Integrantes:
Dennis Adolfo Jimenez Perez
Carlos Antonio Salazar Trinidad
Alan Oswaldo Toledo del Toro
Mauricio Abelardo Cordero Perez

Fecha: 30/abril/2025

DISEÑO DE LA BASE DE DATOS

1. Arquitectura

La arquitectura de la base de datos está diseñada siguiendo un modelo relacional utilizando PostgreSQL con la extensión pgvector para almacenamiento de embeddings vectoriales. Esta arquitectura proporciona una base sólida para almacenar y gestionar todos los datos relacionados con el asistente virtual de WhatsApp, incluyendo:

- **Datos de usuarios y segmentación**: Información de clientes, agentes y administradores.
- **Conversaciones y mensajes**: Historial completo de interacciones.
- **Base de conocimiento**: Contenido estructurado con vectores para búsqueda semántica.
- **Métricas y análisis**: Rendimiento del sistema, modelos LLM y agentes.

Características principales:

- **Esquema normalizado**: Diseño que minimiza la redundancia de datos y mantiene la integridad referencial.
- **Vectorización integrada**: Uso de pgvector para almacenar embeddings y realizar búsquedas semánticas eficientes.
- **Escalabilidad horizontal**: Capacidad para particionar por usuarios o fechas según necesidades futuras.
- **Almacenamiento optimizado**: Tipos de datos adecuados para cada campo, con índices en columnas críticas.
- **Trazabilidad completa**: Registro de todas las interacciones, métricas y cambios para análisis y mejora continua.

La arquitectura está diseñada para soportar operaciones de alta concurrencia, permitiendo múltiples lecturas y escrituras simultáneas mientras se mantiene la consistencia de los datos, crucial para un sistema de atención al cliente en tiempo real.

2. Usabilidad

2.1 Interfaz de Usuario

La interfaz para interactuar con la base de datos se divide en dos categorías principales:

Interfaz de Administración (Backend)

- **Panel de Administración**: Interfaz web desarrollada con React/TypeScript/Vite/Shadcn que permite:
    - Gestión completa de usuarios, agentes y administradores
    - Visualización y análisis de conversaciones
    - Administración de la base de conocimientos
    - Monitoreo de métricas y rendimiento
    - Configuración de parámetros del sistema
- **Autenticación**: Implementada mediante la librería betterauth, garantizando acceso seguro a operaciones críticas.
- **Dashboard Analítico**: Visualizaciones interactivas de tendencias, patrones y KPIs del sistema.

Interfaz para Desarrolladores y Mantenimiento

- **API interna documentada**: Endpoints RESTful para operaciones CRUD sobre todas las entidades.
- **Herramientas de migración**: Scripts para actualizaciones de esquema y migraciones de datos.
- **Monitoreo de rendimiento**: Dashboards específicos para el rendimiento de la base de datos.
- **Administración directa**: Acceso a herramientas como pgAdmin para DBAs y desarrolladores autorizados.

2.2 Compatibilidad con de Escritorio

Aunque la base de datos en sí es independiente del dispositivo, las interfaces de acceso están diseñadas considerando la compatibilidad multiplataforma:

- **Optimización de consultas**: Ajustes automáticos en la cantidad de datos devueltos según el dispositivo para garantizar rendimiento.
- **Caché inteligente**: Implementación de estrategias de caché (Redis) para mejorar tiempos de respuesta en todo tipo de dispositivos.
- **API REST compatible**: La capa de API que interactúa con la base de datos sigue estándares que facilitan el consumo desde cualquier plataforma.

3. Escalabilidad a 5 Años

Proyección de Crecimiento

Anticipamos el siguiente crecimiento en los próximos 5 años:

- **Volumen de usuarios**: Aumento exponencial de usuarios activos.
- **Tamaño de la base de conocimiento**: Incremento gradual de contenido estructurado.
- **Complejidad de consultas**: Mayor sofisticación en análisis y búsquedas.
- **Requisitos de rendimiento**: Expectativas más exigentes de tiempos de respuesta.

Estrategias de Escalabilidad

Para enfrentar este crecimiento, se implementarán las siguientes estrategias:

1. Particionamiento (Sharding)

- **Particionamiento horizontal por fecha**: Segmentación de tablas masivas (Messages, ConversationMetrics) por rangos de fecha.
- **Particionamiento por usuario**: Distribución de datos de usuario en múltiples nodos según patrones de acceso.

2. Replicación y Distribución

- **Réplicas de lectura**: Implementación de nodos dedicados a operaciones de solo lectura.
- **Distribución geográfica**: Réplicas estratégicamente ubicadas para reducir latencia según ubicación de usuarios.

3. Optimización de Consultas y Almacenamiento

- **Índices adaptativos**: Ajuste periódico de índices según patrones de consulta.
- **Políticas de retención**: Archivado automático de datos históricos poco accedidos.
- **Compresión selectiva**: Implementación de compresión para datos históricos.

4. Arquitectura Evolutiva

- **Separación de servicios**: Migración progresiva hacia microservicios con bases de datos dedicadas.
- **CQRS (Command Query Responsibility Segregation)**: Separación de modelos de lectura y escritura.
- **Event Sourcing**: Adopción parcial para operaciones críticas que requieran auditabilidad absoluta.

5. Tecnologías Emergentes

- **Evaluación periódica**: Análisis semestral de nuevas tecnologías de base de datos.
- **Migraciones incrementales**: Adopción selectiva de mejoras tecnológicas sin disrupciones.

3.1 Timeline y Backlog Esperado

Fase 1: Implementación Base (Meses 1-3)

Hitos:

- Despliegue del esquema inicial en PostgreSQL
- Configuración de pgvector y creación de índices iniciales
- Implementación de autenticación con betterauth
- Sistema básico de backups y monitoreo

Backlog:

- Crear scripts de migración y rollback
- Implementar vistas para consultas frecuentes
- Configurar políticas de seguridad a nivel de base de datos
- Documentar esquema y procedimientos de mantenimiento

Fase 2: Optimización y Monitoreo (Meses 4-6)

Hitos:

- Implementación de métricas detalladas de rendimiento
- Optimización de consultas críticas
- Configuración de caché Redis para objetos frecuentes
- Automatización de backups con verificación

Backlog:

- Crear dashboard de monitoreo de rendimiento
- Implementar jobs de mantenimiento periódico
- Desarrollar pruebas de carga y estrés
- Optimizar índices basados en patrones reales de uso

Fase 3: Escalabilidad Inicial (Meses 7-12)

Hitos:

- Implementación de particionamiento por fecha
- Configuración de réplicas de lectura
- Estrategia de archivado para datos históricos
- Expansión de capacidad de almacenamiento vectorial

Backlog:

- Automatizar balanceo de carga entre réplicas
- Implementar estrategia de failover
- Crear procedimientos de escalado horizontal
- Optimizar consultas vectoriales complejas

Fase 4: Evolución Arquitectónica (Año 2)

Hitos:

- Migración parcial a servicios con bases de datos dedicadas
- Implementación de CQRS para módulos críticos
- Estrategia de distribución geográfica
- Adaptación a volúmenes elevados de transacciones

Backlog:

- Desarrollar servicios de sincronización entre bases
- Implementar estrategias avanzadas de caché
- Evaluar tecnologías complementarias (TimescaleDB, Clickhouse)

- Refinar estrategias de particionamiento

Fase 5: Consolidación y Avance (Años 3-5)

Hitos:

- Implementación completa de arquitectura escalable
- Adopción selectiva de tecnologías emergentes
- Automatización avanzada de operaciones de base de datos
- Optimización para analítica predictiva y en tiempo real

Backlog:

- Desarrollar capacidades avanzadas de ML sobre datos
- Implementar estrategias de alta disponibilidad global
- Automatizar completamente la escalabilidad
- Refinar arquitectura para minimizar costos operativos

4. Diccionario de la Base de Datos

Tabla: Users

NOMBRE DEL CAMPO	TIPO	LONGITUD	DESCRIPCIÓN
user_id	INT	-	Identificador único del usuario (clave primaria)
first_name	VARCHAR	100	Nombre del usuario
last_name	VARCHAR	100	Apellido del usuario
email	VARCHAR	150	Correo electrónico del usuario
phone_number	VARCHAR	20	Número telefónico de contacto
whatsapp_number	VARCHAR	20	Número de WhatsApp para comunicación directa
location	VARCHAR	200	Ubicación geográfica del usuario
created_at	TIMESTAMP	-	Fecha y hora de creación del registro
last_active	TIMESTAMP	-	Última vez que el usuario estuvo activo
status	ENUM	-	Estado del usuario: 'active', 'inactive', 'blocked'
notes	TEXT	-	Notas adicionales sobre el usuario

Tabla: UserSegments

NOMBRE DEL CAMPO	TIPO	LONGITUD	DESCRIPCIÓN
segment_id	INT	-	Identificador único del segmento (clave primaria)
user_id	INT	-	Referencia al usuario (clave foránea a Users.user_id)
segment_type	ENUM	-	Tipo de segmento: 'standard', 'premium', 'enterprise'
assigned_at	TIMESTAMP	-	Fecha y hora de asignación al segmento
expires_at	TIMESTAMP	-	Fecha de vencimiento del segmento (opcional)
assigned_by	VARCHAR	100	Persona o proceso que asignó el segmento

Tabla: Conversations

NOMBRE DEL CAMPO	TIPO	LONGITUD	DESCRIPCIÓN
conversation_id	INT	-	Identificador único de la conversación (clave primaria)
user_id	INT	-	Referencia al usuario (clave foránea a Users.user_id)
start_time	TIMESTAMP	-	Fecha y hora de inicio de la conversación
end_time	TIMESTAMP	-	Fecha y hora de finalización (opcional)
topic	VARCHAR	200	Tema principal de la conversación
status	ENUM	-	Estado: 'active', 'resolved', 'escalated', 'abandoned'
escalated_to	VARCHAR	100	Agente al que se escaló la conversación (opcional)
ai_handled	BOOLEAN	-	Indica si la conversación fue manejada por IA

Tabla: Messages

NOMBRE DEL CAMPO	TIPO	LONGITUD	DESCRIPCIÓN
message_id	INT	-	Identificador único del mensaje (clave primaria)
conversation_id	INT	-	Referencia a la conversación (clave foránea)
sender_type	ENUM	-	Tipo de remitente: 'user', 'ai', 'agent'
content	TEXT	-	Contenido del mensaje
sent_at	TIMESTAMP	-	Fecha y hora de envío
intent	VARCHAR	100	Intención detectada en el mensaje
intent_confidence	FLOAT	-	Nivel de confianza en la intención detectada (0-1)
entities	JSON	-	Entidades detectadas en el mensaje en formato JSON

Tabla: ConversationMetrics

NOMBRE DEL CAMPO	TIPO	LONGITUD	DESCRIPCIÓN
metric_id	INT	-	Identificador único de la métrica (clave primaria)
conversation_id	INT	-	Referencia a la conversación (clave foránea)
response_time_avg	FLOAT	-	Tiempo promedio de respuesta en segundos
accuracy_score	FLOAT	-	Puntuación de precisión de las respuestas (0-1)
total_messages	INT	-	Total de mensajes en la conversación
ai_messages	INT	-	Cantidad de mensajes generados por IA
user_messages	INT	-	Cantidad de mensajes del usuario
agent_messages	INT	-	Cantidad de mensajes del agente humano
was_escalated	BOOLEAN	-	Indica si la conversación fue escalada
sentiment_score	FLOAT	-	Puntuación de sentimiento general (-1 a 1)

Tabla: KnowledgeBase

NOMBRE DEL CAMPO	TIPO	LONGITUD	DESCRIPCIÓN
kb_item_id	INT	-	Identificador único del ítem (clave primaria)
title	VARCHAR	200	Título del ítem de conocimiento
content	TEXT	-	Contenido completo del ítem
content_vector	VECTOR	1536	Representación vectorial del contenido para búsqueda semántica
category	ENUM	-	Categoría: 'product', 'faq', 'policy', 'guide'
status	ENUM	-	Estado: 'active', 'draft', 'archived'
created_at	TIMESTAMP	-	Fecha y hora de creación
updated_at	TIMESTAMP	-	Fecha y hora de última actualización
created_by	VARCHAR	100	Persona que creó el ítem

Tabla: KnowledgeBaseUsage

NOMBRE DEL CAMPO	TIPO	LONGITUD	DESCRIPCIÓN
usage_id	INT	-	Identificador único del uso (clave primaria)
kb_item_id	INT	-	Referencia al ítem de conocimiento (clave foránea)
message_id	INT	-	Referencia al mensaje donde se usó (clave foránea)
relevance_score	FLOAT	-	Puntuación de relevancia del ítem para el mensaje (0-1)
used_at	TIMESTAMP	-	Fecha y hora de uso

Tabla: IntentAnalytics

NOMBRE DEL CAMPO	TIPO	LONGITUD	DESCRIPCIÓN
intent_id	INT	-	Identificador único de la intención (clave primaria)
intent_name	VARCHAR	100	Nombre de la intención detectada
occurrence_count	INT	-	Número de veces que se ha detectado
avg_confidence	FLOAT	-	Confianza promedio en la detección (0-1)
escalation_count	INT	-	Número de veces que generó escalación
last_detected	TIMESTAMP	-	Última vez que se detectó
sample_queries	JSON	-	Ejemplos de consultas en las que se detectó

Tabla: ContentGaps

NOMBRE DEL CAMPO	TIPO	LONGITUD	DESCRIPCIÓN
gap_id	INT	-	Identificador único de la brecha (clave primaria)
missing_information	VARCHAR	255	Descripción de la información faltante
frequency	ENUM	-	Frecuencia de detección: 'high', 'medium', 'low'
detection_count	INT	-	Número de veces que se ha detectado
first_detected	TIMESTAMP	-	Primera vez que se detectó
last_detected	TIMESTAMP	-	Última vez que se detectó
status	ENUM	-	Estado: 'open', 'in_progress', 'resolved'

Tabla: Agents

NOMBRE DEL CAMPO	TIPO	LONGITUD	DESCRIPCIÓN
agent_id	INT	-	Identificador único del agente (clave primaria)
first_name	VARCHAR	100	Nombre del agente
last_name	VARCHAR	100	Apellido del agente
email	VARCHAR	150	Correo electrónico del agente
phone_number	VARCHAR	20	Número telefónico de contacto
department	ENUM	-	Departamento: 'technical', 'sales', 'customer_support'
status	ENUM	-	Estado: 'online', 'offline', 'in_meeting'
current_load	FLOAT	-	Carga actual de trabajo (0-1)
last_active	TIMESTAMP	-	Última vez que estuvo activo

Tabla: AgentMetrics

NOMBRE DEL CAMPO	TIPO	LONGITUD	DESCRIPCIÓN
metric_id	INT	-	Identificador único de la métrica (clave primaria)
agent_id	INT	-	Referencia al agente (clave foránea)
metric_date	DATE	-	Fecha de la métrica
conversations_handled	INT	-	Número de conversaciones atendidas
avg_resolution_time	FLOAT	-	Tiempo promedio de resolución en minutos
customer_satisfaction	FLOAT	-	Puntuación de satisfacción del cliente (0-5)
escalations_received	INT	-	Número de escalaciones recibidas

Tabla: LLMModels

NOMBRE DEL CAMPO	TIPO	LONGITUD	DESCRIPCIÓN
model_id	INT	-	Identificador único del modelo (clave primaria)
model_name	VARCHAR	100	Nombre del modelo LLM
version	VARCHAR	50	Versión del modelo
deployed_at	TIMESTAMP	-	Fecha y hora de despliegue
is_active	BOOLEAN	-	Indica si el modelo está activo actualmente
parameters	JSON	-	Parámetros de configuración del modelo
system_prompt	TEXT	-	Prompt de sistema utilizado

Tabla: LLMPerformance

NOMBRE DEL CAMPO	TIPO	LONGITUD	DESCRIPCIÓN
performance_id	INT	-	Identificador único del registro (clave primaria)
model_id	INT	-	Referencia al modelo (clave foránea)
metric_date	DATE	-	Fecha de las métricas
avg_response_time	FLOAT	-	Tiempo promedio de respuesta en segundos
avg_token_usage	FLOAT	-	Uso promedio de tokens por respuesta
resolution_rate	FLOAT	-	Tasa de resolución sin escalación (0-1)
error_rate	FLOAT	-	Tasa de errores o respuestas problemáticas (0-1)
top_failing_intents	JSON	-	Lista de intenciones con mayor tasa de fallo

5. Diseño de la Base de Datos

A continuación se presenta el diseño de la base de datos utilizando la notación ERD (Entity Relationship Diagram), que muestra todas las tablas y sus relaciones:

https://dbdiagram.io/d/AIGen-67ef25bc4f7afba1844c353c

```
// Users table
Table Users {
  user_id int [pk]
  first_name varchar
  last_name varchar
  email varchar
  phone_number varchar
  whatsapp_number varchar
  location varchar
  created_at timestamp
  last_active timestamp
  status enum('active', 'inactive', 'blocked')
  notes text
}

// UserSegments table
Table UserSegments {
  segment_id int [pk]
  user_id int [ref: > Users.user_id]
  segment_type enum('standard', 'premium', 'enterprise')
  assigned_at timestamp
  expires_at timestamp
  assigned_by varchar
}

// Conversations table
Table Conversations {
  conversation_id int [pk]
  user_id int [ref: > Users.user_id]
  start_time timestamp
  end_time timestamp
  topic varchar
  status enum('active', 'resolved', 'escalated', 'abandoned')
  escalated_to varchar
  ai_handled boolean
```

```
}

// Messages table
Table Messages {
  message_id int [pk]
  conversation_id int [ref: > Conversations.conversation_id]
  sender_type enum('user', 'ai', 'agent')
  content text
  sent_at timestamp
  intent varchar
  intent_confidence float
  entities json
}


// ConversationMetrics table
Table ConversationMetrics {
  metric_id int [pk]
  conversation_id int [ref: > Conversations.conversation_id]
  response_time_avg float
  accuracy_score float
  total_messages int
  ai_messages int
  user_messages int
  agent_messages int
  was_escalated boolean
  sentiment_score float
}

// KnowledgeBase table
Table KnowledgeBase {
  kb_item_id int [pk]
  title varchar
  content text
  content_vector vector(1536)
  category enum('product', 'faq', 'policy', 'guide')
  status enum('active', 'draft', 'archived')
  created_at timestamp
  updated_at timestamp
  created_by varchar
}
```

```
// KnowledgeBaseUsage table
Table KnowledgeBaseUsage {
  usage_id int [pk]
  kb_item_id int [ref: > KnowledgeBase.kb_item_id]
  message_id int [ref: > Messages.message_id]
  relevance_score float
  used_at timestamp
}

// IntentAnalytics table
Table IntentAnalytics {
  intent_id int [pk]
  intent_name varchar
  occurrence_count int
  avg_confidence float
  escalation_count int
  last_detected timestamp
  sample_queries json
}



// ContentGaps table
Table ContentGaps {
  gap_id int [pk]
  missing_information varchar
  frequency enum('high', 'medium', 'low')
  detection_count int
  first_detected timestamp
  last_detected timestamp
  status enum('open', 'in_progress', 'resolved')
}

// Agents table
Table Agents {
  agent_id int [pk]
  first_name varchar
  last_name varchar
  email varchar
  phone_number varchar
  department enum('technical', 'sales', 'customer_support')
  status enum('online', 'offline', 'in_meeting')
  current_load float
```

```
  last_active timestamp
}

// AgentMetrics table
Table AgentMetrics {
  metric_id int [pk]
  agent_id int [ref: > Agents.agent_id]
  metric_date date
  conversations_handled int
  avg_resolution_time float
  customer_satisfaction float
  escalations_received int
}

// LLMModels table
Table LLMModels {
  model_id int [pk]
  model_name varchar
  version varchar
  deployed_at timestamp
  is_active boolean
  parameters json
  system_prompt text
}



// LLMPerformance table
Table LLMPerformance {
  performance_id int [pk]
  model_id int [ref: > LLMModels.model_id]
  metric_date date
  avg_response_time float
  avg_token_usage float
  resolution_rate float
  error_rate float
  top_failing_intents json
}

// Additional relationships
Ref: Conversations.escalated_to > Agents.agent_id
```

Este diseño incluye todas las tablas y relaciones necesarias para soportar el asistente virtual con IA generativa integrado con WhatsApp. La estructura está optimizada para consultas eficientes manteniendo la integridad referencial entre entidades relacionadas.

Especificaciones de Requerimientos del Software

Presentación de la Información (Antecedentes)

En los últimos años, el uso de herramientas digitales para mejorar la atención al cliente y los procesos de venta ha tomado gran relevancia, especialmente en plataformas de mensajería instantánea como WhatsApp. Esta plataforma se ha consolidado como uno de los canales de comunicación más utilizados por empresas para interactuar con sus clientes debido a su accesibilidad, inmediatez y alta tasa de respuesta.

Ante este contexto, surge la necesidad de desarrollar soluciones tecnológicas que permitan automatizar procesos de atención y ventas, sin sacrificar la personalización y calidad de la experiencia del usuario. La inteligencia artificial generativa (IA), particularmente los modelos de procesamiento de lenguaje natural, se presenta como una tecnología clave para atender esta demanda, permitiendo la creación de asistentes virtuales capaces de interactuar de forma coherente, personalizada y en tiempo real con los usuarios.

Requisitos no funcionales

ID	NOMBRE DEL REQUISITO	DESCRIPCIÓN	OBSERVACIÓN
RNF01	Seguridad	Los permisos de acceso al sistema podrán ser cambiados solamente por el administrador de acceso a datos. Los sistemas deben respaldarse cada 24 horas. Los respaldos deben ser almacenados en una localidad segura.	Se debe implementar control de acceso basado en roles (RBAC). La seguridad de la autenticación y gestión de credenciales del panel de administración dependerá de la correcta implementación de la librería betterauth. Cifrado de datos sensibles.
RNF02	Acceso	Solo usuarios autenticados mediante credenciales válidas podrán acceder al panel de administración y modificar configuraciones del asistente.	La autenticación de usuarios (administradores, agentes) para el panel se gestionará mediante la librería betterauth. Se recomienda el uso de autenticación de dos factores (2FA) si es soportado por la librería o integrado adicionalmente.
RNF03	Almacenaje	El sistema debe permitir el almacenamiento estructurado de conversaciones, usuarios, métricas y contenido de la base de conocimientos.	Los datos se almacenarán en PostgreSQL, incluyendo vectores (pgvector). Se usará almacenamiento de objetos para datos no estructurados y Redis para caché. Cumplir normativas de protección de datos.

ID	NOMBRE DEL REQUISITO	DESCRIPCIÓN	OBSERVACIÓN
RNF04	Configuración	El asistente debe permitir configurar prompts, parámetros de modelos LLM, gestión de la base de conocimiento y reglas de negocio desde un panel de administración intuitivo (Frontend React/Shadcn).	Las configuraciones se guardarán en la base de datos y se aplicarán dinámicamente por los servicios correspondientes (e.g., LLM Service, RAG Service).
RNF05	Actuación	El asistente debe ofrecer respuestas en menos de 5 segundos (objetivo deseado) tras recibir un mensaje del usuario.	El rendimiento dependerá de la latencia de los servicios de IA (LLM, RAG), la base de datos y la API de WhatsApp. Se implementará monitoreo constante (Logging Service, Metrics Collector).
RNF06	Interoperabilidad	El sistema debe integrarse con la API de WhatsApp Business. Debe poder conectarse con sistemas externos como CRM y ERP a través de una capa de integración.	Se debe asegurar compatibilidad con versiones actualizadas de las APIs y documentar los puntos de integración.

ID	NOMBRE DEL REQUISITO	DESCRIPCIÓN	OBSERVACIÓN
RNF07	Recuperación	En caso de falla de un componente, el sistema deberá ser resiliente. Se buscará una recuperación rápida y minimizar la pérdida de datos.	Implementar estrategias de alta disponibilidad (HA) y recuperación ante desastres (DR) en la infraestructura de nube. Backups regulares de la base de datos PostgreSQL y Object Storage.
RNF08	Accesibilidad	El panel de administración web (Frontend) debe seguir las pautas de accesibilidad WCAG para asegurar su uso por personas con discapacidades.	Se utilizarán componentes UI (Shadcn) que facilitan la implementación de buenas prácticas de accesibilidad.

Casos de Uso

Actores

Descripción de actores primarios:

- **Cliente / Usuario Final**: Interactúa con el asistente vía WhatsApp para consultas, pedidos, etc.
- **Administrador del Sistema**: Inicia sesión (vía betterauth) y gestiona la configuración del sistema, usuarios, base de conocimientos y monitorea el rendimiento a través del panel de administración web.
- **Agente Humano**: Inicia sesión (vía betterauth) e interviene en conversaciones escaladas por el sistema a través de una interfaz de agente.

Descripción de actores secundarios:

- **Jefe de Proyectos**: Supervisa el desarrollo, valida requisitos y gestiona recursos.

Ingeniería de Requisitos

Objetivo del Proyecto

Diseñar y desarrollar un asistente virtual con inteligencia artificial generativa que automatice la atención al cliente y el proceso de ventas a través de la plataforma WhatsApp, brindando respuestas personalizadas y en tiempo real. Este sistema busca mejorar la experiencia del usuario, reducir los tiempos de respuesta, y aumentar la conversión de ventas.

Justificación del Proyecto

En un entorno empresarial cada vez más digitalizado, ofrecer atención al cliente inmediata y personalizada es clave para diferenciarse. WhatsApp se ha convertido en una plataforma preferida por los usuarios para comunicarse con empresas. Un asistente basado en IA generativa permitirá automatizar estas interacciones de forma inteligente, mejorando la eficiencia operativa, reduciendo costos en atención al cliente, y aumentando la satisfacción y fidelización de los clientes. Este proyecto responde a la necesidad de integrar soluciones tecnológicas modernas en la comunicación empresarial.

Requisitos del Software (Detallados)

Categoría	Descripción
Interacción con Hardware	La solución se ejecutará en servidores en la nube (e.g., AWS, Google Cloud, Azure). El frontend requiere un navegador web moderno. Los usuarios finales interactúan vía WhatsApp en sus dispositivos móviles/web.
Interfaces Externas	API de WhatsApp Business, API de Servicios LLM (e.g., OpenAI GPT-4, LLaMA 2), API de Base de Datos Vectorial (si es externa), APIs de CRM/ERP, API de Autenticación (si betterauth expone una).
Velocidad Operativa	Procesamiento de mensajes y generación de respuestas de IA deben ocurrir en segundos (objetivo < 5s).
Tiempo de Respuesta	El tiempo de respuesta end-to-end (usuario envía -> usuario recibe) debe ser inferior a 5 segundos en condiciones normales. Dependiente de latencias externas (WhatsApp, LLM API, DBs).
Portabilidad	La arquitectura basada en contenedores (implícito en diagramas de despliegue en nube) facilita la portabilidad entre proveedores de nube. El frontend es una aplicación web (React/Vite).
Recuperación ante Fallos	Implementar backups automáticos de PostgreSQL y Object Storage. Estrategias de HA/DR en la nube para componentes críticos (API Gateway, Servidor de Aplicación, Bases de Datos).
Seguridad	Autenticación robusta para el panel de administración gestionada por la librería betterauth (se recomienda configurar 2FA si es posible), autorización basada en roles, cifrado de datos en reposo y tránsito, protección de APIs (API Gateway).

Categoría	Descripción
Calidad del Software	Coherencia y relevancia en respuestas de IA (RAG), alta disponibilidad (objetivo >99.9%), usabilidad del panel de administración (React/Shadcn UI).
Limitaciones	La calidad de la respuesta de IA depende de la calidad y cobertura de la Base de Conocimientos y la efectividad del modelo LLM. Requiere monitoreo y ajuste continuo.

Requisitos Funcionales

RF-CORE: Gestión de Mensajes y Conversación

ID	NOMBRE DEL REQUISITO	DESCRIPCIÓN
RF-C01	Recepción de Mensajes WhatsApp	El sistema (Message Handler) debe recibir mensajes entrantes desde la API de WhatsApp Business.
RF-C02	Procesamiento de Tipo de Mensaje	El sistema debe procesar mensajes de texto. Opcionalmente, manejar otros tipos (imágenes, documentos) si se define, o responder indicando limitación al texto.

ID	NOMBRE DEL REQUISITO	DESCRIPCIÓN
RF-C03	Envío de Mensajes WhatsApp	El sistema (Message Handler) debe poder enviar mensajes de respuesta al usuario a través de la API de WhatsApp Business.
RF-C04	Gestión de Contexto de Conversación	El sistema (Conversation Manager, Context Tracker) debe mantener el estado y el historial de la conversación activa para cada usuario.
RF-C05	Almacenamiento de Historial de Conversación	Cada mensaje (entrante/saliente) y metadatos relevantes (intención, timestamp, etc.) deben almacenarse en la base de datos (tabla Messages, Conversations).
RF-C06	Formateo de Respuestas	El sistema (Conversation Manager) debe formatear las respuestas generadas antes de enviarlas al usuario.

RF-USER: Gestión de Usuarios

ID	NOMBRE DEL REQUISITO	DESCRIPCIÓN
RF-U01	Registro/Identificación de Clientes	El sistema debe identificar al usuario por su número de WhatsApp y asociarlo a un registro en la tabla Users.
RF-U02	Almacenamiento de Datos de Cliente	Almacenar información básica del cliente (teléfono, nombre si se obtiene, estado, timestamps, segmento) en la tabla Users.
RF-U03	Segmentación de Usuarios	Permitir la asignación de segmentos a usuarios (e.g., Standard, Premium) y almacenar historial en UserSegments.
RF-U04	Gestión de Agentes	Almacenar información de agentes humanos (nombre, estado, departamento, carga) en la tabla Agents.
RF-U05	Gestión de Administradores	Permitir la gestión de usuarios administradores con diferentes roles en el panel de administración.

RF-AI: Procesamiento con Inteligencia Artificial

ID	NOMBRE DEL REQUISITO	DESCRIPCIÓN
RF-AI01	Clasificación de Intención	El sistema (Intent Classifier) debe analizar el mensaje del usuario para determinar su intención (e.g., consulta producto, soporte, pedido, saludo).
RF-AI02	Generación de Respuestas con LLM	Utilizar un servicio LLM (e.g., GPT-4, LLaMA 2) para generar respuestas coherentes y contextuales basadas en el prompt construido.
RF-AI03	Recuperación Aumentada (RAG)	Implementar un sistema RAG para buscar información relevante en la Base de Conocimiento Vectorial y usarla para enriquecer el contexto del LLM.
RF-AI04	Construcción de Prompts	El sistema (Prompt Engineer/Manager, Context Builder) debe construir prompts efectivos para el LLM, incluyendo historial, contexto recuperado y reglas de negocio.

ID	NOMBRE DEL REQUISITO	DESCRIPCIÓN
RF-AI05	Validación de Respuesta (Opcional)	Incluir un paso para validar la calidad, precisión y seguridad de la respuesta generada por el LLM antes de enviarla.
RF-AI06	Análisis de Sentimiento (Opcional)	Analizar el sentimiento del usuario para adaptar el tono de la respuesta o identificar problemas.

RF-KB: Gestión de Base de Conocimiento

ID	NOMBRE DEL REQUISITO	DESCRIPCIÓN
RF-KB01	Ingesta de Datos	Permitir la ingesta de datos desde diversas fuentes (productos, FAQs, políticas, documentos) para alimentar la base de conocimientos.
RF-KB02	Procesamiento de Datos (ETL)	Implementar un pipeline ETL para limpiar, transformar, segmentar y generar embeddings de los datos ingeridos.

ID	NOMBRE DEL REQUISITO	DESCRIPCIÓN
RF-KB03	Almacenamiento Vectorial	Almacenar los embeddings generados en una base de datos vectorial (e.g., PostgreSQL con pgvector) para búsqueda semántica.
RF-KB04	Almacenamiento de Contenido Original	Mantener el contenido original y sus metadatos (categoría, versión) en la tabla KnowledgeBase.
RF-KB05	Búsqueda Semántica	El sistema RAG debe realizar búsquedas vectoriales eficientes para encontrar documentos relevantes basados en la consulta del usuario.
RF-KB06	Actualización de Conocimiento	Permitir la actualización y versionado del contenido en la base de conocimientos a través del panel de administración.
RF-KB07	Identificación de Brechas	El sistema (Analytics Engine) debe analizar interacciones para identificar consultas sin respuesta o contenido faltante (Content Gaps).

RF-ESC: Escalación a Humanos

ID	NOMBRE DEL REQUISITO	DESCRIPCIÓN
RF-ESC01	Detección de Necesidad de Escalación	El sistema (Intent Classifier, Business Rules, Escalation Service) debe identificar cuándo una conversación requiere intervención humana.
RF-ESC02	Notificación a Agente Humano	Alertar a un agente humano disponible (a través de la Interfaz de Agente) proporcionando el contexto de la conversación.
RF-ESC03	Transferencia de Conversación	Permitir que un agente humano tome control de la conversación desde la interfaz de agente.

RF-ADM: Panel de Administración y Analíticas

ID	NOMBRE DEL REQUISITO	DESCRIPCIÓN
RF-ADM01	Acceso al Panel	Proveer una interfaz web (Frontend React/Shadcn) para administradores y agentes (con roles diferenciados).
RF-ADM10	Autenticación de Usuarios del Panel	El sistema debe autenticar a los administradores y agentes que intentan acceder al panel de administración utilizando la librería betterauth.
RF-ADM02	Visualización de Conversaciones	Permitir buscar, filtrar y visualizar el historial de conversaciones.
RF-ADM03	Configuración del Sistema	Permitir la configuración de parámetros del LLM, plantillas de prompt, reglas de negocio, etc., desde el panel.

ID	NOMBRE DEL REQUISITO	DESCRIPCIÓN
RF-ADM04	Gestión de Base de Conocimiento	Interfaz para añadir, editar, eliminar y organizar contenido (FAQs, productos, documentos) en la base de conocimientos.
RF-ADM05	Gestión de Usuarios	Interfaz para gestionar clientes, agentes y administradores (crear, editar, roles, segmentos, estado).
RF-ADM06	Visualización de Métricas	Mostrar métricas de rendimiento (conversaciones, tiempos de respuesta, resolución, satisfacción, uso de KB, rendimiento LLM) en un dashboard.
RF-ADM07	Registro y Monitoreo	Integrar con servicios de Logging y Métricas para monitorear la salud y rendimiento del sistema en tiempo real.

ID	NOMBRE DEL REQUISITO	DESCRIPCIÓN
RF-ADM08	Recolección de Feedback (Opcional)	Implementar mecanismos para recolectar feedback del usuario sobre la calidad de las respuestas o la conversación.
RF-ADM09	Motor de Analítica	Procesar datos de conversaciones y métricas para generar insights (tendencias, patrones, áreas de mejora).

Diagramas de Caso de Uso

Actores Identificados

- Cliente / Usuario Final
- Administrador del Sistema
- Agente Humano

Casos de Uso Principales

- **Cliente/Usuario Final**: Iniciar conversación, Consultar información (FAQ, Producto, Pedido), Realizar Pedido (Intención de compra), Recibir respuestas (de IA o Agente), Ser escalado a Agente Humano.
- **Administrador del Sistema**: Iniciar sesión (vía betterauth), Configurar sistema (Prompts, Modelos, Reglas), Gestionar Base de Conocimiento, Gestionar Usuarios (Clientes, Agentes, Admins), Visualizar Analíticas y Métricas.
- **Agente Humano**: Iniciar sesión (vía betterauth), Recibir notificación de escalación, Tomar control de conversaciones, Atender consultas complejas.

Caso de Uso: Consultar FAQ/Conocimiento General

Actor Principal: Cliente

Precondición: El cliente ha iniciado una conversación vía WhatsApp.

Flujo de eventos:

1. Cliente envía mensaje con una pregunta.
2. WhatsApp API reenvía el mensaje al Message Handler.
3. Message Handler lo pasa al Conversation Manager, que actualiza contexto.
4. Conversation Manager envía a Intent Classifier, que detecta intención "Consulta Conocimiento".
5. Intent Classifier (o Conversation Manager) invoca al RAG System con la consulta.
6. RAG System convierte consulta a embedding, busca en Base de Datos Vectorial (PostgreSQL/pgvector).
7. Base de Datos Vectorial devuelve documentos relevantes.
8. RAG System construye contexto con documentos y lo pasa al LLM Service.
9. LLM Service genera una respuesta basada en el contexto.
10. LLM Service devuelve respuesta al Conversation Manager.
11. Opcional: Respuesta pasa por Validador/Formateador.
12. Conversation Manager envía respuesta formateada al Message Handler.
13. Message Handler envía respuesta a WhatsApp API, que la entrega al usuario.
14. Conversation Manager registra la interacción en la base de datos (Messages, ConversationMetrics, KnowledgeBaseUsage).

Postcondición: El cliente recibe una respuesta generada por IA basada en la base de conocimientos.

Excepciones:

- 5A: Si RAG no encuentra documentos relevantes, se genera una respuesta indicando que no se encontró información o se escala.
- 9A: Si LLM Service falla o tarda demasiado, se envía mensaje de error o se escala.
- Fallo en conexión con BD Vectorial o LLM API.

Especificaciones del Producto

Tipo: Sistema Web Backend con servicios de IA + Frontend Web de Administración (React/TypeScript/Vite/Shadcn) + Integración con WhatsApp Business API.

Objetivo: Automatizar atención al cliente y procesos de venta usando IA generativa a través de WhatsApp.

Audiencia: Empresas, usuarios finales de WhatsApp, personal administrativo y de soporte.

1. Diagrama de Componentes

Componentes Principales:

- **Client Side**: Usuario final, Aplicación WhatsApp.
- **Integration Layer**: WhatsApp Business API, Message Handler Service.
- **Core AI System**: Conversation Manager, Context Tracker, Intent Classifier, LLM Service, Prompt Engineering Manager, RAG System.
- **Knowledge & Data Layer**: Vector Database (e.g., PostgreSQL+pgvector), Knowledge Base (contenido), Business Rules Engine, Backend Integration Service.
- **Enterprise Integrations**: CRM System, Product Catalog/ERP System.
- **Analytics & Monitoring**: Analytics Engine, Admin Dashboard (Frontend), Reporting Service, Model Training/Fine-tuning loop.
- **Human Support**: Escalation Service, Human Agent Interface, Support Agent.
- **Infrastructure**: API Gateway, Web Server (para Frontend/Admin), Redis Cache, PostgreSQL DB, Object Storage, Logging Service, Metrics Collector, Monitoring Dashboard.

2. Diagrama de Arquitectura del Software

La arquitectura se organiza en capas:

- **Capa de Usuario**: Interactúa a través de WhatsApp.
- **Capa de API Gateway**: Gestiona las solicitudes entrantes/salientes (WhatsApp API, Admin Frontend API).
- **Servidor de Aplicación**: Aloja la lógica de negocio principal, el servidor web para el panel de administración y se comunica con otros servicios. Utiliza caché Redis.

- **Servicios de IA**: Componentes desacoplados para LLM, RAG y la Base de Datos Vectorial.
- **Capa de Base de Datos**: Almacenamiento persistente con PostgreSQL (datos estructurados y vectores) y Object Storage (archivos).
- **Capa de Integración Empresarial**: Conecta con sistemas externos como CRM y ERP.
- **Monitoreo y Registro**: Servicios dedicados para recolectar logs y métricas, visualizados en un dashboard.
- **Panel de Administración**: Interfaz Frontend (React/Shadcn) para gestión y visualización.

El flujo de datos principal sigue la ruta: Usuario -> WhatsApp API -> API Gateway -> Servidor de Aplicación -> Servicios de IA/Bases de Datos -> Servidor de Aplicación -> API Gateway -> WhatsApp API -> Usuario. El panel de administración interactúa a través del API Gateway con el Servidor de Aplicación y las bases de datos.

3. Plataforma

- **Backend**: Arquitectura de microservicios (o servicios desacoplados) alojada en la nube. Lenguaje(s) de backend no especificado (podría ser Node.js, Python, Go, etc.). La autenticación del panel de administración se manejará con la librería betterauth (probablemente en el backend).
- **Frontend (Admin)**: Aplicación web construida con React, TypeScript, Vite, y componentes Shadcn UI.
- **Base de Datos**: PostgreSQL (con extensión pgvector).
- **Base de Datos Vectorial**: Puede ser PostgreSQL/pgvector o un servicio dedicado si se requiere.
- **Caché**: Redis.
- **Canal de Usuario**: WhatsApp Business API.
- **Infraestructura**: Proveedor de nube (AWS, GCP, Azure).

4. Requisitos de Rendimiento

- **Tiempo de Respuesta**: Objetivo < 5 segundos para respuestas de IA. Monitorizar latencias de componentes individuales y APIs externas.
- **Concurrencia**: El sistema debe manejar múltiples conversaciones simultáneas. La capacidad dependerá de los recursos asignados en la nube (CPU, RAM, conexiones a BD) y la escalabilidad de los servicios.
- **Rendimiento de Ingesta/Indexación**: El tiempo para procesar y vectorizar nuevo conocimiento dependerá del volumen de datos, la velocidad de la

API de embedding (si es externa) y la capacidad de escritura de la base de datos vectorial.

5. Lenguajes de Programación

- **Frontend**: TypeScript, JavaScript (JSX/TSX), CSS (Tailwind CSS).
- **Backend**: Node.js
- **Base de Datos**: SQL (PostgreSQL).

6. Herramientas de Desarrollo

- **Frontend**: Vite, Node.js/npm, ESLint, TypeScript, Tailwind CSS, React DevTools, Git.
- **Backend**: Entorno de desarrollo Node.js con npm, Librería de autenticación betterauth, Herramientas de prueba de API (e.g., Postman), Git.
- **Base de Datos**: Cliente PostgreSQL (e.g., pgAdmin, DBeaver).
- **Infraestructura**: Docker.
- **CI/CD**: GitHub Actions, GitLab CI, etc.

7. Hardware

- **Requerimientos de Usuario (finales)**: Dispositivo con WhatsApp y conexión a Internet.
- **Requerimientos del Sistema (Hosting)**: Recursos de servidor en la nube (CPU, RAM, Disco) adecuados para la carga esperada en el Web Server, Backend Services), PostgreSQL, Redis, Object Storage. Escalabilidad vertical y horizontal según necesidad. Conectividad de red fiable.

Expectativas de los Usuarios

- Rapidez, precisión, naturalidad, consistencia, claridad
- Capacidad de resolución de problemas
- Escalada transparente cuando sea necesario

Características más Importantes

- Disponibilidad 24/7
- Comprensión de Lenguaje Natural
- Acceso a Base de Conocimiento
- Registro simple

- Persistencia del contexto del cliente

Requerimientos del Sistema (Técnicos)

Software

- Sistema Operativo de Servidor (Linux preferido).
- Entorno de ejecución Backend Node.js.
- Servidor PostgreSQL (con pgvector).
- Servidor Redis.
- Librería/Dependencia betterauth para el backend.
- Credenciales/Acceso a APIs: WhatsApp Business, LLM Service, CRM/ERP (si aplica).
- Navegador web moderno (para panel de administración).

Red

Conexión estable y de baja latencia a Internet y entre servicios internos.

Interfaces Propuestas

https://preview--genai-whatsapp-assistant.lovable.app/

- **Interfaz de Usuario Final**: Aplicación WhatsApp.
- **Interfaz de Administración**: Panel Web construido con React/TypeScript/Shadcn UI, accesible vía navegador (Login gestionado por betterauth).
- **Interfaz de Agente Humano**: Parte del Panel Web o interfaz separada para gestionar conversaciones escaladas (Login gestionado por betterauth).
- **APIs Internas**: APIs REST/gRPC entre los microservicios (Message Handler, Conversation Manager, LLM Service, RAG Service, Servicio de Auth/betterauth, etc.).
- **APIs Externas**: WhatsApp Business API, LLM API, CRM/ERP API.

Base de Datos del Proyecto

Descripción General: La base de datos PostgreSQL está diseñada para soportar el asistente virtual, almacenando usuarios, conversaciones, agentes, métricas, la base de conocimientos (incluyendo vectores con pgvector) y datos analíticos.

Entidades Principales (Tablas):

- **Gestión de Usuarios**: Users, UserSegments, Agents, AgentMetrics.
- **Seguimiento de conversaciones**: Conversations, Messages, ConversationMetrics.
- **Gestión de Conocimiento**: KnowledgeBase (con vectores), KnowledgeBaseUsage.
- **Analíticas**: IntentAnalytics, ContentGaps, LLMModels, LLMPerformance.

Propósito: Gestionar eficientemente la interacción cliente-asistente, almacenar datos para análisis, mejorar el rendimiento de la IA y facilitar la gestión del sistema.