fidélitas
INGENIERÍA EN SISTEMAS DE COMPUTACIÓN

DN01 Documento de Arquitectura

SMIR

HOJA DE CONTROL

Proyecto	*Sistema de Manejo de Inventarios y Registros*		
Entregable	*IN02 Estudio de factibilidad técnica, tecnológica, económica y operacional,*		
Autores	*W Angelo Segura Muñoz, Sebastian Esquivel Solis, Kevin Mena Chinchilla, Allan Gabriel Molina Hernández.*		
Versión/Edición	0100	**Fecha Versión**	*Lunes, 31 de Marzo de 2025*
		Nº Total de Páginas	*20*

REGISTRO DE CAMBIOS

Versión	Causa del Cambio	Responsable del Cambio	Fecha del Cambio

Contenido

Contenido

HOJA DE CONTROL 2
REGISTRO DE CAMBIOS 2
1. Introducción 4
2. Propósito 5
3. Descripción del sistema 5
4. Objetivos y Restricciones arquitectónicas 8
4.1. Plataforma tecnológica 10
4.1.1. Aplicación 10
4.2. Seguridad 11
4.3. Tecnologías utilizadas 12
4.4. Herramientas utilizadas para el desarrollo 13
4.5. Estándares de desarrollo 13
4.5.1. Datos de diseño 16
4.6. Componentes 17
4.6.1. Servidor de pruebas 17
Infraestructura y Configuración 17
Objetivos del Servidor de Pruebas 18
Pruebas y Validaciones 18
Acceso y Control 18
4.6.2. Servidor de producción 18
Infraestructura y Configuración 18
Objetivos del Servidor de Producción 19
Monitoreo y Seguridad 19
Acceso y Control 19
4.6.3. Base de datos 20

4.6.4. Diagrama entidad-relación 21
4.6.5. Diagrama de infraestructura 24
5. Control de Cambios y Versiones 25
5.1.1. Github 25
6. Referencias 26

1. Introducción

El presente documento de arquitectura tiene como finalidad definir la estructura técnica y los componentes fundamentales para el Sistema de Manejo de Inventarios y Registros (SMIR) el cual busca ser implementado en la empresa Abbott Medical. El desarrollo de este sistema surge en respuesta a la necesidad de modernizar y automatizar la gestión de inventarios en entornos de manufactura en el área de electrofisiología (EP), reemplazando métodos tradicionales utilizados en la empresa basados en hojas de cálculo en Excel y hojas de papel por una solución digital integrada. La arquitectura aquí planteada se basa en el uso de Flutter Flow como plataforma para el desarrollo de la interfaz de usuario y la lógica de la aplicación, mientras que Firebase se utilizará para el almacenamiento y gestión en tiempo real de los datos. En este documento se busca abarcar la descripción detallada de la solución, los componentes tecnológicos, las decisiones de diseño y los lineamientos arquitectónicos que aseguran la escalabilidad, seguridad y eficiencia del sistema.

La elección de Flutter Flow como plataforma de desarrollo nos permite desarrollar aplicaciones multiplataforma con un único código base, lo cual agiliza el proceso de desarrollo y facilita la adaptación de la interfaz a dispositivos móviles y web. Por otro lado, la elección de Firebase garantiza una base de datos NoSQL en tiempo real, lo que favorece la sincronización instantánea de la información entre usuarios y dispositivos, además de ofrecer servicios integrados como autenticación, hosting y funciones en la nube. Con esta combinación de tecnologías, el SMIR busca no solo mejorar la precisión y eficiencia en el manejo de inventarios, sino

también proporcionar una solución robusta y adaptable a las necesidades futuras de la empresa.

2. Propósito

El propósito de este documento es definir la arquitectura del SMIR, estableciendo las bases técnicas y estratégicas que guiarán el desarrollo, integración y despliegue del sistema. En este sentido, el documento tiene los siguientes objetivos:

- **Describir la estructura del sistema:** Proveer una visión global de los componentes, módulos y flujos de datos que conforman la solución.
- **Justificar las decisiones tecnológicas:** Explicar por qué se ha seleccionado Flutter Flow y Firebase, destacando las ventajas que ofrecen en términos de desarrollo ágil, escalabilidad, sincronización en tiempo real y reducción de costos iniciales.
- **Establecer lineamientos para el desarrollo:** Definir las normas y restricciones que regirán la implementación del sistema, asegurando que todas las partes involucradas tengan una referencia común para la integración de funcionalidades.
- **Facilitar la comunicación técnica:** Servir como documento de referencia para los desarrolladores, diseñadores, administradores y stakeholders, garantizando la consistencia y coherencia en el proceso de desarrollo y despliegue.

3. Descripción del sistema

El SMIR busca ser un sistema diseñado para optimizar la gestión de inventarios y el registro de actividades en entornos de manufactura. La solución está orientada a resolver los problemas actuales que enfrenta la empresa en el manejo

de datos mediante hojas de cálculo, los cuales generan errores, insuficiencias y dificultades en el seguimiento en tiempo real de los materiales.

Área Solicitante del Sistema:

El sistema está dirigido a la administración del área de inventarios de la empresa específicamente en el área de producción, la cual es responsable del control y registro de insumos, materiales y productos en proceso de manufactura. Esta área requiere una solución automatizada que facilite la entrada, actualización y consulta de datos, garantizando precisión y disponibilidad de la información en tiempo real.

Necesidades Identificadas:

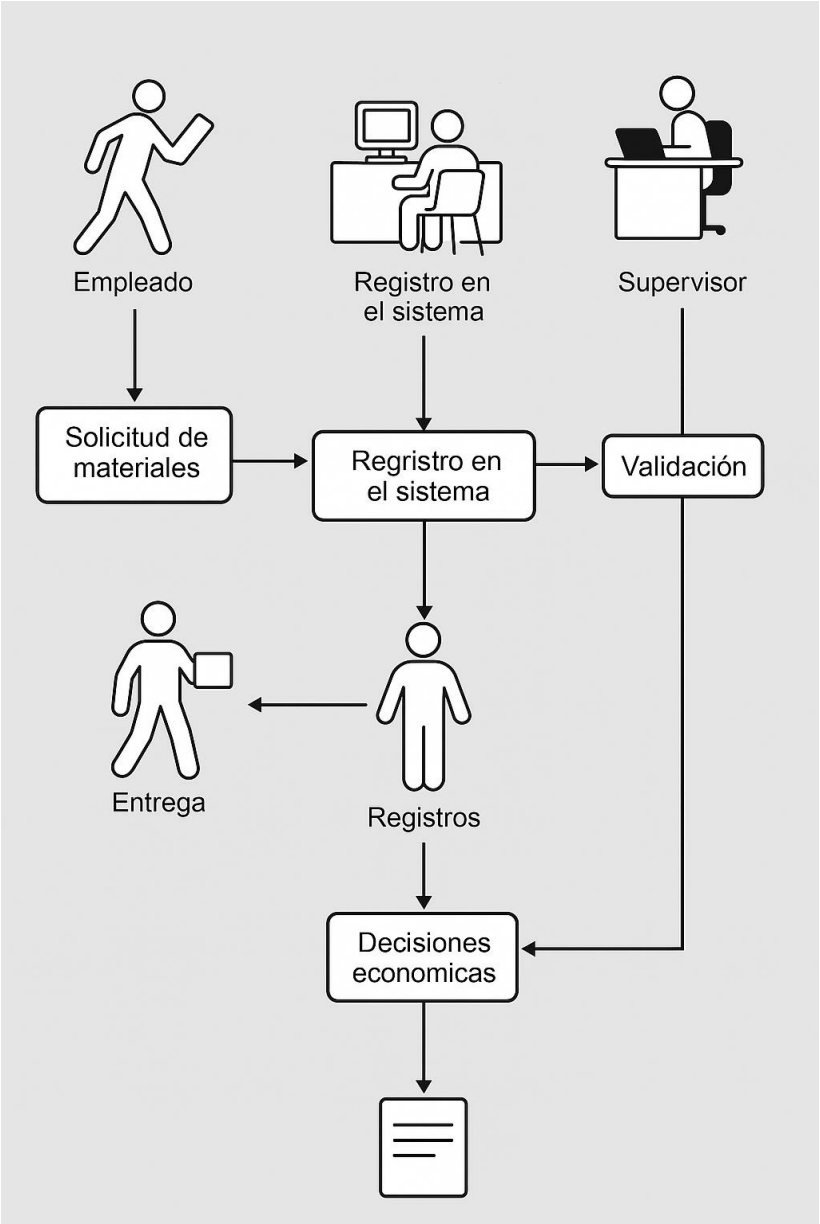
- **Precisión en el registro:** Minimizar los errores humanos derivados del uso de métodos manuales.
- **Actualización en tiempo real:** Permitir que los datos se sincronicen instantáneamente entre diferentes dispositivos y usuarios.
- **Optimización del control de stock:** Garantizar la disponibilidad y correcta gestión de materiales, evitando sobrecostos y faltantes.
- **Facilidad de uso:** Contar con una interfaz intuitiva y responsiva que facilite la adopción del sistema por parte de los usuarios.
- **Seguridad de la información:** Proteger los datos críticos a través de controles de acceso y mecanismos de respaldo.

Funciones del Sistema para Solventar las Necesidades:

- **Gestión de Empleados y Roles:** Registro y administración de usuarios con permisos personalizados para asegurar que cada rol tenga acceso a la información pertinente.
- **Control de Inventarios:** Registro detallado de materiales, seguimiento de entradas y salidas, generación de alertas de bajo stock y fechas de vencimiento.

- **Registro Diario de Actividades:** Documentación de tareas, movimientos de materiales y eventos operativos, facilitando la auditoría y análisis histórico.
- **Integración en Tiempo Real:** Uso de Firebase para sincronización inmediata de datos entre la aplicación y la base de datos, permitiendo actualizaciones en vivo.
- **Interfaz Multiplataforma:** Desarrollo en FlutterFlow para asegurar una experiencia de usuario consistente en dispositivos móviles y web.

Reportes y Análisis: Generación de informes automatizados que faciliten la toma de decisiones estratégicas en la gestión de inventarios.



4. Objetivos y Restricciones arquitectónicas

Objetivos Arquitectónicos:

- **Escalabilidad:**

  El sistema debe ser capaz de soportar un crecimiento en el número de usuarios y en el volumen de datos sin degradar el rendimiento. La arquitectura basada en Firebase y Flutter Flow permite añadir recursos de forma dinámica y escalar horizontalmente según sea necesario.

- **Integración en Tiempo Real:**

  Es crucial que la solución garantice la sincronización instantánea de la información, permitiendo que los datos ingresados o modificados por un usuario se reflejen de inmediato en todas las interfaces. Esto se logra a través de la base de datos NoSQL en tiempo real que ofrece Firebase.

- **Seguridad:**

  La arquitectura debe incorporar mecanismos robustos de autenticación y autorización para proteger la integridad y confidencialidad de los datos. Se utilizarán las capacidades de Firebase Authentication y las reglas de seguridad de Firestore para controlar el acceso.

Restricciones Arquitectónicas:

- **Dependencia de Firebase:**

  La solución estará altamente acoplada a los servicios de Firebase, lo cual implica que cualquier cambio significativo en la plataforma podría requerir ajustes en el sistema. Se deben establecer mecanismos de monitoreo y respaldo para mitigar riesgos asociados.

- **Costos de Escalabilidad:**

  El modelo de pago por uso de Firebase puede resultar en costos crecientes a medida que aumenta el tráfico y el volumen de datos. Se deberá monitorear el consumo de recursos y optimizar las consultas para mantener los costos bajo control.

- **Compatibilidad Multiplataforma:**

  La aplicación debe funcionar de manera óptima en diferentes dispositivos y sistemas operativos. Esto implica restricciones en cuanto a las versiones mínimas de los sistemas operativos soportados, así como en el rendimiento en dispositivos de gama baja.

4.1. Plataforma tecnológica

4.1.1. Aplicación

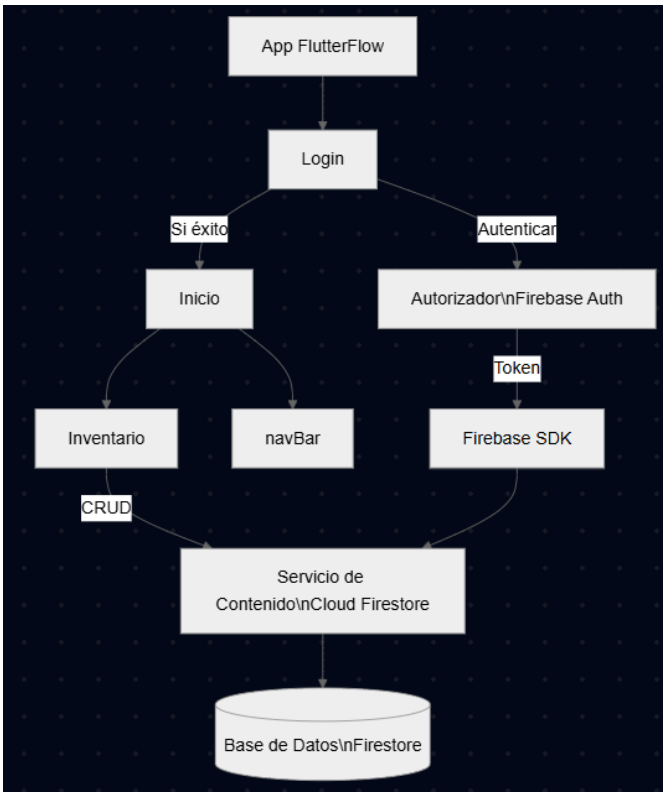
El proyecto se desarrollará utilizando una arquitectura serverless que combina Flutter Flow como solución frontend y Firebase como Backend-as-a-Service (BaaS). Esta selección tecnológica ofrece importantes ventajas para el desarrollo de nuestro sistema de control de inventario con gráficas de métricas.

Flutter Flow permite crear interfaces de usuario complejas mediante su enfoque visual, lo que llega a facilitar bastante la construcción de un frontend. Específicamente para la creación de una aplicación de inventarios, esto nos permitirá implementar rápidamente pantallas para: gestión de productos, seguimiento de stock, generación de reportes y visualización de métricas a través de gráficos interactivos. La capacidad de Flutter Flow de generar código Flutter nativo asegura un rendimiento óptimo en dispositivos móviles y web

Firebase nos proporcionará todos los servicios esenciales para administrar tanto el almacenamiento de la aplicación como la autenticación segura de usuarios. Firebase Authentication ofrece una solución completa para gestionar el acceso de los usuarios, garantizando que cada persona solo pueda visualizar y acceder a las secciones autorizadas según su perfil. Este sistema de autenticación es fundamental para mantener la seguridad e integridad de los datos del inventario.

Para el manejo de datos, implementaremos Cloud Firestore, una base de datos NoSQL en tiempo real especialmente diseñada para aplicaciones como nuestro sistema de control de inventario. Al estar alojada en la nube, Firestore elimina la necesidad de administrar infraestructura física y ofrece escalabilidad automática. Su estructura no relacional nos permite modelar toda la base de datos de manera

flexible, adaptándose perfectamente a los requerimientos cambiantes del inventario sin preocuparnos por relaciones complejas entre tablas o esquemas rígidos.

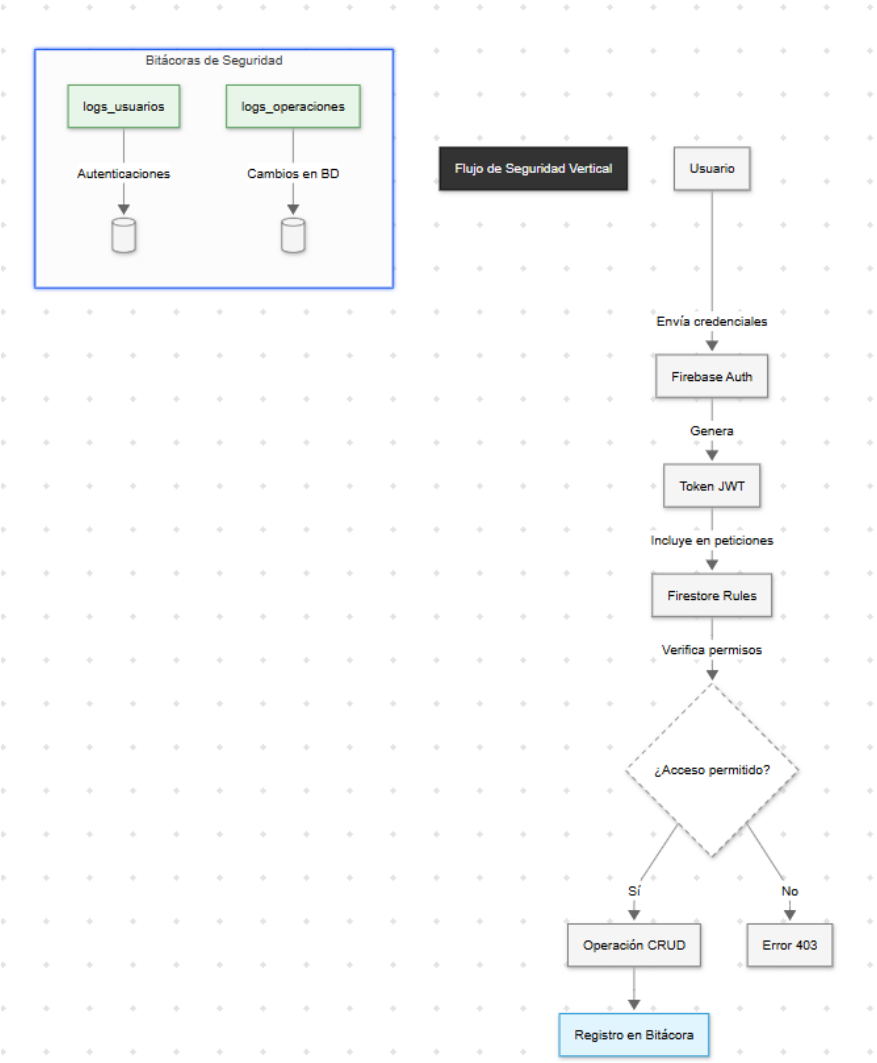


4.2. Seguridad

El sistema implementa un modelo de seguridad robusto basado en el principio de mínimo privilegio, garantizando que los usuarios accedan solamente a los recursos estrictamente necesarios para sus funciones. Combina dos funciones principales las cuales son:

Autenticación fuerte mediante Firebase Auth (email/contraseña encriptada y proveedores OAuth como Google).

Autorización por roles (Admin, Editor, Lector) mediante Firestore Rules, que validan permisos antes de cada operación.



4.3. Tecnologías utilizadas

El proyecto utiliza Flutter Flow como herramienta principal para el desarrollo frontend, permitiendo crear interfaces móviles y web de manera visual mediante componentes arrastrables. Esta plataforma genera código Flutter en Dart, que puede complementarse con programación manual cuando se necesitan funcionalidades más avanzadas. Flutter Flow acelera el desarrollo mientras mantiene la flexibilidad de personalización.

Backend y Base de Datos

Para el backend empleamos Firebase, que ofrece servicios completos sin necesidad de gestionar servidores. Incluye Firebase Auth para autenticación segura, Cloud Firestore como base de datos NoSQL en tiempo real, y Firebase Storage para archivos. Las operaciones complejas pueden implementarse con Cloud Functions usando JavaScript/TypeScript.

Control de Versiones y Colaboración

Utilizamos GitHub para el control de versiones y trabajo colaborativo. Esta plataforma permite gestionar el código generado por Flutter Flow, realizar revisiones, mantener diferentes branches para desarrollo/producción, y coordinar el trabajo entre equipos mediante pull requests y issues.

4.4. Herramientas utilizadas para el desarrollo

Durante el desarrollo del proyecto y con el fin de llegar a un final satisfactorio se utilizarán las siguientes herramientas

Detalle softwares e IDE utilizados en el desarrollo del proyecto

Nombre	Descripción
Flutter Flow Web IDE	Firebase Authentication
Flutter	GIthub
Firestore Storage	

4.5. Estándares de desarrollo

1.1. Estándares para el Desarrollo de Interfaces de Usuario (UI)

a) Consistencia en el Diseño

- **Diseño Unificado con Material Design***:* Siguiendo los principios de Material Design para garantizar una experiencia visualmente coherente y estéticamente agradable en todas las plataformas (Android y Windows).

- **Paleta de Colores y Tipografía:** La paleta de colores y tipografía se seleccionará basándose en la identidad de la empresa, asegurando un contraste adecuado y accesibilidad.

b) Accesibilidad

- **Contraste y Ajuste de Texto:** Se asegurará que el contraste entre texto y fondo sea suficiente para garantizar la legibilidad, y que los tamaños de texto puedan ajustarse fácilmente mediante la configuración de accesibilidad del sistema operativo*.*

c) Responsividad y Adaptabilidad

- **Diseño Responsive con FlutterFlow:** Se emplearán widgets flexibles como Flexible, Expanded y MediaQuery para garantizar que la aplicación se adapte correctamente a diferentes tamaños de pantalla en dispositivos móviles y web.

- **Ajustes en Pantallas Pequeñas:** Utilizando la propiedad LayoutBuilder de FlutterFlow, se ajustará el diseño dinámicamente dependiendo del tamaño de la pantalla, asegurando que la interfaz se vea bien en teléfonos, tabletas y escritorios.

d) Flujo de Usuario

- **Navegación Clara y Fluida:** Se utilizará la arquitectura de navegación declarativa de FlutterFlow, aprovechando el Navigator y las rutas para garantizar una navegación fluida entre pantallas, con transiciones suaves y claras para el usuario.

- **Feedback Visual**: Para proporcionar retroalimentación al usuario en tareas como la carga de información o el manejo de errores, se implementarán snackbars, dialogs y botones de carga utilizando los widgets SnackBar, AlertDialog, CircularProgressIndicator, etc.

2. Herramientas y Tecnologías Utilizadas

- **Framework Principal:** *FlutterFlow para el desarrollo multiplataforma (Android y Windows).*

- **Base de Datos:** *Firestore (de Firebase) para la gestión de inventarios y usuarios, con sincronización en tiempo real.*

- **Autenticación:** *Firebase Authentication para la autenticación de los usuarios y administradores.*

- **Control de versiones:** *FlutterFlow, y GitHub para la gestión del código fuente y la colaboración entre desarrolladores.*

2. Estándares de Codificación

La codificación sigue las mejores prácticas de FlutterFlow y Dart, garantizando que el código sea limpio, eficiente y fácil de mantener:

a) Convenciones de Nombres

- **CamelCase y PascalCase**: *El código sigue la convención camelCase para variables y métodos (por ejemplo, nombreProducto, actualizarInventario) y* **PascalCase** *para clases y componentes (por ejemplo, InventarioPage, ProductoForm).*
- **Nombres Descriptivos**: *Se da prioridad a los nombres descriptivos que reflejan claramente la función o el propósito de las variables y métodos.*

b) Estructura del Código

- **Indentación Consistente**: *Se utiliza una indentación de 2 espacios por nivel para asegurar que el código sea fácilmente legible y consistente en todos los archivos del proyecto.*
- **Líneas Cortas y Legibles**: *Cada línea de código no debe exceder los 80-100 caracteres, favoreciendo la legibilidad y facilitando las revisiones de código.*

c) Comentarios y Documentación

- **Comentarios Claros y Concisos**: *Se documentan los métodos complejos utilizando comentarios breves que describen su lógica y propósito. Se utiliza la sintaxis de Docstrings de Dart para documentar funciones y clases.*
- **Documentación de Funciones y Métodos**: *Cada función y clase crítica está documentada para indicar su propósito, parámetros, y valores de retorno.*

d) Pruebas y Validación

- **Pruebas Unitarias y de Widget**: *Utilizamos FlutterFlow test para realizar pruebas unitarias de los métodos y pruebas de widgets para asegurar que la UI se comporta como se espera.*
- **Pruebas de Integración**: *Se implementan pruebas de integración para garantizar que los módulos del sistema trabajen juntos correctamente. Esto incluye pruebas para la sincronización de datos entre el frontend y Firestore, por ejemplo.*

e) Seguridad en el Código

- **Autenticación y Seguridad de Datos**: *Utilizamos Firebase Authentication para gestionar la autenticación de los usuarios de manera segura, para asegurar las sesiones de los usuarios.*
- **Protección de Datos Sensibles**: *Se aseguran prácticas de seguridad como el cifrado de las contraseñas y el uso de Firestore Security Rules para controlar el acceso a los datos de la base de datos.*

f) Optimización del Rendimiento

- **Optimización de la Carga de Datos**: *Implementamos carga perezosa (Lazy Loading) en listas largas, como la gestión de productos, utilizando widgets.*

- **Minimización de la Red**: *Se usan técnicas como batching de peticiones a Firestore para reducir la cantidad de consultas a la base de datos, mejorando el rendimiento general de la aplicación.*

3. Herramientas y Tecnologías Utilizadas

- **Framework Principal**: *FlutterFlow para el desarrollo multiplataforma (Android, iOS y Web).*
- **Base de Datos**: **Firestore** *(de Firebase) para la gestión de inventarios y usuarios, con sincronización en tiempo real.*
- **Autenticación***: Firebase Authentication para la autenticación de los usuarios y administradores.*
- **Control de versiones**: *FlutterFlow, y GitHub para la gestión del código fuente y la colaboración entre desarrolladores.*

4.5.1. Datos de diseño

Tipografías	
Títulos Primarios	Rubik 45
Títulos Secundarios	Rubik 26
Texto Plano	Rubik 16
Acentos y enlaces	Rubik 16
Cromática	

Iconografía

4.6. Componentes

4.6.1. Servidor de pruebas

El servidor de pruebas para el Sistema de Manejo de Inventarios y Registros (SMIR) estará diseñado para permitir la validación y verificación de todas las funcionalidades antes de su implementación en producción. Este servidor tendrá las siguientes características:

Infraestructura y Configuración

- **Plataforma:** Firebase (Firestore, Authentication, Cloud Functions, Hosting y Cloud Storage)
- **Entorno:** Firebase Project separado exclusivamente para pruebas
- **Base de Datos:** Firestore con un conjunto de datos de prueba limitado, simulando escenarios reales con volúmenes controlados
- **Autenticación:** Firebase Authentication con usuarios ficticios y distintos niveles de acceso para pruebas de seguridad y roles
- **Hosting:** Firebase Hosting para pruebas de despliegue y optimización de carga

- **Funciones en la Nube:** Cloud Functions configuradas para replicar la lógica de negocio sin afectar el entorno de producción
- **Manejo de Versiones:** Implementación en FlutterFlow con integración continua para despliegues controlados en el entorno de pruebas

Objetivos del Servidor de Pruebas

- Validación de nuevas funcionalidades antes de su paso a producción
- Detección y corrección de errores en un entorno seguro
- Pruebas de rendimiento y carga con datos simulados
- Evaluación de la integración en tiempo real de Firebase Firestore
- Pruebas de seguridad y validación de reglas de acceso en Firestore

Pruebas y Validaciones

- **Pruebas Unitarias:** Evaluación de funciones individuales de la aplicación
- **Pruebas de Integración:** Validación de comunicación entre módulos y bases de datos
- **Pruebas de Seguridad:** Análisis de accesos y restricciones en Firestore
- **Pruebas de Carga:** Simulación de múltiples usuarios concurrentes
- **Pruebas de Interfaz:** Evaluación de la experiencia de usuario y adaptabilidad a distintos dispositivos

Acceso y Control

- Solo accesible para el equipo de desarrollo y QA
- Datos de prueba limitados y eliminados regularmente
- Registros y logs para monitorear fallos y mejorar el sistema

4.6.2. Servidor de producción

El servidor de producción será la instancia final del SMIR que se desplegará para ser utilizado en la empresa Abbott Medical. Su configuración garantizará la estabilidad, seguridad y escalabilidad necesarias para el correcto funcionamiento del sistema en un entorno real.

Infraestructura y Configuración

- **Plataforma:** Firebase con Firestore, Authentication, Cloud Functions, Hosting y Cloud Storage
- **Entorno:** Firebase Project independiente y optimizado para producción

- **Base de Datos:** Firestore con datos en tiempo real y optimización para grandes volúmenes
- **Autenticación:** Firebase Authentication con integración de seguridad avanzada (Autenticación multi factor, OAuth, etc.)
- **Hosting:** Firebase Hosting con configuración avanzada de caché y rendimiento
- **Funciones en la Nube:** Cloud Functions optimizadas para operaciones críticas de negocio
- **Respaldo y Recuperación:** Políticas de respaldo automático y redundancia de datos
- **Manejo de Versiones:** Despliegues controlados desde GitHub con revisión de código previa

Objetivos del Servidor de Producción

- Garantizar disponibilidad y rendimiento óptimo para los usuarios finales
- Protección de datos con autenticación segura y reglas estrictas de acceso
- Sincronización en tiempo real sin afectar la escalabilidad
- Mantenimiento y actualizaciones sin interrupción del servicio

Monitoreo y Seguridad

- **Reglas de Seguridad en Firestore:** Configuración avanzada para acceso por roles
- **Monitoreo en Firebase:** Uso de Firebase Performance Monitoring y Crashlytics para detectar errores
- **Protección de Datos:** Cifrado de datos en tránsito y en reposo
- **Acceso Restringido:** Implementación de permisos según roles definidos en la empresa
- **Políticas de Escalabilidad:** Ajuste dinámico de recursos para manejar picos de tráfico

Acceso y Control

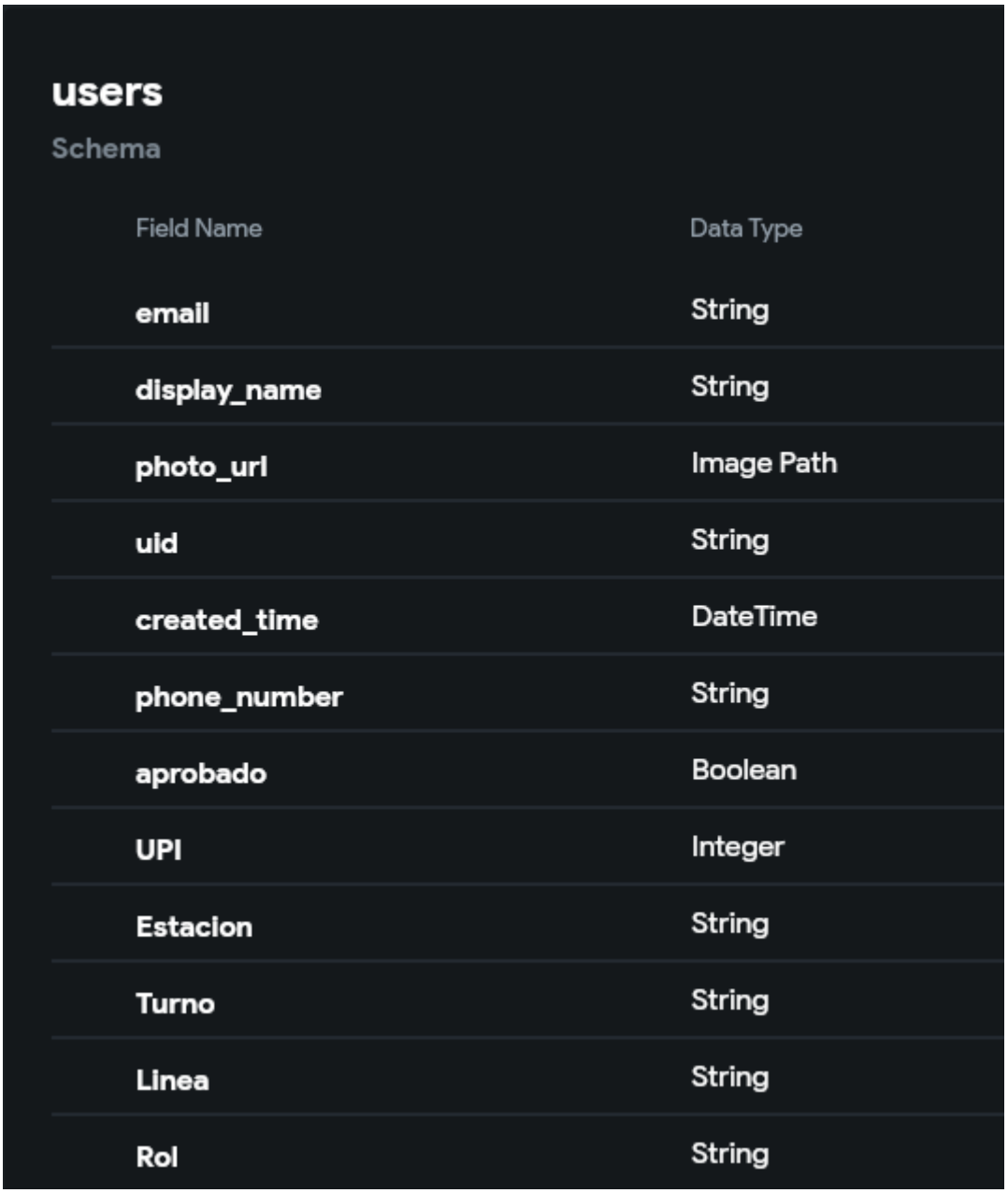
- Accesible sólo por usuarios autenticados y autorizados
- Monitoreo continuo con reportes de actividad
- Procedimientos definidos para mantenimiento y actualización del sistema

4.6.3. Base de datos

Herramientas de Construcción	
Entorno de Desarrollo	Firebase Firestore
Lenguaje	Firestore
Sistema Gestor de Base de Datos	Firebase Firestore (NoSQL)
Justificación	

4.6.4. Diagrama entidad-relación

Debido al uso de firebase la cual es NoSql no existen relaciones pero si las tablas que funcionan como contenedores de datos, a continuación se presentan la lógica de tablas que va a contener la aplicación y todas sus columnas con el tipo de dato de cada una.



Inventario

Schema

Field Name	Data Type
ID	Integer
Nombre	String
Vencimiento	DateTime
Cantidad	Integer
Costo	Double

Entregas

Schema

Field Name	Data Type
Empleado	Doc Reference (users)
Material	Doc Reference (Inventario)
Cantidad	Integer
FechaEntrega	DateTime
Encargado	Doc Reference (users)

Batch

Schema

Field Name	Data Type
Material	Doc Reference (Inventario)
Cantidad	Integer

Alertas

Schema

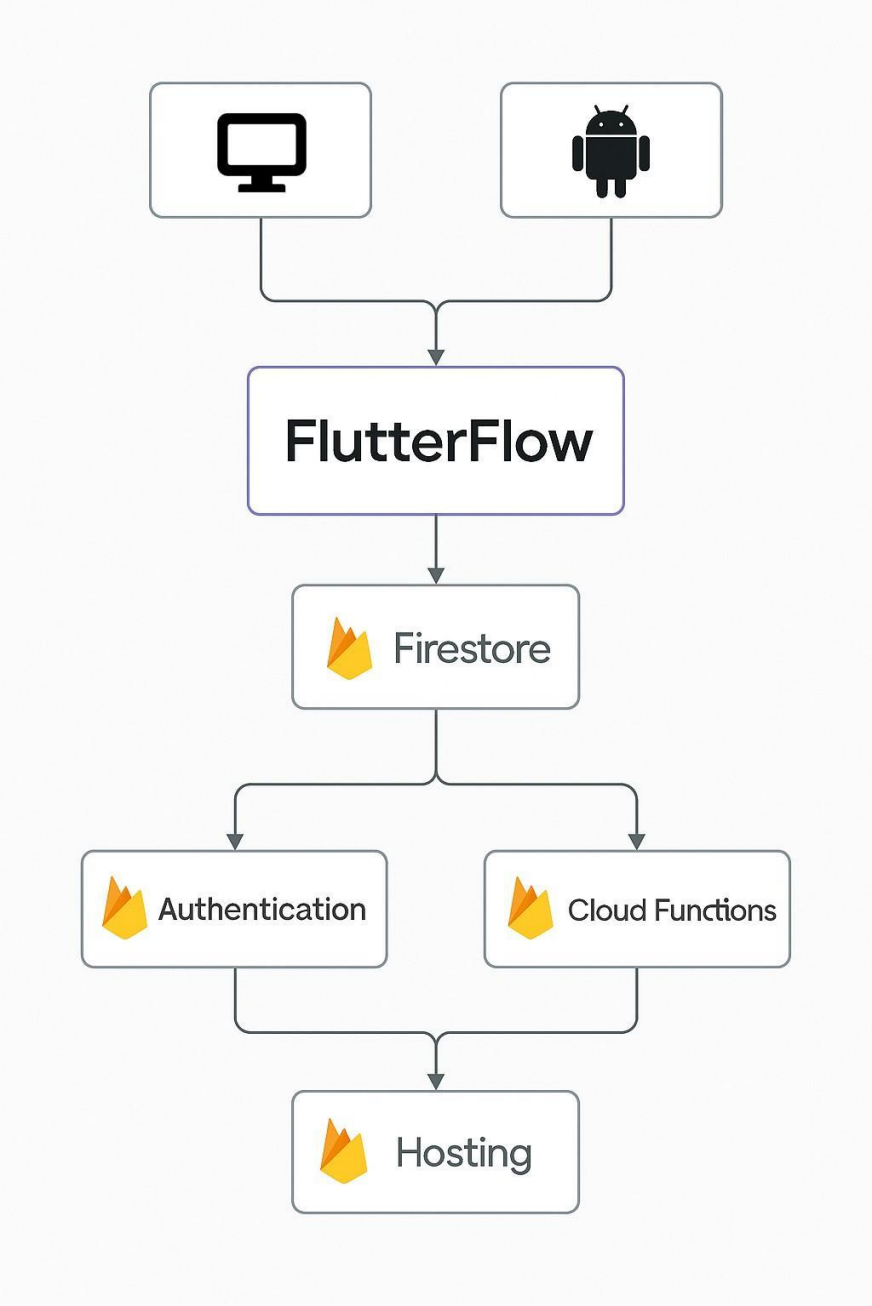
Field Name	Data Type
MensajeAlerta	String

Criticos

Schema

Field Name	Data Type
MaterialCritico	Doc Reference (Inventario)
Cantidad	Integer

4.6.5. Diagrama de infraestructura

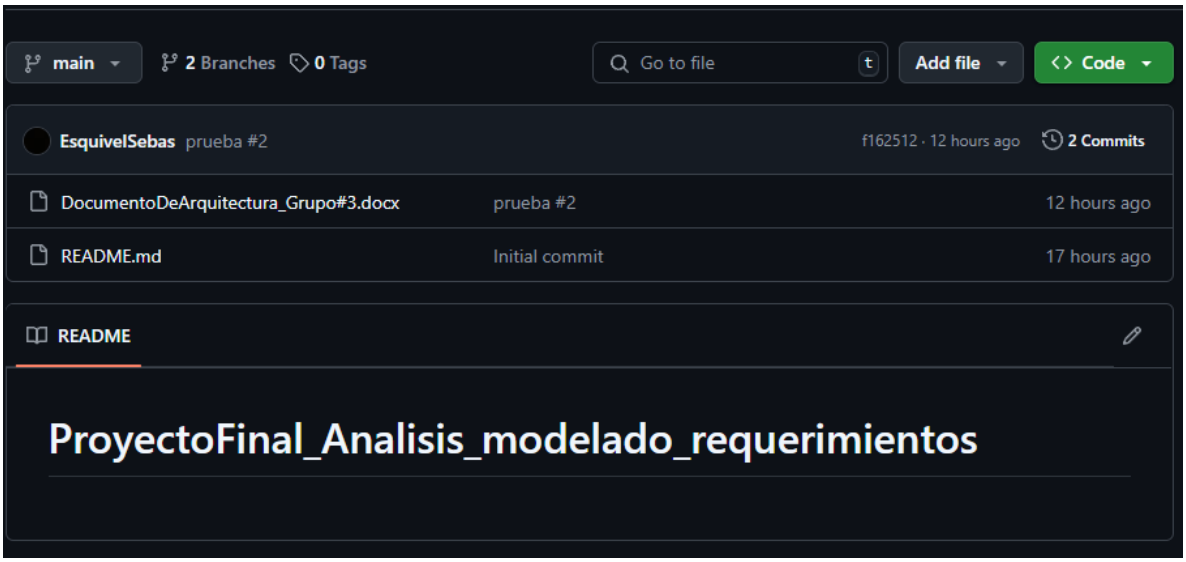


5. Control de Cambios y Versiones

Para el control de versiones del proyecto, implementaremos un flujo de trabajo centrado en Flutter Flow como herramienta principal de desarrollo, complementado con GitHub como repositorio remoto. Todos los cambios realizados en Flutter Flow (diseños de interfaz, lógica y configuraciones) se sincronizan periódicamente con el repositorio de GitHub mediante commits descriptivos, permitiendo mantener un historial completo de versiones. Esta integración nos brinda un respaldo seguro del código generado, facilita la colaboración entre desarrolladores mediante branches independientes, y permite implementar revisiones de código a través de pull requests, asegurando la trazabilidad de cada modificación en el proyecto. Además, utilizaremos tags en GitHub para marcar versiones estables correspondientes a releases importantes de la aplicación.

5.1.1. Github

https://github.com/EsquivelSebas/ProyectoFinal_Analisis_modelado_requerimientos.git



6. Referencias

[1] "Ejemplo de Plantilla de Diagrama de Seguridad de Red: Visualiza la Protección", MyMap.AI. [En línea]. Disponible en: https://www.mymap.ai/es/template/network-security-diagram-example. [Consultado: 31-mar-2025].

[2] "Diagramas de seguridad", Creately.com. [En línea]. Disponible en: https://creately.com/diagram/example/2OuZxWUuaUp/diagramas-de-seguridad. [Consultado: 31-mar-2025].

[3] "Cómo elaborar 5 tipos de diagramas de arquitectura", Lucidchart, 05-abr-2021. [En línea]. Disponible en: https://www.lucidchart.com/blog/es/como-elaborar-diagramas-de-arquitectura. [Consultado: 31-mar-2025].

[4] Amazon.com. [En línea]. Disponible en: https://aws.amazon.com/es/what-is/architecture-diagramming/#:~:text=Los%20diagramas%20de%20arquitectura%20del,la%20arquitectura%20de%20un%20sistema. [Consultado: 31-mar-2025].

.