eventos de reloj
 - Días en que se realizará el evento
 - Rango de hora que durará el evento
 - Nombre del evento
- ❖ Información de eventos del bar
 - Nombre del evento
 - Descripción del evento
 - Fecha y hora
- ❖ Reservaciones de mesas
 - Número de personas
 - Fecha y hora de reservación

III. Implementación y ejecución de pruebas:

Casos de prueba de la aplicación Móvil							
Test ID	Descripción de caso de prueba	Pre condiciones	Pasos de pruebas (Estrategia)	Datos de prueba necesarios	Resultados esperados	Resultado obtenidos (Ejecución)	Pass/Fail (Ejecución)
TST01	Iniciar Sesión con datos correctos	<ul style="list-style-type: none"> • Tener una cuenta dada de alta en el sistema • No tener la sesión iniciada • Estar dentro de la aplicación 	<ol style="list-style-type: none"> 1. Abrir la aplicación. 2. Ingresar tu correo. 3. Ingresar tu contraseña. 4. Presionar el botón de iniciar sesión. 	<ol style="list-style-type: none"> 1. Correo: lara@gmail.com 2. Contraseña: 123contraseña5 	El login será exitoso	Login exitoso	PASS
TST02	Iniciar Sesión con datos incorrectos	<ul style="list-style-type: none"> • Estar dado de alta. • No tener la sesión iniciada. • Estar dentro de 	<ol style="list-style-type: none"> 1. Abrir la aplicación. 2. Ingresar correo incorrecto. 3. Ingresar contraseña incorrecta. 	<ol style="list-style-type: none"> 1. Usuario: almendro@gmail,com 2. Contraseña: 123456 	El login no será exitoso	El login no es exitoso	PASS

		la app.					
TST03	Registrar usuarios con datos válidos	<ul style="list-style-type: none"> • Estar dentro de la aplicación • Ser administrador de la aplicación 	<ol style="list-style-type: none"> 1. Ingresar a la sección “Usuarios”. 2. Presionar la opción “+” que es para registrar nuevo usuario. 3. Ingresar los datos del nuevo usuario 4. Presionar “Registrar usuario”. 	<ol style="list-style-type: none"> 1. Nombre completo del usuario: Lara López 2. Correo del usuario: lara@utt.com 3. Rol: Administrador 	El usuario será registrado con éxito.	El registro es exitoso	PASS
TST04	Registrar usuario con datos inválidos	<ul style="list-style-type: none"> • Estar dentro de la aplicación • Ser administrador de la aplicación 	<ol style="list-style-type: none"> 1. Ingresar a la sección “Usuarios”. 2. Presionar la opción “+” que es para registrar nuevo usuario. 3. Ingresar los datos del nuevo 	<ol style="list-style-type: none"> 5. Nombre completo del usuario: Correo del usuario: 6. Rol: Administrador 	El usuario no será registrado con éxito.	El registro no ha sido exitoso	PASS

			4. usuario 4. Presionar “Registrar usuario”.				
TST05	Editar campos de un usuario	<ul style="list-style-type: none"> • Estar dentro de la aplicación • Ser administrador de la aplicación • Tener usuarios registrados 	<ol style="list-style-type: none"> 1. Ingresar a la sección “Usuarios”. 2. Presionar la opción “Actualizar usuario”. 3. Modificar los datos. 4. Presionar “Actualizar”. 	Datos actualizados del usuario.	El usuario se actualizará con éxito.	La actualización ha sido exitosa.	PASS
TST06	Mover manecillas hacia la derecha	Estar dentro de la aplicación.	<ol style="list-style-type: none"> 1. Estar en la sección de “Home.” 2. Presionar el botón “Right” 	No data	El movimiento se realizará con éxito.	El movimiento se realizó con éxito.	PASS
TST07	Mover manecillas hacia la izquierda	Estar dentro de la aplicación.	<ol style="list-style-type: none"> 3. Estar en la sección de “Home.” 4. Presionar el botón “Left” 	No data	El movimiento se realizará con éxito.	El movimiento se realizó con éxito.	PASS
TST08	Mover manecilla	Estar dentro de la	5. Estar en la sección de	No data	El movimiento	El movimiento	PASS

	s en direcciones opuestas	aplicación.	“Home.” 6. Presionar el botón “Crazy”		o se realizará con éxito.	se realizó con éxito.	
TST09	Mover manecillas de reloj para que se balanceen	Estar dentro de la aplicación.	1. Estar en la sección de “Home.” 2. Presionar el botón “Swing”.	No data	El movimiento se realizará con éxito.	El movimiento no se realizó con éxito.	FAIL
TST10	Poner las manecillas en la hora actual	Estar dentro de la aplicación.	1. Estar en la sección de “Home.” 2. Presionar el botón “Normal”.	No data	El movimiento se realizará con éxito.	El movimiento no se realizó con éxito.	FAIL
TST11	Registrar un evento	Estar dentro de la aplicación.	1. Estar dentro de la sección “Events”. 2. Presionar la opción “+” que es para registrar un nuevo evento. 3. Llenar el formulario	1. Hora inicio: 10:20 pm 2. Hora fin: 11:20 pm 3. Seleccionar días: M T W 4. Nombre del evento: Noche loca.	El registró se realizará con éxito.	El registro se realizó con éxito.	PASS

				5. Seleccionar movimiento: crazy.			
TST12	Editar un evento	Estar dentro de la aplicación.	1. Estar dentro de la sección “Events”. 2. Seleccionar un evento. 3. Editar los campos.	Campos actualizados.	La edición se realizó con éxito.	La edición se realizó con éxito	PASS
TST13	Registrar un movimiento	Estar dentro de la aplicación.	4. Estar dentro de la sección “Movements” . 5. Presionar la opción “+” que es para registrar un nuevo movimiento. 6. Llenar el formulario.	1. Nombre del movimiento: Crazy 2 2. Movimiento para manecilla de horas: left. 3. Velocidad manecilla de horas: 50 4. Movimiento para manecilla de	El registró se realizará con éxito.	El registro se realizó con éxito.	PASS

				minutos: left 5. Velocida d manecilla de minutos: 100				
TST14	Editar un movimiento	Estar dentro de la aplicación.	<ol style="list-style-type: none"> 1. Estar dentro de la sección “Movements” 2. Seleccionar un movimiento. 3. Editar los campos. 	Campos actualizados.	La edición se realizó con éxito.	La edición se realizó con éxito	PASS	

Casos de prueba de la aplicación web							
Test ID	Descripción de caso de prueba	Pre condiciones	Pasos de pruebas (Estrategia)	Datos de prueba necesarios	Resultados esperados	Resultado obtenidos (Ejecución)	Pass/Fail (Ejecución)
TST01	Visualizar menús	Estar en el inicio del sitio web.	Ingresar a la sección de menú.	No data	Se podrá visualizar los menús disponibles.	Se logró visualizar los menús exitosamente.	PASS
TST02	Realizar reserva con valores correctos.	Estar en el inicio del sitio web.	1. Ingresar a la sección de "reservaciones" 2. Llenar el formulario	1. Número de personas: 4 2. Fecha: 20/06/2025 3. Hora de inicio: 10:00 pm 4. Nombre de quien reserva: Alejandro Díaz 5. Correo: alejandro1	Se podrá registrar con éxito la reservación.	Se logró exitosamente registrar la reservación.	PASS

				23@gmail.com			
TST03	Realizar reservaciones con valores no válidos.	Estar en el inicio del sitio web.	1. Ingresar a la sección de "Reservaciones" 2. Llenar el formulario	1. Número de personas: 4 2. Fecha: 20/06/2025 3. Hora de inicio: 10:00 pm 4. Nombre de quien reserva: Alejandro Díaz 5. Correo:	No se podrá registrar con éxito la reservación.	Se logró exitosamente registrar la reservación.	FAIL
TST04	Visualizar eventos provenientes de la base de datos.	Estar en el inicio del sitio web.	Ingresar a la sección de "Eventos"	No data	Se podrá visualizar los eventos disponibles.	Se logró visualizar los eventos exitosamente.	PASS
TST05	Iniciar sesión con datos correctos	<ul style="list-style-type: none"> • Tener una cuenta dada de alta en el sistema • No tener 	1. Entrar al sitio web. 2. Ingresar tu correo. 3. Ingresar tu contraseña 4. Presionar el botón de	1. Correo: lara@gmail.com 2. Contraseña: 123contra5	El login será exitoso	Login exitoso	PASS

		<ul style="list-style-type: none"> • la sesión iniciada • Estar dentro de la aplicación. 	iniciar sesión.				
TST06	Iniciar sesión con datos incorrectos	<ul style="list-style-type: none"> • Estar dado de alta. • No tener la sesión iniciada. • Estar dentro de la app. 	<ol style="list-style-type: none"> 1. Abrir la aplicación. 2. Ingresar correo incorrecto. 3. Ingresar contraseña incorrecta. 	<ol style="list-style-type: none"> 3. Usuario: almendro.com 4. Contraseña: 123456 	El login no será exitoso	El login no es exitoso	PASS
TST07	Visualizar usuarios registrados	<ul style="list-style-type: none"> • Estar dentro de la aplicación . • Ser administrador de la aplicación. • Tener 	Ingresar a la sección "Usuarios".	No data	Se podrá visualizar los usuarios disponibles.	IN PROGRESS	IN PROGRESS

		usuarios registrados					
TST08	Registrar usuarios	<ul style="list-style-type: none"> • Estar dentro de la aplicación. • Ser administrador de la aplicación. 	<ol style="list-style-type: none"> 1. Ingresar a la sección “Usuarios”. 2. Presionar la opción “registrar nuevo usuario”. 3. Ingresar los datos del nuevo usuario 4. Presionar “Registrar usuario”. 	<ol style="list-style-type: none"> 1. Nombre completo del usuario: Lara López 2. Correo del usuario: lara@utt.com 3. Rol: Administrador 	El usuario será registrado con éxito.	IN PROGRESS	IN PROGRESS
TST09	Registrar un menú	<ul style="list-style-type: none"> • Estar dentro de la aplicación. • Ser administrador de la aplicación 	<ol style="list-style-type: none"> 1. Ingresar a la sección “Menú”. 2. Presionar la opción “registrar nuevo Menú”. 3. Ingresar los datos del nuevo usuario 4. Presionar “Registrar 	<ol style="list-style-type: none"> 1. Nombre del menú: Drinks 2. Nombre del platillo/bebida: Paloma 3. Ingredientes: tequila, toronja, refresco, hielo. 	El menú será registrado con éxito.	IN PROGRESS	IN PROGRESS

			Menú”.				
TST10	Visualizar reservaciones	<ul style="list-style-type: none"> • Estar dentro de la aplicación. • Ser administrador de la aplicación 	Ingresar a la sección “Reservaciones”.	No data	Se podrá visualizar las reservaciones registradas.	IN PROGRESS	IN PROGRESS
TST11	Registrar un nuevo evento	<ul style="list-style-type: none"> • Estar dentro de la aplicación. • Ser administrador de la aplicación 	<ol style="list-style-type: none"> 1. Ingrese a la sección de “Eventos” 2. Presionar la opción “registrar nuevo Menú”. 3. Ingresar los datos 	<ol style="list-style-type: none"> 1. Nombre del evento: Noche de Rock 2. Fecha: 24/08/2025 3. Hora de inicio: 10:00 pm 4. Hora fin: 12:00 5. Descripción: La mejor noche de rock que pudieras tener. 	Se podrá registrar un evento de manera exitosa.	IN PROGRESS	IN PROGRESS

TST12	Visualiza la información en tiempo real de los sensores.	<ul style="list-style-type: none"> ● Estar dentro de la aplicación. ● Ser administrador de la aplicación 	Ingresar a la sección de “Dashboard”	No data	Se podrá visualizar la información de los sensores en formato de gráficas.	IN PROGRESS	IN PROGRESS
-------	--	--	--------------------------------------	---------	--	-------------	-------------

IV. Evaluación de los resultados y cierre.

A. Resumen de las pruebas de aplicación Móvil

Se evaluaron todas las funciones principales de la aplicación móvil mediante la ejecución de 14 casos de prueba. Los resultados fueron:

- ❖ 12 pruebas exitosas (TST01-TST08, TST11-TST14)
- ❖ 2 pruebas fallidas (TST09 y TST10)

Esto representa una tasa de aprobación del 85.7%, lo que indica que la mayoría de las funcionalidades operan correctamente, aunque se requiere atención a los casos fallidos antes del despliegue final.

B. Fallos detectados

Los principales problemas se encontraron en el módulo del reloj de la aplicación móvil:

- ❖ TST09: La función de balanceo de manecillas no opera según lo especificado.
- ❖ TST10: Falla en el ajuste automático a la hora actual.

Estos defectos fueron clasificados como de alta prioridad debido a su impacto directo en la experiencia del usuario.

C. Conclusiones y recomendaciones

El sistema demuestra un buen nivel de calidad general, pero requiere:

- ❖ Corrección inmediata de los defectos TST09 y TST10
- ❖ Pruebas adicionales de escalabilidad en la aplicación web
- ❖ Ajustes menores en la interfaz del módulo de Eventos

Con estas mejoras implementadas, el sistema estará listo para su implementación final con un alto estándar de calidad.

D. Resumen de las pruebas del Sitio Web

Se ejecutaron 12 casos de prueba para validar las funcionalidades críticas de la aplicación web. Los resultados fueron:

- ❖ 9 pruebas exitosas (TST01-TST02, TST04-TST06, TST09-TST12).
- ❖ 1 prueba fallida (TST03).
- ❖ 2 pruebas en progreso (TST07-TST08).

Tasa de aprobación: 75% (considerando solo las pruebas completadas).

E. Fallos detectados

Prueba TST03 (Fallida):

- ❖ Descripción: La reservación se registró exitosamente a pesar de ingresar un correo electrónico inválido (campo vacío).
- ❖ Severidad: Alta (afecta la integridad de los datos).
- ❖ Recomendación: Validar campos obligatorios en el formulario de reservaciones.

Pruebas en progreso (TST07-TST12):

- ❖ Visualización de usuarios registrados y registro de nuevos usuarios.
- ❖ Estado: Pendiente de ejecución completa.

F. Conclusiones y recomendaciones

Acciones prioritarias:

- ❖ Corregir fallo en validación de formularios (TST03)
- ❖ Completar pruebas pendientes (TST07-TST12)
- ❖ Optimizar rendimiento para alta demanda
- ❖ Mejorar retroalimentación visual en interfaces

Próximos pasos:

- ❖ Finalizar pruebas pendientes
- ❖ Elaborar informe detallado de defectos
- ❖ Priorizar correcciones para siguiente iteración

V. Gestión de defectos.

En esta sección se documenta el registro y estado de los defectos identificados durante la fase de ejecución de pruebas del sistema "Malbouche". El objetivo es proporcionar un panorama claro de los problemas pendientes que deben ser resueltos por el equipo de desarrollo.

A. Registro de Defectos y Pruebas Pendientes

A continuación, se presenta el listado de los casos de prueba que resultaron en un fallo o que se encuentran pendientes de ejecución.

Aplicación Móvil

ID del Caso	Módulo	Descripción del Fallo	Prioridad
TST09	Reloj	El movimiento para que las manecillas se balanceen ("Swing") no se realizó con éxito.	Alta
TST10	Reloj	El movimiento para poner las manecillas en la hora actual ("Normal") no se realizó con éxito.	Alta

Aplicación Web.

ID del Caso	Módulo	Objetivo de la Prueba / Descripción del Fallo	Estado Actual	Prioridad
TST03	Reservaciones (Admin)	Fallo: Se permitió registrar una reserva con el campo de correo electrónico vacío, violando las reglas de validación.	FAIL	Alta
TST07	Usuarios	Visualizar los usuarios registrados desde el panel de administrador.	IN PROGRESS	Pendiente
TST08	Usuarios	Registrar nuevos usuarios en el sistema desde el panel de administrador.	IN PROGRESS	Pendiente
TST09	Menú	Registrar un nuevo menú en el sistema desde el panel de administrador.	IN PROGRESS	Pendiente

TST10	Reservaciones (Admin)	Visualizar el listado de reservaciones registradas en el sistema.	IN PROGRESS	Pendiente
TST11	Eventos (Admin)	Registrar un nuevo evento desde el panel de administrador.	IN PROGRESS	Pendiente
TST12	Dashboard	Verificar la correcta visualización de los datos de los sensores en tiempo real.	IN PROGRESS	Pendiente

B. Acciones Inmediatas y Pasos a Seguir.

1. Corrección de Defectos en la Aplicación Móvil (TST09 y TST10):

- Acción: El equipo de desarrollo debe priorizar la corrección de los fallos de alta prioridad en el módulo del reloj. Específicamente, se revisará el código encargado de controlar los movimientos "Swing" y "Normal" de las manecillas.
- Pasos a Seguir: Una vez corregidos los errores, el equipo de QA realizará pruebas de verificación para confirmar que las funciones operan como se espera. Adicionalmente, se ejecutarán pruebas de regresión en todo el módulo del reloj para asegurar que las correcciones no introdujeron nuevos defectos.

2. Resolución de Fallo de Validación en la Aplicación Web (TST03):

- Acción: Se ha notificado al equipo de desarrollo sobre el fallo de severidad alta que permite el registro de reservaciones sin un correo electrónico válido.
- Pasos a Seguir: Desarrollo implementará una validación en el formulario de reservaciones para que el campo de correo electrónico sea obligatorio. Posteriormente, QA ejecutará nuevamente el caso de prueba TST03 para confirmar que la validación impide el registro con datos inválidos y el TST02 para asegurar que el registro con datos correctos sigue funcionando.

3. Finalización de Pruebas Pendientes en la Aplicación Web (TST07 - TST12):

- Acción: El equipo de QA debe proceder con la ejecución de los casos de prueba que se encuentran en estado "IN PROGRESS".
- Pasos a Seguir: Se completarán las pruebas de los módulos de Usuarios, Menú, Reservaciones, Eventos y Dashboard del panel de administrador. Cualquier defecto encontrado será documentado y priorizado siguiendo el mismo proceso de gestión.

C. Acciones de solución implementadas

1. TST09 – Balanceo de manecillas (“Swing”) en la aplicación móvil

- Acción tomada: Se revisó y refactorizó la estructura del movimiento en la base de datos, además de corregir el flag de animación.
- Acción de QA: Se re-ejecutó TST09 y validó que las manecillas efectuaran el balanceo correctamente.
- Estado final: Cerrado.

2. TST10 – Ajuste automático de hora (“Normal”) en la aplicación móvil

- Acción tomada: Se corrigió el cálculo de offset de zona horaria y se añadió manejo de excepción para valores nulos de reloj.
- Acción de QA: Se volvió a ejecutar TST10, confirmó que la hora se ajusta correctamente bajo distintas zonas y que no hay errores en consola.
- Estado final: Cerrado.

3. TST03 – Validación de correo en Web (Reservaciones Admin)

- Acción tomada: Se implementó validación front-end y back-end para campo de correo en el formulario de reservaciones.
- Acción de QA: Se re-ejecutó TST03 y el caso negativo equivalente (TST02), confirmó que el registro sin correo ahora ya no es permitido.
- Estado final: Cerrado.

4. Verificación y reporte de cierre

- Acción a tomar: Se ejecutarán de nuevo por bloques todos los casos corregidos para garantizar ausencia de.
- Acción de documentación: Se generarán los informes de defectos corregidos y el estado final de cada problema.
- Estado final: En progreso.

VI. Evaluación de salida y cierre

Esta fase tiene como objetivo cerrar formalmente el proceso de pruebas, revisar los resultados obtenidos, identificar defectos críticos, registrar lecciones aprendidas y confirmar que el sistema está en condiciones para avanzar a la siguiente etapa del ciclo de desarrollo.

Ambas plataformas presentan un buen desempeño general. Sin embargo, los errores detectados deben atenderse antes del despliegue para asegurar una experiencia estable y confiable.

A. Cierre formal del proceso de pruebas

Con base en la evaluación de los resultados y la gestión de los defectos, se procede al cierre formal del proceso de pruebas:

- ❖ **Revisión Final:** Se ha realizado una revisión exhaustiva del plan de pruebas, los casos de prueba ejecutados, los informes de defectos y los resultados obtenidos.
- ❖ **Lecciones Aprendidas:** Se documentaron lecciones aprendidas sobre ejecución, herramientas y comunicación.
- ❖ **Entrega de Artefactos:** Todos los entregables definidos en el plan de pruebas han sido finalizados.
- ❖ **Aprobación para Siguiente Fase:** Una vez que las correcciones de los defectos de alta prioridad sean verificadas, el sistema estará en condiciones de pasar a la siguiente fase del ciclo de vida del desarrollo, ya sea Pruebas de Aceptación del Usuario o despliegue.

En definitiva, este plan ha sido fundamental para identificar riesgos y asegurar la calidad del sistema. Con las correcciones recomendadas y el enfoque continuo en la mejora, el sistema estará bien posicionado para ofrecer una experiencia de usuario sólida y cumplir con sus propósitos. Además, se valoró positivamente el trabajo en

equipo durante las pruebas, así como la disposición para resolver los problemas encontrados.

