|  |
| --- |
| **VyCla - Manual del sistema** |
| **Clariza Maribel Angulo Castilo**  **Olga Vanessa Angulo Meza**  **Jorge Albeiro Rivera Rosero**  **Héctor Andrés Mora** |
|  |
| **GALERAS.NET** |
|  |
| **2025** |
|  |

Contenido

**Pág.**

[PRESENTACION 4](#_Toc201949963)

[INTRODUCCIÓN 5](#_Toc201949964)

[PROPÓSITO DEL MANUAL 5](#_Toc201949965)

[OBJETIVOS DEL MANUAL 5](#_Toc201949966)

[APLICATIVO DE ASISTENCIA VIRTUAL VYCLA 6](#_Toc201949967)

[MARCO TEORICO DEL SISTEMA 7](#_Toc201949968)

[METODOLOGIA 9](#_Toc201949969)

[DIAGRAMAS DE MODELAMIENTO 11](#_Toc201949970)

[Diagrama Arquitectónico 11](#_Toc201949971)

[Diagrama de paquetes 12](#_Toc201949972)

[Diagrama de Casos de uso 13](#_Toc201949973)

[GESTION DE LA INFORMACION 15](#_Toc201949974)

[COMPONENTES TÉCNICOS 19](#_Toc201949975)

[CONFIGURACION DEL ENTORNO 20](#_Toc201949976)

[Instrucciones configuración del entorno 21](#_Toc201949977)

[Ejecución de la aplicación 25](#_Toc201949978)

Ilustraciones

[Ilustración 1 Diagrama Metodologia SCRUM 1](#_Toc189829482)1

[Ilustración 2 Diagrama de Arquitectura del sistema 1](#_Toc189829483)2

[Ilustración 3 Diagrama de paquetes 1](#_Toc189829484)3

[Ilustración 4 Diagrama casos de usos 1 14](#_Toc189829485)

[Ilustración 5 Diagrama casos de usos 2 1](#_Toc189829486)4

[Ilustración 6 Diagrama casos de usos 3 1](#_Toc189829487)5

[Ilustración 7 Recepción y procesamiento de mensajes 1](#_Toc189829488)6

[Ilustración 8 Fragmento del código de BotManager.js 17](#_Toc189829489)

[Ilustración 9 Consulta a la base de datos PostgreSQL 19](#_Toc189829490)

[Ilustración 10 Estructura de IntentsCT.json 19](#_Toc189829491)

[Ilustración 11 Descarga de Node.js 21](#_Toc189829492)

[Ilustración 12 Descarga de Python 22](#_Toc189829493)

[Ilustración 13 Descarga de PostgreSQL 23](#_Toc189829494)

[Ilustración 14 Descarga de pgAdmin 4 23](#_Toc189829495)

[Ilustración 15 Iniciando la conexión con WhastsApp 25](#_Toc189829496)

[Ilustración 16 Ejecutando app.py 26](#_Toc189829497)

[Ilustración 17 Ejecutando apiaudio.py 2](#_Toc189829498)6

[Ilustración 18 Inicio de conversación con usuario 2](#_Toc189829499)6

# PRESENTACIÓN

Bienvenido al manual del sistema del chatbot VyCla. Este documento ha sido creado para proporcionar una guía completa sobre el funcionamiento de VyCla, un asistente virtual desarrollado para apoyar el proceso de inscripción y admisión de aspirantes a los programas académicos de pregrado de la Universidad de Nariño.

El presente manual contiene la información necesaria para la instalación, configuración y uso de VyCla. Ofrece una descripción detallada de la arquitectura del sistema, los componentes que lo integran y el flujo de interacción con los usuarios.

VyCla fue diseñado para integrarse con la plataforma WhatsApp, brindando el acceso de los aspirantes a información clave sobre requisitos, fechas, programas académicos, cálculo de ponderados, entre otros temas. Esta guía garantiza que cualquier persona interesada en implementar o mejorar el sistema pueda hacerlo de manera clara y efectiva, contribuyendo al fortalecimiento del servicio ofrecido por la Oficina de Control, Registro Académico y Admisiones (OCARA) de la Universidad de Nariño.

# INTRODUCCIÓN

VyCla es un chatbot desarrollado como herramienta de apoyo en el proceso de inscripción y admisión de aspirantes a los programas de pregrado de la Universidad de Nariño. Este asistente virtual permite brindar atención inmediata y personalizada a través de WhatsApp, brindando el acceso a información sobre cronogramas, requisitos, programas académicos, cálculo de ponderados y otros aspectos relacionados con el proceso de ingreso.

El sistema está construido sobre una arquitectura modular que combina tecnologías como Node.js para la gestión del flujo conversacional, Python para el procesamiento de lenguaje natural y la transcripción de mensajes de voz, y PostgreSQL para el almacenamiento de datos. Además, incorpora técnicas de web scraping para mantener la información actualizada directamente desde las plataformas institucionales.

Este manual tiene como finalidad proporcionar una guía completa sobre la estructura, funcionamiento, configuración y mantenimiento de VyCla. Aquí se documentan tanto los aspectos técnicos como operativos del sistema, permitiendo que tanto el personal administrativo como los desarrolladores puedan comprender su funcionamiento y realizar adecuadamente procesos de actualización, mejora o adaptación a nuevas necesidades.

# **PROPÓSITO DEL MANUAL**

El propósito de este manual es servir como una guía técnica y operativa para la comprensión, implementación y uso del chatbot VyCla. Está dirigido tanto a desarrolladores como a personal administrativo y usuarios encargados de su mantenimiento y mejora.

Este documento tiene como objetivo principal facilitar la apropiación del sistema, proporcionando información clara sobre su configuración, estructura, funcionamiento y actualización. Asimismo, busca asegurar una adecuada utilización del chatbot como herramienta de apoyo en el proceso de inscripción y admisión de aspirantes a programas de pregrado en la Universidad de Nariño.

Además, el manual pretende documentar las decisiones técnicas adoptadas durante el desarrollo del proyecto, de modo que futuras mejoras, adaptaciones o integraciones puedan realizarse de manera ordenada, garantizando la continuidad y escalabilidad del sistema.

# **OBJETIVOS DEL MANUAL**

**Objetivo general**

Brindar una guía detallada y estructurada para la comprensión, implementación y mantenimiento del chatbot VyCla, con el fin de garantizar su correcto funcionamiento como herramienta de apoyo en el proceso de inscripción y admisión de aspirantes a los programas de pregrado de la Universidad de Nariño.

**Objetivos específicos**

* Describir la arquitectura, componentes y tecnologías utilizadas en el desarrollo del chatbot VyCla, facilitando su comprensión técnica por parte de los desarrolladores y administradores del sistema.
* Instruir sobre los procedimientos necesarios para la instalación, configuración e integración del chatbot con la plataforma WhatsApp.
* Proporcionar lineamientos para la actualización, mantenimiento y mejora continua del sistema, asegurando su adaptabilidad a nuevas necesidades institucionales y su sostenibilidad en el tiempo.

# APLICATIVO DE ASISTENCIA VIRTUAL VYCLA

VyCla es un chatbot desarrollado como una solución tecnológica, con el objetivo de mejorar y automatizar el proceso de inscripción y admisión de aspirantes a los programas de pregrado de la Universidad de Nariño. Este asistente virtual integra técnicas de procesamiento de lenguaje natural (PLN) y una interfaz conversacional accesible mediante la plataforma WhatsApp.

El propósito central de VyCla es ofrecer atención inmediata, precisa y personalizada a las consultas frecuentes relacionadas con cronogramas, requisitos, programas académicos, cálculo de ponderados y otros aspectos del proceso de ingreso universitario. De este modo, se busca optimizar el tiempo de respuesta, reducir la carga operativa del personal de OCARA y garantizar una experiencia informativa más eficiente para los aspirantes.

VyCla está construido sobre una arquitectura modular distribuida en dos entornos de programación. Por un lado, el módulo desarrollado en JavaScript gestiona la lógica conversacional, los flujos de diálogo y la conexión con la API de WhatsApp. Por otro lado, el módulo de Python se encarga del procesamiento de lenguaje natural, empleando técnicas de clasificación de intenciones mediante reglas y patrones definidos en un archivo de configuración.

Un componente central del sistema es un archivo en formato JSON, que contiene toda la información necesaria para que VyCla entienda las intenciones del usuario y emita respuestas adecuadas. Este archivo almacena frases de ejemplo, etiquetas y respuestas asociadas, lo cual permite mantener una estructura ordenada, fácil de editar y escalar sin necesidad de modificar directamente el código fuente del sistema.

El desarrollo del sistema se realizó bajo la metodología ágil Scrum, organizando el trabajo en sprints que permitieron la entrega continua de versiones funcionales del chatbot. Cada iteración incluía planificación, revisión y pruebas, así como retroalimentación de usuarios reales, lo cual garantizó una mejora progresiva del sistema. Esta metodología permitió adaptarse rápidamente a los cambios y necesidades institucionales, manteniendo siempre un enfoque centrado en el usuario final.

La interfaz con WhatsApp asegura que el chatbot sea accesible para una amplia comunidad estudiantil, permitiendo que cualquier aspirante pueda obtener información confiable sin necesidad de desplazarse o hacer llamadas. VyCla está diseñado para responder de forma simultánea a múltiples usuarios, operando de manera autónoma las 24 horas del día.

Gracias a su diseño técnico, su enfoque modular y la metodología empleada, VyCla representa una herramienta eficiente, adaptable y sostenible para apoyar los procesos de admisión en la Universidad de Nariño.

# MARCO TEÓRICO DEL SISTEMA

VyCla es un chatbot diseñado para automatizar la atención de aspirantes a programas de pregrado en la Universidad de Nariño, especialmente en lo relacionado con el proceso de inscripción y admisión. Este asistente virtual integra múltiples tecnologías distribuidas en una arquitectura modular, orientada a la interpretación de lenguaje natural, la interacción conversacional, la consulta dinámica de información desde la web, la transcripción de mensajes de voz y el almacenamiento persistente de datos. El desarrollo se realizó bajo la metodología ágil Scrum, lo que permitió una implementación progresiva y centrada en el usuario.

**Procesamiento de Lenguaje Natural (PLN) en Python**

El módulo de PLN de VyCla fue implementado en Python, con el objetivo de clasificar las intenciones del usuario a partir de sus mensajes. Esta clasificación se basa en un archivo estructurado en formato JSON, el cual contiene:

* Intenciones (intents)
* Nombre de las intenciones (tag)
* Frases de ejemplo (patterns)
* Respuestas asociadas (responses)

A diferencia de sistemas basados en aprendizaje automático, VyCla emplea un enfoque basado en reglas y coincidencia de patrones definidos en un archivo JSON. Esto permite un mayor control sobre el comportamiento del chatbot y facilita su adaptación a nuevos contextos institucionales.

**Interacción y Flujo Conversacional en JavaScript**

La interacción en tiempo real con los usuarios se gestiona desde un módulo desarrollado en JavaScript, el cual realiza las siguientes funciones principales:

* Conexión con WhatsApp mediante Baileys

Se emplea la biblioteca Baileys, que permite establecer una sesión de WhatsApp Web controlada por código. A través de esta conexión, VyCla puede recibir mensajes en formato texto o audio y enviar las respuestas, brindando una experiencia conversacional directa y accesible.

* Gestión del flujo conversacional

Se implementan estructuras condicionales, manejo de estados y lógica de contexto para guiar al usuario según su intención.

* Extracción de información mediante Axios y Puppeteer

VyCla incorpora funciones de web scraping directamente en el módulo de JavaScript.

* Con Axios, se hacen peticiones a sitios con contenido estático.
* Con Puppeteer, se automatiza un navegador Chrome/Chromium para acceder y extraer contenido dinámico de páginas institucionales. Esto permite recuperar información actualizada desde sitios como OCARA, sin necesidad de mantenimiento manual de contenidos.

**Transcripción de Audio a Texto con Whisper**

Para mejorar la accesibilidad, VyCla permite a los usuarios enviar mensajes de voz. Estos audios son transcritos a texto mediante el modelo Whisper de OpenAI, integrado en el backend de Python. El flujo incluye:

* Recepción del audio desde WhatsApp.
* Conversión del archivo a formato compatible.
* Procesamiento con Whisper para extraer el texto.

El resultado es tratado como una entrada textual estándar, sin alterar el flujo natural de la conversación. Esta funcionalidad amplía la inclusión, facilitando el uso por parte de personas con dificultades para escribir.

**Almacenamiento de Mensajes en PostgreSQL**

VyCla implementa persistencia de datos mediante el sistema de gestión de bases de datos PostgreSQL, una solución relacional robusta y de alto rendimiento. Todos los mensajes recibidos por el chatbot, así como información contextual asociada al usuario (número de celular y nombre registrado en WhatsApp), son almacenados en tablas estructuradas.

Este almacenamiento permite:

* Auditar conversaciones.
* Monitorear el comportamiento del chatbot.
* Analizar percepción de los usuarios, ya que pueden responder una breve encuesta sobre el chatbot.

La base de datos se comunica con los módulos de JavaScript, garantizando integridad y trazabilidad de la información sin comprometer la privacidad del usuario.

# METODOLOGÍA

El desarrollo del chatbot VyCla se llevó a cabo bajo el marco de trabajo ágil Scrum, una metodología ampliamente utilizada en ingeniería de software que permite la entrega incremental y continua de productos funcionales. Scrum se centra en la colaboración constante, la adaptación al cambio y la mejora continua mediante ciclos de trabajo iterativos conocidos como sprints.

**Justificación del uso de Scrum**

Scrum fue elegido como metodología de desarrollo debido a su flexibilidad y su enfoque centrado en el usuario final. La naturaleza iterativa de Scrum permitió dividir el proyecto en pequeñas fases con objetivos específicos, facilitando:

* La entrega rápida de versiones funcionales de VyCla.
* La incorporación progresiva de funcionalidades clave.
* La recolección temprana de retroalimentación por parte de usuarios reales (aspirantes y personal institucional).
* La adaptación continua del sistema a requerimientos técnicos y contextuales.

**Organización del trabajo en sprints**

El proyecto se estructuró en sprints de duración fija, cada uno orientado al desarrollo, prueba y validación de funcionalidades específicas. Cada sprint incluyó las siguientes fases:

* Planificación del sprint: se definieron los objetivos, funcionalidades a desarrollar y tareas a ejecutar.
* Desarrollo: se implementaron los módulos correspondientes (en JavaScript y Python).
* Pruebas internas: se realizaron validaciones técnicas y funcionales para asegurar el correcto comportamiento de cada módulo.
* Revisión: se evaluó el producto construido y se identificaron mejoras.
* Retrospectiva: se discutieron los aciertos y aspectos a mejorar en el siguiente sprint.

**Cronograma de sprints y entregables**

A continuación, se describe la distribución de funcionalidades por sprint:

* Sprint 1: Inicio del proyecto
* Configuración del entorno de desarrollo.
* Conexión con WhatsApp mediante la biblioteca Baileys.
* Envío y recepción básica de mensajes de texto.
* Sprint 2: Motor de procesamiento de lenguaje natural
* Implementación del archivo JSON con intents, patterns y responses.
* Integración del módulo de PLN en Python para interpretar las intenciones.
* Desarrollo del flujo conversacional inicial.
* Sprint 3: Extracción de información web
* Implementación de funciones de web scraping en JavaScript usando Axios y Puppeteer.
* Automatización de la consulta de cronogramas, requisitos y enlaces desde sitios institucionales.
* Sprint 4: Accesibilidad mediante voz
* Recepción de mensajes de voz.
* Transcripción de audio a texto usando el modelo Whisper.
* Integración del texto transcrito al flujo de PLN.
* Sprint 5: Persistencia de datos y pruebas con usuarios reales
* Almacenamiento de mensajes y contexto conversacional en una base de datos PostgreSQL.
* Pruebas funcionales completas.
* Ajustes según retroalimentación de aspirantes y personal de OCARA.

**Herramientas de apoyo utilizadas**

Durante el desarrollo bajo Scrum se utilizaron herramientas para gestión colaborativa y control de versiones, como:

* Bitbucket: para el control de versiones del código.
* WhatsApp: como canal real de interacción para pruebas con usuarios.

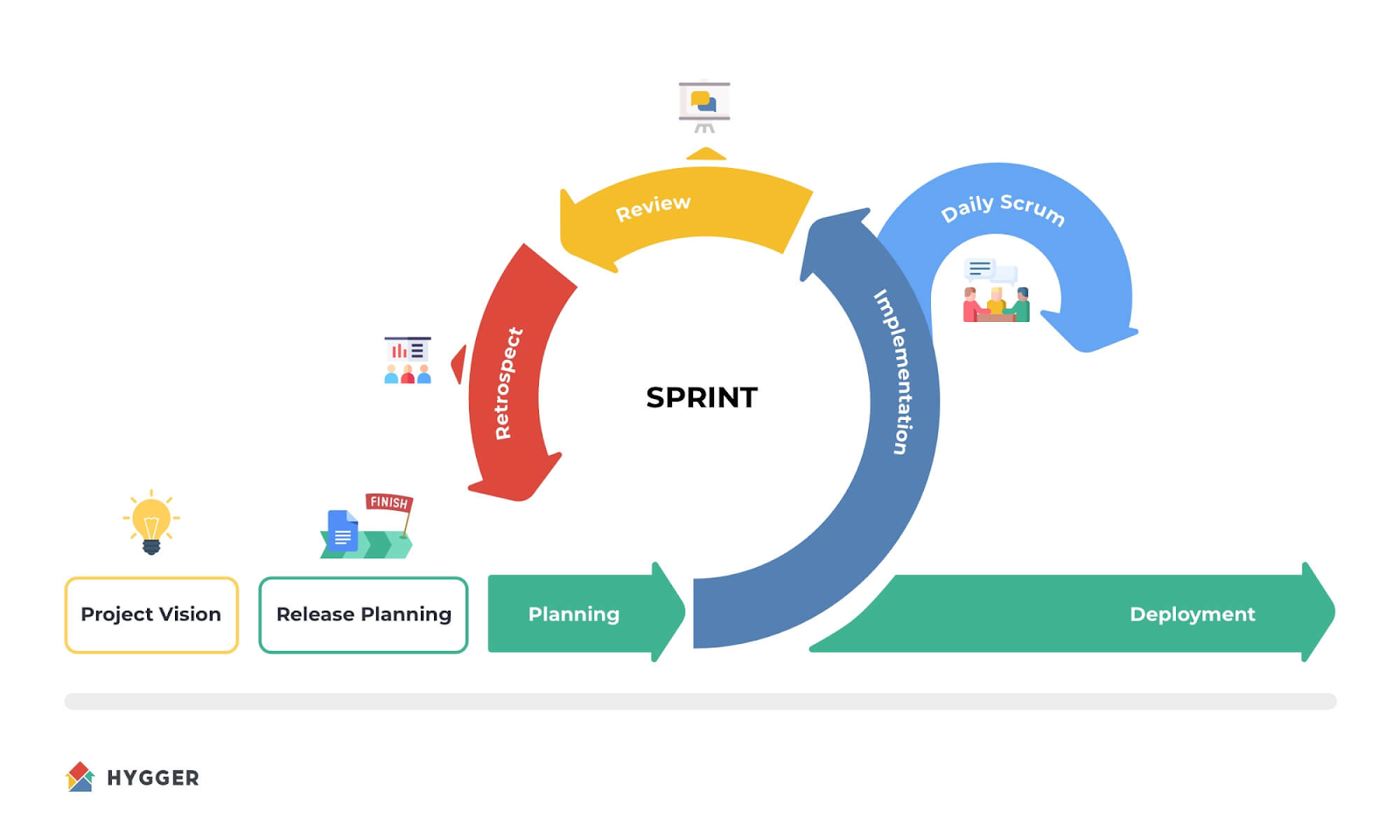


Ilustración 1 Diagrama Metodologia SCRUM

# DIAGRAMAS DE MODELAMIENTO

# Diagrama Arquitectónico

**Arquitectura**

La arquitectura del sistema VyCla se compone de tres módulos principales: la lógica conversacional en Node.js que se comunica con WhatsApp mediante Baileys, una API de procesamiento de lenguaje natural en Python, y una API de transcripción de audio también en Python utilizando Whisper. La información se extrae de sitios web institucionales mediante web scraping (Axios y Puppeteer), y todos los mensajes se almacenan en una base de datos PostgreSQL. El comportamiento del chatbot se basa en un archivo JSON construido a partir del levantamiento de procesos de OCARA.

Se detalla de la siguiente manera:

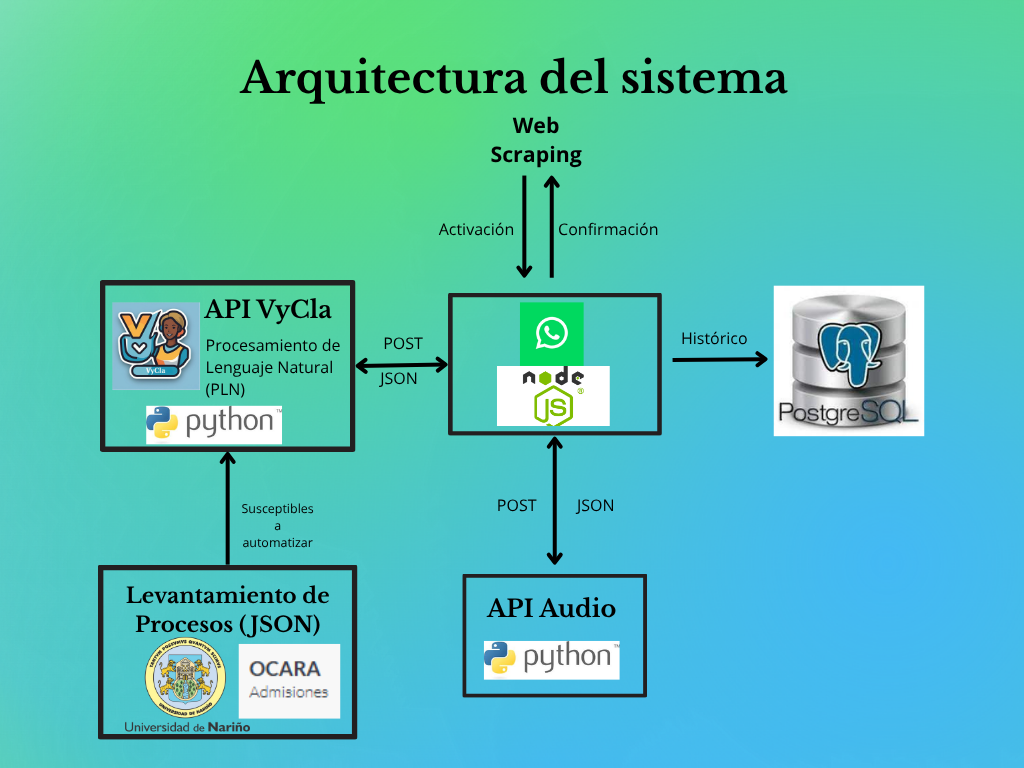


Ilustración 2 Diagrama de Arquitectura del sistema

# Diagrama de paquetes

El diagrama de paquetes del chatbot VyCla muestra una arquitectura modular compuesta por cinco paquetes principales. Controller gestiona el flujo conversacional mediante los archivos app.js y botmanager.js, además de interactuar con la base de datos. WhatsApp Interface permite la comunicación con los usuarios a través de la librería Baileys. El paquete NLP Processor contiene el modelo de predicción, el archivo de entrenamiento (ProcesarYentrenarModelo.py), el procesador de audio (aplaudio.py) y la API del modelo (app.py). JSON Intents almacena las intenciones y patrones lingüísticos en el archivo IntentsCT.json, que son clave para identificar las solicitudes de los usuarios. Finalmente, PostgreSQL gestiona la base de datos y el historial de usuarios, asegurando la persistencia de la información. Esta estructura permite que el chatbot funcione de manera eficiente, escalable y bien organizada.

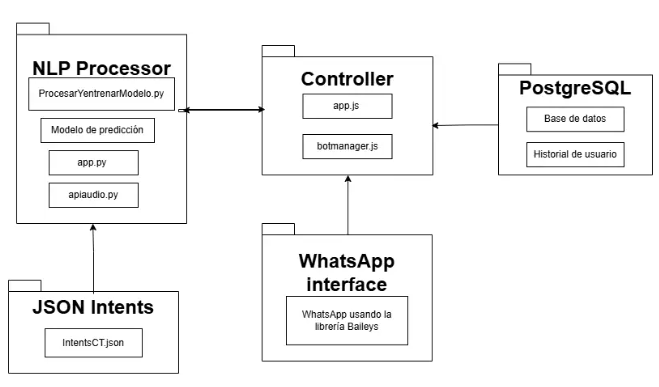


Ilustración 3 Diagrama de paquetes

# Diagrama de Casos de uso

El diagrama de casos de uso proporciona una representación visual de las interacciones entre los usuarios y el chatbot VyCla. Este sistema ha sido diseñado para apoyar el proceso de inscripción y admisión a programas de pregrado en la Universidad de Nariño, permitiendo a los aspirantes acceder de forma rápida y sencilla a información clave mediante una interfaz conversacional integrada a WhatsApp. Además de esta función principal, VyCla ofrece funcionalidades adicionales como la consulta de programas académicos por sede, el cálculo de puntaje ponderado a partir del ICFES, la visualización de fechas importantes del calendario de admisión, entre otros temas. A continuación, se presentan los diagramas que destacan todas las funciones del chatbot VyCla y cómo los aspirantes pueden interactuar con él durante el proceso de inscripción.

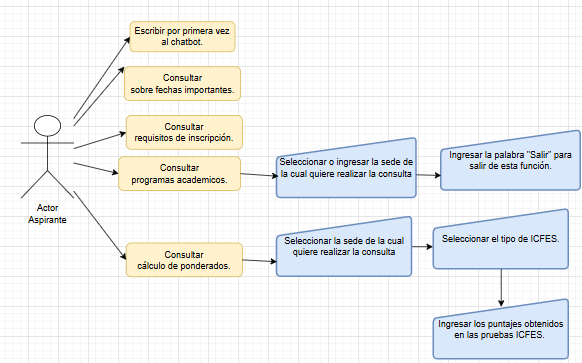


Ilustración 4 Diagrama casos de usos 1

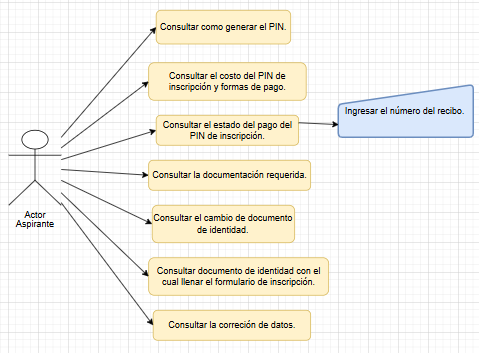


Ilustración 5 Diagrama casos de usos 2

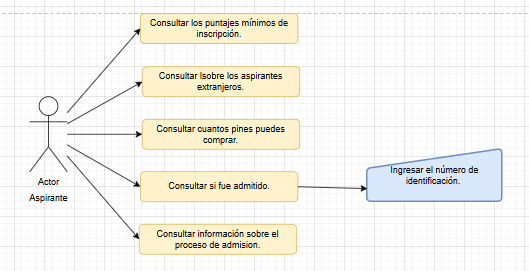


Ilustración 6 Diagrama casos de usos 3

# GESTIÓN DE LA INFORMACIÓN

En VyCla, la gestión de la información se centra en la recepción, procesamiento y almacenamiento de datos provenientes de las interacciones con los usuarios a través de la plataforma WhatsApp, utilizando la librería **Baileys** como intermediario de comunicación.

**Recepción y procesamiento de mensajes**

El archivo app.js, ubicado dentro del paquete Controller, es el responsable de la gestión principal del flujo conversacional. Este módulo recibe todos los mensajes entrantes de los usuarios, ya sean texto o notas de voz y realiza las siguientes funciones:

* Gestiona la sesión de WhatsApp con Baileys.
* Extrae los datos del mensaje recibido.
* Identifica si es un mensaje de texto o audio.
* Si es texto, lo procesa y lo envía al modelo de predicción.
* Si es un audio, activa el módulo de transcripción (aplaudio.py), para convertir el audio en texto antes de enviarlo al modelo.
* Obtiene la intención del mensaje mediante la API expuesta por app.py (modelo de predicción).
* Genera la respuesta correspondiente según la intención detectada y la envía al usuario.
* Registra el historial de conversaciones y datos relevantes en la base de datos PostgreSQL.

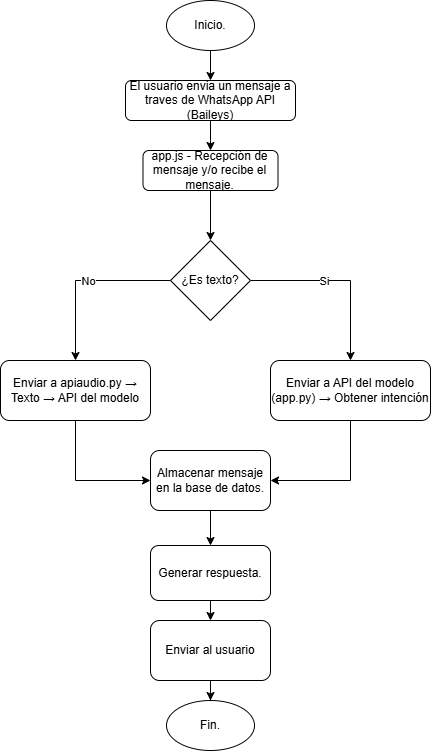


Ilustración 7 Recepción y procesamiento de mensajes

**Gestión de eventos asincrónicos: envío de encuestas**

El archivo BotManager.js también ubicado en el paquete Controller, cumple una función específica dentro de la arquitectura. Este módulo no interviene en la recepción de mensajes en tiempo real, sino que se activa cuando se detecta un período de 10 minutos de inactividad por parte del usuario, ejecutando una tarea automatizada de envío de encuestas de satisfacción.

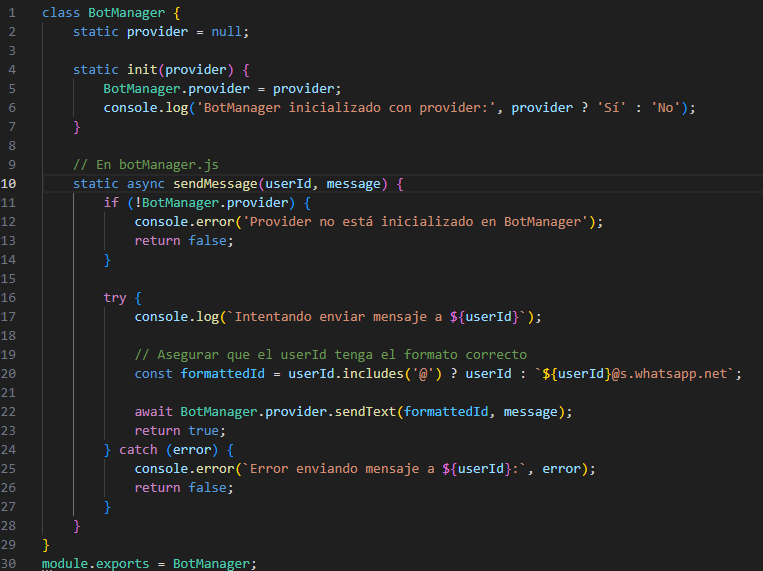


Ilustración 8 Fragmento del código de BotManager.js

Este código permite a VyCla enviar mensajes asincrónicos como encuestas tras tiempos de inactividad, el cual tanto el tiempo de inactividad del usuario y la encuesta como tal están codificados en el archivo app.js.

**Procesamiento del lenguaje natural y predicciones**

El paquete NLP Processor contiene el archivo app.py, que actúa como un microservicio API que recibe mensajes de texto y devuelve la intención detectada utilizando un modelo previamente entrenado. Además, incluye apiaudio.py, encargado de procesar mensajes de voz convirtiéndolos en texto antes de que sean enviados al modelo.

El archivo ProcesarYentrenarModelo.py se utiliza durante las fases de entrenamiento y actualización del modelo, permitiendo que VyCla adapte su comportamiento según los datos y patrones de conversación recolectados.

**Gestión de datos estructurados y no estructurados**

1. Datos no estructurados:

Mensajes de texto y notas de voz son procesados en tiempo real para identificar intenciones y generar respuestas. Los audios son transcritos a texto mediante aplaudio.py.

1. Datos estructurados:

En la base de datos **PostgreSQL**, se almacena exclusivamente el historial de los **mensajes enviados por los usuarios**, junto con datos asociados como:

* + El ID del usuario.
  + Número de contacto del usuario.
  + El mensaje enviado por el usuario.
  + Fecha de envio del mensaje.

Las respuestas generadas por el chatbot no se almacenan, ya que el foco de la gestión de datos está orientado a mantener un registro del comportamiento, solicitudes e interacciones de los usuarios, lo que permite analizar sus necesidades, optimizar el modelo y mejorar los procesos de atención.

Esta decisión responde tanto a criterios de optimización del almacenamiento como al enfoque analítico centrado en el comportamiento del usuario.

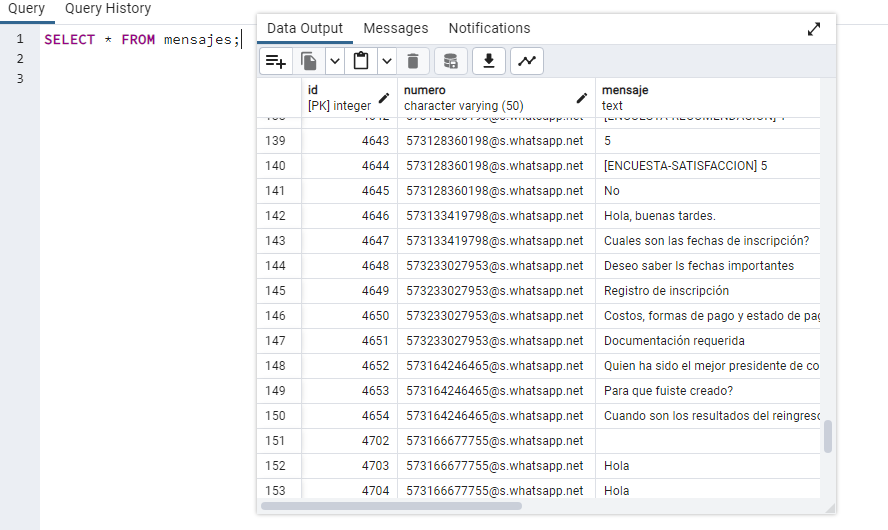
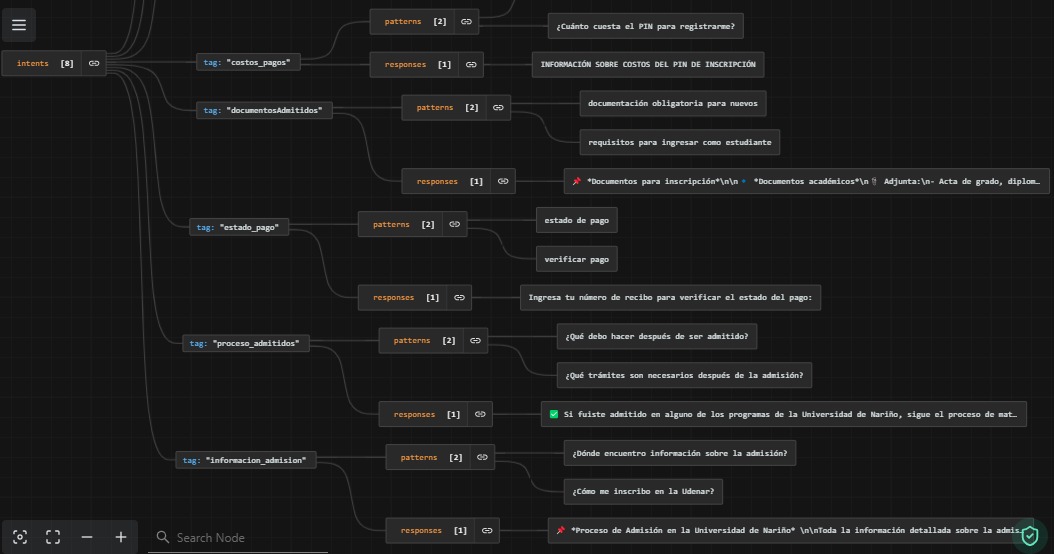


Ilustración 9 Consulta a la base de datos PostgreSQL

**Uso de JSON Intents**

El archivo IntentsCT.json, ubicado en el paquete JSON Intents, contiene la definición de las intenciones, patrones y respuestas del chatbot, las cuales son esenciales tanto para el entrenamiento como para la predicción de las intenciones durante la ejecución del servicio.



*Ilustración 10 Estructura de IntentsCT.json*

# COMPONENTES TÉCNICOS

Para el desarrollo del chatbot VyCla se han empleado tecnologías que garantizan un funcionamiento eficiente, escalabilidad y facilidad de integración con plataformas de mensajería. La selección de estas herramientas responde a los requerimientos actuales del sistema y a su proyección futura, asegurando una base sólida para su mantenimiento y evolución. En la siguiente tabla se detallan los componentes utilizados, organizados por categoría y nivel de relevancia:

|  |  |  |
| --- | --- | --- |
| CATEGORÍA | ELEMENTO | RELEVANCIA |
| Generales | | |
| Sistema operatio: | Windows 10/11 (desarrollo) | Alta |
| Backend | | |
| Lenguaje de programación | Python 3.12.1 | Alta |
| Framework web | Flask 2.3.2 | Alta |
| Librería NLP | NLTK | Alta |
| Librería NLP | WordNetLemmatizer (de NLTK) | Alta |
| Librería NLP | Stopwords (de NLTK) | Media |
| Librería NLP | Whisper | Media |
| Librería NLP | Pydub | Media |
| Librería ML | TensorFlow | Media |
| Librería ML | NumPy | Media |
| Librería de utilidades | OS, JSON, Random | Media |
| Frontend / Integración | | |
| Lenguaje de programación | JavaScript (Node.js) v20.11.0 | Alta |
| Librería bot | @bot-whatsapp/bot | Alta |
| Framework conexión | @bot-whatsapp/provider/baileys | Alta |
| Interfaz gráfica | @bot-whatsapp/portal (QR web) | Alta |
| Base de datos | PostgreSQL versión 17 con @bot-whatsapp/database/postgres | Alta |
| Cliente DB | Node-postgres (`pg`) | Alta |
| Automatización navegador | Puppeteer | Media |
| Cliente HTTP | Axios | Media |
| Manejo de archivos | Fs, path | Media |
| Manejo de PDFs | Pdf-parse | Media |
| Formularios | FormData | Media |
| Descarga multimedia | @whiskeysockets/baileys | Alta |
| Otros | | |
| Control de versiones | Bitbucket | Alta |
| Plataforma de despliegue | Repositorio local | Media |

# CONFIGURACIÓN DEL ENTORNO

# Instrucciones configuración del entorno

Para configurar y ejecutar correctamente el chatbot VyCla, es necesario realizar una serie de pasos de instalación y configuración del entorno. A continuación, se detallan las instrucciones para una instalación adecuada:

**1. Instalación de Node.js**

VyCla utiliza Node.js para gestionar el backend encargado de la conexión con WhatsApp y la gestión del flujo conversacional.

* Acceda a la página de descargas archivadas de Node.js v20.11.0: <https://nodejs.org/download/release/v20.11.0/>
* Seleccione la opción correspondiente a Windows.
* Proceda con la instalación guiada.

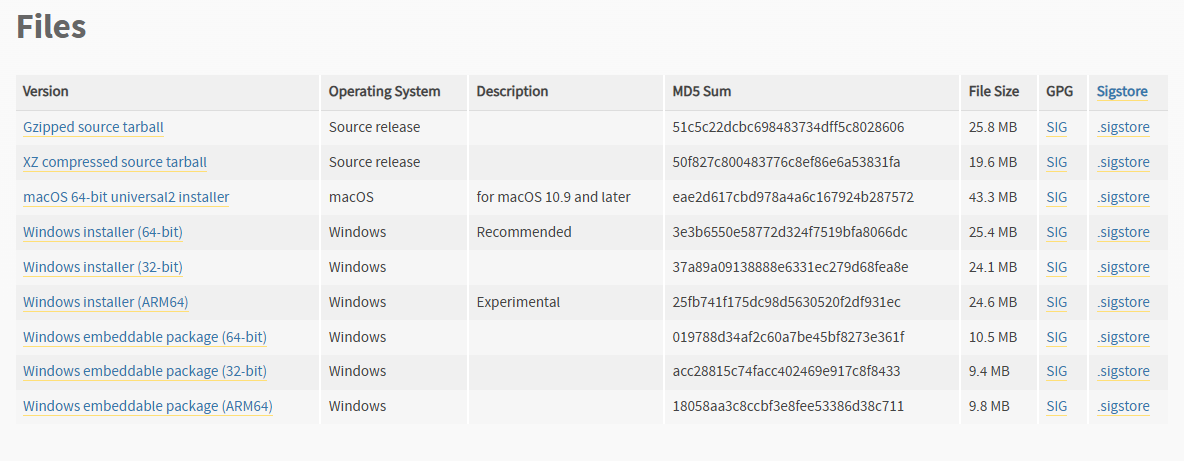
  
  
Ilustración 11 Descarga de Node.js

Para verificar la instalación, abra la terminal (cmd) y ejecute: node -v

**2. Instalación de Python**

VyCla utiliza Python para el procesamiento del lenguaje natural y la transcripción de audios.

* Ingrese a: <https://www.python.org/downloads/release/python-3121/>
* Diríjase a la sección de archivos y seleccione el correspondiente a su sistema operativo.

  
  
Ilustración 12 Descarga de Python

**Para verificar la instalación en cmd: python --version**

**3. Instalación de PostgreSQL**

Para gestionar la base de datos del historial de conversaciones, es necesario instalar PostgreSQL.

* Acceda al siguiente enlace para Windows: <https://www.enterprisedb.com/downloads/postgres-postgresql-downloads>
* Descargue el instalador con la versión 17 o superiores.
* Durante la instalación, configure una contraseña para el usuario postgres.

  
Ilustración 13 Descarga de PostgreSQL

**4. Instalación de PgAdmin 4**

Para la gestión y administración gráfica de la base de datos PostgreSQL, es necesario instalar pgAdmin 4, una herramienta oficial que permite realizar operaciones como:

* Creación de bases de datos.
* Creación de tablas.
* Consulta, inserción y modificación de datos.
* Administración de roles y permisos.

Acceda a la página oficial de descarga: <https://www.pgadmin.org/download/pgadmin-4-windows/>

****Teniendo en cuenta que se desarrolló en Windows, seleccione la versión para dicho sistema operativo, versión 7.8 o superiores.

Ilustración 14 Descarga de pgAdmin 4

Una vez instalado, podrá acceder a pgAdmin desde el menú de inicio o desde el navegador. Al abrirlo por primera vez, se le pedirá establecer una contraseña maestra que protege el acceso a las conexiones de bases de datos guardadas.

Posteriormente, puede realizar la conexión a PostgreSQL local con los datos de:

* Host: localhost
* Puerto: 5432
* Usuario: postgres
* Contraseña: (la configurada en la instalación de PostgreSQL

**5. Instalación de dependencias de Node.js**

Dentro de la carpeta donde se encuentra el archivo app.js, abra la terminal y ejecute:  
**npm install**Esto instalará las dependencias declaradas en package.json, incluyendo Baileys, dotenv, pg y otras dependencias.

**6. Instalación de bibliotecas para Python**

Dentro del directorio del microservicio Python (donde están app.py y aplaudio.py), cree un entorno virtual con el comando: **python -m venv venv**  
Active el entorno: Windows: venv\Scripts\activate

Instale las siguientes bibliotecas:  
- Flask: pip install flask  
- TensorFlow: pip install tensorflow== 2.19.0  
- NumPy: pip install numpy==1.26.4

- Whisper: pip install -U openai-whisper  
- Pydub: pip install pydub

**6. Configuración de variables de entorno**

Cree un archivo .env en la raíz del proyecto Node.js con el siguiente contenido:  
  
POSTGRES\_DB\_HOST=localhost

POSTGRES\_DB\_USER=postgres

POSTGRES\_DB\_PASSWORD=\*\*\*\*\*\*

POSTGRES\_DB\_NAME=VyCla01

POSTGRES\_DB\_PORT=5433

# Ejecución de la aplicación

Para la ejecución del chatbot, es indispensable contar con el entorno de desarrollo Visual Studio Code, así como con los entornos de ejecución de Node.js y Python, además de la herramienta de administración de bases de datos pgAdmin 4 para la gestión de PostgreSQL. La puesta en marcha del sistema se realiza mediante la ejecución de los comandos npm start, python app.py y python apiaudio.py dentro de Visual Studio Code. Si se desea actualizar información en el archivo IntensCT.json, se puede realizar siguiendo la estructura del mismo y volver a entrenar el modelo con el comando python ProcesarYentrenarModelo.py, luego de esto reliazar los anteriors comandos para su ejecución.

Es fundamental que, antes de ejecutar el sistema, la base de datos en PostgreSQL se encuentre debidamente creada, incluyendo la tabla y las columnas necesarias para el correcto almacenamiento y gestión de la información generada por el chatbot.

Cabe destacar que, durante la primera ejecución, el sistema solicita el escaneo de un código QR con el fin de vincular la cuenta de WhatsApp, lo cual permite establecer la conexión entre la aplicación y el número desde el cual operará el chatbot. Este proceso de vinculación solo se realiza una vez, siempre y cuando no se cierre la sesión del número asociado.

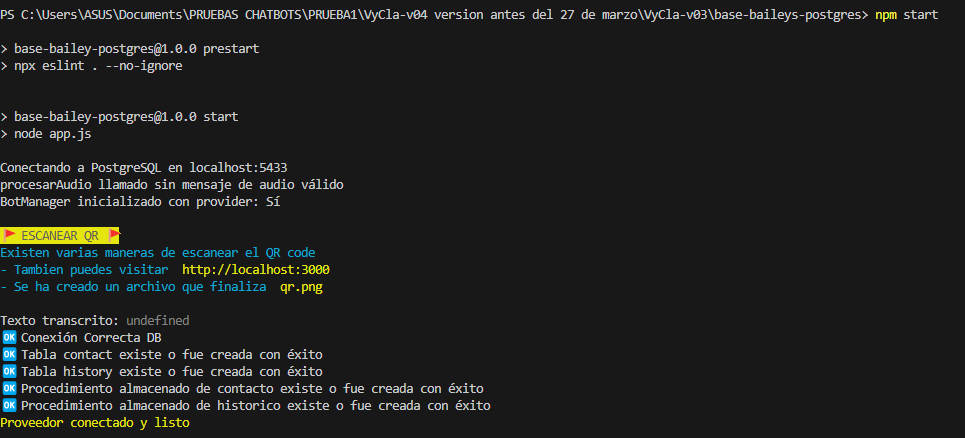


Ilustración 15 Iniciando la conexión con WhastsApp

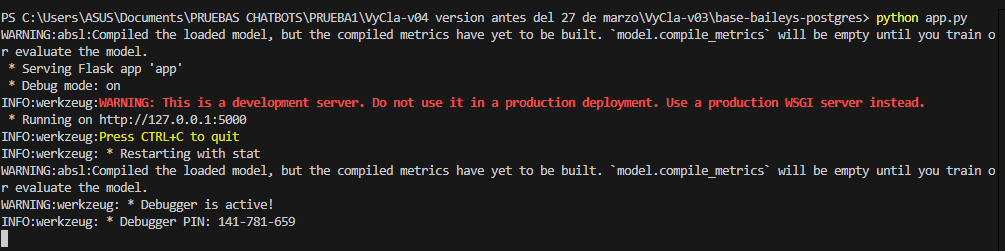


Ilustración 16 Ejecutando app.py

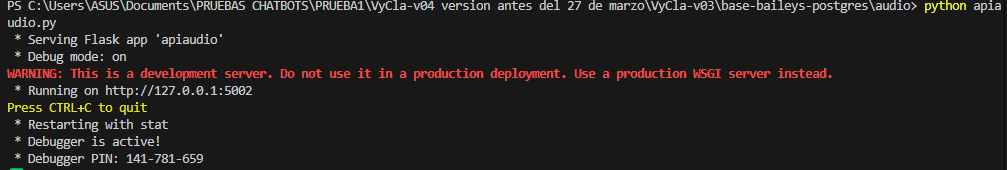


Ilustración 17 Ejecutando apiaudio.py

Tras la ejecución exitosa de todos los componentes del sistema y con el chatbot operativo, el usuario puede comenzar a interactuar con él mediante mensajes, como se evidencia en la Ilustración 18.

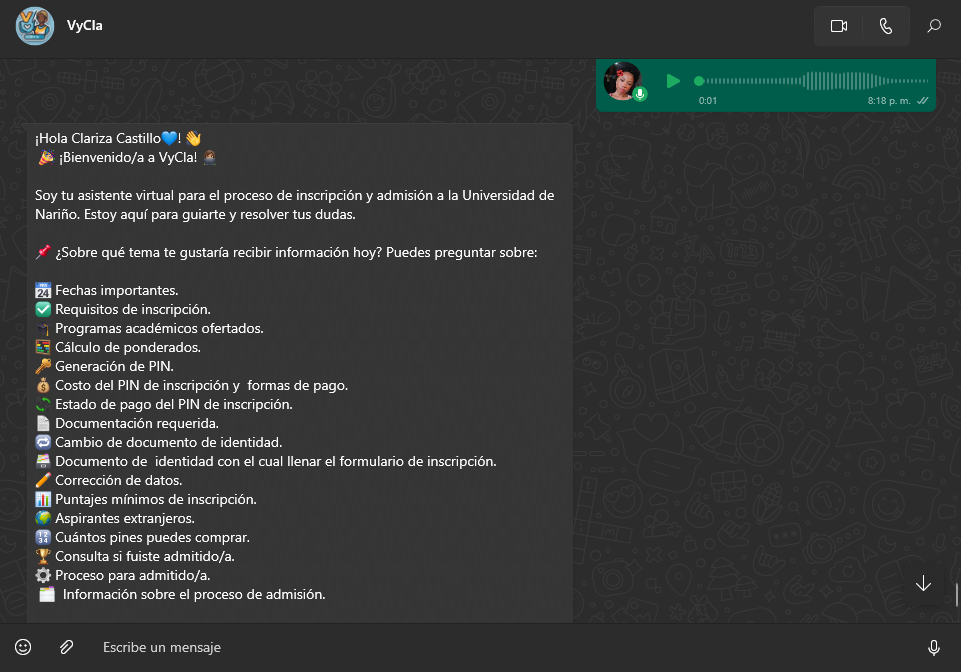


Ilustración 18 Inicio de conversación con usuario