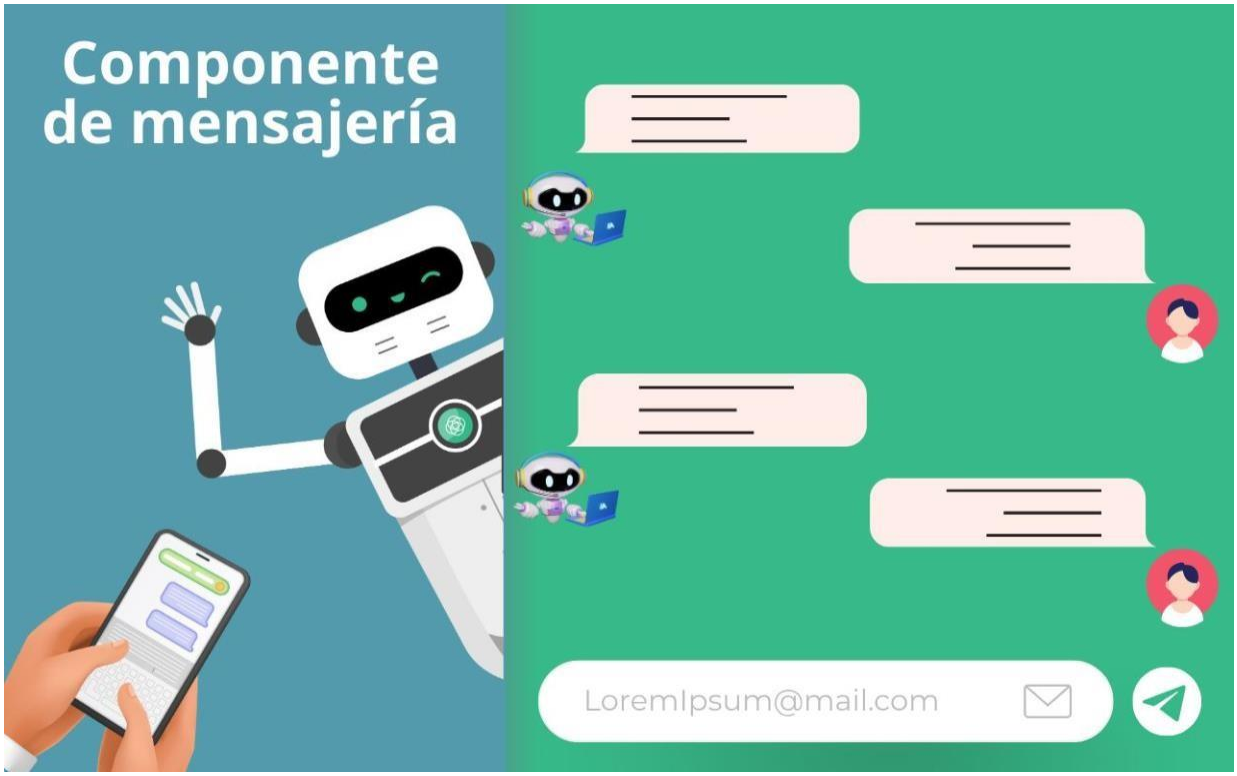
Documento de Propuesta de Diseño de Software I, II y II

Componente de mensajería

Autores: Jaider Martínez Paternina - María Camila Salgado Montiel - Andrés Felipe López León- Sergio Andrés Pineda Ruiz



Descripción del software

Este documento ofrece una exhaustiva exposición sobre la concepción, diseño y ejecución de un componente de mensajería de vanguardia, que integra un chat-bot impulsado por inteligencia artificial y un servicio de soporte brindado por expertos. La génesis de este componente responde a la necesidad imperante de optimizar la experiencia del usuario, a través de la entrega ágil y eficiente de respuestas pertinentes a sus interrogantes.

El enfoque innovador de este proyecto radica en la sinergia entre la avanzada tecnología de chat-bot basado en inteligencia artificial y la intervención humana de expertos en el servicio desoporte. Esta integración no solo permite la resolución instantánea de consultas comunes mediante el chat-bot, sino que también asegura una atención personalizada y de alta calidad cuando se requiere la intervención humana para cuestiones más complejas o específicas.

Etapa 1 Diseño de la Aplicación y Análisis de Requisitos

1. Introducción

Propósito del Documento

El propósito de este documento es recopilar de manera exhaustiva la información relacionada con la creación del componente de mensajería que se integrará en la plataforma plataforma de contenido digital. Esta documentación se enfoca en detallar las distintas fases necesarias para la obtención de un producto final, haciendo uso de la metodología de Design Thinking estructurada en 6 módulos, con el objetivo de clarificar y organizar las ideas de manera efectiva.

Alcance del Proyecto componente de mensajería

El proyecto tiene como objetivo principal desarrollar un componente de mensajería dentro de la plataforma plataforma de contenido digital, que permitirá a los usuarios interactuar de manera efectiva y eficiente con un chat-bot de respuestas rápidas, el soporte técnico (creador de la página) y otros usuarios. La implementación del componente se llevará a cabo en tres fases durante tres semestres académicos. En el primer semestre, se está enfocando en la documentación detallada de los componentes y sus funcionalidades.

Funcionalidades del componente:

1. **Chat Bot de Respuestas Rápidas:**
   Los usuarios podrán interactuar con un chat-bot que proporcionará respuestas rápidas a preguntas frecuentes y consultas comunes.
2. **Soporte Técnico:**
   Los usuarios podrán comunicarse con el soporte técnico, que es el creador de la página, para resolver problemas técnicos, recibir asistencia y hacer consultas específicas relacionadas con la plataforma plataforma de contenido digital.
3. **Comunicación entre Usuarios:**
   Los usuarios podrán enviar mensajes entre ellos, facilitando la interacción y la colaboración en la plataforma.

Visión Futura:

- El componente de mensajería proporcionará una base sólida para futuras mejoras y expansiones en plataforma de contenido digital. Se podrían considerar las siguientes mejoras a futuro:
- Integración de Multimedia: Permitir a los usuarios enviar y recibir imágenes, vídeos y otros archivos multimedia a través del componente de mensajería.
- Funcionalidades Avanzadas del Chat Bot: Mejorar la inteligencia artificial del chat bot para comprender consultas más complejas y proporcionar respuestas más detalladas.
- Interacción con Otras Plataformas: Integrar el componente de mensajería con otras plataformas y servicios externos para una experiencia de usuario más completa.

Este alcance del proyecto proporciona una visión clara de las funcionalidades a implementar y ver el enfoque por fases para su desarrollo exitoso en el transcurso de tres semestres.

4. **plataforma de contenido digital:** Nombre de la plataforma para la cual se está desarrollando el componente de mensajería. Se refiere al entorno digital donde los usuarios interactúan y colaboran.

5. **Chat-Bot:** Programa informático diseñado para simular una conversación con usuarios, especialmente a través de Internet. En este contexto, se refiere al sistema automatizadoque proporciona respuestas rápidas a las preguntas de los usuarios.

6. **Soporte Técnico:** Equipo o individuo responsable de brindar asistencia técnica a los usuarios de la plataforma plataforma de contenido digital. Se refiere al creador de la página o al equipo dedicado a resolver problemas y consultas técnicas.

7. **Componente de Mensajería:** Módulo o funcionalidad específica dentro de plataforma de contenido digital que permite a los usuarios enviar, recibir y gestionar mensajes de texto y otros tipos de contenido a través del chat-bot, el soporte técnico y otros usuarios.

2. Descripción General

Objetivos del Sistema

Chat-Bot

1. Interactividad y Engage:
   Generar respuestas interactivas y atractivas para los usuarios, manteniendo su interés y participación en la conversación.
2. Comprensión y Respuesta Precisa:
   Mejorar la capacidad de comprensión del chat-bot para proporcionar respuestas precisas y relevantes a las preguntas y consultas de los usuarios.
3. **Personalización:**
   Personalizar las respuestas del chat-bot según el historial y preferencias del usuario para brindar una experiencia más individualizada.

Chat Usuario

Experiencia del

Usuario:

Garantizar una interfaz de usuario intuitiva y fácil de usar que permita a los usuarioscomunicarse de manera efectiva y cómoda.

Seguridad y Privacidad:

Garantizar la seguridad y privacidad de las conversaciones de los usuarios, implementando medidas adecuadas de encriptación y protección de datos.

Chat de Soporte:

Eficiencia en la Respuesta:

Proporcionar respuestas rápidas y precisas a las consultas y problemas de los usuarios para mejorar la satisfacción del cliente.

Resolución de Problemas:

Garantizar que el chat de soporte tenga la capacidad de resolver problemas y consultas de los usuarios de manera efectiva y un tiempo hábil de 3 días.

Reducción de Cargas de Trabajo:

Automatizar tareas rutinarias para reducir la carga de trabajo del personal de soporte ypermitirles enfocarse en problemas más complejos y estratégicos.

Recopilación de Datos y Retroalimentación:

Recopilar datos y comentarios de los usuarios para identificar áreas de mejora en el servicio de soporte y optimizar la calidad de las respuestas.

Funcionalidad General

La funcionalidad general de un chat, en el contexto de las aplicaciones de mensajería y comunicación en línea, incluye una serie de características que permiten a los usuarios interactuar y comunicarse entre sí. Algunas de las funcionalidades típicas de un chat son las siguientes:

- Capacidad de enviar y recibir mensajes en tiempo real.
- Capacidad de reconocer y procesar las intenciones del usuario.
- Capacidad de responder automáticamente a las preguntas frecuentes.
- Capacidad para hacer preguntas al chat-bot
- Chat-bot redirige al usuario para enviar un mensaje a soporte
- Capacidad para hacer pregunta al chat soporte

Usuarios del Sistema

Los siguientes usuarios pueden interactuar con el chat dependiendo de las funcionalidades.

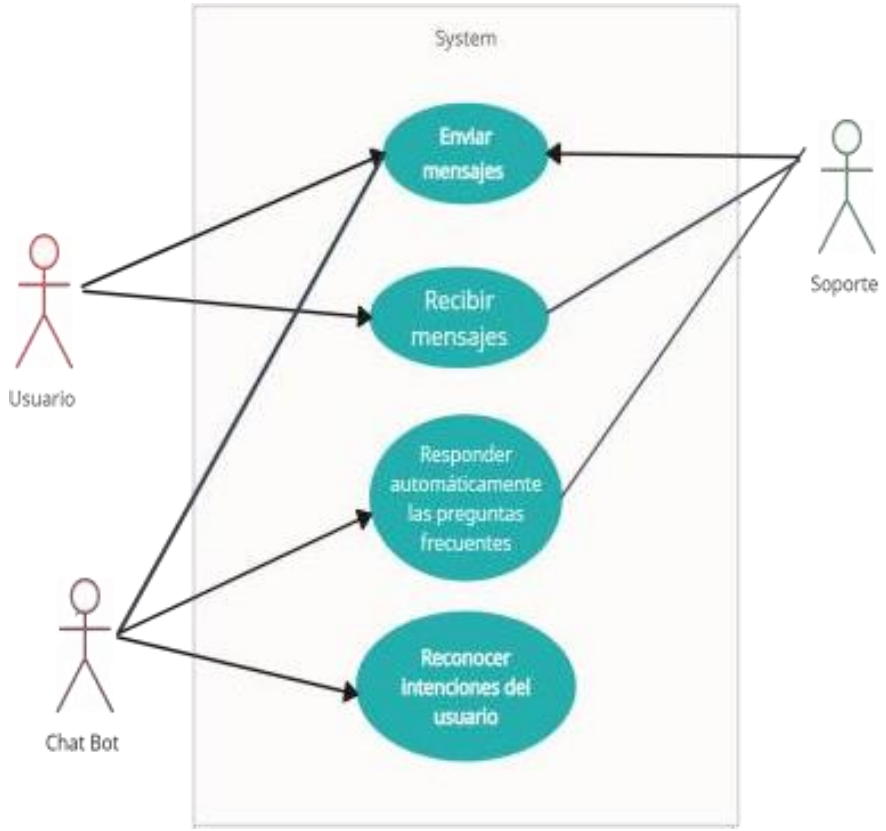
Funcionalidad	Sistema	Usuario	Chat-bot	Soporte
Enviar mensajes	✓	✓	✓	✓
Recibir mensajes	✓	✓	✓	✓
Responder automáticamente a las preguntas frecuentes	✓		✓	
Reconocer intención del usuario	✓		✓	

3. Requisitos Funcionales

Casos de Uso

1. Enviar mensajes
2. Recibir mensajes
3. Responder automáticamente las preguntas frecuentes
4. Reconocer intenciones del usuario

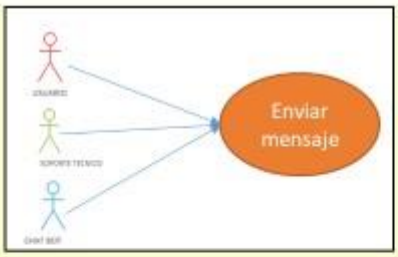
Diagrama de caso de uso



https://app.creately.com/d/start/dashboard

Diagramas de Flujo de Casos de Uso

CU-1



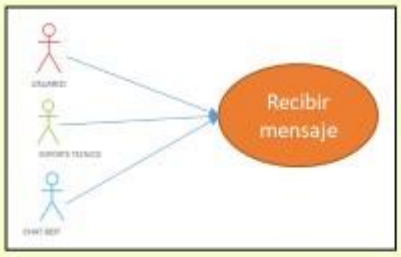
Urgencia: 4
Versión: 1
Esfuerzo: 5

Enviar mensaje
Flujo: enviar mensaje
Prueba: variable de comprobación mensaje enviado.

IC: Iniciar conversación
MOEM: Muestra la opción de "Escribir mensaje"
SOEM: Selecciona la opción "Escribir mensaje"
EM: Escribe su mensaje
SEM: enviar mensaje
ECB: se envia a chat bot
PM: procesa el mensaje
GR: genera respuesta
RRS: retorna respuesta al sistema
RRU: retorna respuesta al usuario
CC: cierra conversación

Enviar mensaje
Flujo: IC, MOEM, SOEM, EM, SEM, ECB, PM, GR, RRS, RRU, CC.

CU-2



Urgencia: 4
Versión: 1
Esfuerzo: 5

Recibir mensaje
Flujo: Recibir mensaje
Prueba: variable de comprobación mensaje recibido.

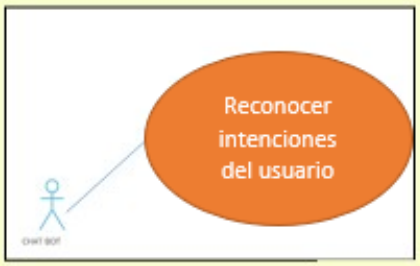
EM: Emite mensaje
PM: Procesa mensaje
EU: envía usuario
RM: Recibe mensajes

EM: Emite mensaje
PM: Procesa mensaje
ES: envía a soporte
RM: recibe los mensajes

EM: Emite mensaje
PM: Procesa mensaje
ECB: envía a chat bot
RM: recibe los mensajes

Recibir mensaje
Flujo: EM, PM, EU, RM.
Flujo alternativo 1: EM, PM, ECB, RM
Flujo alternativo 2: EM, PM, ES, RM

CU-3



Urgencia: 2

Versión:1

Esfuerzo: 3

Reconocer las intenciones del usuario

Flujo: reconocer intenciones del usuario

EM: emite los mensajes

PM: procesa los mensajes

EM: envía el mensaje al chat-Bot

PM: procesa los mensajes

AI: analiza e identifica la intención

RI: responde a la intención identificada

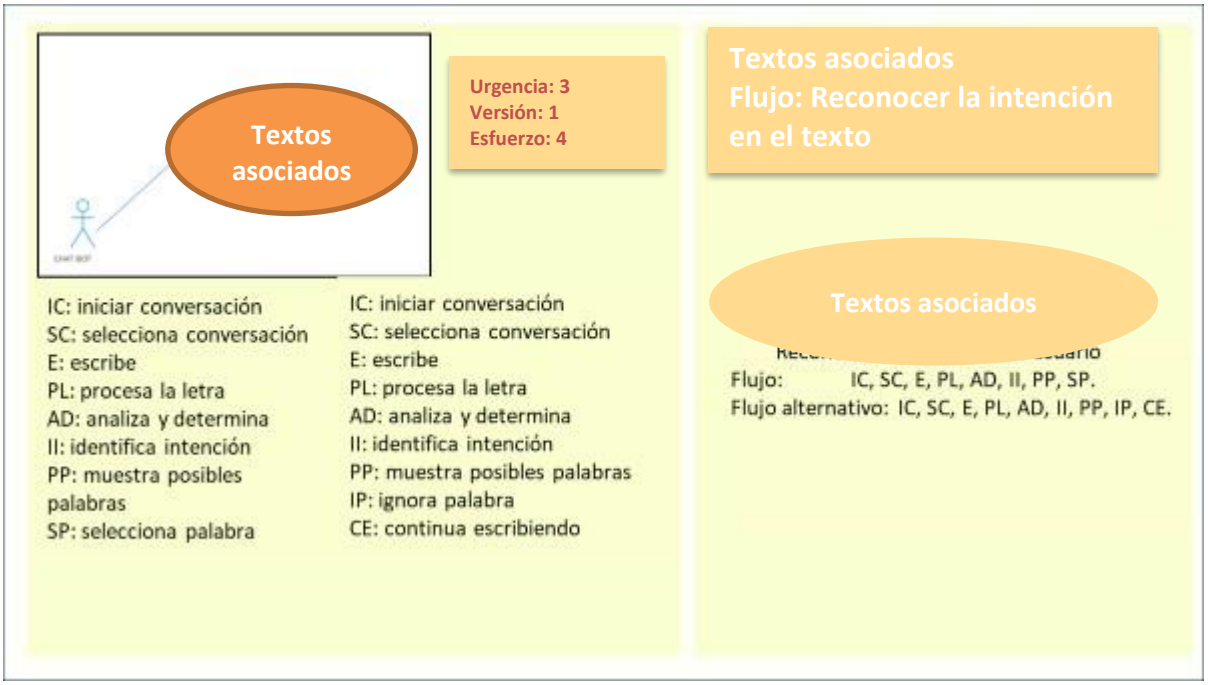
EM: envía el mensaje al usuario

RM: recibe el mensaje del chat-bot

Reconocer intenciones del usuario

Flujo: EM, PM, EM, PM, AI, RI, EM, RM.

CU-4


Textos asociados
Urgencia: 3
Versión: 1
Esfuerzo: 4
IC: iniciar conversación
SC: selecciona conversación
E: escribe
PL: procesa la letra
AD: analiza y determina
II: identifica intención
PP: muestra posibles palabras
SP: selecciona palabra
IC: iniciar conversación
SC: selecciona conversación
E: escribe
PL: procesa la letra
AD: analiza y determina
II: identifica intención
PP: muestra posibles palabras
IP: ignora palabra
CE: continua escribiendo
Textos asociados
Flujo: Reconocer la intención en el texto
Textos asociados
Flujo: IC, SC, E, PL, AD, II, PP, SP.
Flujo alternativo: IC, SC, E, PL, AD, II, PP, IP, CE.

CASO No. 1 Enviar mensajes

ID:	CU-1			
Nombre	Enviar mensajes			
Actores	Usuario, chat-bot, soporte			
Objetivo	Permitir enviar mensajes			
Urgencia	4			
Esfuerzo	5			
Pre-condiciones	Estar autenticado			
Flujo Normal	**USUARIO**	**SISTEMA**	**CHAT-BOT**	**SOPORTE**
	Iniciar conversación			
		Muestra la opción de “Escribir mensaje”		
	Selecciona la opción "Escribir mensaje"			
	Escribe su mensaje			
	Enviar mensaje			

		Se envía a chat-bot		
			Procesa el mensaje	
			Genera respuesta	
			Retorna respuesta al sistema	
		Retorna respuesta al usuario		
	Cierra conversación			
Flujo Alternativo1	Iniciar conversación			
		Muestra la opción de “Escribir mensaje”		
	Selecciona la opción "Escribir mensaje"			
	Escribe su mensaje			
	Enviar mensaje			
		Se envía a chat-bot		

			Procesa el mensaje	
			No puede responder al mensaje	
			Retorna mensaje al sistema	
		Se envía a soporte		
				Procesa el mensaje
				Genera respuesta
				Retorna respuesta al sistema
		Retorna respuesta al usuario		
	Cierra conversación			

CASO No. 2 Recibir mensajes

ID:	CU-2
Nombre	Recibir mensajes
Actores	Usuario, chat-bot, soporte

Objetivo	Permitir recibir mensajes			
Urgencia	4			
Esfuerzo	5			
Pre-condiciones	Estar autenticado			
Flujo Normal	**USUARIO**	**SISTEMA**	**CHAT-BOT**	**SOPORTE**
	Emite mensajes			
		Procesa los mensajes		
		Envía a usuario		
	Recibe mensaje			
Flujo Alternativo1	Emite mensajes			
		Procesa los mensajes		
		Envía a Chat-bot		
			Recibe los mensajes	
	Emite mensajes			

Flujo Alternativo2		Procesa los mensajes		
		Envía a Soporte		
				Recibe los mensajes

CASO No. 3 Reconocer intenciones del usuario

ID:	CU-3			
Nombre	Reconocer intenciones del usuario			
Actores	Chat-bot			
Objetivo	Permitir responder preguntas			
Urgencia	3			
Esfuerzo	4			
Pre-condiciones	Estar autenticado			
Flujo Normal	**USUARIO**	**SISTEMA**	**CHAT-BOT**	**SOPORTE**
	Emite un mensaje			
		Procesa los mensajes		
		Envía al chat bot		

			Procesa las palabras	
			Analiza e identifica la intención	
			Responde a la intención identificada	
		Envía mensaje al usuario		
	Recibe el mensaje del chat bot			

}

			automáticamente	
		Despliega respuesta		
	Visualiza larespuesta			

CASO No.4 Textos asociados

ID:	CU-4			
Nombre	Textos asociados			
Actores	Chat-bot			
Objetivo	Permitir reconocer las intenciones del usuario			
Urgencia	2			
Esfuerzo	3			
Pre-condiciones	Estar autenticado			
	USUARIO	**SISTEMA**	**CHAT-BOT**	**SOPORTE**
	Iniciar conversación			
	Selecciona conversación			
	Escribe			

Flujo Normal		Procesa las letras		
		Envía letras achat-bot		
			Analiza las letras para determinar su intención	
			Identifica intención	
		Muestra posibles palabras		
	Selecciona palabra			
Flujo alternativo	Iniciar conversación			
	Selecciona conversación			
	Escribe			
		Procesa las letras		
		Envía letras achat-bot		
			Analiza las letras para determinar su intención	
			Identifica intención	

		Muestra posibles palabras		
	Ignora palabras			
	Continúa escribiendo			

Prioridad de Requerimientos

A continuación, se presentan algunos

requisitosObligatorios:

- Capacidad de enviar y recibir mensajes de texto.
- Capacidad de enviar y recibir mensajes multimedia (imágenes, videos, etc.).
- Capacidad de enviar y recibir mensajes en tiempo real.
- Capacidad de enviar y recibir mensajes en grupos.
- Capacidad de reconocer y procesar las intenciones del usuario.
- Capacidad de responder automáticamente a las preguntas frecuentes.

Importantes:

- Capacidad de enviar y recibir mensajes encriptados.
- Capacidad de enviar y recibir mensajes programados.
- Capacidad de enviar y recibir mensajes con confirmación de lectura.
- Capacidad de personalizar la experiencia del usuario.
- Capacidad de integrarse con sistemas de gestión de tickets y soporte.

Deseables:

- Capacidad de enviar y recibir mensajes con traducción automática.
- Capacidad de enviar y recibir mensajes con filtros de spam.
- Capacidad de enviar y recibir mensajes con análisis de sentimiento.
- Capacidad de enviar y recibir mensajes con recomendaciones personalizadas.

- Capacidad de enviar y recibir mensajes con integración de chat-bot de terceros.

A partir del análisis de requerimientos, funcionalidades y el proceso de design thinking, se concreta la siguiente matriz de prioridad de requerimientos.

Para la interpretación se tiene en cuenta la siguiente escala con sus valores.

Eje de Urgencia:

- Obligatoria (5)
- Alta (4)
- Moderada (3)
- Menor (2)
- Baja (1)

Eje de Esfuerzo:

- Muy alto (5)
- Alto (4)
- Medio (3)
- Bajo (2)
- Muy bajo (1)

		Urgencia				
Impacto		1-Baja	2-Menor	3-Moderada	4-Alta	5-Obligatoria
	5-Muy alto	5	10	15	20	25
					CU-1 **CU-2**	
	4-Alto	4	8	12	16	20
				CU-3		
	3-Medio	3	6	9	12	15
			CU-4			
	2-Bajo	2	4	6	8	10
	1-Muy bajo	1	2	3	4	5

4. Requisitos No Funcionales

Requisitos de Desempeño

Rendimiento de chat-bot: Este debe proporcionar un rendimiento de preguntas rápidas en tiempo real la cual deben de reflejarse con el usuario para resolver múltiples problemas asignadas y calificadas como las que siempre preguntan simultáneamente por pasos para devolverse por caso de que la persona cometa un error.

Rendimiento de chat (usuario con contactos): en esta parte se cargarán rápido los mensajes de los usuarios para que no tengan inconvenientes al momento de chatear, en caso de fallar plataforma de contenido digital cuenta la opción de chat-bot y chat al soporte en casos mayores.

Rendimiento de chat de soporte: Este debe proporcionar en tiempo real la ayuda brindada en caso de que no encuentren posibles soluciones con el chat-bot, en esta opción se comunicara un cliente que es llamado usuario con una persona administradora de la página plataforma de contenido digital con conocimiento avanzado en el tema para brindarle la ayuda requerida siempre y cuando este en los parámetros de la página.

Requisitos de seguridad:

- **Acceso seguro:** Se debe implementar una autenticación segura para garantizar que solo usuarios autorizados tengan acceso a el chat. Esto puede incluir autenticación de 3 factores, inicio de sesión único, autenticación de seguridad con extremo a bot, y Autenticación de seguridad con extremo y soporte.
- **Alojamiento:** La interfaz del chat-bot es un aspecto a tener en cuenta. Si el bot se utilizaa través de una plataforma de chat como plataforma de contenido digital, la informacióndel cliente quedará expuesta y registrada por esta plataforma. Esto puede o no ser un problema de seguridad del chat-bot, pero es evidente que hay que tenerlo en cuenta.
- **Registro de actividades:** El sistema debe mantener registros de actividades, lo que incluye registros de cambios en la plataforma más que todo el chat, acceso de usuariosy eventos relevantes para la seguridad.
- **Políticas y procedimientos:** Por supuesto, es fundamental para todo lo anterior que se establezcan políticas y procedimientos pertinentes que rijan las normas de seguridad dela información. La seguridad de la información no se establece una sola vez, sino que esuna actividad continua. Estas políticas y procedimientos regirán no sólo la configuracióndel software pertinente, sino que también especificarán cuándo y cómo se llevarán a

cabo las sesiones periódicas de formación y las pruebas de seguridad. Proteger un chat-bot no difiere de proteger cualquier otro programa informático. Al principio del proceso debe evaluarse el grado de confidencialidad de los datos subyacentes, lo que determinará las medidas que debe adoptar la organización para garantizar la seguridadde los datos.

Requisitos de Seguridad

Acceso seguro: Se debe implementar una autenticación segura para garantizar que solo usuarios autorizados tengan acceso a el chat. Esto puede incluir autenticación de 3 factores, inicio de sesión único, autenticación de seguridad con extremo a bot, y Autenticación de seguridad con Extremo y soporte

Alojamiento: La interfaz del chat-bot es un aspecto a tener en cuenta. Si el bot se utiliza a través de una plataforma de chat como plataforma de contenido digital, la información del cliente quedará expuesta y registrada por esta plataforma. Esto puede o no ser un problema de seguridad del chat-bot, pero es evidente que hay que tenerlo en cuenta.

Registro de actividades: El sistema debe mantener registros de actividades, lo que incluye registros de cambios en la plataforma más que todo el chat, acceso de usuarios y eventos relevantes para la seguridad.

Políticas y procedimientos: Por supuesto, es fundamental para todo lo anterior que se establezcan políticas y procedimientos pertinentes que rijan las normas de seguridad de la información. La seguridad de la información no se establece una sola vez, sino que es una actividad continua. Estas políticas y procedimientos regirán no sólo la configuración del software pertinente, sino que también especificarán cuándo y cómo se llevarán a cabo las sesiones periódicas de formación y las pruebas de seguridad. Proteger un chatbot no difiere deproteger cualquier otro programa informático. Al principio del proceso debe evaluarse el grado de confidencialidad de los datos subyacentes, lo que determinará las medidas que debe adoptarla organización para garantizar la seguridad de los datos.

Requisitos de Usabilidad

Facilidad de Ingreso y Registro:

- Los usuarios deben poder ingresar al chat de manera sencilla y rápida, sin obstáculos innecesarios.
- Proporciona opciones de registro simples, como un correo electrónico.

Navegación Intuitiva:

- El chat debe tener una interfaz de usuario fácil de usar y una navegación clara.
- Los usuarios deben poder alternar de manera intuitiva entre conversaciones con otros, Chatbot y chat de soporte.

Interfaz Conversacional:

- El diseño del chat debe imitar una conversación real para que los usuarios se sientan cómodos.
- Utiliza avatares de quién está hablando para que las conversaciones sean fáciles de seguir.

Retroalimentación Clara: El sistema debe proporcionar retroalimentación clara, como indicadores de escritura, confirmaciones de envío y notificaciones de mensajes no leídos.

integración de Chat-bot: El chat- bot debe ser capaz de comprender las consultas de los usuarios y proporcionar respuestas útiles y precisas.

Privacidad y Seguridad: Garantiza la privacidad de las conversaciones y proporciona opciones de seguridad.

Atención al Cliente Efectiva: El chat de soporte debe ser accesible y brindar respuestas rápidas y útiles a las preguntas y problemas de los usuarios.

Tiempo de Respuesta Rápida: Tanto los usuarios como los chat-bot y el chat de soportedeben responder de manera eficiente para mantener las conversaciones fluidas.

Requisitos de Escalabilidad

Gestión de Conversaciones Múltiples: El sistema debe ser capaz de gestionar múltiples conversaciones simultáneas entre usuarios y Chat-bot, así como entre usuarios y agentes de soporte. Esto implica mantener un estado eficiente para cada conversación y asegurarse de quelas interacciones no se mezclen.

Distribución de Cargas Equitativas: Implementar un mecanismo de balanceo de carga que distribuya de manera equitativa las solicitudes de los usuarios entre los servidores o instanciasdel Chat-bot y los agentes de soporte.

Notificaciones: El sistema debe ser capaz de manejar notificaciones para los agentes de soporte,lo que significa que los agentes deben recibir alertas cuando un usuario solicita asistencia.

Integración de IA y Aprendizaje Automático: Si se utiliza IA o aprendizaje automático en el Chat-bot, asegúrate de que el sistema pueda escalar para manejar el entrenamiento y la inferencia de modelos a medida que la cantidad de datos y usuarios aumente.

5. Modelado E/R

A partir de la abstracción de los datos de las funcionalidades los datos preliminaresrecolectados son:

	Variables	Datos
CU-1 Enviar mensaje	**Mensaje Enviado**	Contenido: Hola, necesito ayuda para encontrar los contenidos.
	Verificación de enviado	True
	Verificación de recibido	True
	Hora de enviado	8:23 AM
	Fecha de enviado	15/03/2024

CU-2 Recibir mensaje		
	Mensaje recibido	Contenido: Hola, necesitoayuda para encontrar los contenidos.
	Hora de recibido	8:25 AM
	Fecha de recibido	15/03/2024

CU-3 Reconocer intenciones del usuario		
	Mensaje de usuario	¿Dónde puedo encontrar los contenidos?
	Preguntas Frecuentes	Lista de preguntas. ¿Cómo buscar contenidos?

CU-4 Textos asociados		
	texto asociado	Hola. Necesito ayuda para resolver un problema podrias contactarme con soporte
	función	darSoporteTecnico(){}
	Intención	Soporte Tecnico

Diagrama de Entidad-Relación

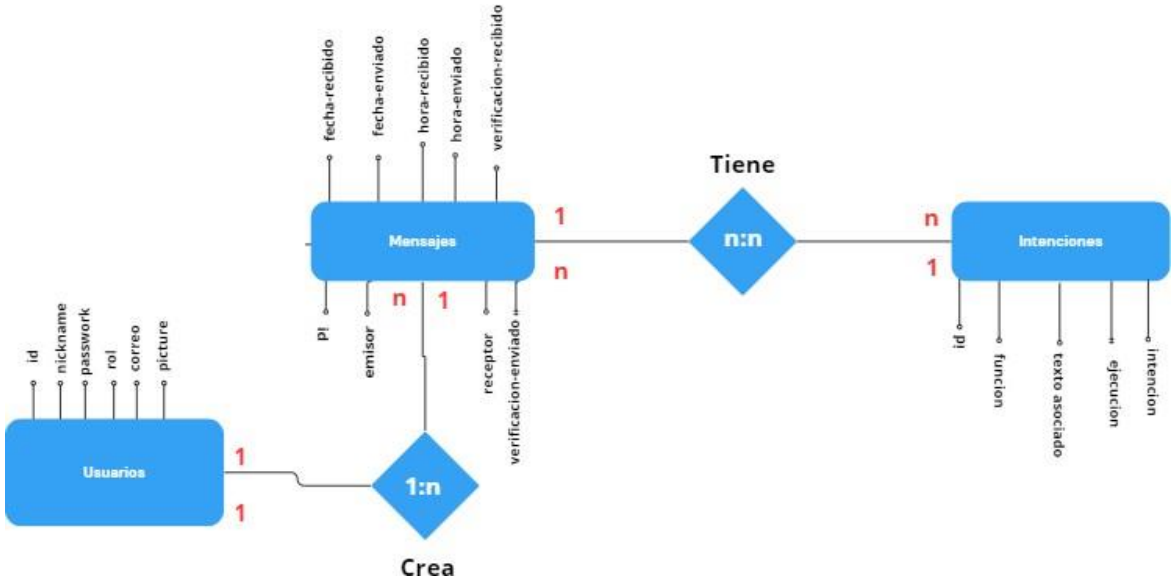
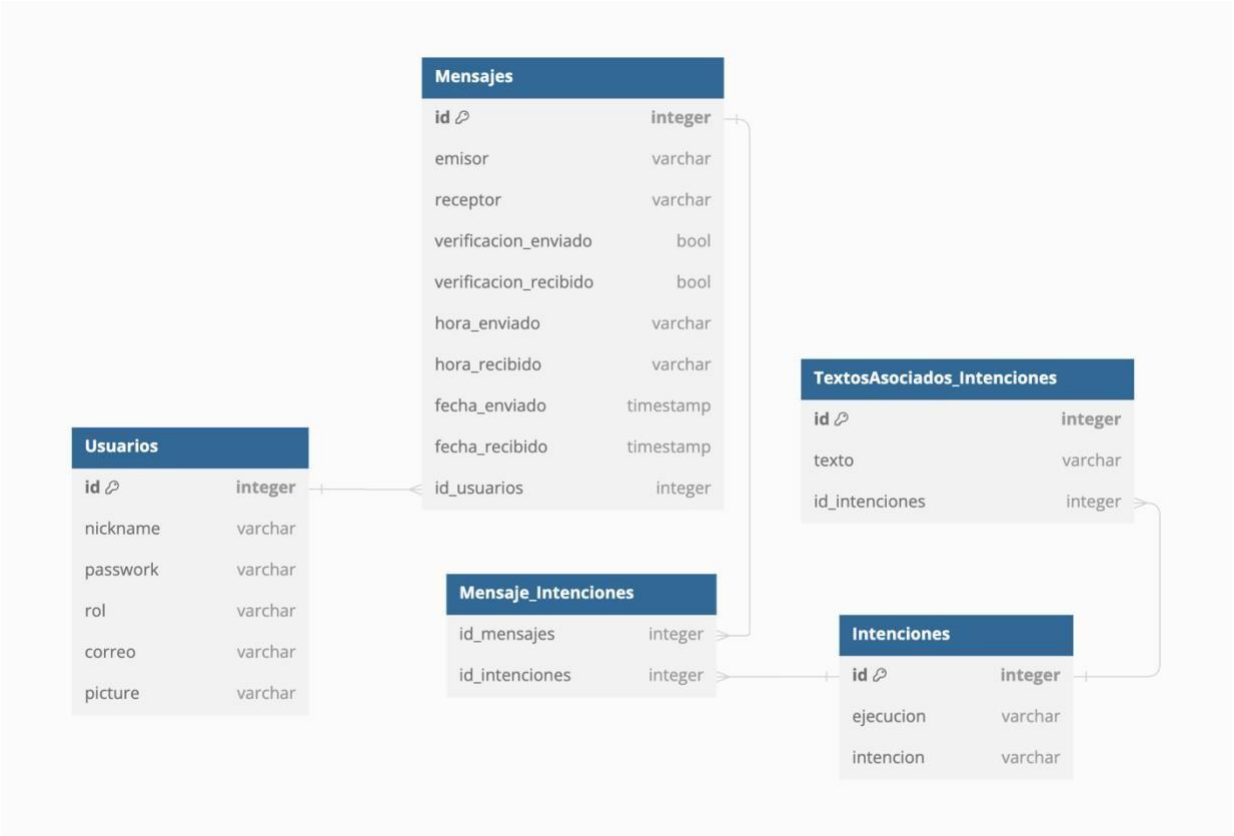


Diagrama Relacional



Script de modelo relacional

https://dbdiagram.io/

```
// Use DBML to define your database structure
// Docs: https://dbml.dbdiagram.io/docs

Table Usuarios {
  id integer [primary key]
  nickname varchar
  passwork varchar
  rol varchar
  correo varchar
  picture varchar
}

Table Mensajes {
  id integer [primary key]
  emisor varchar receptor
  varchar
  verificacion_enviado bool
  verificacion_recibido bool
  hora_enviado varchar
  hora_recibido varchar
  fecha_enviado timestamp
  fecha_recibido timestamp
  id_usuarios integer [ref: >Usuarios.id]
}

Table Intenciones {
  id integer [primary key]
  ejecucion varchar
  intencion varchar
}

Table Mensaje_Intenciones {
  id_mensajes integer [ref: > Mensajes.id]
```

```
  id_intenciones integer [ref: > Intenciones.id]
}

Table TextosAsociados_Intenciones
  {id integer [primary key]
  texto varchar
  id_intenciones integer [ref: > Intenciones.id]
}
```

Descripción de Entidades y Relaciones

Entidades:

1. **User (Usuario):**
   Almacena información sobre los usuarios que pueden acceder al módulo de mensajes
   Atributos: ID (identificador único), nickname(), passwork(),rol(usuario, administrador), correo(dirección de correo electrónico para comunicación), picture(imagen de perfil del usuario)
   Relaciones: Cada usuario puede estar asociado con varios mensajes
2. **Messages (Mensajes):**
   Almacena información de los mensajes de los usuarios en el módulo.
   Atributos: ID (identificador único), emisor (ente que envía un mensaje), receptor (ente que recibe un mensaje), verificación_enviado(marca de enviado), verificacio_recibido(marca de recibido), hora_enviado(números), hora recibido(números), fecha_recibido(letras y números), fecha_enviado(letras y números), id_usuarios()
   Relaciones: Un mensaje puede ser escrito por un solo usuario, pero puede tener muchas intenciones.
3. **Intentions (intenciones):**
   Almacena contenido en el modulo de lo que quiere un usuario a través de su mensaje.
   Atributos: ID (identificador único), funcion(texto variado), texto_asociado(texto variado que incluye la intención), ejecicion(), intención()
   Relaciones: Las intenciones se asocian con un usuario a través de la relación con los mensajes.
4. **Associated texts (Textos asociados):**
   Se asocia a la intención del usuario relacionado con los mensajes.
   Atributos: ID (identificador único), texto, id_intenciones
   Relaciones: Cada mensaje puede tener varias intenciones y una intención puede tener varios mensajes

Relaciones:

- "User" se relaciona con "Messages" para indicar que un usuario puede tener varios mensajes.
- "Messages" se relaciona con "Intentions" para registrar que los usuarios que pueden tener intenciones en los mensajes.
  - "intenciones" se relaciona con "Textos asociados" para registrar que las intenciones se derivan de los textos.

Reglas de Integridad Referencial

1. **Integridad Referencial entre "Usert" y "menssages"**: Cada registro en el módulo "Messages" debe estar asociado con un usuario existente en la tabla "Users" a través de la clave foránea "user_id". Esto asegura que cada mensaje enviado o recibido esté asociado a un usuario válido en el sistema.
2. **Integridad Referencial entre "Messages" y "Conversations":** Cada registro en el módulo "Messages" debe estar asociado con una conversación existente en "Conversations" a través de la clave foránea "conversation_id".

3. **Integridad Referencial entre "Intents" y "Messages":** Cada registro en la tabla "Intents" debe estar asociado con un mensaje existente en la tabla "Messages" a través de la clave foránea "message_id". Descripción: Esto asegura que cada intención reconocida esté vinculada a un mensaje específico que la generó.

4. Integridad Referencial entre "Intents" y "Associated_Texts": Cada registro en la tabla "Associated_Texts" debe estar asociado con una intención existente en la tabla "Intents". Esto asegura que cada texto asociado para la identificación de intenciones esté vinculado a una intención válida en el sistema.

Colecciones (NoSLQ)

Mensajes
{
Id: String,
Emisor: String,
Receptor: String,
verificacion_enviado: boolean,
Verificacion_recibido: boolean,
Hora_enviado: Date,
Hora_recibido: Date,
Fecha_enviado: Date,
Fecha_recibido: Date,
id_usuario: Object.Id,
Intenciones:[Object.Id]
}

Intenciones: {
Id: Object: Id,
Intencion: String,
Mensajes: [Object.Id]
}

TextoAsociado_Intenciones:{
Id:Object.Id,
Texto: String,
Id.Intencion:Object.Id
}

6. Anexos

Diagramas

Adicionales

Referencias

Etapa 2: Persistencia de Datos con Backend

7. Introducción

La persistencia de la base de datos se refiere a los datos o metadatos guardados durante la ejecución de una integración, En esta etapa, detallaremos cómo se gestionan, almacenan y recuperan los datos a través del backend, garantizando la integridad y disponibilidad de la información en el sistema. Utilizamos MongoDB Atlas como base de datos y Postman para pruebas y conexiones.

Propósito de la Etapa

Esta etapa tiene como objetivo crear un sistema sólido y eficiente para la persistencia de datos que apoye las funciones del módulo de mensajes. Esto incluye procesar las intenciones del usuario, enviar y recibir mensajes de texto y garantizar que todos los datos estén correctamente almacenados y recuperados de manera confiable.

Alcance de la Etapa

 Diseño de la Base de Datos:

- Definición de las colecciones necesarias (Users, Messages, Intents, Associated_Texts).
- Establecimiento de las relaciones entre las colecciones usando claves foráneas y referencias.

 CRUD Operaciones:

- Implementación de operaciones de creación, lectura, actualización y eliminación (CRUD) para cada colección.

 Conexión y Configuración del Backend:

- Configuración de la conexión con MongoDB Atlas.
- Desarrollo de endpoints en el backend para gestionar las operaciones CRUD.

 Integración y Pruebas:

- Uso de Postman para probar y validar los endpoints del backend.

Definiciones y Acrónimos

Definiciones

- **Backend:** Parte del sistema que se encarga de la lógica del servidor, gestión de bases de datos, y procesamiento de solicitudes del cliente. Es invisible para el usuario final.
- **CRUD:** Acrónimo de Create, Read, Update, Delete. Se refiere a las operaciones básicas que se pueden realizar en una base de datos.
- **Intención (Intent):** Representa la intención del usuario detrás de una entrada de texto. Por ejemplo, una intención podría ser "saludo" o "consulta".
- **Mensaje (Message):** Unidad de comunicación que contiene texto, multimedia o ambos, enviado entre usuarios o desde el sistema al usuario.
- **Texto asociado (Associated Text):** Fragmentos de texto que se utilizan para identificar intenciones específicas en los mensajes del usuario.

 Usuario (User): Persona que interactúa con el sistema, enviando y recibiendo mensajes.

- **MongoDB Atlas:** Servicio de base de datos en la nube para MongoDB, que ofrece una plataforma escalable y segura para almacenar y gestionar datos.

Acrónimos

- **API:** Application Programming Interface (Interfaz de Programación de Aplicaciones)
- **CRUD:** Create, Read, Update, Delete
- **DB:** Database (Base de Datos)
- **ID:** Identifier (Identificador)
- **JSON:** JavaScript Object Notation (Notación de Objetos de JavaScript)
- **REST:** Representational State Transfer (Transferencia de Estado Representacional)
- **HTTPS:** Hypertext Transfer Protocol Secure (Protocolo Seguro de Transferencia de Hipertexto)

8. Diseño de la Arquitectura de Backend

La arquitectura propuesta para el backend del módulo de mensajes está diseñada para ser escalable, segura y eficiente. Utiliza una estructura basada en microservicios para gestionar las distintas funcionalidades del sistema, permitiendo una fácil implementación y mantenimiento.

CRUD Operaciones

Ejemplos de Endpoints CRUD:

1. **Users**

- **Crear Usuario:** POST /users
- **Buscar Usuario:** GET /users/:id
- **Actualizar Usuario:** PUT /users/:id
- **Eliminar Usuario:** DELETE /users/:id

2. **Messages**

- **Crear Mensajes:** POST /messages
- **Buscar Mensajes:** GET /messages/:id
- **Actualizar Mensajes:** PUT /messages/:id
- **Actualizar Mensajes:** PATCH /messages/:id
- **Eliminar Mensajes:** DELETE /messages/:id

3. **Intents**

- **Crear Intención:** POST /intents
- **Buscar Intención:** GET /intents/:id
- **Actualizar Intención:** PUT /intents/:id
- **Actualizar Intención:** PACH /intents/:id
- **Eliminar Intención:** DELETE /intents/:id

4. **Associated_Texts**

- **Crear Texto Asociado:** POST /associated_texts
- **Leer Texto Asociado:** GET /associated_texts/:id

- **Actualizar Texto Asociado:** PUT /associated_texts/:id
- **Eliminar Texto Asociado:** DELETE /associated_texts/:id

Descripción de la Arquitectura Propuesta

Componentes del Backend

Diagramas de Arquitectura

9. Elección de la Base de Datos

Evaluación de Opciones (SQL o NoSQL)

Para el proyecto de mensajería detallado en este documento, donde se mencionan características como el manejo de mensajes, intenciones y texto asociado, la flexibilidad y escalabilidad proporcionada por una base de datos **NoSQL** como MongoDB Atlas puede ser más adecuada. Esto se debe a la naturaleza dinámica y potencialmente voluminosa de los datos manejados.

Justificación de la Elección

La elección de MongoDB Atlas como la base de datos NoSQL para el proyecto de mensajería se justifica por su flexibilidad de esquemas, capacidad de escalabilidad horizontal, alto rendimiento en operaciones de lectura y escritura, y eficiencia en el manejo de datos desestructurados. Estas características son esenciales para satisfacer los requisitos dinámicos y de alto rendimiento del sistema de mensajería, asegurando que pueda crecer y adaptarse según sea necesario.

Diseño de Esquema de Base de Datos

Aquí se presenta un esquema propuesto para la aplicación de mensajería, considerando los requisitos funcionales y las mejores prácticas para MongoDB.

Entidades Principales

1. **Usuarios**
2. **Mensajes**
3. **Conversaciones**
4. **Intenciones**

Esquema

1. Usuarios
Cada usuario tendrá un documento que almacena información relevante sobre el usuario.

```
{
   "_id": ObjectId("60d5ec49d6e01e1d8c8b4567"),
   "nombre": "Juan Pérez",
   "email": "juan.perez@example.com",
   "fechaRegistro": ISODate("2023-06-21T00:00:00Z"),
   "estado": "activo",
   "conversaciones": [
      {
         "conversacionId": ObjectId("60d5ec49d6e01e1d8c8b1234"),
         "ultimoMensaje": ISODate("2024-06-21T12:00:00Z")
      }
   ]
}
```

2. Mensajes
Cada mensaje estará representado como un documento, permitiendo almacenar los metadatos y el contenido del mensaje.

```
{
   "_id": ObjectId("60d5ec49d6e01e1d8c8b7890"),
   "conversacionId": ObjectId("60d5ec49d6e01e1d8c8b1234"),
   "remitenteId": ObjectId("60d5ec49d6e01e1d8c8b4567"),
   "contenido": "Hola, ¿cómo estás?",
   "timestamp": ISODate("2024-06-21T12:00:00Z"),
   "tipo": "texto",
   "leido": false
}
```

3. Conversaciones
Cada conversación almacenará la información de los participantes y los mensajes.

```
{
   "_id": ObjectId("60d5ec49d6e01e1d8c8b1234"),
   "participantes": [
      ObjectId("60d5ec49d6e01e1d8c8b4567"),
      ObjectId("60d5ec49d6e01e1d8c8b8901")
   ],
   "mensajes": [
      {
         "mensajeId": ObjectId("60d5ec49d6e01e1d8c8b7890"),
         "timestamp": ISODate("2024-06-21T12:00:00Z")
      }
   ]
}
```

4. Intenciones
Las intenciones de los mensajes, como comandos o solicitudes específicas, también estarán almacenadas.

```
{
   "_id": ObjectId("60d5ec49d6e01e1d8c8b4568"),
   "mensajeId": ObjectId("60d5ec49d6e01e1d8c8b7890"),
   "intencion": "consulta_clima",
   "parametros": {
      "ciudad": "Madrid",
      "fecha": "2024-06-21"
   }
}
```

Relación entre las Entidades

- Usuarios y Conversaciones: Un usuario puede participar en múltiples conversaciones.
- Conversaciones y Mensajes: Cada conversación contiene múltiples mensajes.
- Mensajes y Intenciones: Cada mensaje puede tener una intención asociada, dependiendo del contenido del mensaje.

Conclusión

El diseño del esquema de base de datos en MongoDB Atlas permite aprovechar la flexibilidad y escalabilidad de NoSQL, adaptándose a los requisitos dinámicos de la aplicación de mensajería. Este esquema no solo facilita el manejo de grandes volúmenes de datos y su rápida consulta, sino que también asegura que la estructura de datos pueda evolucionar con el tiempo sin necesidad de cambios disruptivos en el modelo.

10. Implementación del Backend

Elección del Lenguaje de Programación

La elección del lenguaje de programación es un paso fundamental en el desarrollo de cualquier sistema de software. Esta decisión debe basarse en varios factores, como los requisitos del proyecto, la experiencia del equipo de desarrollo, las características del lenguaje y el ecosistema de herramientas disponibles. En este contexto, analizaremos las razones para seleccionar un lenguaje de programación adecuado para el sistema de mensajería descrito en este documento.

Requisitos del Proyecto

1. **Rendimiento**: El sistema de mensajería debe ser capaz de manejar un alto volumen de mensajes en tiempo real.
2. **Escalabilidad**: Debe ser fácil de escalar para soportar un número creciente de usuarios y mensajes.
3. **Integración con MongoDB**: Debe integrarse eficientemente con la base de datos NoSQL seleccionada (MongoDB Atlas).
4. **Desarrollo Rápido**: El lenguaje debe permitir un desarrollo rápido y ágil para iterar rápidamente sobre nuevas funcionalidades.
5. **Comunidad y Soporte**: Una comunidad activa y un buen soporte de herramientas y librerías son esenciales para resolver problemas y mejorar la productividad del equipo.

Candidatos Principales

1. **JavaScript (Node.js)**
2. **Python**
3. **Java**
4. **C#**

Creación de la Lógica de Negocio

Para el sistema de mensajería, la lógica de negocio se encargará de manejar las operaciones relacionadas con usuarios, mensajes, conversaciones e intenciones.

Arquitectura

La arquitectura de la lógica de negocio se compondrá de varios servicios y controladores que interactúan con la base de datos y otros componentes de la aplicación. Aquí se presenta un diseño general utilizando JavaScript (Node.js) y el framework Express.js, que es adecuado para la creación de aplicaciones web rápidas y escalables.

Componentes de la Lógica de Negocio

1. **Servicio de Usuarios**
2. **Servicio de Mensajes**
3. **Servicio de Conversaciones**
4. **Servicio de Intenciones**

Desarrollo de Endpoints y APIs

La implementación de los endpoints y APIs en Node.js con Express.js permite crear un sistema de mensajería robusto y escalable. Los servicios están modularizados para facilitar el mantenimiento y la escalabilidad. Este diseño asegura que el sistema puede manejar eficientemente las operaciones CRUD para usuarios, mensajes, conversaciones e intenciones, proporcionando una base sólida para el desarrollo de nuevas funcionalidades en el futuro.

Autenticación y Autorización

11. Conexión a la Base de Datos

Configuración de la Conexión

Desarrollo de Operaciones CRUD

Manejo de Transacciones

12. Pruebas del Backend

Diseño de Casos de Prueba

Ejecución de Pruebas Unitarias y de Integración

Manejo de Errores y Excepciones

Etapa 3: Consumo de Datos y Desarrollo Frontend

13. Introducción

Propósito de la Etapa

Alcance de la Etapa

Definiciones y Acrónimos

14. Creación de la Interfaz de Usuario (UI)

Diseño de la Interfaz de Usuario (UI) con HTML y CSS

Consideraciones de Usabilidad

Maquetación Responsiva

15. Programación Frontend con JavaScript (JS)

Desarrollo de la Lógica del Frontend

Manejo de Eventos y Comportamientos Dinámicos

Uso de Bibliotecas y Frameworks (si aplicable)

16. Consumo de Datos desde el Backend

Configuración de Conexiones al Backend

Obtención y Presentación de Datos

Actualización en Tiempo Real (si aplicable)

17. Interacción Usuario-Interfaz

Manejo de Formularios y Validación de Datos

Implementación de Funcionalidades Interactivas

Mejoras en la Experiencia del Usuario

18. Pruebas y Depuración del Frontend

Diseño de Casos de Prueba de Frontend

Pruebas de Usabilidad

Depuración de Errores y Optimización del Código

19. Implementación de la Lógica de Negocio en el Frontend

Migración de la Lógica de Negocio desde el Backend (si necesario)

Validación de Datos y Reglas de Negocio en el Frontend

20. Integración con el Backend

Verificación de la Comunicación Efectiva con el Backend

Pruebas de Integración Frontend-Backend

ANEXOS

Diagramas UML

- **Diagrama de Casos de Uso (Use Case Diagram):** Este diagrama muestra las interacciones entre los actores (usuarios) y el sistema. Puede ayudar a identificar las funcionalidades clave y los actores involucrados.
- **Diagrama de Secuencia (Sequence Diagram):** Estos diagramas muestran la interacción entre objetos y actores a lo largo del tiempo. Puedes utilizarlos para representar cómo los usuarios interactúan con la pizarra en un flujo de trabajo específico.
- **Diagrama de Clases (Class Diagram):** Puedes utilizar este diagrama para modelar las clases y estructuras de datos subyacentes en el sistema, como usuarios, pizarras, comentarios, revisiones, etc.

- **Diagrama de Estados (State Diagram):** Este diagrama puede ser útil para modelar el comportamiento de la pizarra en diferentes estados, como "edición", "visualización", "comentario", etc.
- **Diagrama de Despliegue (Deployment Diagram):** Puedes utilizar este diagrama para representar cómo se despliega la aplicación en servidores y cómo interactúa con otros componentes del sistema, como el CMS.
- **Diagrama de Componentes (Component Diagram):** Este diagrama puede ayudar a representar la estructura de componentes del software, como la interfaz de usuario, la lógica de negocio, las bibliotecas y los servicios utilizados.
- **Diagrama de Actividad (Activity Diagram):** Puedes usar este diagrama para modelar flujos de trabajo o procesos específicos, como el flujo de trabajo de creación y edición de contenido en la pizarra.
- **Diagrama de Comunicación (Communication Diagram):** Similar a los diagramas de secuencia, estos diagramas muestran interacciones entre objetos y actores, pero pueden ser más simples y enfocados en la comunicación.
- **Diagrama de Paquetes (Package Diagram):** Este diagrama puede ayudar a organizar y visualizar los paquetes y módulos del software, lo que es útil para el diseño modular.
- **Diagrama de Objetos (Object Diagram):** Puedes utilizar este diagrama para representar instancias de clases y cómo interactúan en un escenario específico.