|  |
| --- |
| **Analista Programador Computacional**  **Portafolio de Título**  **Docente: Felix Cifuentes** |
| Propuesta de Proyecto y Especificación de Requisitos de Software |
| *Proyecto: TatasApp* |
|  |
|  |
| **02 de junio 2025** |

|  |
| --- |
|  |

**Contenido**

**Contenido**

[**1. Introducción 4**](#_heading=h.wy6pzxb3hvws)

[1.1. Propósito 4](#_heading=h.ry4f1amcena9)

[1.2. Ámbito del Sistema 4](#_heading=h.yhect7d0spql)

[1.3. Visión General del Documento 5](#_heading=h.kldva25fd18y)

[**2. Descripción General 5**](#_heading=h.wwvjl5tif9r7)

[2.1. Perspectiva del Producto 5](#_heading=h.kc1mvnxy96z9)

[2.2. Funciones del Producto 6](#_heading=h.uzmg02hor4uv)

[2.3. Características de los Usuarios 8](#_heading=h.hdvoem3tosh0)

[2.4. Arquitectura 8](#_heading=h.hdvoem3tosh0)

[2.5. Restricciones](#_heading=h.ojkewwnf99i1) 12

[2.6. Suposiciones y Dependencias](#_heading=h.nt4zjbwrdjno) 12

[**3. Requisitos Específicos**](#_heading=h.dwrtx6ex4p0o) **13**

[3.1 Requisitos comunes de las interfaces](#_heading=h.jci6cv72hg5h) 13

[3.1.1 Interfaces de usuario 1](#_heading=h.ix4n9w5xb4mn)3

[3.1.2 Interfaces de hardware 1](#_heading=h.58yv8qlnazdp)3

[3.1.3 Interfaces de software 1](#_heading=h.soqbw720o96x)3

[3.1.4 Interfaces de comunicación 1](#_heading=h.7sm2dfqar3ve)3

[3.2 Requisitos funcionales y no funcionales 1](#_heading=h.vlntd6lcecxr)4

[**4. Propuesta de Planificación 1**](#_heading=h.uet6w719gsy2)**6**

[4.1 Descripción general acerca de la Planificación 1](#_heading=h.4m8eljj8w9qq)6

[4.1.1 Definición del Equipo de Trabajo 1](#_heading=h.hvwbyieb00s)6

[4.1.2 Carta Gantt 1](#_heading=h.pyltuwefi33c)7

[**5. Conclusión 1**](#_heading=h.vj3e1x1uhpz)**9**

**Ficha del documento**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Revisión** | **Autor** | **Modificación** |
| 02/06/2025 | 01 | Alexander Aguilera  Andrea Pino  David Guentelican | V1.0 |

**Integrantes:**

|  |  |
| --- | --- |
| Nombre Integrante del Equipo | Rol Definido |
| David Guentelicán |  |
| Alexander Aguilera |  |
| Andrea Pino |  |

# 1. Introducción

TatasApp es una solución móvil diseñada para apoyar el bienestar y la seguridad de los adultos mayores, desarrollada en el marco del proyecto de título de la carrera de Analista Programador Computacional. La aplicación busca fortalecer el vínculo entre los adultos mayores y sus familiares mediante herramientas tecnológicas que promuevan la autonomía, el acompañamiento y la prevención de riesgos.

En la actualidad, muchas familias se enfrentan al desafío de cuidar a personas mayores que viven solas o requieren supervisión periódica. Esto genera la necesidad de contar con soluciones accesibles y eficaces que permitan monitorear su estado, comunicarse rápidamente ante emergencias y organizar sus rutinas diarias.

Por lo anterior, el equipo desarrollador ha creado TatasApp, una aplicación móvil multiplataforma desarrollada con Ionic y Angular, conectada a una API desarrollada en FastAPI, con base de datos en PostgreSQL y soporte de notificaciones mediante Capacitor.

## 1.1. Propósito

El propósito del presente documento es definir de manera clara y precisa los requerimientos del sistema TatasApp, con el fin de establecer una base sólida para su desarrollo, pruebas y posterior implementación. Este documento detalla tanto los requisitos funcionales como no funcionales del sistema, orientados a garantizar la calidad, seguridad y utilidad de la aplicación.

TatasApp tiene como objetivo principal brindar una herramienta que permita a los adultos mayores mantenerse seguros y conectados, al mismo tiempo que sus familiares o cuidadores puedan monitorear su estado en tiempo real, recibir alertas ante emergencias y organizar eventos de manera compartida.

Además, este documento sirve como guía para el equipo de desarrollo y punto de referencia para evaluar el cumplimiento de los objetivos del proyecto, facilitando una gestión efectiva del ciclo de vida del software.

## 1.2. Ámbito del Sistema

El sistema TatasApp está orientado a cubrir funcionalidades específicas dentro del entorno doméstico de apoyo al adulto mayor, limitándose a la gestión de usuarios, administración de rutinas, alertas de emergencia y funcionalidades básicas de localización. Se implementa exclusivamente en dispositivos móviles Android y está destinado al uso directo de adultos mayores y sus familiares o cuidadores.

Dentro del alcance de esta versión se consideran las siguientes capacidades: autenticación de usuarios con roles diferenciados, calendario compartido para eventos diarios, generación de alertas de emergencia (manuales o simuladas), detección de inactividad, configuración de zonas seguras mediante coordenadas, almacenamiento local con sincronización posterior, y emisión de notificaciones locales tanto visuales como habladas. Estas funcionalidades son gestionadas a través de una aplicación móvil conectada a un backend propio, con base de datos alojada en Supabase.

No se contempla la interoperabilidad con sistemas clínicos, bases de datos institucionales ni la integración con dispositivos externos como smartwatches, sensores biomédicos o plataformas de monitoreo remoto. El sistema tampoco realiza diagnósticos ni reemplaza la supervisión de personal especializado en salud.

## 1.3. Visión General del Documento

El presente documento tiene como finalidad ofrecer una visión integral del sistema TatasApp, comenzando por una descripción general del producto y su propósito. Posteriormente, se profundiza en las funcionalidades principales que ofrece la aplicación, los perfiles de usuarios que la utilizarán y las características específicas asociadas a cada uno de ellos.

Asimismo, se detallan los requisitos funcionales y no funcionales del sistema, así como los requerimientos técnicos necesarios para su correcto funcionamiento, tanto en el entorno móvil como en el backend desarrollado en FastAPI.

Finalmente, se incluye una planificación detallada del desarrollo del sistema, en la cual se definen los roles del equipo de trabajo, sus funciones específicas y los plazos establecidos para cada una de las etapas del proyecto.

# 2. Descripción General

## 2.1. Perspectiva del Producto

TatasApp es una aplicación móvil multiplataforma enfocada en mejorar la calidad de vida de los adultos mayores y facilitar la gestión del cuidado por parte de sus familiares o cuidadores. El sistema está diseñado para funcionar de forma integrada entre la aplicación móvil, una API desarrollada en FastAPI y una base de datos PostgreSQL. Para efectos del desarrollo y pruebas, la base de datos está alojada en Supabase, plataforma que proporciona un entorno gestionado que facilita la conexión con el backend, el manejo de autenticación y el acceso a datos mediante API automáticas. PostgreSQL, como motor de base de datos, es responsable de almacenar y estructurar la información del sistema de forma robusta y confiable, mientras que Supabase actúa como intermediario que simplifica el acceso y la administración durante el proceso de desarrollo.

El producto permite automatizar y simplificar tareas esenciales como la generación de alertas de emergencia (SOS o por caída), el seguimiento por geolocalización mediante zonas seguras, y la organización de eventos mediante un calendario interno compartido. Además, la aplicación utiliza el sistema de notificaciones locales provisto por Capacitor para alertar a los usuarios sobre eventos importantes de forma oportuna, incluso sin conexión a internet. Complementariamente, se ha integrado la funcionalidad de notificaciones habladas mediante la librería Text-to-Speech, lo que mejora la accesibilidad y la experiencia del usuario, especialmente en adultos mayores con dificultades visuales o de lectura.

Esta integración de funcionalidades permite minimizar riesgos, facilitar la supervisión remota y aumentar la autonomía del adulto mayor, al mismo tiempo que entrega a los familiares herramientas eficaces para el acompañamiento y la toma de decisiones informadas.

## 2.2. Funciones del Producto

Las funciones del producto se estructuran en torno a tres ejes fundamentales: seguridad, gestión de rutinas y conectividad familiar. A continuación, se detallan las principales funcionalidades ofrecidas por el sistema:

* **Registro y gestión de usuarios**

TatasApp permite la creación de cuentas diferenciadas para adultos mayores y familiares o cuidadores, utilizando credenciales seguras compuestas por correo electrónico y contraseña. Cada tipo de usuario accede a funcionalidades específicas definidas según su rol dentro de la aplicación. Durante el proceso de registro, se recopilan datos personales como nombres, apellidos, fecha de nacimiento, dirección (con autocompletado e identificación de comuna y región mediante Google Places), correo electrónico, número de teléfono y tipo de usuario.

* **Inicio de sesión con validación**

El sistema permite a los usuarios iniciar sesión mediante un proceso de autenticación contra una API desarrollada en FastAPI, validando en tiempo real las credenciales ingresadas, las cuales se encuentran almacenadas en la base de datos principal. Una vez autenticado, se genera un token de acceso que es almacenado de forma segura en la base de datos local del dispositivo, lo que permite mantener la sesión activa y agilizar futuros accesos sin necesidad de volver a ingresar las credenciales.

* **Gestión de eventos personales**

Los usuarios pueden crear, visualizar, editar y eliminar eventos asociados a la rutina del adulto mayor, tales como la toma de medicamentos, comidas, controles médicos o actividades recreativas. Estos eventos se organizan y presentan a través de un calendario interno que permite una visualización clara y cronológica de las actividades programadas. La interfaz del calendario ha sido diseñada con un enfoque en la usabilidad y accesibilidad, adaptándose visualmente para facilitar su uso por personas mayores, mediante elementos gráficos simples, tipografías legibles y disposición intuitiva.

* **Botón de emergencia (SOS)**

El adulto mayor cuenta con un botón de emergencia accesible desde la pantalla principal de la aplicación, diseñado para ser activado de forma rápida y sencilla en caso de una situación crítica. Al presionarlo, el sistema genera de inmediato una notificación dirigida al familiar o cuidador registrado, informando sobre la activación del botón SOS e indicando el tipo de alerta. Adicionalmente, se adjunta la ubicación geográfica actual del adulto mayor, obtenida mediante el GPS del dispositivo, con el fin de facilitar una respuesta oportuna y localizada por parte de su red de apoyo.

* **Detección de caídas**

El sistema permite registrar alertas relacionadas con posibles caídas del adulto mayor, las cuales pueden ser activadas manualmente o simuladas como parte del prototipo desarrollado en esta versión. Al generarse, estas alertas desencadenan una notificación local inmediata en el dispositivo del adulto mayor y quedan registradas en el sistema, permitiendo su revisión posterior por parte del familiar o cuidador responsable, con el fin de monitorear situaciones de riesgo y actuar de forma preventiva o reactiva según corresponda.

* **Geolocalización y zonas seguras**

El sistema contempla una función de seguimiento basada en la configuración de zonas seguras, definidas previamente. Esta funcionalidad permite establecer un perímetro geográfico dentro del cual el adulto mayor puede desplazarse; en caso de que salga de esta área, se puede generar una alerta. Aunque la función de geolocalización mediante GPS está implementada, su precisión y disponibilidad pueden verse limitadas por los permisos otorgados por el sistema operativo del dispositivo y por la conexión a la red.

* **Notificaciones**

El sistema genera notificaciones locales utilizando Capacitor para alertar a los usuarios sobre eventos programados con al menos 15 minutos de antelación. Estas alertas incluyen información relevante como el título, la descripción y la hora del evento, y se muestran directamente en el dispositivo móvil, incluso sin conexión a internet. Además, en caso de emergencias —como la activación del botón SOS o la detección de inactividad— se emiten notificaciones dirigidas también a los familiares o cuidadores registrados, permitiendo una respuesta rápida ante situaciones críticas. Para mejorar la accesibilidad, especialmente en personas con dificultades visuales, las notificaciones también pueden ser reproducidas mediante voz, a través de la funcionalidad Text-to-Speech integrada en la aplicación.

* **Persistencia local y sincronización**

Parte de la información del sistema se almacena localmente en el dispositivo para permitir un funcionamiento parcial en modo offline, asegurando que ciertas funcionalidades clave estén disponibles incluso sin conexión a internet. Una vez restablecida la conectividad, los datos se sincronizan automáticamente con el backend desarrollado en FastAPI y con la base de datos del sistema, gestionada mediante PostgreSQL como motor principal. Esta base de datos se encuentra alojada en Supabase, una plataforma en la nube que ofrece una capa adicional de servicios como autenticación, gestión de API y almacenamiento seguro, facilitando el desarrollo y despliegue del sistema.

* **Accesibilidad y experiencia adaptada**

TatasApp ha sido diseñada con un enfoque en la accesibilidad, priorizando el uso de tipografías legibles, botones grandes, colores neutros y una distribución visual clara, que facilite la navegación y comprensión de la interfaz. Además, se ha incorporado soporte para lectura por voz mediante la funcionalidad Text-to-Speech, lo que refuerza la usabilidad en personas con dificultades visuales. En conjunto, estos elementos buscan entregar una experiencia intuitiva y adaptada específicamente al perfil del usuario adulto mayor.

## 2.3. Características de los Usuarios

|  |  |  |
| --- | --- | --- |
| **Perfil de Usuario** | **Rol en el Sistema** | **Características Generales** |
| Adulto Mayor | Usuario principal | Persona mayor que utiliza la app para recibir recordatorios, activar alertas SOS, visualizar su calendario y recibir notificaciones habladas. Interfaz simplificada y accesible. |
| Familiar o Cuidador | Usuario secundario con acceso extendido | Persona responsable del seguimiento, configuración de rutinas, gestión de eventos, revisión de alertas y recepción de notificaciones asociadas al adulto mayor. Accede a funcionalidades de apoyo y monitoreo. |

## 2.4. Arquitectura

El sistema está diseñado bajo un modelo cliente-servidor, una arquitectura robusta y ampliamente adoptada que separa las responsabilidades entre los componentes para garantizar escalabilidad, mantenibilidad y un flujo de datos eficiente. Esta estructura se compone de tres capas principales:

* Cliente: La aplicación móvil, desarrollada con Ionic y Angular, funciona como la capa de presentación, encargada de brindar una interfaz intuitiva y receptiva para los usuarios. Desde aquí, los usuarios interactúan con el sistema, realizando solicitudes que son enviadas al servidor para su procesamiento. Adicionalmente, la aplicación integra servicios externos como Google Maps API, utilizado para funcionalidades críticas como geolocalización, visualización de mapas y cálculo de rutas. Esta conexión se realiza directamente desde el cliente (sin pasar por el backend), optimizando el rendimiento y reduciendo la carga sobre el servidor.
* Servidor: El núcleo del sistema es una API REST construida con FastAPI (Python), que actúa como intermediario entre el cliente y la base de datos. Su rol principal es recibir las solicitudes HTTP/HTTPS del frontend, validarlas, ejecutar la lógica de negocio necesaria (como autenticación, procesamiento de datos o reglas de negocio) y generar respuestas estructuradas (generalmente en formato JSON). FastAPI fue elegido por su alto rendimiento, tipado estático integrado y compatibilidad con estándares modernos (OpenAPI, ASGI), lo que facilita el desarrollo ágil y la documentación automática.
* Base de Datos: Los datos del sistema son almacenados y gestionados en PostgreSQL, un sistema de gestión de bases de datos relacional (RDBMS) conocido por su confiabilidad, soporte para transacciones complejas y capacidad de manejar grandes volúmenes de información. PostgreSQL se integra con el servidor mediante conexiones seguras, permitiendo operaciones CRUD (Crear, Leer, Actualizar, Eliminar) de manera eficiente, así como consultas avanzadas con JOINs o índices para optimizar el rendimiento.

El flujo de comunicación comienza cuando el cliente (aplicación Ionic) envía solicitudes —como inicio de sesión, consultas de datos o actualizaciones— al servidor mediante peticiones HTTP/HTTPS, utilizando los métodos RESTful estándar (GET para recuperar datos, POST para crear, PUT/PATCH para actualizar y DELETE para eliminar). El servidor, implementado en FastAPI, recibe estas solicitudes, valida los parámetros (incluyendo autenticación mediante tokens JWT si es requerido) y ejecuta la lógica de negocio correspondiente. Durante este proceso, el servidor interactúa con la base de datos PostgreSQL para persistir o recuperar información, realizando operaciones como consultas SQL complejas o transacciones ACID. Una vez procesada la solicitud, el servidor genera una respuesta estructurada (normalmente en formato JSON) que el cliente recibe, interpreta y muestra al usuario —por ejemplo, un listado de rutas disponibles o un mensaje de confirmación—. Cabe destacar que, para funcionalidades específicas como mapas o geolocalización, la aplicación móvil se comunica directamente con Google Maps API mediante claves de acceso embebidas en el frontend, lo que agiliza el proceso y reduce la carga sobre el backend.

Esta arquitectura ofrece múltiples beneficios: Escalabilidad, ya que cada componente (cliente, servidor o base de datos) puede escalarse de manera independiente —por ejemplo, añadiendo más instancias del servidor en un balanceador de carga bajo picos de demanda—. Seguridad, gracias a la separación de capas que permite implementar controles de acceso rigurosos (como tokens JWT, cifrado de datos y firewalls) y aislar posibles fallos. Además, la flexibilidad inherente al uso de APIs REST simplifica la integración con nuevos servicios externos (como pasarelas de pago) o plataformas futuras (versión web o APIs de terceros), asegurando la sostenibilidad del sistema a largo plazo sin requerir cambios estructurales mayores.

* **Diagrama de Modelo Entidad-Relación (ER)**

El modelo relacional de TatasApp está compuesto por las siguientes entidades:

* Usuario: Almacena información personal (nombres, apellidos, correo, etc.) y se relaciona
* con Direccion mediante direccion\_id. También tiene un campo tipo\_usuario para roles.
* Familiar: Establece relaciones entre usuarios (ejemplo: adulto mayor y su familiar)
* mediante claves foráneas (adulto\_mayor\_id, familiar\_id).
* Evento: Registra actividades creadas por usuarios (usuario\_id), con detalles como título,
* fecha/hora y tipo.
* Alerta: Gestiona alertas generadas por usuarios (usuario\_id), incluyendo ubicación,
* mensaje y estado.
* Dirección: Almacena direcciones físicas con un campo opcional (adicional).

Relaciones:

* Usuario-Direccion: 1 a 1 (un usuario tiene una dirección).
* Usuario-Evento/Alerta: 1 a N (un usuario puede crear múltiples eventos/alertas).
* Familiar-Usuario: N a N (implementada como tabla intermedia para relaciones familiares).

Pantalla de un video juego

El contenido generado por IA puede ser incorrecto.

* **Diagrama de Componentes (Cliente-Servidor)**

Cliente (App móvil) - Arquitectura MVC

* Modelo: Gestiona datos locales (DB local) y estructura de datos.
* Vista: Representado en "Pages" e "Interfaces" (UI).
* Controlador: Coordina lógica mediante "Services" (Alertas, API, DB local).
* Integraciones: Usa Google Maps API y funcionalidades nativas de Android (ejemplo: GPS, notificaciones).

Servidor (API) - Arquitectura en Capas

* Routers: Maneja endpoints y redirección de solicitudes.
* Auth: Gestiona autenticación/autorización.
* Models/Schemas: Define estructuras de datos y validaciones.
* DB: Capa de persistencia (base de datos central).

Diagrama

El contenido generado por IA puede ser incorrecto.

* **Diagrama de Paquetes**

Refuerza la estructura cliente-servidor:

* Cliente (MVC): Paquetes como Vista (interfaz), Controlador (lógica), y Modelo (datos locales).
* Servidor (Capas): Paquetes separados para Rutas, Esquemas, Modelo, y Autorización.

Diagrama

El contenido generado por IA puede ser incorrecto.

## 2.5. Restricciones

* El sistema está diseñado exclusivamente para dispositivos móviles con sistema operativo Android versión 10 o superior; no se garantiza compatibilidad con versiones anteriores ni con otros sistemas operativos como iOS.
* Para el correcto funcionamiento de las funcionalidades de sincronización, geolocalización y envío de alertas, se requiere acceso activo a internet en el dispositivo.
* El almacenamiento y procesamiento de datos se realiza mediante PostgreSQL, un sistema gestor de bases de datos relacional robusto y escalable. Durante la fase de desarrollo, esta base de datos está alojada en Supabase, una plataforma en la nube que ofrece herramientas de administración y conectividad. Esta configuración puede presentar ciertos límites en cuanto a rendimiento, capacidad y escalabilidad, ya que depende de los recursos asignados en el entorno de desarrollo gratuito de Supabase.
* El acceso al sistema está restringido mediante autenticación con correo electrónico y contraseña, y el mantenimiento de la sesión se realiza mediante tokens de acceso seguros almacenados localmente.
* No se contempla, en esta versión, la integración con dispositivos médicos, relojes inteligentes (smartwatches) u otros sensores externos.
* La precisión de la funcionalidad de geolocalización puede verse afectada por los permisos otorgados por el usuario, la configuración del dispositivo, la disponibilidad de red o la cobertura GPS.

## 2.6. Suposiciones y Dependencias

* Se asume que los usuarios utilizarán dispositivos móviles con sistema operativo Android versión 10 o superior, requisito indispensable para garantizar la compatibilidad y funcionamiento correcto de la aplicación.
* Se presupone que los dispositivos de los usuarios disponen de conectividad a internet de manera regular, ya que algunas funcionalidades clave (como sincronización de datos, envío de alertas y geolocalización) requieren acceso a la red.
* Se considera que los usuarios otorgan los permisos necesarios al momento de instalar y utilizar la aplicación, como acceso a la ubicación, notificaciones y almacenamiento local.
* La aplicación depende del correcto funcionamiento de servicios externos como Supabase (para el alojamiento de la base de datos durante la fase de desarrollo) y Google Places API (para el autocompletado de direcciones).
* Se asume que los usuarios conocen los conceptos básicos de uso de una aplicación móvil, o cuentan con el apoyo de un familiar o cuidador en caso de dificultades técnicas.

# 3. Requisitos Específicos

## 3.1 Requisitos comunes de las interfaces

### 3.1.1 Interfaces de usuario

La interfaz de TatasApp está diseñada específicamente para dispositivos móviles con sistema operativo Android (versión 10 o superior). La aplicación contará con una pantalla de acceso mediante correo electrónico y contraseña, seguida de una interfaz principal personalizada según el tipo de usuario (adulto mayor o familiar/cuidador). El diseño prioriza la accesibilidad, empleando botones grandes, colores neutros y tipografías legibles. Además, incorpora soporte para lectura por voz mediante Text-to-Speech. No se utilizarán colores institucionales ni fuentes corporativas, ya que el diseño está enfocado en la usabilidad y simplicidad visual para adultos mayores.

### 3.1.2 Interfaces de hardware

* Dispositivo móvil: El usuario final requiere un teléfono con sistema operativo Android 10 o superior.
* GPS: Necesario para las funciones de geolocalización y zonas seguras.
* Almacenamiento local: Utilizado para guardar el token de autenticación y permitir funnionamiento offline parcial.
* Micrófono y altavoz: Requeridos para la funcionalidad de notificaciones habladas (Text-to-Speech).
* Conexión a internet: Requerida para sincronización con el backend, envío de alertas y validación de credenciales.

### 3.1.3 Interfaces de software

* Sistema operativo: Android 10 o superior.
* Frameworks utilizados: Ionic y Angular para la aplicación móvil; FastAPI para el backend.
* Base de datos: PostgreSQL alojada en Supabase durante la fase de desarrollo.
* API de ubicación: Google Places para autocompletar direcciones y obtener coordenadas.
* Notificaciones: Capacitor Local Notifications para alertas en el dispositivo; Text-to-Speech para lectura por voz.

### 3.1.4 Interfaces de comunicación

* API REST: La app se comunica con el backend mediante solicitudes HTTP utilizando el protocolo REST y datos en formato JSON.
* Base de datos remota: Comunicación con una base de datos PostgreSQL alojada en Supabase, utilizada durante la fase de desarrollo para sincronizar datos entre el dispositivo y la nube.
* API de terceros: Google Places API se utiliza para obtener direcciones sugeridas y coordenadas geográficas.

## 3.2 Requisitos funcionales y no funcionales

| **ID** | **Nombre del requerimiento** | **Tipo de requerimiento** | **Actores relacionados** | **Descripción corta del requerimiento** | **Prioridad** |
| --- | --- | --- | --- | --- | --- |
| F01 | Crear dos tipos de usuarios: familiares y adulto mayor | Funcional | Adulto Mayor, Familiar | Permite crear una cuenta para el adulto mayor y familiares, ingresando datos personales y credenciales. | Alta |
| F02 | Inicio de sesión con validación | Funcional | Adulto Mayor, Familiar | El usuario puede iniciar sesión con correo y contraseña, validando los datos ingresados. | Alta |
| F03 | Integración con Google Calendar | Funcional | Adulto Mayor, Familiar | Permite sincronizar eventos con Google Calendar para gestionar rutinas y recordatorios. | Alta |
| F04 | Uso de Funcionalidades Nativas de Android | Funcional | Adulto Mayor, Familiar | Utiliza componentes como GPS, sensores de movimiento, giroscopio y lista de contactos. | Alta |
| F05 | Diferenciación de Perfiles | Funcional | Adulto Mayor, Familiar | La app muestra diferentes interfaces y funciones según el perfil del usuario. | Media |
| F06 | Asociación de Grupo Familiar | Funcional | Adulto Mayor, Familiar | Permite asociar múltiples familiares a un adulto mayor y crear grupos familiares. | Media |
| F07 | Botón SOS para emergencias | Funcional | Adulto Mayor, Familiar | Al presionar el botón SOS se envía una alerta inmediata al grupo familiar con la ubicación en tiempo real del adulto mayor. | Media |
| F08 | Notificaciones por voz para adulto mayor | Funcional | Adulto Mayor, Familiar | Las notificaciones de eventos son leídas en voz alta automáticamente usando TTS de Google. | Media |
| F09 | Alerta de caída con sensores del dispositivo | Funcional | Adulto Mayor, Familiar | Detecta caídas con sensores del celular y envía alertas a los familiares con la ubicación en tiempo real del adulto mayor. | Media |
| F10 | Creacion edición y eliminación de eventos | Funcional | Adulto Mayor | Permite crear, modificar o eliminar eventos creados desde la app. | Media |
| F11 | Monitoreo de Ubicación y Zona Segura | Funcional | Adulto Mayor, Familiar | Permite localizar al adulto mayor y configurar zonas seguras con alertas al salir de ellas. | Media |
| F12 | Soporte Offline | Funcional | Adulto Mayor | La app debe poder funcionar sin conexión a internet. | Media |
| F13 | Historial de eventos y alertas | Funcional | Adulto Mayor, Familiar | Permitir visualizar un historial con los eventos programados y alertas generadas. | Baja |
| F14 | Gestión de perfil del usuario | Funcional | Adulto Mayor, Familiar | El usuario podrá editar su nombre, datos de contacto y foto desde la app. | Baja |
| F15 | Carga de foto o avatar para el perfil | Funcional | Adulto Mayor, Familiar | El usuario puede subir una foto o avatar personalizado en su perfil. | Baja |
| NF01 | Autenticación persistente | No Funcional | Adulto Mayor, Familiar | El sistema debe mantener la sesión activa sin necesidad de iniciar sesión en cada apertura. | Alta |
| NF02 | Estética accesible para personas mayores | No Funcional | Adulto Mayor, Familiar | La interfaz utiliza tipografía legible, botones grandes y colores contrastados para facilitar el uso. | Alta |
| NF03 | Sincronización de datos entre usuarios | No Funcional | Adulto Mayor, Familiar | Los datos de eventos, alertas y notificaciones deben mantenerse sincronizados con el grupo familiar. | Media |
| NF04 | Seguridad de la información | No Funcional | Adulto Mayor, Familiar | Los datos personales y de salud deben mantenerse protegidos mediante cifrado y buenas prácticas | Baja |
| NF05 | Rendimiento y Compatibilidad en Android | No Funcional | Adulto Mayor, Familiar | La app debe ejecutarse con fluidez en dispositivos Android 10 o superior, incluso con recursos limitados. | Baja |

# 4. Propuesta de Planificación

## 4.1 Descripción general acerca de la Planificación

La planificación del proyecto TatasApp se estructuró considerando los requerimientos establecidos en el contexto académico del Portafolio de Título de la carrera de Analista Programador Computacional. En esta sección se detallan los elementos fundamentales de dicha planificación, incluyendo la definición del equipo de trabajo, la asignación de roles y tareas, así como la calendarización general de actividades representada mediante una carta Gantt.

El desarrollo del sistema se organizó bajo la metodología tradicional en cascada, lo que implicó avanzar de forma secuencial por cada fase del proyecto: análisis, diseño, desarrollo, pruebas y documentación. La distribución de responsabilidades se realizó en función de módulos funcionales, permitiendo que cada integrante asumiera la implementación de componentes completos del sistema.

Además, para asegurar un trabajo colaborativo ordenado y con trazabilidad, el equipo utilizó Git como sistema de control de versiones y GitHub como repositorio remoto, lo que permitió gestionar el código de forma distribuida, mantener registros de cambios y facilitar la integración de los distintos aportes individuales al proyecto.

En los siguientes apartados se detallan los roles asignados, las responsabilidades específicas de cada miembro del equipo y la planificación temporal de las tareas clave que permitieron llevar a cabo el desarrollo de TatasApp.

### 4.1.1 Definición del Equipo de Trabajo

TatasApp fue desarrollado por un equipo de estudiantes de la carrera de Analista Programador Computacional, como parte de su proyecto de título. El desarrollo se estructuró bajo la metodología tradicional en cascada, avanzando ordenadamente a través de las etapas de análisis de requerimientos, diseño, desarrollo, pruebas y documentación.

La asignación de responsabilidades se realizó principalmente en función de las funcionalidades del sistema, más que por áreas técnicas tradicionales. Este enfoque permitió que cada integrante asumiera el desarrollo completo de ciertos módulos, facilitando la comprensión integral del sistema y una colaboración efectiva entre todos los miembros.

| **Nombre** | **Rol funcional** | **Principales responsabilidades** |
| --- | --- | --- |
| **Andrea Pino** | Coordinadora funcional y desarrolladora de interfaces | Encargada del diseño funcional del sistema, creación de formularios de registro e inicio de sesión, gestión de eventos, integración de notificaciones locales y lectura por voz, así como validaciones y experiencia de usuario. |
| **David Guentelicán** | Responsable de API y lógica de sincronización | Desarrollo de la API utilizando FastAPI, gestión de usuarios y eventos en backend, integración con Supabase (plataforma que provee una base de datos PostgreSQL) y validación de credenciales. |
| **Alexander Aguilera** | Integrador de servicios externos y soporte técnico | Responsable de la integración de servicios de terceros como Google Places API, gestión de permisos del dispositivo (como acceso a contactos), y apoyo en pruebas funcionales y de conectividad. |

### 4.1.2 Carta Gantt

La planificación del desarrollo de TatasApp se enmarca dentro del proyecto de portafolio de la carrera de Analista Programador Computacional. De acuerdo con la estructura general establecida por la asignatura, el desarrollo de la aplicación comienza en la segunda fase del proyecto de portafolio, correspondiente a la semana 5 del semestre académico, y se extiende hasta la semana 15.

La carta Gantt que se presenta a continuación detalla las principales actividades realizadas durante este período, abarcando desde el levantamiento de requerimientos hasta la preparación del paquete de entrega. Cada tarea está distribuida temporalmente en función de su complejidad y dependencia, lo que permitió una ejecución organizada y progresiva conforme a la metodología en cascada.

| **Macromódulo** | **Módulo** | **Tareas** | **S 5** | **S 6** | **S 7** | **S 8** | **S 9** | **S 10** | **S 11** | **S 12** | **S 13** | **S 14** | **S 15** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Levantamiento de Requerimientos | x |  |  |  |  |  |  |  |  |  |  |
|  |  | Elaboración de Plan de Trabajo | x |  |  |  |  |  |  |  |  |  |  |
|  |  | Diseño de la Arquitectura | x |  |  |  |  |  |  |  |  |  |  |
| Usuarios | Registro de Usuario | Crear interfaces de Usuario |  | x |  |  |  |  |  |  |  |  |  |
| Inicio de Sesión | Crear interfaces de inicio de sesión |  |  | x |  |  |  |  |  |  |  |  |
| Autenticación persistente | Habilitar SQLite |  |  |  | x |  |  |  |  |  |  |  |
| Familiares | Visualización de contactos registrados en la app | Asignar permisos para poder usar los contactos |  |  | x |  |  |  |  |  |  |  |  |
| Agregar familiares | Crear interfaces para agregar familiares |  |  | x |  |  |  |  |  |  |  |  |
| Visualizar grupo familiar | Crear interfaces para visualizar grupo familiar "diferido" |  |  |  | x |  |  |  |  |  |  |  |
| Eliminar familiares | Crear interfaces para la eliminación de familiares |  |  |  | x |  |  |  |  |  |  |  |
| Eventos | Crear evento | Crear interfaces (sólo un archivo evento.ts) para crear evento |  | x |  |  |  |  |  |  |  |  |  |
| Visualizar eventos | Crear interfaces para visualizar eventos |  |  | x |  |  |  |  |  |  |  |  |
| Recordatorio de eventos | Crear interfaces para los recordatorios de eventos |  |  |  | x |  |  |  |  |  |  |  |
| Alertas | Botón SOS | Crear interfaces (sólo un archivo alerta.ts) para los modelos de tipo de datos de las alertas |  |  |  |  | x | x |  |  |  |  |  |
| Detección de caídas | Crear interfaces para el envío de alertas de caídas |  |  |  |  | x | x |  |  |  |  |  |
| Zona segura | Crear interfaces para el envío de alertas de zona segura |  |  |  |  |  |  | x |  |  |  |  |
| Notificación de alertas a familiares | Crear interfaz para manejar los datos de la ruta GET |  |  |  |  |  |  | x |  |  |  |  |
| Historial | Crear interfaces para definir los tipos de dato que se van a mostrar |  |  |  | x |  |  |  |  |  |  |  |
| Servicios externos | Tecnologías de Android | Instalar capacitor-comunity/contacts y permisos de contactos del teléfono |  |  |  | x |  |  |  |  |  |  |  |
| Tecnologías de Google | API de Google para Maps |  |  |  | x |  |  |  |  |  |  |  |
| Dependencias de Ionic | Instalar dependencia para usar iconos |  |  |  |  |  |  |  | x |  |  |  |
| Configuración | Editar datos de Usuario | Implementar cambio nombre visible, email, telefono, foto de perfil |  |  |  | x |  |  |  |  |  |  |  |
| Permisos para pantallas APP | Validar token y tipo usuario en DB local para rutas restringidas |  |  |  |  | x |  |  |  |  |  |  |
| Estilo - Diseño | Accesibilidad y Usabilidad | Diseño intuitivo |  |  |  |  |  |  | x | x |  |  |  |
|  |  | Pruebas de Funcionalidad |  |  |  |  |  |  |  |  | x | x |  |
|  |  | Corrección de Errores |  |  |  |  |  |  |  |  |  | x |  |
|  |  | Documentación Técnica | x | x | x | x | x | x | x | x | x | x | x |

# 5. Conclusión

El presente documento ha definido de manera estructurada y detallada los requerimientos del sistema TatasApp, una aplicación móvil orientada a mejorar la calidad de vida de adultos mayores y facilitar su acompañamiento por parte de familiares o cuidadores. A través de un enfoque centrado en la accesibilidad, seguridad y usabilidad, se han especificado los objetivos funcionales y técnicos del sistema, considerando sus limitaciones, dependencias y contexto de uso.

La definición del equipo de trabajo, el levantamiento de requerimientos, la descripción de interfaces y la planificación mediante carta Gantt reflejan una organización basada en la metodología en cascada, con una distribución de tareas por funcionalidades. Asimismo, se establecieron criterios técnicos como la compatibilidad con Android 10 o superior, el uso de tecnologías como Ionic, Angular, FastAPI y Supabase, y la integración de servicios como Google Places y Capacitor.

El uso de herramientas colaborativas como Git y GitHub, junto con una clara asignación de responsabilidades entre los miembros del equipo, permitió llevar a cabo un desarrollo ordenado y coherente con los objetivos del proyecto de título. Este informe sienta así las bases necesarias para evaluar, validar y escalar el sistema en futuras etapas de desarrollo o implementación real.