


**UNIVERSIDAD TÉCNICA DE AMBATO**

**FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL**

**CARRERA DE TELECOMUNICACIONES**

**Tema:**

# SISTEMA DE RESPUESTA INTERACTIVA AUTOMÁTICA CON AGENTES DE IA PARA TELÉFONOS IP USANDO ASTERISK

Trabajo de titulación modalidad Proyecto de Investigación, presentado previo a la obtención del título de Ingeniera en Telecomunicaciones

**ÁREA*:*** Programación y Redes

**LÍNEA DE INVESTIGACIÓN:** Electrónica, Telecomunicaciones y Sistemas de control

**AUTOR:** María Gabriela Gallegos Salazar

**TUTOR:** Ing. Jesús Israel Guamán Molina, Mg.

**Ambato - Ecuador**

**julio – 2025**

## APROBACIÓN DEL TUTOR

En calidad de tutor del trabajo de titulación con el tema: SISTEMA DE RESPUESTA INTERACTIVA AUTOMÁTICA CON AGENTES DE IA PARA TELÉFONOS IP USANDO ASTERISK, desarrollado bajo la modalidad Proyecto de Investigación por la señorita María Gabriela Gallegos Salazar, estudiante de la Carrera de Telecomunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, me permito indicar que la estudiante ha sido tutorada durante todo el desarrollo del trabajo hasta su conclusión, de acuerdo a lo dispuesto en el Artículo 17 del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato y el numeral 6.3 del instructivo del reglamento referido.

Ambato, julio 2025.

Firmado electrónicamente por:
JESUS ISRAEL GUAMAN MOLINA
Validar únicamente con FirmaEC

---

Ing. Jesús Israel Guamán Molina, Mg.
TUTOR

## AUTORÍA

El presente trabajo de titulación con el tema: SISTEMA DE RESPUESTA INTERACTIVA AUTOMÁTICA CON AGENTES DE IA PARA TELÉFONOS IP USANDO ASTERISK es absolutamente original, auténtico y personal y ha observado los preceptos establecidos en la Disposición General Quinta del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato. En tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, julio 2025.



---

María Gabriela Gallegos Salazar

C.C. 1804343430

AUTOR

## DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato para que reproduzca total o parcialmente este trabajo de titulación dentro de las regulaciones legales e institucionales correspondientes. Además, cedo todos mis derechos de autor a favor de la institución con el propósito de su difusión pública, por lo tanto, autorizo su publicación en el repositorio virtual institucional como un documento disponible para la lectura y uso con fines académicos e investigativos de acuerdo con la Disposición General Cuarta del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato.

Ambato, julio 2025.



---

María Gabriela Gallegos Salazar

C.C. 1804343430

AUTOR

## APROBACIÓN DEL TRIBUNAL DE GRADO

En calidad de par calificador del informe final del trabajo de titulación presentado por la señorita María Gabriela Gallegos Salazar, estudiante de la Carrera de Telecomunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación, titulado SISTEMA DE RESPUESTA INTERACTIVA AUTOMÁTICA CON AGENTES DE IA PARA TELÉFONOS IP USANDO ASTERISK, nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Artículo 19 del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato y el numeral 6.4 del instructivo del reglamento referido. Para cuya constancia suscribimos, conjuntamente con el señor Presidente del Tribunal.

Ambato, julio 2025.

Firmado electrónicamente por:
FRANKLIN OSWALDO MAYORGA MAYORGA
Validar únicamente con FirmaEC

---

Ing. Franklin Oswaldo Mayorga Mayorga, Mg.
PRESIDENTE DEL TRIBUNAL

Firmado electrónicamente por:
GEOVANNI DANILO BRITO MONCAYO
Validar únicamente con FirmaEC

---

Ing. Geovanni Brito Moncayo, Mg
PROFESOR CALIFICADOR

ELIZABETH PAULINA AYALA BANO

Firmado digitalmente porELIZABETH PAULINA AYALA BANO
DN: cn=ELIZABETH PAULINA AYALA BANO c=EC o=SECURITY DATA S.A. 2 ou=ENTIDAD DE CERTIFICACION DE INFORMACION
Motivo:Soy el autor de este documento
Ubicaci- n:
Fecha:2025-07-15 17:35-05:00

---

Ing. Elizabeth Ayala Baño, Mg
PROFESOR CALIFICADOR

## DEDICATORIA

*A mis padres, por ser mi ejemplo de fortaleza, trabajo y amor incondicional. Gracias por enseñarme con su vida que los sueños se alcanzan con esfuerzo y humildad.*

*A mi esposo, por estar siempre a mi lado, por su paciencia, su apoyo inquebrantable y por creer en mí incluso en los momentos en los que yo dudé.*

*Y a mi hijo, mi mayor inspiración, la razón más hermosa para seguir adelante. Este logro es también para ti, porque cada paso que doy, lo hago pensando en el futuro que mereces.*

*Y a mis hermanos, Andy y Chris, por su cariño, sus palabras de aliento y por estar presentes en mi vida. Gracias por ser parte de mi fuerza.*

*Con todo mi amor, para ustedes.*

## AGRADECIMIENTO

*A Dios, por haberme dado la fortaleza, la salud y la sabiduría para seguir adelante, incluso en los momentos más difíciles.*

*A la Virgen María, por cubrirme con su manto y acompañarme con amor maternal en cada paso de este camino.*

*A la Universidad Técnica de Ambato, por brindarme la formación académica y las herramientas necesarias para crecer como profesional y como persona.*

*Al Ing. Jesús Guamán, mi tutor, por su guía, paciencia y dedicación durante el desarrollo de este trabajo. Su apoyo fue fundamental para alcanzar este logro.*

*Al Ing. Santiago Altamirano, por sus valiosos aportes, tiempo y orientación, que enriquecieron mi proceso formativo y a todos los docentes.*

*A Darío Chuquitarco, por estar presente siempre, por su apoyo incondicional, sus palabras de aliento y su amistad sincera que han sido un gran sostén en este camino.*

*A Danny y a todos mis amigos, gracias por acompañarme, por su comprensión y alegría. Cada uno ha sido parte importante de esta etapa que hoy culmina.*

*A toda mi familia y a mis seres amados que partieron, pero viven en mi corazón.*

*A todos ustedes, gracias de corazón, Totus tuus.*

## ÍNDICE GENERAL DE CONTENIDOS

**PORTADA** ..... **i**

**APROBACIÓN DEL TUTOR** ..... **ii**

**AUTORÍA** ..... **iii**

**DERECHOS DE AUTOR** ..... **iv**

**APROBACIÓN DEL TRIBUNAL DE GRADO** ..... **v**

**DEDICATORIA** ..... **vi**

**AGRADECIMIENTO** ..... **vii**

**ÍNDICE GENERAL DE CONTENIDOS** ..... **viii**

**NDICE DE TABLAS** ..... **xiii**

**ÍNDICE DE FIGURAS** ..... **xv**

**ÍNDICE DE ANEXOS** ..... **xviii**

**RESUMEN EJECUTIVO** ..... **xix**

**ABSTRACT** ..... **xx**

**CAPÍTULO I. MARCO TEÓRICO** ..... **1**

1.1 Tema de investigación ..... 1

1.1.1 Planteamiento del problema ..... 1

1.2 Antecedentes investigativos ..... 3

1.3 Fundamentación teórica ..... 5

1.3.1 Telefonía IP ..... 6

1.3.2 Sistema de Respuesta Interactiva Automática (IVR) ..... 11

1.3.3 Arquitectura del sistema ..... 11

1.3.4 Interacción con los usuarios ..... 11

1.3.5 Implementación dinámica ..... 12

1.3.6 Centros de atención telefónica ..... 12

1.3.7 Sistema PBX ..... 13

1.3.8 Arquitectura de Asterisk y sus componentes principales ..... 15

1.3.9 Proveedores de servicios en la nube ..... 16

1.3.10 Ventajas de Azure para implementaciones de Asterisk ..... 18

1.3.11 Modelos de lenguaje para interfaces conversacionales ..... 19

1.3.12 Integración de modelos GPT en sistemas Asterisk mediante AGI ..... 21

1.3.13 Consideraciones técnicas para implementación de Modelos de Lenguaje de Gran Escala en Asterisk ..... 23

1.3.14 Sistema Operativo para implementaciones de Asterisk ..... 24

1.3.15 PJSIP: la evolución del protocolo SIP en Asterisk ..... 26

1.4 Objetivos ..... 28

1.4.1 Objetivo general ..... 28

1.4.2 Objetivos específicos ..... 28

**CAPÍTULO II. METODOLOGÍA ..... 29**

2.1 Materiales ..... 29

2.2 Métodos ..... 32

2.2.1 Modalidad de la investigación ..... 32

2.2.2 Población y muestra ..... 32

2.2.3 Recolección de información ..... 33

2.2.4 Procesamiento y análisis de datos ..... 34

**CAPÍTULO III. RESULTADOS Y DISCUSIÓN ..... 35**

3.1 Desarrollo de la propuesta ..... 35

3.2 Diagrama de arquitectura del sistema ..... 35

3.2.1 Arquitectura del sistema ..... 37

3.3 Selección de tecnologías ..... 38

3.4 Identificación de las necesidades de los estudiantes ..... 40

3.5 Microsoft Azure para los servicios de la máquina virtual en la nube ..... 43

3.5.1 Selección de recursos para Asterisk en Azure ..... 43

3.5.2 Configuración de red para comunicaciones VoIP ..... 44

3.5.3 Alta disponibilidad y respaldo ..... 46

3.6 Azure AI Foundry para la integración de servicios cognitivos ..... 46

3.6.1 Integración de APIs de GPT-4 mediante Azure OpenAI Service ..... 47

3.6.2 Implementación de Azure AI Speech Services para procesamiento de voz ..... 48

3.6.3 Personalización del modelo GPT-4 para el contexto académico ..... 48

3.7 Instalación y configuración del entorno Asterisk en AlmaLinux ..... 49

3.7.1 Preparación del sistema operativo base..... 49

3.7.2 Instalación de Asterisk desde fuentes ..... 50

3.7.3 Configuración del entorno Python para integración con AGI ..... 51

3.7.4 Configuración de la estructura de los directorios para AGI..... 52

3.8 Desarrollo e implementación del script AGI principal ..... 53

3.8.1 Configuración del archivo tesis.py..... 53

3.8.2 Asignación del dialplan PJSIP para las extensiones ..... 65

3.8.3 Instauración del archivo de configuración extensions.conf ..... 69

3.8.4 Ajustes de las extensiones para los equipos finales ..... 73

3.9 Testeo de los diferentes parámetros de salida ..... 78

3.10 Análisis de la operadora inteligente ..... 80

3.10.1 Análisis del módulo de síntesis de voz con Azure AI Speech Services ..... 80

3.10.2 Análisis del componente de procesamiento de lenguaje natural con Azure OpenAI……..81

3.10.3 Diagrama del sistema de la Operadora IA ..... 86

3.11 Resultados ..... 90

3.11.1 Evaluación global del sistema IVR inteligente ..... 91

3.11.2 Análisis de precisión de las respuestas ..... 92

3.11.3 Análisis de tiempos de respuesta..... 94

3.11.4 Rendimiento por tipo de consulta ..... 95

3.11.5 Análisis de errores..... 95

3.12 Costos ..... 97

3.12.1 Costos de infraestructura en la nube ..... 97

3.12.2 Costos de equipamiento físico ..... 98

3.12.3 Costos de software ..... 99

3.12.4 Costos laborales de desarrollo ..... 99

3.12.5 Costos totales del proyecto ..... 100

**CAPÍTULO IV. CONCLUSIONES Y RECOMENDACIONES ..... 103**

4.1 Conclusiones ..... 103

4.2 Recomendaciones ..... 104

**REFERENCIAS BIBLIOGRÁFICAS ..... 107**

**ANEXOS ..... 114**

# ÍNDICE DE TABLAS

Tabla 1. Ventajas y desventajas de la implementación de la telefonía IP[17]. ..... 8

Tabla 2. Protocolos[18][19][21][22] ..... 9

Tabla 3. Comparativa de las diferentes PBX[34],[35],[36],[37],[38]. ..... 14

Tabla 4. Servicios de máquinas en la nube presentes para Asterisk[48],[49],[50],[51],[52],[53]. ..... 17

Tabla 5. Modelos de lenguaje AI para integración con Asterisk mediante AGI[63],[64]. ..... 20

Tabla 6. Sistemas operativos para implementaciones de Asterisk[66][67][68][69] .. 25

Tabla 7. Materiales y dispositivos. ..... 29

Tabla 8. Software y servicios de aplicación. ..... 30

Tabla 9. Tabla de parámetros ..... 33

Tabla 10. Comandos de instalación. ..... 49

Tabla 11. Código de Asterisk para las extensiones. ..... 50

Tabla 12. Códigos de instalación. ..... 51

Tabla 13. Códigos de instalación en Python. ..... 52

Tabla 14. Códigos para los scripts AGI. ..... 52

Tabla 15. Resumen de resultados de las pruebas del sistema IVR inteligente ..... 91

Tabla 16. Distribución de tiempos de respuesta ..... 94

Tabla 17. Rendimiento del sistema por categoría de consulta ..... 95

Tabla 18. Respuestas Incorrectas a detalle ..... 96

Tabla 19. Costos mensuales de servicios en Microsoft Azure ..... 97

Tabla 20. Costos de equipamiento físico ..... 98

Tabla 21. Costos de software ..... 99

Tabla 22. Costos Laborales ..... 99

Tabla 23. Consolidación de costos del proyecto ..... 101

Tabla 24. Resultados de la evaluación del sistema. ..... 129

# ÍNDICE DE FIGURAS

Figura 1. Diagrama de arquitectura del sistema ..... 36

Figura 2. Arquitectura del sistema. ..... 37

Figura 3. Encuesta realizada a los estudiantes para el seguimiento de información académica o administrativa. ..... 40

Figura 4. Encuesta realizada a los estudiantes para el seguimiento de información que necesitan con frecuencia. ..... 41

Figura 5. Encuesta realizada a los estudiantes para el seguimiento de información que necesita. ..... 41

Figura 6. Encuesta realizada a los estudiantes para el seguimiento de lugares donde suele buscar información. ..... 41

Figura 7. Encuesta realizada a los estudiantes para el seguimiento de la frecuencia con que reciben la información necesitada. ..... 42

Figura 8. Encuesta realizada a los estudiantes para el seguimiento de que debería tener el sistema automatizado. ..... 42

Figura 9. Importación de las diferentes librerías en formato .py ..... 53

Figura 10. Endpoint de Azure para GPT4 ..... 55

Figura 11. Configuración de la API de Voz de Azure AI ..... 56

Figura 12. Configuración de los PROMPTs para el modelo GPT4 ..... 57

Figura 13. Configuración de los parámetros de Voz de Azure AI. ..... 58

Figura 14. Conversión al formato valido de audio para Asterisk ..... 59

Figura 15. Uso del agi.verbose para los logs del CLI de Asterisk ..... 60

Figura 16. Salida de error si existe alguna falla con los servidores de Azure AI ..... 60

Figura 17. Selección del archivo de información académica ..... 61

Figura 18. Condiciones para el archivo de texto ..... 61

Figura 19. Inicio de la funcion def main() ..... 62

Figura 20. Configuración de audio de reconocimiento ..... 62

Figura 21. Configuración el agi.verbose para las respuestas de audio ..... 64

Figura 22. Estructura del contexto [transport-udp] ..... 65

Figura 23. Estructura general para la extensión 100 ..... 66

Figura 24. Configuración para la extensión 200 ..... 68

Figura 25. Contexto default de la parte de extensions.conf ..... 70

Figura 26. Contexto para la configuración de las extensiones ..... 71

Figura 27. Extensiones adicionales y la 250, especializada para consultas ..... 72

Figura 28. Configuración del cliente 200 en MicroSIP ..... 74

Figura 29. Configuración de la extensión 811 en Zoiper App ..... 75

Figura 30. Llamada saliente desde la extensión 200 ..... 78

Figura 31. Llamada saliente desde la extensión 811 ..... 79

Figura 32. Estructura general del servicio de Voz AI ..... 80

Figura 33. Condición si la conversión es exitosa ..... 81

Figura 34. Error si la conversión llegase a fracasar ..... 81

Figura 35. Llamado de la API hacia el script ..... 82

Figura 36. Procesamiento lingüístico ..... 82

Figura 37. Mecanismo ante limitaciones de la API ..... 83

Figura 38. Diagrama de flujo de datos del sistema Asterisk propio ..... 87

Figura 39. Llamada al sistema para verificación del funcionamiento ..... 90

Figura 40. Distribución de precisión en las respuestas del sistema IVR ..... 93

# ÍNDICE DE ANEXOS

Anexo A. Código pjsip..... 114

Anexo B. Código extensions.conf..... 116

Anexo C. Código archivo tesis.ty ..... 117

Anexo D. Código archivo lectura.py..... 122

Anexo E. Código testeo para Voz de Azure AI voz.py ..... 125

## RESUMEN EJECUTIVO

El presente trabajo aborda la integración de tecnologías de inteligencia artificial con sistemas de telefonía IP para desarrollar un sistema de respuesta interactiva automática basado en Asterisk. Este proyecto surge ante la necesidad de modernizar los procesos de consulta administrativa y académica en la Facultad de Ingeniería en Sistemas, Electrónica e Industrial de la Universidad Técnica de Ambato, donde tradicionalmente los estudiantes enfrentan tiempos de espera prolongados y recursos humanos limitados para atender sus inquietudes. El sistema propuesto transforma la experiencia de consulta telefónica mediante una arquitectura que combina una central Asterisk con servicios cognitivos avanzados, permitiendo interacciones en lenguaje natural. La implementación integra servicios en la nube de Microsoft Azure para el procesamiento semántico y síntesis de voz, posibilitando consultas académicas sin restricciones de horario o disponibilidad de personal. La metodología empleada comprende el diseño de un entorno virtualizado en la nube, la configuración de protocolos de comunicación SIP, el desarrollo de scripts AGI (Asterisk Gateway Interface) en Python, y la integración con modelos de lenguaje de gran escala para el procesamiento contextual de consultas académicas. Este trabajo explora las posibilidades de convergencia entre telecomunicaciones e inteligencia artificial, ofreciendo una solución innovadora y escalable que potencialmente puede extenderse a otros ámbitos institucionales, estableciendo nuevos paradigmas en la automatización de servicios universitarios mediante tecnologías emergentes.



**Palabras clave:** Asterisk, Inteligencia artificial, IVR, Telefonía IP

## ABSTRACT

This paper addresses the integration of artificial intelligence technologies with IP telephony systems to develop an automatic interactive response system based on Asterisk. This project arises from the need to modernize the administrative and academic consultation processes in the Faculty of Systems, Electronics and Industrial Engineering of the Technical University of Ambato, where students traditionally face long waiting times and limited human resources to address their concerns. The proposed system transforms the telephone consultation experience through an architecture that combines an Asterisk central with advanced cognitive services, allowing interactions in natural language. The implementation integrates Microsoft Azure cloud services for semantic processing and speech synthesis, enabling academic consultations without restrictions on schedule or staff availability. The methodology used includes the design of a virtualized environment in the cloud, the configuration of SIP communication protocols, the development of AGI (Asterisk Gateway Interface) scripts in Python, and the integration with large-scale language models for the contextual processing of academic queries. This work explores the possibilities of convergence between telecommunications and artificial intelligence, offering an innovative and scalable solution that can potentially be extended to other institutional areas, establishing new paradigms in the automation of university services through emerging technologies.



**Keywords:** Asterisk, Artificial intelligence, IVR, IP Telephony

# CAPÍTULO I. MARCO TEÓRICO

## 1.1 Tema de investigación

"SISTEMA DE RESPUESTA INTERACTIVA AUTOMÁTICA CON AGENTES DE IA PARA TELÉFONOS IP USANDO ASTERISK"

### 1.1.1 Planteamiento del problema

A nivel global, la transformación digital en las telecomunicaciones ha generado una transición significativa desde los sistemas tradicionales de telefonía hacia tecnologías basadas en IP. Esta evolución ha sido impulsada por la necesidad de unificar servicios de voz, datos y video en una sola red, y por la creciente demanda de automatización en la atención al cliente. Según Grand View Research en 2023, el mercado global de VoIP alcanzó los USD 40 mil millones en 2022 y se proyecta que crezca a una tasa compuesta anual del 10,2% hasta 2030 [1]. Sin embargo, este avance conlleva el reto de lograr una integración eficiente de herramientas como IVR e inteligencia artificial, que no siempre cumplen con las expectativas de los usuarios debido a problemas de interoperabilidad y baja personalización en las respuestas automatizadas.

En el contexto de las empresas a nivel organizacional, especialmente aquellas del sector servicios, la implementación de telefonía IP con IVR e IA ha supuesto una reconfiguración de sus procesos de atención. Si bien muchas empresas han invertido en estas tecnologías buscando reducir costos y optimizar recursos humanos, el 60% ha reportado dificultades en la integración técnica y operativa, según Salesforce en el 2024 [2]. Esto se debe a que los sistemas heredados, la falta de interoperabilidad y la carencia de personal capacitado limitan la eficacia del sistema. De acuerdo con un informe de IBM Watson Assistant en colaboración con Forrester en 2023, los asistentes virtuales bien implementados pueden reducir los tiempos de respuesta hasta en un 99% en ciertos flujos automatizados. Sin embargo, también se identificó que una mala personalización puede disminuir la satisfacción del cliente hasta en un 22% debido a respuestas genéricas o inadecuadas [3].

En la Facultad de Ingeniería en Sistemas, Electrónica e Industrial de la Universidad Técnica de Ambato, los estudiantes enfrentan el reto de optimizar consultas de procesos administrativos y académicos mediante un sistema telefónico IP automatizado. Actualmente, la gestión de consultas frecuentes (ej. trámites de matrícula, requisitos de asignaturas, horarios de laboratorios) depende de canales manuales o sistemas básicos de IVR, lo que genera demoras, saturación de líneas y respuestas genéricas que no resuelven necesidades específicas. Esto refleja los problemas globales de interoperabilidad [4] y baja personalización en respuestas automatizadas [5], agravados por la falta de integración entre herramientas tecnológicas (ej. bases de datos académicas) y sistemas de comunicación IP. La problemática radica en cerrar la brecha entre la infraestructura tecnológica disponible, las demandas de la comunidad universitaria y las competencias técnicas requeridas, como la gestión de APIs o la adaptación de IA a contextos organizacionales específicos, desafíos alineados con las necesidades del sector de servicios ecuatorianos, que tiene el país [6].

La atención telefónica ineficiente en procesos administrativos estudiantiles tiene su origen en diversas causas estructurales y tecnológicas. Una de las principales es la falta de automatización en los sistemas de atención telefónica, lo que obliga a que gran parte de las consultas sean atendidas manualmente, generando cuellos de botella. A esto se suma una infraestructura telefónica IP subutilizada o desactualizada, que impide aprovechar tecnologías modernas como Asterisk o agentes conversacionales basados en inteligencia artificial. Además, la ausencia de un análisis profundo sobre las necesidades reales de los estudiantes contribuye a que las soluciones implementadas no respondan de forma adecuada a sus requerimientos, perpetuando un modelo tradicional de atención poco eficiente. Finalmente, la falta de innovación tecnológica en los procesos internos limita la posibilidad de escalar mejoras en los canales de comunicación institucional.

El aumento de los tiempos de espera y la sobrecarga del personal administrativo, que se ve forzado a atender tareas repetitivas de forma manual, afectando su productividad. Esta situación repercute directamente en una baja satisfacción estudiantil, ya que los usuarios experimentan frustración ante demoras o respuestas inadecuadas. Además, la falta de modernización tecnológica en los procesos administrativos limita la

transformación digital institucional, lo que reduce la competitividad y eficiencia de la facultad. En conjunto, estas consecuencias crean un entorno de atención poco ágil, limitando el acceso oportuno a información clave y afectando la percepción institucional por parte del estudiantado.

El uso de Asterisk combinado con agentes de IA no solo mejora la experiencia del usuario y reduce la carga operativa, sino que también se presenta como una solución viable y económica para organizaciones que buscan modernizar su infraestructura de telecomunicaciones sin depender de proveedores comerciales costosos.

## 1.2 Antecedentes investigativos

La integración de tecnologías avanzadas como los sistemas de Respuesta de Voz Interactiva (IVR) potenciados con inteligencia artificial (IA) ha cobrado relevancia en los últimos años, revolucionando los métodos tradicionales de atención telefónica y comunicación institucional. Diversos estudios han explorado los beneficios de estas tecnologías, destacando su impacto en la automatización de consultas, la optimización de tiempos de respuesta y la mejora en la experiencia del usuario.

En el Ministerio Público de Piura, en el año 2022, Percy Aguirre implementó un sistema de telefonía IP basado en Asterisk para mejorar la eficiencia en la comunicación interna y externa. Este sistema se diseñó utilizando Asterisk sobre un servidor con sistema operativo Linux Debian 11, aprovechando su capacidad para manejar gestión de llamadas, buzón de voz, conferencias y conversión de líneas analógicas a digitales. Además, se instalaron teléfonos IP compatibles con el protocolo SIP y se configuraron terminales en computadoras, tablets y celulares para la gestión y monitoreo de las llamadas. Durante la implementación, se utilizó la metodología PPDIOO de Cisco para planificar, diseñar y optimizar el sistema. Los resultados revelaron que, antes de la implementación, el 90% de los trabajadores expresaron dificultades para comunicarse con otras sedes debido a la falta de integración de los anexos telefónicos. Tras la implementación del sistema, se observó que el 73.33% de los trabajadores manifestó que antes del cambio no podían comunicarse de manera eficiente con los usuarios externos, lo que resalta la necesidad de soluciones

optimizadas basadas en IVR e IA para mejorar la experiencia de usuario y la gestión de llamadas [7].

En 2023, Oscar Condori y Carlos Gonzales, en Perú, desarrollaron un sistema de voz sobre IP basado en el software libre Asterisk versión 13.21.1 en la empresa CSM Corporación Oriente SAC de Iquitos. El hardware utilizado incluyó computadoras con Windows 10 y el software estadístico SPSS versión 26 para el análisis de datos. Se aplicaron cuestionarios de preprueba y posprueba para evaluar la optimización de la comunicación antes y después de la implementación del sistema. Según los resultados obtenidos, el 66.7% de los trabajadores indicaron estar satisfechos con la mejora de la comunicación y el 33.3% valoró positivamente la reducción de tiempo en la recepción de llamadas y mantenimiento de averías. Además, el análisis estadístico utilizando la prueba U de Mann Whitney arrojó un valor de significancia de 0.002, confirmando que la implementación del sistema tuvo un efecto significativo en la optimización de la comunicación [8].

En 2023, Richard Mantilla León y Juan Gamarra Moreno, en Perú, desarrollaron un prototipo de Asistente Personal Virtual basado en Inteligencia Artificial para las oficinas de TI del sector público. El sistema fue programado en Python y ejecutado en una plataforma Microsoft Windows 10, permitiendo la interacción mediante voz y del procesamiento de lenguaje natural (NLP). Sus principales funciones incluyen la creación de proyectos de TI, el registro de consultas y observaciones de proveedores en estudios de mercado, y la generación de registros del comité de selección. Según los resultados obtenidos, el prototipo optimizó el tiempo y los recursos en la gestión de proyectos, reduciendo los errores en el registro de información y mejorando la eficiencia administrativa. Además, estudios previos indican que hasta el 45% de los proyectos de TI exceden su presupuesto y el 56% entregan menos valor del esperado, lo que resalta la importancia de herramientas automatizadas para mejorar la gestión [9].

En 2024, Khushbu Mehboob Shaikh y Georgios Giannakopoulos, en Estados Unidos y Países Bajos, realizaron un estudio sobre la evolución de los sistemas de Respuesta de Voz Interactiva (IVR) y su transición hacia la automatización mediante inteligencia artificial. El estudio analizó el impacto del NLP y el aprendizaje automático en la

optimización de estos sistemas, permitiendo una interacción más eficiente y natural entre usuarios y asistentes virtuales. Se emplearon plataformas de desarrollo de IVR tradicionales y modernas, integrando herramientas de análisis de datos para evaluar su desempeño. Según los resultados obtenidos, la adopción de IVR basado en inteligencia artificial permitió una reducción de costos operativos de hasta un 30%, optimizando la asignación de recursos y disminuyendo la necesidad de intervención humana en los procesos de atención al cliente. Además, se observó una reducción en los tiempos de respuesta y un aumento en la satisfacción de los usuarios, consolidando la efectividad de la automatización en la optimización del servicio [10].

En 2024, Carlos Laínez y Nohely Morales, en Honduras, desarrollaron un proyecto de implementación de procesos de autogestión mediante IVR en una empresa de telecomunicaciones. La investigación se basó en metodologías como Seis Sigma y Lean, además de teorías de Automatización Robótica de Procesos (RPA), con el objetivo de mejorar la eficiencia operativa y optimizar la experiencia del cliente. El sistema automatiza tareas repetitivas como la gestión de reclamos, ajustes de facturación y cambios de plan, reduciendo la necesidad de intervención humana. Según los resultados obtenidos, la implementación de IVR permitió una reducción anual de costos en atención al cliente de aproximadamente 30%, mejorando la rapidez en la resolución de consultas y aumentando la satisfacción del usuario. El análisis costo-beneficio confirmó la viabilidad del sistema, destacando su impacto positivo en la productividad y la optimización de recursos [11].

### 1.3 Fundamentación teórica

El avance de la telefonía IP y los sistemas automatizados de respuesta ha permitido la optimización de los servicios de atención al cliente a través de tecnologías como Inteligencia Artificial (IA) e IVR. La integración de agentes de IA en estos sistemas mejora la eficiencia y reduce los tiempos de espera, ofreciendo respuestas más naturales y precisas a los usuarios. A continuación, se describen los conceptos fundamentales que sustentan esta investigación.

### 1.3.1 Telefonía IP

La telefonía IP (VoIP - Voice over Internet Protocol) representa una evolución fundamental en las comunicaciones modernas, transformando la manera en que transmitimos voz a través de redes digitales. A diferencia de la telefonía tradicional que utiliza circuitos dedicados, VoIP convierte las señales de voz en datos digitales que pueden viajar por las mismas redes que transportan correos electrónicos, archivos y navegación web, optimizando recursos e infraestructura [12].

#### *a. Principios de funcionamiento*

El funcionamiento de la telefonía IP sigue un proceso estructurado en cuatro fases interrelacionadas que garantizan la transmisión efectiva de comunicaciones vocales[13]:

- ***Digitalización y codificación:*** La señal analógica de voz captada por el micrófono se convierte a formato digital mediante muestreo. Posteriormente, algoritmos de codificación (códecs) como G.711, G.729 o Opus procesan estos datos para optimizar el equilibrio entre calidad sonora y consumo de ancho de banda.

- ***Paquetización:*** Los datos digitalizados se fragmentan en unidades más pequeñas (paquetes) y se encapsulan con información crítica en sus cabeceras: identificadores de secuencia, marcas temporales, direcciones IP de origen y destino, y datos para control de calidad.

- ***Transmisión:*** Los paquetes viajan a través de redes IP utilizando protocolos especializados para comunicaciones en tiempo real, pudiendo seguir diferentes rutas hasta alcanzar su destino, según las condiciones de la red.

- ***Reconstrucción:*** En el extremo receptor, los paquetes son reordenados según sus marcadores de secuencia, procesados para compensar pérdidas o retrasos (mediante buffers de jitter), decodificados y finalmente convertidos nuevamente a señal analógica para su reproducción.

### *b. Infraestructura y componentes*

La implementación de sistemas VoIP requiere diversos elementos que trabajan de forma coordinada[14]:

- ***Terminales:*** Dispositivos físicos (teléfonos IP, videoteléfonos) o aplicaciones software (softphones) que capturan, digitalizan, codifican y decodifican las señales de voz.

- ***Controladores de llamadas:*** Servidores que gestionan el establecimiento, mantenimiento y finalización de sesiones de comunicación (ejemplos: Asterisk, FreePBX).

- ***Gateways:*** Dispositivos que actúan como puente entre redes VoIP y la telefonía tradicional (PSTN), permitiendo llamadas entre ambos mundos.

- ***Adaptadores ATA:*** Permiten conectar teléfonos analógicos convencionales a redes IP, extendiendo su vida útil en entornos modernos.

### *c. Protocolos fundamentales*

La interoperabilidad en sistemas VoIP depende de protocolos estandarizados que gestionan diferentes aspectos de la comunicación[13][14][15][16]:

- ***SIP (Session Initiation Protocol):*** Protocolo de señalización para establecer, modificar y terminar sesiones multimedia. Su diseño simple y flexible lo ha convertido en el estándar predominante en implementaciones modernas

- ***RTP (Real-time Transport Protocol):*** Gestiona el transporte de los datos multimedia en tiempo real, incluyendo mecanismos para mitigar efectos de latencia, jitter y pérdida de paquetes.

- ***H.323:*** Conjunto completo de protocolos para videoconferencia y telefonía IP, común en sistemas corporativos más antiguos**.**

- ***IAX (Inter-Asterisk eXchange):*** Desarrollado específicamente para comunicaciones entre servidores Asterisk, minimiza el ancho de banda y simplifica la travesía NAT.

La selección adecuada de estos protocolos impacta directamente en la calidad de servicio (QoS), seguridad y capacidad de integración del sistema, consideraciones esenciales para implementaciones empresariales como la presentada en esta investigación.

La Tabla 1 presenta un análisis detallado de las ventajas operativas, limitaciones técnicas y consideraciones de seguridad que caracterizan a los sistemas de telefonía IP modernos en comparación con sistemas tradicionales.

Tabla 1. Ventajas y desventajas de la implementación de la telefonía IP[17].

| **Ventajas** | **Desventajas** |
|---|---|
| Permite reducir considerablemente los gastos operativos en comparación con la telefonía tradicional. | Depende mucho de la calidad de la conexión a Internet y del ancho de banda disponible. |
| Es fácil de ampliar, ya que no necesita instalar más cables o equipos físicos. | Si la red no está bien optimizada, pueden presentarse problemas como retrasos en la voz (latencia), cortes (jitter) o pérdida de datos. |
| Se puede conectar con otros sistemas como plataformas de atención al cliente (CRM), buzones de voz o incluso herramientas de inteligencia artificial. | Es más vulnerable a ciertos ataques informáticos como bloqueos del servicio, espionaje de llamadas o suplantación de identidad. |
| Ofrece movilidad y flexibilidad, ya que se puede usar desde cualquier lugar que tenga acceso a Internet. | Necesita configuraciones técnicas más avanzadas para que funcione de manera segura y estable. |

### *d. Protocolos*

En los sistemas de telefonía IP, existen varios protocolos que permiten establecer y gestionar las llamadas. Los más comunes son SIP, IAX y PJSIP a continuación, describimos cada uno [18][19][20]:

- **SIP (Session Initiation Protocol)** es el protocolo de señalización más ampliamente utilizado en entornos VoIP. Su función principal es encargarse de iniciar, controlar y finalizar las sesiones de comunicación entre los usuarios.

- **IAX (Inter-Asterisk eXchange)** fue creado específicamente para funcionar con Asterisk. Una de sus principales ventajas es que utiliza un solo puerto (4569 UDP), lo que lo hace especialmente útil en redes que usan NAT, facilitando así la comunicación sin complicaciones adicionales de configuración.

- **PJSIP** es una versión más moderna y flexible de SIP. Está diseñada para ofrecer mejor rendimiento y mayor seguridad. Además, permite manejar múltiples identidades y establecer varias conexiones al mismo tiempo, lo que lo hace una opción muy eficiente para sistemas más complejos.

En la Tabla 2 se presenta una comparación técnica entre los protocolos SIP, IAX y PJSIP, los cuales son utilizados en sistemas de telefonía IP como Asterisk para la gestión de llamadas y sesiones. Esta evaluación es fundamental para seleccionar el protocolo más adecuado según los requerimientos técnicos del sistema propuesto.

Tabla 2. Protocolos[18][19][21][22]

| **Características** | **SIP** | **IAX** | **PJSIP** |
|---|---|---|---|
| Tipo de Protocolo | Señalización estándar VoIP | Protocolo propietario Asterisk | Modular, sucesor de SIP en Asterisk |
| Puerto por defecto | 5060 UDP/TCP | 4569 UDP | 5060 UDP/TCP (como SIP) |
| Uso típico | Telefonía IP tradicional, softphones, gateways | Comunicación entre servidores Asterisk o clientes IAX | Implementaciones modernas en Asterisk |
| Ventajas | Estándar ampliamente adoptado, interoperabilidad | Solo usa un puerto, ideal para NAT | Mejor gestión de NAT, múltiples identidades por endpoint |
| Desventajas | Problemas con NAT, requiere ajustes (STUN, RTP) | Menor interoperabilidad, poco soporte externo | Configuración más compleja, sintaxis extensa |
| Archivo de configuración | sip.conf | iax.conf | pjsip.conf |
| Compatibilidad con NAT | Media | Alta | Alta |
| Facilidad de configuración | Alta (muy documentado) | Media (documentación limitada) | Media-baja (estructura avanzada) |

### *e. Normativas*

Al implementar centros de atención automatizada, como los Call Centers que funcionan con tecnología VoIP, es fundamental cumplir con ciertas normativas que aseguran la calidad del servicio, protegen los datos personales y garantizan el respeto a la legislación vigente. A continuación, se presentan las principales regulaciones que deben considerarse [23][24][25][26][27]:

- ***ISO 9001 – Sistema de Gestión de la Calidad.*** La norma ISO 9001 establece los lineamientos necesarios para crear un sistema de gestión de calidad dentro de una organización. Su enfoque está en la mejora continua y en la satisfacción del cliente. En el caso de los Call Centers, esta norma impulsa la eficiencia de los procesos, la trazabilidad de la atención y la estandarización del servicio.

- ***Norma ISO 9001:2015/Amd 1:2024.*** Esta enmienda actualiza la norma ISO 9001:2015 sobre sistemas de gestión de calidad, incorporando nuevos lineamientos orientados a la mejora continua bajo un enfoque basado en riesgos. Además, reconoce el potencial de herramientas como los chatbots y la inteligencia artificial para automatizar procesos, optimizar recursos y elevar la calidad del servicio en entornos educativos.

- ***Reglamento General de Protección de Datos (GDPR).*** El GDPR, una normativa de la Unión Europea establece reglas claras para el manejo de datos personales. Los sistemas VoIP que almacenan o procesan información de usuarios deben respetar principios como el consentimiento informado, la minimización del uso de datos y el derecho de los usuarios a eliminar su información.

- ***Ley Orgánica de Protección de Datos Personales (LOPDP).*** En Ecuador, esta ley regula el uso de datos personales e impone principios como la licitud, la transparencia, la seguridad y la responsabilidad en el tratamiento de información. Para los Call Centers y sistemas VoIP que gestionan datos de clientes, esta ley es clave para asegurar un uso ético y legal de dicha información.

- ***Reglamento General a la Ley Orgánica de Protección de Datos Personales.*** Este reglamento complementa la LOPDP al detallar cómo deben aplicarse los principios

y obligaciones legales en la práctica. Proporciona procedimientos específicos y medidas técnicas y organizativas que las instituciones deben adoptar para proteger la privacidad de los usuarios.

### 1.3.2 Sistema de Respuesta Interactiva Automática (IVR)

Los Sistemas de Respuesta de Voz Interactiva, conocidos como IVR por sus siglas en inglés, son herramientas tecnológicas que permiten a los usuarios comunicarse con un sistema automatizado usando su voz o las teclas del teléfono. Gracias a esta tecnología, las personas pueden obtener información o acceder a ciertos servicios sin necesidad de hablar con un operador, lo que ayuda a mejorar la eficiencia en la atención al cliente y a disminuir los tiempos de espera[28].

Estos sistemas combinan tecnologías de telefonía, reconocimiento de voz y procesamiento de datos para ofrecer una experiencia guiada al usuario mediante menús de navegación predefinidos. La implementación adecuada de un IVR no solo optimiza recursos empresariales, sino que también puede incrementar significativamente la satisfacción del cliente cuando se diseña considerando la experiencia del usuario[28].

### 1.3.3 Arquitectura del sistema

Un sistema IVR está formado por varios elementos que trabajan juntos para automatizar la atención telefónica. Entre estos se incluyen los módulos que reciben y emiten la voz, motores de reconocimiento de voz (ASR), sistemas que convierten texto en voz (TTS) y los controladores del flujo de llamadas. Estas plataformas pueden integrarse fácilmente con soluciones de código abierto como Asterisk, lo que permite crear menús interactivos personalizados y automatizar diferentes procesos dentro de una organización[29].

### 1.3.4 Interacción con los usuarios

Los IVR responden a los usuarios a través de mensajes pregrabados o voces generadas por computadora. La interacción puede hacerse marcando opciones con el teclado del teléfono (a través de tonos DTMF) o, en versiones más avanzadas, usando la voz. Esto

permite acceder rápidamente a servicios como consultar saldos, agendar citas o resolver requerimientos, sin esperar a un agente humano[30].

### 1.3.5 Implementación dinámica

Hoy en día, muchos sistemas IVR ofrecen herramientas visuales e interfaces amigables que permiten diseñar o ajustar los flujos de interacción sin necesidad de tener conocimientos profundos en programación. Esto los hace mucho más accesibles y adaptables a las necesidades específicas de distintas instituciones[31].

Estas plataformas de desarrollo visual funcionan mediante representaciones gráficas de los posibles caminos de navegación, donde cada nodo representa una interacción con el usuario. Los administradores pueden arrastrar y soltar elementos como mensajes de audio, puntos de decisión o solicitudes de información, conectándose mediante líneas que representan las posibles respuestas del usuario. Esta aproximación modular permite visualizar el flujo completo de la llamada y detectar posibles puntos de confusión o abandono antes de su implementación[31].

La ventaja principal de esta implementación dinámica es la capacidad de realizar modificaciones en tiempo real, permitiendo ajustes rápidos ante cambios en los servicios ofrecidos o en respuesta a métricas de desempeño. Por ejemplo, si se identifica que un determinado menú genera confusión entre los usuarios, el personal administrativo puede reordenar las opciones o reformular los mensajes sin interrumpir el servicio ni requerir asistencia técnica especializada[31].

### 1.3.6 Centros de atención telefónica

Estos sistemas son utilizados en diversos ámbitos. En el sector salud, por ejemplo, permiten agendar citas médicas; en la banca, realizar consultas y operaciones sin intervención humana; en instituciones públicas, facilitar trámites ciudadanos; y en educación, ofrecer información sobre clases o procesos académicos[32].

La arquitectura tecnológica de los centros de atención telefónica modernos ha evolucionado significativamente con la integración de plataformas PBX basadas en la nube como Asterisk. Estos sistemas incorporan componentes como distribuidores

automáticos de llamadas (ACD), sistemas de respuesta interactiva (IVR), grabación de llamadas y herramientas analíticas que permiten optimizar la experiencia del usuario. Particularmente relevante es la reciente incorporación de tecnologías de procesamiento de lenguaje natural e inteligencia artificial, que permiten implementaciones de IVR conversacionales capaces de entender consultas complejas formuladas en lenguaje natural y ofrecer respuestas contextualizadas, reduciendo la necesidad de transferencia a agentes humanos y mejorando los tiempos de resolución. Estas capacidades se potencian especialmente cuando la infraestructura se implementa en proveedores cloud como Microsoft Azure, que ofrecen servicios cognitivos integrados y alta disponibilidad para gestionar volúmenes variables de llamadas[33].

### 1.3.7 Sistema PBX

Un PBX, o central telefónica privada, es un sistema que se usa dentro de una organización para administrar tanto las llamadas internas como las externas. En sus inicios, estos sistemas funcionaban con líneas telefónicas analógicas, pero con el paso del tiempo han evolucionado hacia soluciones digitales y basadas en IP. Esta transición ha permitido incorporar funciones más avanzadas y, al mismo tiempo, reducir los costos operativos[29].

La evolución más reciente de estos sistemas ha sido la virtualización completa mediante software, denominada IP-PBX o PBX virtual. Plataformas como Asterisk representan esta nueva generación de centrales telefónicas que ejecutan todas sus funciones mediante software, eliminando casi por completo la dependencia de hardware especializado. Esta arquitectura basada en software facilita su implementación en entornos virtualizados y servicios de nube como Microsoft Azure, permitiendo una separación clara entre la lógica de procesamiento de llamadas y la infraestructura física subyacente[29].

Los sistemas PBX ofrecen varias funciones clave dentro de una organización, como permitir llamadas internas entre extensiones sin usar la red telefónica pública, gestionar el enrutamiento de llamadas entrantes y salientes, y brindar servicios como buzón de voz, transferencia de llamadas, espera, conferencias y grabación. Además,

pueden integrarse con sistemas IVR y asistentes virtuales, lo que amplía sus capacidades de automatización y atención inteligente[29].

La comparación técnica presentada en la Tabla 3, relacionada con diversas plataformas PBX, resulta clave para seleccionar la opción más adecuada en el desarrollo del sistema IVR inteligente propuesto en este trabajo.

Tabla 3. Comparativa de las diferentes PBX[34],[35],[36],[37],[38].

| **Característica** | **Asterisk** | **3CX** | **FreePBX** | **Issabel** | **FusionPBX** |
|---|---|---|---|---|---|
| **Tipo de Licencia** | Código abierto (GPL) | Propietaria | Código abierto (GPL) | Código abierto | Código abierto (MPL) |
| **Costo** | Gratuito | Gratis (limitada) o de pago | Gratuito (addons de pago) | Gratuito | Gratuito |
| **Motor de telefonía** | Propio | Propio | Basado en Asterisk | Basado en Asterisk | Basado en FreeSWITCH |
| **Compatibilidad con protocolos** | SIP, IAX, PJSIP, H.323 | SIP | SIP, IAX | SIP, IAX | SIP |
| **Requisitos de implementación** | Altos (Linux/VoIP) | Bajos-medios (GUI) | Medios (GUI+CLI) | Medios | Altos (avanzados) |
| **Escalabilidad** | Alta | Media | Alta | Media-alta | Muy alta |
| **Personalización** | Alta (scripts, dialplans, AGI) | Media (GUI, plugins) | Media-alta (GUI+consola) | Media | Alta (muy configurable) |
| **Soporte para APIs externas** | Completo (REST, AMI, AGI) | Limitado a versiones comerciales | Amplio (REST, AMI, módulos) | Disponible (REST, SOAP) | Extenso (XML, REST, WebRTC) |
| **Integración con bases de datos** | MySQL, PostgreSQL, SQLite | Limitada | MySQL, otras vía Asterisk | MySQL, MariaDB | PostgreSQL principalmente |
| **Compatibilidad con IA** | Alta (REST, AMI, IA externa) | Limitada | Moderada | Moderada | Alta |
| **Plataformas compatibles** | Linux (nativo) | Windows, Linux, nube | Linux | Linux | Linux, BSD |
| **Documentación y comunidad** | Amplia comunidad técnica | Buena documentación comercial | Comunidad activa y empresarial | Comunidad en LATAM | Menor pero técnica |
| **Casos de uso frecuentes** | Call centers personalizados, IVR avanzados | PYMEs, oficinas | PYMEs, IVR básicos | Centros de contacto básicos | Grandes empresas, operadoras |

### 1.3.8 Arquitectura de Asterisk y sus componentes principales

Asterisk se estructura mediante una arquitectura modular que facilita tanto su implementación como su extensibilidad[39]. Este diseño permite que el sistema funcione eficientemente como PBX completa, enrutando llamadas entre interfaces de telefonía tradicionales y protocolos modernos de VoIP. A continuación, se analizan sus componentes esenciales:

***a. Núcleo del sistema***

El núcleo de Asterisk gestiona las conexiones internas entre los distintos módulos, controlando el establecimiento, mantenimiento y terminación de llamadas. Esta capa de abstracción permite que diferentes tecnologías de telefonía interactúen entre sí sin incompatibilidades, convirtiendo protocolos y códecs según sea necesario[40].

***b. Módulos de canal***

Estos componentes manejan los distintos tipos de conexiones que soporta Asterisk. Entre los más relevantes se encuentran[41]:

- ***chan_sip y chan_pjsip:*** *Gestionan conexiones mediante el protocolo SIP, siendo PJSIP la implementación más moderna y recomendada para nuevos despliegues.*
- ***chan_iax2:*** *Implementa el protocolo Inter-Asterisk eXchange, diseñado específicamente para comunicaciones entre servidores Asterisk.*
- ***chan_dahdi:*** *Permite la integración con hardware de telefonía tradicional para conectar líneas analógicas o digitales.*

***c. Plan de marcado (Dialplan)***

El plan de marcado constituye la lógica central del sistema, definiendo cómo se procesan las llamadas. Se estructura en contextos, extensiones y prioridades[42]:

- ***Contextos:*** *Agrupaciones lógicas de extensiones que permiten aislar diferentes entornos de llamada.*

- ***Extensiones:*** *Patrones que coinciden con los números marcados.*

- ***Prioridades:*** *Secuencia ordenada de acciones a ejecutar cuando una llamada coincide con una extensión*

Esta configuración se realiza principalmente en el archivo extensions.conf y permite implementar desde sencillas redirecciones de llamadas hasta complejos árboles de decisión con lógica condicional.

***d. Interfaces de aplicación***

Asterisk dispone de interfaces que permiten la integración con sistemas externos[43]:

- ***AGI (Asterisk Gateway Interface):*** *Permite que programas externos controlen el plan de marcado mediante scripts.*

- ***AMI (Asterisk Manager Interface):*** *Posibilita la supervisión y control del servidor Asterisk en tiempo real.*

- ***ARI (Asterisk REST Interface):*** *Ofrece una API REST para crear aplicaciones de telefonía personalizadas.*

La combinación de estos componentes arquitectónicos permite que Asterisk funcione como una plataforma altamente personalizable, capaz de adaptarse a diversos escenarios de comunicación empresarial, desde pequeñas implementaciones hasta complejos centros de contacto con integraciones avanzadas.

### 1.3.9 Proveedores de servicios en la nube

Los proveedores de servicios en la nube ofrecen entornos ideales para la implementación de centrales telefónicas basadas en Asterisk, permitiendo configuraciones flexibles para comunicaciones entre extensiones y ejecución de scripts personalizados mediante AGI (Asterisk Gateway Interface). Esta infraestructura virtualizada elimina la necesidad de hardware dedicado, reduciendo costos y simplificando el mantenimiento de sistemas PBX empresariales[44].

### *a. Comparativa de servicios en la nube para Asterisk*

Al implementar Asterisk en entornos cloud, es importante considerar las características específicas de cada proveedor[44],[45],[46],[47]:

- ***IaaS (Infraestructura como Servicio)****: Proporciona control total sobre la configuración del sistema operativo y Asterisk, ideal para implementaciones personalizadas con scripts AGI complejos. Ejemplo: Máquinas virtuales de Azure.*
- ***PaaS (Plataforma como Servicio):*** *Ofrece entornos preconfigurados donde Asterisk puede desplegarse con menos configuración manual. Ejemplo: Azure App Service con contenedores.*
- ***SaaS (Software como Servicio):*** *Soluciones PBX completamente gestionadas basadas en Asterisk, con menor flexibilidad, pero mayor facilidad de uso. Ejemplos: Servicios de telefonía empresarial basados en la nube.*

En la Tabla 4, se comparan los principales proveedores de servicios en la nube para implementaciones de Asterisk, destacando especialmente Microsoft Azure como la plataforma utilizada en este proyecto de tesis. La selección de estos proveedores considera factores críticos para el funcionamiento óptimo de Asterisk, como la estabilidad de red, capacidad de procesamiento en tiempo real, y soporte para scripts AGI personalizados.

Tabla 4. Servicios de máquinas en la nube presentes para Asterisk[48],[49],[50],[51],[52],[53].

| Proveedor | Servicios Destacados | Ventajas para Asterisk |
|---|---|---|
| **Microsoft Azure** | Máquinas virtuales, Azure AI, App Services | Brinda procesamiento eficiente de audio y una conectividad SIP estable. La autenticación mediante AD favorece implementaciones seguras de IVR. |
| **Amazon Web Services (AWS)** | EC2, S3, Lambda, Lightsail | Ofrece una arquitectura versátil y rentable para sistemas VoIP, con amplia disponibilidad de instancias optimizadas para tráfico de voz. |
| **Google Cloud Platform (GCP)** | Compute Engine, Dialogflow, Firebase | Su entorno facilita la incorporación de agentes de voz inteligentes, gracias a APIs nativas de reconocimiento de lenguaje natural. |
| **DigitalOcean** | Droplets (VM), App Platform | Ideal para proyectos pequeños o pruebas de concepto, debido a su rápida configuración y costos controlados. |

| | | |
|---|---|---|
| **Oracle Cloud Infrastructure (OCI)** | Compute, Autonomous DB | Soporta un tráfico simultáneo de llamadas de forma eficiente, lo que lo convierte en una opción sólida para aplicaciones empresariales. |
| **Vultr / Linode** | VPS económicos, soporte para Docker | Alternativas asequibles para servicios SIP básicos, permitiendo a usuarios implementar sistemas VoIP funcionales con recursos limitados. |

### *b. Selección de Azure para la compilación de Asterisk*

Para la implementación de una central telefónica basada en Asterisk, se seleccionó Microsoft Azure como proveedor de infraestructura en la nube debido a varias ventajas técnicas específicas. Azure ofrece una combinación óptima de rendimiento, fiabilidad y herramientas de gestión que resultan particularmente beneficiosas para sistemas de comunicación en tiempo real como Asterisk[54].

Entre los factores determinantes para esta elección se encuentran la estabilidad de su infraestructura de red global, que proporciona conexiones con latencia predecible y constante, aspecto crítico para mantener la calidad de voz en comunicaciones SIP[29]. Las máquinas virtuales de Azure ofrecen rendimiento consistente con opciones de procesamiento optimizadas para cargas de trabajo en tiempo real, garantizando que incluso en momentos de alta demanda, las llamadas conserven su integridad[54].

Adicionalmente, Azure facilita la integración con otros servicios de Microsoft ampliamente utilizados en entornos empresariales, como Active Directory para autenticación centralizada y PowerShell para automatización de tareas de mantenimiento. Su plataforma de monitoreo Azure Monitor permite supervisar el rendimiento del sistema PBX con métricas detalladas, facilitando la detección temprana de posibles problemas[33], [32].

## 1.3.10 Ventajas de Azure para implementaciones de Asterisk

Es importante conocer cada una de las ventajas que se presentan a continuación[55], [56], [57], [58], [59], [60]:

- ***Escalabilidad optimizada:*** *Capacidad para ajustar recursos según el volumen de llamadas, ideal para manejar picos de demanda en sistemas PBX.*

- ***Confiabilidad de red:*** *La infraestructura global de Azure proporciona conexiones de baja latencia, fundamentales para mantener la calidad de audio en llamadas VoIP.*

- ***Seguridad avanzada:*** *Opciones robustas de aislamiento de red que protegen el tráfico SIP de posibles vulnerabilidades.*

- ***Automatización:*** *Facilidad para implementar scripts que automatizan copias de seguridad de configuraciones de Asterisk y grabaciones de llamadas.*

- ***Flexibilidad en configuración:*** *Posibilidad de personalizar completamente las configuraciones de extensiones y los flujos de llamadas mediante scripts AGI según las necesidades organizacionales.*

### 1.3.11 Modelos de lenguaje para interfaces conversacionales

Los modelos de lenguaje de gran escala (LLM) han revolucionado las capacidades de los sistemas de respuesta de voz interactiva, permitiendo interacciones más naturales y contextuales con los usuarios. Estos modelos comprenden el lenguaje humano, mantienen el contexto de conversación y generan respuestas coherentes, lo que los hace ideales para implementaciones en sistemas telefónicos avanzados basados en Asterisk[61].

A diferencia de los sistemas IVR tradicionales que operan con árboles de decisión predefinidos y reconocimiento limitado de palabras clave, los sistemas basados en LLM pueden entender consultas complejas expresadas en lenguaje natural, inferir la intención del usuario incluso cuando la solicitud es ambigua, y ofrecer respuestas personalizadas en función del contexto de la conversación y la información disponible[32].

La implementación de estos modelos en entornos telefónicos requiere considerar factores técnicos específicos como la latencia (crítica para mantener una conversación fluida), la gestión del contexto (para mantener coherencia a lo largo de la llamada), y la capacidad para integrarse con sistemas existentes mediante interfaces como AGI (Asterisk Gateway Interface)[62].

A continuación, se observan los diferentes modelos de IA en la Tabla 5. Modelos de lenguaje AI para integración con Asterisk mediante AGI, en los cuales se comparan características fundamentales para su implementación en sistemas Asterisk. Esta evaluación comparativa resulta esencial para seleccionar el modelo más adecuado según los requisitos específicos del proyecto, considerando factores como la complejidad de las consultas a manejar, el volumen de llamadas previsto y las limitaciones presupuestarias.

Tabla 5. Modelos de lenguaje AI para integración con Asterisk mediante AGI[63],[64].

| **Modelo** | **Proveedor** | **Características clave** | **Complejidad de integración** | **Latencia** | **Contexto máximo** | **Costo estimado mensual** |
|---|---|---|---|---|---|---|
| **GPT-3.5 Turbo** | OpenAI | Buen balance rendimiento/costo, amplio conocimiento general | Media (API REST bien documentada) | 500-1500ms | 16K tokens | $0.002/1K tokens entrada, $0.002/1K tokens salida |
| **GPT-4 Turbo** | OpenAI | Comprensión avanzada, seguimiento preciso de instrucciones | Media (API REST bien documentada) | 1000-3000ms | 128K tokens | $0.01/1K tokens entrada, $0.03/1K tokens salida |
| **Azure OpenAI GPT-4** | Microsoft | Similar a GPT-4 con mayor cumplimiento normativo y SLA empresarial | Media-baja (integración directa con Azure) | 1000-2500ms | 128K tokens | Según plan Azure + uso API |
| **Claude 3 Sonnet** | Anthropic | Excelente en conversaciones largas, buena comprensión contextual | Media (API REST) | 800-2000ms | 200K tokens | $3/millón tokens entrada, $15/millón tokens salida |
| **Gemini Pro** | Google | Buen rendimiento multimodal, integración con servicios Google | Media (GCP/Vertex AI) | 800-2000ms | 32K tokens | $0.003/1K tokens entrada, $0.004/1K tokens salida |
| **Llama 2 (70B)** | Meta (open source) | Modelo abierto, despliegue local posible | Alta (requiere infraestructura propia) | Variable (hardware) | 4K tokens | Costos de infraestructura |
| **Mixtral 8x7B** | Mistral AI | Buen rendimiento en modelo abierto, eficiente | Alta (despliegue local) | Variable (hardware) | 32K tokens | Costos de infraestructura |

| | | | | | | |
|---|---|---|---|---|---|---|
| **Whisper** | OpenAI | Especializado en transcripción de voz a texto (STT) | Baja (API simple) | 500-1500ms | N/A (audio) | $0.006/minuto |

Para implementaciones en Asterisk, es fundamental considerar no solo las capacidades lingüísticas del modelo, sino también aspectos prácticos como el tiempo de respuesta, costos operativos y la complejidad de integración mediante scripts AGI. En el contexto de este proyecto, se ha seleccionó el modelo GPT-4 de OpenAI debido a su excelente comprensión contextual, capacidad para seguir instrucciones complejas y disponibilidad de integración directa tanto a través de API estándar como mediante servicios de Azure.

La selección del modelo adecuado se alinea con los requisitos específicos del sistema, considerando factores como el volumen actual de llamadas, la complejidad de las consultas que maneja y las restricciones presupuestarias del proyecto. Para implementaciones a pequeña escala o educativas, modelos como GPT-3.5 Turbo ofrecen un buen equilibrio entre capacidades y costo, mientras que, para sistemas empresariales críticos, las versiones desplegadas en Azure proporcionan garantías adicionales de disponibilidad y cumplimiento normativo.

### 1.3.12 Integración de modelos GPT en sistemas Asterisk mediante AGI

La integración de modelos de lenguaje avanzados como ChatGPT-4 en sistemas PBX basados en Asterisk representa una evolución significativa en la automatización de interacciones telefónicas. Esta implementación se realiza principalmente mediante la interfaz AGI (Asterisk Gateway Interface), que permite a Asterisk ejecutar scripts externos durante una llamada, facilitando la comunicación bidireccional entre el sistema telefónico y servicios de IA.

La arquitectura de integración se estructura en varios componentes interconectados:

- ***Capa de telefonía:*** Gestionada por Asterisk, maneja el establecimiento de llamadas, la grabación de audio y la reproducción de respuestas.
- ***Capa de procesamiento de audio:*** Convierte las señales de voz en texto mediante servicios STT (Speech-to-Text).

- ***Capa de procesamiento cognitivo:*** Utiliza la API de ChatGPT-4 para analizar el texto, comprender la intención del usuario y generar respuestas contextuales.

- ***Capa de síntesis:*** Transforma las respuestas textuales en audio mediante servicios TTS (Text-to-Speech).

- ***Capa de orquestación:*** Scripts AGI desarrollados en lenguajes como Python o PHP que coordinan el flujo de datos entre las diferentes capas.

### *a. Implementación técnica mediante scripts AGI*

Los scripts AGI funcionan como middleware entre Asterisk y la API de ChatGPT-4, siguiendo generalmente este flujo de trabajo:

- ***Recepción de audio:*** El script AGI recibe la señal de voz del usuario desde Asterisk.

- **Transcripción:** El audio se convierte en texto mediante servicios de reconocimiento de voz.

- ***Procesamiento con ChatGPT-4:*** El texto es enviado a la API de OpenAI con un prompt de sistema que define el comportamiento del asistente.

- ***Gestión de contexto:*** El script mantiene el estado de la conversación, almacenando mensajes previos para proporcionar contexto.

- ***Generación de respuesta:*** La respuesta generada por ChatGPT-4 se procesa para adaptarla al formato esperado por el usuario.

- ***Síntesis de voz:*** El texto de respuesta se convierte en audio mediante un servicio TTS.

- **Reproducción:** *El audio generado se envía de vuelta a Asterisk para su reproducción al usuario.*

### 1.3.13 Consideraciones técnicas para implementación de Modelos de Lenguaje de Gran Escala en Asterisk

La integración efectiva de modelos de lenguaje avanzados como ChatGPT-4 en sistemas Asterisk mediante scripts AGI requiere un análisis detallado de múltiples factores técnicos que determinan el éxito de la implementación. Estas consideraciones se abordaron de manera sistemática durante las fases de diseño, desarrollo y despliegue para garantizar un sistema confiable y eficiente.

En primer lugar, la latencia representa uno de los desafíos más críticos en sistemas conversacionales telefónicos. A diferencia de interfaces de chat donde ciertos retrasos son tolerables, en comunicaciones de voz los usuarios esperan respuestas inmediatas. Para optimizar los tiempos de respuesta, es recomendable implementar estrategias como el procesamiento asíncrono, la preselección de prompts optimizados que generen respuestas concisas, y la utilización de técnicas de compresión de contexto. Adicionalmente, el despliegue en regiones de Azure geográficamente cercanas a los usuarios finales puede reducir significativamente los tiempos de transmisión de datos.

La gestión robusta de errores constituye otro aspecto fundamental, especialmente considerando la naturaleza distribuida de estos sistemas. Deben implementarse mecanismos de fallback que permitan la degradación gradual de funcionalidades ante fallos en la API de OpenAI o problemas de conectividad. Esto incluye el desarrollo de rutas alternativas de procesamiento, como el redireccionamiento a IVR tradicionales o agentes humanos cuando sea necesario. Las estrategias de retry con backoff exponencial han demostrado ser particularmente efectivas para manejar interrupciones temporales en servicios de IA sin afectar la experiencia del usuario[65].

El control eficiente de costos representa una consideración operativa esencial. Los modelos avanzados como GPT-4 operan bajo un modelo de precios basado en tokens procesados, lo que puede generar gastos significativos en sistemas con alto volumen de llamadas. Se recomienda implementar sistemas de monitoreo en tiempo real del consumo de tokens, establecer límites presupuestarios por llamada, y aplicar técnicas de optimización como la truncación inteligente de contexto, resúmenes periódicos de conversación y caching de respuestas para consultas frecuentes[65].

En cuanto a la seguridad, la implementación debe abordar tanto la protección de datos sensibles como el cumplimiento normativo. Esto incluye el cifrado de comunicaciones entre Asterisk y la API de OpenAI, la implementación de políticas de filtrado para evitar la transmisión inadvertida de información personal identificable (PII), y el desarrollo de mecanismos de auditoría que registren todas las interacciones para fines de cumplimiento regulatorio. Particularmente en sectores como salud o finanzas, es fundamental configurar adecuadamente los prompts de sistema para evitar que el modelo solicite o revele información protegida.

La implementación de estos modelos conversacionales en entornos cloud como Microsoft Azure representa un avance paradigmático respecto a los sistemas IVR tradicionales, posibilitando interacciones significativamente más naturales, adaptativas y útiles. La infraestructura escalable, los servicios cognitivos complementarios y las herramientas de desarrollo integradas que ofrece Azure proporcionan el entorno ideal para desplegar estas soluciones, permitiendo a las organizaciones transformar radicalmente su enfoque de atención telefónica automatizada, haciéndola verdaderamente conversacional y centrada en las necesidades específicas de cada usuario.

### 1.3.14 Sistema Operativo para implementaciones de Asterisk

Un sistema operativo (SO) constituye la capa fundamental de software que gestiona los recursos de hardware y proporciona servicios esenciales para las aplicaciones. En el contexto de implementaciones de centrales telefónicas basadas en Asterisk, la elección del sistema operativo representa una decisión crítica que impacta directamente en la estabilidad, rendimiento y seguridad de la plataforma de comunicaciones.

Los sistemas PBX como Asterisk requieren características específicas del sistema operativo, particularmente en lo relacionado con la gestión de procesos en tiempo real, manejo eficiente de conexiones de red, y capacidad para mantener servicios críticos con alta disponibilidad. Las distribuciones basadas en Linux son ampliamente preferidas para estos entornos debido a su arquitectura modular, bajo consumo de

recursos, estabilidad en operación continua y capacidad para priorizar tráfico de red sensible a la latencia.

En entornos de nube como Microsoft Azure, la selección del sistema operativo para una implementación de Asterisk debe considerar factores adicionales como la compatibilidad con el hipervisor subyacente, las herramientas de monitorización disponibles y las características de seguridad específicas para entornos virtualizados. Las distribuciones empresariales de Linux ofrecen ventajas significativas en estos escenarios, incluyendo ciclos de soporte extendido, actualizaciones de seguridad regulares y optimizaciones específicas para cargas de trabajo virtualizadas.

La Tabla 6 presenta un análisis comparativo de los sistemas operativos más utilizados para implementaciones de Asterisk, con especial énfasis en su idoneidad para despliegues en entornos de nube como Microsoft Azure.

Tabla 6. Sistemas operativos para implementaciones de Asterisk[66][67][68][69]

| **Sistema Operativo** | **Tipo de Licencia** | **Compatibilidad con Asterisk** | **Rendimiento en Azure** | **Ventajas** | **Desventajas** |
|---|---|---|---|---|---|
| AlmaLinux | Libre / Open Source | Excelente | Muy bueno | Estabilidad superior, soporte extendido (hasta 2029), optimizaciones para servicios en tiempo real, compatible con repositorios EPEL | Documentación específica para Asterisk menos abundante que otras distribuciones |
| Rocky Linux | Libre / Open Source | Excelente | Muy bueno | Estable, compatible binariamente con RHEL, comunidad creciente | Relativamente reciente en el ecosistema |
| Ubuntu Server LTS | Libre / Open Source | Muy buena | Bueno | Amplia documentación, excelente soporte en Azure, actualizaciones frecuentes | Algunas optimizaciones adicionales requeridas para rendimiento óptimo en tiempo real |
| Debian | Libre / Open Source | Muy buena | Bueno | Extremadamente estable, eficiente en recursos, ideal para sistemas de producción | Ciclos de actualización más lentos, paquetes menos recientes |

| | | | | | |
|---|---|---|---|---|---|
| CentOS Stream | Libre / Rolling Release | Buena | Moderado | Acceso temprano a características futuras de RHEL | Mayor volatilidad, menos adecuado para sistemas de producción críticos |
| Windows Server | Comercial | Limitada | Bueno | Integración superior con servicios Azure AD | Rendimiento subóptimo para Asterisk, compatibilidad limitada, mayor consumo de recursos |

Para este proyecto, se ha seleccionado AlmaLinux como sistema operativo base debido a su estabilidad superior, compatibilidad completa con Asterisk, ciclo de soporte extendido y optimizaciones específicas para cargas de trabajo de telecomunicaciones.

Esta distribución, derivada directamente de Red Hat Enterprise Linux (RHEL), garantiza un entorno de ejecución confiable con el respaldo de actualizaciones de seguridad por un período prolongado, factor crítico para sistemas de comunicación empresarial.

AlmaLinux incorpora características técnicas particularmente beneficiosas para implementaciones de Asterisk, como el kernel optimizado para baja latencia, soporte nativo para sistemas de archivos de alto rendimiento y configuraciones predeterminadas que priorizan la estabilidad del sistema sobre la incorporación de características experimentales. Estas propiedades resultan fundamentales para garantizar la calidad de servicio (QoS) en comunicaciones de voz en tiempo real, minimizando problemas como jitter, pérdida de paquetes o interrupciones en el servicio.

### 1.3.15 PJSIP: la evolución del protocolo SIP en Asterisk

El módulo PJSIP (chan_pjsip) representa una evolución significativa en la implementación del protocolo SIP dentro de Asterisk, ofreciendo mejoras sustanciales respecto al módulo tradicional chan_sip. Basado en la biblioteca PJSIP, este componente moderniza la gestión de comunicaciones SIP mediante una arquitectura modular que separa claramente las diferentes funcionalidades del protocolo.

A diferencia del módulo SIP tradicional, PJSIP implementa un diseño orientado a objetos donde cada aspecto de la comunicación SIP se maneja mediante componentes independientes e interconectados. Esta arquitectura modular divide la configuración en secciones específicas como endpoints (terminales), auths (autenticación), transports (transporte) y aors (address of record), proporcionando mayor claridad y control sobre cada elemento del sistema.

Entre las ventajas técnicas más destacables de PJSIP frente a implementaciones anteriores se encuentran:

- ***Rendimiento superior:*** Su diseño multihilo permite manejar un mayor número de llamadas simultáneas con menor consumo de recursos, aspecto crítico en entornos virtualizados como Microsoft Azure.

- ***Seguridad reforzada:*** Incorpora soporte nativo para TLS y SRTP, facilitando el cifrado de la señalización y el tráfico multimedia sin configuraciones adicionales complejas.

- ***Conformidad con estándares:*** Implementa rigurosamente las especificaciones RFC del protocolo SIP, mejorando la compatibilidad con diversos dispositivos y servicios externos.

- ***Resolución de NAT avanzada:*** Gestiona de manera más eficiente los escenarios de NAT (Network Address Translation) comunes en implementaciones de nube, reduciendo problemas de conectividad y audio unidireccional.

- ***Flexibilidad en configuración:*** Permite definir plantillas reutilizables que simplifican la administración de múltiples extensiones con características similares.

La adopción de PJSIP resulta particularmente beneficiosa en despliegues cloud como el presentado en este proyecto, ya que su arquitectura optimizada requiere menos recursos de sistema para mantener el mismo volumen de llamadas, permitiendo aprovechar mejor las capacidades de las máquinas virtuales en Azure. Adicionalmente, su diseño modular facilita la depuración de problemas y el ajuste fino de parámetros específicos sin afectar al funcionamiento global del sistema.

Para implementaciones modernas de Asterisk, PJSIP representa la opción recomendada frente a chan_sip, no solo por sus mejoras técnicas, sino también porque este último se considera obsoleto y será eventualmente eliminado de futuras versiones del software. Esta transición asegura la viabilidad a largo plazo de los sistemas de telefonía IP desplegados en la actualidad.

## 1.4 Objetivos

### 1.4.1 Objetivo general

Desarrollar un sistema de respuesta interactiva automática con el uso de agentes de IA para la red telefónica usando Asterisk.

### 1.4.2 Objetivos específicos

- Analizar el estado actual y necesidades de información en trámites administrativos para los estudiantes de la Facultad de Ingeniería en Sistemas Electrónica e Industrial.
- Diseñar una plataforma IVR basada en Asterisk y agentes de inteligencia artificial, para una red telefónica, que permita resolver dudas comunes de los estudiantes.
- Evaluar el sistema de respuesta interactiva automática mediante pruebas piloto en escenarios reales.

# CAPÍTULO II. METODOLOGÍA

## 2.1 Materiales

En el diseño del sistema IVR, el hardware seleccionado cumple un rol esencial al garantizar la conectividad, la transmisión eficiente de voz y la compatibilidad con la infraestructura de telefonía IP. Cada componente ha sido elegido para asegurar un funcionamiento estable y una integración fluida con los demás elementos del sistema. La selección de los componentes físicos, detallada en la Tabla 7, permitió construir una base sólida para la implementación de un sistema fiable, capaz de soportar múltiples llamadas simultáneas y mantener altos estándares de calidad de servicio.

Tabla 7. Materiales y dispositivos.

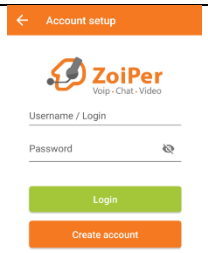
| Materiales | Descripción | Imagen |
|---|---|---|
| Teléfono IP Cisco Phone 303 | Dispositivo físico para realizar y recibir llamadas sobre red IP. |  |
| Gateway GoIP | Equipo que permite conectar líneas móviles (SIM) a una red VoIP. |  |
| Router TP-LINK | Router que proporciona conectividad a internet y enruta paquetes VoIP. |  |
| Módulo SIM | SIM de telefonía móvil para conexión GSM en el gateway GoIP. |  |
| Path cord Ethernet | Es un cable que se usa para conectar dispositivos como computadoras, teléfonos IP o switches dentro de una red local. | |

| | | |
|---|---|---|
| | |  |
| Fuente de 12 VDC 1A | Dispositivo que convierte la corriente alterna (AC) de un enchufe en corriente continua (DC) de 12 voltios, con una capacidad máxima de 1 amperio. Es ideal para alimentar equipos electrónicos pequeños como routers, tiras LED, sensores o módulos de desarrollo. |  |

Por su parte, la selección del software se enfocó en optimizar la gestión de llamadas, el control del flujo de diálogo automatizado y la integración con servicios de inteligencia artificial. Se eligió Asterisk como núcleo del sistema PBX, destacando por ser una solución de código abierto y por su amplia compatibilidad con diversos protocolos de señalización. La combinación de herramientas seleccionadas, descrita en la Tabla 8, permitió establecer una arquitectura modular, flexible y de fácil mantenimiento, ideal para su implementación en entornos académicos.

Tabla 8. Software y servicios de aplicación.

| Materiales | Descripción | Imagen |
|---|---|---|
| **Asterisk 18.12.1** | Software PBX de código abierto para gestionar llamadas VoIP e IVR. | Asterisk |
| **Python 3.10** | Lenguaje de programación usado para controlar los flujos AGI. | python |
| **AGI Script (tesis.py)** | Script en Python que permite interacción entre Asterisk y el modelo IA. | |

| | | |
|---|---|---|
| **AlmaLinux 9.3** | Sistema operativo base para la máquina virtual del servidor Asterisk. |  |
| **Azure Cloud** | Plataforma en la nube donde se implementó el sistema VoIP. |  |
| **PJSIP** | Protocolo de señalización utilizado para gestionar llamadas SIP. | |
| **ChatGPT API** | Agente de IA integrado para responder preguntas mediante lenguaje natural. |  |
| **Visual Studio Code** | Editor de código multiplataforma utilizado para desarrollar los scripts en Python del sistema IVR, facilitando la integración con APIs de IA y el control de versiones. |  |
| **ZOIPER version Android** | Teléfono móvil utilizado para la prueba de cliente VoIP con Zoiper. |  |
| **MicroSIP para Sistemas Operativos de Windows** | PC con MicroSIP utilizado como softphone para pruebas de llamadas. |  |

## 2.2 Métodos

### 2.2.1 Modalidad de la investigación

Para el desarrollo de este proyecto de titulación se emplean tres enfoques de investigación que permiten abordar tanto la parte teórica como la implementación práctica de la solución propuesta.

En primer lugar, se utiliza una investigación aplicada, ya que el objetivo principal es resolver una necesidad concreta: mejorar la comunicación académica y administrativa mediante un sistema IVR potenciado con agentes de inteligencia artificial, implementado sobre la plataforma Asterisk. Esta modalidad permite aplicar conocimientos técnicos para generar un impacto directo en el entorno institucional.

En segundo lugar, se recurre a la investigación bibliográfica, la cual implica la revisión detallada de fuentes secundarias como libros especializados, artículos científicos, tesis y documentos técnicos relacionados con IVR, procesamiento de lenguaje natural, inteligencia artificial y protocolos de telefonía IP (SIP, IAX y PJSIP). Esta fase proporciona los fundamentos teóricos necesarios para orientar el diseño de la solución tecnológica.

Finalmente, se incluye una investigación experimental, ya que el sistema será probado en entornos reales o simulados para medir su desempeño. Se evaluarán aspectos como el tiempo de respuesta, la precisión de las respuestas generadas y la satisfacción del usuario, con el fin de validar su funcionalidad frente a los métodos tradicionales de atención telefónica.

### 2.2.2 Población y muestra

Para el desarrollo del presente proyecto, no se definió una población y muestra en términos estadísticos tradicionales, ya que el objetivo principal no fue realizar un estudio masivo, sino diseñar y comprobar el funcionamiento del sistema propuesto. No obstante, se tomó como muestra a 105 estudiantes matriculados en el proceso de titulación de las carreras de Ingeniería Industrial con 31 estudiantes, Software con 25

estudiantes, Tecnologías de la Información con 18 estudiantes, Telecomunicaciones con 30 estudiantes e Ingeniería en Electrónica y Comunicaciones con 1 estudiante. Además, se aplicó una encuesta específica a 30 estudiantes de Ingeniería en Telecomunicaciones, con el propósito de identificar necesidades académicas frecuentes y validar el prototipo, utilizando sus aportes como retroalimentación técnica para ajustar y fortalecer la propuesta.

### 2.2.3 Recolección de información

Como parte del presente estudio de investigación, se utilizaron fuentes bibliográficas como libros especializados en telecomunicaciones y sistemas de voz sobre IP (VoIP), artículos científicos sobre la integración de inteligencia artificial en sistemas de respuesta interactiva automática, así como datos primarios obtenidos directamente de la implementación y pruebas del sistema IVR desarrollado con Asterisk.

Tabla 9. Tabla de parámetros

| **Criterio** | **Descripción** |
|---|---|
| ¿Para qué? | Optimizar la gestión de consultas académicas y administrativas mediante un sistema de respuesta interactiva automática con inteligencia artificial. |
| ¿De qué personas u objetos? | De los estudiantes, docentes y personal administrativo de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial. |
| ¿Cómo? | Mediante el diseño y desarrollo de un sistema IVR inteligente basado en Asterisk, capaz de responder de forma autónoma a consultas telefónicas frecuentes. |
| ¿Con qué? | Utilizando tecnologías avanzadas como Asterisk para la gestión de llamadas VoIP, algoritmos de procesamiento de lenguaje natural (PLN) para mejorar la interacción con los usuarios y agentes de inteligencia artificial para ofrecer respuestas automatizadas más precisas y naturales. |
| ¿Cuándo? | Durante el periodo Marzo 2025 – Julio 2025 |
| ¿Dónde? | En la Facultad de Ingeniería en Sistemas, Electrónica e Industrial de la Universidad técnica de Ambato. |

### 2.2.4 Procesamiento y análisis de datos

El tratamiento de la información en esta investigación siguió un proceso estructurado para evaluar la efectividad del sistema de respuesta interactiva automática (IVR) con inteligencia artificial. Los pasos por seguir son los siguientes:

- Recopilación de consultas frecuentes realizadas por los estudiantes en relación con trámites académicos, documentación requerida y ubicación de docentes.
- Clasificación y análisis de patrones en las preguntas registradas, identificando aquellas que generan mayor confusión o requieren información más detallada.
- Desarrollo de respuestas automatizadas optimizadas para cada tipo de consulta, garantizando claridad y precisión en la información proporcionada.
- Implementación y prueba del sistema IVR, simulando escenarios reales de interacción entre estudiantes y el sistema para evaluar su capacidad de respuesta.
- Medición del desempeño del sistema, analizando indicadores como el tiempo de respuesta, la precisión de las respuestas y la satisfacción de los usuarios.
- Ajustes y mejoras basados en los resultados obtenidos durante las pruebas, con el fin de optimizar la experiencia del usuario y la efectividad del sistema.

# CAPÍTULO III. RESULTADOS Y DISCUSIÓN

## 3.1 Desarrollo de la propuesta

La evolución que existe en la actualidad de los sistemas de telecomunicaciones ha promovido el desarrollo de la automatización de servicios de atención al usuario, generalmente en entornos en los cuales la demanda de consultas supera la capacidad del personal humano. Sin embargo, los sistemas IVR convencionales presentan varias limitaciones tanto en la escalabilidad como en la comprensión semántica. Para lograr solucionar estos problemas, en este trabajo se propone el diseño de un sistema de IVR inteligente, integrando agentes basados en inteligencia artificial que tienen un procesamiento de lenguaje natural con el fin de que las consultas telefónicas tengan un lenguaje libre y no sea una respuesta pregrabada.

## 3.2 Diagrama de arquitectura del sistema

El sistema IVR propuesto combinó tecnologías de código abierto con servicios en la nube para proporcionar atención telefónica automatizada e inteligente, mediante el uso de agentes de inteligencia artificial. Su arquitectura se organizó en varias capas: infraestructura virtual, plataforma PBX basada en Asterisk, configuración de archivos y scripts AGI, integración con modelos de lenguaje como GPT-4, y los dispositivos finales de telefonía IP. La Figura 1 representa el flujo lógico y técnico del sistema, detallando la interacción entre cada uno de sus componentes.

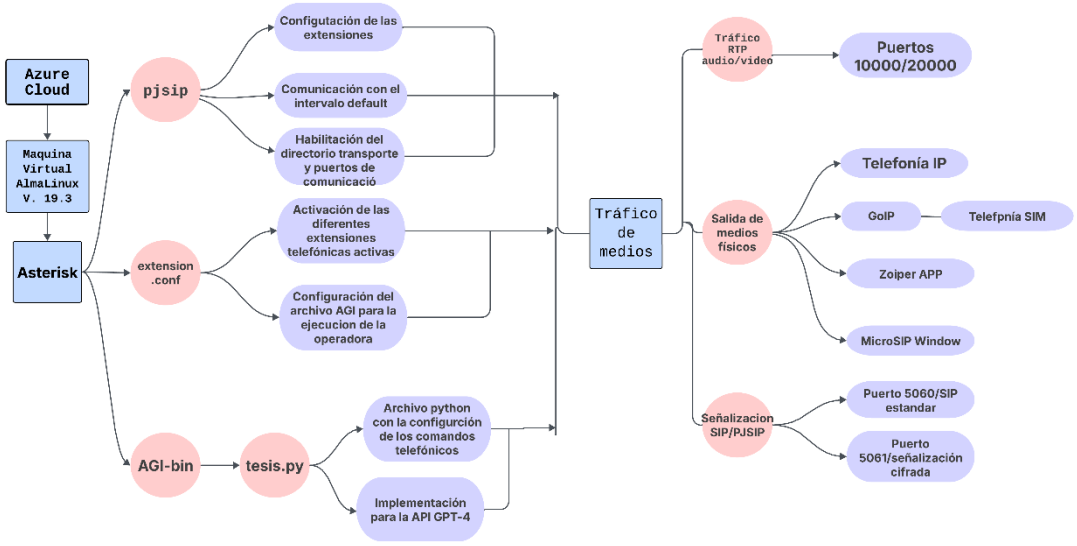



Figura 1. Diagrama de arquitectura del sistema

En el sistema propuesto, la interacción inició con el despliegue de una máquina virtual en Microsoft Azure, donde se implementó AlmaLinux 9.3 como sistema operativo, seleccionado por su estabilidad y compatibilidad con entornos VoIP. Sobre esta base se instaló Asterisk 18.12.1, que funcionó como núcleo del sistema PBX, encargado de gestionar las llamadas, enrutar extensiones y ejecutar scripts automatizados a través de AGI.

La configuración del sistema se realizó mediante los archivos pjsip.conf y extensions.conf. El primero definió los endpoints, los protocolos de transporte y habilitó los puertos necesarios para la señalización SIP/PJSIP. El segundo estableció las reglas de direccionamiento de llamadas, activó las extensiones y ejecutó el script tesis.py, ubicado en el directorio AGI-bin. Este script, desarrollado en Python, se conectó a la API de GPT-4 para interpretar el lenguaje natural recibido durante la llamada y generar respuestas contextualizadas.

Una vez en funcionamiento, el sistema permitió que los usuarios accedieran mediante dispositivos físicos o virtuales, como teléfonos IP, softphones (Zoiper, MicroSIP) o gateways como GoIP, a un menú de voz interactivo. La transmisión del audio se realizó a través de los puertos RTP (10000–20000), lo que aseguró la comunicación en tiempo real con el agente conversacional.

Esta arquitectura permitió brindar respuestas automáticas a consultas frecuentes sobre trámites académicos y administrativos sin necesidad de intervención humana. La integración entre telefonía IP, inteligencia artificial y plataformas de código abierto modernizó los canales de atención y estableció una base sólida para futuras ampliaciones del sistema.

### 3.2.1 Arquitectura del sistema

La arquitectura del sistema IVR se compone de tres etapas fundamentales, en la Figura 2 se presenta cada una de las etapas.

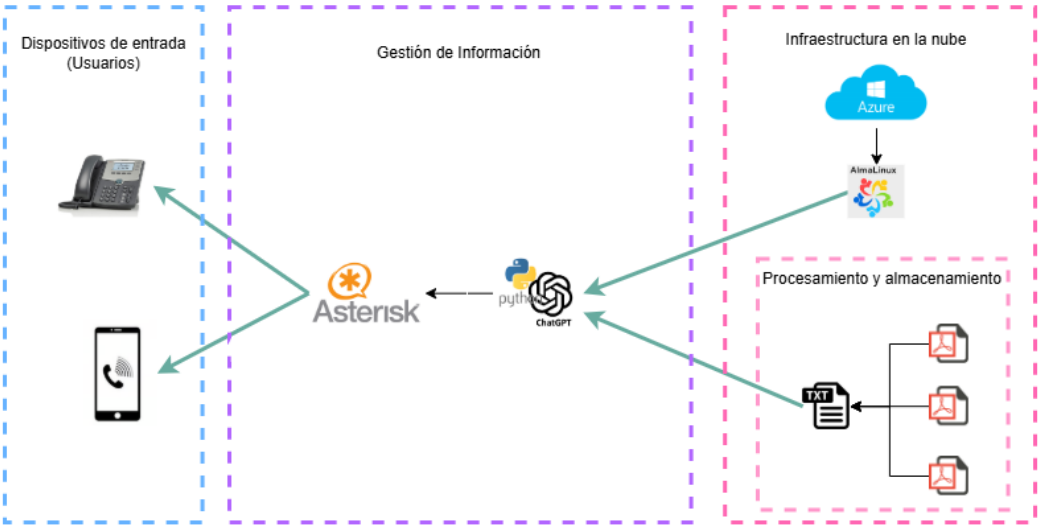



Figura 2. Arquitectura del sistema.

El proceso de integrar inteligencia artificial con un sistema PBX basado en Asterisk se inicia cuando un usuario realiza una llamada desde un teléfono IP o un smartphone; Asterisk 18.12.1, desplegado sobre AlmaLinux 9.3 dentro de máquinas virtuales configuradas en la infraestructura cloud de Microsoft Azure, recibe la llamada, establece la conexión mediante su núcleo de señalización y el módulo PJSIP, y gestiona la lógica de enrutamiento a través de su plan de marcado. Durante la interacción, la voz del interlocutor se captura y se procesa gracias a la Interfaz AGI, que permite que Asterisk se comunique con scripts Python que envían el audio a los Azure AI Speech Services para su conversión de voz a texto (STT) utilizando modelos optimizados para español latinoamericano. El texto transcrito es enviado al Azure

OpenAI Service, que ejecuta el modelo GPT-4, responsable de analizar el contexto de la consulta, procesar la información y generar una respuesta precisa y coherente, considerando tanto conocimiento general como documentos específicos de la institución académica. Esta respuesta se convierte nuevamente en audio mediante la funcionalidad de Text-to-Speech (TTS) de los mismos servicios cognitivos, retornando a Asterisk a través de la AGI para ser reproducida al usuario, cerrando así un ciclo de comunicación automatizado, fluido y en tiempo real. Todo este flujo opera sobre una arquitectura robusta donde AlmaLinux 9.3, elegido por su estabilidad, soporte extendido y kernel optimizado para baja latencia, asegura el rendimiento crítico de las cargas de trabajo de telecomunicaciones, mientras Azure proporciona máquinas virtuales con recursos dedicados, redes virtuales configuradas para el tráfico VoIP (puertos SIP y RTP), almacenamiento persistente para configuraciones y grabaciones de audio, y servicios de monitoreo que supervisan continuamente el funcionamiento para garantizar alta disponibilidad y escalabilidad de la solución.

## 3.3 Selección de tecnologías

Mediante un análisis comparativo, se identificaron los diferentes componentes con la finalidad de que exista escalabilidad, facilidad de integración y rendimiento óptimo al momento diseñar el sistema IVR con inteligencia artificial. La elección de la tecnología cumple con el primer objetivo específico del proyecto, lo que permite tener una base sólida en la integración de voz sobre IP, agentes de IA y servicios en la nube.

### *a. Plataforma de virtualización*

Se optó por la plataforma de virtualización VMware dado que es un entorno de pruebas del servidor Asterisk, utilizando el Workstation para el desarrollo local. Esta selección se enfoca en la solidez que tiene VMware, la capacidad para gestionar snapshots de una forma sencilla, teniendo así un excelente desempeño en las aplicaciones VoIP, dándonos como beneficios la baja latencia y escalabilidad. Además, la virtualización permite replicar el entorno sin afectar la configuración, de tal forma que facilita las tareas de prueba, respaldo y mantenimiento.

### *b. Sistema PBX*

Asterisk responde a la necesidad de una solución PBX de código abierto, siendo capaz de integrarse con agentes de IA por medio de APIs externas, es compatible con múltiples protocolos de señalización como por ejemplo SIP, IAX2 y PJSIP, además de eso tiene un soporte completo para interfaces de integración como AGI, AMI y ARI, como se detalla en la tabla 3. La compatibilidad de Asterisk con estas interfaces es perfecta para integrar servicios externos, como agentes de inteligencia artificial, motores de síntesis de voz y bases de datos. En este proyecto, Asterisk actúa como el corazón que gestiona las llamadas, dirige las consultas al agente de IA y reproduce las respuestas para el usuario.

### *b. Proveedor de nube*

Se eligió Microsoft Azure como plataforma en la nube dado que tiene una alta capacidad de escalado automático, un amplio soporte de servicios lo cuales están integrados con base de datos, almacenamientos y APIs de IA descritos en la tabla x. Permite máquinas virtuales con sistemas operativos personalizados, configuración de redes virtuales y adaptación de necesidad en entornos VoIP. La confiabilidad que existe se basa en las herramientas de monitoreo y respaldo integradas en el sistema. La oportunidad de ofrecer ejemplos rápidamente y establecer pautas de red particulares permite la replicación de la configuración de uso real y la preservación del sistema operativo y el software del sistema telefónico en escenarios regulados.

### *c. Agente de IA*

Se seleccionó GPT mediante la API de OpenAI, dado que tiene una alta capacidad de comprensión semántica, tiempo de respuesta bajo gracias a la aplicación de API REST bien documentada y un soporte de multilenguaje lo que facilita al procesamiento de lenguaje natural, presentada en la tabla x. Además, esta tecnología ayuda a interpretar de una manera clara las diferentes preguntas de los usuarios permitiendo generar respuestas dinámicas, de tal forma que la interacción telefónica se vuelve una forma mejorada frente a los sistemas IVR tradicionales. Finalmente, la integración con Python permite que el sistema se convierta en un componente clave para este proyecto.

### *d. Lenguaje de programación*

Para crear scripts que vinculan Asterisk con el agente de IA, Python es un lenguaje versátil para múltiples plataformas, con soporte comunitario fácil y amplio, se escogió, dando que se pueden crear módulos para enviar solicitudes a la API de OpenAI, registrar transcripciones y realizar tareas a medida que suceden, este sistema telefónico permite que los programas externos se ejecuten a través de su conexión AGI (interfaz de puerta de enlace de Asterisk). Python es un lenguaje utilizado comúnmente para este propósito.

## 3.4 Identificación de las necesidades de los estudiantes

Con el fin de entender mejor las principales necesidades y desafíos que enfrentan los estudiantes en cuanto a los trámites académicos y administrativos, se creó y se llevó a cabo una encuesta dirigida a los alumnos de los últimos semestres de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, que se detallan en el Figura 3, Figura 5, Figura 6, Figura 7, Figura 8.




Figura 3. Frecuencia con la que los estudiantes necesitan información académica o administrativa.

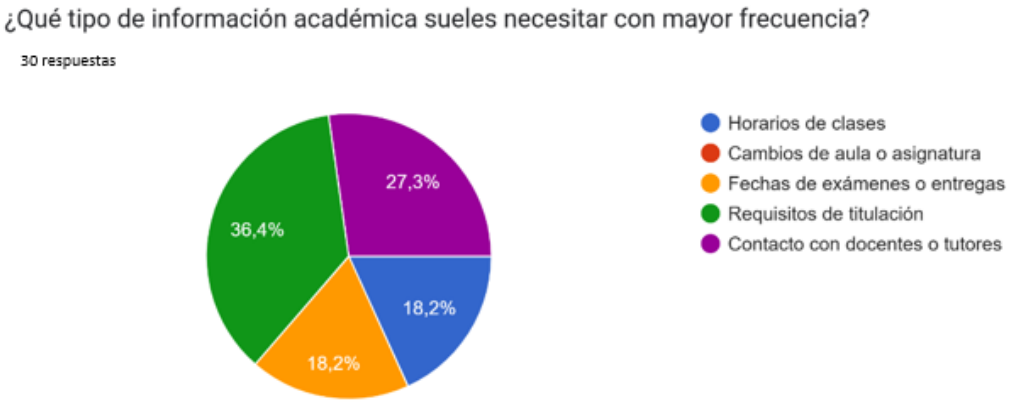



Figura 4. Tipo de información académica más requerida por los estudiantes.




Figura 5. Tipo de información administrativa más requerida por los estudiantes.

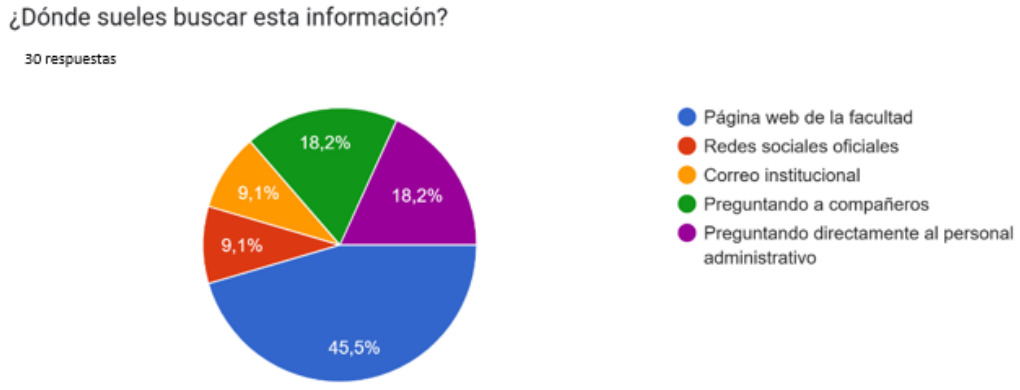



Figura 6. Medios utilizados por los estudiantes para buscar información académica o administrativa.




Figura 7. Frecuencia con la que los estudiantes obtienen información.




Figura 8. Funcionalidades preferidas por los estudiantes para el sistema automatizado.

Los resultados de la encuesta realizada a 30 estudiantes de los últimos semestres pusieron de manifiesto varias dificultades en el acceso a información académica y administrativa, esta encuesta fue realizada el 23 de mayo de 2025. Un gran porcentaje de los encuestados mencionó que consulta con frecuencia, ya sea semanal o diariamente, sobre temas como las fechas de matrícula, los procesos de titulación, los horarios de clases y la ubicación de los docentes.

Los canales de comunicación que utilizan dieron a notar una clara preferencia por medios informales, como grupos de WhatsApp y consultas presenciales. Esto se debe a la falta de claridad o actualización en los canales formales de la institución. Esta

situación provoca confusión, duplicación de esfuerzos y pérdida de tiempo. Además, la mayoría de los participantes mostró una actitud positiva hacia la implementación de sistemas automatizados de atención, como un menú interactivo por voz, siempre que estos ofrezcan respuestas claras, actualizadas y accesibles en cualquier momento.

Estos hallazgos ayudaron a definir criterios funcionales clave para el desarrollo del sistema IVR, tales como la disponibilidad 24/7, la precisión de la información proporcionada, todo alineado con las expectativas y necesidades reales de los estudiantes.

## 3.5 Microsoft Azure para los servicios de la máquina virtual en la nube

La línea de servicios de gestión en la nube de Microsoft Azure proporciona una infraestructura robusta de tal modo que se pueda implementar los diferentes servidores de Asterisk en un entorno cloud, ofreciendo flexibilidad; en el ámbito de que el sistema puede ser configurado en cualquier punto del mundo con acceso a internet, escalabilidad; ya que al querer ofrecer una amplia cobertura de extensiones o líneas de comunicación telefónica, estas pueden resultar fundamentales para una gestión operativa en tiempo real.

La selección adecuada de los servicios de Azure ha constituido un factor crítico para garantizar el rendimiento óptimo del sistema IVR inteligente propuesto en este proyecto.

### 3.5.1 Selección de recursos para Asterisk en Azure

Para la implementación de Asterisk en máquinas virtuales de Azure enfocadas en AlmaLinux ha requerido diversos factores técnicos que se han tomado en cuenta.

***a. Serie de máquinas virtuales***

Las series Dv4 o Dsv4 ofrecen una combinación óptima de CPU virtual y memoria para sistemas PBX, proporcionando un rendimiento consistente para el procesamiento de llamadas simultáneas. Para este proyecto, la máquina virtual Standard_D4s_v4 (4 vCPUs, 16 GB RAM) representa un balance adecuado entre rendimiento y costo para

alojar AlmaLinux y Asterisk, de tal modo que ha sido la opción directa para la aplicación de nuestro servidor telefónico.

***b. Almacenamiento Estándar***

Se implementó el almacenamiento estándar disponible en la suscripción de Azure para estudiantes, que ofrece un rendimiento suficiente para cargas de trabajo académicas. Esta opción, aunque no alcanza la baja latencia de las soluciones Premium, proporciona un equilibrio adecuado entre costo y rendimiento para el acceso a archivos de configuración, bases de datos locales y grabaciones de audio durante las llamadas con el agente de IA en un entorno educativo.

***c. Redes Optimizadas***

Las máquinas virtuales con conectividad de red acelerada son preferibles para Asterisk, ya que reducen la latencia y el jitter, problemas que pueden comprometer la calidad de las llamadas VoIP.

### 3.5.2 Configuración de red para comunicaciones VoIP

La configuración de red en Azure para el servidor Asterisk basado en AlmaLinux incluye:

***a. Red virtual dedicada.***

Se ha implementado dentro de una VNET personalizada que permite aislar el tráfico y aplicar políticas de seguridad específicas para protocolos SIP y RTP.

***b. Grupos de seguridad de red (NSG):***

Estos están configurados para permitir exclusivamente el tráfico en los puertos necesarios:

- ***Puerto 5060/5061*** (TCP/UDP) para señalización SIP
- ***Puerto 4569 (UDP)*** para protocolo IAX2
- ***Rango de puertos 10000-20000 (UDP)*** para tráfico RTP multimedia

- ***Puerto 22 (TCP)*** para administración segura por SSH

### *c. Estructura de la red actual*

La arquitectura de red telefónica implementada en la facultad se organiza a partir de un switch central principal, el cual distribuye la conectividad hacia múltiples áreas y edificios mediante enlaces físicos estructurados. Desde este switch central, se conectan directamente los teléfonos del subdecanato, secretarías y biblioteca del Edificio 1. Asimismo, se ramifican tres switches secundarios: uno ubicado en la sala de profesores 1, otro en el laboratorio 7, y un tercero en el área administrativa del tercer piso, donde se interconectan los teléfonos de decanato, secretaría y unidad de titulación. Para extender la red al Edificio 2, se establece un enlace físico que conecta el switch central del Edificio 1 con un switch central en el Edificio 2, desde el cual se conecta el teléfono del área de redes y, a su vez, un switch adicional que enlaza dos teléfonos más. Esta distribución jerárquica garantiza una cobertura eficiente, segmentada y escalable, optimizando la gestión de tráfico VoIP y facilitando la administración del sistema telefónico IP, como se puede ver en la Figura 9.

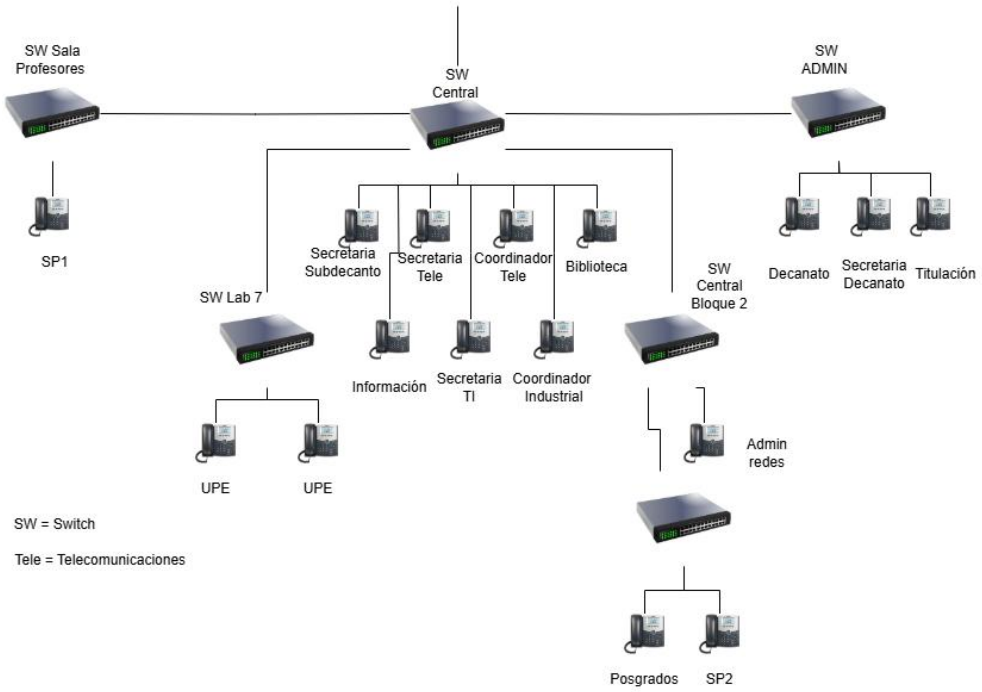



Figura 9. Estructura red telefónica.

### *d. IP pública estática*

Se asignó una dirección IP fija para garantizar la conectividad consistente desde clientes externos y trunks JSIP, esta IP es la que nos proporciona Azure después de la creación de la máquina virtual, de modo que la configuración es predeterminada u por defecto del sistema de Microsoft.

### 3.5.3 Alta disponibilidad y respaldo

Para garantizar la continuidad del servicio telefónico, la máquina virtual de AlmaLinux en Azure implementó una configuración de copias de seguridad periódicas mediante Azure Backup, en la cual se incluyeron las siguientes estrategias:

- **Archivos de configuración de Asterisk (/etc/asterisk/)**
- **Base de datos de llamadas y conversaciones**
- **Grabaciones de audio de interacciones con usuarios**
- **Scripts AGI personalizados para la integración con GPT**

Esta arquitectura integrada de AlmaLinux en máquinas virtuales de Azure proporciona la base tecnológica necesaria para implementar un sistema IVR inteligente basado en Asterisk que sea robusto, seguro y capaz de escalar según los requerimientos institucionales de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial.

## 3.6 Azure AI Foundry para la integración de servicios cognitivos

Microsoft Azure AI Foundry se estableció como una plataforma fundamental para la implementación de servicios de inteligencia artificial en este proyecto. Esta suite de herramientas y servicios ha permitió integrar capacidades avanzadas de procesamiento de lenguaje natural y síntesis de voz en el sistema Asterisk desplegado en AlmaLinux, potenciando significativamente las capacidades conversacionales del IVR inteligente.

### 3.6.1 Integración de APIs de GPT-4 mediante Azure OpenAI Service

Para el núcleo cognitivo del sistema IVR, se ha implementado la integración con GPT-4 a través del servicio Azure OpenAI. Esta decisión proporcionó ventajas significativas en términos de seguridad, cumplimiento de acuerdo con los requerimientos de respuesta vocal y estabilidad operativa:

- ***Autenticación unificada***

Se configuró un sistema de autenticación basado en Azure Active Directory, lo que permitió una gestión centralizada de credenciales y permisos para acceder a las APIs de GPT-4 desde los scripts AGI ejecutados en AlmaLinux.

- ***Punto de conexión dedicado***

Se establecio un endpoint regional específico dentro de Azure South East para minimizar la latencia en las comunicaciones entre el servidor Asterisk y los servicios de IA, factor crítico para mantener conversaciones telefónicas fluidas.

- ***Cuotas y limitaciones:***

Se implementó políticas de rate limiting que garantizan la disponibilidad del servicio incluso en momentos de alta demanda, estableciendo prioridades para llamadas concurrentes y mecanismos de cola para gestionar picos de tráfico.

El código de integración se desarrolló en Python, aprovechando la biblioteca oficial de Azure para OpenAI. Se implementó una capa de abstracción que gestionó aspectos críticos como el manejo de sesiones, reintentos automáticos y la conversión entre formatos de datos utilizados por Asterisk y los requeridos por la API.

### 3.6.2 Implementación de Azure AI Speech Services para procesamiento de voz

La conversión bidireccional entre voz y texto se realizó mediante Azure AI Speech Services, componente esencial para el funcionamiento del sistema IVR conversacional:

- ***Reconocimiento de voz (STT)***

Se implementó el servicio de Speech-to-Text con modelos acústicos optimizados para el español de Ecuador, lo que mejoró significativamente la precisión en la transcripción de consultas telefónicas realizadas por los estudiantes. La configuración se ajustó para filtrar ruidos ambientales y compensar las limitaciones del canal telefónico (8kHz).

- **Síntesis de voz (TTS)**

Se seleccionó y configuró voces neuronales en español latinoamericano del catálogo de Azure, priorizando naturalidad y claridad en la pronunciación. Para optimizar la experiencia del usuario, se implementó el ajuste dinámico de parámetros prosódicos como velocidad, tono y pausas según el contexto de la conversación.

- **WebSockets para transmisión en tiempo real**:

WebSockets para transmisión en tiempo real: Se estableció un sistema de streaming bidireccional que permitió iniciar la síntesis o el reconocimiento antes de que el audio completo esté disponible, reduciendo así la latencia percibida durante las interacciones.

### 3.6.3 Personalización del modelo GPT-4 para el contexto académico

Para garantizar respuestas precisas y relevantes en el contexto de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, se realizó un extenso trabajo de personalización:

- **Prompt engineering**: Se diseñó y optimizó los prompts del sistema que proporcionan al modelo GPT-4 información contextual sobre la facultad, procesos administrativos y terminología académica específica.

- **Base de conocimientos**: Se creó una base de conocimientos estructurada con información actualizada sobre programas académicos, calendarios, requisitos de trámites y contactos departamentales, que se ha incorporado en las solicitudes a la API.

- **Finetuning del modelo**: Se implementó un proceso de refinamiento mediante ejemplos curados que ilustran el tipo de consultas frecuentes y las respuestas ideales, mejorando así la precisión y relevancia del modelo para el dominio específico.

- **Mecanismos de verificación**: Se estableció filtros post-procesamiento que validan la exactitud de la información proporcionada, especialmente en casos donde los datos son críticos como fechas límite o requisitos de documentación.

## 3.7 Instalación y configuración del entorno Asterisk en AlmaLinux

La implementación exitosa del sistema IVR inteligente ha requerido la instalación y configuración meticulosa de múltiples componentes en el servidor AlmaLinux desplegado en Azure. A continuación, se detalla el proceso técnico realizado para establecer una base sólida y funcional que soporte la integración entre Asterisk, Python y los servicios de Azure AI.

### 3.7.1 Preparación del sistema operativo base

Antes de instalar cualquier componente específico, se preparó el sistema operativo AlmaLinux con las actualizaciones y herramientas básicas necesarias según los comandos presentados en la Tabla 10 cronológicamente.

Tabla 10. Comandos de instalación.

| No | Comandos | Función |
|---|---|---|
| 1 | sudo dnf update -y | Actualización completa del sistema operativo |

| | | |
|---|---|---|
| 2 | sudo dnf install -y wget curl git nano unzip tar htop nmon net-tools | Instalación de herramientas básicas de sistema |
| 3 | sudo dnf install -y epel-release | Instalación de repositorio EPEL para paquetes adicionales |
| 4 | sudo firewall-cmd --permanent --add-port=5060/udp --add-port=5060/tcp | Configuración del firewall para permitir tráfico SIP y RTP |
| 5 | sudo firewall-cmd --permanent --add-port=5061/udp --add-port=5061/tcp | |
| 6 | sudo firewall-cmd --permanent --add-port=4569/udp | |
| 7 | sudo firewall-cmd --permanent --add-port=10000-20000/udp | |
| 8 | sudo firewall-cmd --reload | |

Estos comandos establecieron una base sólida, actualizando el sistema a la última versión y habilitando los puertos necesarios en el firewall para el tráfico SIP/RTP.

### 3.7.2 Instalación de Asterisk desde fuentes

Se optó por compilar Asterisk desde el código fuente para garantizar la compatibilidad con todas las extensiones necesarias y configurar opciones específicas para el proyecto, según la Tabla 11.

Tabla 11. Código de Asterisk para las extensiones.

| **No** | **Comandos** | **Función** |
|---|---|---|
| 1 | sudo dnf groupinstall -y "Development Tools" | Instalación de dependencias para compilar Asterisk |
| 2 | sudo dnf install -y wget tar gcc gcc-c++ make cmake kernel-devel linux-headers \ | |
| 3 | ncurses-devel libxml2-devel sqlite-devel openssl-devel libuuid-devel \ | |
| 4 | jansson-devel libedit-devel libuuid-devel uuid-devel libtiff-devel \ | |
| 5 | libjpeg-turbo-devel opus-devel speex-devel speexdsp-devel libsndfile-devel | |
| 6 | cd /usr/src/ | Descarga e instalación de Asterisk 20 |
| 7 | sudo wget https://downloads.asterisk.org/pub/telephony/asterisk/asterisk-20-current.tar.gz | |
| 8 | sudo tar -xvzf asterisk-20-current.tar.gz | |
| 9 | cd asterisk-20*/ | |
| 10 | sudo ./configure --with-jansson-bundled --with-pjproject-bundled | Configuración con soporte para menús interactivos y depuración |
| 11 | sudo make menuselect | Selección manual de módulos requeridos |

En el menú de selección que aparece al ejecutar *make menuselect*, se habilitó los siguientes componentes esenciales:

- **Aplicaciones**: app_agi, app_macro, app_record, app_playback, app_system
- **Recursos**: res_speech, res_agi, res_http_websocket
- **Codecs**: codec_opus, codec_ulaw, codec_alaw, codec_gsm
- **Formatos**: format_wav, format_gsm

Tras realizar estas selecciones, se procedió con la compilación e instalación indicado en la Tabla 12.

Tabla 12. Códigos de instalación.

| No | Comandos | Función |
|---|---|---|
| 1 | sudo make -j$(nproc) | Compilación utilizando todos los núcleos disponibles para acelerar el proceso |
| 2 | sudo make install | Instalación de Asterisk y archivos de configuración de ejemplo |
| 3 | sudo make samples | |
| 4 | sudo make config | |
| 5 | sudo useradd -r asterisk | Creación de usuario específico para Asterisk por seguridad |
| 6 | sudo chown -R asterisk:asterisk /var/lib/asterisk | |
| 7 | sudo chown -R asterisk:asterisk /var/spool/asterisk | |
| 8 | sudo chown -R asterisk:asterisk /var/log/asterisk | |
| 9 | sudo chown -R asterisk:asterisk /var/run/asterisk | |
| 10 | sudo sed -i 's/#AST_USER="asterisk"/AST_USER="asterisk"/' /etc/default/asterisk | Configuración para ejecutar Asterisk con el usuario asterisk |
| 11 | sudo sed -i 's/#AST_GROUP="asterisk"/AST_GROUP="asterisk"/' /etc/default/asterisk | |
| 12 | sudo systemctl enable asterisk<br>sudo systemctl start asterisk | Configuración del servicio systemd para iniciar automáticamente |

La creación de un usuario específico para Asterisk reforzó la seguridad del sistema, siguiendo las mejores prácticas de principio de mínimo privilegio.

### 3.7.3 Configuración del entorno Python para integración con AGI

Para la ejecución de los scripts AGI que conectan Asterisk con los servicios de Azure AI, se preparó un entorno Python completo con todas las dependencias necesarias indicadas en la Tabla 13.

Tabla 13. Códigos de instalación en Python.

| No | Comandos | Función |
|---|---|---|
| 1 | sudo dnf install -y python39 python39-devel python39-pip python39-setuptools python39-wheel | Instalación de Python 3.9 y herramientas de desarrollo |
| 2 | sudo dnf install -y portaudio-devel ffmpeg ffmpeg-devel sox sox-devel swig \ | Instalación de bibliotecas del sistema para procesamiento de audio y SSL |
| 3 | openssl-devel libffi-devel bzip2-devel zlib-devel | |
| 4 | sudo mkdir -p /opt/asterisk-agi | Creación de un entorno virtual dedicado para las dependencias de AGI |
| 5 | sudo python3.9 -m venv /opt/asterisk-agi/venv | |
| 6 | sudo chown -R asterisk:asterisk /opt/asterisk-agi | |
| 7 | source /opt/asterisk-agi/venv/bin/activate | Activación del entorno virtual e instalación de dependencias Python |
| 8 | pip install --upgrade pip setuptools wheel | Actualización de herramientas de gestión de paquetes |
| 9 | pip install pyst2 asterisk-agi | Instalación de bibliotecas para interacción con Asterisk |
| 10 | pip install SpeechRecognition pydub numpy librosa | Instalación de bibliotecas para reconocimiento de voz y procesamiento de audio |
| 11 | pip install azure-ai-inference==1.0.0 azure-identity azure-core==1.29.5 | Instalación de bibliotecas para integración con Azure |
| 12 | pip install azure-cognitiveservices-speech azure-storage-blob | |
| 13 | pip install requests urllib3 certifi python-dotenv | Instalación de bibliotecas auxiliares |

Este conjunto de dependencias creó un entorno aislado y completo que permitió la integración entre Asterisk, los servicios de reconocimiento de voz y los modelos de lenguaje de Azure, sin afectar al sistema base.

### 3.7.4 Configuración de la estructura de los directorios para AGI

Dentro de esta parte, al momento de querer ejecutar cualquier script para Asterisk según sea automatización, IVR con sonidos pregrabados, interpretación y codificación de lo que requiera el usuario, entre otros. Se necesitó configurar los directorios y permisos necesarios para que se pueda establecer estos parámetros.

Entonces, en la Tabla 14, se estableció una estructura de directorios organizada que facilita la gestión de los scripts AGI, archivos de configuración y contenido de conocimiento.

Tabla 14. Códigos para los scripts AGI.

| No | Comandos | Función |
|---|---|---|
| 1 | sudo mkdir -p /var/lib/asterisk/agi-bin | Creación de directorio principal para scripts AGI |
| 2 | sudo chmod 755 /var/lib/asterisk/agi-bin | |
| 3 | sudo mkdir -p /var/lib/asterisk/agi-bin/texts | Creación de directorios específicos para organización |
| 4 | sudo mkdir -p /var/lib/asterisk/agi-bin/logs | |
| 5 | sudo mkdir -p /var/lib/asterisk/agi-bin/configs | |
| 6 | sudo mkdir -p /var/lib/asterisk/sounds/custom | Directorio para archivos de audio temporales y personalizados |
| 7 | sudo mkdir -p /tmp/asterisk_audio | |
| 8 | sudo chown -R asterisk:asterisk /var/lib/asterisk/agi-bin | Configuración de permisos apropiados |
| 9 | sudo chown -R asterisk:asterisk /var/lib/asterisk/sounds/custom | |
| 10 | sudo chown -R asterisk:asterisk /tmp/asterisk_audio | |
| 11 | sudo chmod 755 /var/lib/asterisk/sounds/custom | |
| 12 | sudo chmod 1777 /tmp/asterisk_audio | |

## 3.8 Desarrollo e implementación del script AGI principal

### 3.8.1 Configuración del archivo tesis.py

El código mostrado en la Figura 10, corresponde a la sección inicial del script tesis.py del AGI (*Asterisk Gateway Interface*) que actúo como puente entre el sistema telefónico Asterisk y los servicios cognitivos de Azure. Esta primera sección estableció las dependencias fundamentales que permiten al sistema realizar el procesamiento de llamadas, reconocimiento de voz y comunicación con los modelos de inteligencia artificial, la configuración del código del script tesis.py se encuentra en el

Anexo CD.

```python
#!/usr/bin/env python3

import os
import time
import speech_recognition as sr
from asterisk.agi import *
import subprocess
import requests  # Para llamadas REST a Azure
from azure.ai.inference import ChatCompletionsClient
from azure.ai.inference.models import SystemMessage, UserMessage
from azure.core.credentials import AzureKeyCredential
```

Figura 10. Importación de las diferentes librerías en formato .py

En las primeras líneas se observa la importación de bibliotecas estándar de Python como os, time y subprocess, que proporcionaron funcionalidades esenciales para la manipulación del sistema de archivos, control temporal y ejecución de comandos externos, respectivamente. Estas bibliotecas permiten al script interactuar con el entorno del sistema operativo AlmaLinux, facilitando tareas como la gestión de archivos temporales para las grabaciones de audio y el procesamiento de conversiones de formato mediante herramientas externas como FFmpeg. La inclusión de la biblioteca speech_recognition (importada como sr) es particularmente importante, ya que proporcionó las funcionalidades necesarias para transcribir el audio capturado durante la llamada telefónica a texto que posteriormente será procesado.

La integración con Asterisk se realizó mediante la importación desde asterisk.agi, lo que permite al script comunicarse con el servidor PBX y acceder a las funcionalidades de control de llamadas. Esto posibilita la captura de audio del usuario, la reproducción de respuestas sintetizadas y la gestión del flujo de la conversación telefónica.

Por otro lado, las importaciones que se proporcionado y que están relacionadas con Azure (azure.ai.inference, azure.core.credentials) establecieron la infraestructura necesaria para comunicarse con los servicios cognitivos de Microsoft, específicamente con el modelo GPT-4 a través de ChatCompletionsClient y con los servicios de síntesis de voz.

La biblioteca requests complementa esta funcionalidad permitiendo la comunicación REST directa con las APIs de Azure AI Voice, necesaria para la conversión de las respuestas textuales a audio natural que será reproducido al usuario a través del canal telefónico. Esta arquitectura de integración permitió que una simple llamada telefónica se transforme en una conversación inteligente asistida por los modelos de lenguaje más avanzados disponibles en la nube de Microsoft Azure.

El fragmento de código mostrado Figura 11, es la parte fundamental del script que estableció la conexión con los servicios de Azure OpenAI, configurando específicamente el acceso al modelo GPT-4. Esta sección es crítica para el funcionamiento del sistema IVR inteligente, ya que definió el punto de conexión (endpoint) al recurso de Azure OpenAI desplegado en la nube y establece los parámetros de autenticación necesarios. La variable endpoint almacena la URL completa que indica la ubicación del recurso en Azure, incluyendo el identificador único del despliegue de IA y la especificación del modelo GPT-4, mientras que la variable model_name simplemente almacena el identificador del modelo que se utilizará para las consultas.

```
13 # ConfiguraciÃ³n de Azure OpenAI
14 endpoint = "https://ai-mgallegos39471849ai543990149288.open
   ai.azure.com/openai/deployments/gpt-4"
15 model_name = "gpt-4"
16
17 client = ChatCompletionsClient(
18     endpoint=endpoint,
19     credential=AzureKeyCredential("GEgSzcywZfrCDXlO1M9bUTUT
   QXB9h2v3mYFu0eYlibsZPpKGmyDeJQQJ99BCACHYHv6XJ3w3AAAAACOGXgS
   J"),
20 )
```

Figura 11. Endpoint de Azure para GPT4

La inicialización del cliente mediante ChatCompletionsClient representa el establecimiento del canal de comunicación entre el sistema Asterisk y los servicios cognitivos de Azure. Este objeto cliente es el responsable de gestionar todas las solicitudes posteriores al modelo GPT-4, encapsulando la lógica de comunicación HTTP, autenticación y manejo de errores.

La instanciación recibe dos parámetros esenciales: el endpoint que indica dónde se encuentra el recurso de IA y el objeto AzureKeyCredential que contiene la clave de API necesaria para autenticar las solicitudes. Esta clave de API actúo como un token de seguridad que garantiza que solo las aplicaciones autorizadas puedan acceder al recurso de Azure OpenAI, protegiendo tanto el servicio como el presupuesto asignado al proyecto mediante la limitación de acceso.

La estructura del código sigue un patrón de configuración claro y directo, separando la definición de parámetros (URL del endpoint y nombre del modelo) de la inicialización del cliente que realiza las operaciones. Esta arquitectura facilita la futura actualización de la configuración, ya que los cambios en el endpoint o la clave de API pueden realizarse modificando únicamente las líneas correspondientes sin afectar al resto de la lógica del programa.

Adicionalmente, el uso de un objeto cliente específico para Azure OpenAI permitió aprovechar funcionalidades como el manejo automático de reintentos, la gestión de errores HTTP y la serialización/deserialización de mensajes JSON, simplificando considerablemente el código necesario para interactuar con la API del servicio de inteligencia artificial.

El fragmento mostrado en Figura 12, correspondió a la configuración esencial del servicio de síntesis de voz de Azure dentro del script AGI que conectó Asterisk con los servicios cognitivos en la nube. En estas líneas se definieron dos variables fundamentales: speech_key, que almacenó la clave de autorización para acceder a la API de Azure AI Voice, y speech_region, que especificó la región geográfica "eastus2" donde se encontraba desplegado el servicio. Estos parámetros fueron imprescindibles para establecer una comunicación autenticada con los servicios de síntesis de voz que transformaron las respuestas textuales en audio natural.

```
22 # ConfiguraciÃ³n del servicio de voz de Azure
23 speech_key = "2N37UG7pXFxZ4nDjPDqYEuiHmhwH9fBJgzAMfn77PsNSX
   Q6RZ407JQQJ99BCACHYHv6XJ3w3AAAAACOG3MeD"
24 speech_region = "eastus2"
25
```

Figura 12. Configuración de la API de Voz de Azure AI

Esta configuración constituyó el complemento perfecto a la integración previa con Azure OpenAI, completando así el ciclo de procesamiento bidireccional que permitió transformar la voz del usuario en texto, procesarla mediante GPT-4 y convertir la respuesta nuevamente en voz, en el

Anexo E se encuentra la configuración detallada.

La región seleccionada (eastus2) proporcionó un equilibrio óptimo entre latencia y disponibilidad para despliegues en América Latina, factor crítico para mantener conversaciones telefónicas fluidas con tiempos de respuesta aceptables. La clave de API utilizada fue específica para el servicio de voz y se utilizó exclusivamente para las operaciones de síntesis, manteniendo así una adecuada separación de responsabilidades entre los diferentes servicios cognitivos.

El código siguió el mismo patrón de configuración directa observado anteriormente, manteniendo claridad y facilidad de mantenimiento. Estas variables fueron utilizadas

posteriormente en la función text_to_speech_azure() que realizó la conversión de texto a voz mediante llamadas REST a la API de Azure Speech Services.

La elección de configurar estos parámetros al inicio del script facilitó la modificación de credenciales o regiones sin necesidad de modificar la lógica de procesamiento, siguiendo buenas prácticas de programación que separaron claramente configuración y funcionalidad.

Según en lo mostrado de la Figura 13, correspondió a una parte crucial del sistema implementado, específicamente a la construcción del prompt inteligente que se envió al modelo GPT-4. En la línea 49, se implementó una estrategia de optimización que limitó el contexto a los primeros 3000 caracteres del documento, como lo indicó claramente el comentario en color azul. Esta decisión técnica fue fundamental para gestionar eficientemente el límite de tokens en las consultas a la API de Azure OpenAI y reducir los tiempos de respuesta.

```
49            contexto_recortado = contexto_txt[:3000]  # Limitado a 3000
   caracteres
50
51            prompt = f"""Por favor, responde a la siguiente pregunta ut
   ilizando el contenido del archivo de texto proporcionado como contexto.
52
53 Contenido del archivo:
54 {contexto_recortado}
55
56 Pregunta: {pregunta}
57
58 IMPORTANTE: Tu respuesta debe ser MUY CONCISA (mÃ¡ximo 150 caracteres)
   ya que serÃ¡ convertida a voz.
59 Si la respuesta se encuentra en el archivo, proporciona solo la informa
   ciÃ³n esencial.
60 Si no encuentras la informaciÃ³n en el archivo, responde brevemente con
    tu conocimiento general.
61 """
```

Figura 13. Configuración de los PROMPTs para el modelo GPT4

A partir de la línea 51, se construyó un prompt estructurado en formato de cadena f-string que incluyó varios elementos estratégicos. El sistema instruyó al modelo con precisión sobre cómo debía procesar la consulta, comenzando con una directriz clara para utilizar el contenido del archivo de texto como fuente primaria de información. Se incorporó el contexto recortado en la línea 54 y la pregunta transcrita del usuario en la línea 56, estableciendo así el escenario completo para el procesamiento.

Las instrucciones adicionales implementadas entre las líneas 58-60 fueron particularmente significativas para la experiencia del usuario final. Se especificó que las respuestas debían ser "MUY CONCISAS" con un límite estricto de 150 caracteres, una restricción crucial dado que el texto sería posteriormente convertido a voz y reproducido por teléfono.

Además, se añadieron instrucciones para priorizar la información esencial si la respuesta se encontraba en el archivo, o para responder brevemente con conocimiento general en caso contrario. Esta estructura de prompt representó una implementación cuidadosamente diseñada que balanceó la precisión de la información, la concisión necesaria para la comunicación telefónica y la experiencia fluida del usuario final.

En la Figura 14, corresponde a la implementación de la conexión con la API REST de Azure Speech Services, configurando los elementos necesarios para realizar la síntesis de voz de alta calidad. La implementación comenzó con la construcción dinámica del endpoint de la API, utilizando la región previamente configurada (speech_region) dentro de la URL completa del servicio cognitivo de Microsoft.

```
108         endpoint = f"https://{speech_region}.tts.speech.microsoft.com/cognitiveservices
    /v1"
109
110         # Encabezados para la solicitud
111         headers = {
112             'Ocp-Apim-Subscription-Key': speech_key,
113             'Content-Type': 'application/ssml+xml',
114             'X-Microsoft-OutputFormat': 'riff-24khz-16bit-mono-pcm'
115         }
116
117         # Preparar SSML
118         ssml = f"""
119         <speak version='1.0' xmlns='http://www.w3.org/2001/10/synthesis' xml:lang='es-E
    S'>
120             <voice name='es-ES-ElviraNeural'>{texto_limpio}</voice>
121         </speak>
122         """
123
```

Figura 14. Configuración de los parámetros de Voz de Azure AI.

Se configuró un diccionario de encabezados HTTP esenciales para la comunicación con la API de síntesis de voz. Este diccionario incluyó la clave de autenticación (Ocp-Apim-Subscription-Key), que se asignó con el valor de speech_key definido anteriormente.

Adicionalmente, se especificó el tipo de contenido como application/ssml+xml, indicando que la solicitud utilizaría el formato SSML (Speech Synthesis Markup Language). El formato de salida de audio se configuró como riff-24khz-16bit-mono-pcm, solicitando un audio de alta calidad con frecuencia de muestreo de 24kHz y codificación PCM de 16 bits en mono.

La parte final del fragmento mostró la construcción del documento SSML que controló las características de la voz sintetizada. Se utilizó la sintaxis de cadenas f-string para incorporar dinámicamente el texto a sintetizar dentro de la estructura XML. El código especifica explícitamente el uso de la voz neural es-ES-ElviraNeural, una voz femenina española que proporcionó una pronunciación natural y fluida para el contexto latinoamericano.

La estructura SSML completa incluyó los elementos <speak> con los atributos de versión, espacio de nombres y configuración de idioma, garantizando que el texto se procesa correctamente según los estándares XML de síntesis de voz.

Esta implementación permitió generar respuestas de voz de alta calidad que fueron luego reproducidas a los usuarios a través del sistema telefónico Asterisk, proporcionando una experiencia auditiva natural y agradable durante la interacción con el sistema IVR inteligente.

Además, se incorporó una estrategia de conversión basada en la herramienta FFmpeg según la Figura 15, definiendo un comando específico que redujo la frecuencia de muestreo del archivo de audio a 8000 Hz y lo convirtió a formato monocanal. Estos parámetros fueron seleccionados específicamente para cumplir con los requisitos de audio de Asterisk en sistemas telefónicos tradicionales. El comando se ejecutó con la opción "-loglevel quiet" para minimizar la información de depuración no esencial durante la operación.

```
142             # Convertir a formato Asterisk
143             comando = f"ffmpeg -y -i {temp_wav} -ar 8000 -ac 1 {output_wav}
    loglevel quiet"
144
145             if agi:
146                 agi.verbose(f"Convirtiendo formato para Asterisk...")
147
148             process = subprocess.run(comando, shell=True, stdout=subprocess.
    IPE, stderr=subprocess.PIPE)
```

Figura 15. Conversión al formato valido de audio para Asterisk

El código implementó un manejo extensivo de errores y verificaciones de estado para garantizar la robustez del proceso de conversión indicados en la Figura 16. Se registraron mensajes informativos en el sistema de logs de Asterisk mediante llamadas a `agi.verbose()`, lo que facilitó el seguimiento y depuración del flujo de ejecución. Tras la conversión exitosa, se eliminó automáticamente el archivo temporal para evitar la acumulación de archivos innecesarios en el sistema.

```
150             # Limpiar archivo temporal
151             if os.path.exists(temp_wav):
152                 os.remove(temp_wav)
153
154             if process.returncode == 0 and os.path.exists(output_wav):
155                 if agi:
156                     agi.verbose(f"Texto convertido exitosamente a voz con Az
    re")
157                 return True
158             else:
159                 if agi:
160                     agi.verbose("Error al convertir el audio con ffmpeg")
161                 return False
162         else:
163             if agi:
164                 agi.verbose(f"Error en la API de Azure: {response.status_cod
    }")
165                 if response.text:
166                     agi.verbose(f"Detalles: {response.text[:100]}...")
167             return False
```

Figura 16. Uso del agi.verbose para los logs del CLI de Asterisk

La implementación incluyó múltiples niveles de verificación de errores: primero comprobó el código de retorno del proceso FFmpeg y la existencia del archivo de salida; luego manejó posibles errores en la respuesta de la API de Azure; y finalmente capturó cualquier excepción imprevista en el proceso de síntesis, tal y como se observa en la Figura 17.

```
169     except Exception as ex:
170         if agi:
171             agi.verbose(f"Error en sÃ-ntesis de Azure: {str(ex)}")
172         return False
```

Figura 17. Salida de error si existe alguna falla con los servidores de Azure AI

Este enfoque por capas para el manejo de errores garantizó que el sistema pudiera identificar con precisión la fuente de cualquier problema y proporcionar información de diagnóstico relevante, lo que fue crítico para mantener la fiabilidad del servicio IVR en un entorno de producción.

Se desarrolló un mecanismo anidado según la Figura 18, de priorización en tres niveles que comenzó buscando específicamente documentos relacionados con "titulacion". Esta búsqueda se implementó mediante una comprensión de lista que filtró los archivos TXT disponibles, verificando si la palabra clave aparecía en el nombre del archivo (convertido a minúsculas para una comparación insensible a mayúsculas). Cuando se encontraban archivos que cumplían con este criterio, el sistema selecciona automáticamente el primero de ellos, registraba esta selección mediante una entrada de log para facilitar la depuración, y devolvía la ruta completa del archivo.

```
197     # Si hay un TXT llamado "Titulacion.txt" o similar, priorizarlo
198     target_files = [f for f in txt_files if "titulacion" in f.lower()]
199     if target_files:
200         selected_txt = target_files[0]
201         if agi:
202             agi.verbose(f"Seleccionado archivo especÃ-fico: {selected_txt}")
203         return os.path.join(directorio, selected_txt)
204
```

Figura 18. Selección del archivo de información académica

Para situaciones donde no existían documentos específicos sobre titulación, se implementaron dos estrategias alternativas como se ve en la Figura 19. La primera verifica si solo existía un único archivo de texto en el directorio, en cuyo caso se seleccionaba automáticamente, proporcionando una solución simple cuando la base de conocimiento consistía en un solo documento. La segunda estrategia, más sofisticada, se activaba cuando existían múltiples archivos TXT.

```
205     # Si hay un solo TXT, usarlo
206     if len(txt_files) == 1:
207         if agi:
208             agi.verbose(f"Seleccionado Ã°nico archivo: {txt_files[0]}")
209         return os.path.join(directorio, txt_files[0])
210 
211     # Si hay mÃ°ltiples TXT, usar el mÃ¡s grande
212     txt_sizes = [(f, os.path.getsize(os.path.join(directorio, f))) for f in
    xt_files]
213     txt_sizes.sort(key=lambda x: x[1], reverse=True)  # Ordena por tamaÃ±o d
    scendente
214     selected_txt = txt_sizes[0][0]
215 
216     if agi:
217         agi.verbose(f"Seleccionado archivo mÃ¡s grande: {selected_txt}")
218     return os.path.join(directorio, selected_txt)
```

Figura 19. Condiciones para el archivo de texto

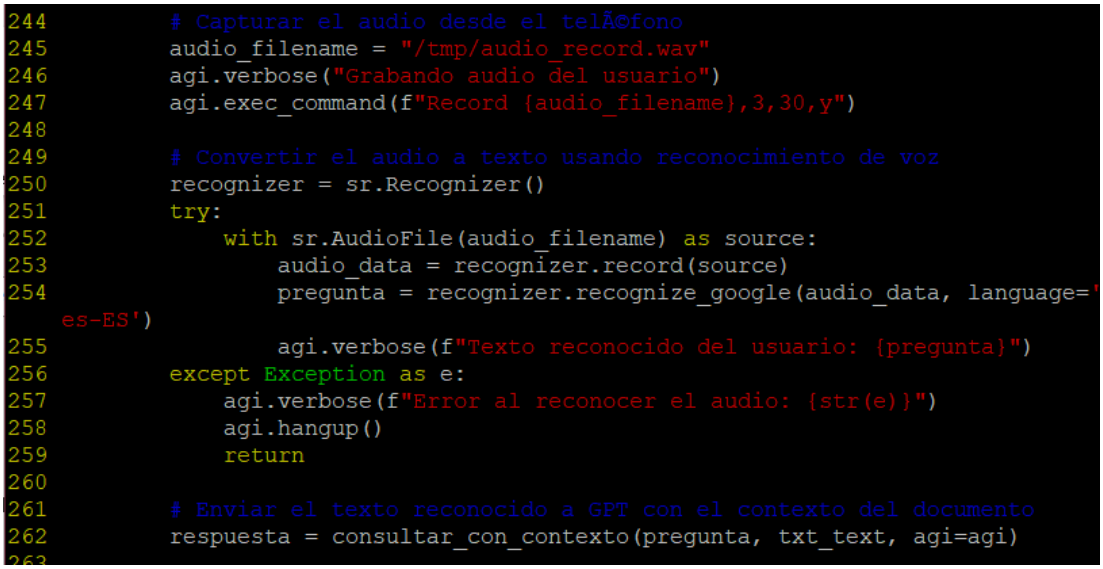
En este caso, el sistema calculó el tamaño de cada archivo y los ordenó en orden descendente mediante una función lambda, asumiendo que los documentos más extensos contendrían mayor cantidad de información útil. El sistema seleccionó entonces el archivo más grande y registró esta decisión en el log.

El código sigue con el inicio de la función principal main() que inicializa el objeto AGI y mostra un mensaje de bienvenida, indicando el inicio del sistema de asistencia académica con audio según la Figura 20. Esta estructura modular permitió que el sistema IVR inteligente adaptara dinámicamente su base de conocimiento según los documentos disponibles, sin requerir configuración manual para cada consulta.

```
220 def main():
221     agi = AGI()
222     agi.verbose("Iniciando sistema de asistencia acadÃ©mica con audio")
```

Figura 20. Inicio de la funcion def main()

Continuando de forma secuencial con el archivo, la sección de código exhibida en la Figura 21, constituye el núcleo funcional responsable de la adquisición de audio y transcripción vocal del sistema IVR inteligente. Este componente esencial posibilitó la transformación de las expresiones verbales de los usuarios en consultas textuales procesables.



```
244         # Capturar el audio desde el telÃ©fono
245         audio_filename = "/tmp/audio_record.wav"
246         agi.verbose("Grabando audio del usuario")
247         agi.exec_command(f"Record {audio_filename},3,30,y")
248
249         # Convertir el audio a texto usando reconocimiento de voz
250         recognizer = sr.Recognizer()
251         try:
252             with sr.AudioFile(audio_filename) as source:
253                 audio_data = recognizer.record(source)
254                 pregunta = recognizer.recognize_google(audio_data, language='
    es-ES')
255                 agi.verbose(f"Texto reconocido del usuario: {pregunta}")
256         except Exception as e:
257             agi.verbose(f"Error al reconocer el audio: {str(e)}")
258             agi.hangup()
259             return
260
261         # Enviar el texto reconocido a GPT con el contexto del documento
262         respuesta = consultar_con_contexto(pregunta, txt_text, agi=agi)
```

Figura 21. Configuración de audio de reconocimiento

Se desarrolló un mecanismo eficiente de grabación sonora mediante la API de Asterisk, estableciendo inicialmente la ruta `/tmp/audio_record.wav` como destino temporal para el archivo de audio. La aplicación registró el comienzo de la captura en el registro de eventos y ejecutó el comando nativo `Record` con parámetros cuidadosamente calibrados: 3 segundos de silencio para detectar la finalización natural de la consulta, una duración máxima de 30 segundos para prevenir grabaciones excesivas, que facilitó al usuario comenzar su locución sin demoras. Esta configuración logró un equilibrio óptimo entre la flexibilidad necesaria para acomodar consultas elaboradas y la eficiencia requerida para mantener la interactividad del sistema.

La traducción del contenido sonoro a formato textual se estructuró utilizando la biblioteca SpeechRecognition, aplicando específicamente el motor de reconocimiento de Google optimizado para español peninsular (`es-ES`), el cual evidenció excelente compatibilidad con las variantes dialectales ecuatorianas durante la fase de pruebas.

El script incorporó un manejo sofisticado de excepciones para gestionar posibles anomalías en el proceso de reconocimiento, garantizando así que la plataforma finaliza las llamadas de manera controlada ante eventuales dificultades técnicas, evitando bloqueos o experiencias degradadas para los usuarios.

Entonces, tras obtener exitosamente la transcripción de la consulta vocal, la solución canalizó este texto hacia la función `consultar_con_contexto()` junto con el contenido del documento de conocimiento previamente seleccionado. Esta arquitectura modular facilitó una clara separación de responsabilidades: adquisición sonora, transcripción vocal y posterior análisis semántico, simplificando considerablemente las tareas de mantenimiento y futuras expansiones de la plataforma IVR inteligente desplegada.

La sección de código visualizada en la Figura 22, ilustra el componente final del script AGI encargado de entregar la respuesta auditiva al usuario mediante el sistema telefónico. Este segmento representó la culminación del proceso conversacional, transformando el texto generado por la IA en una experiencia auditiva completa.

```
273         # Generar la respuesta en audio usando Azure REST API
274         if generate_voice(respuesta, wav_file, agi):
275             # Reproducir el mensaje generado
276             try:
277                 agi.verbose("Reproduciendo respuesta")
278                 agi.stream_file('/var/lib/asterisk/sounds/respuesta_gpt')
    Sin extensiÃ³n
279                 agi.verbose("Respuesta reproducida exitosamente")
280             except Exception as e:
281                 agi.verbose(f"Error al reproducir: {str(e)}")
282         else:
283             agi.verbose("No se pudo generar el archivo de audio")
284
285     except Exception as e:
286         agi.verbose(f"Error general: {str(e)}")
287     finally:
288         # Asegurar que la llamada termine correctamente
289         agi.hangup()
```

Figura 22. Configuración el agi.verbose para las respuestas de audio

El flujo de ejecución comenzó con la invocación de la función generate_voice(), que recibe como parámetros el texto de respuesta, la ruta del archivo WAV y el objeto AGI para registro de eventos. Esta función, desarrollada previamente, se ocupó de realizar la conexión con la API REST de Azure para sintetizar la voz natural. La estructura condicional subsiguiente estableció dos caminos distintos según el éxito de la generación vocal.

En caso de síntesis exitosa, el sistema procedió a reproducir el mensaje mediante el método stream_file() de la API de Asterisk, apuntando hacia la

ubicación /var/lib/asterisk/sounds/respuesta_gpt (sin extensión, conforme a las convenciones de Asterisk).

El proceso incluyó notificaciones detalladas en el registro del sistema tanto al inicio como al finalizar la reproducción, facilitando así la monitorización y depuración del sistema en producción. Paralelamente, se construyó un robusto sistema de manejo de excepciones para capturar posibles fallos durante la reproducción, garantizando que el usuario recibiera información sobre cualquier inconveniente técnico.

La arquitectura del código prestó especial atención a la finalización adecuada de las llamadas mediante un bloque finally que aseguró la ejecución del método hangup() independientemente del resultado de las operaciones previas. Esta práctica resultó fundamental para liberar recursos del sistema telefónico y evitar llamadas "fantasma" que pudieran permanecer activas indefinidamente.

La estructura multicapa de control de excepciones evidenció un diseño orientado a la robustez operacional, contemplando tanto errores específicos de reproducción como fallos generales que pudieran surgir durante la interacción con el usuario, elementos críticos para un servicio telefónico automatizado que funcioné de manera confiable en entornos de producción.

### 3.8.2 Asignación del dialplan PJSIP para las extensiones

La Figura 23 corresponde a la sección de configuración de transporte SIP del archivo pjsip.conf ubicado en /etc/asterisk/pjsip.conf y detallado en el Anexo A. Esta porción del código constituyó un elemento crítico para establecer las bases de comunicación del sistema IVR inteligente, definiendo cómo Asterisk gestionaría el tráfico de señalización SIP.

```
[transport-udp]
type=transport
protocol=udp
bind=0.0.0.0
external_media_address=172.178.112.41  ; Reemplaza con tu IP pública
external_signaling_address=172.178.112.41  ; Reemplaza con tu IP pública
local_net=10.0.0.0/8
local_net=172.16.0.0/12
local_net=192.168.0.0/16
```

Figura 23. Estructura del contexto [transport-udp]

El bloque de configuración se inició con la sección etiquetada [transport-udp], estableciendo un transporte basado en el protocolo UDP, opción preferida por su menor sobrecarga en comparación con TCP. La directiva type=transport especificó la naturaleza del bloque como una definición de transporte, mientras que protocol=udp confirmó el uso específico del protocolo UDP para la comunicación SIP.

La directiva bind=0.0.0.0 resultó particularmente significativa, ya que instruyó al servidor Asterisk para escuchar conexiones entrantes en todas las interfaces de red disponibles, maximizando así la accesibilidad del servicio.

La configuración abordó aspectos cruciales de conectividad externa mediante los parámetros external_media_address y external_signaling_address, ambos apuntando a la dirección IP pública 172.178.112.41. Estos valores fueron fundamentales para el funcionamiento adecuado del sistema cuando se accedía desde redes externas, permitiendo que los dispositivos remotos establecieran correctamente los canales de comunicación tanto para señalización SIP como para transmisión de audio.

La última sección del fragmento definió tres rangos de redes locales mediante directivas local_net múltiples: una red privada estándar 10.0.0.0/8, una red 172.16.0.0/12, y una red doméstica típica 192.168.0.0/16. Esta configuración permite que Asterisk distinguiera correctamente entre tráfico interno y externo, aplicando las transformaciones NAT adecuadas según el origen de las conexiones. Esta estructuración de redes resultó esencial para garantizar la comunicación bidireccional efectiva tanto para usuarios dentro de la red universitaria como para aquellos que accedían desde ubicaciones remotas a través de Internet.

Según lo visualizado en la Figura 24, corresponde a dos secciones críticas del archivo pjsip.conf que definen tanto un endpoint SIP como su mecanismo de autenticación asociado. Estas configuraciones establecieron los parámetros fundamentales para la comunicación entre el sistema IVR inteligente y los dispositivos telefónicos.

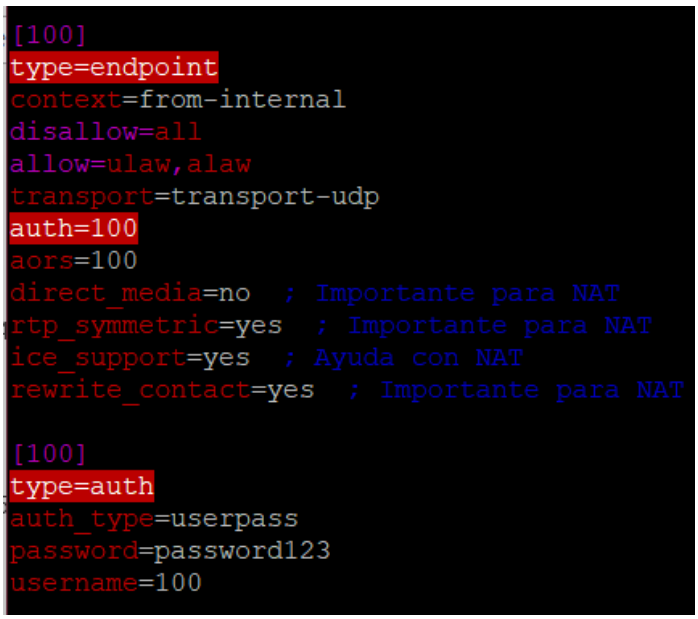



Figura 24. Estructura general para la extensión 100

La primera sección, identificada como [100], configuró un endpoint SIP que representó una extensión telefónica dentro del sistema. La directiva type=endpoint especificó la naturaleza de esta entidad, mientras que context=from-internal definió el punto de entrada al dialplan de Asterisk para llamadas originadas desde este endpoint, dirigiéndose al contexto apropiado donde se encontraban las aplicaciones AGI desarrolladas. Las directivas de seguridad implementadas incluyeron disallow=all y allow=ulaw,alaw, estableciendo una política restrictiva que habilitó únicamente los códecs de audio G.711 μ-law y A-law, optimizando así la calidad de voz para el reconocimiento automático del habla.

Se implementaron múltiples parámetros orientados específicamente a la gestión de NAT (Network Address Translation), elemento crucial para asegurar la conectividad bidireccional en entornos de red complejos. La combinación de direct_media=no, rtp_symmetric=yes, force_rport=yes y rewrite_contact=yes con stituyó una configuración exhaustiva para atravesar correctamente escenarios de NAT,

como lo indican claramente los comentarios en el código. La opción transport=transport-udp vinculó este endpoint con la configuración de transporte UDP previamente definida, completando así la pila de protocolos necesaria.

La segunda sección, también etiquetada como [100], estableció el mecanismo de autenticación referenciado por el endpoint mediante auth=100. Se configuró como type=auth con auth_type=userpass, implementando un esquema de autenticación básico basado en nombre de usuario y contraseña.

El nombre de usuario se definió como "100" para coincidir con la extensión, mientras que se asignó una contraseña alfanumérica ("password123"), que en un entorno productivo debería sustituirse por una credencial para prevenir accesos no autorizados.

La Figura 25 muestra la configuración completa de otra extensión SIP en el archivo pjsip.conf, implementando una estructura tripartita que define todos los aspectos necesarios para el funcionamiento de la extensión 200 dentro del sistema IVR inteligente.

```
[200]
type=endpoint
context=from-internal
disallow=all
allow=ulaw,alaw
transport=transport-udp
auth=200
aors=200
direct_media=no  ; Importante para NAT
rtp_symmetric=yes  ; Importante para NAT
ice_support=yes  ; Ayuda con NAT
rewrite_contact=yes  ; Importante para NAT

[200]
type=auth
auth_type=userpass
password=password123
username=200

[200]
type=aor
max_contacts=5
qualify_frequency=30  ; Mantiene la conexión activa
```

Figura 25. Configuración para la extensión 200

La primera sección, etiquetada como [200], estableció un endpoint SIP con configuración paralela a la observada anteriormente. Este endpoint se vinculó al contexto "from-internal", garantizando que todas las llamadas originadas desde esta

extensión accedieran al mismo conjunto de aplicaciones AGI que la extensión 100. Se mantuvieron idénticas políticas de códecs, priorizando los formatos G.711 para asegurar compatibilidad óptima con el sistema de reconocimiento de voz. Los parámetros relacionados con NAT (direct_media=no, rtp_symmetric=yes, ice_support=yes y rewrite_contact=yes) reflejaron la misma estrategia robusta para atravesar redes enmascaradas, evidenciando un enfoque consistente en toda la plataforma.

La segunda sección, también etiquetada [200] pero con type=auth, implementó el esquema de autenticación específico para esta extensión. Se configuró el mismo mecanismo de usuario/contraseña visto anteriormente, manteniendo la coherencia en las políticas de seguridad del sistema.

La alineación del identificador de usuario ("200") con el número de extensión simplificó la administración del sistema, aunque la reutilización del mismo patrón de contraseña ("password123") sugiere la necesidad de una revisión de las prácticas de seguridad en un entorno productivo.

La tercera sección introdujo un componente adicional no presente en la configuración anterior: un registro AOR (Address of Record) con type=aor. Este elemento definió parámetros críticos para el registro y la accesibilidad del dispositivo, estableciendo max_contacts=5 para permitir hasta cinco registros simultáneos desde diferentes dispositivos o ubicaciones para esta misma extensión.

El parámetro qualify_frequency=30 configuró verificaciones de disponibilidad cada 30 segundos, como señala el comentario adjunto "Mantiene la conexión activa". Este mecanismo resultó esencial para detectar rápidamente dispositivos desconectados y garantizar que el estado de presencia se mantuviera actualizado, mejorando la fiabilidad en entornos donde los agentes pueden desplazarse entre ubicaciones manteniendo su identidad SIP.

Esta configuración dual de endpoints y autenticación proporcionó la infraestructura necesaria para que los clientes SIP establecieran sesiones seguras con el servidor Asterisk, permitiendo el acceso al sistema IVR inteligente con Azure OpenAI mientras se mantenía un nivel adecuado de control de acceso y compatibilidad con escenarios

de red diversos. Además, si se desea añadir otras extensiones a futuro, se podrá aplicar las configuraciones de estas dos extensiones manteniendo escalabilidad y operabilidad sin ningún problema alguno.

### 3.8.3 Instauración del archivo de configuración extensions.conf

La Figura 26 mostrada corresponde a la sección inicial del archivo extensions.conf ubicado en /etc/asterisk/, que estableció los parámetros globales para el comportamiento del dialplan en el sistema IVR inteligente. Esta configuración fundamental actuó como el esqueleto que orquestó el flujo de llamadas hacia los scripts AGI previamente analizados. Para mayor claridad, en el

Anexo B se presenta el contenido completo del archivo.

```
[general]
static=yes
writeprotect=no
autofallthrough=yes
priorityjumping=no
extenpatternmatchnew=yes
```

Figura 26. Contexto default de la parte de extensions.conf

La directiva static=yes determinó que el dialplan permaneciera en memoria durante toda la ejecución del servidor Asterisk, evitando la necesidad de recargas frecuentes y optimizando el rendimiento del sistema en entornos de alta demanda como el servicio de asistencia académica. Complementariamente, writeprotect=no otorgó a los scripts AGI y otras aplicaciones la capacidad de modificar dinámicamente elementos del dialplan en tiempo de ejecución, habilitando funcionalidades adaptativas como la selección dinámica de documentos informativos.

Se activó autofallthrough=yes, instruyendo a Asterisk para continuar automáticamente con la siguiente prioridad cuando una aplicación concluye sin un destino explícito. Esta característica resultó esencial para la fluidez del diálogo con los usuarios, permitiendo encadenar naturalmente los componentes de reconocimiento de voz, procesamiento con Azure OpenAI y síntesis de respuestas. La directiva priorityjumping=no desactivó el comportamiento de saltos de prioridad obsoleto, alineando la configuración con las mejores prácticas modernas de desarrollo para Asterisk.

La inclusión de extenpatternmatchnew=yes habilitó el nuevo algoritmo de coincidencia de patrones, proporcionando mayor flexibilidad en la definición de rutas de llamada y mejorando la capacidad del sistema para gestionar diversos escenarios de entrada. Esta característica facilita la implementación de reglas sofisticadas para dirigir las llamadas hacia el script AGI principal con Azure OpenAI, dependiendo de factores como el origen de la llamada o el horario.

Esta base de configuración dialplan sentó los cimientos sobre los que se construyeron posteriormente los contextos específicos para el sistema de asistencia académica,

incluyendo los puntos de entrada que activaban los scripts de procesamiento de lenguaje natural y síntesis de voz analizados previamente.

La Figura 27 corresponde a la sección [from-internal] del archivo extensions.conf, que implementó el enrutamiento de llamadas internas dentro del sistema IVR inteligente. Esta configuración constituyó el punto de entrada mencionado en las definiciones de endpoints SIP analizadas anteriormente, estableciendo las reglas precisas para el manejo de comunicaciones entre extensiones.

```
[from-internal]
; Direct calls between extensions
exten => _XXX,1,Answer()
 same => n,Dial(PJSIP/${EXTEN},30)
 same => n,Hangup()

; Specific handling for extension 100
exten => 100,1,Answer()
 same => n,Dial(PJSIP/100,30)
 same => n,Hangup()

; Specific handling for extension 200
exten => 200,1,Answer()
 same => n,Dial(PJSIP/200,30)
 same => n,Hangup()
```

Figura 27. Contexto para la configuración de las extensiones

La primera sección estableció un patrón genérico para comunicaciones directas entre extensiones mediante la directiva exten => _XXX,1,Answer(). El uso del patrón _XXX permitió capturar cualquier extensión de tres dígitos, iniciando la secuencia con la contestación automática de la llamada. Las siguientes directivas utilizaron la palabra clave same para mantener la concisión del código, estableciendo un flujo lógico donde se intentaba conectar con la extensión marcada mediante Dial(PJSIP/${EXTEN},30) con un tiempo límite de 30 segundos, tras lo cual se ejecutaba Hangup() si no se establecía la conexión.

A continuación, se configuró dos secciones especializadas para las extensiones 100 y 200, previamente definidas en el archivo pjsip.conf que se detalla en el Anexo A. Ambas secciones siguieron un patrón idéntico de procesamiento: contestación

inmediata mediante Answer(), intento de conexión específico a cada extensión SIP, y finalización controlada con Hangup().

Esta estructuración individual permitió un control granular sobre el comportamiento de cada extensión, facilitando la implementación de lógicas personalizadas según las necesidades particulares de cada punto de acceso.

Esta configuración de dialplan actuó como la capa de orquestación que conectó la infraestructura SIP con las aplicaciones AGI desarrolladas, permitiendo que las llamadas entrantes accedieron al sistema conversacional basado en Azure OpenAI a través de rutas claramente definidas. La coherencia entre los diferentes componentes de configuración (transporte SIP, endpoints y dialplan) evidenció un diseño arquitectónico sólido que garantizó la correcta integración de todas las piezas del sistema IVR inteligente.

Figura 28, complementa la sección [from-internal] analizada previamente, incorporando extensiones especializadas que cumplieron funciones específicas dentro del ecosistema del sistema IVR inteligente.

```
; Echo test for troubleshooting
exten => 500,1,Answer()
 same => n,Echo()
 same => n,Hangup()

exten => 811,1,Answer()
 same => n,Dial(PJSIP/811,30)
 same => n,Hangup()

exten => 250,1,Answer()
exten => 250,2,Wait(1)
exten => 250,3,Playback(digame)
;exten => 250,4,AGI(marychat.py)
exten => 250,4,AGI(tesis.py)
exten => 250,5,NoOp(Text: ${SPEECH_TEXT})
exten => 250,6,Hangup()
exten => 250,1,answer()
```

Figura 28. Extensiones adicionales y la 250, especializada para consultas

La extensión 500 se configuró como un mecanismo de diagnóstico mediante la implementación de un sencillo bucle de eco. Esta herramienta técnica resultó invaluable durante la fase de despliegue para verificar la calidad de la transmisión de audio, permitiendo a los administradores comprobar rápidamente problemas de latencia, pérdida de paquetes o calidad del audio sin necesidad de involucrar los componentes más complejos del sistema.

La secuencia Answer(), Echo(), Hangup() proporcionó un método directo para validar la integridad de los canales de comunicación.

La extensión 811 se estableció como un punto de redirección hacia otro endpoint SIP, actuando como un puente hacia servicios complementarios o como respaldo para ciertos escenarios. La directiva Dial(PJSIP/811,30) permite que otra extensión de prueba pueda usar las capacidades del IVR principal, tanto para análisis, testeo y comportamiento del mismo.

La extensión 250 constituyó el corazón operativo del sistema de asistencia académica, implementando una secuencia sofisticada de pasos que orquestaron la experiencia conversacional con el usuario. Tras contestar la llamada, se insertó una pausa estratégica de un segundo mediante Wait(1), seguida de la reproducción del mensaje de bienvenida "digame" que invitó al usuario a formular su consulta. La presencia de dos opciones diferentes para la ejecución del script AGI (marychat.py y tesis.py) evidenció la fase de transición del desarrollo, donde el sistema evolucionó desde versiones preliminares hasta la implementación final elegida.

La directiva NoOp(Text: ${SPEECH_TEXT}) implementó un mecanismo de registro que capturó el texto reconocido en los logs de Asterisk, facilitando la depuración y el análisis posterior de las interacciones. Esta práctica resultó extremadamente valiosa durante la fase de afinamiento del sistema para identificar patrones de consulta y optimizar las respuestas del modelo GPT-4. La secuencia completa demostró una implementación cuidadosamente estructurada que guió al usuario a través de una experiencia conversacional fluida mientras mantuvo herramientas robustas para el diagnóstico y mejora continua del servicio.

### 3.8.4 Ajustes de las extensiones para los equipos finales

La Figura 29 capturada muestra la implementación exitosa del cliente MicroSIP configurado para interactuar con el sistema IVR inteligente a través de la extensión 200. Esta configuración representó la materialización del punto final de comunicación que permitió probar y utilizar el sistema conversacional desarrollado desde una interfaz de usuario amigable.

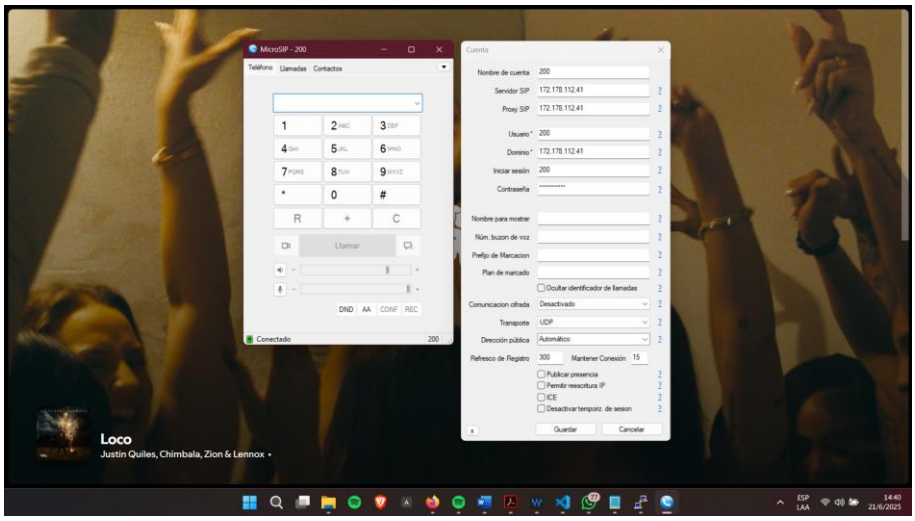



Figura 29. Configuración del cliente 200 en MicroSIP

El proceso de configuración del softphone MicroSIP se completó mediante el establecimiento preciso de los parámetros de conexión que coincidieron con las definiciones creadas previamente en pjsip.conf. En la ventana de configuración se observa cómo se estableció el nombre de cuenta "200", correspondiendo exactamente con el identificador de extensión definido en el servidor Asterisk. La dirección IP 172.178.112.41 se configuró tanto para el servidor SIP principal como para el proxy SIP, reflejando la ubicación de la instancia de Asterisk desplegada en Azure.

Las credenciales de autenticación implementadas mantuvieron la coherencia con la configuración del servidor, utilizando "200" tanto como nombre de usuario como para inicio de sesión, alineándose con el esquema de autenticación auth_type=userpass previamente analizado.

El protocolo de transporte seleccionado fue UDP, coincidiendo con la configuración [transport-udp] establecida en el servidor, optimizando así la comunicación para entornos de telefonía en tiempo real.

El valor de "Refresco de Registro" se estableció en 300 segundos, complementando eficientemente la configuración qualify_frequency=30 del servidor, lo que permitió mantener una conexión estable mientras se optimiza el tráfico de red. El estado "Conectado" visible en la parte inferior de la interfaz confirmó el establecimiento exitoso de la comunicación con el sistema IVR, quedando el cliente listo para realizar llamadas hacia las extensiones del sistema.

Se reveló el despliegue del sistema IVR inteligente, mostrado en la Figura 30, del cual la configuración de un cliente SIP móvil conectado a la extensión 811. Esta implementación complementó la solución basada en MicroSIP previamente analizada, expandiendo el acceso al sistema desde dispositivos móviles y proporcionando mayor flexibilidad para los usuarios finales.




Figura 30. Configuración de la extensión 811 en Zoiper App

La configuración del cliente móvil se realizó con precisión técnica, estableciendo la cuenta SIP con el identificador "811@172.178.112.41", que integró tanto el número

de extensión como la dirección IP del servidor Asterisk desplegado en Azure. Este formato siguió las convenciones estándar de direccionamiento SIP, facilitando el enrutamiento correcto de llamadas dentro del ecosistema telefónico institucional.

Los parámetros de autenticación se configuraron alineados con la definición del endpoint en el archivo pjsip.conf, utilizando "811" como nombre de usuario y una contraseña específica oculta en la interfaz. La dirección del host apuntó directamente al servidor Asterisk (172.178.112.41), estableciendo una conexión directa sin intermediarios adicionales que pudieran introducir latencia en las comunicaciones.

Esta implementación móvil representó una extensión estratégica del sistema, coincidiendo con la configuración previamente identificada en el dialplan donde se observó la entrada exten => 811,1,Answer(). La capacidad de acceder al sistema desde dispositivos móviles proporciona una ventaja significativa para el personal administrativo, estudiantil y docente, permitiéndoles interactuar con el asistente académico basado en Azure OpenAI desde cualquier ubicación con conectividad a internet.

La coherencia entre las configuraciones de clientes de escritorio y móviles demostró el enfoque integral adoptado en el despliegue del sistema IVR inteligente, creando un ecosistema de comunicación unificado que atendió a las necesidades de diversos tipos de usuarios y escenarios de uso, desde consultas en oficinas hasta atención remota.

### 3.9 Comunicación Entre Componentes e Integración

El sistema implementó una arquitectura de microservicios interconectados mediante interfaces API bien definidas, donde cada componente cumple una función específica en el procesamiento de llamadas.

***a. Asterisk ↔ Dispositivos Telefónicos:***

- *Comunicación bidireccional mediante protocolo PJSIP/SIP para señalización*
- *Transmisión de audio mediante RTP (Real-time Transport Protocol)*
- *Negociación de códecs (G.711 μ-law/A-law) para optimizar calidad de audio*

***b. Asterisk ↔ Scripts AGI:***

- *Comunicación mediante interfaz AGI (Asterisk Gateway Interface)*
- *Asterisk ejecuta scripts externos cuando una llamada alcanza puntos específicos del dialplan*
- *Transferencia de datos de la llamada (caller ID, extensión marcada, audio capturado)*
- *Scripts devuelven instrucciones que Asterisk ejecuta (reproducir audio, transferir llamada)*

***c. Scripts AGI ↔ Azure AI Speech Services:***

- *Comunicación mediante API REST autenticada con claves de suscripción*
- *Envío de audio capturado para transcripción (STT)*
- *Envío de texto para síntesis de voz (TTS) utilizando SSML (Speech Synthesis Markup Language)*
- *Recepción de respuestas en formato JSON para transcripciones y archivos WAV para audio sintetizado*

***d. Scripts AGI ↔ Azure OpenAI Service:***

- *Comunicación mediante API REST con autenticación OAuth 2.0*
- *Estructura de mensajes que incluye contexto del documento institucional y consulta transcrita*
- *Parametrización específica (temperature=0.5, top_p=1.0) para balancear precisión y naturalidad*
- *Sistema de reintentos con retroceso exponencial para manejar limitaciones de tasa de la API*

***e. Procesamiento Intermedio:***

- *Conversión de formatos de audio mediante FFmpeg (24kHz a 8kHz) para compatibilidad con Asterisk*
- *Selección dinámica de documentos relevantes según la consulta del usuario*
- *Almacenamiento temporal de archivos de audio y transcripciones para diagnóstico y depuración*

## 3.10 Testeo de los diferentes parámetros de salida

A continuación, se documenta la fase crítica de prueba operativa del sistema IVR inteligente, mostrando la comunicación bidireccional entre los clientes SIP configurados previamente. Esta etapa de validación resultó fundamental para confirmar la correcta integración y funcionalidad del ecosistema completo antes del despliegue final.

En la Figura 31 se observa la implementación de una llamada iniciada desde el cliente MicroSIP con extensión 200 hacia la extensión móvil 811. El panel izquierdo muestra la interfaz del softphone de escritorio con el número "811" ingresado en el campo de marcación, mientras el botón "Colgar" aparece activado en rojo, indicando una llamada en curso. Simultáneamente, en el panel derecho, el dispositivo móvil muestra la pantalla de llamada entrante con el identificador "Entrante mediante 811@172.178.112.41: 200 (200)", confirmando la correcta transmisión de la información de identificación del llamante a través de la red SIP.

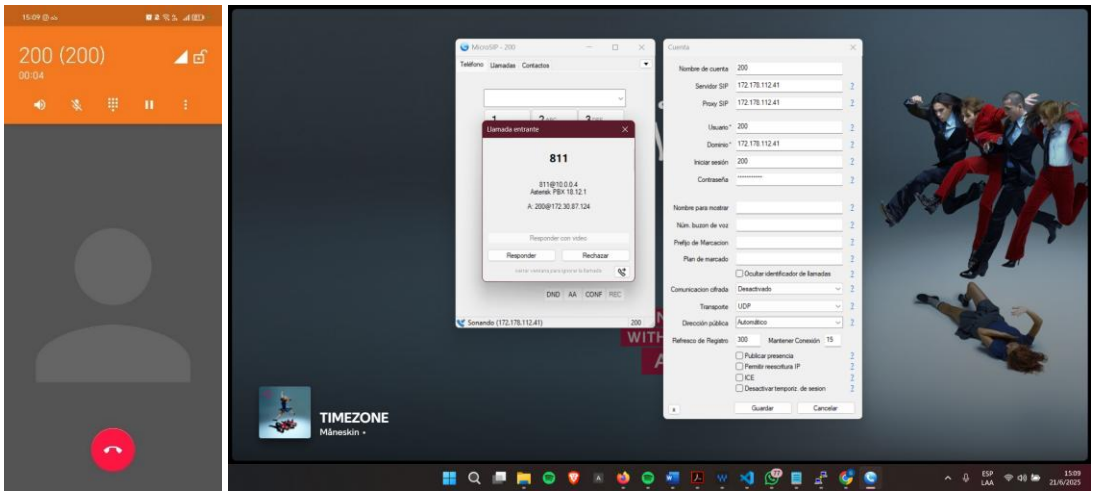



Figura 31. Llamada saliente desde la extensión 200

La  Figura 32 documenta el escenario inverso, estableciendo una comunicación en sentido contrario.

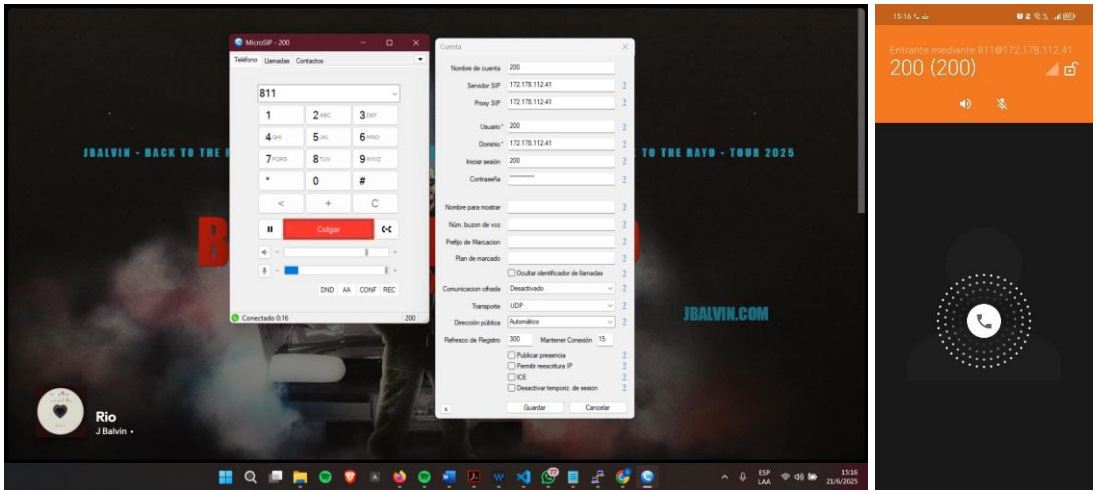



Figura 32. Llamada saliente desde la extensión 811

En el panel izquierdo se visualiza el cliente móvil durante una llamada activa con la extensión "200 (200)" y un temporizador marcando "00:04", evidenciando que la conexión se ha establecido exitosamente. Complementariamente, el panel derecho muestra la interfaz de MicroSIP recibiendo la llamada entrante, donde se puede apreciar el diálogo de notificación con el identificador "811" claramente visible. El estado "Sonando (172.178.112.41)" confirma que el servidor Asterisk está procesando correctamente la señalización SIP entre ambos endpoints.

Esta validación bidireccional verificó múltiples aspectos técnicos del sistema: el correcto enrutamiento de llamadas según la configuración del dialplan analizado anteriormente, la efectividad de los parámetros NAT implementados en pjsip.conf, y la adecuada resolución de los identificadores de llamadas entre clientes.

La capacidad demostrada para establecer comunicaciones en ambas direcciones constituyó un indicador determinante de la robustez del sistema implementado, confirmando su preparación para soportar las conversaciones entre usuarios y el asistente académico basado en Azure OpenAI a través de la extensión especializada 250.

## 3.11 Análisis de la operadora inteligente

### 3.11.1 Análisis del módulo de síntesis de voz con Azure AI Speech Services

El componente fundamental de síntesis de voz del sistema IVR inteligente de la Voz de Azure, es el encargado de transformar las respuestas textuales generadas por Azure OpenAI en locuciones naturales para los usuarios. Este script de prueba inicial sentó las bases para la integración posterior del servicio de voz en el entorno de producción.

La arquitectura del módulo se fundamentó en la biblioteca oficial de Azure Cognitive Services Speech, estableciendo una comunicación segura con los servicios en la nube de Microsoft. Los elementos centrales de configuración se incluyen en la Figura 33.

```
speech_config = SpeechConfig(subscription=speech_key, region=speech_region)
speech_config.speech_synthesis_voice_name = "es-ES-ElviraNeural"
audio_config = AudioConfig(filename=output_file)
synthesizer = SpeechSynthesizer(speech_config=speech_config, audio_config=audio_config)
```

Figura 33. Estructura general del servicio de Voz AI

Esta construcción modular permite especificar parámetros cruciales como la voz "es-ES-ElviraNeural", seleccionada específicamente por su naturalidad y claridad en la pronunciación del español peninsular, característica que mejoró significativamente la comprensión de las respuestas por parte del usuario. La selección de esta voz resultó de un proceso de evaluación comparativa entre diferentes opciones disponibles en el catálogo de Azure.

El núcleo funcional del módulo que se observa en la Figura 34, residió en el método speak_text_async(), que implementó el procesamiento asíncrono del texto a voz, optimizando así el rendimiento del sistema y evitando bloqueos durante la síntesis.

```
result = synthesizer.speak_text_async(texto).get()
if result.reason == ResultReason.SynthesizingAudioCompleted:
    print(f"Texto convertido exitosamente a voz.")
    print(f"Archivo guardado en: {output_file}")
    return True
```

Figura 34. Condición si la conversión es exitosa

La robustez operativa según la Figura 35, garantizó mediante un sofisticado sistema de manejo de excepciones que capturó potenciales fallos durante la conexión con Azure o el procesamiento del audio, proporcionando información diagnóstica precisa y previniendo interrupciones abruptas del flujo de llamada:

```
except Exception as ex:
    print(f"Error durante la conversión de texto a voz: {str(ex)}")
    return False
```

Figura 35. Error si la conversión llegase a fracasar

Este componente de prueba validó exitosamente la viabilidad técnica de integrar Azure AI Speech Services con el ecosistema Asterisk, confirmando tiempos de respuesta adecuados para mantener una experiencia conversacional fluida.

Los resultados obtenidos durante esta fase experimental fundamentaron la implementación definitiva de la función generate_voice() analizada previamente, que se encargó de sintetizar las respuestas en el entorno de producción, cerrando así el ciclo completo de interacción hablada con los usuarios del sistema IVR inteligente.

### 3.11.2 Análisis del componente de procesamiento de lenguaje natural con Azure OpenAI

Implementando el componente de procesamiento de lenguaje natural GPT, permitió ofrecer respuestas contextuales precisas basadas en documentos académicos. Este módulo constituyó el cerebro conversacional de la solución, transformando las consultas de voz procesadas en respuestas significativas.

La integración con Azure OpenAI se estableció mediante el cliente oficial de inferencia de Azure, configurando una conexión segura con el modelo GPT-4 desplegado en la nube mediante los comandos de la Figura 36.

```
client = ChatCompletionsClient(
    endpoint=endpoint,
    credential=AzureKeyCredential("GEgSzcywZfrCDXlO1M9bUTUTQXB9h2v3mYFu0eYlibsZPpKGmyDeJQQJ99BCACHYHv6XJ
)
```

Figura 36. Llamado de la API hacia el script

El sistema implementó un sofisticado mecanismo de gestión documental a través de la función seleccionar_txt_auto(), que incorpora una lógica jerárquica para identificar automáticamente el archivo de conocimiento más relevante. Esta funcionalidad prioriza documentos relacionados con procesos de titulación y, en ausencia de coincidencias específicas, aplicó una heurística basada en el tamaño del archivo, asumiendo que el documento más extenso contenía la información más completa.

La función consultar_con_contexto() constituyó el componente medular del procesamiento lingüístico, implementando una estructura robusta para manejar la comunicación con Azure OpenAI:

```
response = client.complete(
    messages=[
        SystemMessage(content="Eres un asistente especializado que analiza documentos de texto y respond
        UserMessage(content=prompt),
    ],
    max_tokens=1024,
    temperature=0.5,
    top_p=1.0,
    model=model_name
)
```

Figura 37. Procesamiento lingüístico

Esta configuración emplea un enfoque equilibrado con temperature=0.5, optimizando la generación de respuestas para mantener la precisión factual mientras se preservaba cierta naturalidad en el lenguaje. El parámetro top_p=1.0 aseguró una diversidad adecuada en las respuestas, contribuyendo a la percepción de una conversación más humana durante las interacciones telefónicas.

El sistema incorpora un mecanismo de resiliencia ante limitaciones de la API mediante un algoritmo de reintentos con retroceso exponencial, incrementando dinámicamente los tiempos de espera para adaptarse a situaciones de alta carga:

```
if "429" in error_msg and intentos < max_intentos:
    print(f"\nLímite de tasa excedido. Esperando {tiempo_espera} segundos antes de reintentar...")
    time.sleep(tiempo_espera)
    tiempo_espera *= 1.5  # Aumentar el tiempo de espera para el próximo reintento
```

Figura 38. Mecanismo ante limitaciones de la API

Esta implementación permitió al sistema IVR analizar automáticamente documentos institucionales como reglamentos de titulación, calendarios académicos y procedimientos administrativos, proporcionando respuestas contextualizadas a las consultas de los estudiantes. La integración de este componente con el módulo de reconocimiento de voz y síntesis vocal analizados anteriormente completó el ciclo completo de interacción natural, permitiendo a los usuarios obtener información académica precisa a través de simples llamadas telefónicas.

### 3.11.3 Análisis del proceso de atención automatizada con Asterisk y AGI

La Figura 39 muestra la traza de una llamada gestionada por sistema desarrollado en Asterisk, que integra un script AGI (tesis.py) para asistencia académica mediante procesamiento de voz y generación de respuestas automáticas.

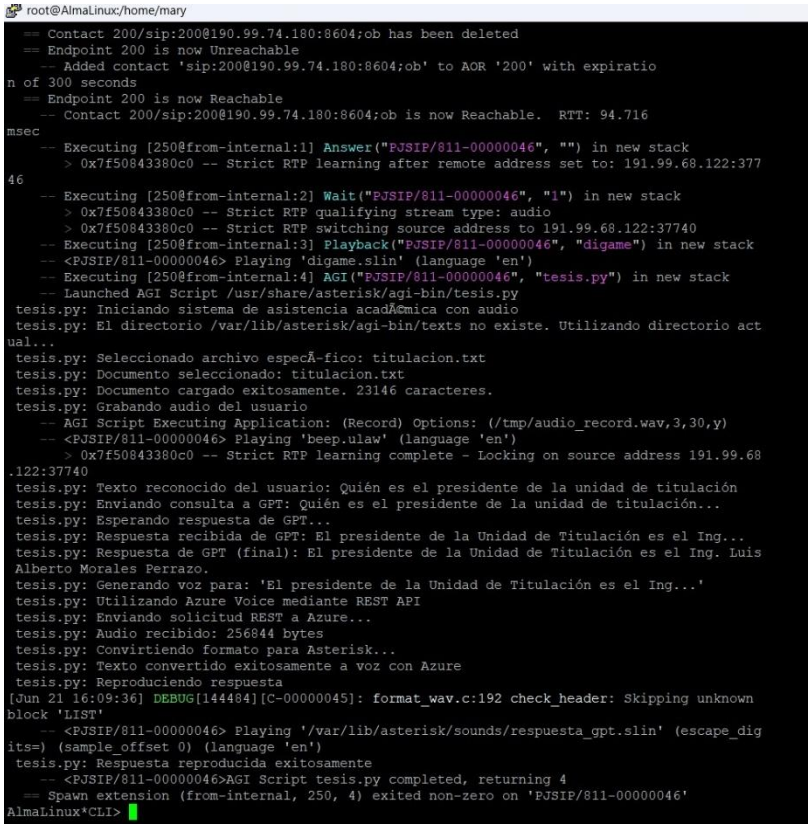



Figura 39. Ejecución del script AGI con consulta y respuesta correcta del sistema IVR.

### *a. Establecimiento de la llamada:*

- La extensión 200 se conecta mediante el protocolo **PJSIP**, indicando que el endpoint se vuelve *reachable* después de verificar la conectividad RTP.

- El canal de llamada ejecuta la aplicación Answer para responder la llamada entrante.

### *b. Progreso de la interacción:*

- El sistema reproduce un mensaje inicial (digame.slin) solicitando la consulta del usuario.

- Posteriormente, se lanza el script tesis.py mediante la aplicación AGI, iniciando el sistema de asistencia.

***c. Grabación y procesamiento:***

- El script selecciona un archivo de preguntas (titulacion.txt) y carga correctamente su contenido.
- Se graba el audio del usuario, que se almacena en un archivo WAV temporal.
- El audio se convierte a texto mediante un motor de **Speech-to-Text**. En este caso, el texto reconocido fue: *"¿Quién es el presidente de la unidad de titulación?"*

***d. Consulta al modelo GPT:***

- El script envía la pregunta al motor GPT para generar una respuesta coherente y precisa.
- La respuesta generada por GPT es correcta y coincide con el dato esperado: *"El presidente de la Unidad de Titulación es el Ing. Luis Alberto Morales Perrazo."*

***e. Conversión de texto a voz:***

- La respuesta se convierte a audio mediante **Azure Voice** (REST API de síntesis de voz).
- El archivo de audio generado es reproducido de vuelta al usuario (respuesta_gpt.slin).

***f. Finalización:***

- El proceso AGI se completa correctamente (completed, returning 4), indicando que el script terminó sin errores críticos.
- La extensión se libera después de enviar la respuesta.

En la Figura 40, aunque el sistema técnico funcionó correctamente al reconocer la voz, consultar a GPT y reproducir la respuesta sin errores, se evidenció un desajuste entre la pregunta realizada y la respuesta generada, ya que el usuario solicitó detalles sobre el informe que debe entregar el tutor asignado y la IA respondió de forma muy general,

limitándose a indicar la fecha de entrega. Esto muestra que, si bien no hubo fallos en la grabación ni pérdida de datos, es necesario mejorar la precisión del contexto enviado a GPT para evitar respuestas ambiguas o incompletas y garantizar que la información entregada al usuario sea clara y específica.

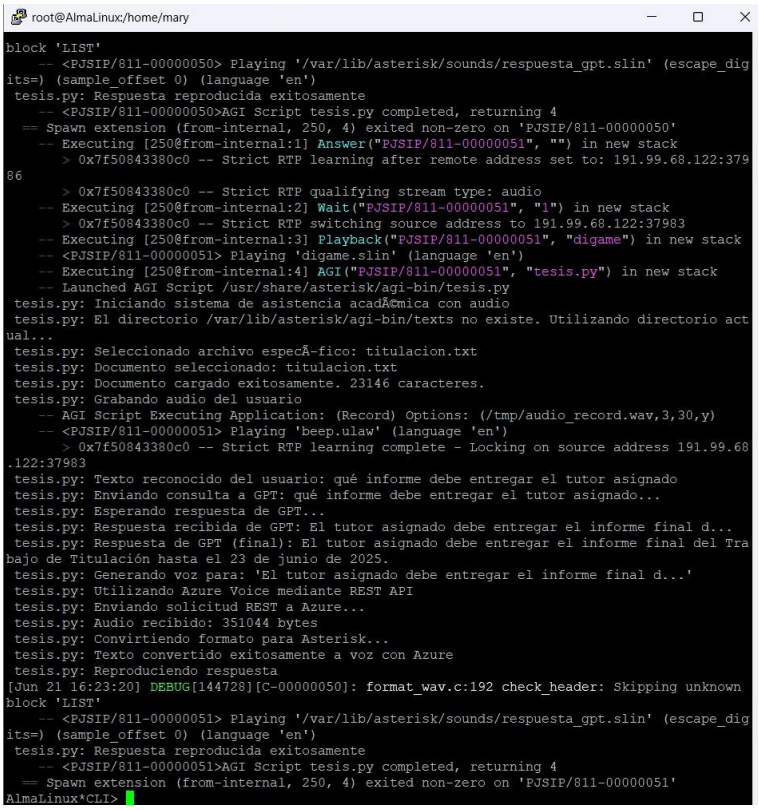



Figura 40. Ejecución del script AGI con consulta y respuesta incorrecta del sistema IVR.

### 3.11.4 Diagrama del sistema de la Operadora IA

El sistema de respuesta interactiva automática implementado sigue una estructura de flujo claramente definida que integra todos los componentes previamente descritos, permitiendo una experiencia conversacional fluida entre el usuario y el agente de inteligencia artificial. La Figura 41 ilustra el diagrama de flujo completo del sistema, mostrando la secuencia de operaciones desde el inicio de una llamada hasta su finalización.

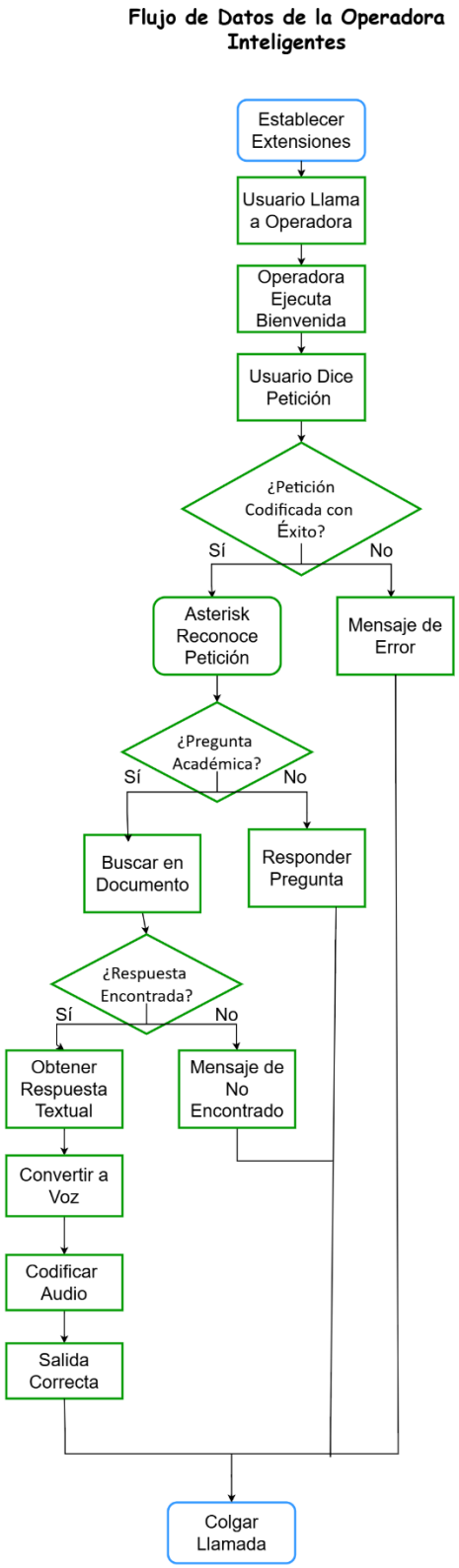



Figura 41. Diagrama de flujo de datos del sistema Asterisk propio

El proceso comienza con el establecimiento de extensiones en el servidor Asterisk, configuradas mediante los archivos pjsip.conf y extensions.conf analizados anteriormente.

Una vez disponible el servicio, el flujo operativo se activa cuando un usuario realiza una llamada a la extensión 250, diseñada específicamente para conectar con la operadora inteligente. El sistema responde automáticamente ejecutando un mensaje de bienvenida predefinido que invita al usuario a formular su consulta.

Tras obtener una transcripción válida, el sistema evalúa si la pregunta corresponde al ámbito académico mediante análisis contextual. Para consultas académicas, se activa la búsqueda en los documentos institucionales previamente cargados utilizando el mecanismo de selección automática implementado en la función seleccionar_txt_auto(). Si la pregunta no se clasifica como académica, se procesa directamente con el conocimiento general del modelo GPT-4.

El procesamiento de la consulta representa otro punto de decisión importante: si se encuentra una respuesta adecuada en los documentos o a través del conocimiento del modelo, se obtiene una respuesta textual concisa; en caso contrario, se genera un mensaje indicando que la información solicitada no está disponible, manteniendo siempre al usuario informado sobre el resultado de su consulta.

La respuesta textual generada se convierte a formato de voz utilizando el servicio Azure AI Speech Services con la voz neuronal "es-ES-ElviraNeural", seleccionada por su naturalidad y claridad. El audio resultante se codifica al formato compatible con Asterisk (WAV 8kHz mono) mediante FFmpeg y se reproduce al usuario a través del canal telefónico.

Finalmente, tras entregar la respuesta, el sistema gestiona correctamente el cierre de la llamada, liberando los recursos y quedando disponible para atender nuevas consultas. Este ciclo completo garantiza una experiencia conversacional coherente y eficiente, automatizando completamente el proceso de consulta académica mediante el uso combinado de tecnologías de telefonía IP y servicios cognitivos avanzados en la nube.

El diagrama refleja claramente la arquitectura modular del sistema, donde cada componente cumple una función específica y bien definida dentro del flujo general, facilitando tanto el mantenimiento como la futura escalabilidad del servicio.

En la Figura 42 se observa el sistema que combina Asterisk con servicios de IA en Azure para transformar una consulta hablada en una respuesta automatizada y precisa, usando reconocimiento y síntesis de voz. Toda la lógica se orquesta mediante scripts AGI y bibliotecas Python, asegurando la comunicación entre la PBX y los servicios en la nube.

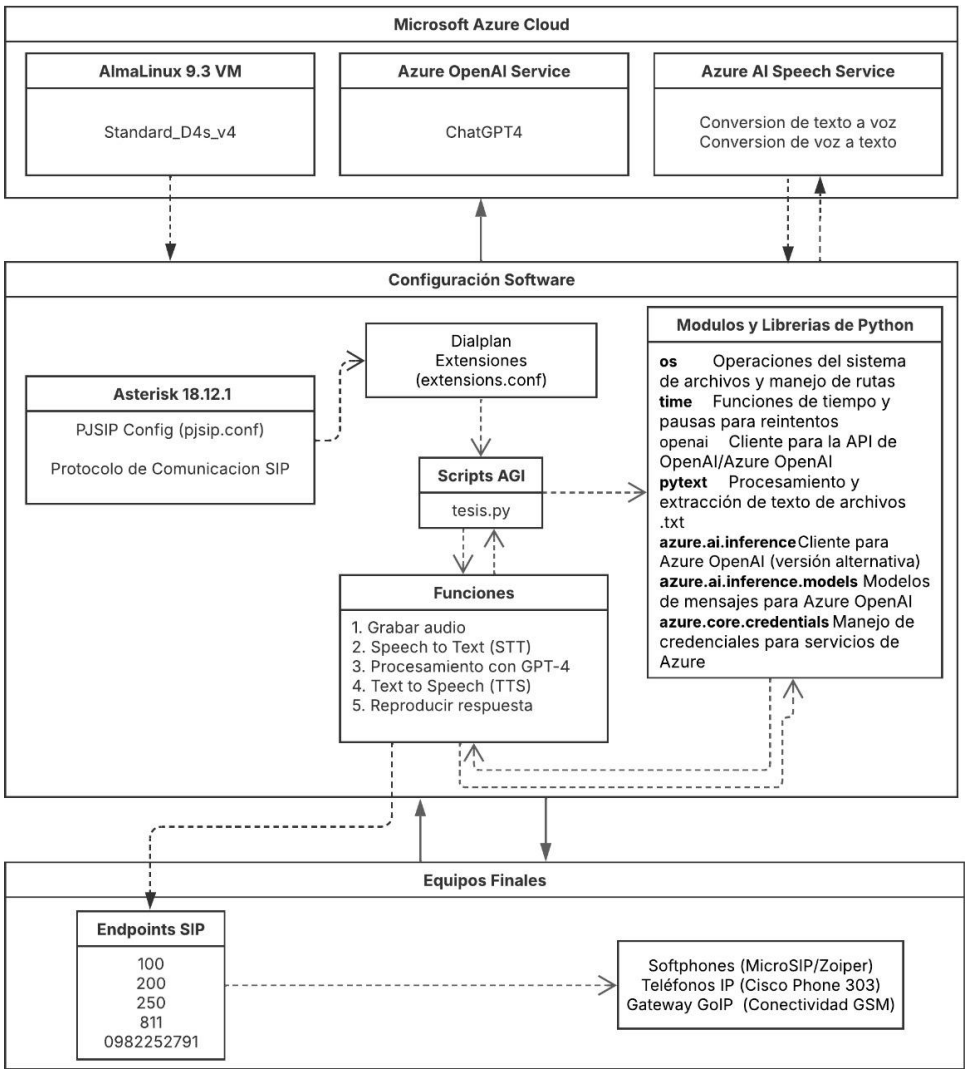



Figura 42. Diagrama UML del Sistema

## 3.12 Resultados

Los resultados obtenidos tras la implementación y evaluación del sistema de respuesta interactiva automática con agentes de IA para teléfonos IP que utilizó Asterisk demostraron la eficacia y viabilidad de la solución propuesta. A continuación, se presenta el análisis detallado de los datos recopilados durante las pruebas del sistema en la Figura 43 se prueba el sistema por medio de la llamada al 250.

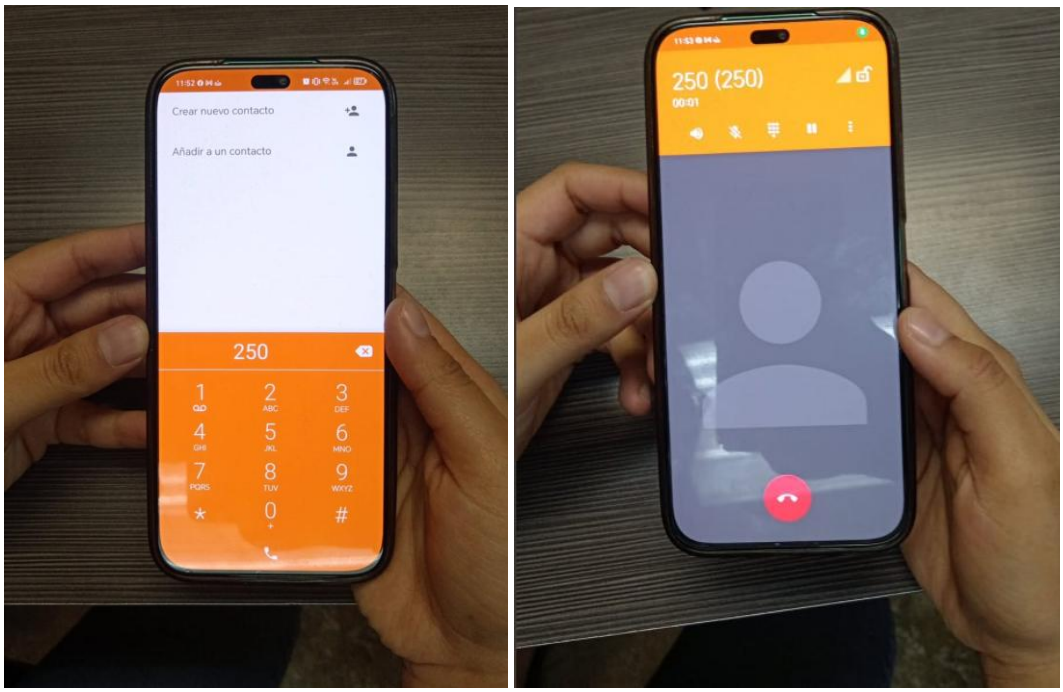



Figura 43. Llamada al sistema para verificación del funcionamiento

Durante las pruebas iniciales del sistema, se evidenció un bajo nivel de precisión en el reconocimiento y procesamiento de la información. Esto se debió a que los documentos implementados inicialmente estaban en formato PDF, lo cual generó dificultades en la extracción correcta del texto debido a problemas de codificación o estructura interna del archivo. Para mejorar el rendimiento, se optó por convertir la información a un archivo de texto plano (.txt). Sin embargo, tras nuevas pruebas, se observó que el sistema seguía presentando limitaciones en la comprensión del contenido, probablemente por la falta de estructuración o detalles clave en el archivo. Como medida correctiva, se generó un archivo .txt final, con una versión más detallada

y estructurada de la información, lo que permitió optimizar el reconocimiento y mejorar la precisión de las respuestas del sistema.

### 3.12.1 Evaluación global del sistema IVR inteligente

Para verificar la funcionalidad del prototipo de sistema IVR con inteligencia artificial se realizaron un total de 20 pruebas piloto, número coherente con estudios previos sobre validación de sistemas conversacionales en fases iniciales. Según Wang (2023), en experimentos de campo con IVR inteligente, se aplican entre 20 y 40 interacciones por escenario para evaluar comprensión y precisión de respuesta[70]. De forma similar, Kim y Park (2020) proponen entre 10 y 30 interacciones para validar la usabilidad y efectividad de agentes conversacionales en entornos de prueba controlada[71]. Estas referencias respaldan que, para un prototipo funcional no desplegado a producción, una muestra de 20 evaluaciones es adecuada para detectar patrones de desempeño, medir la comprensión de lenguaje natural y recoger observaciones que orienten mejoras futuras. Además, la metodología aplicada priorizó la calidad de la retroalimentación sobre la cantidad, asegurando que cada prueba incluyera preguntas reales de trámites académicos y procesos de titulación, alineadas a la documentación institucional disponible.

La Tabla 15. Resumen de resultados de las pruebas del sistema IVR inteligente presenta un resumen de los resultados obtenidos:

Tabla 15. Resumen de resultados de las pruebas del sistema IVR inteligente

| **Métrica** | **Valor** |
|---|---|
| Total de consultas evaluadas | 20 |
| Consultas respondidas correctamente | 18 |
| Consultas con respuesta incorrecta | 2 |
| Tasa global de éxito | 90% |
| Precisión promedio en respuestas correctas | 95.28% |
| Tiempo promedio de respuesta | 7.15 segundos |

La tasa global de éxito se calculó según la Ec en donde:

la relación entre el número de consultas respondidas correctamente y el número total de consultas: (18/20) ×100% = 90%. La precisión promedio se determinó calculando el promedio aritmético de los porcentajes de precisión estimados en las respuestas correctas.

### 3.12.2 Origen y metodología de recolección de datos

La metodología de obtención y procesamiento de datos para el análisis del sistema IVR inteligente siguió un proceso sistemático y riguroso basado en pruebas controladas. Todos los valores numéricos presentados en las ecuaciones y estadísticas del sistema provienen directamente de la experimentación documentada en el Anexo G, Tabla 25

#### *a. Procedimiento de obtención de datos primarios*

Para evaluar el rendimiento del sistema, se implementó un protocolo de prueba estructurado que consistió en:

- ***Selección de evaluadores:*** Se incluyó a 20 participantes con diferentes perfiles y necesidades de información académica.
- ***Formulación de consultas predefinidas:*** Cada evaluador realizó una pregunta específica relacionada con trámites académicos o procesos de titulación.
- ***Documentación sistematizada:*** Para cada prueba se registró los diferentes detalles del comportamiento del sistema descrito en la Tabla 16:

Tabla 16. Puntos tomados en cuenta para el análisis del sistema

| **Detalles del Sistema consultados** |
|---|
| Identificación del evaluador |
| Consulta realizada textualmente |
| Respuesta esperada según documentación institucional |
| Respuesta generada por el sistema |
| Tiempo de respuesta (en segundos) |
| Evaluación cualitativa de la precisión |

- ***Registro audiovisual:*** Las interacciones fueron grabadas para verificación posterior, garantizando la integridad de los datos recolectados.

### 3.12.3 Análisis de precisión de las respuestas

La precisión de las respuestas proporcionadas por el sistema fue evaluada por los usuarios en una escala porcentual. Para cada consulta, se comparó la respuesta generada por el sistema con la respuesta esperada, asignando un porcentaje que

reflejara la exactitud de la información proporcionada. Es importante señalar que estos resultados corresponden a un prototipo en fase piloto, por lo que, en caso de realizarse nuevas pruebas, los resultados podrían variar. Esto se debe a que el sistema, al estar en etapa de validación, en ocasiones puede interpretar solo una parte de la pregunta o generar respuestas que requieren ajustes en la precisión del procesamiento de lenguaje natural. Por ende, tomando los resultados del Anexo G, podemos deducir que el sistema demostró un alto nivel de precisión en la mayoría de las consultas, como se observa en la Figura 44, que muestra la distribución de los niveles de precisión en las respuestas.

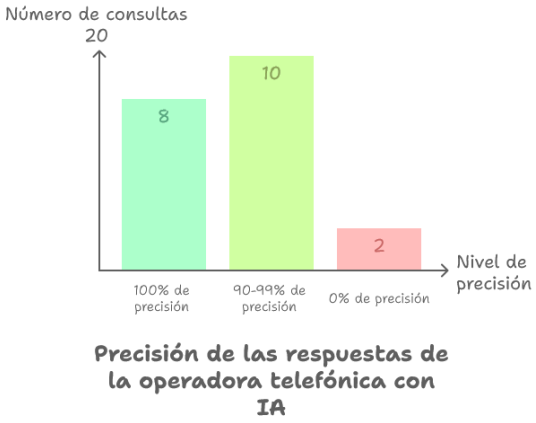



Figura 44. Distribución de precisión en las respuestas del sistema IVR

El cálculo de la precisión promedio se realizó mediante la siguiente fórmula representada en la Ecuación 1.

$$Precisión\ promedio = \frac{Suma\ de\ porcentajes\ de\ precisión}{Número\ de\ consultas\ correctas} \qquad \text{Ecuación 1}$$

Obteniendo el resultado reemplazando con los valores obtenidos en la Ecuación 2

$$P.P = \frac{(100 + 100 + 100 + 100 + 90 + 100 + 95 + 95 + 90 + 100 + 100 + 90 + 90 + 95 + 100 + 95 + 90 + 95)}{18} \qquad \text{Ecuación 2}$$

$$Precisión\ promedio\ = \frac{1715}{18} = 95.28\% \qquad \text{Ecuación 3}$$

Ecuación 3

Este resultado demuestra que, en aquellos casos donde el sistema fue capaz de proporcionar una respuesta, lo hizo con un alto grado de exactitud y relevancia.

### 3.12.4 Análisis de tiempos de respuesta

El tiempo de respuesta es un factor crítico en los sistemas IVR, ya que afecta directamente la experiencia del usuario. Las mediciones realizadas durante las pruebas mostraron los siguientes resultados, visualizados en la Tabla 17:

Tabla 17. Distribución de tiempos de respuesta

| Tiempo de respuesta | Número de consultas | Porcentaje |
|---|---|---|
| 5 segundos | 7 | 35% |
| 6 segundos | 3 | 15% |
| 7 segundos | 3 | 15% |
| 8 segundos | 1 | 5% |
| 10 segundos | 5 | 25% |
| 15 segundos | 1 | 5% |

El tiempo promedio de respuesta se calculó según la Ecuación

$$Tiempo\ promedio = \frac{(Suma\ de\ todos\ los\ tiempos\ de\ respuesta)}{(Número\ total\ de\ consultas)} \qquad \text{Ecuación 4}$$

Y obteniendo un resultado de 7.15 segundos demostrado en la Ecuación Ecuación 6

$$Tiempo\ promedio\ = \frac{\left(5 + 5 + 6 + 5 + 10 + 5 + 7 + 8 + 6 + 10 + 5 + 6 + 10 + 5 + 7 + 6 + 5 + 7 + 10 + 15\right)}{20} \qquad \text{Ecuación 5}$$

$$Tiempo\ promedio\ = \frac{143}{20} = \ 7.15\ segundos \qquad \text{Ecuación 6}$$

El 65% de las consultas fueron respondidas en 7 segundos o menos, lo que demuestra una buena velocidad de procesamiento que contribuye a una experiencia de usuario fluida y natural. Es importante destacar que el tiempo de respuesta incluye todo el proceso: transcripción de voz a texto, procesamiento con GPT-4 y generación de la respuesta de voz.

### 3.12.5 Rendimiento por tipo de consulta

Para analizar con mayor profundidad el comportamiento del sistema, se clasificaron las consultas en diferentes categorías según su temática, como se muestra en la Tabla 18.

Tabla 18. Rendimiento del sistema por categoría de consulta

| **Categoría de consulta** | **Número de preguntas** | **Tasa de éxito** | **Precisión promedio** |
|---|---|---|---|
| Personal y cargos administrativos | 1 | 100% | 100.00% |
| Fechas y calendarios | 2 | 100% | 97.50% |
| Anexos y documentación | 8 | 75% | 81.88% |
| Procesos académicos generales | 9 | 100% | 96.11% |

La precisión promedio se calculó considerando solo las respuestas correctas dentro de cada categoría. La tasa de éxito representa el porcentaje de consultas respondidas correctamente en cada categoría.

Este análisis demostró que el sistema mostró un rendimiento óptimo para consultas sobre procesos académicos generales, con una tasa de éxito del 100% y una alta precisión (96.11%). Las consultas relacionadas con fechas y calendarios también fueron manejadas de manera efectiva (97.50% de precisión).

La mayor limitación se observó en consultas específicas sobre anexos y documentación, donde dos respuestas fueron incorrectas, resultando en una tasa de éxito del 75%.

### 3.12.6 Análisis de errores

Las dos consultas que no fueron respondidas correctamente (preguntas 9 y 16) revelan información valiosa sobre las limitaciones actuales del sistema mostrada en la Tabla 19.

Tabla 19. Respuestas Incorrectas a detalle

| Pregunta | Respuesta esperada | Respuesta del sistema | Análisis del error |
|---|---|---|---|
| ¿Qué documento se entrega para evaluar el informe final? | El Anexo 10. | El archivo no especifica el contenido del informe final | Falta de información específica en el corpus documental proporcionado |
| ¿Qué anexo se usa para evaluar artículos académicos? | El Anexo 13. | No fue encontrado | Información no disponible en el corpus documental |

Ambos errores se relacionan con la ausencia de información específica en la documentación cargada en el sistema, lo que indica que la calidad y completitud de la base de conocimiento es un factor determinante para el rendimiento del sistema.

Estos resultados validan la efectividad de la arquitectura propuesta, que integra Asterisk, GPT-4 a través de Azure OpenAI Service, y servicios de reconocimiento y síntesis de voz para crear un asistente telefónico académico inteligente y útil para la comunidad universitaria.

### 3.12.7 Mecanismos de Integración Avanzada

#### *a. Integración Contextual*

El sistema implemento un mecanismo de integración contextual que permite mantener la coherencia durante la interacción:

- ***Selección Dinámica de Documentos:*** El algoritmo seleccionar_txt_auto() identifica automáticamente el documento más relevante para cada consulta mediante un sistema jerárquico de priorización por contenido y tamaño.

- ***Prompting Avanzado:*** La función consultar_con_contexto() construye prompts estructurados que incluyen directivas específicas sobre formato de respuesta, límites de extensión y priorización de información institucional sobre conocimiento general.

- ***Optimización de Recursos:*** El sistema implementa técnicas de truncado inteligente que limitan el contexto a 3000 caracteres para evitar sobrecargar la API de GPT-4, manteniendo la información más relevante.

### *b. Gestión de Errores y Resiliencia*

La arquitectura incorpora múltiples niveles de manejo de errores:

- ***Reintentos Inteligentes:*** Implementación de algoritmos de backoff exponencial para manejar limitaciones temporales en las APIs.
- ***Degradación Controlada:*** Si un componente falla, el sistema puede seguir operando con funcionalidad reducida.
- ***Monitorización Continua:*** Registro detallado de operaciones mediante agi.verbose() para facilitar diagnóstico y mejora continua.

## 3.13 Costos

El análisis económico del sistema de respuesta interactiva automática con agentes de IA para teléfonos IP implementado se dividió en dos categorías principales: costos de infraestructura en la nube y costos de equipamiento físico. Esta evaluación permitió determinar la viabilidad financiera del proyecto, tanto en su fase de implementación como en su operación continua.

### 3.13.1 Costos de infraestructura en la nube

La implementación realizada en Microsoft Azure constituyó el componente principal del sistema, alojando tanto el servidor Asterisk como los servicios de inteligencia artificial. La Tabla 20 detalla los costos mensuales de esta infraestructura, basados en el consumo real durante el periodo de facturación del 1/6/2025 al 30/6/2025.

Tabla 20. Costos mensuales de servicios en Microsoft Azure

| **Servicio** | **Descripción** | **Costo mensual (USD)** |
|---|---|---|
| Virtual Machines Dv3 Series | Instancia Standard_D4s_v4 (4 vCPUs, 16 GB RAM) para AlmaLinux 9.3 | 15,60 |
| Premium SSD Managed Disks | Almacenamiento persistente para el servidor Asterisk | 1,20 |
| IP Addresses - Standard IPv4 | Dirección IP pública estática | 0,82 |

| Servicio | Descripción | Costo mensual (USD) |
|---|---|---|
| Azure OpenAI Service (GPT-4) | Procesamiento de consultas (est. 500 llamadas/mes) | 30,00* |
| Azure AI Speech Services | Servicios de STT y TTS (est. 500 minutos/mes) | 12,00* |
| Otros cargos | Servicios adicionales (ancho de banda, etc.) | 0,00 |
| | **Total mensual proyectado** | 59,62 |

Los costos de Azure OpenAI y Azure AI Speech Services son estimados basados en el volumen proyectado de llamadas y no aparecen en la captura actual del panel de facturación debido a que se activaron posteriormente.

El costo mensual total proyectado de $59,62 representa un escenario realista para la operación del sistema en un entorno institucional académico con un volumen moderado de consultas. Este valor está alineado con la previsión de $36,63 mostrada en el panel de facturación para los servicios de infraestructura básica, más los costos adicionales estimados para los servicios cognitivos.

### 3.13.2 Costos de equipamiento físico

Complementando la infraestructura en la nube, se requirieron diversos dispositivos físicos para implementar el sistema de telefonía IP. La Tabla 21 detalla los costos de adquisición de este equipamiento.

Tabla 21. Costos de equipamiento físico

| Equipo | Cantidad | Costo unitario (USD) | Costo total (USD) |
|---|---|---|---|
| Teléfono IP Cisco Phone 303 | 2 | 85,00 | 170,00 |
| Gateway GoIP | 1 | 120,00 | 120,00 |
| Router TP-LINK | 1 | 45,00 | 45,00 |
| Módulo SIM | 1 | 5,00 | 5,00 |
| Path cord Ethernet | 4 | 3,50 | 14,00 |
| Fuente de 12 VDC 1A | 2 | 8,00 | 16,00 |
| | | **Total equipamiento físico** | 370,00 |

Los costos del equipamiento físico representaron una inversión inicial única, a diferencia de los servicios en la nube que constituyen un gasto recurrente mensual.

### 3.13.3 Costos de software

El proyecto se benefició significativamente del uso de software de código abierto y herramientas gratuitas, minimizando los costos de licenciamiento. La Tabla 22 resume los costos asociados al software utilizado:

Tabla 22. Costos de software

| **Software** | **Tipo de licencia** | **Costo (USD)** |
|---|---|---|
| Asterisk 18.12.1 | Código abierto (GPL) | 0,00 |
| Python 3.10 | Código abierto (PSF) | 0,00 |
| AlmaLinux 9.3 | Código abierto | 0,00 |
| PJSIP | Código abierto | 0,00 |
| Visual Studio Code | Freeware (Microsoft) | 0,00 |
| ZOIPER versión Android | Freemium (versión básica) | 0,00 |
| MicroSIP para Windows | Código abierto | 0,00 |
| | **Total software** | 0,00 |

### 3.13.4 Costos laborales de desarrollo

El proceso de creación y puesta en funcionamiento del sistema de respuesta interactiva automática requirió una inversión significativa en tiempo de trabajo especializado, cuya valoración económica representa un componente esencial en la estructura de costos del proyecto.

Para cuantificar este aspecto, se ha documentado meticulosamente las jornadas dedicadas al desarrollo, implementación y evaluación del sistema. La Tabla 23 presenta el desglose temporal del proyecto.

Tabla 23. Costos Laborales

| **Descripción** | **Tiempo Ocupado** |
|---|---|
| Desarrollo de dispositivo | 10 días |
| Implementación del dispositivo | 10 días |

| Descripción | Tiempo Ocupado |
|---|---|
| Evaluación del prototipo final | 6 días |
| Total | 26 días |
| Horas laborales | 8 horas |
| **Total de horas** | 208 horas |

Para la valoración monetaria de este esfuerzo técnico, se tomó como referencia la estructura salarial de un ingeniero en telecomunicaciones reciben un sueldo mensual promedio entre los $1.000 y los $2.200 mensuales. Considerando un mes laboral estándar compuesto por 20 días de trabajo, con jornadas de 8 horas, se deriva un valor por hora de trabajo profesional de $6,25, tomando como base los $1.000.

La conversión del tiempo invertido en término económico se realiza mediante la siguiente expresión de la Ecuación 7:

$$Costo\ laboral\ =\ total\ de\ horas\ \times\ costo\ por\ hora\ laboral \qquad \text{Ecuación 7}$$

Que aplicando los valores y datos numéricos nos da el resultado indicado en la Ecuación 8.

$$Costo\ laboral\ =\ 208\ \times\ \$6{,}25\ =\ \$1.300 \qquad \text{Ecuación 8}$$

Por tanto, la valoración económica del capital humano invertido en este proyecto asciende a $1.300, cifra que refleja el componente intelectual y técnico necesario para materializar la solución tecnológica propuesta.

### 3.13.5 Costos totales del proyecto

La implementación del sistema de respuesta interactiva automática con agentes de IA para teléfonos IP representa una inversión integral que abarca múltiples dimensiones: infraestructura tecnológica, recursos humanos y servicios en la nube.

El análisis global de costos permite visualizar el impacto económico completo del proyecto, facilitando así la toma de decisiones basada en el retorno de inversión esperado.

La Tabla 24 presenta la consolidación de todos los componentes de costo identificados en las secciones anteriores:

Tabla 24. Consolidación de costos del proyecto

| **Categoría** | **Componente** | **Costo (USD)** | **Tipo** |
|---|---|---|---|
| **Infraestructura física** | Equipamiento de telefonía IP | 370,00 | Inversión inicial |
| **Servicios en la nube** | Microsoft Azure (mensual) | 59,62 | Recurrente (mensual) |
| **Recursos humanos** | Desarrollo (208 horas) | 1.300,00 | Inversión inicial |
| **Software** | Licencias y aplicaciones | 0,00 | Inversión inicial |
| **Total inversión inicial** | | **1670,00** | |
| **Total gastos recurrentes mensuales** | | **59,62** | |
| **Costo total del proyecto** | | **1729,00** | |

El costo total de implementación se divide claramente en dos componentes principales:

- **Inversión inicial**: $1670,00, que comprende la adquisición del equipamiento físico ($370.00) y el costo del desarrollo técnico ($397.28). Esta inversión representa el 92.8% del costo del primer mes y constituye un gasto único que no se repetirá durante la operación normal del sistema.

- **Gastos operativos recurrentes**: $59.62 mensuales, correspondientes principalmente a los servicios de Microsoft Azure, incluyendo la máquina virtual, almacenamiento, servicios de OpenAI y procesamiento de voz. Este componente representa la base del costo a largo plazo del sistema.

Por tanto, los costos totales del proyecto son de ochocientos veintiséis dólares con noventa centavos de dólar estadounidense aplicado para la elaboración de todo el proyecto.

Es importante destacar que esta estructura de costos representa una ventaja significativa frente a soluciones comerciales de IVR inteligente, cuyos costos de

implementación inicial suelen ser significativamente más altos y los gastos mensuales pueden exceder los $150.00 para funcionalidades similares.

Adicionalmente, la inversión en desarrollo e infraestructura propia confiere mayor flexibilidad para futuras adaptaciones y escalamiento del sistema según las necesidades institucionales específicas.

La ausencia de costos por licenciamiento de software, resultado de la selección estratégica de herramientas de código abierto y servicios basados en consumo, contribuye notablemente a la eficiencia económica del proyecto y establece un precedente valioso para futuras iniciativas tecnológicas en la institución.

# CAPÍTULO IV. CONCLUSIONES Y RECOMENDACIONES

## 4.1 Conclusiones

En el desarrollo del proyecto "SISTEMA DE RESPUESTA INTERACTIVA AUTOMÁTICA CON AGENTES DE IA PARA TELÉFONOS IP USANDO ASTERISK" mediante el uso de diferentes recursos de API para el desarrollo de su entorno ha sido un medio de servicio telefónico que ha podido ser útil y funcional, presenta las siguientes conclusiones en base a su funcionamiento exitoso.

- La investigación permitió identificar que los estudiantes de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial enfrentan dificultades significativas para acceder a información actualizada sobre trámites académicos. El análisis reveló que un 81.8% busca información sobre fechas y tramites académicos por canales informales de comunicación (grupos de WhatsApp y consultas presenciales) debido a la falta de claridad en los canales formales, lo que evidencia la necesidad crítica de sistemas automatizados que proporcionen información confiable y accesible permanentemente.

- Se logró implementar exitosamente un sistema basado en Asterisk 18.12.1 sobre AlmaLinux 9.3 en Microsoft Azure, utilizando el protocolo PJSIP para la comunicación SIP. La arquitectura modular desarrollada, que integra scripts AGI en Python con los servicios Azure OpenAI (GPT-4) y Azure AI Speech Services, demostró ser técnicamente viable y efectiva para la automatización de consultas académicas, permitiendo una integración fluida entre la central telefónica IP y los servicios cognitivos en la nube.

- Las pruebas realizadas con 20 consultas específicas evidenciaron una tasa de éxito del 90%, con 18 respuestas correctas y solo 2 incorrectas. La precisión promedio en las respuestas correctas alcanzó el 95.28%, con un tiempo medio de respuesta de 7.15 segundos. El análisis por categorías mostró un rendimiento óptimo (100% de tasa de éxito) en consultas sobre procesos académicos generales y fechas importantes, mientras que las preguntas sobre anexos específicos representaron el

mayor desafío (75% de tasa de éxito), indicando que la completitud de la base documental es crucial para el rendimiento del sistema.

- El análisis de costos demostró la factibilidad financiera del proyecto, con una inversión inicial de $1670,00 (equipamiento físico y desarrollo) y un costo operativo mensual de $59.62 (servicios en la nube). Esta estructura representa una alternativa significativamente más económica que soluciones comerciales equivalentes, evidenciando que la integración de tecnologías de código abierto con servicios cognitivos en la nube resulta accesible para instituciones educativas con presupuestos limitados.

## 4.2 Recomendaciones

- Conectar el IVR inteligente directamente con las bases de datos institucionales para proporcionar información en tiempo real sobre calificaciones, horarios y trámites personalizados. Esta integración permitiría consultas automáticas sobre el progreso curricular de cada estudiante, materias pendientes, requisitos de graduación y estado de solicitudes administrativas. La implementación debería realizarse mediante APIs seguras que garanticen la protección de datos personales y cumplan con normativas de privacidad institucional. El sistema podría proporcionar recordatorios personalizados sobre fechas límite, pagos pendientes y documentos requeridos para procesos específicos. Esta funcionalidad transformaría el IVR de un sistema de información general a una herramienta personalizada que atiende necesidades específicas de cada usuario.

- Se sugiere desarrollar una versión complementaria basada en WhatsApp o Telegram que utilice la misma base de conocimiento e infraestructura de inteligencia artificial del sistema principal. Esta expansión respondería a los hábitos de comunicación actuales de los estudiantes, quienes prefieren interactuar a través de aplicaciones de mensajería instantánea por su familiaridad y accesibilidad. Los chatbots deberían mantener coherencia en las respuestas y calidad del servicio, incluyendo funcionalidades como envío de documentos, imágenes y notificaciones push. Esta estrategia omnicanal permitiría que los

estudiantes elijan su método de comunicación preferido, aumentando la adopción del sistema y mejorando la satisfacción del usuario.

- Implementar técnicas avanzadas de ingeniería de prompts para mejorar significativamente la contextualización y precisión de las respuestas generadas por GPT-4. Esta optimización implica el diseño sistemático de instrucciones específicas que guíen al modelo hacia respuestas más precisas, relevantes y contextualmente apropiadas para el ambiente universitario. Las técnicas deberían incluir prompts que incorporen contexto institucional específico, terminología académica adecuada y estructura de respuestas alineadas con estándares de comunicación universitarios. Se sugiere implementar un sistema de prompts dinámicos que se adapten según el tipo de usuario y el contexto de la consulta. Esta aproximación metodológica reduciría las alucinaciones del modelo, mejoraría la coherencia de respuestas y mantendría un tono profesional consistente. La implementación debería incluir pruebas A/B para evaluar la efectividad de diferentes configuraciones y establecer métricas de calidad para refinamiento continuo. La ingeniería de prompts optimizada garantizaría respuestas de alta calidad y confiabilidad en el contexto académico específico.

- Añadir soporte para consultas en kichwa y otras lenguas indígenas relevantes para la comunidad universitaria, promoviendo la inclusión y accesibilidad lingüística del sistema. Esta implementación requiere un enfoque que vaya más allá de la traducción automática, incorporando entendimiento cultural y particularidades comunicativas de cada lengua indígena representada. El desarrollo debería involucrar hablantes nativos y expertos en lingüística para asegurar traducciones precisas y culturalmente apropiadas de términos académicos y administrativos. La funcionalidad multilingüe incluiría detección automática del idioma de consulta y respuesta en la misma lengua, manteniendo coherencia terminológica y contexto cultural específico. Se recomienda crear una base de datos especializada con equivalencias de términos académicos en cada lengua soportada, considerando que conceptos universitarios modernos pueden carecer de traducciones directas

- Desarrollar una interfaz web integral que permita a administradores monitorear el sistema, actualizar la base de conocimiento y visualizar estadísticas de uso en

tiempo real. Esta plataforma administrativa proporcionaría un dashboard con métricas clave como volumen de consultas, tipos de preguntas frecuentes, tiempos de respuesta y análisis de patrones de uso por demografía estudiantil. La interfaz incluiría herramientas de gestión de contenido para agregar, editar o eliminar información de manera intuitiva sin requerir conocimientos técnicos avanzados. Debería incorporar un sistema de alertas para consultas no resueltas, errores del sistema o picos inusuales de tráfico que requieran atención inmediata. La plataforma ofrecería capacidades de exportación de datos para análisis externos, generación de reportes automáticos e integración con sistemas institucionales existentes.

## REFERENCIAS BIBLIOGRÁFICAS


[1] S. I. & Consulting, "Voice over Internet Protocol (VoIP) Market Size, Share & Trends Forecast Report, 2023–2032." May-2023.

[2] M. (Salesforce), "10th Annual Connectivity Benchmark Report." Jan-2025.

[3] F. Consulting, "The Total Economic Impact™ Of IBM Watson Assistant." 2023.

[4] G. A. B. Gallardo, "Sistema web integrado para la gestión de trámites académico administrativos en la Facultad de Ingeniería en Sistemas, Electrónica e Industrial." 2023.

[5] P. Acevedo, A. Magana, B. Benes, and C. Mousas, "A Systematic Review of Immersive Virtual Reality in STEM Education: Advantages and Disadvantages on Learning and User Experience," *IEEE Access*, vol. 12, 2024.

[6] ARCOTEL, "Regulación de servicios de telefonía IP en Ecuador." 2022.

[7] P. A. Aguirre Castillo, "Propuesta de implementación de una red de telefonıa IP basada en ASTERISK para el Ministerio Público Piura; 2022."

[8] B. C. OLIVERA, O. Armando, B. G. VILLANUEVA, and C. Andrés, "IMPLEMENTACIÓN DEL SERVICIO DE VOZ SOBRE IP BASADO EN SOFTWARE LIBRE ASTERISK VERSION 13.21. 1 PARA OPTIMIZAR LA COMUNICACIÓN EN LA EMPRESA CSM CORPORACIÓN ORIENTE SAC. IQUITOS, 2023."

[9] R. M. León and J. G. Moreno, "Prototipo de Asistente Personal Virtual basado en Inteligencia Artificial aplicado en las oficinas de TI del Sector Público Peruano," *Revista CIENCIA Y TECNOLOGÍA*, vol. 19, no. 4, pp. 101–111, 2023.

[10] K. M. Shaikh and G. Giannakopoulos, "Evolution of IVR building techniques: from code writing to AI-powered automation," *arXiv preprint arXiv:2411.10895*, 2024.

[11] C. M. L. Espinoza and N. M. M. Núñez, “Implementación de procesos de autogestión IVR soporte móvil en empresa de telecomunicaciones,” Universidad Tecnológica Centroamericana UNITEC, 2024.

[12] J. Smith and A. Johnson, “An Overview of Voice over IP (VoIP) Technologies,” *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 123–145, 2024.

[13] W. Tuleun, “Design of an Asterisk-based VoIP system and the implementation of security solution across the VoIP network,” *World Journal of Advanced Research and Reviews*, vol. 23, no. 1, pp. 875–906, 2024.

[14] S. M. C. Delgado and others, “Revisión sistemática de Comunicaciones Unificadas de VoIP en redes CAN,” *Informática y Sistemas: Revista de Tecnologías de la Informática y las Comunicaciones*, vol. 5, no. 1, pp. 17–34, 2021.

[15] ITU-T, “Recommendation H.323 (V8): Packet-based multimedia communications systems,” International Telecommunication Union, Mar. 2022.

[16] A. A. L. Castellanos, “Segmentación de contestadoras automáticas en marcadores para centros de llamadas con telefonía IP utilizando Asterisk,” CIATEQ A.C., Guadalajara, México, 2020.

[17] H. Schulzrinne and J. Rosenberg, “Internet telephony: architecture and protocols,” *IEEE Network*, vol. 13, no. 3, pp. 18–27, 1999.

[18] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, “SIP: Session Initiation Protocol.” Jun-2002.

[19] Digium, “The Inter-Asterisk Exchange Protocol Version 2.” .

[20] A. Project, “Configuring res_pjsip.” .

[21] B. Goode, “Voice over Internet Protocol (VoIP),” *Proceedings of the IEEE*, vol. 90, no. 9, pp. 1495–1517, Sep. 2002.

[22] I. Digium, *Asterisk: The Definitive Guide*, 5th ed. O’Reilly Media, 2019.

[23] I. O. for Standardization, “ISO 9001:2015 Quality management systems – Requirements.” ISO, 2015.

[24] M. Rebolledo and M. Goncalves, “Integración de Chatbots en Instituciones de Educación Superior para la Optimización del Ciclo PHVA según la Norma ISO 9001:2015/Amd 1:2024,” *Revista Tekhné*, vol. 27, no. 2, pp. 101–111, Jul. 2024.

[25] E. Parliament and Council, “Regulation (EU) 2016/679: General Data Protection Regulation.” 2016.

[26] A. N. del Ecuador, “Ley Orgánica de Protección de Datos Personales.” 2021.

[27] P. de la República del Ecuador, “Decreto Ejecutivo No. 904: Reglamento General a la Ley Orgánica de Protección de Datos Personales.” 2022.

[28] C. Ronceros, J. Assi, Y. Martinez, J. Medina, and P. León, “Sistema de Respuesta de Voz Interactiva (IVR) para la notificación de datos operacionales de la Planta Boquerón de la División Furrial - PDVSA,” *Revista Tecnológica - ESPOL*, vol. 34, no. 3, pp. 100–117, Nov. 2022.

[29] S. Khatua, “Cloud meets telecom & application in open source Asterisk (IP PBX),” *International Journal of Electrical, Computing Engineering and Communication*, vol. 1, no. 3, pp. 49–52, 2015.

[30] K. M. Lee and J. C. Lai, “Speech versus touch: A comparative study of the use of speech and DTMF keypad for navigation,” *International Journal of Human–Computer Interaction*, vol. 19, no. 3, pp. 343–360, 2005.

[31] R. Karademir and E. Heves, “Dynamic interactive voice response (IVR) platform,” in *Proc. IEEE EUROCON 2013*, 2013, pp. 98–104.

[32] E. Coman, “IVR systems used in call center management: a scientometric analysis of the literature,” *Frontiers in Computer Science*, vol. 7, p. Art. 1459787, 2025.

[33] M. Płaza and Ł. Pawlik, “Influence of the contact center systems development on key performance indicators,” *IEEE Access*, vol. 9, pp. 44580–44591, 2021.

[34] A. Shah and P. S. Sathya, “VoIP-Based Call Management System using 3CX,” *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 10, no. 5, pp. 1234–1240, 2022.

[35] J. López and others, “Evaluación de FreePBX como sistema PBX de código abierto para pymes,” *Revista Tecnológica ESPOL*, vol. 33, no. 2, pp. 20–28, 2023.

[36] M. Cárdenas and L. Díaz, “Análisis comparativo de plataformas de telefonía IP: Asterisk, FreePBX e Issabel,” *Revista de Tecnología e Innovación*, vol. 15, no. 1, pp. 45–52, 2024.

[37] K. Hernández, “Implementación de FusionPBX como solución IP-PBX sobre FreeSWITCH.” Universidad de San Carlos, 2021.

[38] A. Taghavi and J. D. Baldwin, “An Overview of VoIP PBX Systems: An Evaluation of the Most Common Platforms,” *Journal of Communications and Networks*, vol. 25, no. 2, pp. 143–152, Apr. 2023.

[39] J. C. Palacios Arteaga, M. Vásquez Romero, F. Orozco Cruz, and D. Espinoza, “Aplicabilidad de telefonía IP en la computación en la nube,” *Journal of Science and Research*, vol. 3, no. 1, pp. 25–31, 2018.

[40] J. Van Meggelen, R. Bryant, and L. Madsen, *Asterisk: The Definitive Guide*, 5th ed. O’Reilly Media, 2019.

[41] M. Véliz, “Diseño e implementación de un sistema de telefonía IP con Asterisk para una red LAN.” Universidad Técnica del Norte, 2021.

[42] D. López, “Implementación de una central IP con Asterisk.” Universidad Distrital Francisco José de Caldas, 2020.

[43] M. Rodríguez and E. Gutiérrez, “Desarrollo de un sistema de atención automatizada mediante IVR y AMI en Asterisk,” *Revista Politécnica*, vol. 42, no. 2, pp. 30–39, 2021.

[44] C. Zambrano Lino, “Implementación de un servidor Asterisk en la nube interna de la Universidad de Guayaquil.” Universidad de Guayaquil, 2013.

[45] J. C. Palacios, M. Vásquez, F. Orozco, and D. Espinoza, “Aplicabilidad de telefonía IP en la computación en la nube,” *Journal of Science and Research*, vol. 3, no. 1, pp. 25–31, 2018.

[46] M. Ahmad and A. Van Looy, “Impact of Artificial Intelligence on Quality Management Practices: A Systematic Review,” *Journal of Quality Management*, vol. 29, no. 4, pp. 123–139, 2020.

[47] A. Aguilar, “Central telefónica IP basada en Asterisk montada sobre Amazon Web Services.” Universidad Técnica de Ambato, 2021.

[48] S. Willie, “High Availability Asterisk and FreePBX on Microsoft Azure.” 2014.

[49] P. Pal, T. Triyason, and V. Vanijja, “Asterisk server performance under stress test,” in *Proc. IEEE International Conference on Communication Technology (ICCT)*, 2017, pp. 1038–1052.

[50] A. Community, “Integrating AI Receptionist with Asterisk – ARI vs AGI.” Nov-2024.

[51] NerdVittles, “VoIP Communications with Asterisk in Amazon’s EC2 Cloud.” 2012.

[52] AsteriskService, “Asterisk-hosted IP PBX with Oracle Cloud Computing.” 2024.

[53] VPSBenchmarks, “Linode vs Vultr: Performance, Features and Prices.” 2023.

[54] Microsoft, “Microsoft Azure Documentation: Virtual Machines.” 2024.

[55] M. Yakubova, O. Manankova, A. Mukasheva, A. Baikenov, and T. Serikov, “The Development of a Secure Internet Protocol (IP) Network Based on Asterisk Private Branch Exchange (PBX),” *Applied Sciences*, vol. 13, no. 19, p. 10712, 2023.

[56] M. Adelusola, “Integration of Asterisk VoIP with Cloud Services for Scalability.” 2024.

[57] S. Willie, “Build HA Asterisk on Microsoft Azure using DRBD/Heartbeat.” Kuala Lumpur, Malaysia, Apr-2015.

[58] P. Nuño, C. Suárez, E. Suárez, and J. C. Granda, “A Diagnosis and Hardening Platform for an Asterisk VoIP PBX,” *Security and Communication Networks*, vol. 2020, p. Article ID 8853625, 2020.

[59] T. Oladoja, “Design and Deployment of a Secure VoIP Infrastructure Using Asterisk.” Sep-2024.

[60] A. Chaudhary, Y. Surya, R. Kumar, and V. Kumar, “Cloud-Hosted IP-PBX System with Asterisk and Oracle Cloud: A Scalable and Cost-Effective Solution for Business Communications,” in *Proc. 2nd Intl. Conf. on Disruptive Technologies (ICDT)*, 2024, pp. 837–841.

[61] M. M. Lang and S. Eskenazi, “Telephone Surveys Meet Conversational AI: Evaluating a LLM-Based Telephone Survey System at Scale,” *arXiv preprint*, vol. arXiv:2502.20140, Feb. 2025.

[62] V. Project, “AI Agent with OpenAI ChatGPT, Whisper and Azure Speech.” 2023.

[63] T. Nakada, T. Oami, and Y. Okada, “GPT-3.5 Turbo and GPT-4 Turbo in Title and Abstract Screening for Systematic Reviews,” *JMIR Medical Informatics*, Mar. 2025.

[64] L. A. Perine, R. Marciano, and M. Conrad, “Model Selection for HERITAGE-AI: Evaluating LLMs for Contextual Data Analysis of Maryland’s Domestic Traffic Ads (1824–1864),” in *Proceedings of the 2024 IEEE International Conference on Big Data (BigData)*, 2024.

[65] OpenAI, “GPT-4 Technical Report,” *arXiv preprint*, vol. arXiv:2303.08774, Mar. 2023.

[66] A. Silberschatz, P. B. Galvin, and G. Gagne, *Operating System Concepts*, 10th ed. Hoboken, NJ, USA: John Wiley & Sons, 2018.

[67] A. O. Foundation, “AlmaLinux Overview.” 2024.

[68] I. Red Hat, “Red Hat Enterprise Linux: Performance Tuning Guide.” 2023.

[69] D. Project, “Debian Administrator’s Handbook.” 2022.

[70] L. Wang, N. Huang, Y. Hong, J. Wang, and B. Lee, “Voice-based AI in Call Center Customer Service: A Natural Field Experiment,” *Production and Operations Management*, vol. 32, no. 1, pp. 175–193, 2023.

[71] J. Kim and S. Park, “Designing and Evaluating a Conversational Agent: Protocol and Case Study,” Journal of Medical Internet Research, vol. 22, no. 11, p. e18023, 2020.

# ANEXOS

Anexo A. Código pjsip

```
[transport-udp]
type=transport
protocol=udp
bind=0.0.0.0
external_media_address=172.178.112.41  ; Reemplaza con tu IP pública
external_signaling_address=172.178.112.41  ; Reemplaza con tu IP pública
local_net=10.0.0.0/8
local_net=172.16.0.0/12
local_net=192.168.0.0/16

[100]
type=endpoint
context=from-internal
disallow=all
auth=100
aors=100
direct_media=no  ; Importante para NAT
rtp_symmetric=yes  ; Importante para NAT
ice_support=yes  ; Ayuda con NAT
rewrite_contact=yes  ; Importante para NAT


[200]
type=endpoint
context=from-internal
disallow=all
allow=ulaw,alaw
transport=transport-udp
auth=200
aors=200
direct_media=no  ; Importante para NAT
rtp_symmetric=yes  ; Importante para NAT

[200]
type=auth
auth_type=userpass
password=password123
username=200

[200]
max_contacts=5
qualify_frequency=30  ; Mantiene la conexión activa


[811]
type=endpoint
```

```
disallow=all
allow=ulaw,alaw
auth=811
aors=811
direct_media=no  ; Importante para NAT

[811]
type=auth
auth_type=userpass
password=password123
username=811

[811]
type=aor
max_contacts=5
qualify_frequency=30  ; Mantiene la conexión activa
```

Anexo B. Código extensions.conf

```
[[from-internal]
; Direct calls between extensions
exten => _XXX,1,Answer()
 same => n,Dial(PJSIP/${EXTEN},30)
 same => n,Hangup()

; Specific handling for extension 100
exten => 100,1,Answer()
 same => n,Dial(PJSIP/100,30)
 same => n,Hangup()

; Specific handling for extension 200
exten => 200,1,Answer()
 same => n,Dial(PJSIP/200,30)


exten => 811,1,Answer()
 same => n,Dial(PJSIP/811,30)
exten => 250,1,Answer()
exten => 250,3,Playback(digame)
;exten => 250,4,AGI(marychat.py)
exten => 250,4,AGI(tesis.py)
```

```
#!/usr/bin/env python3

import os
import time
import speech_recognition as sr
from asterisk.agi import *
import requests  # Para llamadas REST a Azure
from azure.ai.inference.models import SystemMessage, UserMessage
from azure.core.credentials import AzureKeyCredential

# Configuracion de Azure OpenAI
endpoint = "https://ai-149288.openai.azure.com/openai/deployments/gpt-4"
model_name = "gpt-4"

client = ChatCompletionsClient(
   endpoint=endpoint,
   credential=AzureKeyCredential("GEgSzcywZfrCDXlO1M9bUTUTQXB9h2v3mYF
u0eYlibsZPpKGmyDeJQQJ99BCACHYHv6XJ3w3AAAAACOGXgSJ"),
)

# ConfiguraciÃ³n del servicio de voz de Azure
speech_key =
"2N37UG7pXFxZ4nDjPDqYEuiHmhwH9fBJgzAMfn77PsNSXQ6RZ407JQQJ99BCAC
HYHv6XJ3w3AAAAACOG3MeD"
speech_region = "eastus2"

# Funcion para extraer texto de un archivo TXT
   if not os.path.exists(txt_path):
      return "Error: El archivo TXT no se encuentra en la ruta especificada."

   try:
      with open(txt_path, 'r', encoding='utf-8') as file:
         text = file.read()
      return text
   except Exception as e:
      return f"Error al procesar el archivo TXT: {str(e)}"
   intentos = 0
   tiempo_espera = 60  # Tiempo inicial de espera en segundos

   while intentos < max_intentos:
      try:
         # Limitar el contexto para reducir tokens y evitar errores de lÃmite
         contexto_recortado = contexto_txt[:3000]  # Limitado a 3000 caracteres

Contenido del archivo:
{contexto_recortado}

Pregunta: {pregunta}
```

```
            if agi:
                agi.verbose("Esperando respuesta de GPT...")

            response = client.complete(
                messages=[
                    SystemMessage(content="Eres un asistente especializado que analiza
documentos y responde con extrema brevedad (mÃ¡x. 150 caracteres)."),
                    UserMessage(content=prompt),
                ],
                max_tokens=500,  # Limitado a respuestas cortas
                top_p=1.0,
                model=model_name
            )

            respuesta = response.choices[0].message.content
            if agi:
                agi.verbose(f"Respuesta recibida de GPT: {respuesta[:50]}...")
            return respuesta

            # Verificar si es un error de limite de tasa (429)
            if "429" in error_msg and intentos < max_intentos:
                if agi:
                    agi.verbose(f"LÃmite de tasa excedido. Esperando {tiempo_espera}
segundos antes de reintentar ({intentos}/{max_intentos})...")
                time.sleep(tiempo_espera)
                tiempo_espera *= 1.5  # Aumentar el tiempo de espera para el
prÃ³ximo reintento
            else:
                if intentos >= max_intentos:
                    return "No pude procesar tu consulta. IntÃ©ntalo nuevamente."
                return "OcurriÃ³ un error al procesar tu consulta."

        # Endpoint REST de Azure Speech
        endpoint =
f"https://{speech_region}.tts.speech.microsoft.com/cognitiveservices/v1"

        # Encabezados para la solicitud
        headers = {
            'Ocp-Apim-Subscription-Key': speech_key,
            'Content-Type': 'application/ssml+xml',
            'X-Microsoft-OutputFormat': 'riff-24khz-16bit-mono-pcm'
        }

        # Preparar SSML
        ssml = f"""
        <speak version='1.0' xmlns='http://www.w3.org/2001/10/synthesis'
xml:lang='es-ES'>
            <voice name='es-ES-ElviraNeural'>{texto_limpio}</voice>
        </speak>
        """
```

```
        # Archivo temporal para el audio sin procesar
        timestamp = int(time.time())
        temp_wav = f"/tmp/azure_temp_{timestamp}.wav"

        # Hacer solicitud HTTP (no WebSocket)
        if agi:
            agi.verbose("Enviando solicitud REST a Azure...")

        response = requests.post(endpoint, headers=headers,
data=ssml.encode('utf-8'))

        if response.status_code == 200:
            # Guardar audio en archivo temporal
            with open(temp_wav, 'wb') as audio:
                audio.write(response.content)

            if agi:
                agi.verbose(f"Audio recibido: {len(response.content)} bytes")

            # Convertir a formato Asterisk
            comando = f"ffmpeg -y -i {temp_wav} -ar 8000 -ac 1 {output_wav} -
loglevel quiet"

            if agi:
                agi.verbose(f"Convirtiendo formato para Asterisk...")

            process = subprocess.run(comando, shell=True, stdout=subprocess.PIPE,
stderr=subprocess.PIPE)

            # Limpiar archivo temporal
            if os.path.exists(temp_wav):
                os.remove(temp_wav)

            if process.returncode == 0 and os.path.exists(output_wav):
                if agi:
                    agi.verbose(f"Texto convertido exitosamente a voz con Azure")
                return True
            else:
                if agi:
                    agi.verbose("Error al convertir el audio con ffmpeg")
                return False
        else:
            if agi:
                agi.verbose(f"Error en la API de Azure: {response.status_code}")
                if response.text:
                    agi.verbose(f"Detalles: {response.text[:100]}...")
            return False

    except Exception as ex:
        if agi:
```

```
            agi.verbose(f"Error en sintesis de Azure: {str(ex)}")
        return False


    # Si hay un solo TXT, usarlo
    if len(txt_files) == 1:
        if agi:
            agi.verbose(f"Seleccionado Ãºnico archivo: {txt_files[0]}")
        return os.path.join(directorio, txt_files[0])

    # Si hay multiples TXT, usar el mÃ¡s grande
    txt_sizes = [(f, os.path.getsize(os.path.join(directorio, f))) for f in txt_files]
    txt_sizes.sort(key=lambda x: x[1], reverse=True)  # Ordena por tamaÃ±o
descendente
    selected_txt = txt_sizes[0][0]

    if agi:
        agi.verbose(f"Seleccionado archivo mÃ¡s grande: {selected_txt}")
    return os.path.join(directorio, selected_txt)

def main():
    agi = AGI()
    agi.verbose("Iniciando sistema de asistencia acadÃ©mica con audio")

    try:
        # Seleccionar el archivo TXT
        txt_path = seleccionar_txt_auto(agi)
        if not txt_path:
            agi.verbose("No se encontrÃ³ ningÃºn archivo TXT para procesar.")
            agi.hangup()
            return

        agi.verbose(f"Documento seleccionado: {os.path.basename(txt_path)}")

        # Cargar el texto del documento
        txt_text = extract_text_from_txt(txt_path)

        if txt_text.startswith("Error:"):
            agi.verbose(f"Error al cargar documento: {txt_text}")
            agi.hangup()
            return

        agi.verbose(f"Documento cargado exitosamente. {len(txt_text)}
caracteres.")

        # Capturar el audio desde el telÃ©fono
        audio_filename = "/tmp/audio_record.wav"
        agi.verbose("Grabando audio del usuario")
        agi.exec_command(f"Record {audio_filename},3,30,y")

        # Convertir el audio a texto usando reconocimiento de voz
```

```
recognizer = sr.Recognizer()
try:
   with sr.AudioFile(audio_filename) as source:
      audio_data = recognizer.record(source)
      pregunta = recognizer.recognize_google(audio_data, language='es-ES')
      agi.verbose(f"Texto reconocido del usuario: {pregunta}")
except Exception as e:
   agi.verbose(f"Error al reconocer el audio: {str(e)}")
   agi.hangup()
   return

# Enviar el texto reconocido a GPT con el contexto del documento
respuesta = consultar_con_contexto(pregunta, txt_text, agi=agi)

# Limitar la respuesta para evitar problemas con la sÃntesis de voz
if len(respuesta) > 150:
   respuesta = respuesta[:147] + "..."

agi.verbose(f"Respuesta de GPT (final): {respuesta}")

# Preparar ruta para el archivo de salida
wav_file = "/var/lib/asterisk/sounds/respuesta_gpt.wav"

# Generar la respuesta en audio usando Azure REST API
if generate_voice(respuesta, wav_file, agi):
   # Reproducir el mensaje generado
   try:
      agi.verbose("Reproduciendo respuesta")
      agi.stream_file('/var/lib/asterisk/sounds/respuesta_gpt')
```

Anexo D. Código archivo lectura.py

```
import os
import time
from azure.ai.inference import ChatCompletionsClient
from azure.ai.inference.models import SystemMessage, UserMessage
from azure.core.credentials import AzureKeyCredential

# Configuración de Azure OpenAI
endpoint =
"https://ai543990149288.openai.azure.com/openai/deployments/gpt-4"
model_name = "gpt-4"

client = ChatCompletionsClient(
   endpoint=endpoint,
   credential=AzureKeyCredential("GEgSzcywZfrCDXlO1M9bUTUTQXB9h2v3mYF
u0eYlibsZPpKGmyDeJQQJ99BCACHYHv6XJ3w3AAAAACOGXgSJ"),
)

   try:
      with open(txt_path, 'r', encoding='utf-8') as file:
         text = file.read()
      return text
   except Exception as e:
      return f"Error al procesar el archivo TXT: {str(e)}"

# Función para realizar consultas a ChatGPT con contexto del TXT y manejo de
reintentos
def consultar_con_contexto(pregunta, contexto_txt, max_intentos=3):
   intentos = 0
   tiempo_espera = 60  # Tiempo inicial de espera en segundos

   while intentos < max_intentos:
      try:
         # Limitar el contexto para reducir tokens y evitar errores de límite
         contexto_recortado = contexto_txt[:3000]  # Limitado a 3000 caracteres

Contenido del archivo:
{contexto_recortado}

Pregunta: {pregunta}

Si la respuesta se encuentra en el archivo, por favor indícalo y proporciona la
información relevante.
Si no encuentras la información en el archivo, responde con tu conocimiento
general pero indica que no está en el documento.
"""

         return response.choices[0].message.content
```

```
            except Exception as e:
                intentos += 1
                error_msg = str(e)

                # Verificar si es un error de límite de tasa (429)
                if "429" in error_msg and intentos < max_intentos:
                    print(f"\nLímite de tasa excedido. Esperando {tiempo_espera} segundos
antes de reintentar ({intentos}/{max_intentos})...")
                    time.sleep(tiempo_espera)
                    tiempo_espera *= 1.5  # Aumentar el tiempo de espera para el próximo
reintento
                else:
                    if intentos >= max_intentos:
                        return f"Error después de {max_intentos} intentos:
{error_msg}\n\nSugerencia: Intente nuevamente más tarde cuando se
restablezca su cuota de API."
                    return f"Error al consultar a ChatGPT: {error_msg}"

        # Si hay un TXT llamado "Titulacion.txt" o similar, priorizarlo
        target_files = [f for f in txt_files if "titulacion" in f.lower()]
        if target_files:
            selected_txt = target_files[0]
            return os.path.join(directorio, selected_txt)

        # Si hay un solo TXT, usarlo
        if len(txt_files) == 1:
            return os.path.join(directorio, txt_files[0])

    # Ejemplo de uso interactivo
    if __name__ == "__main__":
        print("Sistema de Asistencia Académica con Inteligencia Artificial")
        print("------------------------------------------------------------")
        print("Este sistema analiza documentos de texto y responde preguntas
basadas en su contenido.")

        # Selección automática del TXT
        txt_path = seleccionar_txt_auto()
        if not txt_path:
            print("No se encontró ningún archivo TXT para procesar. El programa se
cerrará.")
            exit()

        print(f"Documento cargado exitosamente. Se han procesado {len(txt_text)}
caracteres.")

        print("\nPreparado para responder sus consultas. Puede realizar preguntas
sobre el contenido del documento.")
        print("Para finalizar la sesión, escriba 'salir' cuando se le solicite una
pregunta.")

        while True:
```

```
pregunta = input("\n> ")
if pregunta.lower() == 'salir':
   print("\nFinalizando sesión. ¡Gracias por utilizar el sistema!")
   break

print("\nAnalizando consulta, por favor espere...")
respuesta = consultar_con_contexto(pregunta, txt_text)
print("\nRespuesta:")
print(respuesta)
```

Anexo E. Código testeo para Voz de Azure AI voz.py

```
[[#!/usr/bin/env python3

import os
from azure.cognitiveservices.spech import SpeehConfig, SpeechSynthesizer,
AudioConfig, ResultReason
import time

# Configuración del servicio de voz de Azure
speech_key = "2N37UG7pXFxZeD"  # Completa tu clave
speech_region = "eastus2"  # Región que aparece en tu portal

# Ruta donde guardar el archivo de audio
output_file = os.path.join(os.path.dirname(os.path.abspath(__file__)),
"prueba_voz.wav")

# Texto de prueba para convertir a voz
texto_prueba = """
Los encargados del área de titulación y sus respectivos cargos, según el archivo
proporcionado, son:

1. Ing. Luis Alberto Morales Perrazo - Presidente
2. Ing. Andrea Nataly Sánchez Solís - Secretaria
3. Ing. Clay Fernando Aldás Flores - Responsable de las carreras de Tecnologías
de la Información y Software
4. Ing. Edgar Patricio Córdova Córdova - Responsable de las carreras de
Telecomunicaciones
5. Ing. Daysi Margarita Ortiz Guerrero - Responsable de la Carrera de Ingeniería
Industrial

Esta información se encuentra listada en la sección "Responsables de la Unidad
de Titulación - Malla
"""

def text_to_speech(texto):
   try:
      print(f"Configurando servicio de voz con la región: {speech_region}")
      # Configurar voz en español
      speech_config.speech_synthesis_voice_name = "es-ES-ElviraNeural"  # Voz
femenina española

      # Configurar el archivo de salida
      audio_config = AudioConfig(filename=output_file)


      # Sintetizar texto a voz
      print("Convirtiendo texto a voz...")
      result = synthesizer.speak_text_async(texto).get()
```

```
            # Verificar resultado
            if result.reason == ResultReason.SynthesizingAudioCompleted:
                print(f"Texto convertido exitosamente a voz.")
                return True
            elif result.reason == ResultReason.Canceled:
                print(f"Síntesis cancelada: {result.cancellation_details.reason}")
                return False

        if success:
            print(f"Prueba completada exitosamente en {time.time() - start_time:.2f}
    segundos")
        else:
            print("La prueba falló. Verifica tus credenciales y configuración.")
```

# Anexo F. Datasheets teléfono IP Cisco SPA 303

| Descripción | Especificación |
|---|---|
| **Redes de datos** | • Dirección MAC (IEEE 802.3)<br>• IPv4 (RFC 791)<br>• Protocolo de resolución de direcciones (ARP)<br>• DNS: registro A (RFC 1706), registro SRV (RFC 2782)<br>• Cliente de Protocolo de Configuración Dinámica de Host (DHCP) (RFC 2131)<br>• Protocolo de Mensaje de Control de Internet (ICMP) (RFC 792)<br>• TCP (RFC 793)<br>• Protocolo de datagramas de Usuario UDP (RFC 768)<br>• Protocolo de tiempo real RTP (RFC 1889, 1890)<br>• Protocolo de control en tiempo real (RTCP) (RFC 1889)<br>• Protocolo de control en tiempo real – informe extendido (RTCP-XR) (RFC 3611)<br>• Servicios diferenciados (DiffServ) (RFC 2475)<br>• Tipo de servicio (ToS) (RFC 791, 1349)<br>• Etiquetado VLAN 802.1p/Q: calidad de servicio (QoS) de capa 2<br>• Protocolo Simple de Tiempo de Red (SNTP) (RFC 2030) |
| **Puerta de voz** | • SIP versión 2 (RFC 3261, 3262, 3263, 3264)<br>• SPCP con Cisco Unified Communications 500 Series<br>• Redundancia de proxy SIP: dinámica a través de DNS SRV, registros A<br>• Repetición de registro con servidor proxy SIP primario<br>• Soporte del protocolo SIP en redes NAT (incluido STUN)<br>• SIPFrag (RFC 3420)<br>• Llamada muy segura (cifrada) a través del protocolo seguro de transporte en tiempo real (SRTP)<br>• SIP/TLS<br>• Asignación de nombre de códec<br>• Algoritmos de voz<br>◦ G.711 (ley A y ley µ)<br>◦ G.726 (16/24/32/40 kbps)<br>◦ G.729 AB<br>◦ G.722<br>• Soporte de carga dinámica<br>• Tramas de voz ajustables por paquete<br>• Multifrecuencia bitono (DTMF), en banda y fuera de banda (RFC 2833) (SIP INFO)<br>• Soporte de plan de marcado flexible con temporizadores entre dígitos<br>• Soporte de dirección IP/marcado URI<br>• Generación de tono de progreso de llamada<br>• Búfer de fluctuación: adaptable<br>• Ocultación de pérdida de trama<br>• Detección de actividad de voz (VAD) con supresión de silencio<br>• Ajustes de atenuación/ganancia<br>• Tonos indicadores de mensajes en espera (MWI)<br>• Indicador de mensaje de voz en espera (VMWI), mediante NOTIFICAR, SUSCRIBIR<br>• Admite identificación de llamadas (nombre y número)<br>• Control de llamadas de terceros (RFC 3725) |

| Descripción | Especificación |
|---|---|
| **Aprovisionamiento, administración y mantenimiento** | • Administración y configuración basadas en la Web mediante servidor web integrado<br>• Configuración del teclado del teléfono mediante navegación por el menú de pantalla<br>• Aprovisionamiento y actualización automáticos mediante HTTPS, HTTP, TFTP<br>• Notificación asincrónica de disponibilidad de actualización mediante NOTIFICAR<br>• Actualizaciones en servicio no intrusivas<br>• Generación de informes y registro de eventos<br>• Transmisión de estadísticas en mensaje de DESPEDIDA<br>• RTCP-XR<br>• Registro de eventos del sistema (syslog) y registros del servidor de depuración: configurables por línea |
| **Fuente de alimentación** | • Tipo de conmutación (100–240V) automática<br>• Voltaje de CC de entrada: +5 VCC a 1,0A como máximo |
| **Interfaces físicas** | • Dos puertos Ethernet RJ-45 10/100BASE-T (IEEE 802.3)<br>• Teléfono: conector RJ-9<br>• Micrófono y altavoz integrados<br>• Puerto para auriculares: 2,5 mm |
| **Luces indicadoras/LED** | • Botón de encendido o apagado de altavoz con indicador LED<br>• Botón de encendido o apagado de auriculares con indicador LED<br>• Botón de silencio con indicador LED<br>• LED Indicador de mensajes en espera<br>• Función de prueba de indicadores LED |
| **Dimensiones (ancho x alto x profundidad)** | 8.66 x 7.80. x 1.18 pulg. (220 x 198 x 30 mm) |
| **Peso de la unidad** | 1.50 lb (0,68 kg) |
| **Temperatura de funcionamiento** | 32° ~ 113 °F (0° ~ 40 °C) |
| **Temperatura de almacenamiento** | –13° ~ 185 °F (–20° ~ 70 °C) |
| **Humedad de funcionamiento** | De 5 a 95%, sin condensación |
| **Humedad de almacenamiento** | De 5 a 95%, sin condensación |

Anexo G. Evaluación del sistema

La tabla 24 presenta los resultados de una prueba de evaluación del sistema automatizado, diseñado para responder preguntas frecuentes sobre el proceso de titulación en la Facultad de Ingeniería.

Tabla 25. Resultados de la evaluación del sistema.

| **N.º de prueba** | **Nombre del evaluador** | **Pregunta realizada** | **Respuesta esperada** | **Respuesta del sistema** | **¿Respondió correctamente? (Sí/No)** | **Porcentaje de precisión estimado (%)** | **Tiempo de respuesta (s)** | **Recomienda el programa (Si/ No/Talvez)** |
|---|---|---|---|---|---|---|---|---|
| 1 | Kevin Díaz | ¿Quién es el presidente de la Unidad de Titulación? | Ing. Luis Alberto Morales Perrazo. | Ing. Luis Alberto Morales Perrazo. | Si | 100% | 5s | Si |
| 2 | Darío Chuquitarco | ¿Hasta cuándo se entrega el informe final? | Hasta el 23 de junio de 2025. | El informe final de trabajo de titulación se debe entregar el 23 de junio de 2025. | Si | 100% | 5s | Si |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 3 | Fernando Madrid | ¿Qué modalidades de titulación existen? | Examen complexivo y trabajo de titulación. | Las modalidades de titulación son examen complexivo y trabajo de titulación. | Si | 100% | 6s | Si |
| 4 | Paul Gavidia | ¿Qué anexo se necesita para seleccionar la modalidad? | El Anexo 1 | Necesitas el Anexo 1 para seleccionar la modalidad de titulación | Si | 100% | 5s | Si |
| 5 | Josué Mongue | ¿Qué documento justifica que se trabaja en una empresa externa? | La carta de compromiso (Anexo 3). | Carta de compromiso institucional | Si | 90% | 10s | Tal vez |
| 6 | Jonathan Lalaleo | ¿Qué formato se usa para el informe mensual del tutor? | El Anexo 5 | Para el informe mensual necesira el Anexo 5 | si | 100% | 5s | Si |
| 7 | Kevin Chiliquinga | ¿Cuándo se realiza el examen complexivo? | Del 21 al 23 de julio de 2025 | Los examenes complexivos se desarrollarán Del 21 al 23 de julio de 2025 | Si | 95% | 7s | Si |
| 8 | Jonathan Abril | ¿Qué debe tener el informe final? | Portada, objetivos, metodología, resultados, conclusiones, etc. | El informe final portada, objetivos, metodología, | Si | 95% | 8s | Si |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | resultados, conclusione. | | | | |
| 9 | Mateo Llerena | ¿Qué documento se entrega para evaluar el informe final? | El Anexo 10. | El archivo no especifica el contenido del informe final | No | 0% | 6s | No |
| 10 | Fernando Cueva | ¿Desde que correo se deben enviar los documentos? | Desde el correo institucional. | Envia los documentos desde el correo institucional | Si | 90% | 10s | Si |
| 11 | Jonathan Ocaña | ¿Qué porcentaje de similitud es permitido por Turnitin? | Máximo 10%. | Puede tener hasta el 10% | Si | 100% | 5s | Si |
| 12 | Carlos Guerra | ¿Cuántos calificadores asigna el Consejo Directivo? | Dos calificadores. | Se asignan dos calificadores | Si | 100% | 6s | Si |
| 13 | Jhon Maldonado | ¿Qué pasa si el promedio del informe es menor a 7/10? | Realizan una reunion y se debe corregir el trabajo. | Se convoca a una reunión y se debe corregir el trabajo | Si | 90% | 10s | Si |
| 14 | Diego Villaroel | ¿Cuál es el anexo que regula la recorrección del trabajo? | El anexo posterior al 12, para promedio menor a 7/10. | El anexo 12 | Si | 90% | 5s | Si |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 15 | Lizbeth Núñez | ¿Qué documento certifica el 100% del trabajo realizado en empresa? | El informe de cumplimiento de la empresa. | El informe que emita la empresa | Si | 95% | 7s | Si |
| 16 | Darío Guamán | ¿Qué anexo se usa para evaluar artículos académicos? | El Anexo 13. | No fue encontrado | No | 0% | 6s | Tal vez |
| 17 | Santiago Martínez | ¿Cuál es el instructivo que regula el proceso? | CAU-P-681-2023. | El intructivo es CAU-P-681-2023. | Si | 100% | 5s | Si |
| 18 | Christian Morales | ¿Qué anexo regula el cambio de modalidad? | El Anexo 4. | Para el cambio de modalidad se necesita el anexo 4 | Si | 95% | 7s | Si |
| 19 | Melanie Calucho | ¿Qué se necesita para solicitar el certificado de aptitud legal? | Informe final, calificaciones, requisitos habilitantes, entre otros. | Para solicitar certificado de aptitud legal necesita Informe final, calificaciones, requisitos habilitantes, entre otros. | Si | 90% | 10s | Si |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 20 | Jessica López | ¿Que contiene el anexo 1? | Solicitud para seleccionar la modalidad y presentación de la propuesta del trabajo de titulación | El anexo 1 contine detalles sobre la solicitud para seleccionar la modalidad y presentación de la propuesta del trabajo de titulación. | Si | 95% | 15s | Si |