Universidad Autónoma de Occidente

DOCUMENTO DE INGENIERÍA

Jonathan López Londoño

Ingeniería de Software 2

Sebastian Medina Garcia
Alfonso Murillo Lora

2025

Introducción

El presente documento de ingeniería tiene como finalidad describir los aspectos técnicos, funcionales y estructurales de Sebbi AI, una plataforma de escritura asistida por inteligencia artificial que tiene como objetivo principal brindar a los usuarios una herramienta capaz de generar, mejorar y asistir en la redacción de contenido textual a partir de indicaciones personalizadas por medio de lenguaje natural.

Por lo tanto, a continuación, se establecen los requisitos que guían su construcción, así como los modelos y representaciones necesarias para garantizar una comprensión clara y completa del diseño, la arquitectura y el comportamiento del software.

Este documento proporciona los elementos necesarios para analizar, desarrollar, mantener y validar el sistema, asegurando que cumpla con los estándares de calidad requeridos y permita su escalabilidad y evolución en futuras etapas del proyecto.

Requisitos Funcionales

Los requerimientos funcionales describen el comportamiento específico que debe tener el sistema para cumplir con los objetivos del negocio. Estos requerimientos detallan las funciones, procesos, reglas y validaciones que la solución debe implementar para dar soporte a las actividades de los usuarios. Su propósito es definir con claridad qué debe hacer el sistema, sin entrar aún en detalles técnicos sobre cómo se implementará.

En esta sección se presentan los requerimientos funcionales identificados durante las sesiones de análisis, los cuales reflejan las necesidades expresadas por los usuarios y demás partes interesadas. Cada requerimiento está orientado a asegurar que el sistema proporcione las capacidades necesarias para operar de manera eficiente y alineada con los procesos del negocio.

Link directo:

https://docs.google.com/spreadsheets/d/1hmSDW8DAvgtkI9Oz8Ga7Eypnulumyfdk/edit?usp=sharing&ouid=116325971939745170714&rtpof=true&sd=true

Número de requisito	RF001					
Nombre de requisito	Crear documentos					
Descripción	El sistema debe permitir al usuario crear nuevos documentos que serán guardados en el perfil del usuario					
Tipo	Requisito	X	Restricción			
Fuente del requisito	Usuario final					
Prioridad del requisito	Alta/Esencial	X	Media/Deseado		Baja/Opcional	

Número de requisito	RF002					
Nombre de requisito	Iniciar sesión					
Descripción	El sistema debe permitir a los usuarios autenticarse mediante correo y contraseña para acceder a sus documentos.					
Tipo	Requisito	X	Restricción			
Fuente del requisito	Usuario final					
Prioridad del requisito	Alta/Esencial	X	Media/Deseado		Baja/ Opcional	

Número de requisito	RF003					
Nombre de requisito	Chat con IA					
Descripción	El sistema debe permitir al usuario desplegar un chat de IA para hacerle cualquiera pregunta al asistente de IA					
Tipo	Requisito	X	Restricción			
Fuente del requisito	Licitaciones					
Prioridad del requisito	Alta/Esencial	X	Media/Deseado		Baja/ Opcional	

Número de requisito	RF004					
Nombre de requisito	Importar Fuentes					
Descripción	El sistema debe permitir al usuario importar documentos como fuentes para ser citadas					
Tipo	Requisito	X	Restricción			
Fuente del requisito	Licitaciones					
Prioridad del requisito	Alta/Esencial	X	Media/Deseado		Baja/ Opcional	

Número de requisito	RF005					
Nombre de requisito	Agregar pdf como contexto.					
Descripción	El sistema debe permitir al usuario darle contexto al chat de IA mediante los pdf importados para obtener respuestas en base a la información dentro de ellos.					
Tipo	Requisito	X	Restricción			
Fuente del requisito	Usuario Final					
Prioridad del	Alta/Esencial	X	Media/Deseado		Baja/ Opcional	

Número de requisito	RF006					
Nombre de requisito	Exportar documentos					
Descripción	El sistema debe permitirle al usuario exportar los documentos en los que está trabajando					
Tipo	Requisito	X	Restricción			
Fuente del requisito	Licitaciones					
Prioridad del requisito	Alta/Esencial	X	Media/Deseado		Baja/ Opcional	

Número de requisito	RF007					
Nombre de requisito	Subir imágenes					
Descripción	El sistema debe permitir al usuario agregar imágenes a sus documentos					
Tipo	Requisito	X	Restricción			
Fuente del requisito	Usuario final					
Prioridad del requisito	Alta/Esencial	X	Media/Deseado		Baja/ Opcional	

Número de requisito	RF008					
Nombre de requisito	Personalización de documento					
Descripción	El sistema debe permitir al usuario personalizar el texto e imágenes de sus documentos cambiandoles propiedades clave mediante una barra de herramientas.					
Tipo	Requisito	X	Restricción			
Fuente del requisito	Licitaciones					
Prioridad del	Alta/Esencial	X	Media/Deseado		Baja/ Opcional	

Número de	RF009					
Nombre de	Visualización de documentos					
Descripción	El sistema debe permitir al usuario ver y buscar todos los documentos que ha creado hasta el momento					
Tipo	Requisito	**X**		Restricción		
Fuente del requisito	Licitaciones					
Prioridad del	Alta/Esencial	**X**	Media/Deseado		Baja/ Opcional	

Número de requisito	RF010					
Nombre de requisito	Autocompletado Inteligente					
Descripción	El sistema debe hacer sugerencias contextuales en tiempo real para continuar la redacción					
Tipo	Requisito	**X**		Restricción		
Fuente del requisito	Licitaciones					
Prioridad del requisito	Alta/Esencial		Media/Desead	**X**	Baja/ Opcional	

Atributos de calidad

Los atributos de calidad, también conocidos como requerimientos no funcionales, definen las características generales que debe tener el sistema para garantizar una experiencia de uso satisfactoria, un desempeño adecuado y una operación confiable en su entorno de ejecución. A diferencia de los requerimientos funcionales, estos no describen funciones específicas del sistema, sino propiedades que influyen en cómo se comportan esas funciones bajo diversas condiciones.

En esta sección se presentan los requerimientos no funcionales de Sebbi, identificados durante el análisis, los cuales reflejan expectativas y restricciones técnicas, operativas y de negocio.

Link directo:

https://docs.google.com/spreadsheets/d/1BkAX0GMlSr79FTBBWQmwJ2AXEL79DH32/edit?usp=sharing&ouid=116325971939745170714&rtpof=true&sd=true

Escenario de Calidad No.	1	**Atributo de Calidad**	Disponibilidad	**Actor**	Usuario final
Fuente	Usuarios de la plataforma.				
Estímulo	Acceso a la plataforma durante horarios no laborables				
Ambiente	Usuario intenta iniciar sesión un domingo a las 2:00 a.m.				
Respuesta	Plataforma responde y permite acceso sin errores				
Medida de la Respuesta	Operación exitosa en el 99.9% de los ingresos.				

Escenario de Calidad No.	2	Atributo de Calidad	Usabilidad	Actor	Nuevo usuario
Fuente	Nuevos registros en la plataforma				
Estímulo	Primer ingreso al sistema y uso del editor				
Ambiente	Plataforma en estado inicial de sesión				
Respuesta	Usuario puede crear un nuevo documento y usar funciones básicas sin entrenamiento				
Medida de la Respuesta	Tiempo de aprendizaje inferior a 10 minutos sin uso de un tutorial				

Escenario de Calidad No.	3	Atributo de Calidad	Concurrencia	Actor	Usuario final
Fuente	Usuarios activos simultáneamente				
Estímulo	Ingreso a la plataforma.				
Ambiente	500 usuarios usando el editor al mismo tiempo				
Respuesta	Información requerida por el usuario desplegada en la pantalla.				
Medida de la Respuesta	Tiempo promedio de respuesta menor a 2 segundos por petición				

Escenario de Calidad No.	4	Atributo de Calidad	Seguridad	Actor	Desarrollador
Fuente	Usuarios de la plataforma				
Estímulo	Intento de acceso a contenido por parte de un usuario no autorizado				
Ambiente	Usuario sin permisos intenta acceder a documentos que no le pertenecen				
Respuesta	Acceso denegado				
Medida de la Respuesta	100% de bloqueos de accesos no autorizados				

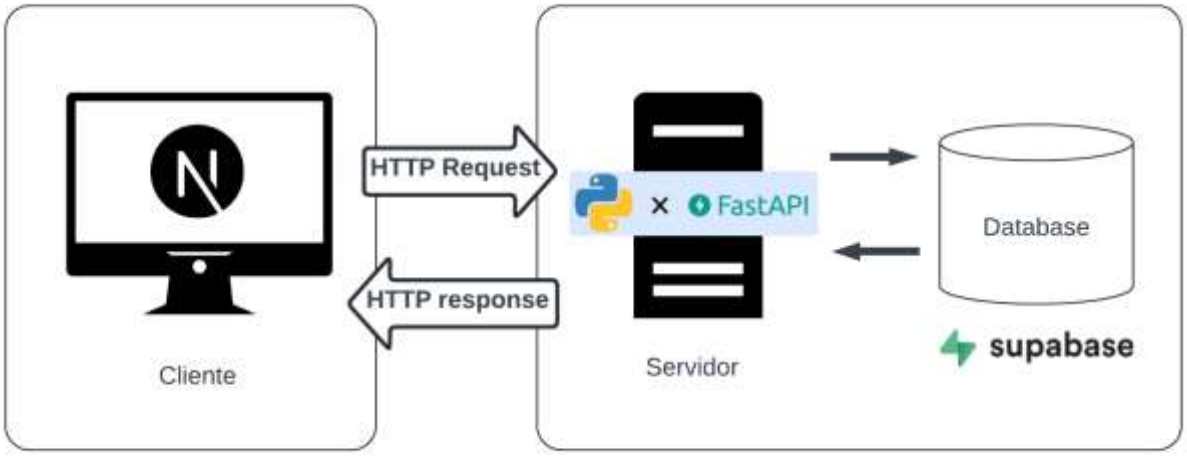
Escenario de Calidad No.	5	Atributo de Calidad	Portabilidad	Actor	Usuario Final
Fuente	Usuarios				
Estímulo	Acceso desde diferentes navegadores				
Ambiente	Usuario abre el sistema desde Chrome, Brave, Safari y Firefox				
Respuesta	Plataforma se adapta y funciona correctamente				
Medida de la Respuesta	Compatibilidad con al menos 5 navegadores				

Escenario de Calidad No.	6	Atributo de Calidad	Rendimiento	Actor	Usuario Final
Fuente	Usuarios de la plataforma				
Estímulo	Carga de archivos				
Ambiente	Pdf importado de hasta 25MB				
Respuesta	El archivo es cargado correctamente.				
Medida de la Respuesta	Tiempo de subida no mayor a (1) minuto.				

Escenario de Calidad No.	7	Atributo de Calidad	Rendimiento	Actor	Usuario final
Fuente	Usuarios de la plataforma				
Estímulo	Consulta de documentos creados.				
Ambiente	Sidebar para consultar documentos creados				
Respuesta	Todos los documentos relacionados con el perfil del usuario.				
Medida de la Respuesta	Todos los documentos del usuario fueron cargados en menos de 10 segundos.				

Diagrama de arquitecturas

Un diagrama de arquitectura es una representación visual que muestra cómo se organizan y se comunican los diferentes componentes de un sistema de software. Este tipo de diagrama ayuda a entender la estructura general del sistema, incluyendo el flujo de datos, las tecnologías utilizadas y la relación entre el cliente, el servidor y otros elementos clave.



En este diagrama el cliente, desarrollado con Next.js, se comunica mediante peticiones HTTP con el servidor backend construido con FastAPI y Python. El servidor recibe las solicitudes, las procesa y accede a la base de datos alojada en Supabase para consultar o modificar la información requerida. Luego, el servidor envía una respuesta HTTP de vuelta al cliente con los datos solicitados o el resultado de la operación.

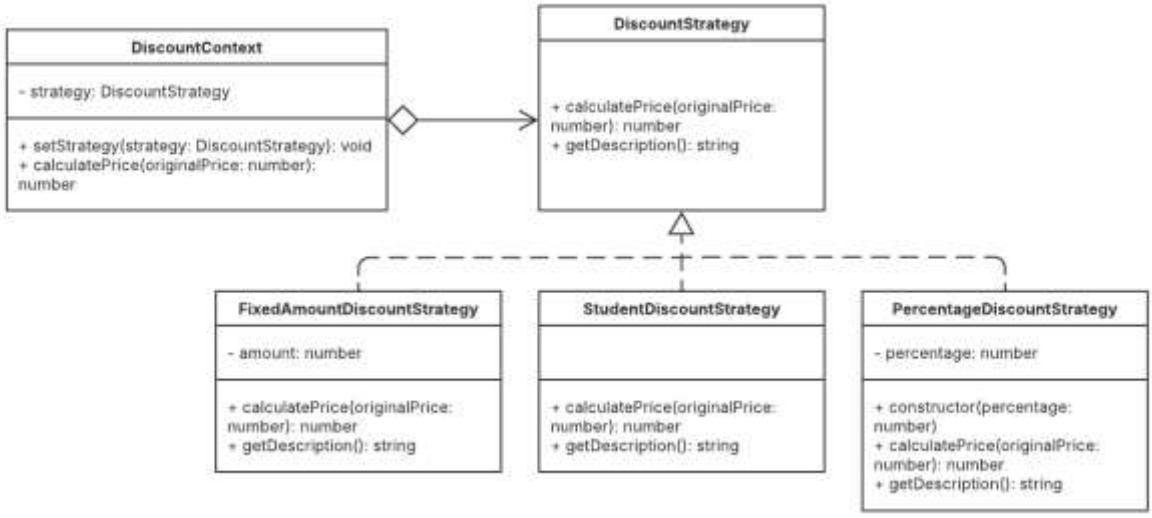
Diagrama de Base de datos

El diagrama de base de datos representa la estructura lógica que soporta el almacenamiento, organización y acceso a la información dentro del sistema. Este modelo describe las entidades principales, sus atributos y las relaciones entre ellas, proporcionando una visión clara de cómo se gestionan los datos que sustentan las funcionalidades del sistema.

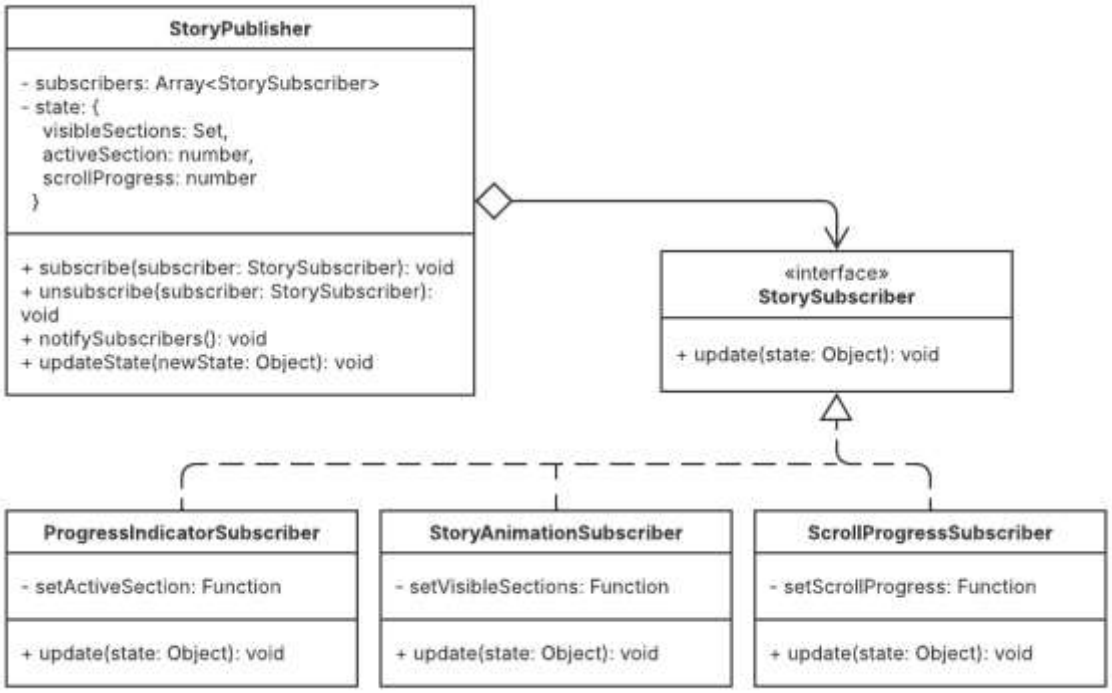
La tabla users almacena los datos de los usuarios registrados (nombre, correo, contraseña). Cada usuario puede crear múltiples documentos (documents), que contienen el contenido generado o editado en la aplicación, junto con fechas de creación y actualización. Además, los usuarios pueden subir archivos PDF (pdfs), en esa tabla se guarda el enlace al archivo y su ubicación en el almacenamiento (bucket_path). Las relaciones entre las tablas aseguran que cada documento o PDF esté vinculado a su respectivo dueño mediante el campo owner_id.

Diagramas UML

Los diagramas UML (Lenguaje Unificado de Modelado) son representaciones gráficas que se utilizan para visualizar, especificar, construir y documentar los elementos de un sistema de software. Se emplean para describir la estructura y el comportamiento del sistema. Estos diagramas ayudan a entender cómo se organizan los componentes del sistema y cómo interactúan entre sí, lo que mejora el diseño, la planificación y el mantenimiento del software. A continuación, se mostrarán los correspondientes a las soluciones realizadas mediante patrones de diseño.



Este diagrama utiliza el Patrón de Estrategia para aplicar diferentes tipos de descuentos de forma flexible y modular. La clase “DiscountContext” actúa como el contexto que mantiene una referencia a un objeto de tipo “DiscountStrategy”, el cual define una interfaz común con métodos para calcular el precio con descuento y obtener su descripción. Las clases concretas como “FixedAmountDiscountStrategy”, “StudentDiscountStrategy” y “PercentageDiscountStrategy” implementan esta interfaz con distintas lógicas de descuento (por monto fijo, por ser estudiante o por porcentaje). Así, el sistema puede cambiar dinámicamente la forma en que se calcula el descuento simplemente reemplazando la estrategia usada en el contexto, sin necesidad de modificar el resto del código.

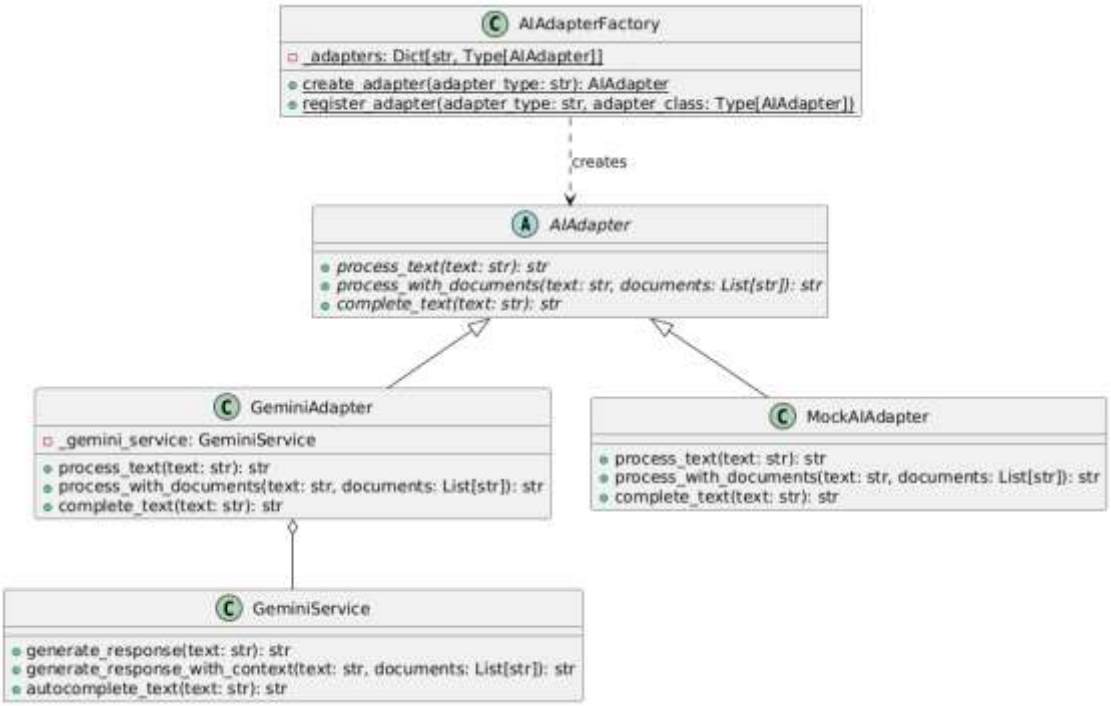


Este diagrama aplica el Patrón de Observador (Observer Pattern) para manejar actualizaciones de estado en una interfaz de usuario basada en la historia del equipo desarrollador (puede verse en la aplicación desplegada).

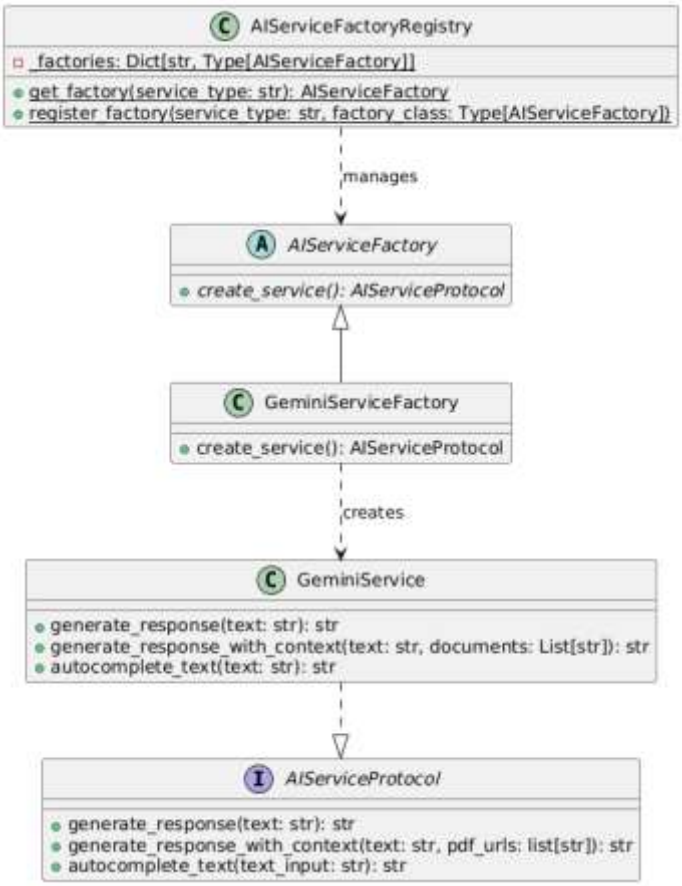
La clase StoryPublisher actúa como el sujeto que mantiene un estado interno (incluyendo secciones visibles, sección activa y progreso de scroll), y una lista de suscriptores que implementan la interfaz StorySubscriber. Cada vez que el estado cambia mediante updateState(newState), el StoryPublisher llama al método notifySubscribers(), que a su vez ejecuta update(state) en cada suscriptor.

Los suscriptores concretos (ProgressIndicatorSubscriber, StoryAnimationSubscriber y ScrollProgressSubscriber) implementan su propia lógica en update para reaccionar a los cambios según su función: actualizar indicadores visuales, gestionar animaciones o reflejar el progreso del scroll. Este patrón permite desacoplar el emisor del estado (publisher) de los elementos que lo consumen (subscribers), facilitando un sistema reactivo, modular y extensible.

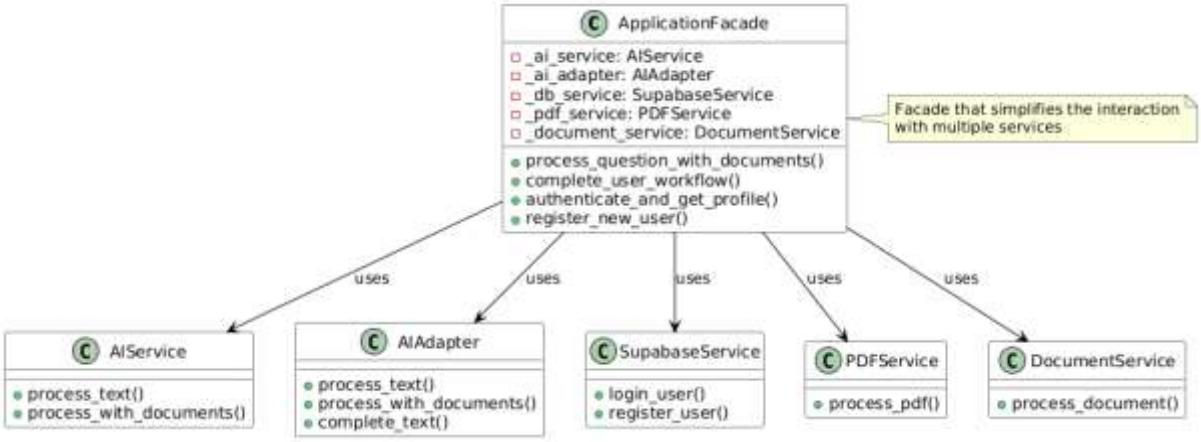
El diagrama UML representa la clase TourService, la cual implementa el patrón Singleton. En el diagrama se observa un atributo privado instance de tipo TourService, que almacena la instancia única de la clase. Además, contiene un segundo atributo privado driverObj, que representa un objeto proveniente de la biblioteca driver.js, utilizado para gestionar el tour interactivo de Sebbi. La clase expone dos métodos públicos: startTour(), que inicia un recorrido o guía interactiva, y destroy(), que finaliza dicha guía. La flecha de retorno al mismo bloque indica la auto-referencia característica del patrón Singleton, reforzando que la clase se instancia una sola vez y proporciona un punto de acceso global a su funcionalidad.



Este diagrama implementa el Patrón Adapter Las clases GeminiAdapter y MockAIAdapter son implementaciones concretas de esta interfaz: GeminiAdapter delega las operaciones al servicio GeminiService, que es responsable de generar respuestas, respuestas con contexto y autocompletar texto. MockAIAdapter, por otro lado, podría usarse para pruebas o entornos controlados.



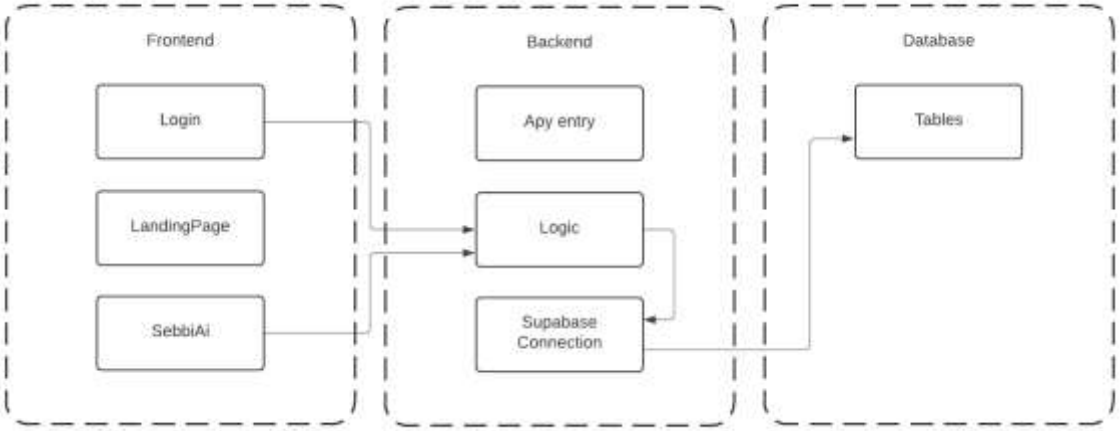
El segundo diagrama muestra cómo se organiza la creación de servicios de inteligencia artificial usando el patrón de fábrica. La clase AIServiceFactoryRegistry actúa como un registro central que almacena distintas fábricas según el tipo de servicio. Estas fábricas, como GeminiServiceFactory, implementan una interfaz común (AIServiceFactory) que define un método para crear servicios concretos, como GeminiService. Este último implementa el protocolo AIServiceProtocol, que asegura que todos los servicios tengan los mismos métodos: generar respuestas, usar contexto y autocompletar texto.



Este diagrama representa el patrón Facade a través de la clase ApplicationFacade, que actúa como un punto de entrada simplificado para interactuar con múltiples servicios del sistema. La fachada coordina y utiliza internamente cinco servicios especializados: AIService, AIAdapter, SupabaseService, PDFService y DocumentService, cada uno con responsabilidades específicas como el procesamiento de texto, gestión de usuarios y manejo de documentos. Gracias a esta estructura, el usuario o cliente del sistema puede acceder a funcionalidades complejas mediante métodos simples como process_question_with_documents() o register_new_user().

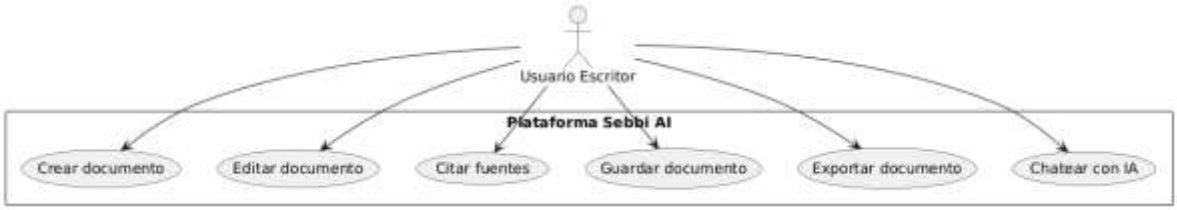
Diagramas de componentes

El diagrama de componentes de software es una herramienta visual que se utiliza para mostrar la estructura estática de un sistema, enfocándose en sus componentes principales y cómo se interconectan entre sí. Este diagrama es especialmente útil para entender la organización interna de una aplicación, ya que describe los componentes de software y sus interfaces, relaciones y dependencias.



Diagramas de casos de uso

El diagrama de casos de uso es una herramienta utilizada en el desarrollo de software para representar de manera gráfica las interacciones entre los usuarios (o actores) y el sistema. Su principal función es identificar las funcionalidades que el sistema debe ofrecer y cómo estos actores interactúan con ellas. Este diagrama ayuda a entender los requisitos del sistema desde la perspectiva del usuario, permitiendo que los desarrolladores, analistas y otras partes interesadas visualicen los objetivos que el software debe cumplir.



Descripción de casos de uso extendidos

En la descripción de casos de uso se utilizará una plantilla con la cual se especificarán todos los detalles necesarios para que se cumplan de una manera adecuada. A continuación, los casos de uso más importantes de Sebbi AI.

Caso de Uso	Crear documento
Referencia	CU-001
Creado por	Sebastian Medina Garcia
Fecha de Creación	15/5/2021
Información General	
Requisito(s) Asociado(s)	RF001
Actor	Usuario Escritor
Descripción	Este caso de uso permite al usuario iniciar un nuevo documento para comenzar la redacción de un texto desde cero.
PRECONDICION	El usuario debe estar autenticado en el sistema.
	El usuario debe estar en la pagina principal
POSTCONDICION	Se crea un nuevo documento en blanco y se asocia al perfil del usuario.
Flujo normal de los eventos	
ACCION DE LOS ACTORES	**RESPUESTA DEL SISTEMA**
1. Ingresa al sistema con sus credenciales	Valida credenciales y muestra la interfaz principal
2. Selecciona la opción "Nuevo documento"	3. Muestra ventana emergente para configurar nuevo documento
4. Introduce el nombre del documento	5. Registra datos y habilita botón "Crear"
6.Presiona "Crear"	7. Genera el nuevo documento y lo abre en el editor de texto
Flujos alternos	
4. Si el nombre del documento está vacío o contiene caracteres inválidos, se muestra un mensaje de error.	

Caso de Uso	Editar documento
Referencia	CU-002
Creado por	Sebastian Medina Garcia
Fecha de Creación	29/5/2025
Información General	
Requisito(s) Asociado(s)	RF002
Actor	Usuario escritor
Descripción	Este caso de uso permite al usuario modificar el contenido de un documento previamente creado, ya sea para añadir, eliminar o reformular el texto.
PRECONDICION	El Usuario debe estar logueado en la plataforma
	El Usuario debe tener al menos un documento creado
POSTCONDICION	Los cambios realizados se guardan en el documento.
Flujo normal de los eventos	
ACCION DE LOS ACTORES	**RESPUESTA DEL SISTEMA**
1. Accede a la lista de documentos creados	2. Carga los documentos del usuario
3. Selecciona un documento de la lista	4. Abre el documento en el editor de texto
5. Realiza modificaciones (escribir, borrar)	6. Actualiza en tiempo real el contenido en el área de texto
	7.Valida y guarda los cambios en la base de datos
Flujos alternos	
1. Si no hay documentos existentes, el sistema muestra un mensaje informando que no hay documentos disponibles para editar.	

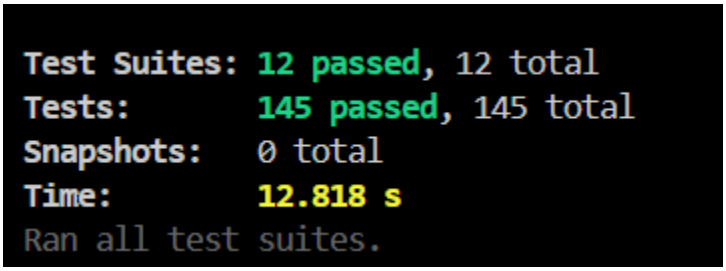
Caso de Uso	Citar Fuentes
Referencia	CU-003
Creado por	Sebastian Medina Garcia
Fecha de Creación	30/5/2025
Información General	
Requisito(s) Asociado(s)	RF003
Actor	Usuario escritor
Descripción	Este caso de uso permite al usuario insertar citas dentro del texto, en estilos como APA, mediante el uso de un asistente de citación.
PRECONDICION	El usuario debe estar autenticado y debe tener un documento abierto en el editor.
POSTCONDICION	La cita se inserta correctamente en el documento, y se guarda su referencia en la sección de bibliografía.
Flujo normal de los eventos	
ACCION DE LOS ACTORES	**RESPUESTA DEL SISTEMA**
1. Selecciona la opción "Citar fuente"	2. Muestra formulario de citación con opciones de estilo (APA, MLA...)
	3. La IA investiga el documento
4. Presiona la opción "Insertar cita"	5. Inserta la cita en el texto y añade la fuente
Flujos alternos	
3. El usuario puede reutilizar una fuente previamente usada desde su historial de citas.	

Caso de Uso	Exportar documento
Referencia	CU-004
Creado por	Sebastian Medina Garcia
Fecha de Creación	30/5/2025
Información General	
Requisito(s) Asociado(s)	RF004
Actor	Usuario editor
Descripción	Este caso de uso permite al usuario exportar el documento creado en distintos formatos como PDF, Word o texto plano, para almacenarlo localmente o compartirlo.
PRECONDICION	El usuario debe tener un documento abierto en el editor y haber iniciado sesión en el sistema.
POSTCONDICION	Se genera y descarga el archivo en el formato seleccionado, conservando el contenido y el formato del documento.
Flujo normal de los eventos	
ACCION DE LOS ACTORES	**RESPUESTA DEL SISTEMA**
1. Abre un documento en el editor de Sebbi AI	2. Muestra el contenido del documento
3. Hace clic en la opción "Exportar"	4. Despliega menú con opciones de formato (PDF, DOCX, TXT...)
5. Selecciona el formato deseado	
6. Presiona "Descargar"	7. Genera el archivo y lanza la descarga
Flujos alternos	
7. Si el usuario cancela la descarga, el sistema regresa al editor sin cambios.	

Caso de Uso	Chatear con IA
Referencia	CU-005
Creado por	Sebastian Medina Garcia
Fecha de Creación	30/5/2025
Información General	
Requisito(s) Asociado(s)	RF005
Actor	Usuario escritor
Descripción	Este caso de uso permite al usuario interactuar con un asistente de inteligencia artificial para resolver dudas, solicitar mejoras en el texto, generar ideas o recibir sugerencias personalizadas.
PRECONDICION	El usuario debe haber iniciado sesión y tener abierto un documento o acceder desde el menú de herramientas.
	Deben existir proyectos registrados
POSTCONDICION	El sistema muestra la respuesta generada por la IA, y esta puede ser insertada directamente en el documento si el usuario así lo decide.
Flujo normal de los eventos	
ACCION DE LOS ACTORES	**RESPUESTA DEL SISTEMA**
1. Accede a la opción "Abrir Chat"	2. Abre ventana de chat integrada
3. Escribe una pregunta o solicitud	4. Recibe y analiza la consulta con el modelo de lenguaje
	5. Genera una respuesta textual basada en la entrada del usuario
	6. Muestra la respuesta en pantalla con opciones para copiar o insertar
	7. Consulta la información en base de datos
Flujos alternos	
4. Si la IA no entiende la solicitud, sugiere reformulaciones o muestra un mensaje de error amigable	

Pruebas

Las pruebas de software se realizan para asegurar que un sistema o aplicación funcione correctamente según los requisitos establecidos y esté libre de errores. Su objetivo es identificar fallos, verificar la calidad del código y garantizar que el software cumpla con los estándares de rendimiento y seguridad. Además, las pruebas permiten detectar posibles problemas antes de que el sistema sea lanzado al público o utilizado en producción, minimizando riesgos y mejorando la experiencia del usuario. Por lo tanto, en esta sección se mostrarán las principales pruebas aplicadas a Sebbi AI de una forma agrupada (estas fueron comprobadas en más de 100 pruebas).



Identificador	**PC001**	
Objetivo de la Prueba	Verificar que el sistema realiza movimientos de recursos desde el almacenista hacia el técnico.	
Nombre del Caso	Validación de Componentes UI Críticos	
Precondiciones	El usuario debe estar logueado en la plataforma.	
	Los componentes deben estar correctamente importados	
	Deben existir datos de prueba para cada variante de componente.	
Paso	**Resultado Esperado**	**Resultado Obtenido**
Renderizar cada componente (Button, Input, Card, Checkbox, Dialog, Avatar) con sus variantes	El componente se muestra correctamente según sus props y variantes	Todos los componentes se muestran de la forma prevista
Simular eventos de usuario (click, input, check, open dialog)	El evento se ejecuta y se actualiza el estado visual	Todos los componentes tienen sus respectivos eventos funcionando.
Probar estados deshabilitados y de error	El componente refleja el estado esperado (disabled, error, etc.)	Los estados están en constante cambio y se reflejan de manera adecuada.

Identificador	ST001	
Objetivo de la Prueba	Validar la gestión de estado global y persistencia.	
Nombre del Caso	Gestión y Persistencia de Estado	
Precondiciones	El usuario debe estar logueado.	
	Deben existir datos en el store (uiStore/editorStore).	
Paso	Resultado Esperado	Resultado Obtenido
Cambiar el estado de sidebars, tema o layout	El estado global se actualiza y refleja en la UI	Toda la UI está en constante cambio gracias a los estados de la aplicación
Recargar la página	El estado persistente se mantiene (localStorage)	El storage conserva la información correspondiente

Identificador	PG001	
Objetivo de la Prueba	Verificar que la página principal muestra correctamente branding, navegación y contenido.	
Nombre del Caso	Validación de HomePage	
Precondiciones	El usuario debe tener acceso a la HomePage.	
	Deben existir elementos de branding y navegación configurados.	
Paso	Resultado Esperado	Resultado Obtenido
Renderizar la HomePage	Se muestran logo, navegación y contenido principal	Contenido generado correctamente.
Probar enlaces de navegación	Los enlaces funcionan y abren el destino correcto	Todos los enlaces corresctos.
Cambiar tamaño de pantalla	El contenido se adapta correctamente (responsive)	Contenido actualizado.

Identificador	**UT001**	
Objetivo de la Prueba	Validar utilidades críticas de la aplicación.	
Nombre del Caso	Cargar recursos del ProyectoValidación de Funciones Utilitarias	
Precondiciones	Las funciones deben estar importadas y listas para test.	
Paso	**Resultado Esperado**	**Resultado Obtenido**
Ejecutar formatDate con una fecha válida	Retorna la fecha formateada correctamente	Fecha generada.
Ejecutar validateEmail con emails válidos/erróneos	Valida correctamente el formato	Validación completada
Ejecutar generateId varias veces	Genera IDs únicos cada vez	Id generados.

Identificador	**TS001**	
Objetivo de la Prueba	Validar el funcionamiento del servicio de tours interactivos.	
Nombre del Caso	Gestión de Tours con driver.js	
Precondiciones	El usuario debe tener acceso a la funcionalidad de tours.	
	Deben existir pasos de tour restantes.	
Paso	**Resultado Esperado**	**Resultado Obtenido**
Iniciar un tour	El tour comienza desde el primer paso	Tour funcionando.
Navegar entre pasos	El tour avanza o retrocede correctamente	Tour funcionando.
Finalizar el tour	El estado de progreso se guarda correctamente	Tour funcionando.

Identificador	**HK001**	
Objetivo de la Prueba	Verificar la detección de dispositivos móviles y breakpoints.	
Nombre del Caso	Detección Responsive con useMobile	
Precondiciones	El hook debe estar implementado en un componente.	
Paso	**Resultado Esperado**	**Resultado Obtenido**
Cambiar el tamaño de la ventana a < 768px	El hook detecta modo móvil (true)	Responsive activo
Cambiar el tamaño a >= 768px	El hook detecta modo desktop (false)	Responsive activo
Renderizar en SSR	El hook no genera errores y retorna valor por defecto	Responsive activo

Identificador	**DG001**	
Objetivo de la Prueba	Validar el correcto funcionamiento de los diálogos modales.	
Nombre del Caso	Gestión de Diálogos y Accesibilidad	
Precondiciones	El componente Dialog debe estar implementado y renderizable.	
Paso	**Resultado Esperado**	**Resultado Obtenido**
Abrir el diálogo	El foco se mantiene dentro del modal	Dialogos modales validados.
Cerrar con icono x	El diálogo se cierra correctamente	Dialogos modales validados.

Identificador	**IN001**	
Objetivo de la Prueba	Validar el funcionamiento de los campos de entrada de datos.	
Nombre del Caso	Validación de Inputs	
Precondiciones	El componente Input debe estar implementado y renderizable.	
Paso	**Resultado Esperado**	**Resultado Obtenido**
Escribir texto en el input	El valor del input se actualiza correctamente y constantemente	Inputs validados

CONCLUSIÓN

Este proyecto de software ha permitido diseñar, desarrollar y documentar una solución que cumple con los objetivos y requisitos establecidos al inicio del proceso. A lo largo de su desarrollo, se ha seguido una metodología estructurada que ha garantizado la calidad, escalabilidad y usabilidad del sistema, realizando pruebas para asegurar su correcto funcionamiento. La documentación detallada proporcionada en este informe facilita la comprensión del diseño, arquitectura y funcionamiento del software, permitiendo que futuros desarrolladores puedan mantener, mejorar o expandir la solución de manera eficiente. Este proyecto no solo cubre las necesidades iniciales, sino que también establece una base sólida para el crecimiento y la evolución del sistema en el futuro.