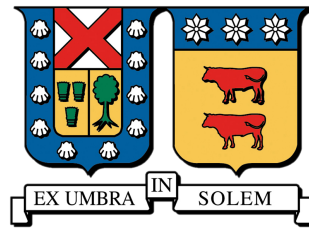


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAÍSO - CHILE



«APLICACIÓN MÓVIL DE APRENDIZAJE CULINARIO GAMIFICADO - BACKEND»

JAVIER ZAMORA BREVIS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL TELEMÁTICO

PROFESOR GUÍA:

DR. MARCOS DAVID ZÚÑIGA BARRAZA

CO-REFERENTE:

DR. PATRICIO ANDRÉS OLIVARES RONCAGLIOLO

Diciembre 2025



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título; ☐ Tesis de Postgrado;

Título del trabajo: Aplicación Móvil de Aprendizaje
Culinario Gamificado - Backend

Nombre del candidato(a): Javier Alexander Zamora Brevis

Carrera / Grado: Ingeniería Civil Telemática

Campus: Casa Central Valparaíso ; **Departamento:** Electrónica/Telemática

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, **Marcos David Zúñiga Barraza**, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL

☒ El trabajo **NO contiene información que amerite confidencialidad** y puede ser publicado de inmediato en repositorio con acceso abierto.

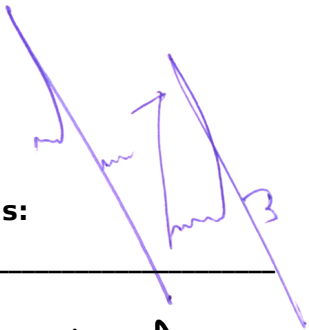
☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por:

☐ 6 meses; ☐ 12 meses; ☐ 2 años; ☐ 3 años; ☐ 5 años; ☐ 10 años

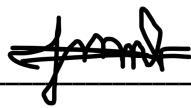
Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 09/12/2025; **Firma:** 

Estudiante o Candidato(a):

Fecha: 09/12/2025; **Firma:** 

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

Resumen

El aprendizaje culinario en plataformas digitales suele carecer de retroalimentación práctica y personalización, limitando la motivación y el progreso del usuario. Esta memoria presenta el diseño y desarrollo de la arquitectura técnica de **SHEFU**, una aplicación móvil gamificada que integra modelos de visión computacional para la evaluación automática de preparaciones culinarias.

La solución se construyó sobre una arquitectura modular utilizando Flutter para el cliente y Firebase como backend *serverless* (BaaS), gestionando la autenticación, la persistencia de datos y la lógica de negocio. Se destaca la implementación del módulo de Visión Computacional, el cual ejecuta un modelo EfficientNet-Lite0 directamente en el dispositivo (*on-device*) mediante TensorFlow Lite. Esta estrategia permitió evaluar la presentación de los platos mediante criterios objetivos con una latencia de inferencia inferior a 3 s, garantizando una experiencia fluida sin dependencia de conectividad constante. Adicionalmente, se desarrolló un sistema de gamificación integral que gestiona rankings, moneda virtual y una tienda de accesorios.

La validación del sistema, realizada con usuarios finales, arrojó un puntaje de usabilidad SUS de 85.0 (grado «Excelente») y un NPS de +50, confirmando una alta aceptación y lealtad. Los resultados demuestran que la arquitectura propuesta es robusta y escalable, soportando eficientemente la integración de tecnologías avanzadas en un entorno móvil para mejorar la experiencia de aprendizaje.

Palabras clave: Arquitectura Serverless, Flutter, Firebase, Visión Computacional On-Device, Gamificación, TensorFlow Lite.

Abstract

Culinary learning on digital platforms often lacks practical feedback and personalization, limiting user motivation and progress. This thesis presents the design and development of the technical architecture for SHEFU, a gamified mobile application that integrates artificial intelligence for the automatic evaluation of culinary preparations.

The solution was built upon a modular architecture using Flutter for the client and Firebase as a *serverless* backend (BaaS), handling authentication, data persistence, and business logic. A key contribution is the implementation of the Computer Vision module, which runs an EfficientNet-Lite0 model directly on-device using TensorFlow Lite. This strategy allowed for the evaluation of dish presentation based on objective criteria with an inference latency of less than 3 s, ensuring a fluid experience without reliance on constant connectivity. Additionally, a comprehensive gamification system was developed to manage rankings, virtual currency, and an accessories store.

System validation with end-users yielded a SUS score of 85.0 («Excellent» grade) and an NPS of +50, confirming high acceptance and loyalty. The results demonstrate that the proposed architecture is robust and scalable, efficiently supporting the integration of advanced technologies in a mobile environment to enhance the learning experience.

Keywords: Serverless Architecture, Flutter, Firebase, On-Device Computer Vision, Gamification, TensorFlow Lite.

Índice general

1. Introducción	5
1.1. Contexto	5
1.2. Problema a Resolver	6
1.3. Acercamiento a la Solución	6
1.4. Hipótesis	7
1.4.1. Objetivos Específicos del Proyecto	8
1.5. Estructura de la Memoria	8
2. Estado del Arte	10
2.1. Soluciones Existentes	10
2.2. Herramientas y Métodos (Marco Teórico)	12
2.3. Discusión	13
3. Desarrollo de la Solución	15
3.1. Análisis de Requerimientos	15
3.1.1. Requisitos Funcionales	16
3.1.2. Requisitos No Funcionales	17
3.2. Diseño	19
3.2.1. Definición de módulos	19
3.2.2. Interacciones entre módulos y tecnologías	21
3.2.3. Esquema General de la Arquitectura	21
3.2.4. Modelo de Datos	22
3.3. Implementación y Pruebas Unitarias	27
3.3.1. Configuración del Entorno y Tecnologías	27
3.3.2. Implementación de Módulos	28
3.3.3. Pruebas Unitarias y de Integración	32

4. Validación	36
4.1. Metodología de Validación	36
4.2. Tareas y Métricas de Evaluación	37
4.3. Resultados Obtenidos	38
4.3.1. Análisis de Resultados Cuantitativos	40
4.3.2. Análisis de Resultados Cualitativos	40
4.4. Discusión y Verificación de Hipótesis	41
5. Conclusiones y Trabajo Futuro	42
5.1. Conclusión	42
5.2. Trabajo Futuro	43
A. Anexos	47
A.1. Resultados Detallados de la Validación	47

Agradecimientos

Este trabajo representa el cierre de una etapa fundamental en mi vida, y su realización no habría sido posible sin el apoyo, la guía y la compañía de muchas personas a quienes dedico estas líneas.

Quiero expresar mi sincero agradecimiento a mi profesor guía, Marcos Zuñiga, y a los encargados del desafío, Patricio Olivares y Samuel, por su orientación y confianza en este proyecto. Un reconocimiento a mi equipo de trabajo: Carolina, Anays y Allison. Gracias por el compromiso, por las noches de programación y diseño, y por la resiliencia para superar cada desafío técnico que se nos presentó. Este proyecto es el resultado de nuestra sinergia y esfuerzo conjunto.

A mi familia, el pilar fundamental de todo lo que soy. Gracias por su amor incondicional, por siempre creer en mi y por el sacrificio silencioso que hicieron para permitirme llegar hasta aquí. Una mención especial a mis gatas, a pesar de que dos de tres no se encuentran con nosotros fueron una parte esencial de mi vida los primeros años universitarios.

A mis amigos, tanto a los que han estado siempre como a los que el camino universitario me regaló. Agradezco profundamente a quienes conocí en medio de la incertidumbre de la pandemia y las clases online, y que hasta el día de hoy se han mantenido como grandes amigos y amigas. A 'Los Avengers', más que un grupo de amigos, mi círculo de confianza. Gracias por la constancia de compartir semana a semana, por las risas y por enseñarme que la vida es más que solo resultados académicos. Sin duda, su compañía permitió hacer de esta etapa universitaria una experiencia inolvidable.

Por último, a Carolina: mi compañera, mi socia y mi mejor amiga. Gra-

cias por estar en cada trabajo, en cada entrega y por todos los momentos que compartimos durante estos años. Cerrar este ciclo contigo no fue una coincidencia, fue el mejor final posible para esta etapa. Gracias por todo.

A todos ustedes, muchas gracias.

Capítulo 1

Introducción

1.1. Contexto

El aprendizaje culinario en entornos digitales presenta desafíos que trascienden lo meramente pedagógico, involucrando aspectos tecnológicos, de interacción y de infraestructura. Aunque en los últimos años han surgido múltiples aplicaciones móviles orientadas a la cocina, la mayoría de ellas se limita a ofrecer recetarios digitales o comunidades de intercambio de contenido gastronómico. Estas soluciones carecen de mecanismos de retroalimentación inmediata que permitan al usuario evaluar sus resultados y aprender de sus errores en tiempo real, además de no incorporar estructuras que fomenten la constancia o la progresión en el aprendizaje.

Frente a este escenario, se hace evidente la necesidad de desarrollar herramientas que integren tanto un modelo pedagógico innovador como una base tecnológica sólida. En este sentido, la propuesta de SHEFU busca responder a esta brecha mediante una arquitectura técnica que soporte la integración de componentes interactivos, algoritmos de visión por computador y elementos de gamificación, todo dentro de una infraestructura escalable y segura. De esta manera, la robustez del sistema no solo sustenta la experiencia del usuario, sino que también garantiza la continuidad del servicio, la protección de los datos personales y la posibilidad de expansión futura hacia nuevos contextos educativos y geográficos [1].

1.2. Problema a Resolver

El problema central que aborda este trabajo radica en cómo diseñar e implementar una infraestructura técnica que permita integrar, de manera coherente y eficiente, diversos componentes funcionales dentro de una única aplicación móvil. En particular, el sistema debe ser capaz de combinar guías interactivas orientadas al aprendizaje culinario, mecanismos de almacenamiento seguro de información de usuario, módulos de evaluación de imágenes basados en visión computacional y elementos de gamificación que promuevan la participación y la constancia.

El desafío técnico no solo consiste en lograr que estos módulos funcionen correctamente de forma individual, sino también en garantizar su interoperabilidad dentro de un entorno móvil con recursos limitados. Esto implica abordar aspectos como la optimización del rendimiento en dispositivos con diferentes capacidades de hardware, la gestión eficiente de la conectividad a internet y la protección de los datos personales conforme a las normativas vigentes. Asimismo, resulta esencial que la arquitectura propuesta sea escalable y adaptable, de modo que el sistema pueda incorporar nuevas funcionalidades o soportar un mayor volumen de usuarios sin comprometer la estabilidad o la experiencia de uso.

1.3. Acercamiento a la Solución

Para dar respuesta a las barreras identificadas, proponemos el diseño de SHEFU como un sistema integral de aprendizaje culinario. En su capa de presentación, la aplicación móvil guiará al usuario desde el registro de sus ingredientes y utensilios hasta la finalización de cada receta, ofreciendo descripciones claras de cada etapa y recopilando fotografías del progreso y del plato terminado.

A nivel tecnológico, SHEFU aprovecha API REST para facilitar la comunicación entre servicios independientes, integrando frameworks de desarrollo móvil multiplataforma y bibliotecas de visión como TensorFlow Lite. La arquitectura basada en API REST permite una interacción eficiente entre los distintos módulos del sistema, asegurando escalabilidad y flexibilidad. Además, el sistema se apoya en estudios de gamificación educativa y en tra-

bajos previos de visión artificial aplicada a la cocina.

El presente trabajo enfatiza la definición de la arquitectura técnica y la implementación de los módulos principales (autenticación, gestión de sesiones, persistencia de datos), asegurando que SHEFU pueda operar de forma estable y escalar a un mayor número de usuarios en el futuro [2].

1.4. Hipótesis

Una arquitectura técnica basada en Flutter, Firebase y TensorFlow Lite, estructurada de manera modular y escalable, permite implementar un sistema móvil robusto que integra de forma eficiente retroalimentación automática y gamificación [3]. Esta arquitectura facilita la gestión segura de usuarios, almacenamiento de datos y procesamiento de puntuaciones, contribuyendo a mejorar la experiencia, motivación y progresión de los usuarios en el aprendizaje culinario.

1.4.1. Objetivos Específicos del Proyecto

Los objetivos específicos definidos para el proyecto se resumen en la siguiente tabla:

N°	Objetivo Específico
1	Desarrollar la arquitectura general de la aplicación, integrando cliente móvil, procesamiento en la nube y persistencia de datos.
2	Implementar un flujo pedagógico con recetas guiadas, retroalimentación visual y elementos de gamificación.
3	Entrenar y ajustar un modelo de visión computacional capaz de evaluar imágenes de preparaciones culinarias.
4	Validar la usabilidad de la aplicación mediante pruebas con usuarios y métricas de retención y motivación.
5	Desarrollar la interfaz de usuario en Flutter, garantizando accesibilidad y consistencia visual.
6	Integrar Firebase para la gestión de autenticación, sesiones y almacenamiento en la nube.
7	Realizar pruebas unitarias y de integración para asegurar robustez en los módulos implementados.

Cuadro 1.1: Objetivos específicos del proyecto SHEFU.

Esta memoria se centra en los objetivos **1, 4, 6 y 7**, que corresponden a la definición de la arquitectura técnica, la integración con servicios en la nube y la validación de los módulos mediante pruebas unitarias e integración. Estos objetivos reflejan el aporte específico del memorista en el desarrollo de SHEFU desde una perspectiva técnica, orientada a garantizar la robustez, seguridad y escalabilidad de la solución.

1.5. Estructura de la Memoria

Este documento se organiza de la siguiente manera: el Capítulo 2 presenta el estado del arte, revisando soluciones existentes y herramientas utilizadas. El Capítulo 3 describe el desarrollo de la solución con énfasis en el diseño e

implementación de la arquitectura técnica. El Capítulo 4 aborda la validación de la hipótesis, objetivos y preguntas de investigación mediante pruebas y métricas. Finalmente, el Capítulo 5 expone las conclusiones obtenidas y el trabajo futuro proyectado.

Capítulo 2

Estado del Arte

2.1. Soluciones Existentes

El desarrollo de soluciones digitales en cocina ha avanzado en ámbitos como la personalización de recetas, la visión por computador aplicada a platos, y la gamificación del proceso de aprendizaje. Sin embargo, aún no existe una solución que combine de manera integrada estas dimensiones con enfoque educativo, como propone SHEFU.

En cuanto a experiencias de usuario gamificadas, destaca Zest, una aplicación que transforma el aprendizaje culinario en una progresión estructurada con niveles, desafíos y recompensas, reforzando la motivación intrínseca mediante elementos lúdicos [4]. Estudios sobre gamificación en educación han validado esta estrategia como efectiva para aumentar el compromiso del usuario y favorecer el aprendizaje continuo en entornos prácticos [3], [5].

En el área de personalización de contenidos culinarios mediante IA, SideChef ha incorporado algoritmos capaces de generar recetas desde fotografías, ajustando automáticamente los ingredientes a restricciones o preferencias del usuario [6]. Aplicaciones como Yummly e Innit han explorado esta línea desde el diseño de perfiles de sabor personalizados, conectividad con dispositivos inteligentes y sugerencias contextuales basadas en el historial culinario [7], [8]. Si bien son plataformas potentes en términos de adaptación, no abordan la enseñanza como proceso progresivo ni incluyen validación visual activa por parte del sistema.

Respecto a la asistencia inteligente y adaptativa, IBM ha registrado una patente que contempla la generación dinámica de recetas según los recursos del usuario y su historial, integrando sensores y una interfaz instructiva como soporte continuo durante la preparación [9]. June Life, por su parte, plantea un sistema de evaluación visual por comparación entre una imagen objetivo y la imagen real del alimento, lo cual valida la posibilidad de utilizar mecanismos visuales automatizados como forma de retroalimentación final, sin necesidad de intervenir durante el proceso [10].

La visión artificial aplicada al reconocimiento de ingredientes en imágenes finales ha sido explorada en investigaciones como MIRecipe [11], un dataset diseñado para entrenar modelos que identifican los componentes de un plato a partir de su apariencia terminada. Este tipo de recurso técnico permite estructurar algoritmos de retroalimentación visual en SHEFU sin depender de una supervisión paso a paso. De forma complementaria, herramientas como TensorFlow Lite permiten implementar dichos modelos directamente en el dispositivo móvil, habilitando análisis de imágenes sin requerir conexión a internet ni procesamiento en servidores externos [12].

Desde el punto de vista pedagógico, Marinescu et al. han abordado el diseño de sistemas de tutoría inteligente adaptados a tareas prácticas, donde el sistema ajusta sus sugerencias y contenidos en tiempo real según el desempeño observado [13]. Esta lógica de retroalimentación contextual y progresiva respalda la propuesta de un árbol de recetas personalizado que varíe según los resultados y el nivel del usuario. Además, estudios recientes han explorado el uso de visión por computador para evaluar directamente la calidad visual de platos, incluyendo la detección de errores comunes como quemado, sobrecocción o presentación desordenada, mediante modelos entrenados con fotos reales [14], [15].

A nivel técnico, frameworks como Flutter y Firebase permiten desarrollar aplicaciones móviles multiplataforma con soporte para procesamiento local de imágenes, integración de cámaras, notificaciones y visualización interactiva del progreso [2]. Estas tecnologías habilitan un entorno de desarrollo eficiente y escalable para construir la solución que SHEFU plantea, sin limitar la experiencia a un solo sistema operativo.

En conjunto, estos desarrollos muestran que existen elementos tecnológicos, pedagógicos y visuales que pueden ser combinados para construir un sistema de aprendizaje culinario adaptativo. SHEFU busca precisamente integrar estas dimensiones en una solución móvil orientada a guiar, motivar y evaluar al usuario en su progreso culinario desde su propio contexto.

2.2. Herramientas y Métodos (Marco Teórico)

Para el desarrollo de **SHEFU** se seleccionaron herramientas y metodologías que permiten construir una aplicación móvil multiplataforma con potencial de integrar inteligencia artificial y servicios en la nube, priorizando la modularidad, la escalabilidad y la facilidad de mantenimiento. En la capa de desarrollo frontend se utiliza el framework **Flutter**, que ofrece compatibilidad nativa con los sistemas operativos Android e iOS a partir de una única base de código. Este enfoque simplifica el ciclo de desarrollo, reduce los costos de mantenimiento y proporciona un rendimiento gráfico adecuado para dispositivos móviles, gracias a su arquitectura basada en widgets y su motor de renderizado eficiente.

Para la gestión de datos y la autenticación de usuarios se optó por **Firestore**, una plataforma en la nube que integra servicios de autenticación, almacenamiento y sincronización en tiempo real. Esta tecnología se seleccionó por su capacidad de proveer un backend administrado que elimina la necesidad de servidores propios, ofreciendo una infraestructura flexible y con escalabilidad automática según la demanda. Además, su estrecha integración con Flutter facilita la implementación de mecanismos de registro, inicio de sesión y manejo de información persistente sin comprometer la seguridad de los datos del usuario.

En el ámbito del procesamiento inteligente, se emplea **TensorFlow Lite** como framework de visión computacional optimizado para dispositivos móviles. Esta herramienta permite ejecutar modelos de aprendizaje profundo directamente en el dispositivo, reduciendo la dependencia de la conectividad a internet y contribuyendo a una menor latencia en la evaluación de imágenes [12].

Finalmente, la arquitectura técnica de SHEFU se estructura bajo principios de modularidad y separación de responsabilidades, asegurando una comunicación eficiente entre los distintos componentes y una base sólida para futuras extensiones funcionales.

2.3. Discusión

La selección de las herramientas y frameworks utilizados en el desarrollo de **SHEFU** responde a la necesidad de equilibrar rendimiento, facilidad de desarrollo y compatibilidad multiplataforma. En el ámbito del frontend, se evaluaron alternativas como **React Native** y **Kotlin Multiplatform**. Sin embargo, **Flutter** fue la opción preferida debido a su madurez, su capacidad para compilar de manera nativa y su consistencia visual entre sistemas operativos. A diferencia de otras tecnologías híbridas, Flutter ofrece un motor de renderizado propio que reduce la dependencia de componentes nativos, lo que resulta ventajoso en términos de rendimiento gráfico y predictibilidad del comportamiento de la interfaz.

Para la capa de backend y gestión de datos, se consideraron opciones como la implementación de un servidor propio en **Node.js** o **Django**, así como servicios en la nube de tipo **AWS** o **Google Cloud Platform**. No obstante, **Firestore** fue seleccionada por su integración directa con Flutter y su enfoque *Backend as a Service (BaaS)*, que permite reducir la complejidad operativa y los costos de mantenimiento. Su sistema de autenticación preconfigurado y su base de datos en tiempo real ofrecen un entorno seguro y escalable, ideal para un proyecto en fase de validación académica o prototípica.

En relación con el módulo de visión por computadora, se analizaron librerías como **OpenCV** y **PyTorch Mobile**. Inicialmente se consideró el modelo *MobileNetV3* por su balance entre velocidad y precisión. Sin embargo, durante la fase de implementación, se migró a **EfficientNet-Lite0**. Esta decisión se basó en la necesidad de mejorar el rendimiento y la precisión del modelo, obteniendo una arquitectura capaz de ejecutarse *on-device* (localmente) y proveer no solo un puntaje, sino criterios de evaluación detallados (flags), reduciendo la latencia y mejorando la privacidad del usuario.

En conjunto, las decisiones tecnológicas adoptadas se orientaron a asegurar un equilibrio entre robustez técnica y viabilidad de desarrollo. La combinación de Flutter, Firebase y TensorFlow Lite ofrece un ecosistema coherente y mantenible, capaz de sostener la propuesta pedagógica de SHEFU sin requerir una infraestructura compleja. Estas herramientas permiten concentrar los esfuerzos en el diseño de la experiencia de aprendizaje y en la validación funcional del sistema, sin comprometer aspectos críticos como seguridad, rendimiento o escalabilidad.

A diferencia de aplicaciones populares como Tasty o Kitchen Stories, que funcionan principalmente como repositorios de recetas, la contribución fundamental de SHEFU es posicionarse como una herramienta de aprendizaje activo. Su principal innovación radica en la integración sinérgica de un aprendizaje progresivo y personalizado con una retroalimentación inmediata y visual. El sistema crea un itinerario estructurado que adapta las recetas al nivel de habilidad y a los recursos del usuario, un enfoque que se complementa con su característica más distintiva: la evaluación automática del plato final mediante visión por computadora. Este mecanismo de feedback es clave para corregir errores y consolidar técnicas, algo que las plataformas existentes no ofrecen.

Para cohesionar la experiencia, todo el ciclo de aprendizaje está envuelto en una capa de gamificación. La incorporación de niveles, puntos e insignias transforma el proceso en una experiencia lúdica, fomentando la constancia y el desarrollo de habilidades a largo plazo. De esta manera, SHEFU no es solo una aplicación de recetas, sino un tutor culinario personal que guía, evalúa y motiva al usuario, llenando un vacío significativo en el ecosistema de aplicaciones de cocina.

Capítulo 3

Desarrollo de la Solución

El desarrollo de **SHEFU** se abordó a partir de un proceso iterativo que combinó fases de análisis, diseño, implementación y prueba, buscando construir una arquitectura técnica sólida y adaptable. Desde el punto de vista del memorista, el foco estuvo en el diseño y la implementación de la infraestructura móvil, la integración de los servicios en la nube y la integración del módulo de visión por computadora, garantizando la interoperabilidad entre los distintos componentes del sistema.

A continuación, se detallan las principales etapas técnicas que permitieron materializar la solución propuesta, desde la definición de los requerimientos iniciales hasta la implementación de los módulos funcionales y las pruebas de verificación correspondientes.

3.1. Análisis de Requerimientos

La definición de los requerimientos funcionales y no funcionales constituye la base del diseño técnico de **SHEFU**. En esta etapa se identificaron las funcionalidades esenciales del sistema y las condiciones bajo las cuales deben operar los distintos componentes. El proceso de levantamiento de requerimientos se realizó considerando tanto la experiencia esperada del usuario como las limitaciones técnicas propias del entorno móvil, priorizando la escalabilidad, la eficiencia y la seguridad.

3.1.1. Requisitos Funcionales

Código	Descripción del Requisito Funcional
RF01	El sistema debe permitir a nuevos usuarios crear una cuenta de forma segura.
RF02	El sistema debe permitir a usuarios existentes iniciar y cerrar sesión.
RF03	El sistema debe permitir al usuario crear y personalizar su perfil con datos clave para la experiencia culinaria.
RF04	El sistema debe permitir al usuario actualizar la información de su perfil.
RF05	El sistema debe presentar un quiz inicial para determinar el nivel de conocimiento culinario del usuario.
RF06	El sistema debe calcular y asignar un nivel inicial (Básico, Intermedio, Avanzado) basado en el quiz.
RF07	El sistema debe recalcular y actualizar el nivel del usuario según su desempeño acumulado y scores finales.
RF08	El sistema debe generar y presentar recetas adaptadas al nivel, ingredientes, utensilios, tiempo y preferencias del usuario.
RF09	El sistema debe permitir al usuario filtrar recetas por categorías y otros criterios.
RF10	El sistema debe generar recomendaciones automáticas para sustituir ingredientes o utensilios.
RF11	El sistema debe sugerir recetas o desafíos semanales personalizados al usuario.
RF12	El sistema debe guiar al usuario paso a paso en la ejecución de una receta.
RF13	El sistema debe incluir imágenes y/o opciones de audio/video en cada paso de la receta.
RF14	El sistema debe permitir al usuario controlar el progreso (iniciar, avanzar, retroceder, finalizar) de la sesión de cocina.
RF15	El sistema debe permitir al usuario subir imágenes de su progreso y/o plato final para evaluación.

RF16	El sistema debe analizar automáticamente las imágenes para evaluar la presentación y el punto de cocción del plato.
RF17	El sistema debe proporcionar feedback inmediato al usuario basado en el análisis de las imágenes.
RF18	El sistema debe lanzar un breve cuestionario al final de la sesión de cocina para reforzar conceptos.
RF19	El sistema debe integrar resultados de Visión Computacional y Quiz Final para calcular un score de desempeño.
RF20	El sistema debe asignar puntos al usuario por completar recetas y buen desempeño.
RF21	El sistema debe otorgar insignias por el cumplimiento de logros específicos.
RF22	El sistema debe gestionar el progreso del usuario a través de niveles gamificados.
RF23	El sistema debe permitir la participación en retos colaborativos.
RF24	El sistema debe generar y mostrar un historial visual del progreso, puntos, insignias y evolución del usuario.

Cuadro 3.1: Requisitos Funcionales del sistema SHEFU

3.1.2. Requisitos No Funcionales

Código	Descripción del Requisito No Funcional
RNF1	El sistema debe ser capaz de funcionar correctamente en dispositivos móviles con sistemas operativos Android e iOS.
RNF2	El sistema debe generar respuestas (como recomendaciones o retroalimentación) en un tiempo menor a 5 segundos para no afectar la experiencia del usuario.
RNF3	El sistema debe ser escalable para soportar un número creciente de usuarios y recetas sin perder rendimiento.
RNF4	El sistema debe garantizar la privacidad de los datos del usuario, incluyendo sus imágenes, perfil alimentario y actividad.

RNF5	El sistema debe implementar mecanismos de autenticación y autorización seguros, incluyendo opciones de login social.
RNF6	El sistema debe contar con una interfaz intuitiva, accesible a usuarios con poca experiencia digital o de cocina.
RNF7	El sistema debe incluir documentación y tutoriales dentro de la app para apoyar a nuevos usuarios en el uso de sus funcionalidades.
RNF8	El código del sistema debe seguir buenas prácticas de programación, empleando una estructura modular, control de versiones y comentarios claros.
RNF9	El sistema debe ser compatible con estándares actuales de desarrollo móvil, permitiendo su despliegue en tiendas de aplicaciones oficiales.
RNF10	El sistema debe utilizar modelos de IA optimizados para dispositivos móviles para minimizar el uso de recursos y consumo de batería.

Cuadro 3.2: Requisitos no funcionales del sistema SHEFU

En conjunto, estos requerimientos definen tanto el alcance funcional del sistema como las condiciones técnicas necesarias para su correcto desempeño. La contribución de esta memoria se centra en la arquitectura de backend, que funciona como la columna vertebral para todo el sistema SHEFU.

En este contexto, se desarrollaron directamente los componentes de backend responsables de la gestión de usuarios y perfiles (RF01-RF04), la persistencia del historial (RF24) y la implementación de la infraestructura de seguridad (RNF4, RNF5). Asimismo, el diseño de la arquitectura modular y el uso de servicios gestionados como Firebase permitieron cumplir con requerimientos no funcionales clave como la escalabilidad (RNF3) y las buenas prácticas de desarrollo (RNF8, RNF9).

Adicionalmente, se integró la infraestructura necesaria para habilitar las funcionalidades más complejas de la aplicación. Esto incluyó la creación de

los servicios para almacenar y gestionar los resultados del módulo de visión computacional (RF15-RF17), y los puntajes que alimentan tanto el cálculo del puntaje final por sesión como el sistema de gamificación (RF18-RF22).

De esta manera, aunque el foco fue la capa de servicios y datos, la contribución del backend es transversal, proveyendo el soporte fundamental sobre el cual operan todos los requerimientos funcionales y no funcionales del proyecto, garantizando una experiencia de usuario coherente, segura y robusta.

3.2. Diseño

El sistema SHEFU sigue un modelo cliente-servidor, donde la aplicación móvil funciona como interfaz de usuario, mientras que la lógica de negocio, almacenamiento y procesamiento de datos se gestiona en la capa de backend, que constituye la contribución principal del memorista.

3.2.1. Definición de módulos

La arquitectura de backend de SHEFU se organiza en distintos módulos, cada uno responsable de funcionalidades específicas que aseguran el correcto funcionamiento del sistema y el cumplimiento de los requerimientos funcionales y no funcionales.

Gestión de Usuarios y Autenticación (GUA): Este módulo gestiona el registro, inicio de sesión, recuperación de contraseñas y cierre de sesión de los usuarios. Incluye:

- Registro seguro y validación de datos de usuario.
- Autenticación mediante Firebase Authentication, incluyendo login social.
- Control de sesiones activas y revocación de accesos.

Requerimientos cubiertos: RF01–RF02 y RNF5.

Persistencia de Datos (PD): Responsable de almacenar toda la información relacionada con el usuario, recetas, puntajes, logros y progreso histórico. Subfunciones:

- Organización de datos en colecciones y subcolecciones de Firestore.
- Sincronización en tiempo real entre dispositivos y la nube.
- Gestión de backups y redundancia para integridad de datos.
- Consultas eficientes y control de índices para mejorar rendimiento.

Requerimientos cubiertos: RF03–RF04, RF15–RF19 y RNF1–RNF4.

Procesamiento y Evaluación de Imágenes (PEI): Este módulo se encarga de analizar la foto final del plato, tomada por el usuario, para proveer retroalimentación visual. A diferencia del plan inicial, este módulo no retorna un puntaje directo, sino que ejecuta un modelo **EfficientNet-Lite0** *on-device*. Este modelo evalúa la imagen y retorna un conjunto de cuatro criterios binarios (flags): cocción inadecuada, falta de ingrediente, desorden en la presentación y buen balance visual. Estos flags son la base para el cálculo del puntaje en el módulo GPG [12].

- Integración del modelo TFLite en la app Flutter.
- Preprocesamiento de la imagen capturada para adaptarla al modelo.
- Retorno de los 4 flags binarios y sus probabilidades asociadas.

Requerimientos cubiertos: RF16–RF17, RF19 y RNF10.

Gestión de Puntajes y Gamificación (GPG): Calcula el score final del usuario y gestiona los elementos de gamificación. Subfunciones:

- Cálculo del `finalScore` por receta, basado en la nueva fórmula que integra los flags de PEI.
- Asignación de “estrellas” (moneda de la app) por recetas completadas.
- Gestión de la Tienda y el inventario de accesorios del usuario.
- Actualización del Ranking de usuarios basado en el puntaje acumulado.

Requerimientos cubiertos: RF19–RF24.

Seguridad y Escalabilidad (SS): Se encarga de garantizar un sistema seguro y capaz de manejar crecimiento de usuarios y datos. Subfunciones:

- Validación de datos entrantes y control de accesos.
- Cifrado de información sensible y protección de endpoints.
- Monitoreo de rendimiento y optimización de consultas.
- Diseño modular para facilitar escalabilidad horizontal y vertical.

Requerimientos cubiertos: RNF3, RNF4, RNF8 y RNF9.

3.2.2. Interacciones entre módulos y tecnologías

Los módulos se comunican mediante APIs internas y servicios en la nube, asegurando un flujo de información consistente:

- **GUA** envía información de autenticación a **PD** y recibe confirmaciones de validación desde **SS**.
- **PEI** procesa imágenes localmente en el dispositivo mediante TensorFlow Lite y almacena los scores en **PD**.
- **GPG** consulta **PD** para calcular puntajes, gestionar logros y actualizar niveles.
- Todos los módulos interactúan con Firebase para persistencia, sincronización en tiempo real y gestión de usuarios.

3.2.3. Esquema General de la Arquitectura

El diagrama general del sistema se puede representar de la siguiente manera:

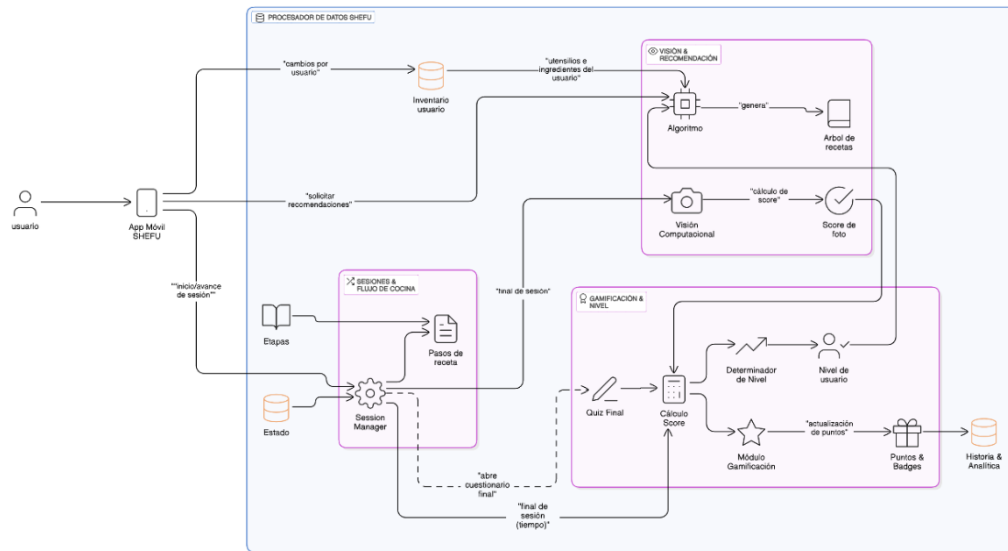


Figura 3.1: Esquema general del sistema.

En este esquema, la capa de backend se encarga de garantizar que los requerimientos funcionales y no funcionales relacionados con autenticación, almacenamiento, seguridad y procesamiento de datos se cumplan de manera eficiente y escalable.

3.2.4. Modelo de Datos

Para dar soporte a la arquitectura de servicios y a los requerimientos funcionales de SHEFU, se diseñó un modelo de datos NoSQL implementado sobre Cloud Firestore. La elección de esta tecnología responde a la necesidad de contar con una estructura flexible y escalable, capaz de adaptarse a la naturaleza dinámica de los datos de la aplicación, como el progreso del usuario, su inventario y su historial de actividades.

A diferencia de los modelos relacionales, un esquema basado en colecciones y documentos permite representar de manera más natural las relaciones jerárquicas y anidadas de la información, como las recetas completadas por un usuario o las insignias obtenidas. Esta estructura no solo simplifica las consultas y optimiza los tiempos de respuesta en un entorno móvil, sino

que también facilita la evolución del sistema, permitiendo incorporar nuevos campos o entidades sin requerir migraciones complejas. A continuación, se describen las colecciones principales que conforman la base de datos de SHEFU.

Colección usuarios

Esta colección principal almacena toda la información del perfil de cada persona registrada en la aplicación. Cada documento es identificado de forma única por el `userId` proporcionado por Firebase Authentication, lo que garantiza una vinculación segura entre la autenticación y los datos del perfil. Los campos de cada documento de usuario incluyen:

- **nombres:** Nombres del usuario.
- **apellidos:** Apellidos del usuario.
- **correo:** Correo electrónico asociado a la cuenta.
- **edad:** Edad del usuario.
- **genero:** Género del usuario.
- **nivelActual:** Categoría de habilidad actual (ej. "Básico", "Intermedio", "Avanzado").
- **numRecetasRealizadas:** Cantidad de recetas completadas por el usuario.
- **puntajeActual:** Suma del puntaje obtenido por el usuario.
- **resultadoQuizInicial:** Puntaje o nivel obtenido en el quiz inicial.
- **subNivel:** Indica el sub-nivel correspondiente (ej. "B1 a B2").
- **progresoSiguienteNivel:** Porcentaje faltante para alcanzar el próximo sub-nivel.
- **admin:** Booleano para permisos de administrador.
- **inventario:** Array que almacena los objetos comprados en la tienda.

Colección recetas

Contiene el catálogo completo de preparaciones culinarias disponibles en SHEFU. Inicialmente poblada con 110 recetas, esta colección permite escalar y añadir nuevo contenido. Cada documento representa una única receta:

- **nombre:** Nombre de la receta (ej. "Leche Asada").
- **categoria:** Tipo de plato (ej. "Postres y Dulces").
- **dificultad:** Nivel de habilidad requerido.
- **ingredientes:** Un array de strings con los ingredientes.
- **numIngredientes:** Cantidad de ingredientes de la receta.
- **pasosLista:** Un Array de strings con los pasos.
- **numPasos:** Cantidad de pasos para la receta.
- **tiempoPreparacion:** Tiempo estimado en minutos.
- **puntosOtorgados:** Puntos base otorgados por completar la receta.
- **vision:** Booleano que indica si la receta tiene evaluación por VC habilitada.

Subcolección historial

Para mantener un registro escalable y seguro de las actividades de cada usuario, se optó por una subcolección anidada dentro de cada documento de **usuarios**. Este diseño optimiza las consultas y simplifica las reglas de seguridad. La ruta es **usuarios/userId/historial/sesionId**. Cada documento representa una receta completada y almacena:

- **receta:** Nombre de la receta completada.
- **categoria:** Categoría de la receta.
- **fecha:** Timestamp de cuándo se completó la sesión.
- **tiempo:** Tiempo total que tardó el usuario en la sesión.
- **correo:** Correo del usuario.
- **puntajeFinal:** El **finalScore** obtenido en esa sesión.

Colección Quiz_Inicial

Almacena las preguntas y resultados del quiz de perfilamiento, vinculado al `id_usuario`. Cada documento representa una pregunta o el resultado del usuario.

- `usuario`: Referencia al usuario.
- `puntajeObtenido`: Puntaje total del quiz.
- `nivelObtenido`: Nivel asignado post-quiz.
- `preguntas`: Array con las preguntas y respuestas del usuario.

Colección Quiz_Final

Esta colección gestiona los cuestionarios de refuerzo pedagógico que se presentan al finalizar una receta. Cada documento actúa como un contenedor para el quiz de una preparación específica, agrupando sus preguntas en una subcolección.

- `idReceta`: La referencia a la colección `recetas`.

Subcolección Preguntas

Anidada dentro de cada documento de `Quiz_Final`, esta subcolección contiene las preguntas individuales:

- `pregunta`: El texto de la pregunta a mostrar.
- `opciones`: Un array de strings con las posibles respuestas.
- `respuestaCorrecta`: El string de la opción correcta.

Colección Tienda

Nueva colección para el módulo de gamificación (GPG). Almacena los ítems (accesorios de perfil, "Shefcitos") que los usuarios pueden comprar con la moneda de la app. Cada documento es un ítem:

- `nombre`: Nombre del ítem.
- `descripcion`: Texto descriptivo del ítem.

- **imagenURL**: Ruta de la imagen del ítem (ej. `images/Shefcitos/tiburon.png`).
- **precio**: Costo del ítem en la moneda del juego ('estrellas').
- **tipo**: Categoría del ítem (ej. "Imagen de perfil").

Colección Evaluaciones

Esta colección de soporte es fundamental para la retroalimentación y el reentrenamiento del modelo de Visión Computacional (PEI). Los administradores (chefs) la utilizan para evaluar y etiquetar manualmente las imágenes subidas por los usuarios, generando así nuevos datos de entrenamiento.

- **userId**: Identificador del usuario que subió la foto.
- **pathLower**: Ruta de almacenamiento de la imagen.
- **timestamp**: Fecha y hora de la evaluación.
- **comentario**: Comentario cualitativo del evaluador.
- **nota**: Nota general asignada por el evaluador (1-10).
- **buenBalance**: Flag binario (0 o 1).
- **coccionInadecuada**: Flag binario (0 o 1).
- **desordenPresentacion**: Flag binario (0 o 1).
- **faltaIngrediente**: Flag binario (0 o 1).
- **textura**: Flag binario (0 o 1).

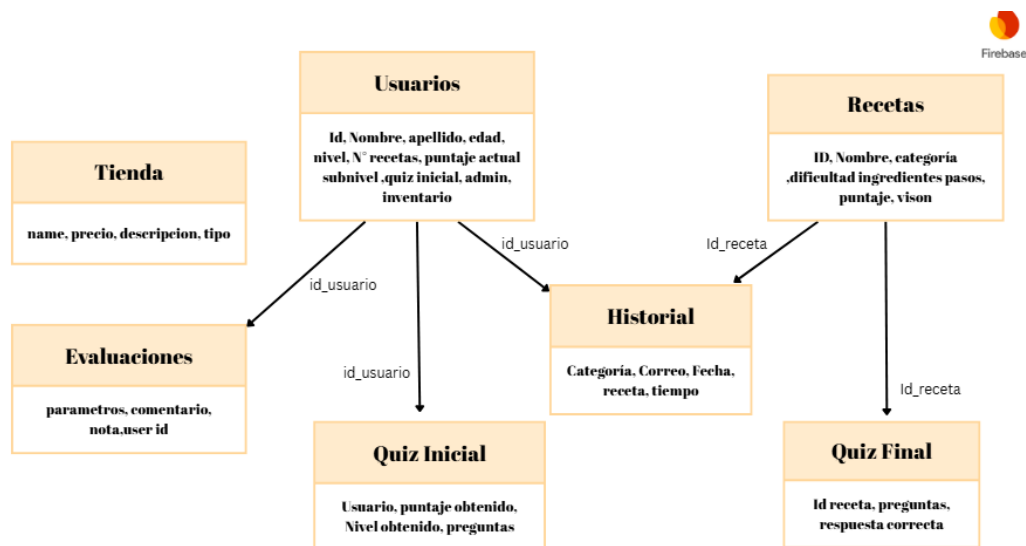


Figura 3.2: Diagrama de Base de Datos

3.3. Implementación y Pruebas Unitarias

Una vez definidos el diseño arquitectónico y el modelo de datos, se procedió con la implementación técnica de la solución. Esta fase se centró en traducir los requerimientos del backend en código funcional, configurar los servicios en la nube y verificar la correcta operación de cada componente. El desarrollo se realizó siguiendo un enfoque modular, lo que facilitó la integración y la ejecución de pruebas unitarias para asegurar la robustez del sistema.

3.3.1. Configuración del Entorno y Tecnologías

El desarrollo de SHEFU se unificó bajo un ecosistema tecnológico coherente para garantizar la compatibilidad y eficiencia entre el cliente y el servidor. El entorno de desarrollo se configuró con las siguientes herramientas principales:

- **Framework y Lenguaje:** La aplicación móvil se desarrolló con el framework **Flutter** (versión 3.32.5) y el lenguaje **Dart** (SDK 3.8.1),

permitiendo una base de código única para los sistemas operativos Android e iOS.

- **Servicios de Backend (BaaS):** Se utilizó **Firebase** como plataforma de administración de backend. Se integraron los SDKs de **firebase_core** para la conexión, **firebase_auth** para la gestión de autenticación de usuarios y **cloud_firestore** para la persistencia de datos NoSQL.
- **Entorno de Desarrollo (IDE):** El desarrollo y la depuración del código se llevaron a cabo utilizando **Visual Studio Code**, complementado con las extensiones de Flutter y Dart.
- **Control de Versiones:** Se empleó **Git**, con un repositorio centralizado en **GitHub**, para gestionar el control de versiones, el trabajo colaborativo y la integración de los distintos módulos.

3.3.2. Implementación de Módulos

La contribución de esta memoria se enfocó en la implementación de la lógica de negocio y los servicios de datos del backend. Esto implicó la creación de una capa de abstracción en Dart (capa de servicios) para gestionar la comunicación con Firebase, asegurando que el resto de la aplicación (la interfaz de usuario) interactúe con el backend de forma segura y estandarizada.

A continuación, se detallan los principales endpoints (funciones de servicio) implementados para cada módulo de la arquitectura:

Módulo GUA (Gestión de Usuarios y Autenticación) Este módulo implementa el ciclo de vida completo del usuario. Gestiona la autenticación (registro, login, logout) y el perfil de datos, incluyendo la lógica para el Quiz Inicial y la actualización de niveles. Los endpoints implementados son:

- **createUser(UserModel userModel, String password):** Crea la cuenta en Auth y guarda el perfil en Firestore.
- **signInUser(String email, String password):** Autentica a un usuario existente.
- **getUsers():** Obtiene un flujo en tiempo real (Stream) de todos los usuarios.

- `updateUser(String userId, Map<...> newData)`: Actualiza un documento de usuario específico en Firestore.
- `deleteUser(String userId)`: Elimina un documento de usuario de Firestore.
- `getCurrentUserLevel()`: Obtiene el nivel de habilidad del usuario actualmente logueado.
- `getUsersList()`: Obtiene una lista de una sola vez de todos los usuarios.
- `getUsersByLevel(String? nivel)`: Obtiene una lista de usuarios filtrada por un nivel específico.
- `getActualPointsUser()`: Obtiene el puntaje actual del usuario logueado.
- `getIfUserIsAdmin()`: Verifica si el usuario logueado tiene permisos de administrador.

Módulo PD (Persistencia de Datos) Este módulo implementa la lógica de persistencia de datos (CRUD) y las consultas de negocio para la colección `recetas`. Expone los siguientes endpoints:

- `addRecipe(Map<String, dynamic> recipeData)`: Crea un nuevo documento de receta en Firestore.
- `getRecipes()`: Obtiene un flujo en tiempo real (Stream) de todas las recetas.
- `getRecipeByName(String recipeName)`: Busca y obtiene una receta específica por su nombre.
- `updateRecipe(String recipeId, Map<...> newData)`: Actualiza una receta existente usando su ID de documento.
- `deleteRecipe(String recipeId)`: Elimina una receta de la base de datos usando su ID.
- `loadRecipeData(String recipeName)`: Carga los datos de una receta específica para su visualización.
- `findRecipeImage(String recipeName)`: Busca la ruta de la imagen de la receta en los assets locales de la app.
- `getRecipesByLevel(String nivel)`: Obtiene una lista de recetas filtradas por nivel de dificultad.

- `getAllRecipes()`: Obtiene una lista de todas las recetas (una sola vez, no stream).
- `getRecipesForUserLevel(String userLevel)`: Aplica la lógica de negocio para obtener las recetas adecuadas al nivel del usuario (ej. "Intermedio" ve "Básico." "Intermedio").

Módulo PEI (Procesamiento de Imágenes) Este módulo implementa la pipeline completa de visión computacional. Incluye la orquestación de la captura de la cámara, el almacenamiento de imágenes en la nube, la inferencia del modelo *on-device* y la lógica de la herramienta de evaluación para administradores.

1. Endpoints de VisionService:

- `analyzeImage(String path)`: Ejecuta la pipeline completa (detección y regresión) una sola vez para obtener un puntaje.
- `analyzeImageWithConfidence(String path, {int runs, ...})`: Ejecuta la pipeline múltiples veces (TTA) con ruido para calcular un puntaje y un nivel de confianza.
- `analyzeImageWithConfidenceIsolate(Map<...> params)`: Ejecuta el análisis de confianza en un Isolate (hilo) separado para no bloquear la interfaz de usuario (UI).

2. Endpoints de CameraService : Este servicio actúa como un puente entre el hardware del dispositivo (cámara), el almacenamiento en la nube (Dropbox) y el `VisionService`.

- `initializeDropbox()`: Inicializa la conexión con la API de Dropbox usando las credenciales.
- `uploadImageToDropbox(String imagePath)`: Sube un archivo de imagen desde una ruta local a la nube de Dropbox.
- `takePictureAndAnalyze(CameraController controller)`: Orquesta el flujo completo: 1. Toma la foto. 2. La guarda temporalmente. 3. La sube a Dropbox y 4. Llama a `VisionService` para obtener la evaluación.
- `loadDropboxImages()`: Consulta la API de Dropbox para listar todas las imágenes subidas por los usuarios y cruza esta información con la colección `Evaluaciones` de Firestore para determinar cuáles están pendientes de revisión.

- `_saveEvaluation(pathLower, flags, nota, ...)`: Guarda o actualiza un documento en la colección **Evaluaciones** con los flags y nota asignados por el administrador, completando el ciclo de re-entrenamiento.

Módulo GPG (Gestión de Puntajes y Gamificación) Este módulo implementa la lógica de negocio para la gamificación. Orquesta el cálculo del puntaje final de la sesión, gestiona la economía de la Tienda (compras y creación de ítems), administra el contenido del Quiz Final y actualiza el progreso y nivel del usuario.

- `_finishQuiz(userId, startTime, visionScore, ...)`: Orquesta el fin de la sesión, aplicando la fórmula de `combinedScore` (lógica GPG), y guarda el resultado final en la colección **Historial** (módulo PD).
- `_fetchPurchasedChefs()`: Obtiene la lista de IDs de los 'Shefcitos' comprados por el usuario desde su documento.
- `_pointsRemaining()`: Calcula los puntos faltantes para el siguiente nivel.
- `_submitQuestion(recipeId, questionData, ...)`: Crea o actualiza el documento de un quiz en **QuizFinal** y añade una nueva pregunta a su subcolección **Preguntas**.
- `saveQuizResult(score, level, subLevel)`: Orquesta el guardado del Quiz Inicial, llamando a `UserService.updateUser` para actualizar el nivel, subnivel y puntaje del usuario en Firestore.
- `updateQuizInicial(quizinicial)`: Llama a `UserService.updateUser` para marcar el Quiz Inicial como completado en el documento del usuario.
- `_updateUserScoreAndLevel(userId, ...)`: Ejecuta una transacción de Firestore para actualizar el perfil, recalculando el Puntaje Actual, Nivel actual y subNivel del usuario.
- `_handlePurchase(chefDocId, precio)`: Verifica los puntos del usuario y, si son suficientes, ejecuta una actualización en Firestore para restar los puntos y añadir el ítem al array `productosComprados` del usuario.
- `createShef(ShefModel shefModel)`: Endpoint de administrador para añadir un nuevo ítem a la colección **Tienda** en Firestore.

3.3.3. Pruebas Unitarias y de Integración

Para validar la robustez y el correcto funcionamiento de la arquitectura de backend, se siguió el plan de testing definido en la Tabla 3.4. Las pruebas se enfocaron en verificar los componentes de lógica de negocio y la interacción con los servicios en la nube, utilizando tanto pruebas unitarias para funciones aisladas como pruebas de integración para los flujos completos. Cada prueba se diseñó con criterios de aceptación claros para definir su éxito o fracaso.

ID	Módulo	Objetivo	Resultado esperado	Estado
T-01	GUA	Iniciar sesión	Redirige al Menú	Éxito
T-02	GUA	Manejo de error	Mensaje de error claro	Éxito
T-03	PD	Cargar recetas	Detalle renderizado	Éxito
T-04	PD	Paso a paso	Secuencia sin bloqueos	Éxito
T-05	GPG	Cuestionario	Puntaje mostrado	Éxito
T-06	PD	Editar datos	Persistencia correcta	Éxito
T-07	PEI	Inferencia Modelo	Tiempo de inferencia < 3s	Éxito
T-08	PD	Firestore CRUD	Documento creado	Éxito
T-09	GUA	Inventario	Auto-check en receta	Éxito
T-10	GPG	Compra Tienda	Ítem en inventario	Éxito

Cuadro 3.4: Resumen de ejecución del Plan de Testing.

Pruebas de Autenticación (T-01, T-02): Se realizaron pruebas para el servicio de autenticación (GUA). El criterio de aceptación para T-01 (inicio de sesión exitoso) fue que la función de `firebase_auth` retornara un objeto `User` no nulo. Para la prueba de manejo de error (T-02), la aserción de éxito fue que el SDK lanzara una excepción `FirebaseAuthException` específica (ej. `wrong-password`), validando así que la lógica de la app gestiona correctamente las fallas.

Pruebas de Base de Datos (T-03, T-08): Se implementaron pruebas de integración clave para el módulo de persistencia (PD). Para T-03 (lectura de recetas), la prueba se consideró exitosa si la consulta a la colección `recetas` devolvía una lista no vacía. Para el ciclo CRUD (T-08), la prueba se validó creando un documento, luego ejecutando una consulta de lectura (`get`) y **aseverando** que los datos leídos coincidían exactamente con los datos escritos.

Pruebas de Gestión de Inventario (T-10): Se validó la nueva funcionalidad de personalización de ingredientes. La prueba consistió en agregar una lista de ingredientes al perfil del usuario (GUA) y luego cargar una receta que contuviera dichos elementos. El criterio de éxito fue verificar que la función de lógica de negocio marcara automáticamente (*auto-check*) los ingredientes coincidentes en la interfaz de la receta, facilitando la experiencia del usuario.

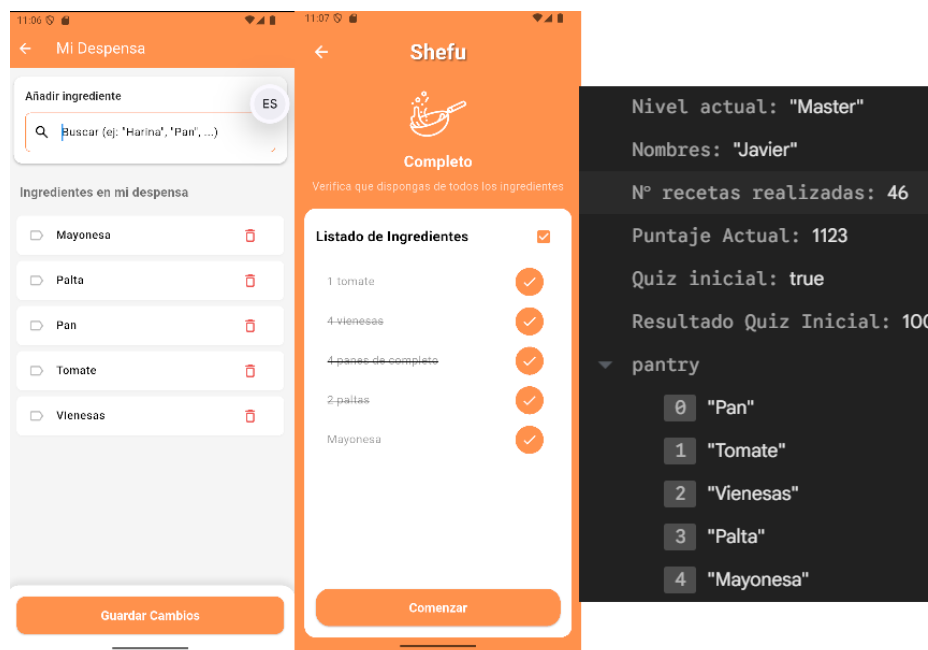


Figura 3.3: Evidencia T-11: Actualización en Firestore de array de ingredientes y correcto auto marcaje en recetas.

Pruebas de Compra en Tienda (T-09): Se validó la lógica transaccional

crítica del módulo de gamificación (GPG). La prueba simuló la compra de un accesorio ("Shefcito") por parte de un usuario con saldo suficiente. El criterio de éxito fue verificar que, tras la ejecución, se descontaran los puntos correspondientes del perfil del usuario y que el ID del ítem comprado se añadiera correctamente al array `productosComprados` (inventario) en Firestore.

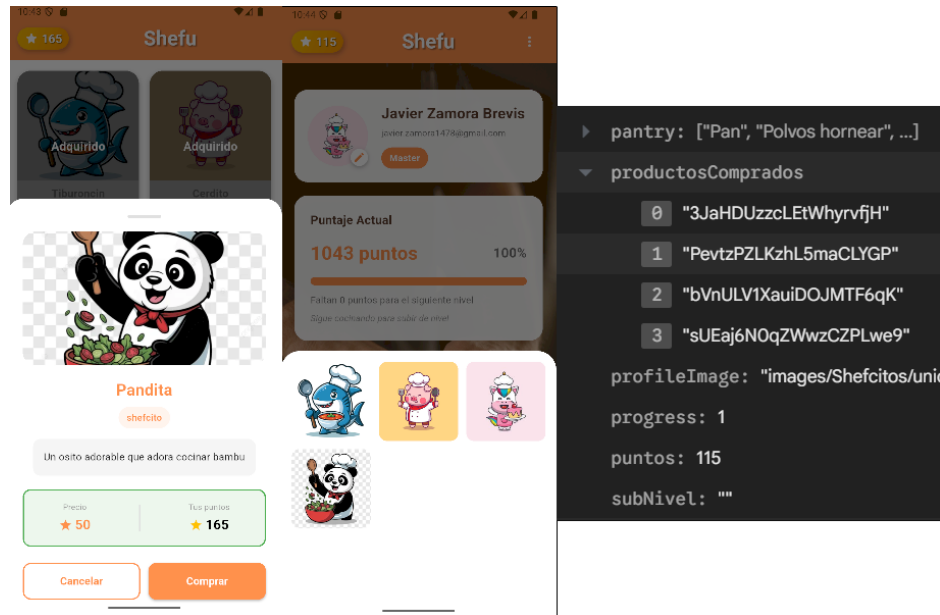


Figura 3.4: Evidencia T-12: Actualización en Firestore de puntos y array de productos tras una compra.

Prueba del Flujo de Sesión (T-04, T-05, T-07): Se ejecutó una prueba de integración **extremo a extremo (E2E)** simulando un ciclo de juego completo. Esta prueba llamaba a la función de inicio de sesión (GUA), seleccionaba una receta (PD, T-04), y simulaba la finalización de la misma. Se enviaba una imagen de prueba al **VisionService** (PEI), el cual ejecutaba el modelo **EfficientNet** real *on-device*. La prueba se consideró **exitosa** si el **Score final** se calculaba correctamente usando la fórmula de ponderación, el registro persistía en **historial**, y el **puntajeActual** del usuario se actualizaba en Firestore.

Verificación del Módulo de Visión Computacional (PEI): A diferencia de las pruebas de lógica de negocio, la verificación del módulo PEI se

centró en sus métricas de **integración y rendimiento no funcional** [12]. Si bien la calidad del modelo (medida con métricas como $AUC \geq 0.80$ o $F1\text{-Score} \geq 0.70$ según el informe técnico) fue validada por el equipo de VC, la responsabilidad de esta arquitectura fue asegurar su correcta ejecución *on-device*. Por lo tanto, el criterio de aceptación principal fue la **latencia** (T-09, RNF2, RNF10), midiéndose que el tiempo de inferencia (desde que se envía la foto hasta que se reciben los flags) se mantuviera por debajo del objetivo de 3s en dispositivos de gama media.

```
I/flutter (13084): -----  
I/flutter (13084): Tiempo Total de Inferencia (x8 corridas): 2675 ms  
I/flutter (13084): Promedio por corrida: 334.38 ms  
I/flutter (13084): -----
```

Figura 3.5: Captura de logs de ejecución del módulo PEI. Se destaca el tiempo de inferencia *on-device*.

Capítulo 4

Validación

Este capítulo detalla el proceso y los resultados de la validación del sistema SHEFU con usuarios finales. El objetivo de esta fase fue validar la hipótesis central de la memoria: que la arquitectura implementada es capaz de soportar una aplicación robusta que integra eficientemente la retroalimentación automática y la gamificación, mejorando la experiencia y motivación del usuario.

4.1. Metodología de Validación

La metodología de validación se centró en una **evaluación holística** del flujo completo de la aplicación. Se adoptó un enfoque de **pruebas de usabilidad moderadas**, donde un facilitador guió a los participantes a través de un guión predefinido.

- **Objetivo:** Medir la usabilidad general del sistema integrado, la comprensión del feedback de VC por parte del usuario y el impacto percibido por el módulo de gamificación en la motivación [4].
- **Participantes:** Se reclutó una muestra de **8 usuarios**, cuyas edades se encontraban en el rango objetivo de 12 a 60 años. La muestra incluyó estudiantes, jóvenes profesionales y adultos con diversos niveles de familiaridad tecnológica y culinaria.
- **Entorno:** Las pruebas se realizaron de forma híbrida (presencial y remota), utilizando el prototipo funcional completo de la aplicación

instalado en dispositivos de gama media (Android) para verificar el rendimiento *on-device*.

- **Recolección de Datos:** Se utilizaron tareas guiadas (ver sección 4.2), el cuestionario **System Usability Scale (SUS)**, una encuesta de **Net Promoter Score (NPS)** y una entrevista semi-estructurada final.

4.2. Tareas y Métricas de Evaluación

A cada participante se le solicitó realizar un conjunto de tareas que simulaban el «embudo funcional» completo de la aplicación:

- **Tarea 1 (Onboarding):** Registrarse y completar el Quiz Inicial.
- **Tarea 2 (Flujo de Cocina):** Seleccionar una receta, seguir la guía paso a paso, tomar la foto para la evaluación visual y completar el Quiz Final.
- **Tarea 3 (Resultados y Feedback):** Revisar la pantalla de resultados, interpretar el puntaje obtenido y el feedback visual.
- **Tarea 4 (Gamificación):** Revisar su nuevo puntaje en el Ranking, navegar a la Tienda y realizar una compra de accesorio («Shefcito») con los puntos ganados.

Para evaluar el éxito de estas tareas, se utilizaron los siguientes instrumentos:

Instrumentos Cuantitativos

- **Usabilidad (SUS):** Cuestionario estándar de 10 preguntas. Criterio de aceptación: Puntaje ≥ 75 .
- **Lealtad (NPS):** «¿Qué tan probable es que recomiendes SHEFU a un amigo?» (Escala 0-10).

Instrumentos Cualitativos

Se profundizó en la experiencia con las siguientes preguntas clave:

■ Sobre Visión Computacional:

- «¿Entiendes por qué obtuviste esta nota?»
- «¿Que te pareció el tiempo que se demoró la aplicación en analizar la imagen subida?»

■ Sobre Gamificación:

- «¿Te parecieron justos los precios de la tienda respecto a lo que puntos que ganaste cocinando?»
- «¿Te motiva el ver tu nombre en el ranking?»
- «¿Te motiva la posibilidad de comprar accesorios cosméticos en la tienda con los puntos ganados?»

■ Sobre Experiencia General:

- «¿Hubo algún momento en que la aplicación se sintiera lenta o fallara? Si es así, ¿En que momentos?»
- «¿Que aspectos destacarías o mejorarías de la aplicación?»

4.3. Resultados Obtenidos

La ejecución de las pruebas con los 8 participantes arrojó resultados positivos que respaldan la estabilidad de la arquitectura. A continuación se detalla el perfil de los usuarios y sus métricas asociadas.

Usuario	Edad	Ocupación	SUS Score	NPS (0-10)
U1	13	Estudiante	97.5	10
U2	19	Universitario	90	9
U3	24	Ingeniero	90	9
U4	28	Ingeniero	72.5	8
U5	35	Chef Amateur	97.5	10
U6	42	Administrativo	82.5	8
U7	49	Dueña de casa	70.0	7
U8	55	Profesor	80.0	8
Promedio	33.1	-	85	8.6 (Excelente)

Cuadro 4.1: Resultados cuantitativos de la validación con usuarios (SUS y NPS).

Puntuación (0-10)	Categoría	Interpretación del Usuario
9 – 10	Promotores	Usuarios leales y entusiastas. Tienen una alta probabilidad de recomendar la aplicación y de seguir utilizándola para cocinar (retención alta).
7 – 8	Pasivos	Usuarios satisfechos pero no entusiastas. Son indiferentes a la gamificación y vulnerables a ofertas de la competencia. No se consideran para el cálculo del NPS.
0 – 6	Detractores	Usuarios insatisfechos. Es probable que no vuelvan a usar la aplicación y pueden dañar la reputación de la marca mediante el boca a boca negativo.

Cuadro 4.3: Niveles de clasificación del Net Promoter Score (NPS).

4.3.1. Análisis de Resultados Cuantitativos

- **Usabilidad (SUS):** El puntaje promedio de **85** supera el criterio de aceptación (75) y sitúa a la aplicación en el percentil 90 de usabilidad (grado «Excelente»). Destaca el puntaje casi perfecto en el segmento más joven (U1, U2), validando que la interfaz es intuitiva para nativos digitales.
- **NPS:** Con un promedio de 8.6 y una mayoría de «Promotores» (9-10), existe una alta intención de recomendación, lo cual es un fuerte indicador de satisfacción general.

4.3.2. Análisis de Resultados Cualitativos

El análisis de las respuestas abiertas permite conectar estos números con los componentes de la arquitectura:

1. Rendimiento y Arquitectura (Backend/Flutter): Ningún usuario reportó bloqueos o cierres inesperados. U3 y U4 destacaron explícitamente la fluidez: *«Para todo lo que tiene que haber por detrás del análisis de la foto se demora muy poco»*. Esto valida la eficiencia de la integración de **TensorFlow Lite on-device**, confirmando que la arquitectura modular no compromete el rendimiento en dispositivos reales.

2. Comprensión del Feedback: El 100 % de los usuarios comprendió que el ítem «Aporte visión» en la pantalla de resultados correspondía a una evaluación automática de su fotografía. Sin embargo, dado que los criterios específicos (flags de cocción, desorden, etc.) son internos y no se muestran en la UI, participantes como U4 y U5 preguntaron explícitamente: *«¿Qué me faltó para tener el puntaje completo?»*. Esto valida que el modelo funciona e impacta la nota, pero evidencia una oportunidad de mejora en la interfaz para transparentar estos criterios y aumentar el valor pedagógico.

El equipo encargado de la interfaz dispondrá visualmente los criterios de evaluación visual en una versión futura.

3. Impacto de la Gamificación : Se observó una clara diferencia generacional.

- **Usuarios 13-24 años:** Mostraron alto entusiasmo por la Tienda. U1 comentó: «*Quiero cocinar más solo para comprar el tiburón*». Esto valida directamente la hipótesis de que la gamificación mejora la **motivación**.
- **Usuarios 35-55 años:** Valoraron más el Ranking como herramienta de progreso personal que los accesorios estéticos. U8 mencionó: «*Me gusta ver que subo de nivel, me hace sentir que aprendo*».

4.4. Discusión y Verificación de Hipótesis

Los resultados obtenidos permiten contrastar directamente la hipótesis planteada al inicio de este trabajo:

1. **Hipótesis:** «Una arquitectura basada en Flutter, Firebase y TensorFlow Lite permite implementar un sistema móvil robusto». **Validación:** El puntaje SUS de 85.0 y la ausencia de fallos técnicos durante las pruebas con 8 usuarios distintos confirman la robustez de la solución basada en Flutter y Firebase. La gestión segura de usuarios y datos funcionó sin fricciones [2].
2. **Hipótesis:** «El sistema integra de forma eficiente retroalimentación automática en dispositivos móviles». **Validación:** La percepción de «inmediatez» en el análisis de la foto (reportada cualitativamente) valida que el uso de modelos *on-device* (TFLite) fue la decisión arquitectónica correcta para evitar latencias de red, mejorando la experiencia [12].
3. **Hipótesis:** «El diseño de la aplicación contribuye a mejorar la experiencia, motivación y progresión». **Validación:** El NPS de 8.6 y los comentarios sobre el deseo de «seguir cocinando» para ganar puntos (especialmente en el segmento joven) demuestran empíricamente que el módulo de gamificación, soportado por el backend, actúa como un motor efectivo de motivación y retención [3].

En conclusión, la validación con usuarios confirma que la arquitectura técnica propuesta no solo es funcional, sino que habilita exitosamente la experiencia educativa y lúdica que SHEFU promete.

Capítulo 5

Conclusiones y Trabajo Futuro

5.1. Conclusión

El desarrollo de SHEFU ha permitido validar la viabilidad técnica de integrar tecnologías avanzadas de visión computacional y mecánicas de gamificación en una aplicación móvil educativa, soportada por una arquitectura de backend robusta y escalable. La principal conclusión de este trabajo es que el uso de una arquitectura basada en Flutter y Firebase, combinada con la ejecución de modelos de visión computacional *on-device* (TensorFlow Lite), es una estrategia altamente efectiva para aplicaciones móviles que requieren interactividad en tiempo real sin incurrir en altos costos de infraestructura [2], [12].

Esta validación se sustenta directamente en el cumplimiento de los objetivos planteados al inicio de la memoria. En primer lugar, se cumplió el **OE N°1** mediante el diseño e implementación de una arquitectura modular que integra exitosamente el cliente móvil Flutter con los servicios de Firebase. Esta estructura permitió desacoplar la lógica de presentación de la lógica de negocio, facilitando la escalabilidad y el mantenimiento del sistema.

En cuanto a la robustez del sistema, se dio cumplimiento al **OE N°6** a través de la implementación de servicios de backend que gestionan de manera segura y eficiente el ciclo de vida del usuario y la persistencia de datos. Esto fue verificado mediante la ejecución de un plan de pruebas exhaustivo, cubriendo el **OE N°7**, el cual validó la corrección de la lógica de negocio

crítica, como las transacciones de la tienda y la integridad de los datos en escenarios de uso real.

Finalmente, la integración de los componentes de visión computacional y gamificación permitió cumplir con los **OE N°2, 3 y 4**. Desde la perspectiva del backend, se proveyó la infraestructura necesaria para soportar el flujo pedagógico, almacenar los resultados del análisis visual y registrar las métricas de progreso. Los resultados obtenidos en la validación con usuarios (SUS 85.0) confirman que esta base técnica no solo es funcional, sino que impacta positivamente en la experiencia final, demostrando que la arquitectura propuesta contribuye efectivamente a la motivación y progresión en el aprendizaje culinario [3].

5.2. Trabajo Futuro

Si bien el prototipo actual es funcional y valida la hipótesis central, se han identificado líneas de desarrollo clave para transformar SHEFU en un producto comercialmente viable y con mayor impacto en el usuario:

- **Expansión del Modelo de Visión Computacional:** En el prototipo actual, la validación visual se implementó y entrenó específicamente para una receta piloto. El siguiente paso crítico es escalar el entrenamiento del modelo *EfficientNet* para cubrir la totalidad del catálogo de recetas, permitiendo una retroalimentación generalizada en toda la aplicación.
- **Modelo de Negocio y Monetización:** Implementar una pasarela de pagos real para habilitar una economía dentro de la aplicación. Esto incluye la opción de comprar «estrellas» (moneda virtual) para adquirir accesorios en la tienda, o bien, desarrollar una versión *Premium* de suscripción que ofrezca características exclusivas, como recetas avanzadas o análisis visual ilimitado [16].
- **Integración con Ecosistema de Productos:** Complementar la experiencia digital con productos físicos, ofreciendo la venta de utensilios de cocina propios de la marca SHEFU (ej. delantales, espátulas, medidores) que puedan estar vinculados a la app mediante códigos QR o beneficios exclusivos en el juego.

- **Evolución de la Interfaz:** Realizar mejoras visuales en la interfaz de usuario (UI) para alejarla de una estética de «aplicación de utilidad» y acercarla a la de un videojuego. Esto implica añadir más animaciones, efectos de sonido al completar tareas, y una narrativa visual más inmersiva que refuerce el ciclo de gamificación [4].

Bibliografía

- [1] Global Market Insights, «E-Learning Market Size, Share and Growth Report,» 2024.
- [2] Flutter Team. «Flutter - Build Apps for Any Screen. »dirección: <https://flutter.dev>.
- [3] J. Hamari et al., «Does Gamification Work?» En *HICSS*, 2014.
- [4] N. Cheng. «The Ingredients of Design: Diving into the UI of the Zest Cooking App. »dirección: <https://medium.com>.
- [5] Education Technology Insights, «Gamification in EdTech: Revolutionizing the Learning Experience,» 2024.
- [6] SideChef. «RecipeGen AI: Generate Recipes From Photos.»
- [7] Whirlpool Corp., *Yummly Smart Cooking Platform*, Press Release, 2020.
- [8] Innit Inc., *Personalized Cooking Experience with Smart Appliances*, Supermarket News, 2018.
- [9] IBM Corp., «Monitoring recipe preparation using instructive device,» US8992225B2, 2015.
- [10] June Life Inc., «Generative Food Doneness Prediction,» US12050662B2, 2024.
- [11] Y. Zhang et al., «MIRecipe: A Dataset for Stage-Aware Ingredient Recognition,» en *ACM Multimedia*, 2020.
- [12] TensorFlow. «TensorFlow Lite Guide: Edge ML in Mobile Devices. »dirección: <https://viso.ai>.
- [13] R. Marinescu et al., «Data-Driven ITS for Practical Skills,» en *CSEDU Conference*, 2021.

- [14] Lin et al., «Automatic Judgment of Food Color and Cooking Conditions Using AI,» *Processes*, 2021.
- [15] R. Qiao et al., «Food Image Quality Evaluation Using Deep Learning,» *IEEE Access*, 2020.
- [16] Rocket Lab, «Reporte del mercado Mobile App Chile 2024,» 2024.

Apéndice A

Anexos

A.1. Resultados Detallados de la Validación

Para garantizar la transparencia de los datos presentados en el Capítulo 4, se adjunta a continuación el enlace al repositorio digital que contiene el registro detallado de las respuestas obtenidas durante las sesiones de validación.

Este registro incluye el desglose de los puntajes del *System Usability Scale* (SUS), las calificaciones del *Net Promoter Score* (NPS) y la transcripción de los comentarios cualitativos de los 8 participantes.

Enlace a la hoja de resultados:

- https://usmcl-my.sharepoint.com/:x:/g/personal/javier_zamorab_usm_cl/IQBbkWHorh4QSo_st8KQtnZKAVqcg6E48zb_JUNQuHsPRKw?e=ka3xnc