Especificación de requisitos de software

Para

Integración de MSA: TaskHub

Versión 4.2 aprobado

Preparado por Team Software Company

Universidad Tecnológica de Bolívar

7 de junio del 2025

Índice

Historial de revisiones

Nombre	**Fecha**	**Motivo de los cambios**	**Versión**
Inicio del proyecto	17-03-25	Inicializar la documentación del proyecto	0.1
MVP	17-03-25	Documentación del MVP del proyecto	0.2
RF y RNF	21-03-25	Ajuste en las descripciones y clasificación de los RF y RNF	1.1
Cambio de Nombre del Proyecto	21-03-25	Simplificar el nombre del proyecto	1.2
Versión 2.0	22-03-25	Integración de arquitectura detalladas y requisitos funcionales y no funcionales expandidos	2.0
Versión 3.0	23-03-25	Integración completa de arquitectura, patrones, interfaz, MVP y relaciones	3.0
Versión 4.0	24-04-25	Actualización de la arquitectura de microservicios con detalles de implementación	4.0
Versión 4.1	25-04-25	Mejora de los requisitos de seguridad y autenticación	4.1
Versión 4.2	26-04-25	Detalle de la integración con herramientas externas	4.2

1. Introducción

TaskHub es una plataforma de colaboración unificada diseñada para integrar las herramientas de gestión de proyectos, creación de documentos y comunicación de una empresa. El objetivo principal es abordar las limitaciones de las herramientas existentes que operan de forma independiente, lo que resulta en datos aislados, falta de personalización e integración limitada con aplicaciones externas.

La plataforma se ha desarrollado utilizando una arquitectura de microservicios moderna, implementada con Python, FastAPI, SQLAlchemy y Supabase, lo que permite una escalabilidad, mantenibilidad y flexibilidad óptimas. El frontend está desarrollado con Flutter Web, proporcionando una interfaz de usuario responsiva y consistente en diferentes dispositivos.

En esta versión 4.2, se ha puesto especial énfasis en la integración con herramientas externas, detallando la implementación del servicio de integración con GitHub, Google Drive, Microsoft Outlook y Slack, así como el sistema de extensibilidad y plugins.

2. Descripción general

La empresa ofrece una suite de herramientas de colaboración que incluye gestión de proyectos, creación de documentos y comunicación en tiempo real. Sin embargo, estas herramientas funcionan de manera independiente, lo que genera datos dispersos y dificulta la integración de funcionalidades. Este proyecto busca desarrollar una plataforma unificada basada en microservicios que permita la integración de datos y funcionalidades, mejorando la experiencia del usuario y la capacidad de la empresa para ofrecer soluciones personalizadas y escalables.

2.1 Perspectiva del Producto

TaskHub se concibe como una solución centralizada que permite a los usuarios gestionar proyectos, colaborar en documentos y comunicarse en tiempo real dentro de una única plataforma. El sistema se basa en una arquitectura de microservicios para garantizar la escalabilidad, la mantenibilidad y la flexibilidad.

2.2 Características del producto

TaskHub ofrece las siguientes características clave:

- **Arquitectura Modular Basada en Microservicios**: El sistema está compuesto por servicios independientes que se comunican a través de APIs, lo que permite el desarrollo, la implementación y el escalamiento independientes.
- **Personalización de Flujos de Trabajo**: La plataforma se puede adaptar para satisfacer las necesidades específicas de diferentes equipos y proyectos.
- **Gestión de Proyectos y Documentos**: Los usuarios pueden crear, gestionar y colaborar en proyectos y documentos dentro de la plataforma.

- **Integración con Herramientas Externas**: TaskHub se integra con aplicaciones externas como GitHub, Google Drive, Microsoft Outlook y Slack para optimizar los flujos de trabajo.
- **Seguridad y Control de Accesos**: El sistema proporciona mecanismos sólidos para la autenticación, la autorización y el control de accesos para proteger los datos confidenciales.
- **Análisis y Optimización de Procesos**: TaskHub permite el análisis de datos y la optimización de los flujos de trabajo del proyecto.

2.3 Clases de usuarios y características

- **Administrador**: Acceso total a la configuración y monitoreo del sistema. Puede gestionar usuarios, roles y permisos en toda la plataforma.
- **Owner (Propietario)**: Acceso completo a los proyectos y recursos que posee. Puede crear, modificar y eliminar proyectos, así como gestionar los permisos de los miembros del proyecto.
- **Member (Miembro)**: Acceso limitado basado en los permisos del proyecto. Puede editar documentos y participar en proyectos según los permisos asignados.
- **Invitado**: Visualización limitada, sin capacidad de edición. Puede ver proyectos y documentos específicos a los que ha sido invitado.

2.4 Limitaciones de diseño y aplicación

- **Frontend único con Flutter Web**: La interfaz de usuario se desarrolla exclusivamente con Flutter Web para garantizar la consistencia en diferentes plataformas.
- **Comunicación mediante REST/JSON y WebSockets**: Los servicios se comunican a través de APIs REST para operaciones síncronas y WebSockets para comunicación en tiempo real.
- **Arquitectura desacoplada**: Los microservicios deben estar completamente desacoplados para permitir el desarrollo, la implementación y el escalamiento independientes.
- **Patrones de diseño requeridos**: Se utilizan patrones específicos como Command, Observer, Factory Method, Adapter, Decorator, Facade y Circuit Breaker para garantizar un diseño robusto y mantenible.

2.5 Documentación del Usuario

TaskHub proporciona la siguiente documentación para los usuarios:

- **Manual de usuario digital**: Documentación completa accesible desde la plataforma.
- **Guías en línea**: Tutoriales paso a paso para funcionalidades específicas.

- **Tutoriales interactivos**: Guías interactivas dentro de la plataforma para ayudar a los nuevos usuarios.
- **Documentación de API**: Documentación completa de la API utilizando OpenAPI 3.0, accesible a través de Swagger UI.

2.6 Supuestos y dependencias

- **Uso de navegadores modernos**: Se asume que los usuarios utilizarán navegadores web modernos compatibles con las tecnologías web más recientes.
- **Conectividad constante**: Se requiere una conexión a Internet estable para la sincronización en tiempo real.
- **Dependencia de APIs externas**: La integración con servicios externos depende de la disponibilidad y estabilidad de sus APIs.
- **Supabase**: Se utiliza Supabase como plataforma de backend para autenticación, base de datos y almacenamiento.

3. Problema Por Resolver

La empresa enfrenta varios desafíos debido a la falta de integración entre sus herramientas:

- **Datos Aislados**: La información sobre proyectos, documentos y tareas está dispersa en diferentes sistemas, lo que dificulta el acceso y la toma de decisiones en tiempo real.
- **Falta de Personalización**: Los usuarios no pueden adaptar las herramientas a sus flujos de trabajo específicos, lo que reduce la eficiencia.
- **Integración Limitada**: No hay una forma sencilla de conectar las herramientas con aplicaciones externas, como GitHub, Google Drive, Microsoft Outlook y Slack.
- **Falta de un Almacenamiento Centralizado**: No se cuenta con un sistema que unifique la información y permita realizar análisis para mejorar la eficiencia operativa.
- **Escalabilidad y Mantenimiento**: Las soluciones monolíticas actuales son difíciles de escalar y mantener, lo que limita la capacidad de la empresa para crecer y adaptarse a nuevas necesidades.
- **Seguridad y Control de Accesos**: Los sistemas actuales carecen de mecanismos robustos de seguridad y control de accesos, lo que puede comprometer la confidencialidad e integridad de los datos.

4. Arquitectura de Microservicios

4.1 Descripción general de la arquitectura

TaskHub sigue una arquitectura de microservicios, donde la aplicación se estructura como un conjunto de servicios pequeños e independientes. Cada microservicio representa una capacidad empresarial específica y se comunica con otros servicios a través de APIs bien definidas.

Esta arquitectura proporciona varias ventajas:

- **Escalabilidad**: Cada microservicio puede ser escalado independientemente según sus necesidades específicas.
- **Resiliencia**: Un fallo en un servicio no afecta al funcionamiento global del sistema gracias a patrones como Circuit Breaker.
- **Flexibilidad**: Los servicios pueden ser desarrollados, desplegados y escalados de forma independiente.
- **Evolución Tecnológica**: Diferentes servicios pueden adoptar nuevas tecnologías sin afectar a otros.
- **Organización de Equipos**: Los equipos pueden trabajar en diferentes servicios en paralelo.

4.2 Componentes del sistema

El sistema TaskHub consta de los siguientes microservicios y componentes:

4.2.1 API Gateway

El API Gateway actúa como un punto de entrada único para todas las solicitudes de los clientes. Es responsable del enrutamiento de solicitudes, la autenticación y otras funciones transversales.

Funcionalidades principales:

- Enrutamiento de solicitudes a los microservicios apropiados
- Autenticación y validación de tokens
- Implementación de patrones Circuit Breaker para prevenir fallos en cascada
- Manejo de errores y respuestas unificadas
- Logging y monitoreo centralizado

4.2.2 Auth Service

El servicio de autenticación gestiona la autenticación y autorización de usuarios. Proporciona funcionalidades como el registro de usuarios, el inicio de sesión y la gestión de tokens.

Funcionalidades principales:

- Registro de usuarios
- Autenticación de usuarios
- Gestión de tokens JWT
- Control de acceso basado en roles
- Integración con Supabase Auth

4.2.3 Project Service

El servicio de gestión de proyectos gestiona el ciclo de vida de los proyectos. Permite a los usuarios crear, recuperar, actualizar y eliminar proyectos, así como gestionar tareas y flujos de trabajo.

Funcionalidades principales:

- CRUD de proyectos
- Gestión de tareas y actividades
- Operaciones de deshacer/rehacer mediante el patrón Command
- Seguimiento de actividades y cambios
- Asignación de recursos y permisos

4.2.4 Document Service

El servicio de gestión de documentos permite a los usuarios crear, almacenar, recuperar y colaborar en documentos.

Funcionalidades principales:

- Creación y gestión de documentos
- Edición colaborativa en tiempo real
- Control de versiones
- Gestión de permisos de documentos
- Diferentes tipos de documentos mediante el patrón Factory Method

4.2.5 Notification Service

El servicio de notificación gestiona el envío de notificaciones a los usuarios a través de diferentes canales, como correo electrónico, chat y notificaciones push.

Funcionalidades principales:

- Envío de notificaciones en tiempo real
- Notificaciones tipo commit para cambios en proyectos y documentos
- Diferentes canales de notificación (in-app, email, push, SMS)
- Implementación del patrón Observer para la entrega de notificaciones

4.2.6 External Tools Service

El servicio de herramientas externas se encarga de la integración con servicios de terceros como GitHub, Google Drive, Microsoft Outlook y Slack.

Funcionalidades principales:

- Integración con servicios externos
- Sincronización de datos entre TaskHub y herramientas externas
- Implementación del patrón Adapter para diferentes integraciones
- Autenticación OAuth con servicios externos

4.3 Comunicación entre servicios

Los microservicios se comunican entre sí mediante una combinación de protocolos síncronos y asíncronos:

4.3.1 REST/JSON

Se utiliza para la comunicación síncrona basada en solicitudes-respuestas entre los servicios. Todas las APIs siguen principios RESTful y están documentadas con OpenAPI 3.0.

Características:

- APIs versionadas (ej. /v1/projects)
- Formato de respuesta JSON estandarizado
- Manejo de errores consistente
- Autenticación mediante JWT

4.3.2 WebSockets

Se utiliza para la comunicación en tiempo real, como la edición colaborativa de documentos y las notificaciones push.

Características:

- Comunicación bidireccional en tiempo real
- Notificaciones instantáneas
- Edición colaborativa de documentos
- Actualizaciones de estado en tiempo real

4.3.3 RabbitMQ

Se utiliza para la comunicación asíncrona entre servicios, implementando patrones como publicación/suscripción para eventos.

Características:

- Comunicación asíncrona
- Colas de mensajes persistentes
- Patrón publicación/suscripción
- Garantía de entrega de mensajes

4.4 Tecnologías utilizadas

Las siguientes tecnologías se utilizan para desarrollar TaskHub:

4.4.1 Frontend

- **Flutter Web (Dart)**: Framework para el desarrollo de la interfaz de usuario web, proporcionando una experiencia consistente en diferentes dispositivos.
- Diseño responsivo para escritorio (1024px) y tabletas (768px)
- Lazy loading para rutas no esenciales
- WebSockets para edición colaborativa y notificaciones en tiempo real

4.4.2 Backend

- **Python**: Lenguaje principal para el desarrollo de microservicios.
- **FastAPI**: Framework web moderno y de alto rendimiento para la creación de APIs con Python.
- **SQLAlchemy**: ORM (Object-Relational Mapping) para interactuar con bases de datos relacionales.
- **Supabase**: Plataforma de backend como servicio que proporciona autenticación, base de datos y almacenamiento.

4.4.3 Base de datos

- **PostgreSQL**: Sistema de gestión de bases de datos relacional para datos transaccionales.
- **MongoDB**: Base de datos NoSQL para documentos y datos no estructurados.

4.4.4 Caché

- **Redis**: Almacén de estructura de datos en memoria utilizado como caché y broker de mensajes.

4.4.5 Infraestructura

- **Docker**: Plataforma de contenedores para empaquetar, distribuir y ejecutar aplicaciones.
- **Kubernetes**: Sistema de orquestación de contenedores para automatizar el despliegue, el escalado y la gestión de aplicaciones en contenedores.

5. Diseño de los microservicios

Esta sección detalla el diseño e implementación de cada microservicio en TaskHub, incluyendo sus funcionalidades, patrones de diseño, APIs y especificaciones técnicas.

5.1 API Gateway

El API Gateway actúa como un punto de entrada único para todas las solicitudes de los clientes, implementando el patrón Facade para simplificar la interacción con el sistema.

5.1.1 Funcionalidades

- **Enrutamiento de solicitudes**: Dirige las solicitudes al microservicio apropiado basándose en la ruta y el método HTTP.
- **Autenticación**: Valida los tokens JWT y asegura que los usuarios estén autenticados antes de acceder a los recursos protegidos.
- **Validación de tokens**: Verifica la validez y la expiración de los tokens JWT.
- **Manejo de fallos**: Implementa el patrón Circuit Breaker para prevenir fallos en cascada cuando un servicio no está disponible.
- **Logging y monitoreo**: Registra todas las solicitudes y respuestas para análisis y depuración.

5.1.2 Implementación

Listing 1: Ejemplo de implementación del API Gateway con FastAPI

```
from fastapi import FastAPI, Depends, HTTPException
from fastapi.middleware.cors import CORSMiddleware
from .middleware.auth_middleware import verify_token
from .middleware.circuit_breaker import CircuitBreaker
from .utils.service_registry import ServiceRegistry

app = FastAPI(title="TaskHub API Gateway")

# Configuraci n de CORS
app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

# Registro de servicios
service_registry = ServiceRegistry()
service_registry.register("auth", "http://auth-service:8001")
service_registry.register("project", "http://project-service:8002")
service_registry.register("document", "http://document-service:8003")
service_registry.register("notification", "http://notification-service
    :8004")
service_registry.register("external-tools", "http://external-tools-
    service:8005")

# Circuit breakers para cada servicio
auth_cb = CircuitBreaker(name="auth", failure_threshold=5,
    recovery_timeout=30)
project_cb = CircuitBreaker(name="project", failure_threshold=5,
    recovery_timeout=30)

@app.get("/")
async def root():
    return {"message": "Welcome to TaskHub API Gateway"}

# Rutas para el servicio de autenticaci n
@app.get("/v1/auth/{path:path}", dependencies=[Depends(auth_cb)])
async def auth_proxy(path: str):
    # L gica de proxy para el servicio de autenticaci n
    pass

# Rutas para el servicio de proyectos (protegidas)
@app.get("/v1/projects/{path:path}",
         dependencies=[Depends(verify_token), Depends(project_cb)])
async def project_proxy(path: str):
    # L gica de proxy para el servicio de proyectos
    pass
```

5.1.3 Patrones de diseño

- **Facade**: El API Gateway actúa como una fachada que simplifica la interacción con el sistema complejo de microservicios.
- **Circuit Breaker**: Previene fallos en cascada cuando un servicio no está disponible, proporcionando respuestas alternativas o errores controlados.

5.1.4 API

- **Especificación**: OpenAPI 3.0
- **Documentación**: Swagger UI accesible en `/docs`
- **Rutas principales**:
  - `/v1/auth/*`: Rutas para el servicio de autenticación
  - `/v1/projects/*`: Rutas para el servicio de proyectos
  - `/v1/documents/*`: Rutas para el servicio de documentos
  - `/v1/notifications/*`: Rutas para el servicio de notificaciones
  - `/v1/external-tools/*`: Rutas para el servicio de herramientas externas

5.2 Auth Service

El servicio de autenticación gestiona la autenticación y autorización de usuarios, implementando patrones como Singleton para la conexión a la base de datos y Strategy para diferentes métodos de autenticación.

5.2.1 Funcionalidades

- **Registro de usuarios**: Permite a los usuarios crear nuevas cuentas con información básica.
- **Inicio de sesión de usuarios**: Autentica a los usuarios y genera tokens JWT.
- **Gestión de tokens**: Genera, valida y revoca tokens JWT.
- **Control de acceso basado en roles**: Define y gestiona roles y permisos para diferentes tipos de usuarios.
- **Integración con Supabase Auth**: Utiliza Supabase para la autenticación y autorización.
- **Autenticación multifactor**: Proporciona opciones para autenticación de dos factores (2FA).
- **Gestión de sesiones**: Controla las sesiones activas de los usuarios.
- **Recuperación de contraseñas**: Permite a los usuarios restablecer sus contraseñas de forma segura.
- **Bloqueo de cuentas**: Bloquea temporalmente las cuentas después de múltiples intentos fallidos de inicio de sesión.

5.2.2 Patrones de diseño

- **Singleton**: Utilizado para la conexión a la base de datos y Supabase, asegurando una única instancia compartida.
- **Strategy**: Implementado para diferentes métodos de autenticación (email/password, OAuth, 2FA).
- **Factory**: Utilizado para crear diferentes tipos de tokens (acceso, actualización, restablecimiento).
- **Observer**: Implementado para notificar eventos de seguridad (inicio de sesión, bloqueo de cuenta, etc.).

5.2.3 API

- **Especificación**: OpenAPI 3.0
- **Documentación**: Swagger UI accesible en `/docs`
- **Rutas principales**:
  - `POST /v1/auth/register`: Registro de nuevos usuarios
  - `POST /v1/auth/login`: Inicio de sesión y generación de tokens
  - `POST /v1/auth/refresh`: Renovación de tokens
  - `POST /v1/auth/logout`: Cierre de sesión y revocación de tokens
  - `GET /v1/auth/me`: Obtención de información del usuario actual
  - `POST /v1/auth/password/reset`: Solicitud de restablecimiento de contraseña
  - `POST /v1/auth/password/update`: Actualización de contraseña
  - `POST /v1/auth/2fa/enable`: Activación de autenticación de dos factores
  - `POST /v1/auth/2fa/verify`: Verificación de código de autenticación de dos factores
  - `POST /v1/auth/2fa/disable`: Desactivación de autenticación de dos factores
  - `GET /v1/auth/sessions`: Obtención de sesiones activas
  - `DELETE /v1/auth/sessions/{session_id}`: Revocación de sesión específica

5.3 Project Service

El servicio de gestión de proyectos gestiona el ciclo de vida de los proyectos, implementando el patrón Command para operaciones de deshacer/rehacer y Observer para notificaciones.

5.3.1 Funcionalidades

- **CRUD de proyectos**: Permite a los usuarios crear, recuperar, actualizar y eliminar proyectos.
- **Gestión de tareas**: Permite a los usuarios crear, asignar y seguir el progreso de tareas dentro de los proyectos.
- **Operaciones de deshacer/rehacer**: Implementa el patrón Command para permitir a los usuarios deshacer y rehacer operaciones.
- **Seguimiento de actividades**: Registra todas las actividades y cambios en los proyectos.
- **Asignación de recursos**: Permite a los usuarios asignar recursos a proyectos y tareas.
- **Gestión de permisos**: Controla quién puede acceder y modificar los proyectos y tareas.

5.3.2 Patrones de diseño

- **Command**: Utilizado para encapsular operaciones como objetos, permitiendo deshacer/rehacer y registrar comandos.
- **Observer**: Implementado para notificar a otros servicios sobre cambios en proyectos y tareas.
- **Repository**: Utilizado para abstraer el acceso a datos y proporcionar una interfaz unificada.
- **Factory Method**: Implementado para crear diferentes tipos de proyectos y tareas.

5.3.3 API

- **Especificación**: OpenAPI 3.0
- **Documentación**: Swagger UI accesible en `/docs`
- **Rutas principales**:
  - `POST /v1/projects`: Creación de proyectos
  - `GET /v1/projects`: Obtención de proyectos
  - `GET /v1/projects/{project_id}`: Obtención de un proyecto específico
  - `PUT /v1/projects/{project_id}`: Actualización de un proyecto
  - `DELETE /v1/projects/{project_id}`: Eliminación de un proyecto
  - `POST /v1/projects/{project_id}/undo`: Deshacer última operación
  - `POST /v1/projects/{project_id}/redo`: Rehacer última operación deshecha
  - `POST /v1/projects/{project_id}/tasks`: Creación de tareas
  - `GET /v1/projects/{project_id}/tasks`: Obtención de tareas

- `GET /v1/projects/{project_id}/tasks/{task_id}`: Obtención de una tarea específica
- `PUT /v1/projects/{project_id}/tasks/{task_id}`: Actualización de una tarea
- `DELETE /v1/projects/{project_id}/tasks/{task_id}`: Eliminación de una tarea

5.4 Document Service

El servicio de gestión de documentos permite a los usuarios crear, almacenar, recuperar y colaborar en documentos, implementando patrones como Factory Method para la creación de diferentes tipos de documentos y Decorator para funcionalidades adicionales.

5.4.1 Funcionalidades

- **Creación y gestión de documentos**: Permite a los usuarios crear, recuperar, actualizar y eliminar documentos.
- **Edición colaborativa en tiempo real**: Permite a múltiples usuarios editar un documento simultáneamente.
- **Control de versiones**: Mantiene un historial de cambios en los documentos.
- **Gestión de permisos**: Controla quién puede acceder y modificar los documentos.
- **Diferentes tipos de documentos**: Soporta diferentes tipos de documentos (texto, hoja de cálculo, presentación) mediante el patrón Factory Method.
- **Funcionalidades adicionales**: Permite añadir funcionalidades adicionales a los documentos (comentarios, etiquetas, etc.) mediante el patrón Decorator.

5.4.2 Patrones de diseño

- **Factory Method**: Utilizado para crear diferentes tipos de documentos (texto, hoja de cálculo, presentación).
- **Decorator**: Implementado para añadir funcionalidades adicionales a los documentos (comentarios, etiquetas, etc.).
- **Observer**: Utilizado para notificar a los clientes sobre cambios en los documentos en tiempo real.
- **Memento**: Implementado para el control de versiones de documentos.

5.4.3 API

- **Especificación**: OpenAPI 3.0
- **Documentación**: Swagger UI accesible en `/docs`
- **Rutas principales**:

- `POST /v1/documents`: Creación de documentos
- `GET /v1/documents`: Obtención de documentos
- `GET /v1/documents/{document_id}`: Obtención de un documento específico
- `PUT /v1/documents/{document_id}`: Actualización de un documento
- `DELETE /v1/documents/{document_id}`: Eliminación de un documento
- `GET /v1/documents/{document_id}/versions`: Obtención de versiones de un documento
- `GET /v1/documents/{document_id}/versions/{version_id}`: Obtención de una versión específica
- `POST /v1/documents/{document_id}/versions/{version_id}/restore`: Restauración de una versión
- `POST /v1/documents/{document_id}/comments`: Añadir comentario a un documento
- `POST /v1/documents/{document_id}/tags`: Añadir etiqueta a un documento
- `WebSocket /v1/documents/{document_id}/ws`: Endpoint para edición colaborativa en tiempo real

5.5 Notification Service

El servicio de notificación gestiona el envío de notificaciones a los usuarios a través de diferentes canales, implementando el patrón Observer para la entrega de notificaciones.

5.5.1 Funcionalidades

- **Envío de notificaciones en tiempo real**: Permite enviar notificaciones a los usuarios en tiempo real.
- **Notificaciones tipo commit**: Notificaciones detalladas sobre cambios en proyectos y documentos, similares a los commits en Git.
- **Diferentes canales de notificación**: Soporta diferentes canales de notificación (in-app, email, push, SMS).
- **Preferencias de notificación**: Permite a los usuarios configurar sus preferencias de notificación.
- **Historial de notificaciones**: Mantiene un historial de notificaciones para cada usuario.

5.5.2 Patrones de diseño

- **Observer**: Implementado para notificar a los usuarios a través de diferentes canales.
- **Strategy**: Utilizado para diferentes estrategias de entrega de notificaciones.
- **Template Method**: Implementado para definir el esqueleto de un algoritmo de notificación.
- **Chain of Responsibility**: Utilizado para procesar notificaciones a través de diferentes manejadores.

5.5.3 API

- **Especificación**: OpenAPI 3.0
- **Documentación**: Swagger UI accesible en `/docs`
- **Rutas principales**:
  - `POST /v1/notifications`: Creación de notificaciones
  - `GET /v1/notifications`: Obtención de notificaciones
  - `GET /v1/notifications/{notification_id}`: Obtención de una notificación específica
  - `PUT /v1/notifications/{notification_id}`: Actualización de una notificación
  - `DELETE /v1/notifications/{notification_id}`: Eliminación de una notificación
  - `POST /v1/notifications/mark-all-read`: Marcar todas las notificaciones como leídas
  - `GET /v1/notifications/preferences`: Obtención de preferencias de notificación
  - `PUT /v1/notifications/preferences`: Actualización de preferencias de notificación
  - `WebSocket /v1/notifications/ws`: Endpoint para notificaciones en tiempo real

5.6 External Tools Service

El servicio de herramientas externas se encarga de la integración con servicios de terceros como GitHub, Google Drive, Microsoft Outlook y Slack, implementando el patrón Adapter para diferentes integraciones.

5.6.1 Funcionalidades

- **Integración con servicios externos**: Permite a los usuarios conectar TaskHub con servicios externos.
- **Sincronización de datos**: Sincroniza datos entre TaskHub y herramientas externas.
- **Autenticación OAuth**: Implementa autenticación OAuth para servicios externos.
- **Webhooks**: Recibe y procesa webhooks de servicios externos.
- **Extensibilidad**: Permite añadir nuevas integraciones mediante un sistema de plugins.

5.6.2 Patrones de diseño

- **Adapter**: Implementado para adaptar diferentes APIs externas a una interfaz común.
- **Facade**: Utilizado para proporcionar una interfaz unificada para diferentes servicios externos.
- **Strategy**: Implementado para diferentes estrategias de sincronización de datos.
- **Plugin**: Utilizado para permitir la extensibilidad del sistema mediante plugins.

5.6.3 API

- **Especificación**: OpenAPI 3.0
- **Documentación**: Swagger UI accesible en `/docs`
- **Rutas principales**:
  - `POST /v1/integrations`: Creación de integraciones
  - `GET /v1/integrations`: Obtención de integraciones
  - `GET /v1/integrations/{integration_id}`: Obtención de una integración específica
  - `PUT /v1/integrations/{integration_id}`: Actualización de una integración
  - `DELETE /v1/integrations/{integration_id}`: Eliminación de una integración
  - `POST /v1/integrations/{integration_type}/oauth`: Procesamiento de callbacks de OAuth
  - `POST /v1/integrations/{integration_type}/webhook`: Procesamiento de webhooks
  - `GET /v1/plugins`: Obtención de plugins disponibles
  - `POST /v1/plugins/{plugin_id}/install`: Instalación de un plugin
  - `POST /v1/plugins/{plugin_id}/uninstall`: Desinstalación de un plugin

6. Seguridad y Autenticación

Esta sección detalla la arquitectura de seguridad de TaskHub, incluyendo los mecanismos de autenticación, autorización, gestión de tokens, roles y permisos, protección de datos, y auditoría y monitoreo.

6.1 Arquitectura de Seguridad

La arquitectura de seguridad de TaskHub se basa en un enfoque de defensa en profundidad, implementando múltiples capas de seguridad para proteger los datos y recursos del sistema.

6.1.1 Capas de Seguridad

1. **Seguridad de Red**:
   - Todo el tráfico se cifra mediante HTTPS con TLS 1.3
   - Implementación de firewalls y listas de control de acceso
   - Protección contra ataques DDoS
   - Segmentación de red para aislar componentes críticos
2. **Seguridad de Aplicación**:
   - Validación de entrada para prevenir inyecciones
   - Protección contra ataques XSS y CSRF
   - Implementación de Content Security Policy (CSP)
   - Cabeceras de seguridad HTTP (HSTS, X-Content-Type-Options, etc.)
3. **Seguridad de Autenticación**:
   - Autenticación basada en tokens JWT con RSA-256
   - Soporte para autenticación multifactor (2FA)
   - Bloqueo de cuentas después de múltiples intentos fallidos
   - Gestión segura de sesiones
4. **Seguridad de Autorización**:
   - Control de acceso basado en roles (RBAC)
   - Permisos granulares a nivel de recurso
   - Principio de privilegio mínimo
   - Separación de responsabilidades
5. **Seguridad de Datos**:
   - Cifrado de datos en reposo y en tránsito
   - Tokenización de datos sensibles
   - Anonimización de datos para análisis
   - Copias de seguridad cifradas

6.2 Autenticación y Autorización

TaskHub implementa un sistema robusto de autenticación y autorización para garantizar que solo los usuarios autorizados puedan acceder a los recursos del sistema.

6.2.1 Métodos de Autenticación

- **Email y contraseña**: Método tradicional con hash seguro de contraseñas usando bcrypt
- **OAuth 2.0**: Integración con proveedores externos como Google, GitHub y Microsoft
- **Autenticación multifactor (2FA)**: Soporte para TOTP, SMS y email
- **Tokens JWT**: Firmados con RSA-256 para garantizar la integridad

6.2.2 Sistema de Autorización

- **Control de acceso basado en roles (RBAC)**: Asignación de permisos a través de roles
- **Permisos granulares**: Control a nivel de recurso individual
- **Políticas de acceso**: Reglas configurables para diferentes escenarios
- **Separación de responsabilidades**: Prevención de conflictos de interés

6.3 Gestión de Tokens

TaskHub utiliza tokens JWT para la autenticación y autorización, proporcionando un mecanismo seguro y escalable para gestionar el acceso de los usuarios.

6.3.1 Tipos de Tokens

- **Access Tokens**: Para acceso a recursos protegidos (corta duración: 15 minutos)
- **Refresh Tokens**: Para renovar access tokens (larga duración: 7 días)
- **ID Tokens**: Para información de identidad del usuario
- **Reset Tokens**: Para restablecimiento de contraseñas (corta duración: 1 hora)

6.3.2 Características de Seguridad

- **Firma con RSA-256**: Garantiza la integridad y autenticidad de los tokens
- **Expiración configurable**: Tiempos de vida apropiados para cada tipo de token
- **Revocación de tokens**: Capacidad de invalidar tokens comprometidos
- **Rotación de claves**: Cambio periódico de claves de firma
- **Validación de claims**: Verificación de audiencia, emisor y expiración

6.4 Roles y Permisos

TaskHub implementa un sistema de control de acceso basado en roles (RBAC) para gestionar los permisos de los usuarios en el sistema.

6.4.1 Roles Predefinidos

- **Administrador**: Acceso total al sistema, gestión de usuarios y configuración
- **Owner**: Control total sobre sus proyectos y recursos
- **Member**: Acceso limitado según permisos asignados
- **Invitado**: Solo visualización de recursos específicos

6.4.2 Permisos Granulares

- **Lectura**: Visualización de recursos
- **Escritura**: Modificación de recursos existentes
- **Creación**: Creación de nuevos recursos
- **Eliminación**: Eliminación de recursos
- **Administración**: Gestión de permisos y configuración

6.5 Protección de Datos

TaskHub implementa múltiples capas de protección de datos para garantizar la confidencialidad, integridad y disponibilidad de la información.

6.5.1 Cifrado

- **En tránsito**: TLS 1.3 para todas las comunicaciones
- **En reposo**: AES-256 para datos sensibles en bases de datos
- **Claves**: Gestión segura de claves con rotación periódica
- **Algoritmos**: Uso de algoritmos criptográficos aprobados

6.5.2 Privacidad

- **Minimización de datos**: Recopilación solo de datos necesarios
- **Anonimización**: Para análisis y métricas
- **Derecho al olvido**: Capacidad de eliminar datos de usuarios
- **Consentimiento**: Gestión de consentimientos para diferentes usos

6.6 Auditoría y Monitoreo

TaskHub implementa un sistema completo de auditoría y monitoreo para detectar y responder a incidentes de seguridad.

6.6.1 Capacidades de Auditoría

- **Registro de acciones**: Todas las acciones de usuarios se registran
- **Cambios de configuración**: Seguimiento de modificaciones del sistema
- **Acceso a datos**: Registro de acceso a información sensible
- **Eventos de seguridad**: Detección de comportamientos anómalos

6.6.2 Monitoreo en Tiempo Real

- **Alertas automáticas**: Para eventos críticos de seguridad
- **Dashboards**: Visualización de métricas de seguridad
- **Análisis de comportamiento**: Detección de patrones anómalos
- **Respuesta a incidentes**: Procedimientos automatizados de respuesta

7. Integración con Herramientas Externas

Esta sección detalla la arquitectura de integración de TaskHub con herramientas externas, incluyendo GitHub, Google Drive, Microsoft Outlook y Slack, así como el sistema de extensibilidad y plugins.

7.1 Arquitectura de Integración

La arquitectura de integración de TaskHub se basa en un enfoque modular y extensible, utilizando el patrón Adapter para proporcionar una interfaz común para diferentes servicios externos.

7.1.1 Componentes de Integración

1. **External Tools Service**: Microservicio dedicado a la gestión de integraciones
2. **Adaptadores de Integración**: Componentes específicos para cada servicio externo
3. **Gestor de Plugins**: Permite añadir nuevas integraciones mediante un sistema de plugins

7.2 GitHub Integration

La integración con GitHub permite a los usuarios conectar sus repositorios de GitHub con TaskHub, sincronizando issues, pull requests y commits.

7.2.1 Funcionalidades

- **Autenticación OAuth**: Conexión segura con GitHub mediante OAuth 2.0
- **Sincronización de Issues**: Creación y actualización de tareas en TaskHub a partir de issues de GitHub
- **Sincronización de Pull Requests**: Seguimiento del estado de pull requests en TaskHub
- **Sincronización de Commits**: Visualización de commits relacionados con tareas y proyectos
- **Webhooks**: Recepción de webhooks de GitHub para actualizaciones en tiempo real
- **Configuración Personalizada**: Selección de repositorios y eventos a sincronizar

7.3 Google Drive Integration

La integración con Google Drive permite a los usuarios conectar sus cuentas de Google Drive con TaskHub, sincronizando archivos y carpetas.

7.3.1 Funcionalidades

- **Autenticación OAuth**: Conexión segura con Google Drive mediante OAuth 2.0
- **Sincronización de Archivos**: Vinculación de archivos de Google Drive a tareas y proyectos
- **Sincronización de Carpetas**: Vinculación de carpetas de Google Drive a proyectos
- **Creación de Archivos**: Creación de nuevos archivos en Google Drive desde TaskHub
- **Permisos de Acceso**: Gestión de permisos de acceso a archivos y carpetas
- **Actualizaciones en Tiempo Real**: Detección de cambios en archivos y carpetas

7.4 Microsoft Outlook Integration

La integración con Microsoft Outlook permite a los usuarios conectar sus cuentas de Outlook con TaskHub, sincronizando correos electrónicos y eventos de calendario.

7.4.1 Funcionalidades

- **Autenticación OAuth**: Conexión segura con Microsoft Outlook mediante OAuth 2.0
- **Sincronización de Correos**: Vinculación de correos electrónicos a tareas y proyectos
- **Sincronización de Calendario**: Creación y actualización de eventos de calendario en TaskHub

- **Creación de Tareas desde Correos**: Creación de tareas en TaskHub a partir de correos electrónicos
- **Notificaciones de Calendario**: Recepción de notificaciones de eventos de calendario en TaskHub
- **Actualizaciones en Tiempo Real**: Detección de cambios mediante Microsoft Graph API

7.5 Slack Integration

La integración con Slack permite a los usuarios conectar sus espacios de trabajo de Slack con TaskHub, enviando notificaciones y creando tareas desde Slack.

7.5.1 Funcionalidades

- **Autenticación OAuth**: Conexión segura con Slack mediante OAuth 2.0
- **Notificaciones en Canales**: Envío de notificaciones de TaskHub a canales de Slack
- **Creación de Tareas desde Slack**: Creación de tareas en TaskHub mediante comandos de Slack
- **Actualizaciones de Estado**: Actualización del estado de tareas desde Slack
- **Comandos Personalizados**: Definición de comandos personalizados para interactuar con TaskHub
- **Webhooks**: Recepción de webhooks de Slack para actualizaciones en tiempo real

7.6 Extensibilidad y Plugins

TaskHub proporciona un sistema de extensibilidad y plugins que permite a los desarrolladores añadir nuevas integraciones y funcionalidades a la plataforma.

7.6.1 Arquitectura de Plugins

- **Interfaz de Plugin**: Define la estructura y los métodos que deben implementar los plugins
- **Gestor de Plugins**: Carga, gestiona y aísla los plugins
- **API de Plugin**: Proporciona acceso a las funcionalidades de TaskHub desde los plugins
- **Seguridad de Plugins**: Implementa medidas para garantizar la seguridad y el aislamiento
- **Marketplace de Plugins**: Plataforma para descubrir e instalar plugins

7.6.2 Desarrollo de Plugins

- **SDK de Plugin**: Conjunto de herramientas y librerías para facilitar el desarrollo
- **Documentación de Plugin**: Guías y ejemplos para el desarrollo de plugins
- **Entorno de Pruebas**: Entorno aislado para probar plugins antes de su publicación
- **Proceso de Revisión**: Proceso de revisión y aprobación antes de la publicación

7.6.3 Tipos de Plugins

- **Plugins de Integración**: Añaden nuevas integraciones con servicios externos
- **Plugins de Funcionalidad**: Añaden nuevas funcionalidades a TaskHub
- **Plugins de UI**: Modifican o extienden la interfaz de usuario
- **Plugins de Automatización**: Automatizan tareas y flujos de trabajo

8. Características del sistema

El Sistema de Colaboración Unificada TaskHub se diseña con un enfoque modular, escalable y seguro, asegurando una integración fluida entre las herramientas de gestión de proyectos, documentos y comunicación. Este sistema permite a la empresa ofrecer una experiencia personalizada a los usuarios, mejorar la visibilidad operativa y optimizar la toma de decisiones basada en datos.

8.1 Descripción y prioridad

Las características del sistema se han priorizado de la siguiente manera:

1. **Arquitectura Modular Basada en Microservicios** (Alta prioridad): Fundamental para la escalabilidad y mantenibilidad del sistema.
2. **Seguridad y Control de Accesos** (Alta prioridad): Crítico para proteger los datos y garantizar el acceso adecuado.
3. **Gestión de Proyectos y Documentos** (Alta prioridad): Funcionalidad central que proporciona valor inmediato a los usuarios.
4. **Integración con Herramientas Externas** (Alta prioridad): Importante para la interoperabilidad con sistemas existentes.
5. **Personalización de Flujos de Trabajo** (Media prioridad): Mejora la experiencia del usuario pero no es crítica para la funcionalidad básica.
6. **Análisis y Optimización de Procesos** (Media prioridad): Proporciona valor añadido pero puede implementarse en fases posteriores.

8.2 Secuencias de estímulo/respuesta

Las principales secuencias de estímulo/respuesta del sistema incluyen:

1. **Autenticación de Usuario**:
   - Estímulo: Usuario intenta iniciar sesión con sus credenciales.
   - Respuesta: Sistema valida las credenciales, genera tokens JWT y establece una sesión segura.
2. **Creación de Proyecto**:
   - Estímulo: Usuario crea un nuevo proyecto con información básica.
   - Respuesta: Sistema almacena el proyecto y envía notificaciones a los miembros relevantes.
3. **Edición de Documento**:
   - Estímulo: Usuario realiza cambios en un documento compartido.
   - Respuesta: Sistema actualiza el documento en tiempo real para todos los usuarios conectados y registra los cambios en el historial de versiones.
4. **Asignación de Tarea**:
   - Estímulo: Usuario asigna una tarea a otro miembro del equipo.
   - Respuesta: Sistema actualiza el estado de la tarea y envía una notificación al usuario asignado.
5. **Integración con Herramienta Externa**:
   - Estímulo: Usuario conecta su cuenta de GitHub, Google Drive, Microsoft Outlook o Slack.
   - Respuesta: Sistema establece la conexión mediante OAuth y sincroniza los datos relevantes.
6. **Notificación de Cambios**:
   - Estímulo: Se realiza un cambio en un proyecto o documento.
   - Respuesta: Sistema envía notificaciones tipo commit a todos los usuarios relevantes a través de los canales configurados.

8.3 Requisitos funcionales

8.3.1 RF1: Desarrollo de Microservicios

- **Descripción**: El sistema debe implementar una arquitectura de microservicios para la gestión de proyectos, documentos colaborativos, comunicación en tiempo real e integración con herramientas externas.
- **Entrada**: Requerimientos de funcionalidades clave.
- **Salida**: Microservicios desarrollados y desplegados.

- **Excepciones**: Fallos en la implementación de microservicios.
- **Autor**: Omar Hernandez Sotelo.
- **Prioridad**: Alta

8.3.2 RF2: Gestión de proyectos

- **Descripción**: Permitir a los usuarios crear, asignar y seguir el progreso de proyectos, con integración de notificaciones en tiempo real.
- **Entrada**: Datos del proyecto (nombre, descripción, tareas, asignaciones).
- **Salida**: Proyectos creados, asignados y seguidos.
- **Excepciones**: Fallos en la creación o seguimiento de proyectos.
- **Autor**: Equipo de Desarrollo.
- **Prioridad**: Alta

8.3.3 RF3: Documentos colaborativos

- **Descripción**: Permitir a los usuarios crear, editar y compartir documentos en tiempo real, con integración de herramientas de edición colaborativa.
- **Entrada**: Documentos y ediciones realizadas por los usuarios.
- **Salida**: Documentos colaborativos actualizados en tiempo real.
- **Excepciones**: Fallos en la sincronización de documentos.
- **Autor**: Equipo de Desarrollo.
- **Prioridad**: Alta

8.3.4 RF4: Notificaciones en Tiempo Real (Tipo Commit)

- **Descripción**: Permitir a los usuarios recibir notificaciones en tiempo real sobre cambios o actualizaciones en proyectos, tareas o documentos, similares a los commits en Git.
- **Entrada**: Cambios en proyectos, tareas o documentos.
- **Salida**: Notificaciones enviadas a los usuarios.
- **Excepciones**: Fallos en el envío de notificaciones.
- **Autor**: Equipo de Desarrollo.
- **Prioridad**: Media

8.3.5 RF5: Integración con Herramientas Externas

- **Descripción**: Permitir a los usuarios conectar la plataforma con aplicaciones externas, como GitHub, Google Drive, Microsoft Outlook y Slack, con sincronización automática de datos.
- **Entrada**: Datos de herramientas externas (correos, calendarios, mensajes, repositorios).
- **Salida**: Datos sincronizados entre la plataforma y las herramientas externas.
- **Excepciones**: Fallos en la integración o sincronización.
- **Autor**: Equipo de Desarrollo.
- **Prioridad**: Alta

8.3.6 RF6: Registro y Perfiles de Usuarios

- **Descripción**: Permitir a los usuarios registrarse y gestionar sus perfiles, con autenticación y roles de usuario (administrador, propietario, miembro, invitado).
- **Entrada**: Datos de registro (nombre, correo, contraseña).
- **Salida**: Perfiles de usuarios creados y gestionados.
- **Excepciones**: Fallos en el registro o autenticación.
- **Autor**: Equipo de Desarrollo.
- **Prioridad**: Alta

8.3.7 RF7: Operaciones de Deshacer/Rehacer

- **Descripción**: Permitir a los usuarios deshacer y rehacer operaciones en proyectos y tareas, implementando el patrón Command.
- **Entrada**: Solicitudes de deshacer o rehacer operaciones.
- **Salida**: Operaciones deshechas o rehechas.
- **Excepciones**: Fallos en las operaciones de deshacer/rehacer.
- **Autor**: Equipo de Desarrollo.
- **Prioridad**: Media

8.3.8 RF8: Control de Versiones de Documentos

- **Descripción**: Mantener un historial de cambios en los documentos, permitiendo a los usuarios ver versiones anteriores y restaurarlas si es necesario.
- **Entrada**: Cambios en documentos.
- **Salida**: Versiones de documentos almacenadas y accesibles.
- **Excepciones**: Fallos en el control de versiones.
- **Autor**: Equipo de Desarrollo.
- **Prioridad**: Media

8.3.9 RF9: Autenticación Segura y Control de Accesos

- **Descripción**: Implementar un sistema robusto de autenticación y autorización, incluyendo autenticación multifactor, gestión de sesiones y control de acceso basado en roles.
- **Entrada**: Credenciales de usuario, tokens, códigos de verificación.
- **Salida**: Acceso seguro a recursos basado en roles y permisos.
- **Excepciones**: Fallos en la autenticación o autorización.
- **Autor**: Equipo de Desarrollo.
- **Prioridad**: Alta

8.3.10 RF10: Sistema de Plugins y Extensibilidad

- **Descripción**: Proporcionar un sistema de plugins que permita a los desarrolladores añadir nuevas integraciones y funcionalidades a la plataforma.
- **Entrada**: Plugins desarrollados por terceros.
- **Salida**: Funcionalidades adicionales integradas en la plataforma.
- **Excepciones**: Fallos en la carga o ejecución de plugins.
- **Autor**: Equipo de Desarrollo.
- **Prioridad**: Media

8.4 Requisitos no funcionales

8.4.1 RNF1: Arquitectura Desacoplada Basada en Microservicios

- **Descripción**: El sistema debe estar diseñado con una arquitectura modular basada en microservicios, permitiendo la escalabilidad y resiliencia.
- **Entrada**: Requerimientos de arquitectura.

- **Salida**: Arquitectura desacoplada implementada.
- **Excepciones**: Fallos en la implementación de la arquitectura.
- **Autor**: Omar Hernandez Sotelo.
- **Prioridad**: Alta

8.4.2 RNF2: APIs Bien Definidas y Unificadas

- **Descripción**: El sistema debe contar con APIs bien definidas y unificadas para facilitar la integración y el desarrollo.
- **Entrada**: Requerimientos de integración.
- **Salida**: APIs implementadas y documentadas.
- **Excepciones**: Fallos en la definición o implementación de APIs.
- **Autor**: Omar Hernandez Sotelo.
- **Prioridad**: Alta

8.4.3 RNF3: Coordinación y Eliminación de Fragmentación de Datos

- **Descripción**: El sistema debe coordinar los datos entre diferentes microservicios y eliminar la fragmentación de datos.
- **Entrada**: Datos de diferentes microservicios.
- **Salida**: Datos coordinados y sin fragmentación.
- **Excepciones**: Fallos en la coordinación de datos.
- **Autor**: Omar Hernandez Sotelo.
- **Prioridad**: Alta

8.4.4 RNF4: Integración Fluida y Segura con Herramientas Externas

- **Descripción**: El sistema debe permitir una integración fluida y segura con herramientas externas, como GitHub, Google Drive, Microsoft Outlook y Slack.
- **Entrada**: Datos de herramientas externas.
- **Salida**: Integración segura y fluida implementada.
- **Excepciones**: Fallos en la integración o seguridad.
- **Autor**: Omar Hernandez Sotelo.
- **Prioridad**: Alta

8.4.5 RNF5: Seguridad y Control de Accesos

- **Descripción**: El sistema debe implementar mecanismos de seguridad como autenticación, autorización y encriptación de datos.
- **Entrada**: Datos de usuarios y accesos.
- **Salida**: Sistema seguro con control de accesos implementado.
- **Excepciones**: Fallos en la seguridad o control de accesos.
- **Autor**: Omar Hernandez Sotelo.
- **Prioridad**: Alta

8.4.6 RNF6: Escalabilidad y Resiliencia

- **Descripción**: El sistema debe ser escalable y resiliente, permitiendo el crecimiento y la recuperación ante fallos.
- **Entrada**: Requerimientos de escalabilidad y resiliencia.
- **Salida**: Sistema escalable y resiliente implementado.
- **Excepciones**: Fallos en la escalabilidad o resiliencia.
- **Autor**: Omar Hernandez Sotelo.
- **Prioridad**: Alta

8.4.7 RNF7: Monitoreo y Registro de Actividades

- **Descripción**: El sistema debe incluir un sistema de monitoreo y registro de actividades para detectar posibles incidentes.
- **Entrada**: Actividades del sistema.
- **Salida**: Registros de actividades y monitoreo implementado.
- **Excepciones**: Fallos en el monitoreo o registro.
- **Autor**: Omar Hernandez Sotelo.
- **Prioridad**: Media

8.4.8 RNF8: Rendimiento y Tiempos de Respuesta Rápidos

- **Descripción**: El sistema debe garantizar un rendimiento óptimo y tiempos de respuesta rápidos para los usuarios.
- **Entrada**: Solicitudes de usuarios.
- **Salida**: Respuestas rápidas y rendimiento óptimo.
- **Excepciones**: Fallos en el rendimiento o tiempos de respuesta.
- **Autor**: Omar Hernandez Sotelo.
- **Prioridad**: Alta

8.4.9 RNF9: Alta Disponibilidad y Tiempo de Actividad del 99.9 %

- **Descripción**: El sistema debe garantizar una alta disponibilidad y un tiempo de actividad del 99.9 %.
- **Entrada**: Requerimientos de disponibilidad.
- **Salida**: Sistema con alta disponibilidad implementado.
- **Excepciones**: Fallos en la disponibilidad.
- **Autor**: Omar Hernandez Sotelo.
- **Prioridad**: Alta

8.4.10 RNF10: Usabilidad y Experiencia del Usuario Intuitiva

- **Descripción**: El sistema debe ofrecer una experiencia de usuario intuitiva y fácil de usar.
- **Entrada**: Requerimientos de usabilidad.
- **Salida**: Interfaz intuitiva y fácil de usar implementada.
- **Excepciones**: Fallos en la usabilidad o experiencia del usuario.
- **Autor**: Omar Hernandez Sotelo.
- **Prioridad**: Media

8.4.11 RNF11: Seguridad de Datos y Cumplimiento Normativo

- **Descripción**: El sistema debe garantizar la seguridad de los datos y cumplir con las normativas aplicables (GDPR, CCPA, etc.).
- **Entrada**: Requerimientos de seguridad y cumplimiento.
- **Salida**: Sistema seguro y conforme a normativas.
- **Excepciones**: Fallos en la seguridad o cumplimiento.
- **Autor**: Omar Hernandez Sotelo.
- **Prioridad**: Alta

8.4.12 RNF12: Extensibilidad y Sistema de Plugins

- **Descripción**: El sistema debe ser extensible y permitir la adición de nuevas funcionalidades mediante un sistema de plugins.
- **Entrada**: Requerimientos de extensibilidad.
- **Salida**: Sistema extensible con soporte para plugins implementado.
- **Excepciones**: Fallos en la extensibilidad o sistema de plugins.
- **Autor**: Omar Hernandez Sotelo.
- **Prioridad**: Media

9. Requisitos de la interfaz

9.1 Interfaces de Usuario

La interfaz de usuario de TaskHub se desarrolla utilizando Flutter Web, proporcionando una experiencia consistente en diferentes dispositivos. Las principales características de la interfaz de usuario incluyen:

- **Diseño responsivo**: La interfaz se adapta a diferentes tamaños de pantalla, con soporte para dispositivos de escritorio (1024px) y tabletas (768px). No se requiere soporte para dispositivos móviles (¡600px).
- **Páginas principales**:
  - Página de inicio de sesión
  - Página de registro
  - Dashboard de proyectos
  - Vista de proyecto individual
  - Editor de documentos colaborativo
  - Configuración de perfil
  - Configuración de integraciones
- **Componentes de UI**:
  - Barra de navegación lateral
  - Barra de herramientas superior
  - Paneles de tareas
  - Editor de documentos WYSIWYG
  - Notificaciones en tiempo real
  - Diálogos modales para acciones importantes

La interfaz de usuario debe seguir las siguientes directrices:

- Uso de componentes de Flutter Web compatibles con navegadores modernos
- Implementación de temas claros y oscuros
- Accesibilidad según las pautas WCAG 2.1 nivel AA
- Tiempo de carga inicial ¡2s (con conexión 4G simulada)
- Lazy loading para rutas no esenciales

9.2 Interfaces de hardware

TaskHub es una aplicación web y no requiere interfaces de hardware específicas. Los requisitos mínimos para los dispositivos de los usuarios son:

- Navegador web moderno compatible con WebSockets y WebAssembly
- Resolución de pantalla mínima de 768px de ancho
- Conexión a Internet estable

9.3 Interfaces de software

TaskHub interactúa con varios sistemas de software a través de interfaces bien definidas:

- **APIs REST**: Todas las comunicaciones entre el frontend y el backend, así como entre microservicios, utilizan APIs REST con formato JSON.
  - Especificación: OpenAPI 3.0
  - Formato de respuesta: JSON estandarizado
  - Versionado: Todas las APIs deben estar versionadas (ej. /v1/projects)
- **WebSockets**: Utilizados para comunicación en tiempo real, como edición colaborativa de documentos y notificaciones push.
  - Protocolo: WebSocket sobre HTTPS
  - Formato de mensajes: JSON
- **Bases de datos**:
  - PostgreSQL: Para datos transaccionales
  - MongoDB: Para documentos y datos no estructurados
  - Redis: Para caché y broker de mensajes
- **Supabase**: Plataforma de backend como servicio que proporciona:
  - Autenticación y autorización
  - Base de datos PostgreSQL
  - Almacenamiento de archivos
  - Funciones en tiempo real
- **APIs externas**: Integración con servicios de terceros como:
  - GitHub API
  - Google Drive API
  - Microsoft Graph API
  - Slack API

9.4 Interfaces de Comunicación

TaskHub utiliza los siguientes protocolos de comunicación:

- **HTTPS**: Todas las comunicaciones entre el cliente y el servidor deben utilizar HTTPS para garantizar la seguridad.
  - TLS 1.3 o superior
  - Certificados SSL válidos
- **WebSockets**: Utilizados para comunicación en tiempo real.

  - Protocolo: WebSocket sobre HTTPS
  - Autenticación: JWT en la conexión inicial
- **RabbitMQ**: Utilizado para comunicación asíncrona entre microservicios.
  - Protocolo: AMQP
  - Patrones: Publicación/suscripción, RPC
- **Autenticación**: JWT (JSON Web Tokens) con RSA-256.
  - Tokens firmados con RSA-256
  - Expiración configurable
  - Refresh tokens para renovación automática

10. Otros requisitos No funcionales

10.1 Requerimientos de desempeño

Los requisitos de desempeño están relacionados con el rendimiento del sistema, incluyendo tiempos de respuesta, disponibilidad y escalabilidad:

- **Tiempo de carga inicial**: ¡2s (con conexión 4G simulada)
- **TTFB (Time To First Byte)**: ¡300ms para APIs
- **Tiempo de respuesta de API**: ¡500ms para el 95 % de las solicitudes
- **Capacidad de usuarios concurrentes**: Soporte para al menos 1000 usuarios concurrentes por instancia
- **Escalabilidad horizontal**: Capacidad para escalar horizontalmente añadiendo más instancias de microservicios
- **Tiempo de recuperación**: ¡30s para recuperarse de fallos de servicio mediante Circuit Breaker
- **Throughput**: Capacidad para manejar al menos 100 solicitudes por segundo por instancia

10.2 Requisitos de seguridad

Los requisitos de seguridad están relacionados con la protección de datos, autenticación, autorización y monitoreo de actividades:

- **HTTPS obligatorio**: Todas las comunicaciones deben utilizar HTTPS con TLS 1.3 o superior.
- **CSP (Content Security Policy)**: Implementación de políticas de seguridad de contenido para prevenir ataques XSS.

- **Protección contra XSS y CSRF**: Implementación de medidas para prevenir ataques de Cross-Site Scripting y Cross-Site Request Forgery.
- **Autenticación**: JWT con RSA-256 para autenticación segura.
- **Autorización**: Control de acceso basado en roles (RBAC) para diferentes niveles de acceso.
- **Encriptación de datos sensibles**: Datos sensibles encriptados en reposo y en tránsito.
- **Validación de entrada**: Validación estricta de todas las entradas de usuario para prevenir inyecciones.
- **Auditoría de seguridad**: Registros de auditoría para todas las acciones sensibles.
- **Límites de tasa**: Implementación de límites de tasa para prevenir ataques de fuerza bruta y DDoS.

10.3 Atributos de calidad del software

Los atributos de calidad del software están relacionados con la usabilidad, mantenibilidad, interoperabilidad y otros aspectos de calidad:

- **Responsividad**: La interfaz debe ser responsiva y adaptarse a diferentes tamaños de pantalla (768px).
- **Mantenibilidad**: Código modular y bien documentado para facilitar el mantenimiento.
- **Testabilidad**: Cobertura de pruebas del 80 % o superior para garantizar la calidad del código.
- **Interoperabilidad**: APIs bien definidas para facilitar la integración con otros sistemas.
- **Extensibilidad**: Arquitectura que permite añadir nuevas funcionalidades sin modificar el código existente.
- **Portabilidad**: Frontend compatible con navegadores modernos (Chrome, Firefox, Safari, Edge).
- **Accesibilidad**: Cumplimiento de las pautas WCAG 2.1 nivel AA.

11. Otros Requerimientos

Esta categoría incluye requisitos que no encajan directamente en las categorías anteriores, como la arquitectura del sistema y la integración con herramientas externas.

11.1 Flutter Web

- **Tamaño máximo**: 1MB inicial (compresión gzip)
- **Lazy loading**: Obligatorio para rutas no esenciales
- **WebSockets**: Implementación para edición colaborativa y notificaciones en tiempo real
- **Compatibilidad**: Soporte para navegadores modernos (Chrome, Firefox, Safari, Edge)
- **Responsividad**: Diseño adaptable para escritorio (1024px) y tabletas (768px)

11.2 Backend

- **Formato de respuesta**: JSON estandarizado
- **Documentación**: OpenAPI 3.0 para todas las APIs
- **Versionado**: Todas las APIs deben estar versionadas (ej. /v1/projects)
- **Manejo de errores**: Respuestas de error consistentes con códigos HTTP apropiados
- **Validación**: Validación estricta de todas las entradas

11.3 Prohibiciones

- **Uso de plugins Flutter no compatibles con la web**: Solo se permiten plugins que funcionen correctamente en Flutter Web.
- **APIs no versionadas**: Todas las APIs deben incluir un número de versión en la ruta.
- **Almacenamiento de tokens JWT en localStorage**: Los tokens deben almacenarse de forma segura, no en localStorage.
- **Componentes de la IU que dependen de características móviles**: No se deben utilizar componentes que requieran funcionalidades específicas de dispositivos móviles.

11.4 Entregables

11.4.1 Frontend

- Código fuente del frontend en Flutter Web
- Pruebas golden (UI) para componentes principales
- Documentación de la interfaz de usuario

11.4.2 Backend

- Código fuente de los microservicios
- Pruebas unitarias (80 % de cobertura de código)
- Documentación de API (OpenAPI 3.0)
- Configuración de Docker y Kubernetes

11.4.3 Integración continua (CI/CD)

- Configuración de pipelines de CI/CD
- Scripts de despliegue automatizado
- Configuración de entornos de desarrollo, pruebas y producción

12. Características del MVP

El Producto Mínimo Viable (MVP) de TaskHub incluye las siguientes características esenciales:

12.1 Registro y Perfiles de Usuarios

- **Descripción**: Permite a los usuarios crear una cuenta y personalizar su perfil.
- **Funcionalidades**:
  - Registro con email y contraseña
  - Autenticación segura con JWT
  - Gestión de perfiles de usuario
  - Roles de usuario (administrador, propietario, miembro, invitado)
- **Relacionado con**: RF6, RF9, RNF5 (Seguridad y Control de Accesos)

12.2 Creación y Gestión de Proyectos

- **Descripción**: Permite a los usuarios crear, asignar y seguir el progreso de proyectos.
- **Funcionalidades**:
  - Creación de proyectos con información básica
  - Asignación de miembros a proyectos
  - Seguimiento del estado del proyecto
  - Historial de actividades del proyecto
- **Relacionado con**: RF2, RNF1 (Arquitectura Modular)

12.3 Microservicio de Documentos Colaborativos

- **Descripción**: Permite a los usuarios crear, editar y compartir documentos en tiempo real.
- **Funcionalidades**:
  - Creación de documentos de texto
  - Edición colaborativa en tiempo real
  - Control de versiones básico
  - Permisos de acceso a documentos
- **Relacionado con**: RF3, RNF3 (Coordinación de Datos)

12.4 Microservicio de Comunicación en Tiempo Real

- **Descripción**: Facilita la comunicación entre usuarios a través de notificaciones.
- **Funcionalidades**:
  - Notificaciones en tiempo real tipo commit
  - Notificaciones in-app
  - Notificaciones por email (básicas)
- **Relacionado con**: RF4, RNF6 (Escalabilidad y Resiliencia)

12.5 Microservicio de Seguimiento de Tareas

- **Descripción**: Permite a los usuarios asignar y seguir el estado de tareas dentro de los proyectos.
- **Funcionalidades**:
  - Creación y asignación de tareas
  - Estados de tareas (pendiente, en progreso, completada)
  - Fechas límite y prioridades
  - Comentarios en tareas
- **Relacionado con**: RF2, RNF7 (Monitoreo y Registro)

12.6 API Unificada para Integración Externa

- **Descripción**: Proporciona acceso a los datos clave de la plataforma mediante APIs bien definidas.
- **Funcionalidades**:
  - APIs REST documentadas con OpenAPI 3.0
  - Autenticación mediante JWT
  - Endpoints para proyectos, tareas y documentos
- **Relacionado con**: RF5, RNF2 (APIs Bien Definidas)

12.7 Pasarela Segura para APIs

- **Descripción**: Controla el acceso a los servicios mediante autenticación y autorización.
- **Funcionalidades**:
  - API Gateway como punto de entrada único
  - Autenticación y validación de tokens
  - Enrutamiento de solicitudes
  - Circuit Breaker para prevenir fallos en cascada
- **Relacionado con**: RF1, RF9, RNF5 (Seguridad y Control de Accesos)

12.8 Almacén de Datos en la Nube

- **Descripción**: Consolida los datos de proyectos, documentos y comunicación para análisis.
- **Funcionalidades**:
  - Almacenamiento en PostgreSQL y MongoDB
  - Caché con Redis
  - Sincronización de datos entre servicios
  - Backup automático
- **Relacionado con**: RF1, RNF3 (Coordinación de Datos)

12.9 Integración con GitHub

- **Descripción**: Permite a los usuarios conectar sus repositorios de GitHub con TaskHub.
- **Funcionalidades**:
  - Autenticación OAuth con GitHub
  - Sincronización de issues con tareas
  - Visualización de commits relacionados con tareas
  - Webhooks para actualizaciones en tiempo real
- **Relacionado con**: RF5, RNF4 (Integración con Herramientas Externas)

13. Resumen de Relaciones

La siguiente tabla muestra la relación entre las características del MVP y los requisitos funcionales y no funcionales:

Característica del MVP	Relacionado con RF	Relacionado con RNF
Registro y Perfiles de Usuarios	RF6, RF9	RNF5, RNF11
Creación y Gestión de Proyectos	RF2	RNF1
Microservicio de Documentos Colaborativos	RF3	RNF3
Microservicio de Comunicación en Tiempo Real	RF4	RNF6
Microservicio de Seguimiento de Tareas	RF2	RNF7
API Unificada para Integración Externa	RF5	RNF2
Pasarela Segura para APIs	RF1, RF9	RNF5
Almacén de Datos en la Nube	RF1	RNF3
Integración con GitHub	RF5, RF10	RNF4, RNF12

14. Referencias

1. Documentación de Flutter Web: https://flutter.dev/docs/development/platform-integration/web
2. Documentación de FastAPI: https://fastapi.tiangolo.com/
3. Documentación de SQLAlchemy: https://docs.sqlalchemy.org/
4. Documentación de Supabase: https://supabase.io/docs
5. Patrones de Diseño de Microservicios: https://microservices.io/patterns/index.html
6. Documentación de Docker: https://docs.docker.com/
7. Documentación de Kubernetes: https://kubernetes.io/docs/
8. Documentación de OpenAPI: https://swagger.io/specification/
9. Pautas de Accesibilidad WCAG 2.1: https://www.w3.org/TR/WCAG21/
10. Repositorio TaskHub: https://github.com/ISCODEVUTB/TaskHub
11. Estándares de Seguridad JWT: https://auth0.com/docs/secure/tokens/json-web-tokens
12. OWASP Top 10: https://owasp.org/www-project-top-ten/
13. Guía de Implementación de RBAC: https://auth0.com/docs/authorization/rbac
14. Estándares de Autenticación Multifactor: https://pages.nist.gov/800-63-3/sp800-63b.html

15. GitHub API Documentation: https://docs.github.com/en/rest

16. Google Drive API Documentation: https://developers.google.com/drive/api

17. Microsoft Graph API Documentation: https://docs.microsoft.com/en-us/graph/overview

18. Slack API Documentation: https://api.slack.com/

19. OAuth 2.0 Specification: https://oauth.net/2/

15. Apéndices

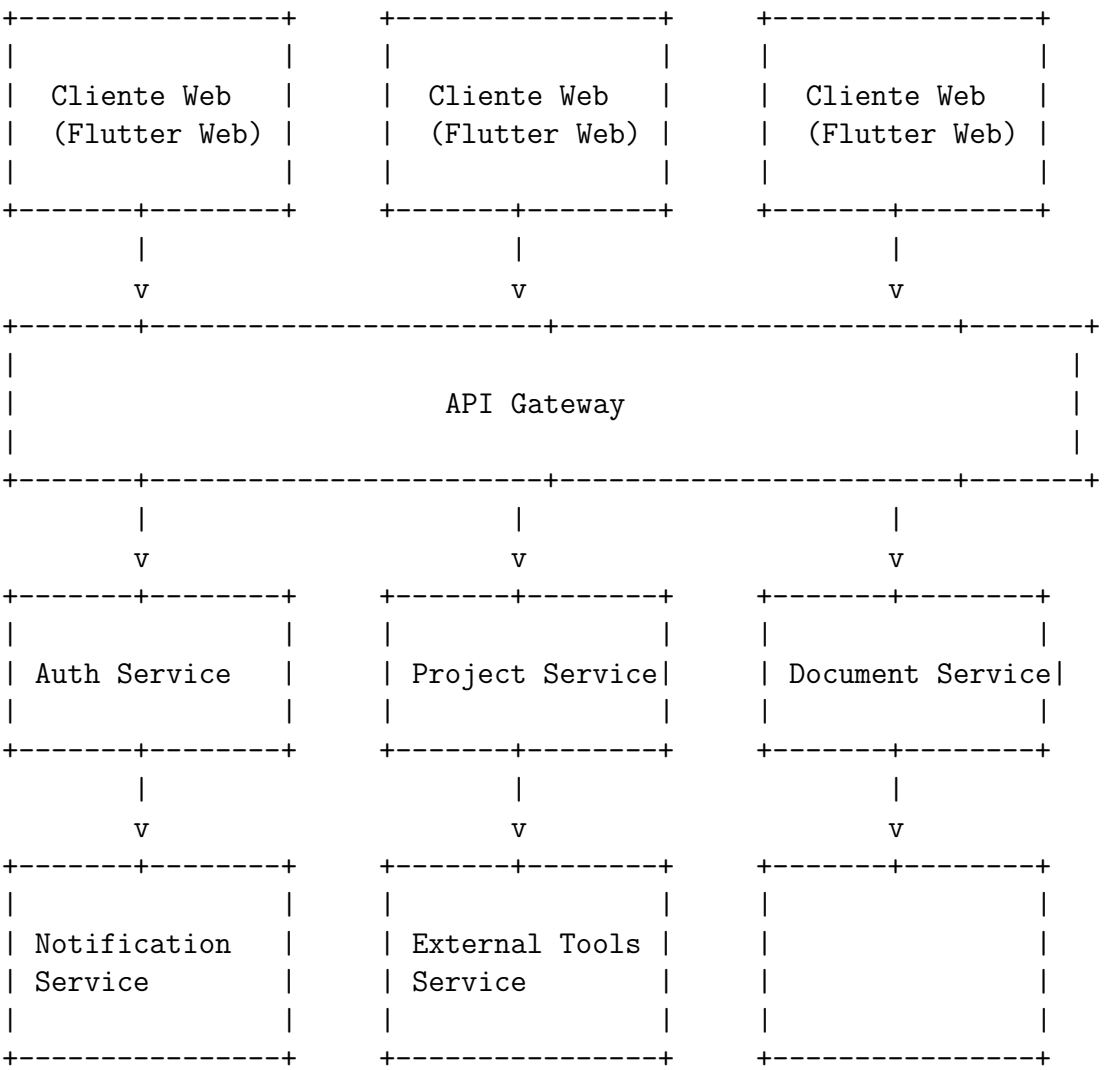
15.1 Apéndice A: Glosario

- **API Gateway**: Punto de entrada único para todas las solicitudes de los clientes, responsable del enrutamiento, autenticación y otras funciones transversales.
- **Circuit Breaker**: Patrón de diseño que previene fallos en cascada cuando un servicio no está disponible.
- **CQRS**: Command Query Responsibility Segregation, patrón que separa las operaciones de lectura y escritura.
- **JWT**: JSON Web Token, estándar para la creación de tokens de acceso.
- **Microservicio**: Servicio pequeño e independiente que representa una capacidad empresarial específica.
- **OAuth**: Protocolo de autorización que permite a aplicaciones de terceros acceder a recursos en nombre de un usuario.
- **Patrón Adapter**: Patrón de diseño que permite que interfaces incompatibles trabajen juntas.
- **Patrón Command**: Patrón de diseño que encapsula una solicitud como un objeto, permitiendo parametrizar clientes con diferentes solicitudes.
- **Patrón Factory Method**: Patrón de diseño que define una interfaz para crear un objeto, pero deja que las subclases decidan qué clase instanciar.
- **Patrón Observer**: Patrón de diseño que define una dependencia uno a muchos entre objetos, de modo que cuando un objeto cambia de estado, todos sus dependientes son notificados.
- **RBAC**: Role-Based Access Control, control de acceso basado en roles.
- **WebSocket**: Protocolo de comunicación bidireccional en tiempo real sobre una única conexión TCP.
- **WYSIWYG**: What You See Is What You Get, editor que muestra el contenido tal como aparecerá en el resultado final.

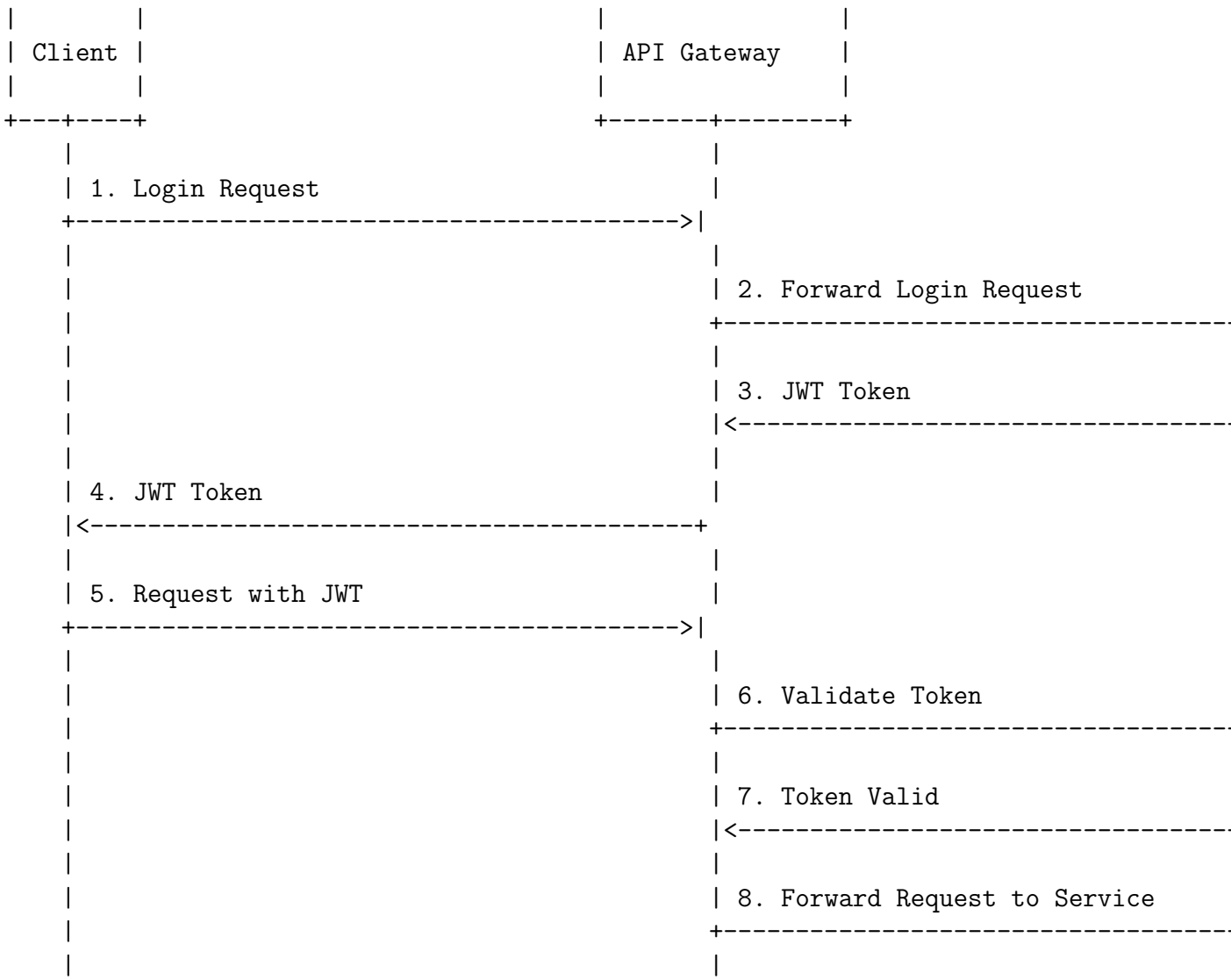
- **2FA**: Two-Factor Authentication, autenticación de dos factores, método de seguridad que requiere dos formas diferentes de verificación de identidad.
- **TOTP**: Time-based One-Time Password, contraseña de un solo uso basada en tiempo, utilizada en autenticación de dos factores.
- **Webhook**: Mecanismo que permite a una aplicación proporcionar a otras aplicaciones información en tiempo real cuando ocurre un evento.
- **Plugin**: Componente de software que añade una característica específica a una aplicación existente.
- **OAuth 2.0**: Protocolo de autorización que permite a aplicaciones de terceros obtener acceso limitado a una cuenta de usuario en un servicio HTTP.

15.2 Apéndice B: Modelos de análisis

15.2.1 Diagrama de Arquitectura de Microservicios



15.2.2 Diagrama de Flujo de Autenticación



15.3 Apéndice C: Lista de problemas

1. **Integración con servicios externos**: La integración con servicios externos como GitHub, Google Drive, Microsoft Outlook y Slack requiere una investigación adicional para garantizar la compatibilidad y la seguridad.

2. **Edición colaborativa en tiempo real**: La implementación de la edición colaborativa en tiempo real para documentos es un desafío técnico que requiere una investigación adicional.

3. **Escalabilidad de WebSockets**: La escalabilidad de las conexiones WebSocket para un gran número de usuarios concurrentes es un área que requiere pruebas adicionales.

4. **Sincronización de datos entre microservicios**: La sincronización de datos entre microservicios es un desafío que requiere la implementación de patrones como Event Sourcing y CQRS.

5. **Despliegue en diferentes plataformas**: El despliegue en diferentes plataformas

como AWS, Azure y Fly.io requiere configuraciones específicas que deben ser investigadas y documentadas.

6. **Gestión de tokens y sesiones**: La implementación de un sistema robusto para la gestión de tokens y sesiones, incluyendo la revocación de tokens y la detección de sesiones sospechosas, requiere una investigación adicional.

7. **Implementación de autenticación multifactor**: La integración de diferentes métodos de autenticación multifactor (TOTP, SMS, email) y la gestión de códigos de recuperación requiere una planificación cuidadosa.

8. **Cumplimiento normativo**: El cumplimiento de normativas como GDPR, CCPA y otras regulaciones de privacidad requiere una revisión legal y técnica adicional.

9. **Gestión de plugins de terceros**: La implementación de un sistema seguro para la carga y ejecución de plugins de terceros requiere un diseño cuidadoso para evitar problemas de seguridad.

10. **Sincronización bidireccional con servicios externos**: La sincronización bidireccional de datos entre TaskHub y servicios externos como GitHub, Google Drive, Microsoft Outlook y Slack requiere un manejo cuidadoso de conflictos y estados inconsistentes.