
Arquitectura Física y Técnica: Chatbot de Atención USB Cali

1. Visión General

Este chatbot, desarrollado en Node.js, interactúa con usuarios vía WhatsApp para proveer información sobre la Universidad de San Buenaventura Cali. Utiliza la librería whatsapp-web.js para la conexión con WhatsApp y técnicas de Procesamiento de Lenguaje Natural (PLN) con las librerías natural y string-similarity para interpretar las consultas y seleccionar respuestas de un corpus definido.

2. Arquitectura Física / Entorno de Ejecución

- **Máquina Local (PC/Laptop del Desarrollador):**
    - **SO:** Windows 11 (o el sistema operativo del desarrollador).
    - **Software Requerido:** Node.js (incluye npm).
    - **Conexión:** Acceso a Internet para comunicación con los servidores de WhatsApp.
- Usuario (WhatsApp App) <--> Servidores WhatsApp <--> [PC con Node.js: bot.js + whatsapp-web.js]

3. Arquitectura Técnica y Lógica del Bot

- **Componentes Clave:**
    - **bot.js (Script Principal):** Orquesta el flujo, maneja eventos de WhatsApp y la lógica de respuesta.
    - **whatsapp-web.js:** Interfaz con WhatsApp Web (vía Puppeteer/Chromium) para enviar/recibir mensajes y gestionar la sesión (autenticación QR, LocalAuth para persistencia).
    - **corpus.json:** Base de conocimiento con pares de preguntas y respuestas sobre la USB Cali.
    - **Módulos PLN:**
        - natural: Para tokenización (dividir texto en palabras) y stemming (reducir palabras a su raíz).

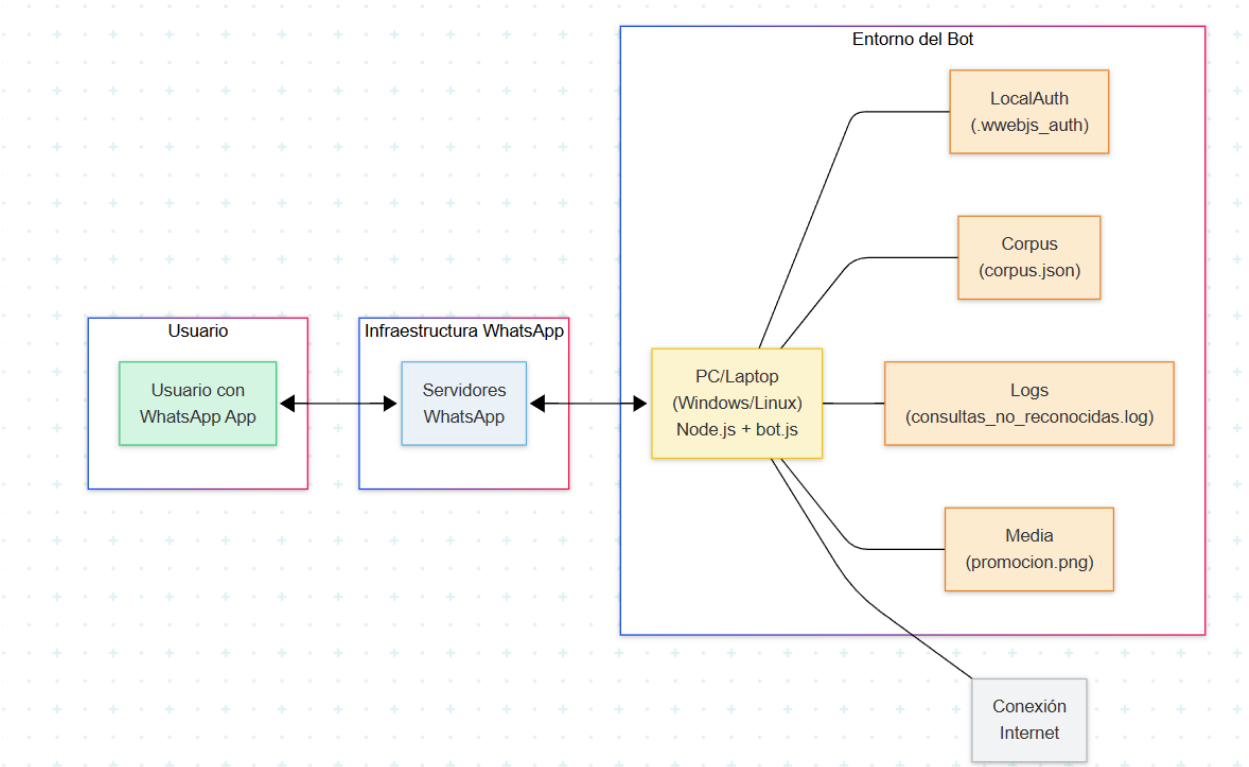
        - string-similarity: Para calcular la similitud entre la consulta del usuario y las preguntas del corpus.
    - fs (File System): Para registrar consultas no reconocidas en consultas_no_reconocidas.log.

- **Flujo Lógico Principal:**

1. **Inicio y Autenticación:**
    - bot.js inicia whatsapp-web.js.
    - Se muestra QR para autenticación o se carga sesión guardada (.wwebjs_auth/).
    - corpus.json se carga y sus preguntas se pre-procesan (tokenización, stemming).
2. **Mensaje de Bienvenida Automático (Evento ready):**
    - Al conectarse, el bot envía un mensaje con imagen promocional a una lista predefinida de usuarios (usuariosBienvenida).
3. **Procesamiento de Mensaje Entrante (Evento message):**
    - **Recepción:** whatsapp-web.js capta el mensaje del usuario.
    - **PLN del Input:** La consulta del usuario se normaliza (minúsculas, tokenización, stemming) en la función buscarRespuesta.
    - **Búsqueda en Corpus:** La consulta procesada se compara con las preguntas pre-procesadas del corpusProcesado usando string-similarity.
    - **Selección de Respuesta:**
        - Si la similitud más alta supera un UMBRAL_SIMILITUD, se elige la respuesta correspondiente del corpus.
        - Si no, se usa una respuesta genérica y la consulta original se guarda en consultas_no_reconocidas.log.
    - **Envío:** La respuesta seleccionada se envía al usuario vía whatsapp-web.js.

-

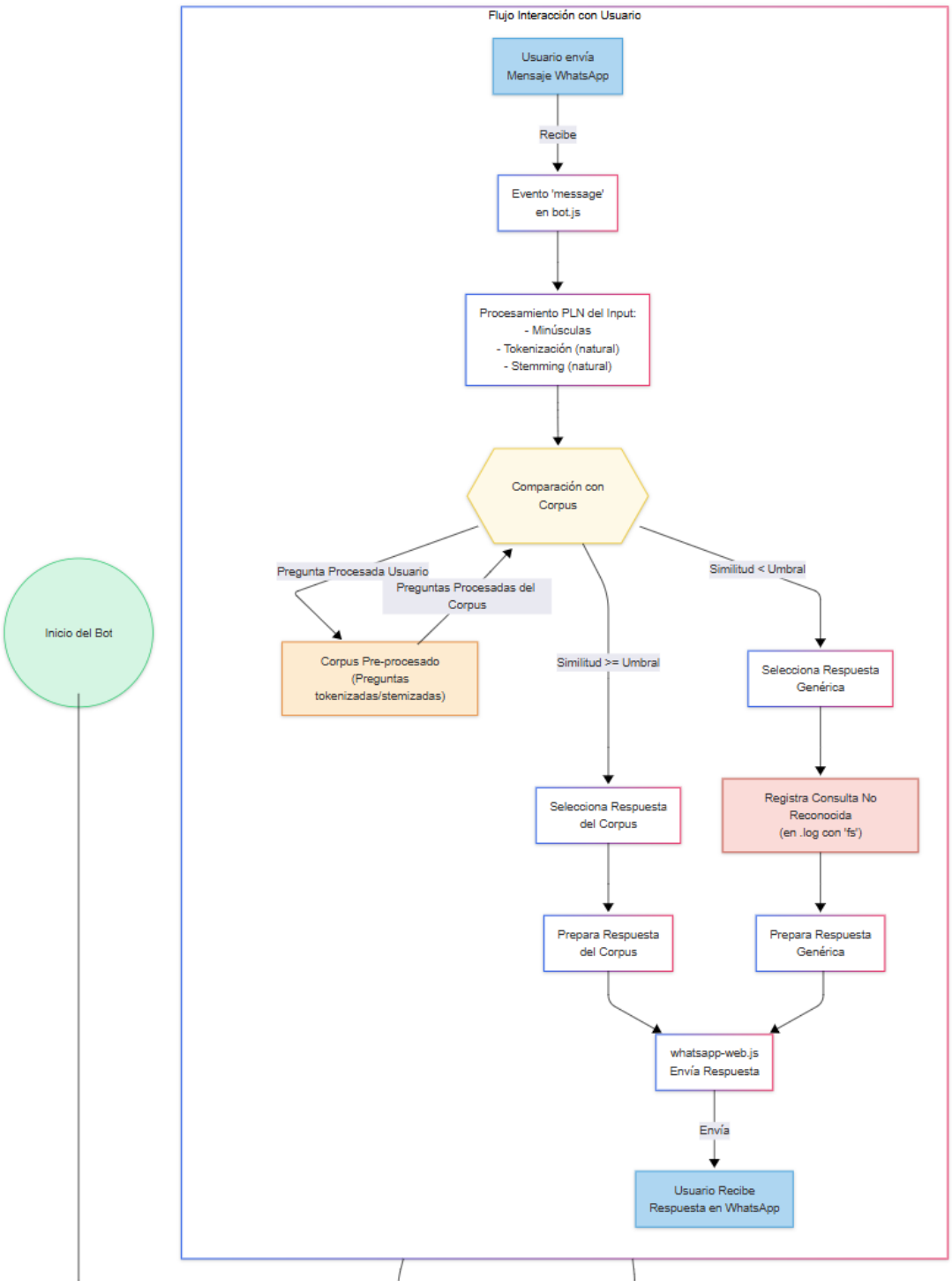
- **Diagrama de Flujo :**

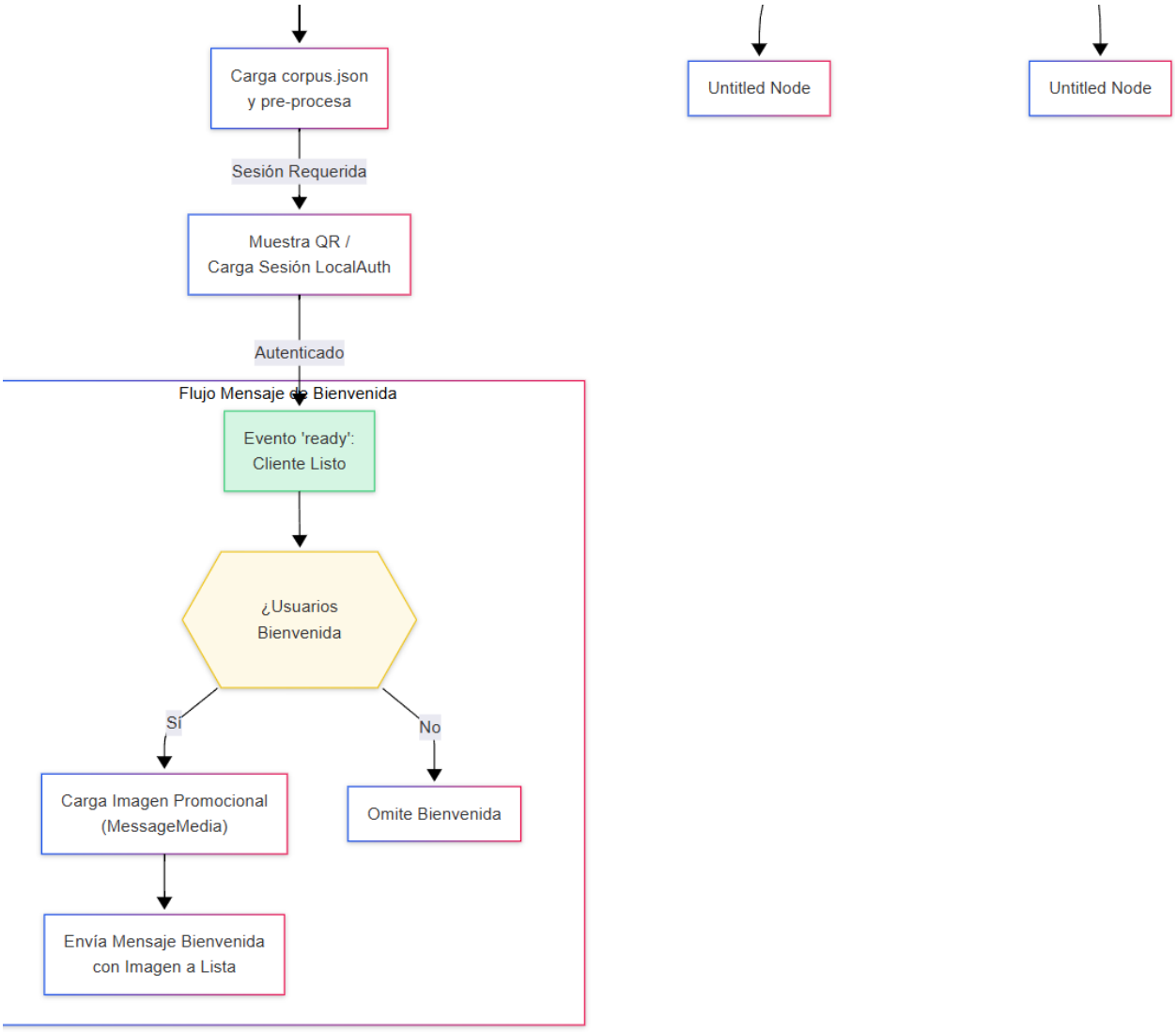


4. Mantenimiento y Escalabilidad

- **Mantenimiento del Corpus:** Fácilmente actualizable editando el archivo corpus.json externo.
- **Registro de Consultas:** El archivo consultas_no_reconocidas.log ayuda a identificar áreas de mejora para el corpus.
- **Autonomía:** El bot opera sin intervención manual tras el inicio y la autenticación. La persistencia de sesión (LocalAuth) reduce la necesidad de escanear el QR repetidamente.

Arquitectura técnica / lógica del Bot (flujo de interacción detallado)




Carga corpus.json
y pre-procesa
Sesión Requerida
Muestra QR /
Carga Sesión LocalAuth
Autenticado
Flujo Mensaje de Bienvenida
Evento 'ready':
Cliente Listo
¿Usuarios
Bienvenida
Sí
No
Carga Imagen Promocional
(MessageMedia)
Omite Bienvenida
Envía Mensaje Bienvenida
con Imagen a Lista
Untitled Node
Untitled Node