UNIVERSIDAD PRIVADA DE TACNA

FACULTAD DE INGENIERÍA

Escuela Profesional de Ingeniería de Sistemas

Desarrollo de un Sistema de Conversión y Organización de Documentos Técnicos en Markdown con Estructuración Automática y Control de Versiones para los estudiantes en la facultad de Ingeniería de Sistemas

Curso: *Patrones de Software*

Docente: Ing. Patrick Jose Cuadros Quiroga

Integrantes:

Chambi Cori, Jerson Roni	***(2021072619)***
Flores Quispe, Jaime Elias	***(2021070309)***
Leyva Sardon, Elvis Ronald	***(2021072614)***
Chite Quispe, Brian Danilo	***(2021070015)***

Tacna – Perú

2025

CONTROL DE VERSIONES					
Versión	Hecha por	Revisada por	Aprobada por	Fecha	Motivo
1.0	MPV	ELV	ARV	10/10/2020	Versión Original

Sistema de Conversión y Organización de Documentos Técnicos en Markdown con Estructuración Automática y Control de Versiones para los estudiantes en la facultad de Ingeniería de Sistemas

Documento de Especificación de Requerimientos de Software

Versión *{1.0}*

CONTROL DE VERSIONES					
Versión	Hecha por	Revisada por	Aprobada por	Fecha	Motivo
1.0	MPV	ELV	ARV	10/10/2020	Versión Original

INDICE GENERAL

INTRODUCCIÓN

En el ámbito académico, especialmente en carreras como Ingeniería de Sistemas, la documentación es una parte fundamental del proceso de aprendizaje y desarrollo de proyectos. Los estudiantes y docentes trabajan constantemente con documentos técnicos, informes, guías de laboratorio y material de estudio en diversos formatos, como Word, PDF, HTML y TXT. Sin embargo, la falta de un estándar unificado y herramientas eficientes para la gestión de estos documentos genera problemas significativos en su organización, accesibilidad y colaboración.

Actualmente, los estudiantes enfrentan dificultades al migrar sus documentos a plataformas de control de versiones como GitHub, donde el formato Markdown (.md) se ha convertido en un estándar ampliamente adoptado para la documentación técnica. La conversión manual de archivos a este formato no solo consume tiempo, sino que también puede generar inconsistencias en la estructura, pérdida de formato y dificultades en la navegación. Además, la falta de un sistema que permita gestionar versiones anteriores de los documentos complica el seguimiento de cambios y la colaboración en equipo.

Esta problemática motivó el desarrollo de un Sistema de Conversión y Organización de Documentos Técnicos en Markdown con Estructuración Automática y Control de Versiones, una solución diseñada para optimizar la gestión de la documentación académica en la Facultad de Ingeniería de Sistemas. El proyecto busca automatizar el proceso de conversión de documentos, garantizando que se preserve su estructura original, al mismo tiempo que incorpora funcionalidades avanzadas como la generación de archivos de navegación (como Sidebar.md y Footer.md), mejora automáticacon IA y control de versiones integrado.

La implementación de este sistema no solo mejorará la eficiencia en la creación y mantenimiento de documentación técnica, sino que también promoverá buenas prácticas entre los estudiantes, facilitando la adopción de herramientas profesionales como GitHub y fomentando el trabajo colaborativo. Asimismo, al estandarizar el formato de los documentos, se reducirán los errores causados por conversiones manuales y se incrementará la accesibilidad de la información académica.

I. Generalidades de la Empresa

1. Nombre de la Empresa

Universidad Privada de Tacna

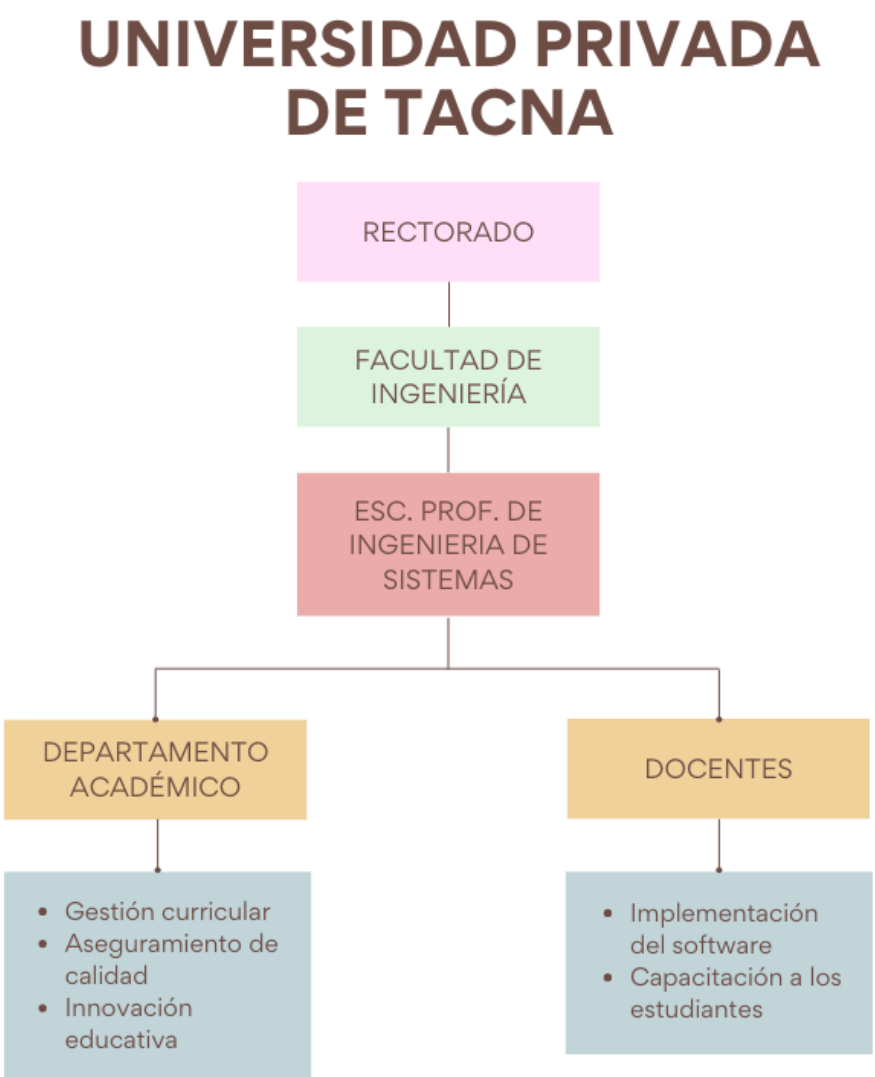
2. Visión

Ser reconocida como la institución líder en formación de ingenieros en el sur del Perú, impulsando la investigación y el uso de tecnologías disruptivas en el ámbito académico y profesional.

3. Misión

Formar profesionales competentes en ingeniería mediante una educación de calidad, integrando herramientas tecnológicas modernas y fomentando la investigación aplicada para contribuir al desarrollo sostenible de la sociedad.

4. Organigrama



II. Visionamiento de la Empresa

1. Descripción del Problema

La Universidad Privada de Tacna (UPT) enfrenta desafíos en la gestión eficiente de documentos técnicos dentro de la Facultad de Ingeniería de Sistemas. Actualmente, estudiantes y docentes trabajan con documentos en formatos diversos (Word, PDF, HTML, TXT), lo que genera:

- Falta de estandarización en la estructura de documentos académicos.
- Dificultad en la conversión manual a Markdown para su uso en plataformas como GitHub.
- Pérdida de tiempo en el formateo y organización de archivos.
- Problemas de colaboración por la ausencia de un sistema integrado de control de versiones.

Este problema afecta directamente la productividad académica y la adopción de buenas prácticas en documentación técnica.

2. Objetivos de Negocios

- Optimizar la gestión documental en la Facultad de Ingeniería de Sistemas mediante un sistema automatizado.
- Reducir tiempos de procesamiento en la conversión y organización de documentos técnicos.
- Fomentar el uso de estándares como Markdown y GitHub entre estudiantes y docentes.
- Mejorar la colaboración académica mediante el control de versiones y acceso estructurado a la información.
- Posicionar a la UPT como una institución innovadora en la adopción de tecnologías educativas.

3. Objetivos de Diseño

- Desarrollar una plataforma web intuitiva que permita la conversión automática de documentos a Markdown.
- Implementar un sistema de control de versiones integrado con GitHub.
- Garantizar la preservación de la estructura (títulos, listas, tablas) durante la conversión.

- Implementar inteligencia artificial (Deepseek) para el mejoramiento de documentos markdown
- Asegurar accesibilidad multiplataforma (web y móvil) con autenticación segura.
- Generar archivos auxiliares (Sidebar.md, Footer.md) para mejorar la navegación.

4. Alcance del proyecto

INCLUYE	NO INCLUYE
- Desarrollo de un módulo de conversión de Word, PDF, HTML y TXT a Markdown.	- Soporte para formatos de ecuaciones complejas (LaTeX).
- Integración con GitHub para gestión de versiones.	- Almacenamiento en la nube externo (Google Drive, Dropbox).
- Sistema de autenticación de usuarios (estudiantes, docentes, administradores).	- Desarrollo de aplicaciones nativas para móviles (solo web responsive).
- Generación automática de estructura de navegación (índices y footers).	
- Funcionalidad de mejora de documentos con inteligencia artificial (API externa).	

5. Viabilidad del Sistema

Tipo de Viabilidad	Análisis Detallado	Soporte Documental
Técnica	El sistema es viable gracias al uso de tecnologías probadas como Python (Flask) para el backend, GitHub API para control de versiones y bibliotecas de conversión de documentos (ej: pdf2md, mammoth). La infraestructura requerida (servidores, bases de datos SQL) está disponible en la UPT.	FD01 (Pág. 8-9): Especifica requisitos de hardware/software. FD02 (Pág. 16): Detalla estándares de compatibilidad técnica.
Económica	El costo total del proyecto (S/ 37,400) incluye desarrollo, hosting y personal. La inversión se justifica por el ahorro en tiempo y la mejora en productividad académica. No requiere licencias costosas (tecnologías open-source).	FD01 (Pág. 9-10): Desglose de costos generales, operativos y de personal.

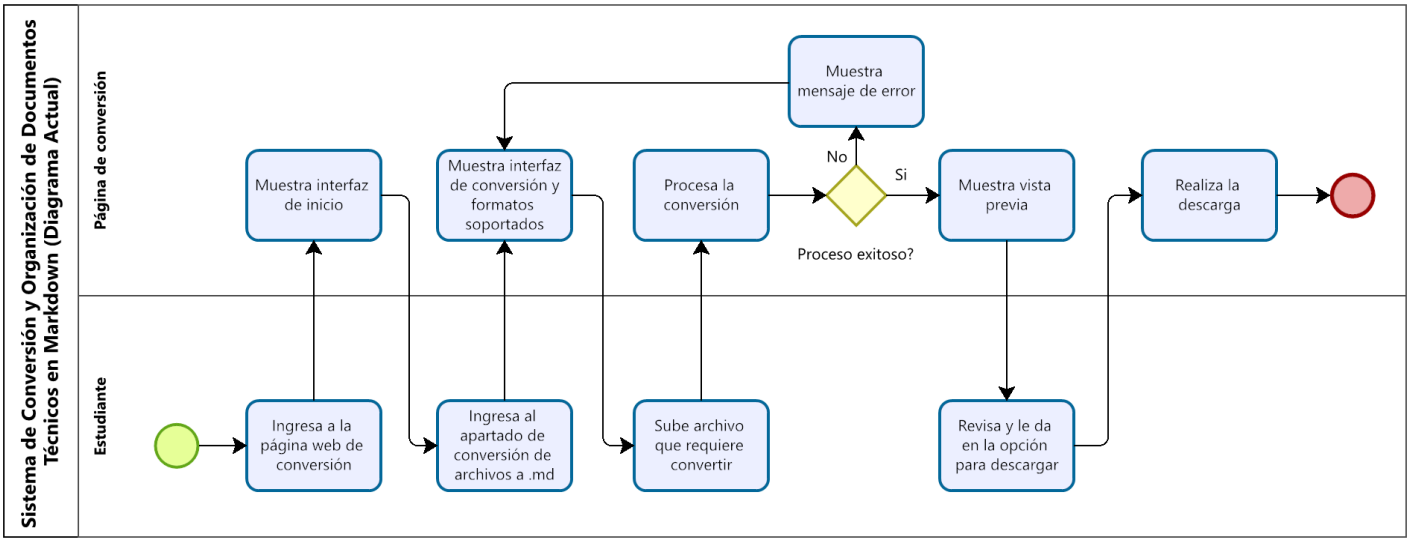
Operativa	La plataforma se integra con el flujo de trabajo actual de estudiantes/docentes. La interfaz intuitiva reduce la curva de aprendizaje. Se prevé capacitaciones breves para su adopción.	FD01 (Pág. 10-11): Beneficios operativos. FD02 (Pág. 10): Entorno de usuario y usabilidad.
Legal	Cumple con: - Ley de Protección de Datos (LPDP). - Términos de uso de GitHub API. - Políticas de propiedad intelectual para documentos académicos.	FD01 (Pág. 11): Factibilidad legal. FD02 (Pág. 17): Estándares legales (privacidad, propiedad intelectual).
Social	Impacto positivo en: - Estudiantes: Facilita documentación técnica estándar. - Docentes: Agiliza revisión de trabajos. - UPT: Promueve innovación educativa.	FD01 (Pág. 11-12): Factibilidad social. FD03: Alineación con visión/misión UPT.
Ambiental	Reduce el uso de papel al digitalizar documentos. Optimiza almacenamiento en la nube (menos duplicación de archivos).	FD01 (Pág. 12): Factibilidad ambiental.

6. Información obtenida del Levantamiento de Información

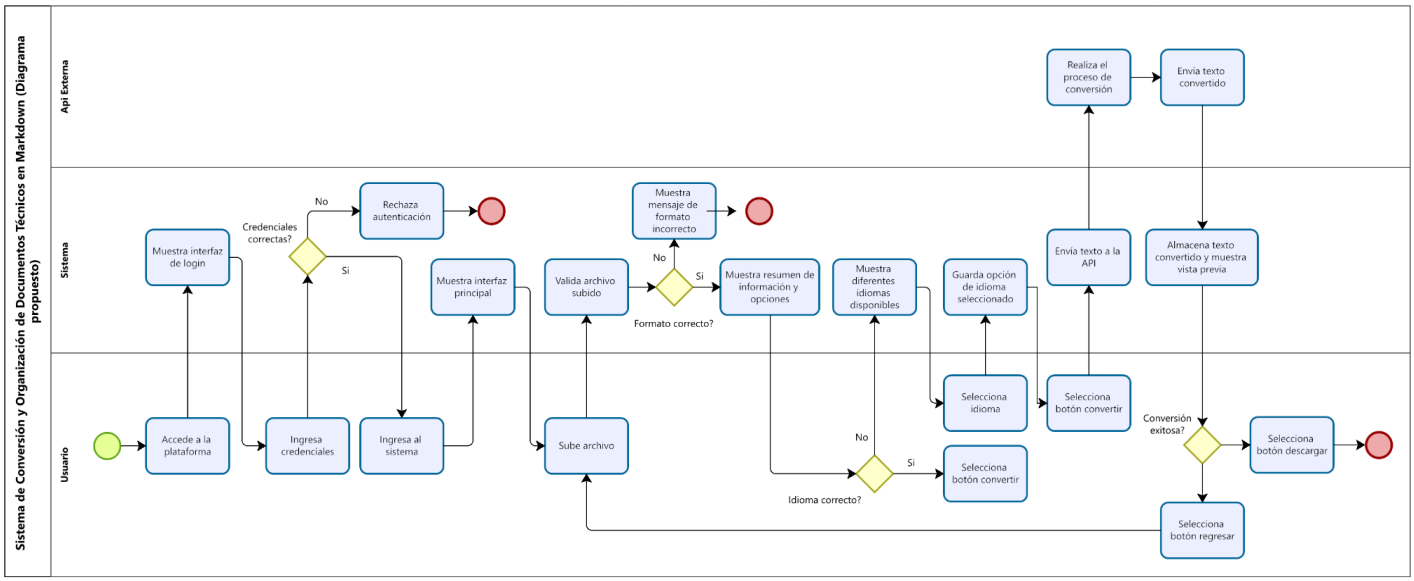
- Entrevistas con docentes y estudiantes: Confirmaron la necesidad de automatizar la conversión a Markdown y mejorar la colaboración.
- Análisis de documentos actuales: 85% de los archivos académicos están en Word/PDF, con estructura inconsistente.
- Benchmarking: Herramientas similares (como Pandoc) no ofrecen control de versiones integrado ni generación de navegación.
- Requerimientos técnicos: La UPT cuenta con infraestructura básica (servidores, internet 100 Mbps) para alojar el sistema.

III. Análisis de Procesos

a) Diagrama del Proceso Actual – Diagrama de actividades



b) Diagrama del Proceso Propuesto – Diagrama de actividades Inicial



IV Especificación de Requerimientos de Software

a) Cuadro de Requerimientos funcionales Inicial

ID	Requerimiento	Descripción	Prioridad
RF01	Registrar usuario	Crear una nueva cuenta con nombre de usuario, email y contraseña segura.	Alta
RF02	Iniciar sesión	Autenticarse con credenciales para acceder al sistema.	Alta
RF03	Subir documento	Cargar archivos (PDF, DOCX, etc.) asignándoles un título descriptivo.	Alta
RF04	Convertir a Markdown	Transformar automáticamente documentos subidos a formato Markdown.	Alta
RF05	Descargar documento	Obtener una copia del documento en formato Markdown.	Alta
RF06	Listar documentos	Visualizar todos los documentos pertenecientes al usuario.	Media
RF07	Gestionar versiones	Mantener un historial de cambios por cada documento.	Media
RF08	Validar formato de archivo	Asegurar que solo se acepten tipos de archivo permitidos (PDF, DOCX, etc.).	Alta
RF09	Previsualizar archivo generado	Cargar la previsualización del archivo markdown generado antes de la descarga	Media

RF10	Mejorar archivo generado	Mejorar el estilo, ortografía, orden y consistencia del archivo markdown a través de inteligencia artificial (deepseek).	Alta
RF11	Buscar documentos	Buscar documentos por título, contenido o fecha de creación.	Media
RF12	Compartir documento	Generar un enlace temporal para compartir el documento con otros usuarios.	Media
RF13	Gestionar categorías	Crear, editar y asignar categorías a los documentos para mejor organización.	Baja
RF14	Exportar a múltiples formatos	Convertir documentos Markdown a otros formatos como HTML, PDF o DOCX.	Alta
RF15	Configurar preferencias de conversión	Personalizar parámetros de conversión según necesidades específicas del usuario.	Media

b) Cuadro de Requerimientos No funcionales

ID	Requerimiento	Descripción	Prioridad
RF01	Seguridad	El sistema debe implementar encriptación SSL/TLS, almacenar contraseñas con hash seguro y protección contra inyecciones SQL y XSS.	Alta

RF02	Rendimiento	La conversión de documentos debe completarse en menos de 60 segundos para archivos de hasta 10MB.	Alta
RF03	Escalabilidad	La plataforma debe evitar la degradación del servicio.	Alta
RF04	Disponibilidad	El sistema debe garantizar un tiempo de actividad.	Alta
RF05	Usabilidad	La interfaz debe ser intuitiva y permitir completar tareas principales con el menor esfuerzo posible.	Alta

c) Cuadro de Requerimientos funcionales Final

ID	Requerimiento	Descripción	Prioridad
RF01	Registrar usuario	Crear una nueva cuenta con nombre de usuario, email y contraseña segura.	Alta
RF02	Iniciar sesión	Autenticarse con credenciales para acceder al sistema.	Alta
RF03	Subir documento	Cargar archivos (PDF, DOCX, etc.) asignándoles un título descriptivo.	Alta
RF04	Convertir a Markdown	Transformar automáticamente documentos subidos a formato Markdown.	Alta
RF05	Descargar documento	Obtener una copia del documento en formato Markdown.	Alta

RF06	Listar documentos	Visualizar todos los documentos pertenecientes al usuario.	Media
RF07	Gestionar versiones	Mantener un historial de cambios por cada documento.	Media
RF08	Validar formato de archivo	Asegurar que solo se acepten tipos de archivo permitidos (PDF, DOCX, etc.).	Alta
RF09	Previsualizar archivo generado	Cargar la previsualización del archivo markdown generado antes de la descarga	Media
RF10	Mejorar archivo generado	Mejorar el estilo, ortografía, orden y consistencia del archivo markdown a través de inteligencia artificial.	Alta
RF11	Analizar similitudes entre documentos	Calcular y mostrar el porcentaje de similitud entre dos documentos markdown utilizando la similitud del coseno.	Media
RF12	Publicar README en GitHub	Permitir al usuario subir el archivo markdown como README.md en el repositorio personal autorizado en GitHub.	Alta
RF13	Publicar archivo en Wiki de GitHub	Subir un archivo markdown al Wiki del repositorio GitHub del usuario, con autenticación mediante token.	Alta
RF14	Publicar múltiples archivos y generar índice en Wiki	Permitir subir varios archivos markdown al Wiki del repositorio en GitHub y generar automáticamente una página de inicio con enlaces (índice).	Alta

V Fase de Desarrollo

1. **Perfiles de Usuario**

- **Perfil de Estudiante**

  Descripción:

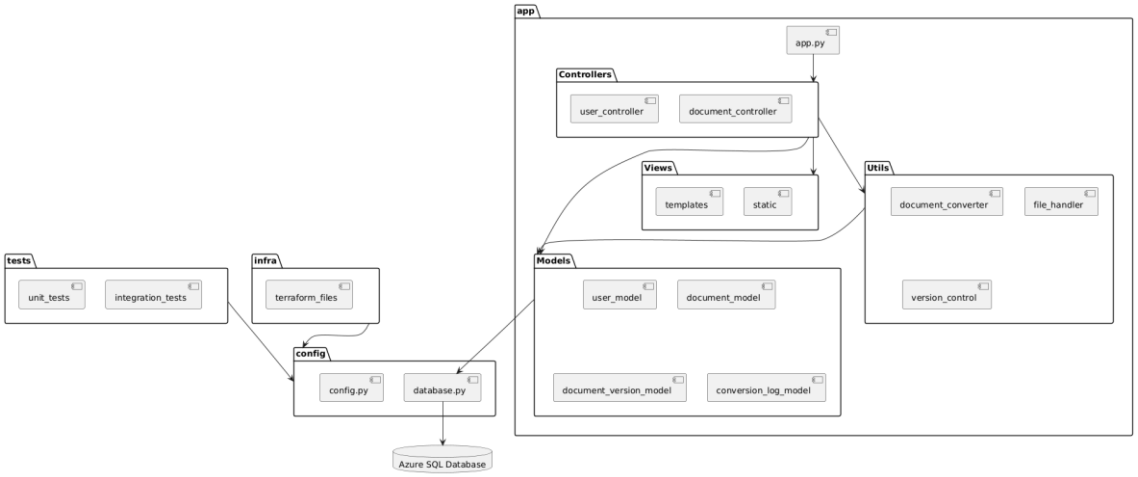
  Usuarios principales del sistema (estudiantes de la Facultad de Ingeniería de Sistemas) que necesitan convertir, organizar y versionar documentos técnicos (tareas, proyectos, apuntes) en formato Markdown.

  Permisos y Accesos:

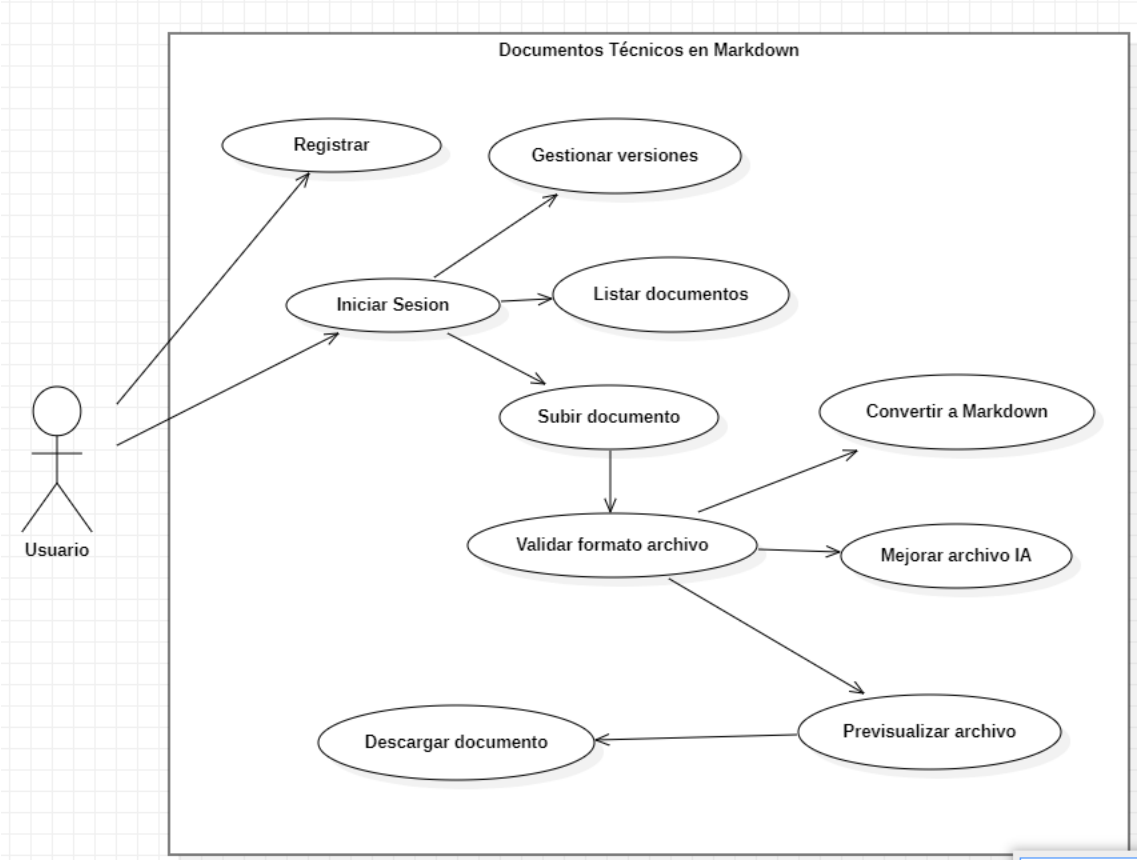
  - **Convertir documentos (PDF, DOCX, etc.) a Markdown.**
  - **Organizar documentos en carpetas/proyectos.**
  - **Gestionar versiones de documentos (historial de cambios).**
  - **Previsualizar documentos antes de guardar.**
  - **Descargar documentos en Markdown.**
  - **Solicitar mejoras automáticas de estructura (IA).**

2. Modelo Conceptual

a) Diagrama de Paquetes



b) Diagrama de Casos de Uso



c) Escenarios de Caso de Uso (narrativa)

RF-001-Registrar usuario

Registrar usuario	
Tipo	**Obligatorio**
Autor(es)	**Brian Danilo Chite Quispe**
Actores	**Usuario Registrado, Sistema**
Descripción	**El usuario ingresa sus datos (nombre de usuario, email y contraseña) en la interfaz de registro. El sistema valida que los datos sean únicos y cumplan con los requisitos de seguridad antes de guardarlos en la base de datos.**
Precondiciones	**1. El usuario no debe estar registrado previamente.** **2. El sistema debe estar operativo y accesible.**
Narrativa de cada de uso	
Acción del actor	**Respuesta del sistema**
1. 1. El usuario ingresa nombre de usuario, email y contraseña en el formulario de registro.	**2. El sistema recibe los datos y los envía al controlador (API) para validación.**
3. El usuario envía el formulario	**4. El sistema verifica si el nombre de usuario ya existe en la base de datos. Si ya existe, muestra: *"El nombre de usuario ya está registrado"*.**
	5. El sistema verifica si el email ya existe. Si ya existe, muestra: *"El email ya está registrado"*.
	6. El sistema valida que la contraseña tenga al menos 8 caracteres e incluya letras y números. Si no cumple,

	muestra: *"Contraseña no válida. Debe tener al menos 8 caracteres y contener letras y números".*
7. Si todos los datos son válidos, el usuario confirma el registro.	**8. El sistema guarda el nuevo usuario en la base de datos y muestra: *"Registro exitoso".***

RF-002- Iniciar sesión

Iniciar sesión	
Tipo	**Obligatorio**
Autor(es)	**Jerson Roni Chambi Cori**
Actores	**Usuario Registrado, Sistema**
Descripción	**Permite a un usuario registrado autenticarse en el sistema mediante nombre de usuario y contraseña. El sistema verifica las credenciales y genera un token de acceso si son válidas.**
Precondiciones	**1. El usuario debe estar registrado previamente.** **2. El sistema debe estar operativo y accesible.**
Narrativa de cada de uso	
Acción del actor	**Respuesta del sistema**
1. El usuario ingresa nombre de usuario y contraseña en el	**2. El sistema (Front-end) envía los datos al Controlador (API).**

formulario de inicio de sesión.	
3. El usuario envía el formulario.	**4. El Controlador (API) solicita al Modelo (API) verificar las credenciales.**
	5. El Modelo (API) consulta en la base de datos si el usuario existe: **- Si no existe: Responde *"Usuario no encontrado"* y muestra mensaje de error.** **- Si existe: Verifica si la contraseña es correcta.**
	6. Si la contraseña es incorrecta, responde *"Contraseña incorrecta"* y muestra mensaje de error (*"Credenciales incorrectas"*).
	7. Si la contraseña es correcta, genera un token de acceso.
	8. El sistema redirige al usuario a la página principal y muestra mensaje de éxito.

RF-003- Subir documento

Subir documento	
Tipo	**Obligatorio**
Autor(es)	**Jaime Elias Flores Quispe**
Actores	**Usuario Registrado, Sistema**
Descripción	**Permite a un usuario subir un documento al sistema para su conversión a Markdown y almacenamiento. El sistema valida el tipo de archivo, lo convierte y guarda los metadatos.**
Precondiciones	**1. El usuario debe estar autenticado (tener token válido).** **2. El sistema debe tener espacio disponible en almacenamiento.**
Narrativa de cada de uso	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona un archivo (PDF, DOCX, etc.) y asigna un título.	**2. El Frontend (UI) envía el archivo y título mediante POST /documents/upload.**
3. El usuario confirma la subida.	**4. El sistema (API de Documentos) valida el tipo de archivo:** **- Si es válido (PDF/DOCX):** **- Almacena el archivo original en la base de datos.** **- Retorna ruta de almacenamiento (200 OK).**

	- Solicita conversión a Markdown al Servicio Conversor. **- Guarda metadatos (título, ruta, formato) y contenido en Markdown.** **- Confirma almacenamiento y muestra mensaje de éxito.** **- Si no es válido (ej. .EXE):** **- Rechaza la subida (400 Bad Request).** **- Muestra error: *"Formato no soportado"*.**
5. Si la conversión es exitosa, el sistema redirige al usuario a la vista de documentos	

RF-004- Convertir a Markdown

Convertir a Markdown	
Tipo	**Obligatorio**
Autor(es)	**Elvis Ronald Leyva Sardon**
Actores	**Usuario Registrado, Sistema**
Descripción	**Permite a un usuario convertir un documento subido (PDF, DOCX, etc.) a formato Markdown. El sistema valida el archivo, realiza la conversión y almacena el resultado.**

Precondiciones	**1. El usuario debe estar autenticado (token JWT válido).** **2. El documento debe estar previamente subido al sistema.**
Narrativa de cada de uso	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona un documento subido para conversión.	**2. El Frontend envía una solicitud POST /upload/ con el archivo, título y token JWT.**
3. El usuario confirma la acción.	**4. El Controlador (FastAPI) valida el token JWT y busca al usuario en la base de datos:** **- Si el usuario no existe: Responde con error 401 Unauthorized.** **- Si existe: Procede a convertir el documento.**
	5. El sistema (DocumentConverter) intenta convertir el archivo a Markdown: **- Si la conversión es exitosa:** **- Crea un registro en documents con los metadatos y contenido Markdown.** **- Retorna respuesta exitosa (200 OK) con document_id, título y contenido Markdown.** **- Si falla la conversión:** **- Responde con error 500 Internal Server Error y mensaje *"No se pudo convertir el archivo"*.**

RF-005- Descargar documento

Descargar documento	
Tipo	**Obligatorio**
Autor(es)	**Brian Danilo Chite Quispe**
Actores	**Usuario Registrado, Sistema**
Descripción	**Permite a un usuario descargar un documento previamente convertido a Markdown, identificado por su document_id.**
Precondiciones	**1. El usuario debe estar autenticado.** **2. El documento debe existir en el sistema.**
Narrativa de cada de uso	
Acción del actor	**Respuesta del sistema**
1. El usuario solicita descargar un documento (haciendo clic en "Descargar").	**2. El Frontend envía una solicitud GET /download/<document_id> al Controlador (FastAPI).**

3. El sistema verifica la existencia del documento:	**4. El Controlador consulta en la Base de Datos (SELECT * FROM documents WHERE document_id = ?):** **- Si el documento existe: Prepara el archivo Markdown (.md) para descarga.** **- Si no existe: Responde con error 404 Not Found y mensaje *"Documento no encontrado"*.**
5. El usuario recibe el archivo o mensaje de error.	

RF-006- Listar documentos

Listar documentos	
Tipo	**Obligatorio**
Autor(es)	**Jerson Roni Chambi Cori**
Actores	**Usuario Registrado, Sistema**
Descripción	**Permite a un usuario visualizar la lista de documentos que ha subido y convertido previamente en el sistema.**
Precondiciones	**1. El usuario debe estar autenticado.** **2. Debe tener al menos un documento subido para ver resultados.**
Narrativa de cada de uso	
Acción del actor	**Respuesta del sistema**

1. El usuario accede a la sección "Mis Documentos".	**2. El Frontend envía una solicitud GET /documents/list/user_id=X (donde X es el ID del usuario).**
3. El sistema verifica la autenticación del usuario.	**4. La API de Documentos consulta en la Base de Datos los documentos asociados al usuario:** **- Si existen documentos:** **- Retorna la lista de documentos con sus metadatos (título, fecha, formato, etc.) en formato JSON (200 OK).** **- El Frontend renderiza la lista con opciones (descargar, eliminar, etc.).** **- Si no hay documentos:** **- Retorna una lista vacía (200 OK).** **- El Frontend muestra el mensaje *"No tienes documentos"*.**
5. El usuario visualiza sus documentos o el mensaje correspondiente.	

RF-007- Gestionar versiones

Gestionar versiones	
Tipo	**Obligatorio**
Autor(es)	**Jaime Elias Flores Quispe**
Actores	**Usuario Registrado, Sistema**

Descripción	**Permite a un usuario visualizar el historial de versiones de un documento específico almacenado en el sistema.**
Precondiciones	**1. El usuario debe estar autenticado (token JWT válido)** **2. El documento debe existir en el sistema**
Narrativa de cada de uso	
Acción del actor	**Respuesta del sistema**
1. El usuario selecciona un documento y solicita ver su historial de versiones.	**2. El Frontend envía solicitud GET /documents/{id}/versions con token JWT.**
3. El sistema verifica la autenticación y permisos del usuario.	**4. El Controlador (FastAPI) consulta las versiones en la Base de Datos:** **SELECT * FROM versions WHERE document_id = ?**
	5. El sistema procesa los resultados: **- Si existen versiones:** **- Retorna lista con metadatos (fecha, descripción) en JSON (200 OK)** **- Frontend muestra lista ordenada cronológicamente** **- Si no hay versiones:** **- Retorna lista vacía (200 OK)** **- Frontend muestra mensaje "No hay versiones disponibles"**

RF-008- Validar formato de archivo

Validar formato de archivo	
Tipo	**Obligatorio**
Autor(es)	**Elvis Ronald Leyva Sardon**
Actores	**Usuario Registrado, Sistema**
Descripción	**Valida que los archivos subidos al sistema tengan formatos permitidos (PDF, DOCX, etc.) antes de su almacenamiento.**
Precondiciones	**1. Usuario autenticado** **2. Archivo seleccionado para carga**
Narrativa de cada de uso	
Acción del actor	**Respuesta del sistema**
1. Usuario selecciona archivo en interfaz	**2. Frontend prepara datos para envío**
3. Usuario confirma carga	**4. Sistema recibe archivo via POST /upload/**
	5. Sistema valida formato: **- Si es válido (PDF/DOCX):** **- Guarda archivo en almacenamiento** **- Registra metadatos en DB** **- Retorna 200 OK + confirmación** **- Si no es válido:** **- Rechaza carga** **- Retorna 400 Bad Request + "Formato no permitido"**

RF-009- Previsualizar archivo generado

Previsualizar archivo generado	
Tipo	**Obligatorio**
Autor(es)	**Brian Danilo Chite Quispe**
Actores	**Usuario Registrado, Sistema**
Descripción	**Permite al usuario previsualizar el contenido de un archivo después de su conversión a Markdown antes de guardarlo definitivamente.**
Precondiciones	**1. Archivo válido cargado** **2. Conversión a Markdown exitosa**
Narrativa de cada de uso	
Acción del actor	**Respuesta del sistema**
1. Usuario carga archivo válido	**2. Sistema convierte archivo a Markdown**
3. Usuario solicita previsualización	**4. Sistema genera vista previa del Markdown**
	5. Sistema muestra: **- Previsualización exitosa:** **- Renderiza contenido en panel especial** **- Muestra botones "Guardar"/"Cancelar"**

	- Error: **- "No se puede previsualizar este archivo"**

RF-010- Mejorar archivo generado

Mejorar archivo generado	
Tipo	**Obligatorio**
Autor(es)	**Jerson Roni Chambi Cori**
Actores	**Usuario Registrado, Sistema**
Descripción	**Permite al usuario optimizar un documento Markdown existente mediante inteligencia artificial, mejorando su estructura, claridad y formato.**
Precondiciones	**1. Documento Markdown existente** **2. Servicio IA disponible**
Narrativa de cada de uso	
Acción del actor	**Respuesta del sistema**
1. Usuario selecciona "Mejorar documento" en un archivo Markdown existente	**2. Frontend envía POST /documents/improve/{document_id}**
	3. Sistema realiza: **- Extrae contenido Markdown original** **- Separa y almacena imágenes temporalmente** **- Envía solo texto al servicio IA.**

	4. Servicio IA procesa y devuelve texto mejorado
	5. Sistema: **- Reintegra imágenes al contenido** **- Crea nueva versión del documento** **- Almacena versión mejorada en DB**
	6. Redirige a vista del documento mejorado.

RF-011- Analizar similitudes entre documentos

Tipo	**Obligatorio**
Autor(es)	**Jerson Roni Chambi Cori**
Actores	**Usuario Registrado, Sistema**
Descripción	**Permite al usuario comparar dos archivos Markdown para obtener un porcentaje de similitud textual mediante el algoritmo de similitud del coseno, útil para detectar duplicación o alto nivel de coincidencia entre documentos.**
Precondiciones	**Ambos documentos deben estar cargados y disponibles en el sistema.** **Ambos archivos deben estar en formato Markdown válido.**
Narrativa de cada de uso	

Acción del actor	Respuesta del sistema
1. Usuario accede a la opción "Analizar similitudes"	**2. El sistema muestra una interfaz para seleccionar dos archivos Markdown.**
3. Usuario selecciona dos archivos	**4. El sistema carga el contenido de ambos archivos.**
5. Usuario hace clic en "Comparar"	**6. El sistema calcula el porcentaje de similitud utilizando el algoritmo de Cosine Similarity.**
	7. El sistema muestra: **- Porcentaje de similitud** **- Interpretación del resultado (Ej. "Muy similares", "Poca similitud")** **- Opción para descargar o guardar el informe si el usuario lo desea**

RF-012- Publicar README en GitHub

Publicar README en GitHub	
Tipo	**Obligatorio**
Autor(es)	**Brian Danilo Chite Quispe**
Actores	**Usuario Registrado, Sistema**
Descripción	**Permite al usuario seleccionar un archivo Markdown previamente generado y publicarlo automáticamente como `README.md` en el repositorio principal de GitHub**

	que el usuario haya autorizado previamente mediante token de acceso personal.
Precondiciones	**El usuario debe haber iniciado sesión en el sistema.** **El usuario debe haber autorizado el acceso a su cuenta de GitHub mediante un token válido.** **El archivo debe estar en formato Markdown.**
Narrativa de cada de uso	
Acción del actor	**Respuesta del sistema**
1. Usuario accede a la opción "Publicar README"	**2. El sistema solicita autenticación con token de GitHub si no ha sido autorizado.**
3. Usuario selecciona el archivo Markdown	**4. El sistema muestra la vista previa del archivo como `README.md`.**
5. Usuario selecciona el repositorio destino	**6. El sistema verifica los permisos de escritura sobre el repositorio seleccionado.**
7. Usuario hace clic en "Publicar"	**8. El sistema realiza la solicitud UPT a la API de GitHub y publica el archivo como `README.md`.**
	9. El sistema muestra un mensaje de confirmación o error según la respuesta de GitHub.

RF-013- Publicar archivo en Wiki de GitHub

Publicar archivo en Wiki de GitHub

Tipo	**Obligatorio**
Autor(es)	**Jerson Roni Chambi Cori**
Actores	**Usuario Registrado, Sistema**
Descripción	**Permite al usuario seleccionar un archivo Markdown y publicarlo como una nueva página dentro del Wiki de un repositorio en GitHub al que tiene acceso autorizado. Esto facilita la documentación técnica directa desde la plataforma sin necesidad de acceso manual a GitHub.**
Precondiciones	**El usuario debe haber iniciado sesión.** **El archivo debe estar en formato Markdown.** **El usuario debe haber autorizado previamente el acceso a su cuenta de GitHub mediante un token.** **El repositorio seleccionado debe tener habilitado el Wiki.**
Narrativa de cada de uso	
Acción del actor	**Respuesta del sistema**
1. Usuario accede a la opción "Publicar en Wiki"	**2. El sistema solicita o valida el token de acceso a GitHub.**
3. Usuario selecciona el archivo Markdown	**4. El sistema muestra una vista previa del archivo como una página Wiki.**
5. Usuario selecciona el repositorio destino	**6. El sistema lista los repositorios con Wiki habilitado y permisos de escritura.**

7. Usuario confirma publicación	**8. El sistema publica el archivo como página en el Wiki del repositorio a través de la API de GitHub.**
	9. El sistema muestra un mensaje de éxito o error según la respuesta de la API.

RF-014- Publicar múltiples archivos y generar índice en Wiki

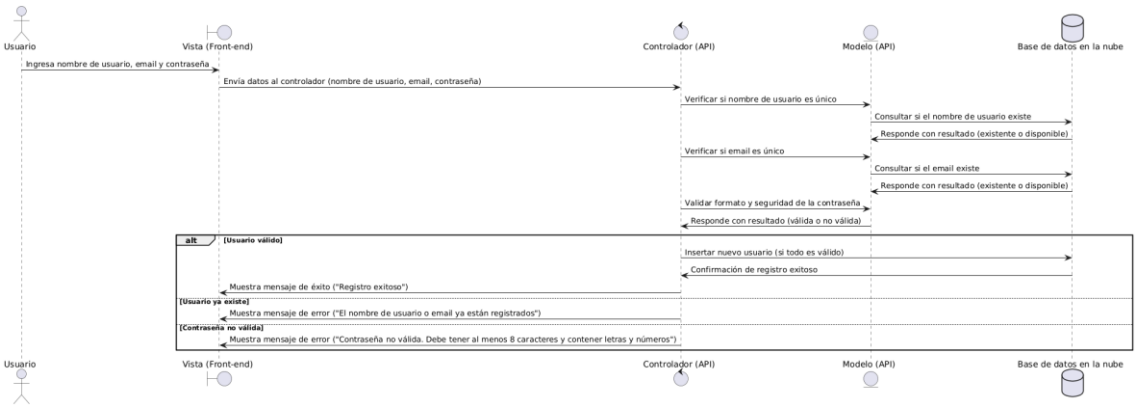
Publicar múltiples archivos y generar índice en Wiki	
Tipo	**Obligatorio**
Autor(es)	**Brian Danilo Chite Quispe**
Actores	**Usuario Registrado, Sistema**
Descripción	**Permite al usuario seleccionar varios archivos en formato Markdown y publicarlos como páginas separadas en el Wiki de un repositorio GitHub autorizado. Además, el sistema generará automáticamente una página de inicio (Home) con un índice enlazado a cada documento subido, facilitando la navegación y organización del contenido.**
Precondiciones	**El usuario debe haber iniciado sesión en el sistema.** **El usuario debe haber autorizado acceso a su cuenta de GitHub con un token válido.** **El repositorio debe tener habilitado el Wiki.** **Los archivos deben estar previamente convertidos a formato Markdown.**
Narrativa de cada de uso	

Acción del actor	Respuesta del sistema
1. Usuario accede a la opción “Publicar múltiples archivos en Wiki”	**1. El sistema valida el token de GitHub y lista los repositorios disponibles.**
2. Usuario selecciona uno o más archivos Markdown	**2. El sistema muestra los archivos seleccionados y solicita el nombre del índice (opcional).**
3. Usuario confirma la acción	**3. El sistema publica cada archivo como una página independiente en el Wiki.**
	4. El sistema genera una página `Home.md` con enlaces a cada archivo publicado.
	5. El sistema muestra un mensaje de éxito o error con el enlace al Wiki actualizado.

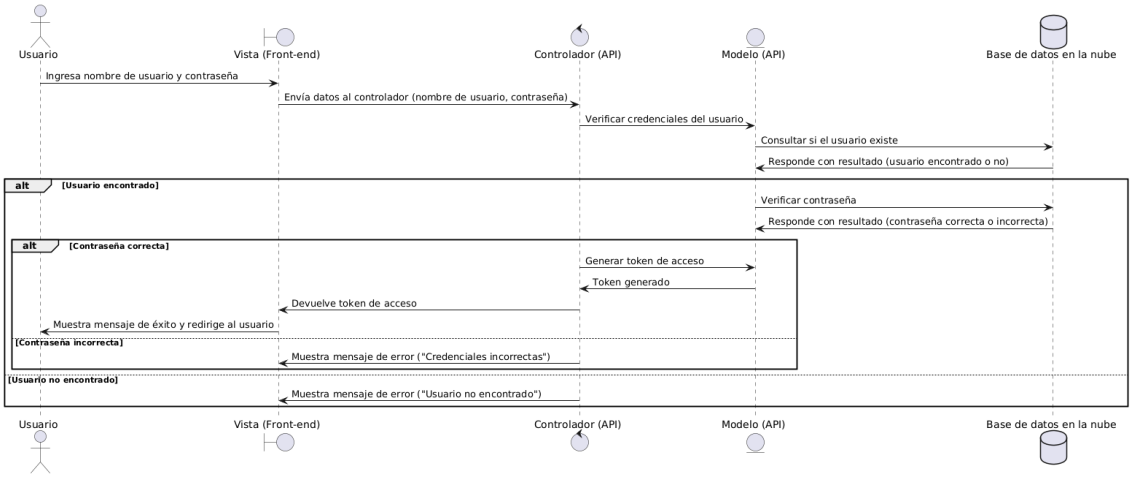
3. Modelo Lógico

a) Diagrama de Secuencia

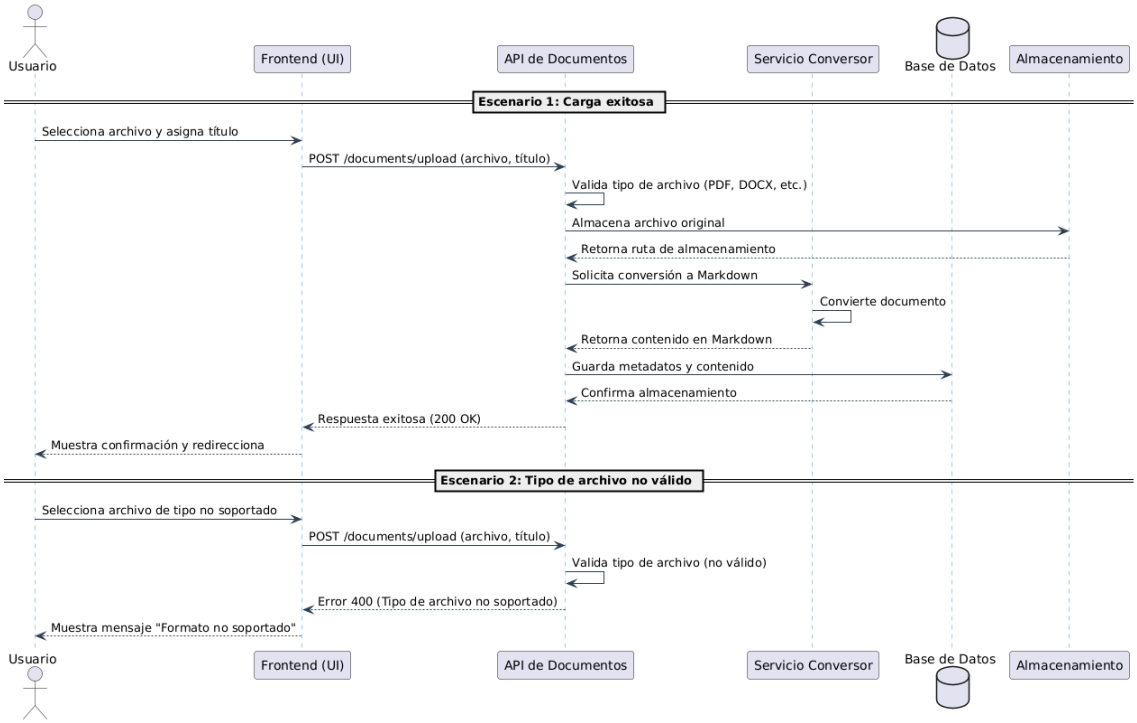
RF01-Registrar usuario



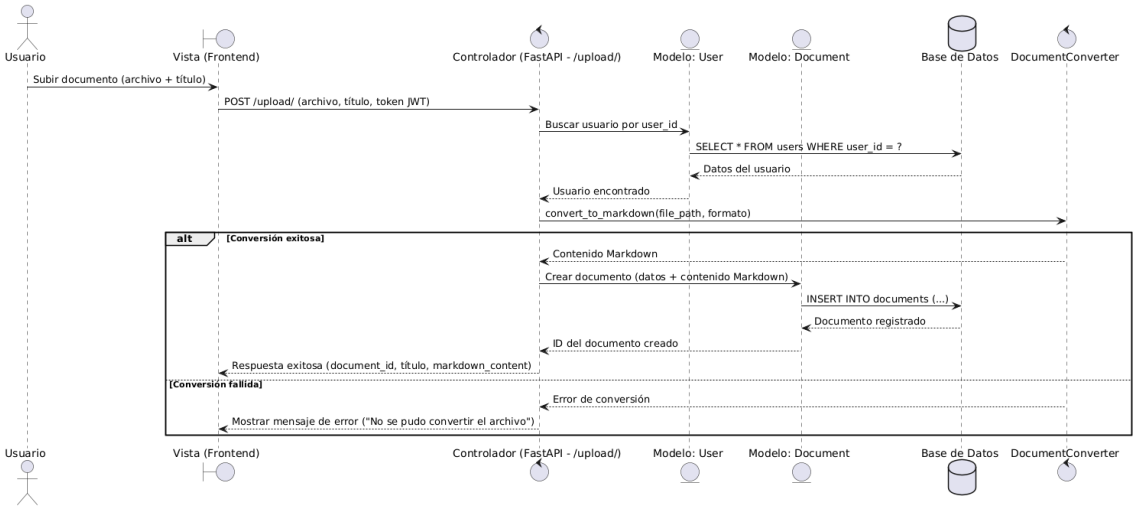
RF02-Iniciar sesión



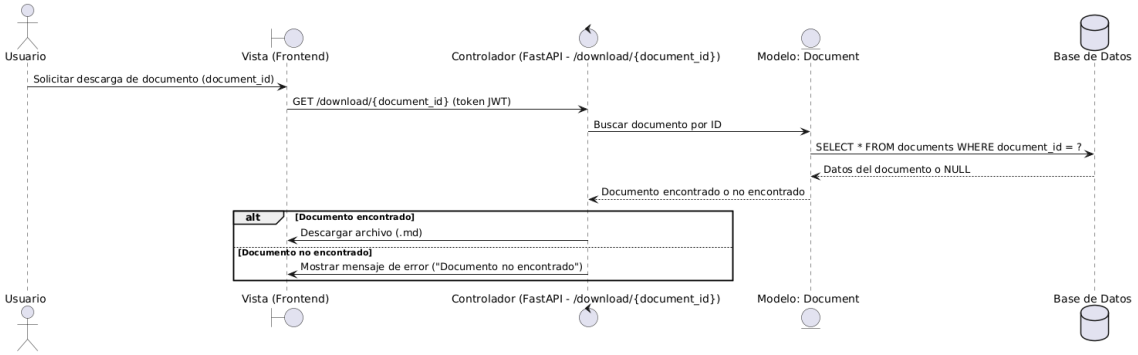
RF03 Subir documento



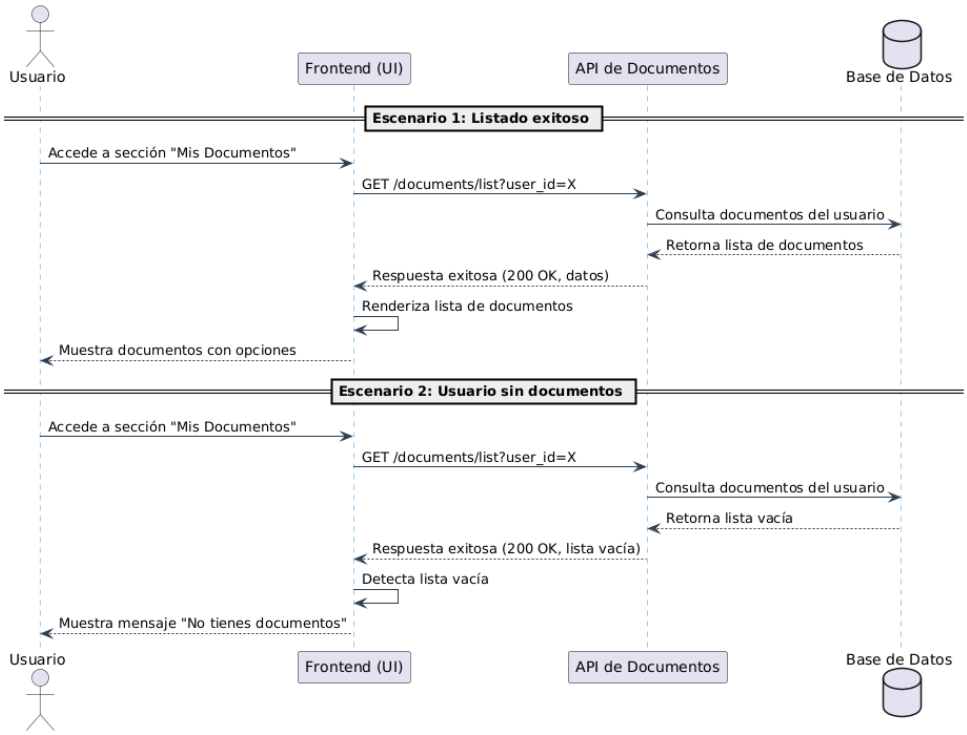
RF04 Convertir a Markdown



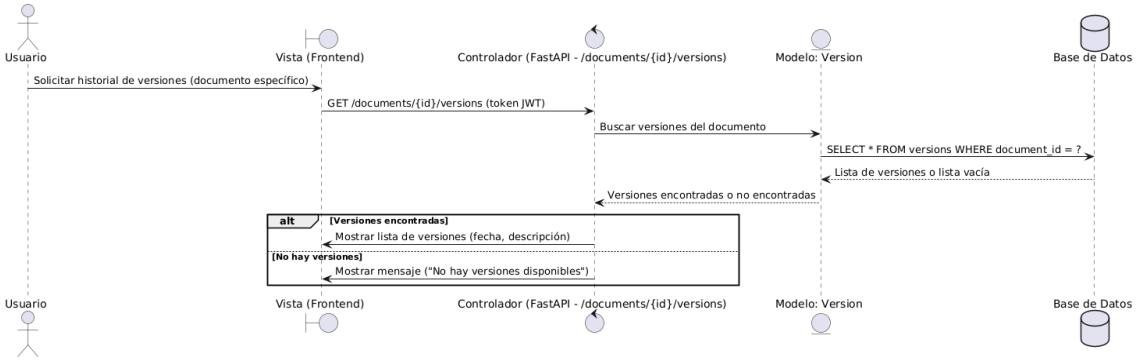
RF05 Descargar documento



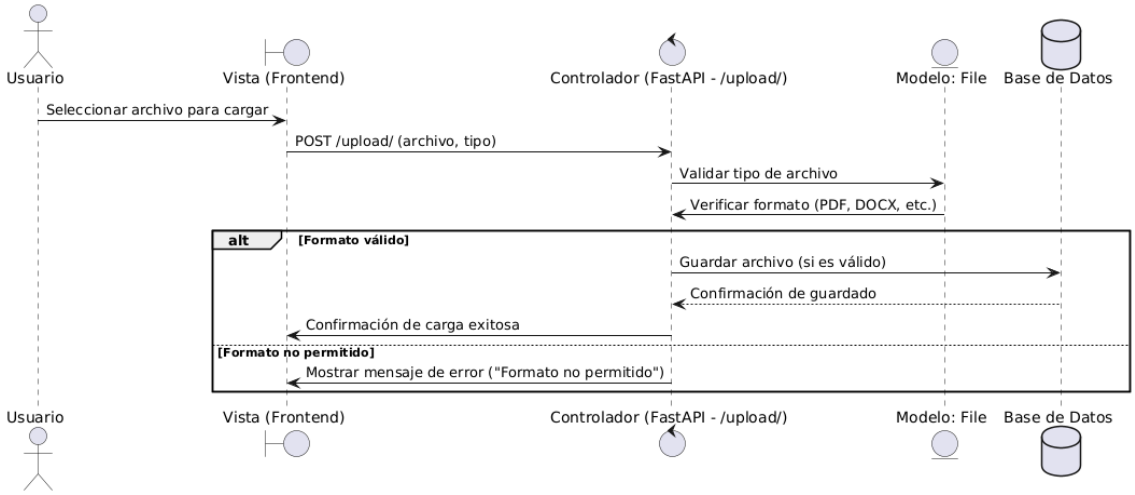
RF06 Listar documentos



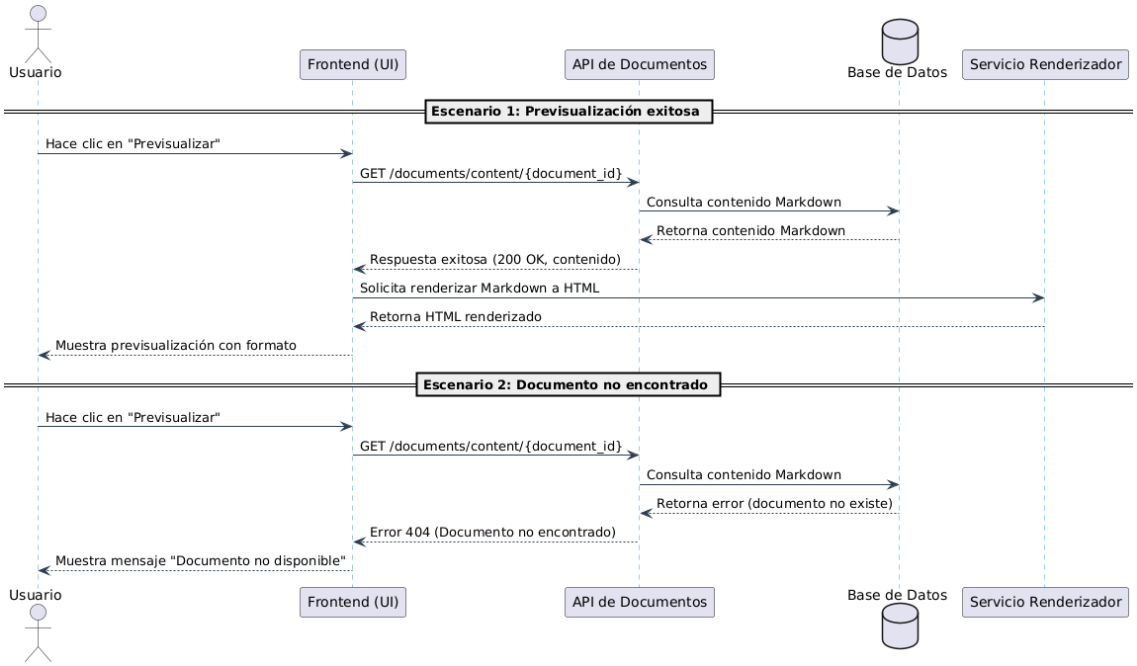
RF07 Gestionar versiones



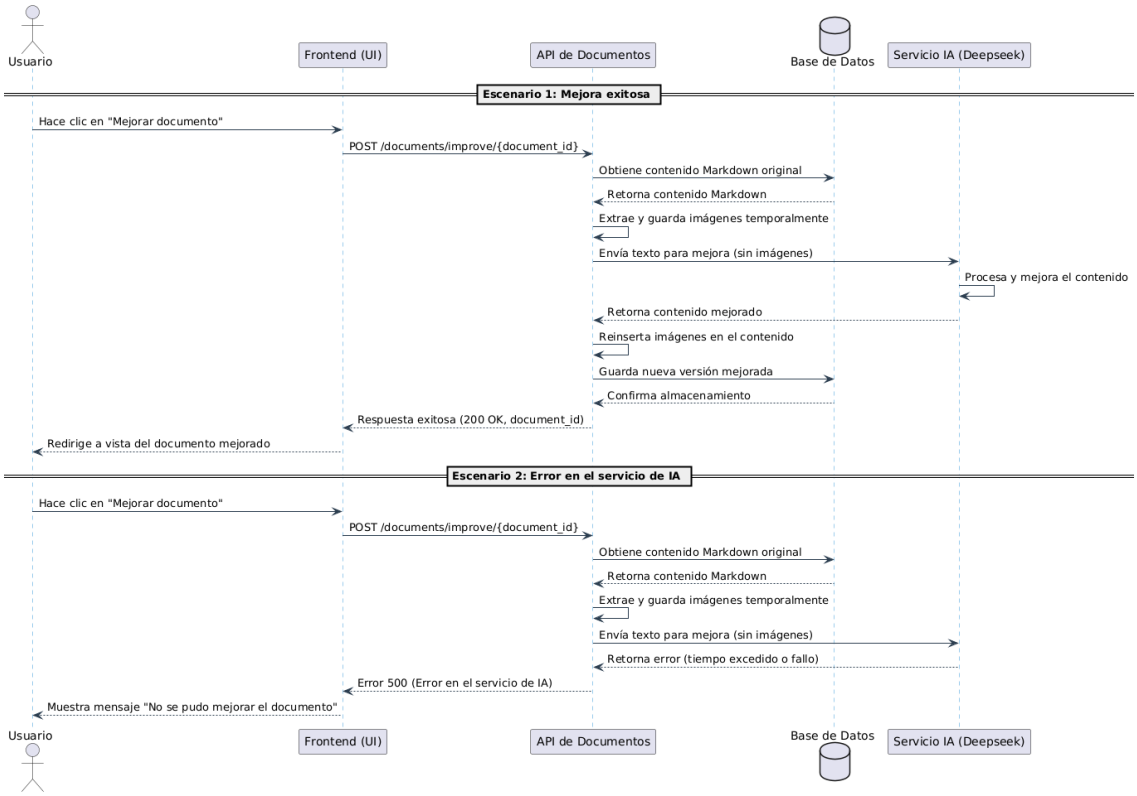
RF08 Validar formato de archivo



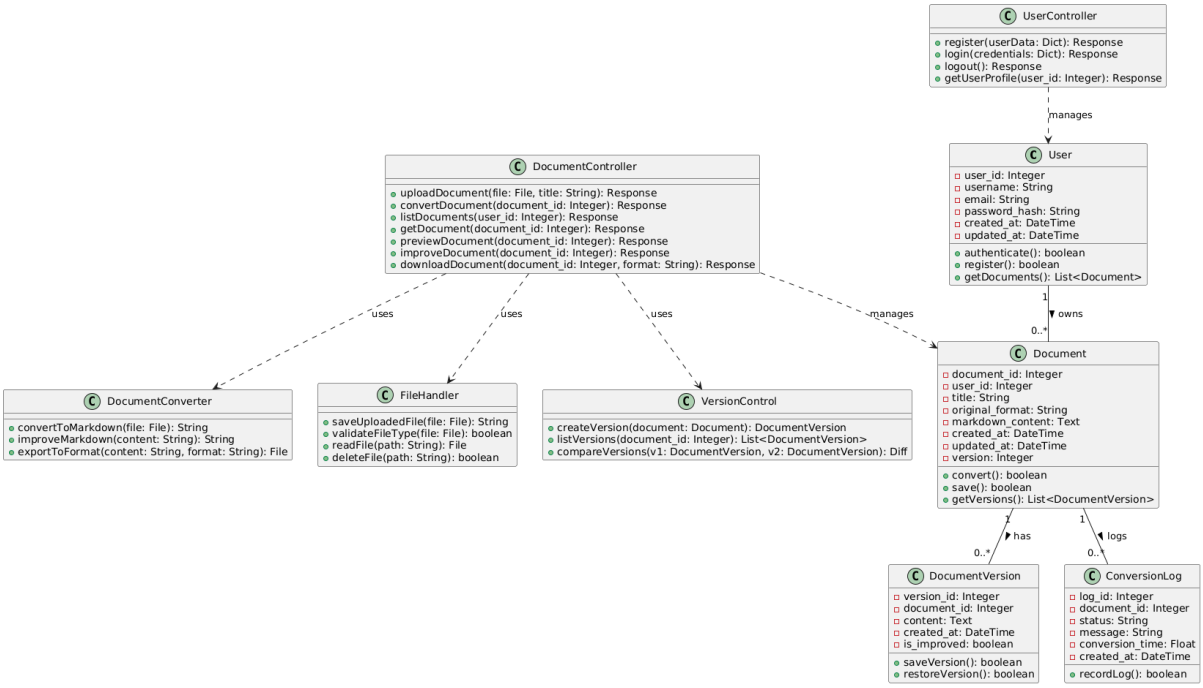
RF09 Previsualizar archivo generado



RF10 Mejorar archivo generado



d) Diagrama de Clases



CONCLUSIONES

- El sistema de autenticación cumple con los requisitos básicos de seguridad, incluyendo validación de credenciales, bloqueo tras múltiples intentos fallidos y generación de tokens JWT con expiración controlada, garantizando acceso seguro al sistema.
- La integración de mensajes claros en caso de errores (como credenciales incorrectas o cuentas bloqueadas) ayuda a los usuarios a entender y resolver problemas sin necesidad de soporte externo.
- El uso de bcrypt para el hashing de contraseñas y JWT para la gestión de sesiones asegura que los datos sensibles estén protegidos contra accesos no autorizados y ataques comunes como fuerza bruta.
- La estructura del flujo de autenticación permite una fácil integración con futuros servicios (como autenticación multifactor o SSO) sin requerir cambios mayores en la arquitectura actual.
- El sistema lleva un historial de intentos de inicio de sesión, facilitando la auditoría y detección de posibles brechas de seguridad o patrones de uso inusuales.

RECOMENDACIONES

- Se sugiere implementar un sistema de monitoreo en tiempo real para la conversión de documentos extensos, permitiendo al usuario conocer el progreso durante el procesamiento.
- Resultaría valioso integrar un sistema de notificaciones que alerte a los usuarios sobre acciones importantes como finalización de conversiones o mejoras aplicadas por la IA.
- Para garantizar la robustez del sistema frente al crecimiento de usuarios, se recomienda implementar una estrategia de escalado horizontal en la infraestructura definida con Terraform.
- La seguridad podría reforzarse implementando autenticación de dos factores y mejorando el sistema de permisos para documentos compartidos entre múltiples usuarios.
- Por último, sería beneficioso desarrollar una API pública documentada que permita a terceros integrar las capacidades de conversión y mejora de documentos en sus propias aplicaciones, ampliando así el alcance y utilidad de Doc2Markdown.

BIBLIOGRAFÍA

Ramírez Jiménez, Ó. (2021). Python a fondo: (1 ed.). Marcombo. https://elibro.net/es/lc/bibliotecaupt/titulos/280038

Nolasco Valenzuela, J. S. (2018). Python: aplicaciones prácticas: (ed.). RA-MA Editorial. https://elibro.net/es/lc/bibliotecaupt/titulos/106523

Moreno Muñoz, A. & Córcoles Córcoles, S. (2019). Python práctico: Herramientas, conceptos y técnicas: (1 ed.). RA-MA Editorial. https://elibro.net/es/lc/bibliotecaupt/titulos/222728

Jiménez de Parga, C. (2021). UML: arquitectura de aplicaciones en Java, C++ y Python: (2 ed.). RA-MA Editorial. https://elibro.net/es/lc/bibliotecaupt/titulos/222720