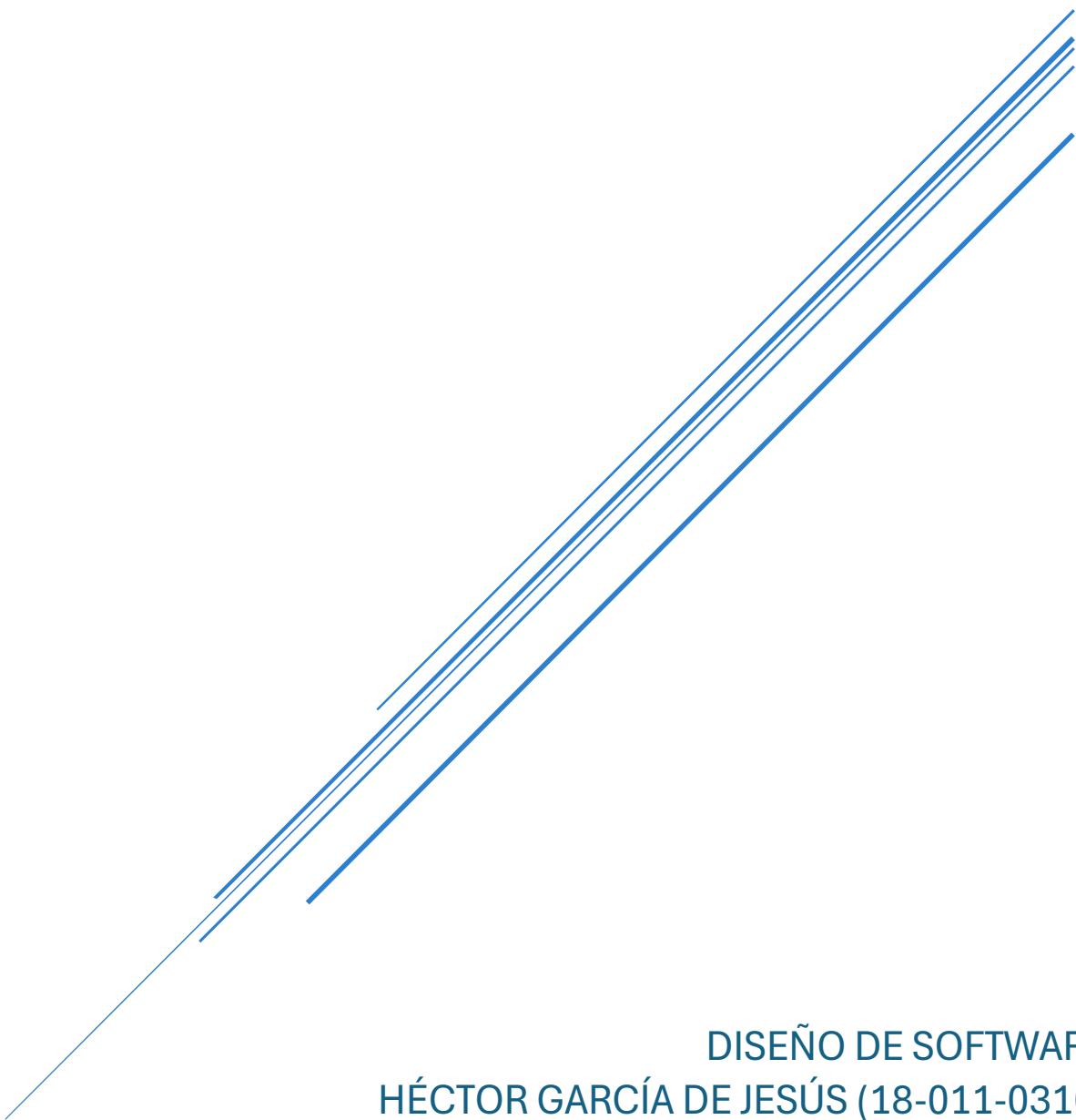


SDD

ESPECIFICACIÓN DE DISEÑO DE SOFTWARE



DISEÑO DE SOFTWARE
HÉCTOR GARCÍA DE JESÚS (18-011-0310)

Índice

Índice de ilustraciones	3
1. Introducción.....	4
2. Propósito.....	5
3. Alcance.....	5
4. Punto de vista de contexto	6
4.1 Problemas de diseño	6
4.2 Elementos de diseño.....	7
4.3 Idiomas de ejemplos.....	8
5. Punto de vista de la composición	14
5.1 Problemas de diseño	14
5.2 Elementos de diseño.....	14
5.2.1 Atributo de función	15
5.2.2 Atributo de subordinados.....	15
5.3 Idiomas de ejemplos.....	16
6. Punto de vista lógico	20
6.1 Problemas de diseño	20
6.2 Elementos de diseño.....	21
6.3 Idiomas de ejemplos.....	22
7. Puntos de vista de la dependencia.....	25
7.1 Problemas de diseño	26
7.2 Elementos de diseño.....	27
7.3 Atributo dependencias.....	27
7.4 Idiomas de ejemplo	28
8. Puntos de vista de la información	32
8.1 Problemas de diseño	32
8.2 Elementos de diseño.....	34
8.3 Atributo de datos	35
8.4 Idiomas de ejemplo	35
9. Punto de vista del uso de patrones	37
9.1 Problemas de diseño	37

9.2	Elementos de diseño.....	38
9.3	Idiomas ejemplo	40
10.	Punto de vista de la interfaz.....	43
10.1	Problemas de diseño	45
10.2	Elementos de diseño.....	46
10.3	Atributos de la interfaz	47
10.4	Idiomas de ejemplo	47
11.	Punto de vista de estructura.....	53
11.1	Problemas de diseño	55
11.2	Elementos de diseño.....	55
11.3	Idiomas de ejemplo	57
12.	Punto de vista de interacción	60
12.1	Problemas de diseño	61
12.2	Elementos de diseño.....	62
12.3	Idiomas de ejemplo	64
13.	Punto de vista dinámica de estados.....	69
13.1	Problemas de diseño	69
13.2	Elementos de diseño.....	70
13.3	Idiomas de ejemplo	72
14.	Punto de vista del algoritmo	77
14.1	Problemas de diseño	77
14.2	Elementos de diseño.....	78
14.3	Idiomas de ejemplos.....	81

Índice de ilustraciones

<i>Ilustración 1 caso de uso general</i>	9
<i>ilustración 2 caso de uso de estudiante</i>	10
<i>ilustración 3 caso de uso de administrador</i>	11
<i>ilustración 4 diagrama de contexto</i>	12
<i>ilustración 5 diagrama de paquetes</i>	16
<i>ilustración 6 diagrama de componentes 1</i>	17
<i>ilustración 7 diagrama de despliegue</i>	18
<i>ilustración 8 diagrama de clases</i>	22
<i>ilustración 9 diagrama de clases en paquetes</i>	22
<i>ilustración 10 diagrama de objetos</i>	23
<i>ilustración 11 diagrama de bloques 2</i>	28
<i>ilustración 12 diagrama de bloques 1</i>	28
<i>ilustración 13 diagrama de componentes 2</i>	30
<i>ilustración 14 patrón singleton</i>	40
<i>ilustración 15 patrón method factory</i>	41
<i>ilustración 16 patrón proxy</i>	42
<i>ilustración 17 diagrama de componentes 3</i>	47
<i>ilustración 18 interfaz principal</i>	50
<i>ilustración 19 interfaz secundaria</i>	51
<i>ilustración 20 interfaz de soporte</i>	52
<i>ilustración 21 diagrama de clases 2</i>	57
<i>ilustración 22 diagrama se secuencia 1</i>	64
<i>ilustración 23 diagrama de secuencia 2</i>	65
<i>ilustración 24 diagrama de secuencia 3</i>	66
<i>ilustración 25 diagrama de secuencia 4</i>	67
<i>ilustración 26 diagrama de estados 1</i>	73
<i>ilustración 27 diagrama de estados 2</i>	75
<i>ilustración 28 diagrama de decisiones</i>	81

1. Introducción

En el presente documento se detalla las especificaciones, así como los requisitos de un sistema para la creación de un "**Chatbot**", diseñado específicamente para brindar apoyo a la comunidad estudiantil de la Universidad Autónoma de la Ciudad de México (UACM). Este sistema busca optimizar la gestión de consultas y trámites dentro de la institución, proporcionando una herramienta tecnológica que agilice la comunicación entre los estudiantes, así como las áreas administrativas de la universidad. La implementación de esta solución responde a la creciente necesidad de modernización en la atención estudiantil, permitiendo una interacción rápida, eficiente además de accesible en cualquier momento.

Por otro lado, se pretende mediante un sistema automatizado de atención, reducir la carga de trabajo de los departamentos administrativos, facilitando el acceso a información relevante sin necesidad de intervención humana constante. Entre las funciones principales del sistema, se encuentran la gestión de solicitudes relacionadas con trámites académicos, información sobre fechas importantes, orientación sobre procedimientos administrativos y respuesta a dudas generales de los estudiantes. El sistema estará diseñado para operar con inteligencia artificial por ende se busca un procesamiento de lenguaje natural, lo que permitirá una interacción fluida además de contextualizada con los usuarios. De este modo, se buscará garantizar que el Chatbot cuente con una interfaz intuitiva, accesible como un sitio web institucional o aplicaciones móviles. De esta manera, los estudiantes podrán acceder al servicio de manera sencilla y sin restricciones de horario. Desde una perspectiva técnica, el proyecto se desarrollará siguiendo las mejores prácticas en ingeniería de software, asegurando que sea seguro, para ello, se ha adoptado el estándar IEEE 830-1998, el cual establece lineamientos para la especificación de requisitos de software. Este estándar proporciona un marco estructurado para definir de manera clara además de precisa para los requisitos funcionales y no funcionales del sistema, facilitando su desarrollo e implementación efectiva dentro del entorno universitario.

En conclusión, este documento describe de manera detallada los requisitos, características del sistema propuesto, abordando aspectos técnicos, funcionales y operativos. Con ello, se busca garantizar que el desarrollo cumpla con los objetivos establecidos y brinde un valor agregado a la comunidad universitaria, impulsando la transformación digital en la UACM.

2. Propósito

El propósito de este documento es definir los requisitos funcionales y no funcionales del **Chatbot de Atención Universitaria para la Universidad Autónoma de la Ciudad de México (UACM)**. Este chatbot está diseñado para asistir a la comunidad estudiantil en la gestión de trámites, resolución de dudas y optimización del acceso a información institucional, con el fin de mejorar la eficiencia de los servicios administrativos y así como académicos dentro de la misma. El sistema proporcionará a los estudiantes una herramienta de comunicación automatizada, accesible a través de distintos canales digitales, que permitirá obtener respuestas rápidas además de precisas sobre procedimientos administrativos, requisitos de inscripción, información académica y otros temas de interés. Este documento sigue el estándar **IEEE 830-1998**, el cual establece las mejores prácticas para las especificaciones de requisitos de software (**SRS**), asegurando que el desarrollo sea claro, estructurado y alineado con las necesidades de los usuarios y los objetivos institucionales de la UACM.

3. Alcance

El sistema está principalmente enfocado en la atención a estudiantes de la **Universidad Autónoma de la Ciudad de México (UACM)**, facilitando el acceso a información además de agilizar la gestión de trámites administrativos a través de un chatbot. Este proyecto tiene contemplados los siguientes puntos:

Consultas generales: Responderá preguntas frecuentes sobre trámites administrativos, calendarios académicos, requisitos de inscripción.

Información académica: Proporcionará detalles sobre horarios de clases, disponibilidad de materias. **Canales de comunicación:** Será accesible a través de la página web de la UACM.

El chatbot será gestionado y supervisado por los administradores de la UACM, quienes podrán actualizar la base de conocimientos y mejorar la precisión de las respuestas. **En la interfaz de "Consulta de trámites"**, los estudiantes podrán obtener información detallada sobre procesos administrativos, pero no podrán realizar modificaciones en sus registros. **En el apartado "Consultar información académica" solo se mostrarán los datos disponibles en la base de datos del chatbot**, sin opción de edición por parte de los usuarios.

4. Punto de vista de contexto

El chatbot de la **UACM** está diseñado para interpretar y responder preguntas dentro de un contexto académico, así como administrativo. Su funcionalidad depende de la interacción con los usuarios (estudiantes, personal administrativo y terceros), además de la información contenida en su base de datos. El sistema actúa como un intermediario, proporcionando respuestas en función del contexto en el que se realicen las consultas.

4.1 Problemas de diseño

Uno de los principales retos es asegurar que el chatbot comprenda correctamente las consultas y proporcione respuestas precisas dentro de su contexto. Los problemas pueden surgir cuando:

- **Ambigüedad en las preguntas:** Los usuarios pueden formular consultas de manera imprecisa, lo que dificulta la interpretación del chatbot.
- **Limitación del conocimiento:** El sistema solo puede responder con base en la información disponible en su base de datos, lo que restringe su alcance.
- **Caja negra del sistema:** Los usuarios pueden conocer las respuestas proporcionadas, pero no el funcionamiento interno del chatbot (algoritmos y procesamiento de lenguaje natural).
- **Acceso restringido a ciertos datos:** El chatbot no podrá proporcionar información confidencial, como calificaciones o datos personales.

Para mitigar estos problemas, se implementarán mejoras en el procesamiento del lenguaje natural, supervisión del contenido y actualización periódica de la base de conocimientos. **Medidas de mitigación:**

- Mejora continua del motor NLP para interpretar mejor el lenguaje natural.
- Capacitación del chatbot con nuevas categorías de preguntas reales detectadas.
- Validación contextual mediante palabras clave y reconocimiento de intenciones.

- Actualización periódica de la base de conocimientos con fuentes confiables de la universidad.

4.2 Elementos de diseño

A continuación, se presentan los elementos fundamentales que definen el diseño contextual del chatbot, enfocados en garantizar que el sistema responda adecuadamente según el tipo de consulta y usuario:

1. Entidades participantes

- **Usuario**

Puede ser estudiante, administrativo o visitante externo. Interactúa con el sistema mediante texto, buscando resolver dudas o recibir orientación.

- **Chatbot (Agente conversacional)**

Componente central que procesa la entrada del usuario, analiza la intención, consulta la base de datos de respuestas y devuelve una respuesta adecuada.

- **Base de conocimientos**

Conjunto estructurado de respuestas prediseñadas, categorías de intención, palabras clave y datos públicos de la universidad. Se actualiza regularmente según los cambios institucionales.

2. Categorías de contexto

El chatbot identifica automáticamente el **tipo de contexto** al que pertenece la consulta. Algunas de las principales categorías contextuales son:

- Académico: horarios de clases, reinscripciones, servicios escolares.
- Administrativo: ubicación de oficinas, trámites, contacto institucional.

- Técnico: fallos de acceso, correo institucional, problemas con plataformas.
- Informativo: eventos, calendarios, noticias institucionales.

3. Módulos de control de contexto

- **Módulo de intención**

Clasifica las consultas según patrones y palabras clave. Decide el camino de la conversación.

- **Módulo de validación**

Revisa si la información pedida está permitida o es confidencial. Si no puede responder, emite un mensaje de advertencia y sugiere contacto humano.

- **Módulo de respuesta dinámica**

Adapta la salida textual del chatbot según el contexto del mensaje, evitando respuestas genéricas o repetitivas.

4. Políticas de privacidad y restricción contextual

- El sistema **no almacena datos** de los usuarios.
- No responde sobre información confidencial.
- No solicita autenticación para mantener conversaciones comunes.
- Las respuestas están basadas en **contenido público institucional**.

4.3 Idiomas de ejemplos

A continuación, se visualiza el diagrama de casos de uso e uso general, en el cual se encontrarán los actores y las funciones que estos van a realizar y los roles que tendrá cada uno.

Diagrama de casos de usos general

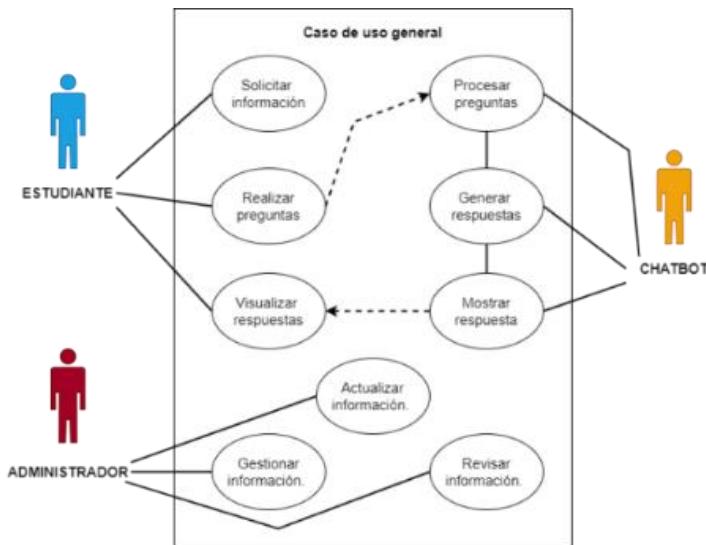


ILUSTRACIÓN 1 CASO DE USO GENERAL

Tabla de caso de uso

Caso de Uso	Actores	Descripción
Solicitar información	Estudiante	El actor solicita información relevante.
Realizar preguntas	Estudiante	El actor formula preguntas para resolver dudas específicas.
Visualizar respuesta	Estudiante	El actor observa las respuestas generadas por el sistema.
Procesar preguntas	Sistema	El sistema analiza y da seguimiento a las preguntas enviadas.
Generar respuestas	Sistema	El sistema crea respuestas basadas en las preguntas procesadas.
Mostrar respuesta	Sistema	El sistema presenta las respuestas generadas al actor.
Actualizar información	Administrador	El sistema realiza cambios en la información existente.
Gestionar información	Administrador	El actor administra y organiza la información disponible.
Revisar información	Administrador	El actor verifica la información que ha sido actualizada.

Diagrama de casos de usos del estudiante

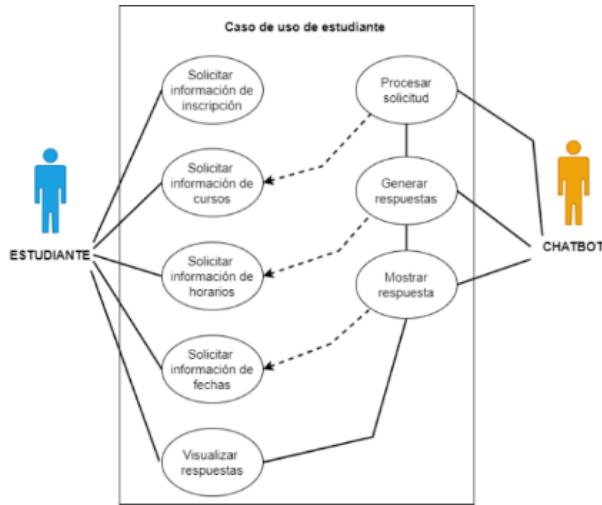


ILUSTRACIÓN 2 CASO DE USO DE ESTUDIANTE

Tabla de caso de uso

Caso de Uso	Actor	Descripción
Solicitar info de inscripción	Estudiante	El estudiante solicita datos sobre los requisitos
Solicitar info de cursos	Estudiante	El estudiante busca detalles sobre los cursos disponibles
Solicitar info de horarios	Estudiante	El estudiante consulta horarios específicos de los cursos.
Solicitar info de fechas	Estudiante	El estudiante pregunta por fechas importantes.
Visualizar respuestas	Estudiante	El estudiante recibe y revisa la información proporcionada
Procesar solicitud	Chatbot	El chatbot analiza las solicitudes realizadas.
Generar respuestas	Chatbot	El chatbot crea respuestas basadas en la información
Mostrar respuesta	Chatbot	El chatbot presenta la información generada.

Diagrama de casos de usos del administrador

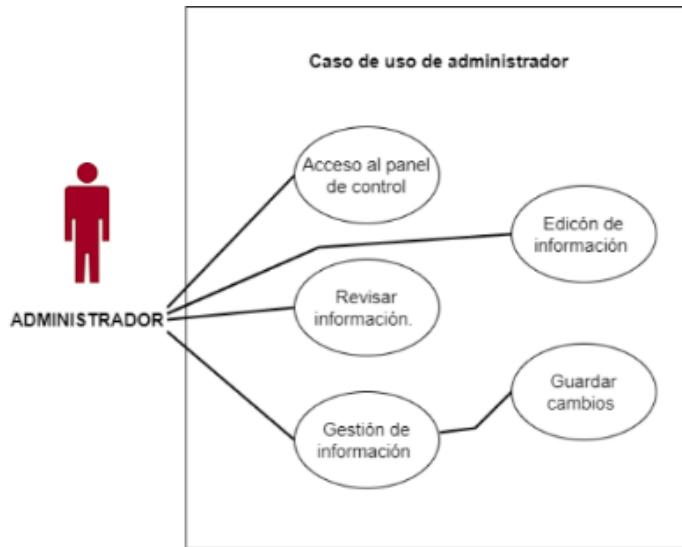


ILUSTRACIÓN 3 CASO DE USO DE ADMINISTRADOR

Tabla de caso de uso

Caso de Uso	Actor	Descripción
Acceso al panel de control	Administrador	El administrador accede al panel de control para gestionar las operaciones del sistema.
Edición de información	Administrador	El administrador realiza modificaciones en los datos disponibles.
Revisar información	Administrador	El administrador verifica la información almacenada
Guardar cambios	Administrador	El administrador guarda los cambios realizados en el sistema.
Gestión de información	Administrador	El administrador organiza y supervisa la información del sistema.

Para la representación del sistema se utilizarán el **Diagrama de contexto**:

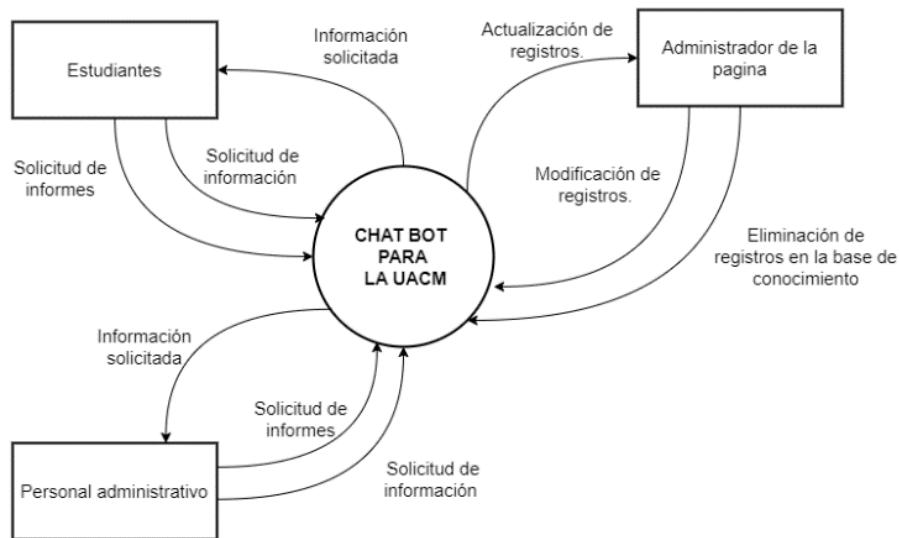


ILUSTRACIÓN 4 DIAGRAMA DE CONTEXTO

Núcleo del diagrama: CHAT BOT PARA LA UACM

Este círculo central representa el sistema principal. Es el chatbot que interactúa tanto con los usuarios comunes (como estudiantes y personal administrativo) como con los usuarios con privilegios especiales (administrador de la página). Este chatbot tiene la función de procesar solicitudes, devolver respuestas, y gestionar la información dentro de su base de datos.

Estudiantes

Interacciones con el chatbot:

- Solicitud de información: También pueden hacer preguntas generales relacionadas con servicios académicos.
- Información solicitada (respuesta): El chatbot responde entregando la información requerida, en función de su base de datos y capacidades de procesamiento de lenguaje natural.

Personal administrativo

Interacciones con el chatbot:

- Muy similar a la de los estudiantes:
 - Solicitan informes y consultan información administrativa.
 - Reciben respuestas automáticas del chatbot con base en las consultas realizadas.

Esto permite que el personal acceda rápidamente a datos sin tener que contactar directamente a otro departamento.

Administrador de la página

Este actor tiene un rol más técnico y de mantenimiento del sistema. Sus interacciones incluyen:

- Actualización de registros: El administrador puede añadir nueva información al sistema, como procedimientos, nuevas preguntas frecuentes, etc.
- Modificación de registros: Puede corregir o ajustar datos existentes si hay errores o cambios en los procesos internos de la universidad.
- Eliminación de registros: Tiene la autoridad para quitar información obsoleta o incorrecta de la base de conocimiento del chatbot.

Este actor no solo interactúa con el sistema, sino que lo mantiene actualizado, garantizando que las respuestas del chatbot sean correctas y actuales.

Este diagrama muestra una visión general y simplificada del flujo de información entre los usuarios y el chatbot, así como entre el chatbot y el administrador del sistema. Refleja cómo el chatbot centraliza la comunicación institucional, facilitando el acceso a información de forma automatizada y permitiendo que el sistema sea administrado de manera eficiente.

5. Punto de vista de la composición

El chatbot de la UACM está compuesto por diferentes módulos que trabajan en conjunto para proporcionar información académica como administrativa. Su diseño se basa en principios de escalabilidad además de modularidad, permitiendo futuras actualizaciones e integración de nuevos servicios.

5.1 Problemas de diseño

Los problemas en el diseño del chatbot pueden surgir en la estructuración del software, así como en la asignación de roles, afectando su funcionamiento y la experiencia del usuario. Para mitigar estos desafíos, se aplicarán estrategias de mantenimiento de software y reutilización de código, lo que facilitará la evolución del sistema.

Entre los problemas más relevantes se encuentran:

- **Problema de vinculación de datos en la base de conocimientos**, afectando la precisión de las respuestas.
- **Fallas en la interpretación del lenguaje natural**, dificultando la comprensión de ciertas consultas.
- **Interrupciones en la conectividad con plataformas externas**, limitando su disponibilidad.
- **Errores en la extracción de información en tiempo real**, lo que podría generar respuestas desactualizadas.

La planificación y el monitoreo constante ayudarán a reducir estos riesgos y mejorar la eficiencia del chatbot.

5.2 Elementos de diseño

El chatbot estará compuesto por distintos elementos organizados en **subsistemas, módulos y componentes**, que incluyen:

- **Módulo de procesamiento de lenguaje natural (PLN):** Interpreta preguntas y genera respuestas en lenguaje claro.
- **Módulo de base de conocimientos:** Contiene información sobre trámites, horarios y normativas.
- **Módulo de interacción:** Gestiona la comunicación con el usuario a través de la web.
- **Módulo de gestión:** Permite a los administradores actualizar la base de conocimientos.

Para la gestión del código y su evolución, se usará **GitHub** como repositorio, permitiendo control de versiones.

5.2.1 Atributo de función

Cada usuario tendrá acceso a funciones específicas dentro del sistema:

- **Estudiantes y terceros:** Podrán consultar información sobre trámites y normativas.
- **Administradores del chatbot:** Podrán supervisar el rendimiento del sistema y corregir posibles errores en las respuestas.

5.2.2 Atributo de subordinados

Los componentes del chatbot estarán organizados por procesos e interfaces diferenciadas:

- **Gestión de información académica:**
 - Consulta de horarios.
 - Disponibilidad de materias.
- **Gestión de trámites administrativos:**
 - Información sobre inscripciones.

- Requisitos y documentos necesarios.
- **Supervisión del chatbot:**
 - Monitoreo del rendimiento.
 - Actualización de la base de conocimientos.

Cada módulo cumplirá una función específica, asegurando un diseño estructurado y eficiente.

5.3 Idiomas de ejemplos

Diagrama de paquetes con la modelo vista controlador

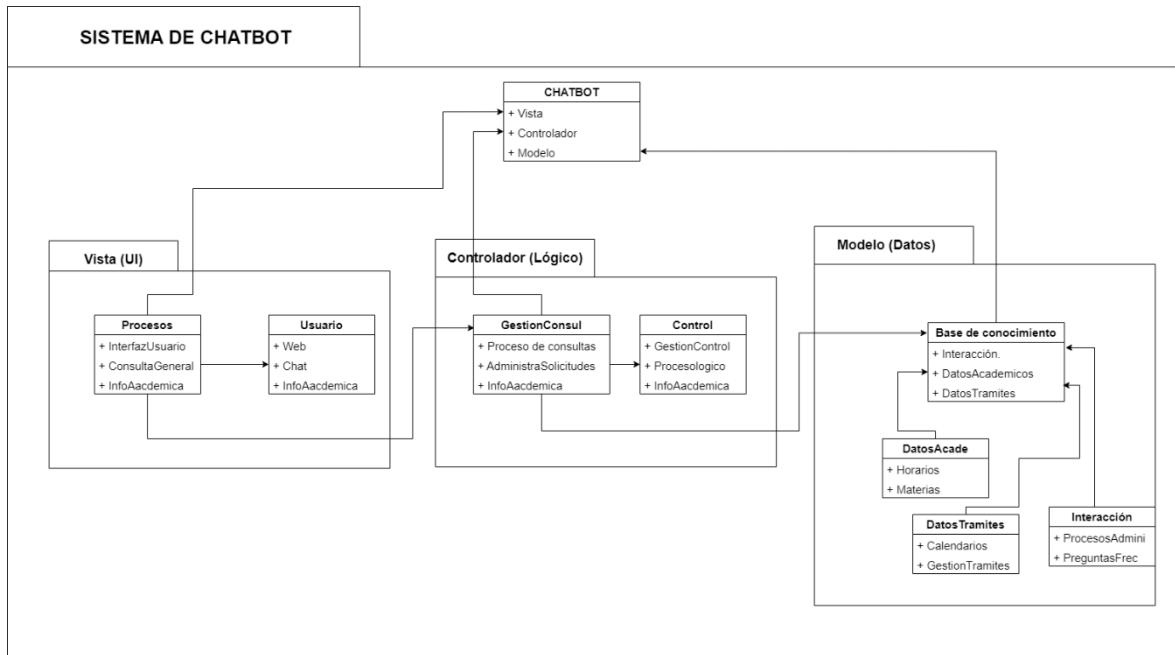


ILUSTRACIÓN 5 DIAGRAMA DE PAQUETES

Explicación de la estructura del diagrama:

- **Paquetes:** Los elementos del sistema están agrupados en paquetes, que son:

- **Vista (UI):** Contiene los elementos de la interfaz de usuario, como la Web y el Chatbot.
- **Controlador (Lógica):** Aquí están los componentes que gestionan las consultas, como **GestionConsultas**, **ProcesamientoLogico**.
- **Modelo (Datos):** Este paquete contiene los componentes relacionados con los datos, como **BaseConocimientos**, **DatosAcademicos** y **DatosAdministrativos**.
-
- **Relaciones:**
 - **Vista a Controlador:** Los componentes de la Vista (Web, Chatbot) se conectan a **GestionConsultas** en el Controlador.
 - **Controlador a Modelo:** **GestionConsultas** interactúa con varios componentes del Modelo.

Diagrama de componentes

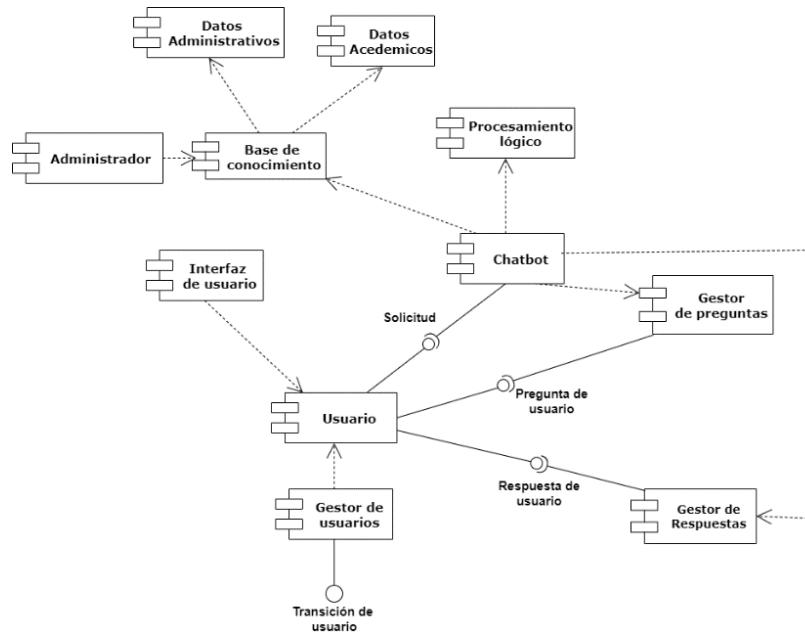


ILUSTRACIÓN 6 DIAGRAMA DE COMPONENTES 1

Explicación del Diagrama:

- **Usuario:** Es el componente con el cual se desarrolla el usuario.
- **Gestor de usuarios:** Componente que interactúa con el usuario.
- Puede estar implementada en **una página web o un chatbot en una aplicación de mensajería.**
- **Gestor de Consultas:** Se encarga de recibir las preguntas del usuario.
- **Procesamiento Logico:** Decide cómo manejar la consulta y envía la información correcta.
- **BaseConocimientos:** Contiene **FAQ** y respuestas predefinidas.
- **DatosAcademicos y DatosAdministrativos:** Almacenan datos sobre **horarios, trámites e inscripciones.**

Diagrama de despliegue

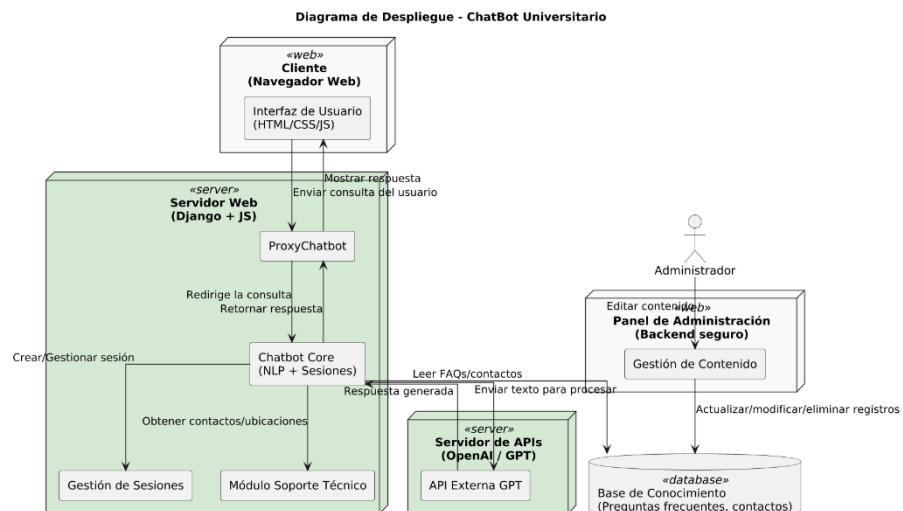


ILUSTRACIÓN 7 DIAGRAMA DE DESPLIEGUE

Explicación del Diagrama:

Usuario

- **Accede al sistema** mediante **un navegador web**.
- Su dispositivo **envía una solicitud HTTP** al **Servidor Web**.

Servidor Web

- **Es el intermediario** entre el usuario y el backend.
- **Maneja solicitudes HTTPS** y redirige la consulta al **Servidor de Aplicación**.

CHATBOT

- Ejecuta la **lógica de negocio del Chatbot** (procesa consultas, decide respuestas, etc.).
- Se comunica con:
 - **La Base de Datos** (para obtener información académica y administrativa).
 - **La API de IA/NLP** (para interpretar preguntas del usuario).

Servidor de Base de conocimientos

- **Almacena información** de usuarios, consultas, preguntas frecuentes, etc.
- Es consultado por el **Servidor de Aplicación** cuando necesita información.

6. Punto de vista lógico

El propósito del diseño lógico del chatbot es definir correctamente sus módulos, así como sus relaciones, asegurando que cada componente cumpla con su función dentro del sistema.

Módulos principales:

1. **Módulo de preguntas generales:** Responde preguntas frecuentes sobre trámites, calendarios y requisitos de inscripción.
2. **Módulo principal de preguntas académicas:** Proporciona detalles sobre horarios, disponibilidad de materias y otros aspectos académicos.
3. **Módulo de gestión y actualización:** Administradores de la UACM podrán actualizar la base de conocimientos para mejorar la precisión del chatbot.
4. **Módulo de integración:** Facilita la conexión con la página web de la UACM.
5. **Módulo de soporte técnico:** Dentro de este apartado, se encuentra la información para reportar los fallos posibles del sistema.

Estos módulos trabajan en conjunto para optimizar la experiencia del usuario y la eficiencia en la gestión de información.

6.1 Problemas de diseño

Uno de los desafíos en el diseño del chatbot es garantizar que su estructura sea **modular, escalable, así como flexible**, permitiendo futuras actualizaciones o mejoras dentro del sistema. El problema principal radica en seleccionar los métodos y algoritmos adecuados para mejorar su capacidad de respuesta y adaptación a nuevas consultas.

Otro reto importante es **minimizar sesgos** en la interpretación de preguntas y asegurar que el sistema no genere respuestas incorrectas o confusas. Para ello, se implementarán estrategias de aprendizaje manual, así como la supervisión por parte de los administradores.

6.2 Elementos de diseño

Entidades de Diseño:

- **Consulta Trámites:** Maneja información sobre cursos, profesores, consultas sobre otros trámites administrativos de interés.
- **ConsultaAcadémica:** Proporciona detalles sobre materias, horarios y profesores.
- **Usuario:** Representa a estudiantes, personal administrativo y terceros que consultan el chatbot.
- **Administrador:** Se encarga de actualizar la base de conocimientos y supervisar el rendimiento del chatbot.

Relaciones de Diseño:

Las entidades de **ConsultaTrámites** y **ConsultaAcadémica** están vinculadas a **Usuario**, quien puede hacer preguntas y recibir respuestas del chatbot. La entidad **Administrador** tiene permisos especiales para modificar la base de conocimientos para mejorar las respuestas.

Restricciones de Diseño:

- El chatbot no podrá modificar registros académicos ni inscripciones.
- No tendrá acceso a información confidencial, como calificaciones o datos bancarios.
- No enviará notificaciones proactivas, funcionando solo bajo demanda.

6.3 Idiomas de ejemplos

Diagrama de clases

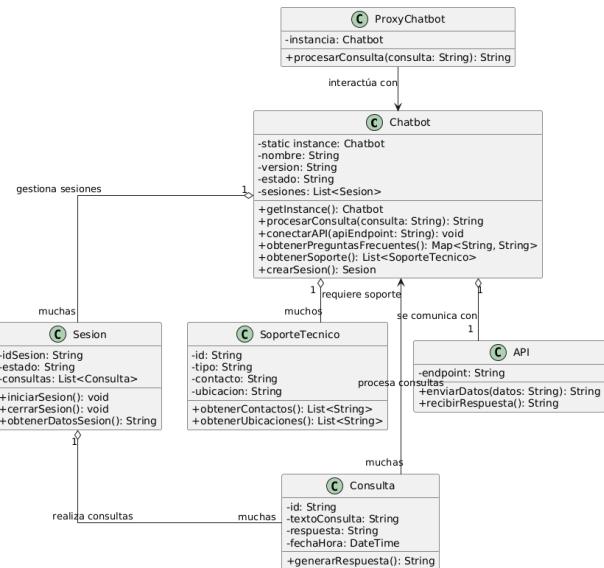


ILUSTRACIÓN 8 DIAGRAMA DE CLASES

Diagrama de clases en paquetes

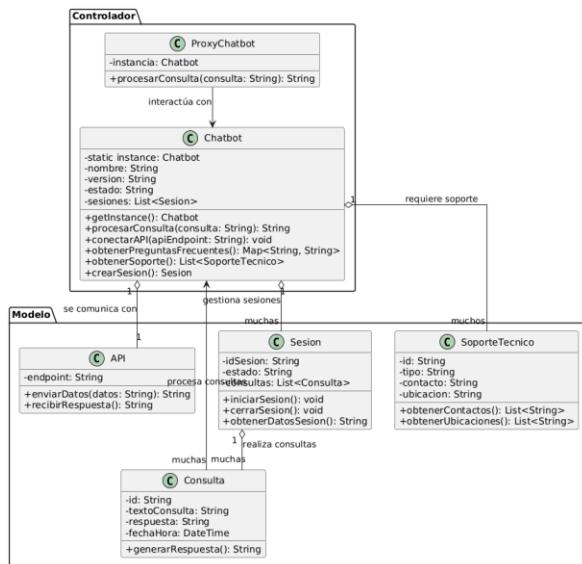


ILUSTRACIÓN 9 DIAGRAMA DE CLASES EN PAQUETES

El diagrama representa un sistema de Chatbot basado en el patrón MVC.

- Chatbot (Singleton): Clase principal que gestiona sesiones, procesa consultas y se comunica con una API externa y soporte técnico.
- Sesión: Representa una interacción con el usuario y almacena las consultas realizadas.
- Consulta: Contiene la información de cada pregunta y su respuesta generada.
- SoporteTecnico: Proporciona contactos y ubicaciones de asistencia.
- API: Facilita la comunicación con sistemas externos.
- ProxyChatbot (Controlador): Intermediario que optimiza el acceso al chatbot.

Relaciones clave:

Chatbot administra múltiples Sesión, que contienen varias Consulta.

Chatbot interactúa con API y SoporteTecnico.

ProxyChatbot delega consultas al Chatbot.

Diagrama de objetos

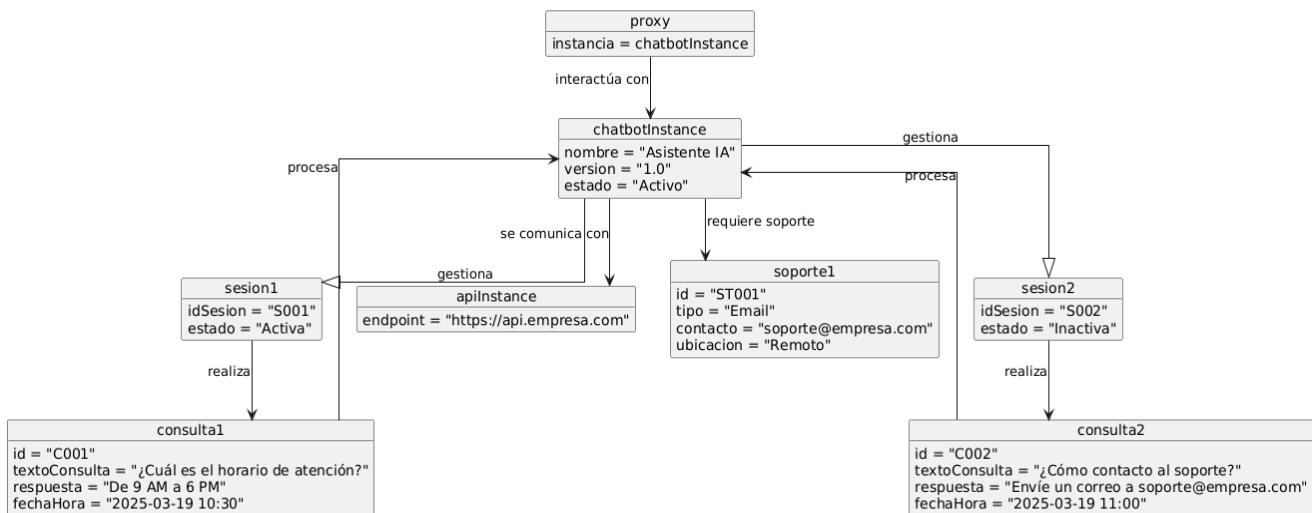


ILUSTRACIÓN 10 DIAGRAMA DE OBJETOS

El diagrama de objetos representa una **instancia en ejecución** del sistema, mostrando cómo interactúan sus componentes en un escenario real.

Objetos principales:

- chatbotInstance: Representa la única instancia del **Chatbot (Singleton)** con estado activo.
- proxyInstance: Es el **ProxyChatbot**, que delega consultas al chatbotInstance.
- apilnstance: Simboliza la API con la que se comunica el chatbot.
- soporte1 y soporte2: Representan los servicios de **soporte técnico** disponibles (email y teléfono).

Sesiones y consultas:

- sesion1 (activa) y sesion2 (inactiva) representan **interacciones de usuarios** con el chatbot.
- consulta1 y consulta2 son preguntas realizadas por los usuarios, con sus respectivas respuestas.

Relaciones clave:

- proxyInstance usa chatbotInstance para procesar consultas.
- chatbotInstance gestiona múltiples **sesiones** y se comunica con la **API** y el **soporte técnico**.
- Cada **sesión** tiene consultas asociadas y puede requerir **soporte técnico**.
- Las **consultas** interactúan con el chatbotInstance y pueden enviar datos a la **API**.

7. Puntos de vista de la dependencia

La arquitectura del software de nuestro proyecto proporciona una vista general de la estructura del **chatbot de asistencia para la UACM**, mostrando cómo los diferentes componentes interactúan y dependen unos de otros para su funcionamiento. Este enfoque permite comprender las interdependencias entre los módulos clave del sistema, facilitando su diseño, desarrollo y mantenimiento.

Para ilustrar la estructura del chatbot, se utilizará un **diagrama de bloques** que representará visualmente los principales componentes del sistema y sus conexiones. Esto permitirá una mejor comprensión del flujo de información dentro del chatbot y su integración con los servicios de la universidad.

Componentes Principales del Sistema

1. Interfaz de Usuario:

- Permite la interacción con los usuarios a través de la web
- Conecta con el motor de procesamiento para interpretar consultas.

2. Procesamiento de Lenguaje Natural (PLN):

- Analiza las preguntas de los usuarios y las convierte en solicitudes estructuradas.
- Depende de la base de conocimientos para generar respuestas adecuadas.

3. Base de Conocimientos:

- Contiene información sobre trámites, calendarios, requisitos de inscripción y normativas académicas.
- Se actualiza periódicamente por los administradores del chatbot.

4. Módulo de Consultas Académicas:

- Proporciona información sobre horarios de clases, disponibilidad de materias y reglamentos.

- Accede a la base de conocimientos para recuperar datos actualizados.

5. Módulo de Consultas Administrativas:

- Responde preguntas sobre inscripciones, pagos, becas y otros trámites.
- Se vincula con fuentes oficiales de la UACM para mantener la precisión de la información.

6. Panel de Administración:

- Permite a los administradores supervisar, actualizar y mejorar la base de conocimientos.
- Facilita el monitoreo del chatbot y la optimización de su desempeño.

7.1 Problemas de diseño

En el desarrollo del Chatbot de Atención Universitaria de la UACM pueden surgir problemas de diseño que afectan su funcionalidad y eficiencia. Uno de los principales problemas es la complejidad excesiva del sistema, lo que puede dificultar su mantenimiento y escalabilidad. Para evitar esto, se debe estructurar el chatbot de manera modular, eliminando componentes innecesarios.

Otro problema es la falta de etiquetas claras en las respuestas y opciones de menú, lo que puede generar confusión en los usuarios. Es importante definir un formato estándar para los mensajes y garantizar que las respuestas sean precisas y comprensibles. Las conexiones confusas entre los módulos del chatbot pueden llevar a errores en las respuestas o a la pérdida de datos. Para solucionar esto, se requiere un diseño estructurado del flujo de interacciones y establecer reglas claras para el procesamiento de consultas.

También existe el problema de una base de conocimientos limitada o desactualizada, lo que puede afectar la precisión de las respuestas del chatbot. Para garantizar que el sistema ofrezca información confiable, se debe implementar un mecanismo de actualización periódica y aprendizaje basado en interacciones previas.

La falta de integración con otros sistemas de la UACM representa otro desafío, ya que el chatbot debe conectarse con bases de datos académicas y administrativas para proporcionar información en tiempo real. Es fundamental establecer conexiones seguras y eficientes con los sistemas institucionales.

7.2 Elementos de diseño

El diseño del Chatbot de Atención Universitaria debe ser modular y eficiente, asegurando una arquitectura clara y funcional. El sistema debe estar compuesto por diferentes módulos, como el procesamiento de lenguaje natural, la gestión de respuestas, la base de datos y la interfaz de usuario. Para mejorar la interacción, el chatbot debe contar con una interfaz accesible desde la página web de la UACM y aplicaciones de mensajería, con opciones intuitivas que permitan a los estudiantes encontrar información de manera rápida.

La base de conocimientos del chatbot debe estar bien organizada y permitir actualizaciones frecuentes para reflejar cambios en los procedimientos administrativos y académicos. Es fundamental garantizar la seguridad de la información, limitando el acceso del chatbot a datos públicos y evitando el manejo de información confidencial de los estudiantes. Se debe incluir autenticación para ciertos procesos sensibles y cifrado en la transmisión de datos. El sistema debe contar con un mecanismo de mejora continua, basado en el análisis de las consultas de los usuarios. A partir de estos datos, se podrán realizar ajustes en la base de conocimientos y en el procesamiento del lenguaje natural para optimizar la precisión de las respuestas.

7.3 Atributo dependencias

El chatbot de atención universitaria depende de diferentes factores para su correcto funcionamiento. En términos de hardware, requiere un servidor que permita alojar el sistema y gestionar múltiples consultas simultáneamente, así como almacenamiento en la nube para guardar interacciones y actualizaciones de la base de conocimientos. En cuanto al software, el chatbot necesita un motor de procesamiento de lenguaje

natural para interpretar preguntas y generar respuestas precisas. Además, debe contar con una base de datos que almacene la información académica y administrativa utilizada en las respuestas.

Las dependencias entre módulos también juegan un papel clave en el diseño del sistema. El módulo de procesamiento de lenguaje natural debe estar conectado con el módulo de gestión de respuestas, que a su vez consulta la base de conocimientos para proporcionar información actualizada. Para mejorar su funcionalidad, el chatbot debe integrarse con otros sistemas de la UACM, como plataformas de gestión académica y servicios administrativos. Esta integración permitirá que los estudiantes obtengan información en tiempo real sobre sus horarios de clases, trámites o procesos de inscripción.

El sistema también depende de la conectividad a internet, ya que tanto los estudiantes como los administradores necesitan acceso en línea para interactuar con el chatbot y realizar actualizaciones en la base de conocimientos.

7.4 Idiomas de ejemplo

A continuación, se presentan los diagramas de bloques, que ilustran de mejor manera:

Diagramas de bloques

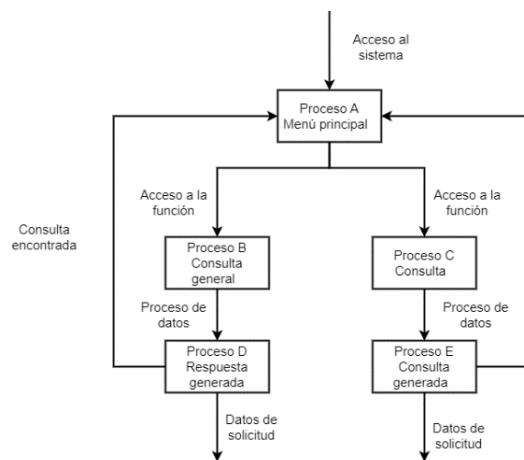


ILUSTRACIÓN 12 DIAGRAMA DE BLOQUES 1

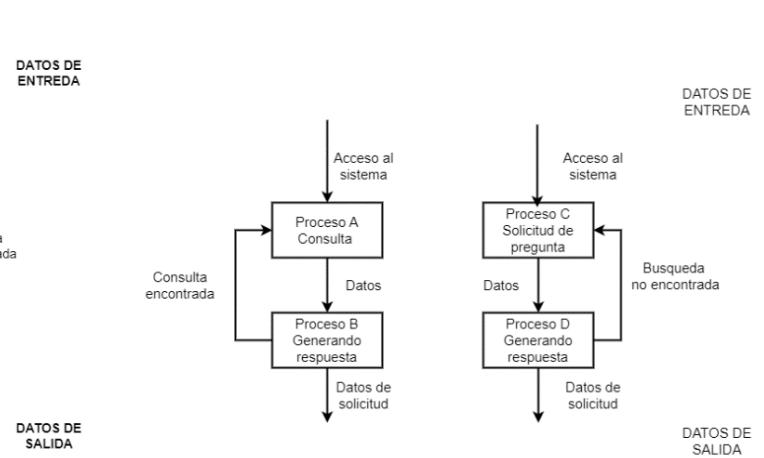


ILUSTRACIÓN 11 DIAGRAMA DE BLOQUES 2

Este diagrama de bloques representa el **flujo de procesamiento de consultas en un sistema**. Se estructura en **datos de entrada, procesos intermedios y datos de salida**, con bifurcaciones según los resultados de la consulta.

Explicación del flujo

1. Entrada al sistema:

- El proceso comienza con el **Proceso A (Menú principal)** cuando un usuario accede al sistema.
- Desde este menú, el usuario puede realizar diferentes consultas.

2. Procesamiento de la consulta:

- Si la consulta es general, se dirige al **Proceso B (Consulta general)**, que procesa los datos y pasa al **Proceso D (Respuesta generada)** si encuentra una coincidencia.
- Si la consulta es más específica, se dirige al **Proceso C (Consulta)**, que también procesa los datos, pero si no encuentra información, genera una nueva consulta a través del **Proceso E (Consulta generada)**.

3. Salida de datos:

- Si se encuentra una respuesta, el sistema genera una salida con los **datos de solicitud** desde el **Proceso D**.
- Si la búsqueda no arroja resultados, se genera una nueva consulta mediante el **Proceso E**, proporcionando otra posible salida de datos.

Puntos clave del diagrama:

- **Menú principal** como punto de acceso al sistema.
- **Bifurcación de procesos** dependiendo del tipo de consulta.
- **Procesos de búsqueda de información** que pueden derivar en una respuesta o en la generación de una nueva consulta.
- **Diferentes salidas de datos**, ya sea con una respuesta encontrada o con una solicitud generada.

Diagrama de componentes

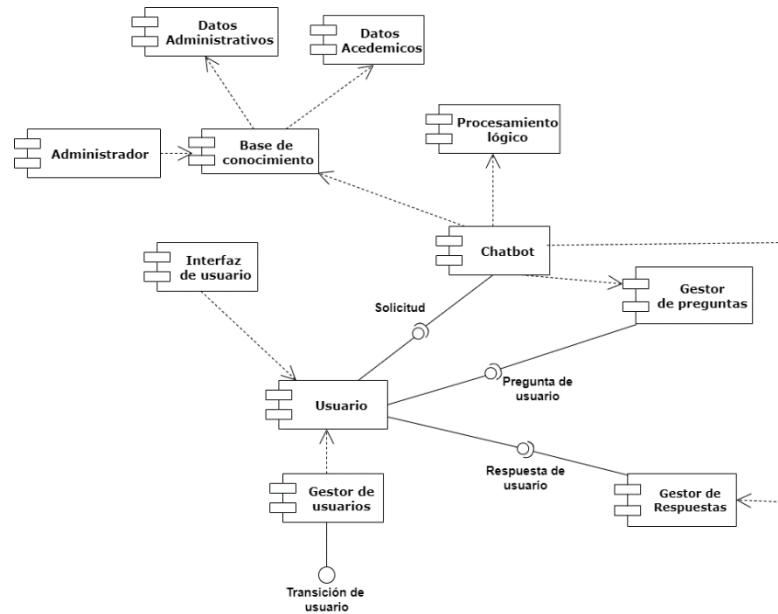


ILUSTRACIÓN 13 DIAGRAMA DE COMPONENTES 2

1. Componentes y dependencias principales

Usuario e Interfaz de Usuario

- El **usuario** interactúa con el sistema a través de la **interfaz de usuario**, que facilita la comunicación con el chatbot.
- Existe una dependencia entre el **usuario** y el **gestor de usuarios**, que maneja la transición y autenticación de los usuarios dentro del sistema.

Chatbot y Procesamiento Lógico

- El **chatbot** es el núcleo del sistema y depende del **procesamiento lógico** para interpretar las solicitudes y responder de manera adecuada.
- Se apoya en una **base de conocimiento**, la cual almacena información administrativa y académica para responder preguntas de los usuarios.

Módulos de Gestión

- **Gestor de Preguntas**: Recibe preguntas del usuario y las procesa, enviándolas al chatbot.

- **Gestor de Respuestas:** Se encarga de devolver respuestas al usuario después de que el chatbot haya procesado la información.
- **Gestor de Usuarios:** Administra las transiciones y sesiones de los usuarios dentro del sistema.

Administrador y Base de Conocimiento

- El **administrador** es responsable de actualizar y mantener la **base de conocimiento**, asegurando que el chatbot tenga información actualizada y precisa.
- La base de conocimiento contiene datos administrativos y académicos que permiten al chatbot responder de manera adecuada.

2. Punto de Vista de la Dependencia en el Software

- **Alta modularidad:** Cada componente del sistema cumple una función específica y se comunica con otros mediante conexiones bien definidas.
- **Acoplamiento controlado:** La mayoría de las dependencias son **unidireccionales**, lo que evita interdependencias innecesarias y facilita la escalabilidad.
- **Separación de responsabilidades:** Cada módulo gestiona un aspecto del chatbot, desde la gestión de usuarios hasta el procesamiento de respuestas.
- **Dependencia en la base de conocimiento:** Tanto el chatbot como el administrador dependen de esta base de datos para garantizar respuestas precisas.

Este diagrama muestra una arquitectura modular donde el **chatbot** es el componente central, apoyado por módulos especializados en preguntas, respuestas y gestión de usuarios. La **base de conocimiento** es un punto crítico de dependencia, asegurando que el chatbot pueda proporcionar información útil.

8. Puntos de vista de la información

Para garantizar el correcto funcionamiento del **Chatbot de la UACM**, es fundamental definir una estructura clara para el manejo y almacenamiento de información.

El chatbot debe procesar datos relacionados con trámites administrativos, inscripciones, normativas y calendarios académicos, asegurando que la información proporcionada sea precisa, actualizada, así como accesible para los estudiantes o el personal universitario.

Dado que el chatbot es un punto de acceso rápido a la información institucional, es esencial que maneje los datos de manera segura, protegiendo su **integridad, confidencialidad y disponibilidad**.

- **Integridad:** La información almacenada y procesada debe mantenerse exacta y libre de alteraciones no autorizadas.
- **Confidencialidad:** El chatbot solo debe proporcionar datos públicos y evitar la exposición de información sensible de los estudiantes.
- **Disponibilidad:** La información debe estar siempre accesible para los usuarios, asegurando que el sistema funcione 24/7 sin interrupciones.

El correcto manejo de estos aspectos garantizará que el chatbot cumpla su función de mejorar la comunicación entre la universidad y los estudiantes sin comprometer la seguridad de los datos.

8.1 Problemas de diseño

El desarrollo de un chatbot presenta varios desafíos en términos de diseño y arquitectura. Algunos de los problemas más comunes incluyen:

1. Falta de adecuación al contexto universitario

- Un chatbot diseñado sin comprender a fondo las necesidades de los estudiantes y la estructura administrativa de la UACM podría generar respuestas inexactas o irrelevantes.

- Para evitar esto, es fundamental realizar un análisis detallado de los requisitos funcionales antes de su implementación.

2. Dificultad para mantener la base de conocimientos actualizada

- Los procesos administrativos y académicos cambian con el tiempo, por lo que el chatbot debe contar con un sistema que permita actualizaciones periódicas.
- Se pueden establecer revisiones programadas donde el personal autorizado edite o agregue información a la base de datos.

3. Exceso o falta de documentación

- Un exceso de documentación puede hacer que el chatbot tenga respuestas demasiado **largas y confusas**.
- Por otro lado, si la documentación es insuficiente, las respuestas del chatbot pueden ser poco claras o incorrectas.
- Es necesario encontrar un equilibrio entre la cantidad y calidad de la información proporcionada.

4. Duplicación de información

- Si existen múltiples fuentes de datos sin una adecuada sincronización, pueden generarse respuestas contradictorias.
- Para evitar esto, el chatbot debe obtener información de una única base de conocimientos centralizada.

5. Relaciones inadecuadas entre módulos

- Un diseño deficiente en la interacción entre los módulos del chatbot puede afectar el procesamiento eficiente de consultas.
- Se debe diseñar una estructura modular clara, donde cada componente tenga una función específica y optimizada.

8.2 Elementos de diseño

El diseño del chatbot debe ser modular y estructurado, permitiendo una implementación escalable tanto como flexible. Los principales elementos que deben considerarse incluyen:

1. Arquitectura del sistema

- Se debe definir una arquitectura basada en microservicios para que cada módulo funcione de manera independiente, mejorando la escalabilidad del chatbot.
- La arquitectura también debe permitir la integración con otras plataformas universitarias.

2. Módulos de interacción

- Se deben desarrollar interfaces intuitivas que permitan a los usuarios interactuar con el chatbot mediante texto o botones de selección.

3. Base de conocimientos

- El sistema debe contar con una base de conocimientos organizada, del mismo modo debe ser estructurada, permitiendo búsquedas rápidas, así como precisas de información relevante.

4. Procesamiento de lenguaje natural (PLN)

- Se utilizarán modelos de inteligencia artificial para interpretar preguntas en lenguaje natural y generar respuestas comprensibles para los estudiantes.

5. Mecanismos de actualización y mantenimiento

- Se debe implementar una funcionalidad que permita actualizar la base de conocimientos de manera sencilla tanto como periódica.

6. Protocolos de seguridad y privacidad

- Se deben establecer medidas para proteger la información del usuario y evitar accesos no autorizados.

8.3 Atributo de datos

Para garantizar que el chatbot cumpla con los requisitos del usuario y ofrezca una experiencia óptima, se deben considerar los siguientes atributos clave:

1. Funcionalidad

- El chatbot debe ser capaz de proporcionar respuestas rápidas, así como precisas a consultas sobre trámites administrativos, inscripciones y normativas.

2. Fiabilidad

- Debe contar con un sistema robusto que minimice errores y asegure la disponibilidad del servicio en todo momento.

3. Usabilidad

- La interfaz debe ser intuitiva y accesible para todos los usuarios, sin necesidad de conocimientos técnicos previos.

4. Eficiencia

- El sistema debe procesar consultas en tiempo real, garantizando tiempos de respuesta rápidos.

5. Mantenimiento

- Se debe facilitar la actualización y mejora continua del chatbot mediante herramientas de administración accesibles.

8.4 Idiomas de ejemplo

Para modelar y documentar el diseño del chatbot, se pueden emplear herramientas de diagramación como:

- **UML (Unified Modeling Language):** Representa la estructura y los flujos del chatbot, facilitando la comprensión de su arquitectura.
- **Diagramas de entidad-relación:** Permiten visualizar la organización de la base de datos y las relaciones entre los diferentes elementos del sistema.

El uso de estos modelos ayuda a optimizar la planificación y la implementación del chatbot, asegurando una integración eficiente con los sistemas de la UACM.

9. Punto de vista del uso de patrones

En esta sección, se presentarán los patrones que se implementarán durante el funcionamiento del sistema, esto con el objetivo de mejorar la calidad eficiencia del chatbot, se aplicarán patrones de diseño que permitan optimizar su arquitectura y de este modo su funcionalidad. Los patrones de diseño proporcionan soluciones comprobadas a problemas recurrentes, facilitando la escalabilidad del mismo modo que el mantenimiento del sistema.

Estos patrones son soluciones probadas, así como efectivas para los problemas más comunes que pueden surgir durante las operaciones cotidianas. Al utilizar soluciones ya conocidas para ciertos problemas específicos, podemos evitar la necesidad de desarrollar soluciones desde cero, lo que nos permite centrarnos en la mejora continua de nuestro software.

Es de este modo, que los patrones que lleguemos a utilizar serán de tipo creacional y diseño que en esta sección nos proporcionará las herramientas necesarias para enfrentar los desafíos que puedan surgir durante el desarrollo, así como las operaciones de nuestro software.

Con este conocimiento, podremos desarrollar un software robusto, eficiente y fácil de mantener. Así que, sin más preámbulos, comenzemos con nuestra exploración de estos patrones y cómo pueden ayudarnos a alcanzar nuestros objetivos.

9.1 Problemas de diseño

Durante el desarrollo y la creación de este sistema se ha visto desde diferentes puntos de vista problemas que han podido retrasar o generar cuestiones que nos impiden resolver de una manera clara las situaciones que se nos presentan, es por ello que se busca la mejor manera de poder enfrentarse a dichas cuestiones en este caso los patrones son siempre una buena opción para realizarlo, pero también se debe elegir aquellos que cumplan con la necesidad del sistema, por lo que a continuación se mencionan algunos ejemplos para una buena elección de estos:

- **Aplicación innecesaria de patrones:** Los patrones son muy útiles a la hora de poder realizar las tareas que se encarguen dentro del sistema, sin embargo, se deben de tomar en cuenta los cuales pueden aplicarse de manera correcta y en otro caso donde se aplican innecesariamente a un diseño, lo que puede hacer que el diseño sea más complejo de lo necesario, esto hará que el software sea más difícil de entender y mantener.

Por ejemplo, dentro de este sistema lo que se buscará será la eficiencia del sistema al dar respuestas, es por ello que se implementarán patrones como el caso de **Singleton, proxy y Factory Method**. Que nos ayudarán precisamente a la organización adecuada de los datos.

- **Selección incorrecta de patrones:** La selección incorrecta de patrones puede hacer que el diseño sea menos eficiente o efectivo de lo que podría ser, por lo que podría llegar a convertirse en una carga en lugar de una ayuda. Por eso siempre es importante tomar en cuenta a donde se aplica dicho caso.
- **Patrones mal implementados:** Si un patrón no se implementa correctamente, puede causar problemas de diseño, es por ello que debemos evitar cualquier nuevo patrón que no se conozca de manera adecuada, además de siempre revisar de manera adecuada la implementación y uso del mismo.

En nuestro sistema a pesar de tener una funcionalidad útil, si el patrón Singleton no es abordado de manera adecuada puede llegar a realizar numerosas fallas al sistema.

9.2 Elementos de diseño

En este apartado se incluirán, del mismo modo que se explicará el uso de los componentes del software que se diseñarán utilizando patrones. Es importante resaltar que estos patrones son aquellos que nos permiten un mejor manejo de los datos, además de ellos nos facilitan encontrar posibles fallas si es que el código tiende a volverse muy complejo.

1. Clase chatbot

- Se definirán el procesamiento del lenguaje acerca de las interacciones que realice el usuario como es el caso de las sesiones, en esta clase de aplicarán los patrones de Methon Factory y Singleton.

2. Gestión de consultas

- Un módulo centralizado procesará las preguntas y generará respuestas basadas en la base de conocimientos, en esta clase de aplicarán los patrones de Methon Factory y Singleton.

3. Módulo de soporte técnico

- Se implementarán mecanismos de comunicación a través de la página web de la UACM con enlaces o teléfonos donde contactar con información elemental en caso de una incidencia.

4. Base de conocimientos fija

- La información será estructurada de acuerdo a lo que necesite por el usuario y será capaz de actualizable periódicamente por el supervisor, para reflejar cambios en los procesos administrativos. en esta clase de aplicarán el patrón Singleton.

5. Seguridad y privacidad

- Se establecerán controles de acceso y encriptación de datos para proteger la información de los usuarios.

Con este diseño detallado, el **Chatbot de Atención Universitaria de la UACM** garantizará un servicio eficiente, seguro y accesible, optimizando la comunicación entre la universidad y la comunidad estudiantil

9.3 Idiomas ejemplo

Aplicación del patrón Singleton

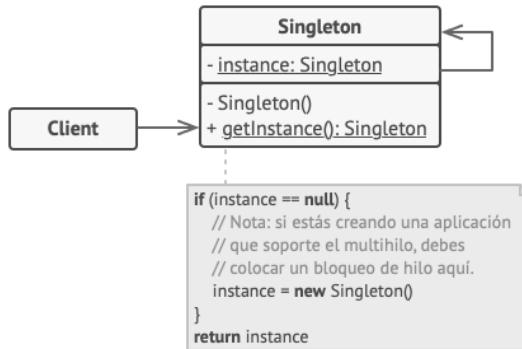


ILUSTRACIÓN 14 PATRÓN SINGLETON

El patrón Singleton es ideal, porque garantiza que solo haya una única instancia de una clase que controle aspectos clave del sistema. Esto es especialmente útil para manejar recursos compartidos o mantener un estado global sin duplicación.

Uso del Singleton

1. Gestión de la base de conocimientos: La clase que maneje la base de conocimientos del chatbot puede implementarse como un Singleton. Esto asegurará que siempre se acceda a la misma instancia, evitando inconsistencias en los datos almacenados y manteniendo la sincronización en las consultas.
2. Gestión de sesiones de usuario: La clase que gestiona las sesiones puede ser Singleton, para un control uniforme de todos los usuarios que interactúan con el sistema. Por ejemplo, si necesitas registrar todas las interacciones, esta instancia será responsable de manejarlo.

Ventajas del uso del Singleton:

- Acceso único: Puedes acceder y modificar datos en tiempo real sin preocuparte por crear múltiples instancias.
- Consistencia: Al utilizar una instancia única, los datos estarán sincronizados y el funcionamiento será más robusto.

Aplicación del patrón Method Factory

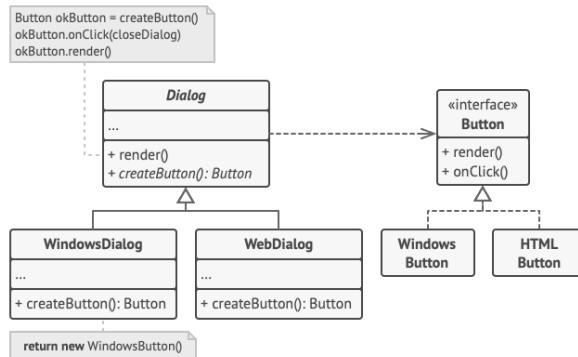


ILUSTRACIÓN 15 PATRÓN METHOD FACTORY

Factory Method puede crear objetos relacionados con las respuestas del chatbot. Por ejemplo, para diferentes tipos de respuestas (texto, enlaces, etc.) y usar subclases específicas para manejar cada tipo.

1. Clase Abstracta (Producto Base): Crea una clase abstracta o una interfaz que define la estructura básica de las respuestas.
2. Subclases Concretas (Productos): Implementa las subclases concretas para diferentes tipos de respuestas.
3. Clase Abstracta de Fábrica: Define una clase abstracta que declare el método para crear objetos.
4. Subclases de Fábrica Concretas: Implementa fábricas concretas para cada tipo de respuesta.

Ventajas de este enfoque:

- Extensibilidad: Es sencillo agregar nuevos tipos de respuestas al sistema sin modificar el código existente.
- Desacoplamiento: El código cliente no necesita conocer las clases concretas, mejorando la mantenibilidad.
- Flexibilidad: Puedes cambiar dinámicamente qué tipo de fábrica utiliza el chatbot, adaptándose a diferentes necesidades.

Aplicación del patrón Proxy

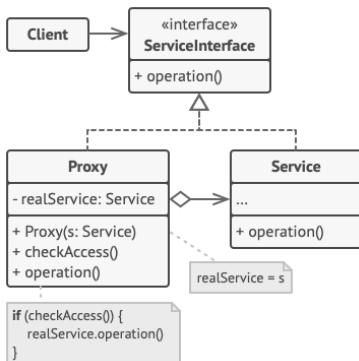


ILUSTRACIÓN 16 PATRÓN PROXY

El Proxy controla el acceso a otro objeto, proporcionando una capa adicional de funcionalidad como seguridad, caché, control de acceso o seguimiento de solicitudes.

Control de acceso a la base de conocimientos: Si la base de conocimientos es una clase robusta con operaciones complejas, el Proxy puede verificar las credenciales del usuario antes de permitir el acceso. Esto asegura que solo los usuarios autorizados puedan realizar consultas.

Optimización con caché: Usa el Proxy para almacenar en caché las respuestas a consultas frecuentes y así evitar el procesamiento redundante. Esto mejora la eficiencia del sistema.

Control de uso: El Proxy puede limitar las consultas por usuario (por ejemplo, un límite de 10 consultas por hora) para evitar el uso abusivo del chatbot.

Ventajas del Proxy en tu chatbot:

- **Seguridad:** Protege los datos sensibles y asegura que solo los usuarios autorizados interactúen con ciertas funcionalidades.
- **Eficiencia:** Reduce la carga de procesamiento mediante el caché y la optimización.
- **Control:** Te permite implementar políticas de acceso y seguimiento de las interacciones de los usuarios.

10. Punto de vista de la interfaz

La interfaz de usuario es uno de los elementos fundamentales en el diseño de cualquier sistema interactivo, en este caso orientado al apoyo académico, tanto administrativo como el que se está desarrollando para la Universidad Autónoma de la Ciudad de México (UACM), cobra aún más relevancia. El propósito de esta interfaz es garantizar una comunicación efectiva entre los usuarios (estudiantes, docentes, administrativos y público en general) así de este modo el sistema, permite acceder a información de manera rápida, clara y segura.

Este chatbot está diseñado bajo el principio de **no almacenar datos personales del usuario**, con el objetivo de preservar la privacidad, reducir vulnerabilidades y mantener un sistema ligero, de fácil mantenimiento. Esto implica que no se requerirá un sistema de registro ni autenticación, y cada sesión se tratará como una interacción anónima. Este enfoque, aunque tiene ventajas en cuanto a privacidad, también plantea ciertos desafíos que se abordan a través de un diseño de interfaz eficaz, orientado a la usabilidad y a la resolución inmediata de consultas.

1. Interfaz de inicio de conversación

La interfaz de inicio será el primer punto de contacto entre el usuario con el sistema, de este modo su propósito principal es generar una experiencia amigable e intuitiva desde el primer momento. En este espacio se mostrará un saludo inicial por parte del chatbot, seguido de un conjunto de botones o accesos rápidos a preguntas comunes como: “Requisitos de inscripción”, “Servicios escolares”, “Trámites frecuentes”, “Contacto con la universidad”, entre otros.

La elección de no utilizar formularios de inicio de sesión implica que el sistema debe ser extremadamente claro en su propósito desde el principio. El diseño de esta interfaz será visualmente limpio, con iconografía sencilla y textos explicativos que orienten al usuario para formular preguntas. Al no haber registro, toda la lógica de la interfaz está centrada en la **consulta directa**, sin pasos intermedios, lo que garantiza agilidad y una experiencia fluida incluso para personas con pocos conocimientos tecnológicos.

2. Interfaz de procesamiento y entrega de respuestas

Una vez que el usuario introduce una consulta, el chatbot debe procesar esa entrada, interpretarla y entregar una respuesta adecuada. Aunque este procesamiento se da en el **backend** mediante un motor de lenguaje natural, la interfaz debe representar este flujo de manera clara y transparente para el usuario.

Visualmente, esta parte del sistema se estructura como una conversación. Los mensajes del usuario aparecen alineados a la derecha, mientras que las respuestas del bot aparecen a la izquierda, con colores diferenciados y posibles elementos gráficos complementarios como botones, listas, enlaces o íconos informativos.

Dado que no existe un historial de usuario, cada consulta es independiente. Esto significa que, si la respuesta requiere más contexto, el sistema debe guiar al usuario de forma interactiva, haciendo preguntas aclaratorias o sugiriendo opciones. La clave de esta interfaz es mantener un equilibrio entre la automatización, así como la claridad, evitando respuestas genéricas, además de ofrecer siempre alternativas útiles.

3. Interfaz de navegación temática estructurada

Además del modelo conversacional libre, el sistema contará con una interfaz de navegación por categorías que ayudará a los usuarios que prefieren no escribir, sino explorar visualmente. Esta navegación se compone de un menú jerárquico donde los usuarios pueden elegir temas específicos, como:

- Fechas importantes del calendario académico.
- Procedimientos para tramitar constancias.
- Becas y apoyos estudiantiles.
- Información de contacto institucional.

Este modelo es especialmente útil para reducir errores de interpretación del lenguaje natural y también para brindar una experiencia más estructurada a usuarios que solo buscan información puntual. Además, facilita la inclusión de elementos visuales complementarios como íconos, desplegables o imágenes que ayuden a una navegación más efectiva.

4. Interfaz de escalamiento a atención humana

Uno de los valores más importantes de un sistema automatizado como este es saber reconocer sus límites. Por ello, se integrará una interfaz para derivar al usuario a un canal de atención humana cuando el chatbot no pueda proporcionar una solución adecuada. Esta interfaz mostrará claramente las opciones disponibles para escalar la consulta: correos electrónicos institucionales, teléfonos de contacto, horarios de atención e incluso enlaces a formularios específicos del sitio web de la universidad. Esto garantiza que, incluso si el chatbot no puede resolver la necesidad, el usuario siempre tendrá una alternativa de contacto válida, lo que refuerza la confianza en el sistema y la satisfacción del usuario.

5. Interfaz de mantenimiento y actualización del sistema (solo para administradores)

Aunque no será visible para los usuarios finales, se contará con una interfaz exclusiva para el personal técnico encargado del mantenimiento del sistema. Esta interfaz permitirá actualizar la base de conocimientos, analizar tendencias de uso, agregar nuevas preguntas frecuentes o ajustar respuestas mal formuladas.

Debido a la naturaleza del sistema sin almacenamiento de datos, no se generarán reportes personalizados por usuario, pero sí se podrá acceder a estadísticas globales como: consultas más comunes, categorías con más actividad, y niveles de éxito en la resolución automática de dudas. Este panel será una herramienta vital para asegurar la mejora continua del sistema.

10.1 Problemas de diseño

Diseñar un sistema de chatbot sin almacenamiento de datos plantea retos importantes. Uno de los principales es la **falta de contexto persistente**, lo que obliga a diseñar una experiencia que no dependa de información previa del usuario. Esto se traduce en una necesidad de precisión en las respuestas, y de estrategias para guiar la conversación en caso de ambigüedad.

Otro problema potencial es el **riesgo de sobrecarga cognitiva**, ya que los usuarios pueden no estar familiarizados con cómo interactuar con un bot. Para mitigar esto, el diseño debe ser auto explicativo, con ejemplos visibles, preguntas sugeridas y ayudas visuales que orienten en todo momento. Además, se deben evitar respuestas excesivamente técnicas o genéricas, priorizando siempre un lenguaje claro, así como accesible.

Finalmente, se debe tener cuidado en no generar una **falsa percepción de personalización**. Ya que no hay almacenamiento de historial, el bot no puede recordar preferencias o preguntas anteriores, por lo tanto, el diseño debe ser honesto y transparente, dejando claro que se trata de un sistema de consulta anónima.

10.2 Elementos de diseño

El diseño del sistema está compuesto por múltiples elementos gráficos y funcionales que trabajan en conjunto para proporcionar una experiencia satisfactoria:

- **Diseño modular y responsivo**, que se adapta a dispositivos móviles, tabletas y computadoras de escritorio.
- **Sistema de preguntas sugeridas**, que aparecerá antes y después de cada consulta, para guiar a los usuarios y facilitar la navegación.
- **Componentes visuales accesibles**, como botones grandes, colores contrastantes y textos legibles.
- **Animaciones mínimas** que den retroalimentación al usuario, como indicadores de que el bot está "escribiendo" o procesando una respuesta.
- **Mensajes de error amigables** que no generen frustración, y siempre ofrezcan una alternativa al usuario.

10.3 Atributos de la interfaz

Los atributos principales de las interfaces del chatbot se centran en:

- **Interactividad sin fricción:** el usuario puede interactuar con el sistema sin ningún tipo de registro, contraseña o procedimiento previo.
- **Alta disponibilidad y rendimiento:** el sistema debe funcionar sin interrupciones, garantizando tiempos de respuesta cortos incluso en momentos de alta demanda.
- **Seguridad sin comprometer la experiencia:** aunque no se almacenan datos, el sistema será auditado para evitar vulnerabilidades técnicas o exposiciones accidentales.
- **Escalabilidad visual y funcional:** el diseño debe permitir que nuevas funcionalidades se integren de forma sencilla, como nuevos módulos temáticos o respuestas multimedia (imágenes, PDFs, etc.).

10.4 Idiomas de ejemplo

Diagrama de componentes

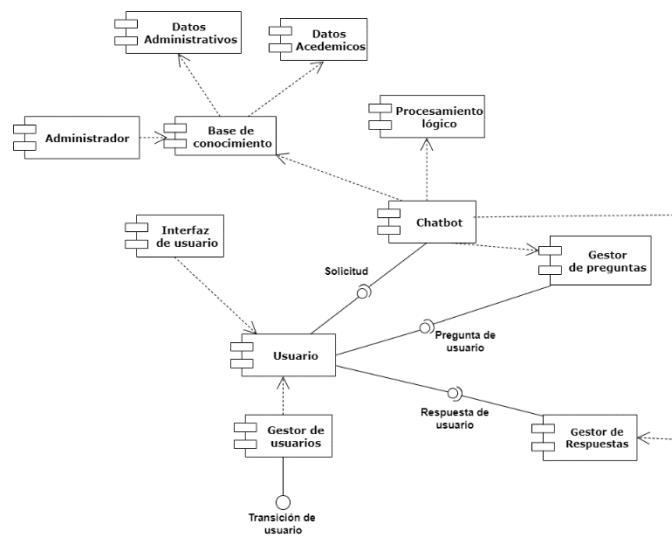


ILUSTRACIÓN 17 DIAGRAMA DE COMPONENTES 3

El diagrama muestra la **arquitectura de componentes del sistema**, con énfasis en cómo se relacionan entre sí los módulos técnicos y funcionales, especialmente los que interactúan directa o indirectamente con el usuario.

Interfaz de usuario (Usuario - Interfaz de usuario)

- **Ubicación en el diagrama:** A la izquierda, conectada al componente “Usuario”.
- **Función:** Esta es la **entrada principal** del sistema. Es donde los usuarios (estudiantes, docentes, administrativos, público) **interactúan con el chatbot**.
- **Relación con el texto:** Corresponde a los apartados:
 - Interfaz de inicio de conversación
 - Interfaz de procesamiento y entrega de respuestas
 - Interfaz de navegación temática estructurada
 - Interfaz de escalamiento a atención humana

Es una interfaz **sin registro ni autenticación**, diseñada para ser clara, rápida y amigable.

Usuario y Chatbot

- **El “Usuario”** formula una pregunta a través de la interfaz.
- La “Solicitud” se canaliza al “Chatbot”.
- El “Chatbot” actúa como **motor central del sistema**, responsable de dirigir, procesar y obtener respuestas.

Gestor de preguntas y respuestas

- **Gestor de preguntas:** Recibe y analiza las preguntas que el usuario hace. Está directamente conectado con el “Chatbot”.
- **Gestor de respuestas:** Devuelve la respuesta procesada desde el “Chatbot” hacia el usuario.
- **Flujo:**

- Pregunta del usuario → Gestor de preguntas
- Chatbot procesa → Gestor de respuestas
- Respuesta al usuario

Interfaz de procesamiento se refleja en esta lógica, donde cada pregunta es tratada como una nueva sesión independiente.

Base de conocimiento

- Es la **fuente de datos** que el chatbot consulta para responder.
- Contiene tanto:
 - **Datos Académicos**
 - **Datos Administrativos**
- El **Administrador** tiene acceso a esta base mediante la interfaz de mantenimiento y actualización (punto 5 del texto).

Procesamiento lógico

- Aquí se encuentran los algoritmos o módulos de inteligencia artificial o lenguaje natural que **interpretan la intención del usuario** y deciden la mejor respuesta.
- Está conectado a:
 - El “Chatbot”
 - La “Base de conocimiento”
- **Ejemplo:** cuando un usuario escribe "¿Cuándo inician las clases?", este módulo interpreta la intención y busca una fecha relevante.

Gestor de usuarios

- Aunque el sistema **no almacena datos personales**, este componente puede representar:
 - La transición de usuario entre estados o tipos de consulta.

- La gestión anónima de sesiones o interacciones temporales.

Interfaz de administrador (implícita)

- No está claramente visualizada como interfaz gráfica, pero se representa mediante:
 - El “Administrador” que actualiza la **Base de conocimiento**.
 - Esto se conecta con el apartado 5 del texto, la *Interfaz de mantenimiento y actualización del sistema*.

Este diagrama **refuerza el diseño modular, funcional y sin almacenamiento de datos personales**, tal como lo detalla el documento. Cada componente cumple un rol específico que **preserva la privacidad**, mientras garantiza un flujo de información eficiente, claro y escalable.

Interfaz principal del sistema

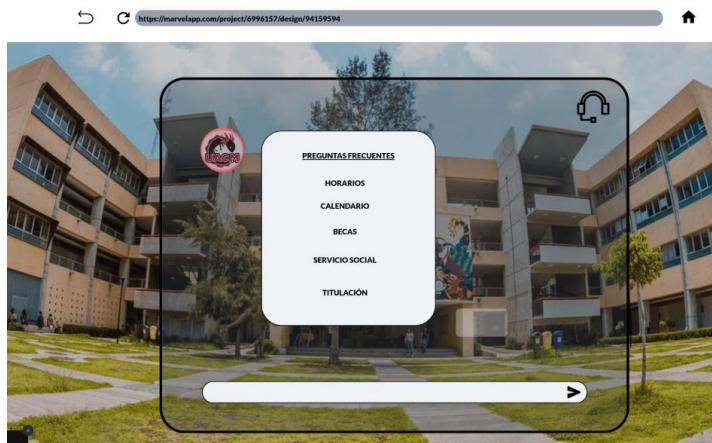


ILUSTRACIÓN 18 INTERFAZ PRINCIPAL

En la imagen anterior se puede observar la página principal del sistema, específicamente el panel inicial del chatbot desarrollado. Esta interfaz representa el punto de entrada para los usuarios que interactúan con el sistema, **ya sean estudiantes, personal administrativo o visitantes externos**.

Desde este espacio, se accede directamente al módulo de preguntas frecuentes, diseñado con el objetivo de proporcionar respuestas rápidas, claras y útiles a las consultas más comunes realizadas por los usuarios.

El diseño de este panel se centra en la simplicidad, así como la usabilidad, permitiendo que cualquier persona, independientemente de su nivel técnico, pueda utilizar el sistema de forma intuitiva. Gracias a la implementación de una interfaz interactiva, los usuarios pueden escribir sus preguntas y recibir respuestas automáticas en tiempo real, basadas en la base de conocimientos previamente integrada. Esto permite *ahorrar* tiempo y reducir la carga operativa del personal, al atender consultas repetitivas sin intervención humana. Además, esta página inicial sirve como punto de orientación dentro del sistema, ya que desde allí también pueden derivarse otras funciones complementarias en futuras versiones, como accesos rápidos a diferentes departamentos, enlaces útiles o formularios de contacto.

Interfaz secundaria del sistema

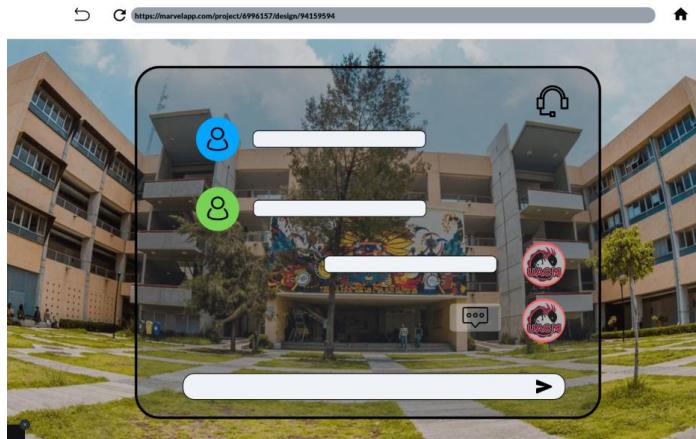


ILUSTRACIÓN 19 INTERFAZ SECUNDARIA

En la siguiente imagen se ilustra el funcionamiento dinámico del sistema, específicamente el comportamiento del chatbot cuando el usuario no encuentra la información deseada en la sección de preguntas frecuentes.

En estos casos, el estudiante, personal administrativo o cualquier otro usuario, tiene la libertad de formular preguntas personalizadas directamente al chatbot a través del campo de texto habilitado para ello. Esta funcionalidad está pensada para atender necesidades específicas que no han sido contempladas en las respuestas predeterminadas.

El chatbot, mediante técnicas de procesamiento de lenguaje natural, analiza la consulta generará una respuesta lo más precisa y contextual posible, utilizando la

información contenida en su base de conocimientos. Este enfoque permite que el sistema no se limite a un catálogo estático de respuestas, sino que sea capaz de adaptarse a las distintas formas en las que los usuarios formulan sus dudas. En conjunto, esta funcionalidad garantiza que el chatbot no solo sea un repositorio de respuestas comunes, sino una herramienta activa de consulta capaz de comprender preguntas variadas y ofrecer soluciones en tiempo real, manteniendo siempre el foco en la utilidad práctica para la comunidad universitaria.

Interfaz de soporte del sistema

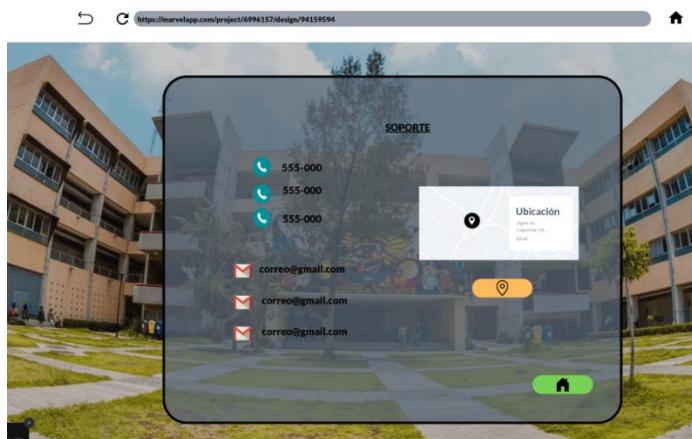


ILUSTRACIÓN 20 INTERFAZ DE SOPORTE

Finalmente, se aprecia el módulo de soporte del sistema, una sección diseñada específicamente para proporcionar a los usuarios información de contacto institucional relevante, en caso de que requieran atención presencial o asistencia adicional por otros medios. Este módulo cumple un papel fundamental dentro del chatbot, ya que reconoce que, **aunque muchas dudas pueden resolverse de manera automática mediante respuestas frecuentes o consultas directas, existen situaciones en las que el usuario necesita comunicarse directamente con personal de la institución.**

Por esta razón, en este apartado se pone a disposición del usuario información clave como números telefónicos, direcciones de correo electrónico y ubicaciones físicas de las oficinas o departamentos correspondientes.

11. Punto de vista de estructura

La estructura de un sistema de software define cómo se organiza, cómo se comunican sus componentes así como estos colaboran para cumplir sus funciones. En el desarrollo de este chatbot universitario, la estructura se ha diseñado con una arquitectura modular, escalable y mantenible, que permita su evolución sin comprometer la simplicidad ni la privacidad.

El sistema hace uso de tecnologías robustas tanto actuales. El núcleo del chatbot será desarrollado a través del uso de tecnología GPT, debido a su sintaxis clara, amplia comunidad de soporte y su integración con potentes bibliotecas de procesamiento de lenguaje natural. Para la parte web, se empleará **Django**, un framework de alto nivel que permite construir aplicaciones web rápidas, seguras y escalables. Complementariamente, se incorporará **JavaScript** para mejorar la interactividad, dinamismo y respuesta en tiempo real de las interfaces, lo cual resulta crucial para mantener una experiencia conversacional fluida.

Dado que el sistema **no almacenará información sensible ni historial de usuario**, su estructura está diseñada para procesar cada interacción como una sesión única, minimizando dependencias y reduciendo riesgos de privacidad. Esto influye directamente en la organización de sus componentes así en la forma en que se diseñan los elementos visuales tanto como funcionales del sistema.

Elementos estructurales clave del sistema

El chatbot estará compuesto por varios componentes funcionales que, aunque independientes entre sí, cooperan de manera integrada:

- **Botones de entrada y salida:** Estos elementos permitirán a los usuarios comenzar o terminar una conversación, así como navegar hacia secciones principales como ayuda, preguntas frecuentes o contacto. Dado que el sistema no requiere inicio de sesión, estos botones ofrecen una forma rápida de moverse dentro del flujo conversacional sin necesidad de autenticación.
- **Botones de acción:** Estos se utilizarán para ejecutar tareas específicas dentro del chatbot, como abrir menús de opciones, activar respuestas rápidas, o acceder a un enlace institucional. Al no requerir formularios complejos, estos botones contribuyen a una experiencia minimalista y eficiente.

- **Colores significativos:** Se aplicará una paleta cromática institucional, asignando colores distintos a los elementos clave (respuestas del chatbot, mensajes del usuario, botones activos, errores o advertencias). Esta codificación visual mejora la accesibilidad y facilita la orientación, especialmente en sesiones cortas donde el usuario debe identificar visualmente cada respuesta con rapidez.
- **Botones de desplazamiento:** Se emplearán para navegar fácilmente por secciones extensas de texto o menús. Por ejemplo, si se proporciona una lista larga de trámites o requisitos, estos botones permitirán al usuario deslizarse sin esfuerzo por el contenido, tanto en dispositivos móviles como en escritorio.
- **Interfaces interactivas:** Las conversaciones estarán diseñadas para ser dinámicas, permitiendo al usuario elegir respuestas mediante clics en lugar de escribir en todo momento. Esto mejora la accesibilidad y reduce errores en la comprensión del lenguaje natural.
- **Campos de texto:** Aunque no se requiere introducir datos personales, se ofrecerán campos de texto para que los usuarios formulen consultas libremente. Estos campos estarán optimizados con sugerencias automáticas y validación básica para mejorar la comprensión de las preguntas.
- **Información sobre los apartados:** Cada sección temática del chatbot estará acompañada de descripciones breves y enlaces informativos. Esto facilitará al usuario entender el propósito de cada módulo sin depender de instrucciones externas.
- **Herramientas de ayuda:** El chatbot incluirá un menú de asistencia donde el usuario podrá ver ejemplos de preguntas comunes, acceder a documentación de apoyo o contactar con personal universitario en caso de dudas más complejas. Estas herramientas estarán accesibles desde cualquier parte del sistema.

11.1 Problemas de diseño

Uno de los desafíos más importantes en la estructuración del sistema radica en su **modularización**, es decir, en la división del chatbot en componentes reutilizables que puedan mantenerse de manera independiente. Esta división incluye la lógica conversacional, las reglas de respuesta, los enlaces a contenido institucional y los elementos visuales.

El sistema, como componente de grano grueso, se divide internamente en módulos de grano fino, como el motor de diálogo, los templates de respuestas, la capa de presentación y el módulo de integración con recursos externos (sitios web, PDFs, calendarios). Esta segmentación no solo mejora la mantenibilidad del sistema, sino que permite que cada parte se pruebe de forma aislada, promoviendo así un desarrollo más controlado y menos propenso a errores.

Además, al no manejar sesiones de usuario persistentes, se evita la complejidad de almacenar o recuperar información personalizada, pero esto obliga a que el diseño sea suficientemente **autosuficiente** en cada interacción. Esto se soluciona con la implementación de patrones de diseño conversacional que simulan continuidad a través de opciones guiadas, sin necesidad de guardar estado.

Otro problema estructural es el equilibrio entre la **flexibilidad y la simplicidad**. Un chatbot demasiado abierto puede generar respuestas erróneas o inadecuadas, mientras que uno muy rígido podría resultar poco útil. Por ello, se diseñará un flujo conversacional híbrido que permita libertad en las consultas, pero con opciones de asistencia y navegación visibles en todo momento.

11.2 Elementos de diseño

Desde el punto de vista de ingeniería de software, la estructura del chatbot también debe cumplir con principios formales de diseño. A continuación, se describen los principales elementos estructurales:

Entidades de diseño:

- **Puerto:** Representa los puntos de entrada y salida del sistema, por ejemplo, los botones que disparan acciones o inician interacciones.

- **Conector:** Es el canal de comunicación entre entidades. En este sistema, puede representarse como el middleware que conecta la entrada del usuario con el motor de procesamiento y las respuestas.
- **Interfaz:** Es el componente visual con el que interactúa el usuario, compuesta por los elementos ya mencionados (botones, menús, texto, etc.).
- **Parte:** Cada uno de los módulos funcionales, como el generador de respuestas, la ayuda contextual, o el motor de navegación temática.
- **Clase:** A nivel de código, son las unidades de lógica que encapsulan funcionalidades específicas como análisis semántico, formateo de respuestas o manejo de enlaces.

Atributos de diseño:

Cada una de estas entidades posee atributos que definen su comportamiento:

- **Nombre:** Identificador del componente (ej. RespuestaRapida, BotonAyuda).
- **Tipo:** Clasificación funcional del componente (ej. visual, lógica, estructural).
- **Propósito:** Qué función cumple dentro del sistema.
- **Definición:** Especificación técnica detallada que describe su funcionamiento.

Restricciones de diseño:

- **Restricciones de interfaz:** La interfaz debe ser minimalista, accesible y sin necesidad de registro.
- **Restricciones de reutilización:** Los componentes deben ser independientes entre sí para poder ser reutilizados en futuras versiones o incluso en otros sistemas.
- **Restricciones de dependencia:** Se debe evitar el acoplamiento entre módulos, garantizando así que cada componente funcione por separado sin afectar a los demás.

11.3 Idiomas de ejemplo

Diagrama de clases

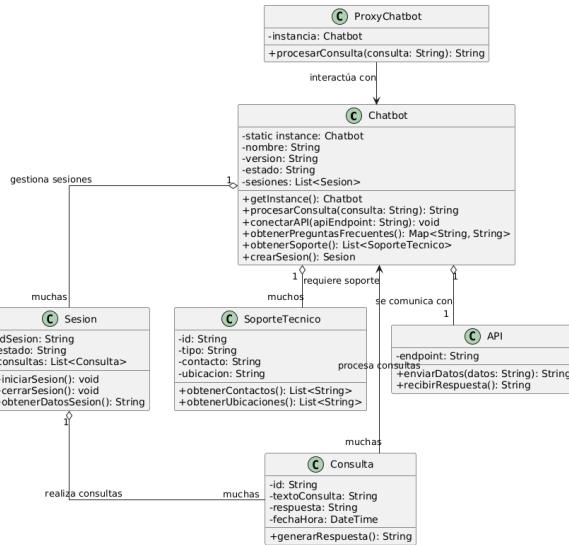


ILUSTRACIÓN 21 DIAGRAMA DE CLASES 2

El diagrama de clases representa la estructura interna del chatbot universitario desde una perspectiva de diseño orientado a objetos. Este diagrama detalla cómo se organizan los componentes lógicos del sistema, cómo se comunican además de qué responsabilidades tiene cada clase. Esta vista es clave para entender y aplicar los principios de modularidad, mantenibilidad, y escalabilidad mencionados en el apartado 11.

Correspondencia entre el punto de vista estructural y el diagrama

Modularidad del sistema

- La estructura del sistema está dividida en módulos funcionales independientes:
 - Chatbot: núcleo central que coordina el funcionamiento.
 - ProxyChatbot: patrón de diseño proxy que encapsula el acceso al chatbot.

- Sesion: módulo que gestiona la interacción del usuario durante cada sesión.
- Consulta: módulo que representa cada pregunta realizada.
- API: módulo que simula la comunicación con fuentes externas de datos o servicios.
- SoporteTecnico: componente especializado que devuelve información adicional de contacto institucional.

Este diseño cumple con el principio de bajo acoplamiento y alta cohesión, ya que cada clase tiene una responsabilidad clara.

Conectores y puertos

- Los métodos públicos como procesarConsulta, crearSesion, obtenerPreguntasFrecuentes o enviarDatos funcionan como conectores entre clases, permitiendo el flujo de datos entre ellas.
- El ProxyChatbot actúa como un puerto de entrada, al recibir las solicitudes del usuario y canalizarlas hacia el núcleo Chatbot.
- La clase API representa un conector externo que permite que el chatbot se comunique con otras plataformas o bases de datos, siguiendo el patrón de integración de servicios descrito en la estructura.

Interfaz visual y experiencia de usuario

Aunque el diagrama se centra en la lógica del backend, se puede mapear su funcionamiento hacia la interfaz descrita:

- Cada sesión (Sesion) representa una interacción única del usuario (como se explicó en el punto sobre privacidad, donde no se guardan datos sensibles).
- Las consultas (Consulta) permiten al sistema procesar preguntas con o sin campos de texto libres (como los mencionados en el diseño de campos optimizados).
- Métodos como generarRespuesta () se vinculan directamente con la lógica conversacional que alimenta las respuestas dinámicas del chatbot.

Independencia y reutilización

- Las clases pueden desarrollarse y mantenerse de forma independiente, como se exige en las restricciones de diseño:
 - Consulta no depende de API directamente.
 - SoporteTecnico puede reutilizarse en otros contextos (por ejemplo, en otros módulos de contacto).
 - Sesion encapsula el historial de una interacción sin almacenarla de forma persistente, cumpliendo con la política de privacidad.

Escalabilidad y mantenibilidad

- El patrón Singleton implementado con getInstance() en Chatbot garantiza que el sistema tenga una única instancia del núcleo, lo que simplifica el control global del comportamiento del bot.
- La relación de composición entre Chatbot y Sesion, y entre Sesion y Consulta, refleja una estructura jerárquica clara y fácil de escalar.
- Se pueden agregar más tipos de consultas, APIs externas o módulos de soporte sin alterar las clases base.

Este diagrama de clases representa fielmente la estructura modular, desacoplada y funcionalmente segmentada que se describe en el punto de vista estructural. Cada componente cumple una función específica dentro del sistema, lo que garantiza flexibilidad, mantenimiento sencillo y adaptación a cambios futuros. Además, se alinea con principios de diseño limpio, lo que facilita su comprensión y expansión por parte de otros desarrolladores o equipos técnicos.

12. Punto de vista de interacción

La interacción con este sistema de chatbot tiene como principal objetivo facilitar el acceso a información institucional relevante para estudiantes, docentes y personal universitario, de una forma sencilla, inmediata y sin requerir conocimientos técnicos. Su implementación responde a la necesidad de mejorar la eficiencia en la atención digital, permitiendo a los usuarios obtener respuestas rápidas a sus dudas frecuentes, acceder a trámites o consultar recursos disponibles en línea sin tener que navegar por múltiples páginas o esperar atención personalizada.

Esta solución también tiene un enfoque claro en la **comodidad y autonomía** del usuario. Gracias a su interfaz conversacional, el sistema permite a los usuarios explorar contenido institucional en lenguaje natural, con una experiencia más amigable que los menús tradicionales. Dado que el sistema no almacena datos personales ni históricos de conversación, garantiza una experiencia segura y respetuosa con la privacidad, ideal para contextos educativos.

La interacción ocurre directamente dentro de una plataforma web universitaria, accesible desde navegadores comunes tanto en computadoras de escritorio como en dispositivos móviles. Esto permite una alta disponibilidad del servicio, eliminando barreras de acceso por sistema operativo, ubicación o tipo de dispositivo.

La interacción entre el usuario y el sistema ocurre mediante una conversación simulada, en la que el usuario formula preguntas o selecciona opciones guiadas, y el chatbot responde en tiempo real con información clara y útil. El motor del chatbot analiza el texto ingresado, identifica intenciones comunes (como pedir requisitos, fechas importantes, horarios, enlaces a trámites), y responde de forma estructurada o contextual.

El sistema ofrece múltiples niveles de interacción según el tipo de consulta:

- Para consultas simples, el usuario puede escribir libremente y recibir respuestas automatizadas.
- En casos más complejos, el chatbot presenta botones interactivos para guiar al usuario por un flujo de conversación estructurado, facilitando la navegación.

- Además, cuenta con accesos rápidos a recursos institucionales, como reglamentos, calendarios académicos o tutoriales.

Todo esto se realiza en sesiones independientes y temporales, sin guardar información personal, garantizando privacidad y simplicidad en el diseño.

La interacción ocurre en distintos niveles, desde el más superficial, como el contacto visual con la interfaz, hasta el más técnico, como la interacción con el motor de procesamiento conversacional. Estos niveles incluyen:

- **Interacción visual y de entrada:** Representada por la interfaz gráfica del chatbot, donde el usuario observa, escribe y hace clic. Los colores, tipografías, disposición de los elementos están diseñados para reducir la carga cognitiva y facilitar la atención.
- **Interacción semántica:** En este nivel, el usuario formula preguntas en lenguaje natural, el sistema interpreta la intención y contexto para generar una respuesta coherente. Aquí interviene el motor de procesamiento de lenguaje (NLP) y las reglas de conversación predefinidas.
- **Interacción funcional:** Se refiere a la conexión entre las preguntas del usuario y las acciones que ejecuta el sistema, como mostrar enlaces, abrir secciones, ofrecer documentos o brindar rutas de contacto.
- **Interacción con servicios externos (opcional):** En caso de integrar recursos externos como sistemas de calendario, ayuda académica o mapas, el chatbot puede actuar como intermediario, redirigiendo al usuario de manera contextual sin manejar datos sensibles.

12.1 Problemas de diseño

Uno de los principales desafíos de diseño es **definir claramente la responsabilidad de cada componente del sistema**, especialmente en un entorno conversacional donde múltiples intenciones pueden superponerse. Es crucial determinar qué partes del chatbot deben encargarse del procesamiento de lenguaje, cuáles gestionan la presentación de respuestas, y cómo se organizan los flujos de conversación para evitar ambigüedad o repetición.

Otro aspecto crítico es el manejo de **transiciones de estado** dentro de la conversación. Por ejemplo, cuando un usuario pregunta por un trámite y luego cambia de tema abruptamente, el chatbot debe poder cerrar el flujo anterior y abrir uno nuevo sin confundir al usuario. Esto requiere una lógica bien definida para manejar múltiples estados conversacionales y evitar respuestas fuera de contexto.

También debe considerarse la **conurrencia de interacciones**, es decir, la capacidad del sistema para atender múltiples usuarios simultáneamente sin degradar el rendimiento ni provocar errores. Esto implica diseñar sesiones ligeras y autónomas, capaces de manejar interacciones paralelas sin necesidad de mantener datos de sesión.

Finalmente, un buen diseño debe contemplar **mecanismos de recuperación**: cuando el sistema no entienda una pregunta, debe ofrecer alternativas, ejemplos o redireccionamientos, en lugar de responder con un simple “no entiendo”, mejorando así la percepción de inteligencia del sistema.

12.2 Elementos de diseño

En esta sección se identifican los elementos técnicos clave utilizados para modelar el comportamiento interactivo del chatbot, tanto a nivel lógico como visual.

Clases:

- **Pregunta Usuario:** Representa una entrada de texto del usuario. Contiene atributos como contenido, intención estimada y fecha/hora de emisión.
- **RespuestaSistema:** Contiene el texto de salida generado por el chatbot. También puede incluir botones, enlaces o elementos multimedia.
- **FlujoConversacional:** Clase que representa un conjunto estructurado de turnos entre usuario y chatbot, como una conversación sobre inscripción o calendarios.

Métodos:

- **interpretarPregunta ():** Analiza el texto del usuario para detectar la intención, usando reglas o modelos NLP.

- **generarRespuesta ()**: Construye una respuesta textual o visual basada en la intención detectada.
- **mostrarOpciones ()**: Presenta botones u opciones contextuales para continuar el flujo conversacional.

Estados:

- **Inicio**: El usuario inicia una nueva interacción.
- **En Flujo**: El chatbot está siguiendo una conversación estructurada.
- **Interrumpido**: El usuario cambia abruptamente de tema o la conexión se pierde.
- **Finalizado**: La conversación concluye de forma natural o por inactividad.

Eventos:

- **Consulta realizada**: El usuario envía una nueva pregunta.
- **Botón presionado**: El usuario selecciona una opción del chatbot.
- **Error de entrada**: El sistema no logra interpretar la intención.
- **Sugerencia aceptada**: El usuario elige una de las recomendaciones ofrecidas.

Jerarquía:

- **Usuario general**: Tiene acceso a toda la funcionalidad sin necesidad de autenticarse.
- **Administrador del sistema (opcional)**: Podría tener acceso a configurar nuevas preguntas o respuestas desde una consola privada, si se desea incluir mantenimiento.

Temporización:

El sistema puede mantener la sesión abierta durante algunos minutos de inactividad (por ejemplo, 5 minutos), tras lo cual se cerrará automáticamente para mantener la eficiencia. Esto evita consumir recursos innecesarios y garantiza que no se retenga ninguna información del usuario tras la interacción.

Sincronización:

Dado que múltiples usuarios pueden estar activos al mismo tiempo, cada sesión se maneja como un proceso aislado. Los cambios en la lógica del chatbot (como nuevos flujos o respuestas) se pueden sincronizar de forma centralizada en el servidor, sin afectar las sesiones activas en curso.

12.3 Idiomas de ejemplo

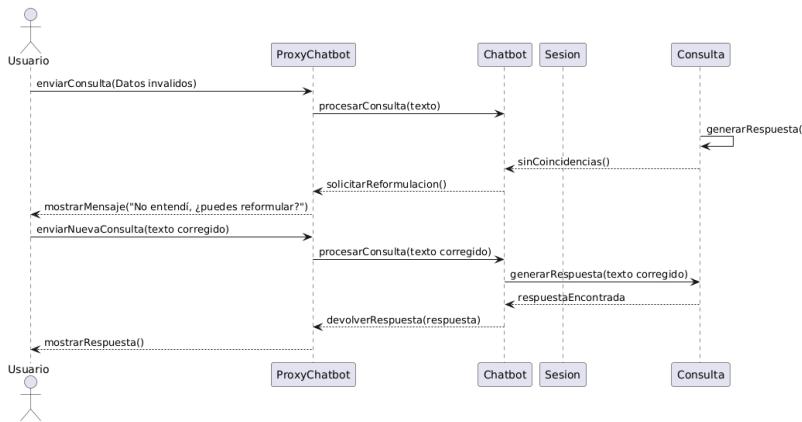


ILUSTRACIÓN 22 DIAGRAMA SE SECUENCIA 1

Manejo de una consulta inválida

Este diagrama muestra cómo el chatbot maneja una consulta inválida. Su propósito es garantizar una **experiencia fluida** para el usuario, evitando respuestas imprecisas o vacías.

- **Interacción visual:** El usuario recibe un mensaje solicitando que reformule su pregunta en caso de error.
- **Interacción semántica:** El sistema identifica que no hay coincidencias y solicita aclaraciones.
- **Interacción funcional:** Se ejecuta una reformulación de la pregunta y el chatbot genera una nueva respuesta.

Desde la perspectiva de diseño

Este diagrama resuelve uno de los desafíos críticos del diseño conversacional: la gestión de **consultas ambiguas**. En lugar de responder con un simple "No entiendo", el sistema ofrece ayuda para que el usuario ajuste su pregunta. Esto mejora la percepción de inteligencia y accesibilidad del chatbot.

Desde los elementos técnicos

- **Clases clave:** Pregunta Usuario, RespuestaSistema, FlujoConversacional.
- **Métodos relevantes:** interpretarPregunta (), generarRespuesta (), sin Coincidencias () .
- **Estados:** Inicio, En Flujo, Interrumpido, Finalizado.

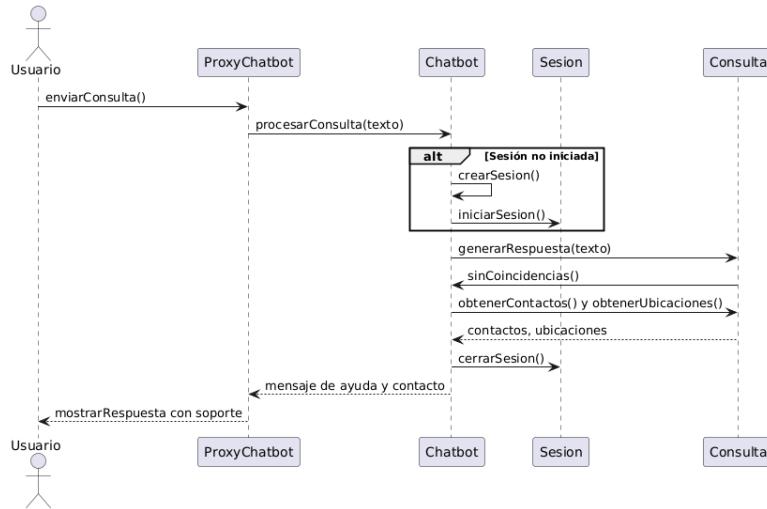


ILUSTRACIÓN 23 DIAGRAMA DE SECUENCIA 2

Manejo de sesiones y soporte

Aquí observamos cómo el chatbot administra sesiones independientes para cada usuario. Esto permite mantener la **privacidad y la simplicidad**, ya que no se almacenan datos personales.

- **Interacción visual:** El usuario recibe respuestas guiadas mediante botones interactivos en conversaciones más estructuradas.

- **Interacción semántica:** El sistema distingue entre consultas simples y complejas, ajustando la respuesta según el nivel de detalle requerido.
- **Interacción funcional:** Se gestionan accesos rápidos a documentos institucionales, evitando que el usuario tenga que navegar en varias páginas.

Desde la perspectiva de diseño

Un reto clave aquí es la **sincronización de sesiones**. Como múltiples usuarios pueden interactuar simultáneamente, el sistema debe garantizar respuestas eficientes sin degradar el rendimiento. También se cuida la **transición entre temas**, asegurando que el chatbot no confunda al usuario cuando cambia de consulta.

Desde los elementos técnicos

- **Clases clave:** Sesión, Consulta, BaseConocimiento.
- **Métodos relevantes:** crearSesion(), cerrarSesion(), obtenerContactos(), obtenerUbicaciones().
- **Estados:** Inicio, En Flujo, Finalizado.

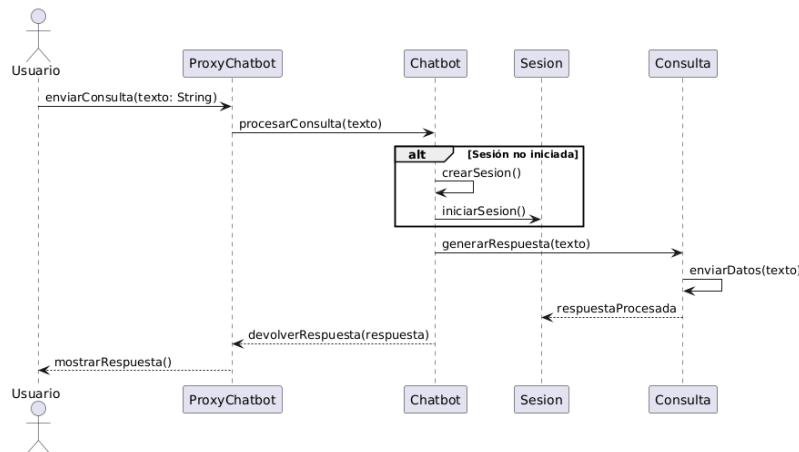


ILUSTRACIÓN 24 DIAGRAMA DE SECUENCIA 3

Procesamiento de consultas estructurado

Este diagrama explica cómo el chatbot analiza una pregunta antes de responder. Se asegura que la interacción siga un **flujo lógico**, optimizando el acceso a información.

- **Interacción visual:** La respuesta del chatbot aparece estructurada, ofreciendo opciones contextuales cuando es necesario.
- **Interacción semántica:** Se interpreta la intención del usuario antes de generar una respuesta.
- **Interacción funcional:** El sistema devuelve respuestas procesadas con enlaces o botones.

Desde la perspectiva de diseño

Aquí se destaca el **manejo de estados** dentro del flujo conversacional. Cuando el usuario cambia de tema, el sistema debe reorganizar la lógica de conversación sin perder coherencia.

Desde los elementos técnicos

- **Clases clave:** Consulta, RespuestaSistema, FlujoConversacional.
- **Métodos relevantes:** interpretarPregunta(), generarRespuesta(), mostrarOpciones().
- **Estados:** Inicio, En Flujo, Interrumpido, Finalizado.

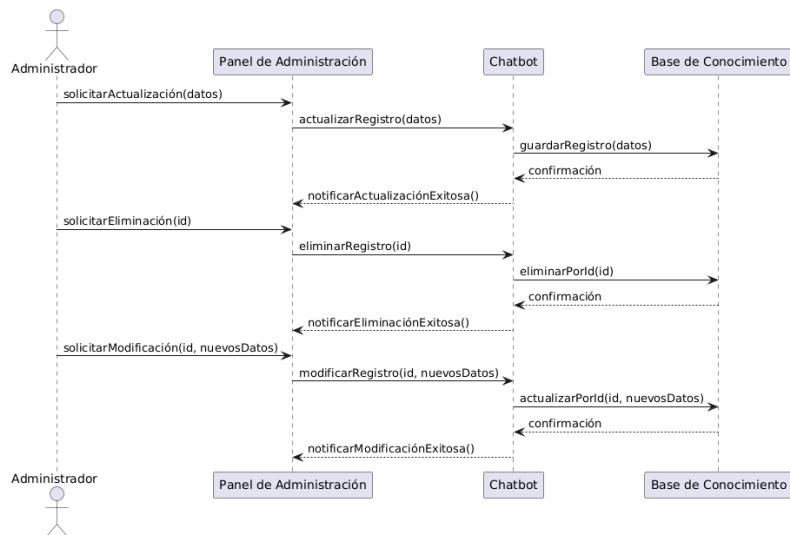


ILUSTRACIÓN 25 DIAGRAMA DE SENCUENCIA 4

Gestión y actualización de información institucional

Este diagrama muestra cómo los administradores pueden actualizar y modificar datos del chatbot. Esto permite ofrecer información actualizada a los usuarios.

- **Interacción visual:** La interfaz del panel de administración permite gestionar registros de manera sencilla.
- **Interacción semántica:** Se valida la coherencia de la información antes de modificarla.
- **Interacción funcional:** Se actualizan datos en la base de conocimiento sin afectar sesiones activas.

Desde la perspectiva de diseño

Un desafío clave aquí es **definir responsabilidades** entre los componentes del sistema. Debe garantizarse que la gestión de datos se haga de forma eficiente sin afectar el flujo de interacción del usuario.

Desde los elementos técnicos

- **Clases clave:** Administrador, PanelAdministración, BaseConocimiento.
- **Métodos relevantes:** solicitarActualización(), guardarRegistro(), notificarActualizaciónExitosa().
- **Estados:** Inicio, En Flujo, Finalizado.

13. Punto de vista dinámica de estados

El enfoque de **dinámica de estados** resulta fundamental para modelar su comportamiento frente a distintos eventos e interacciones que se presentan durante una conversación. Esta técnica permite representar de manera clara y estructurada los **cambios de estado** que sufre el sistema conforme los usuarios lo utilizan. Cada estado refleja una **situación específica** dentro del flujo de conversación: desde el inicio de una sesión hasta la resolución de una consulta o el cierre de la misma por inactividad.

Este enfoque no solo mejora la comprensión del sistema durante la etapa de diseño, sino que también es esencial para mantener la **coherencia, robustez y escalabilidad** de la lógica conversacional. A través de la identificación de estados y las transiciones entre ellos, es posible prever cómo debe comportarse el chatbot en diversas circunstancias, como cuando recibe una nueva pregunta, cambia de tema o se enfrenta a una consulta que no puede interpretar.

Las transiciones entre estados pueden ser provocadas tanto por **eventos externos** (como la entrada de texto por parte del usuario o la pulsación de un botón) como por **eventos internos** (como la detección de una intención, el fin de una conversación o la expiración del tiempo de espera).

13.1 Problemas de diseño

El uso de diagramas de estado para modelar el comportamiento del chatbot trae consigo importantes beneficios, pero también ciertos **retos de diseño** que deben considerarse cuidadosamente para evitar errores durante la implementación:

Ambigüedad

Uno de los principales problemas al diseñar la dinámica de estados es la **ambigüedad semántica**. Si no se define con claridad qué significa cada estado (por ejemplo, "esperando respuesta del usuario", "en flujo activo", "respondiendo error", etc.) y qué condiciones activan las transiciones, pueden surgir múltiples interpretaciones tanto entre diseñadores como en la lógica programada. Esto puede ocasionar respuestas

inconsistentes o flujos de conversación que se comporten de forma inesperada, afectando negativamente la experiencia del usuario.

Dificultad para mantener la consistencia

A medida que el chatbot se expande y se le añaden nuevos flujos de conversación, es cada vez más complejo **mantener la coherencia entre todos los estados y transiciones**. Por ejemplo, si se agregan nuevas rutas de interacción (como información sobre becas o servicios estudiantiles), los diagramas existentes deben ser actualizados para reflejar esos cambios, cualquier omisión puede provocar rupturas en el flujo conversacional. Para evitarlo, es necesario establecer buenas prácticas de documentación, control de versiones y validación de estados.

Complejidad

Los diagramas de estado tienden a volverse **muy complejos** cuando el sistema admite múltiples temas, intenciones y tipos de usuarios. En el caso del chatbot, esto se traduce en una red densa de estados posibles: esperando entrada, mostrando sugerencias, redirigiendo, confirmando acciones, cerrando sesión, etc. Esta complejidad puede dificultar tanto la comprensión del sistema como su mantenimiento futuro, por lo que es clave adoptar una estructura jerárquica y modular que permita dividir los flujos en subestados manejables.

13.2 Elementos de diseño

En el contexto del sistema de chatbot universitario, los **elementos de diseño** representan las unidades clave que permiten modelar de forma estructurada y funcional el comportamiento dinámico del sistema. Estos elementos se centran en capturar el flujo conversacional, las condiciones que lo gobiernan y los posibles resultados ante distintos tipos de interacción.

A continuación, se describen los principales componentes:

Entidades de Diseño

Estados: Representan los distintos momentos o condiciones en las que se puede encontrar el chatbot durante una sesión conversacional. Cada estado refleja una etapa en la interacción del usuario, como, por ejemplo:

- Estado de bienvenida: el chatbot recibe al usuario y presenta opciones generales.
- Estado de espera de entrada: el sistema está a la espera de que el usuario escriba una solicitud.
- Estado de procesamiento: se está interpretando la intención del usuario mediante técnicas de NLP.
- Estado de respuesta exitosa: se ha encontrado una respuesta satisfactoria a la consulta del usuario.
- Estado de no reconocimiento: el chatbot no logró comprender la entrada y propone reformulación o asistencia adicional.
- Estado de cierre de sesión: se ha finalizado la conversación, ya sea por voluntad del usuario o por inactividad.

Eventos: Son las acciones que provocan una transición de un estado a otro. Estos eventos pueden ser externos, como la introducción de texto por parte del usuario, o internos, como la detección de una intención específica o el vencimiento del tiempo de espera. Ejemplos:

- Usuario escribe una pregunta válida
- Usuario solicita cambiar de tema
- No se recibe respuesta en cierto tiempo
- Se detecta una entrada ofensiva o sin sentido

Transiciones: Conectan un estado con otro. Indican cómo el chatbot pasa de una condición a otra como resultado de un evento. Por ejemplo, si el usuario ingresa una pregunta clara, el sistema transita del estado *esperando entrada* al estado *procesando intención*.

Atributos de Diseño

Cada estado y transición posee atributos que definen su comportamiento y condiciones de activación. Estos incluyen:

- **Nombre del estado:** etiqueta clara y semántica que permite identificar el rol del estado (ej. EsperandoEntrada, RespondiendoPregunta, SinReconocimiento).
- **Tipo de interacción esperada:** especifica si el estado espera texto libre, selección de botones, o acciones del sistema.
- **Propósito:** define el objetivo funcional del estado, como obtener información adicional, confirmar una solicitud o proporcionar una respuesta.
- **Condiciones de transición:** criterios lógicos o de tiempo que activan el paso a otro estado. Por ejemplo, “si no se recibe entrada en 30 segundos, transitar a estado de advertencia de cierre”.

Restricciones de Diseño

- **Restricciones de interfaz:** se asegura que el usuario vea elementos interactivos apropiados según el estado (por ejemplo, botones en estados de selección múltiple, o formularios en estados de registro de datos).
- **Restricciones de reutilización:** algunos flujos (como respuestas frecuentes o preguntas comunes) pueden ser invocados desde múltiples estados sin duplicar lógica.
- **Restricciones de dependencia:** ciertos estados solo pueden alcanzarse si se ha pasado antes por otros (por ejemplo, no se puede acceder a información académica sin haber validado la pregunta).

13.3 Idiomas de ejemplo

Diagrama General de Flujo de Conversación (Ejemplo conceptual)

[Inicio]

↓ (Mensaje de usuario recibido)

[Esperando Entrada]

↓ (Entrada clara)

[Procesando Intención]

↓ (Coincidencia encontrada)

[Respondiendo]

↓

[Esperando Nueva Entrada]

↓ (Inactividad prolongada)

[Advertencia de Cierre]

↓ (Sin respuesta)

[Cierre de Sesión]

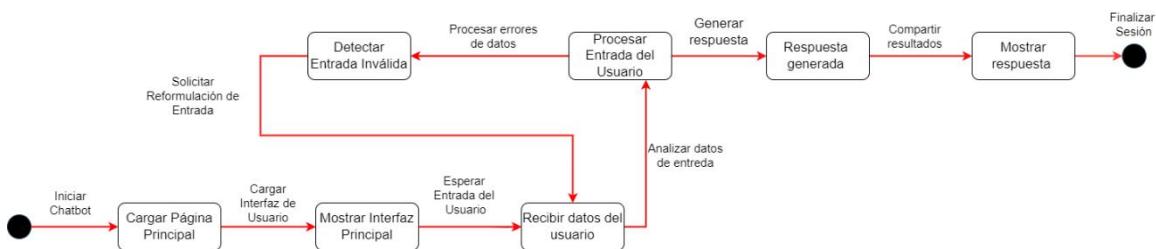


ILUSTRACIÓN 26 DIAGRAMA DE ESTADOS 1

Flujo de trabajo del chatbot desde el inicio hasta la finalización de la sesión

Este diagrama representa cómo el chatbot gestiona una conversación con el usuario, abarcando desde el momento en que se inicia la interacción hasta la finalización de la sesión.

- **Interacción visual:** Se muestra una interfaz intuitiva donde el usuario puede ingresar su solicitud y recibir respuestas de manera clara.
- **Interacción semántica:** Se analiza la entrada del usuario para garantizar que la conversación fluya de manera coherente y estructurada.

- **Interacción funcional:** El chatbot administra cada fase de la conversación, incluyendo la espera de entrada, el procesamiento de datos y la generación de respuestas.

Desde la perspectiva de diseño

El diseño del sistema se basa en un modelo estructurado de estados y transiciones para garantizar que la conversación sea lógica y efectiva.

- **Definición de estados:** Se organizan las etapas clave del chatbot, desde el estado inicial hasta la presentación de respuestas y la detección de errores.
- **Gestión de errores:** Se establece un mecanismo para detectar entradas inválidas, pedir reformulaciones y evitar bloqueos en el flujo de la conversación.
- **Consistencia en el diálogo:** Se implementan estructuras jerárquicas que permiten manejar múltiples interacciones sin perder coherencia.

Desde los elementos técnicos

El sistema se compone de clases y métodos diseñados para gestionar la comunicación entre el chatbot y el usuario.

- **Clases clave:** Chatbot, ProcesadorEntrada, InterfazUsuario.
- **Métodos relevantes:** capturarEntrada(), procesarMensaje(), generarRespuesta().
- **Estados principales:**
 - Inicio: Se activa el chatbot y se prepara la sesión.
 - EsperandoEntrada: Se aguarda la interacción del usuario.
 - ProcesandoEntrada: Se analiza la solicitud del usuario para definir una respuesta.
 - DetectandoErrores: Se identifica si la entrada del usuario es inválida.
 - SolicitandoReformulación: Se le pide al usuario que reformule su mensaje si la solicitud es ambigua.

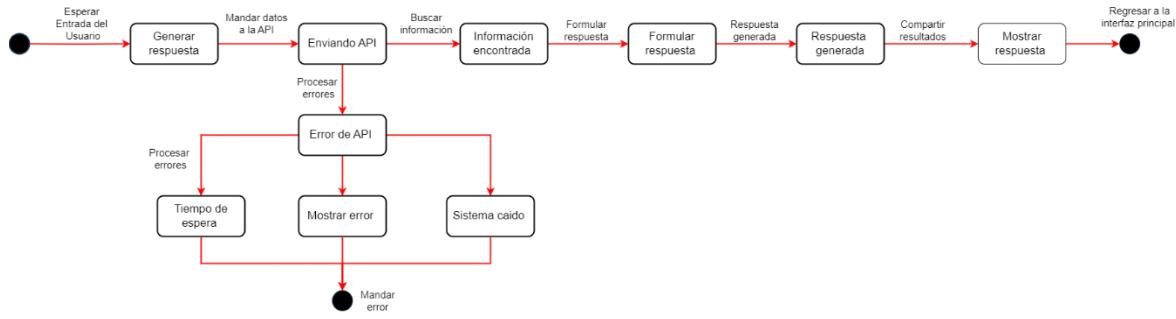


ILUSTRACIÓN 27 DIAGRAMA DE ESTADOS 2

Diagrama: Interacción del Chatbot con la API y Gestión de Errores

Este diagrama ilustra cómo el chatbot gestiona una conversación y consulta una API para obtener respuestas, asegurando que las interacciones sean estructuradas y manejables.

- **Interacción visual:** La interfaz muestra respuestas generadas por la API, indicando al usuario los cambios en el estado de procesamiento.
- **Interacción semántica:** Se analiza la intención del usuario antes de enviar la consulta a la API, asegurando coherencia en la comunicación.
- **Interacción funcional:** La interacción fluye entre los estados principales (entrada, procesamiento, respuesta), mientras se manejan errores y retrasos en la comunicación con la API.

Desde la perspectiva de diseño

El diseño de este sistema implica varios desafíos relacionados con el procesamiento de datos y la gestión de errores.

- **Definición de estados y transiciones:** Cada estado debe estar bien definido para evitar inconsistencias en la conversación. Se estructuran estados como "Esperando entrada", "Procesando intención" y "Generando respuesta".
- **Manejo de errores:** Se establece una serie de pasos en caso de fallos en la comunicación con la API, asegurando que el sistema pueda recuperarse sin bloquear la interacción del usuario.

- **Optimización del procesamiento:** Se implementan estructuras que minimizan la latencia en la consulta de datos, agilizando la respuesta del chatbot.

Desde los elementos técnicos

El chatbot emplea diversas clases y métodos para gestionar la interacción con la API y manejar errores.

- **Clases clave:** Chatbot, ManejadorAPI, ProcesadorErrores.
- **Métodos relevantes:** enviarSolicitudAPI(), interpretarRespuesta(), manejarError().
- **Estados principales:**
 - EsperandoEntrada: El sistema aguarda una solicitud del usuario.
 - GenerandoRespuesta: Se construye la respuesta basada en el procesamiento de la API.
 - ProcesandoErrores: Se detectan y manejan errores si la API no responde correctamente.
 - SistemaCaído: Se identifica una falla crítica en la API, impidiendo continuar la conversación.
 - MostrarRespuesta: El chatbot presenta la respuesta final al usuario.

14. Punto de vista del algoritmo

En esta sección se describen con mayor profundidad los métodos, funciones y estructuras lógicas que gobiernan el funcionamiento del chatbot implementado en el sistema. El diseño del algoritmo contempla tanto las tareas centrales del flujo conversacional como el procesamiento de entradas, interpretación de intenciones, generación de respuestas, así como la transición entre diferentes estados de interacción. Además, se integran mecanismos de validación, control de errores u optimización de la experiencia del usuario, manteniendo en todo momento la seguridad tanto como la privacidad de los datos, ya que este sistema **no almacena información personal de los usuarios.**

Cada bloque funcional del algoritmo ha sido diseñado para responder de forma precisa, eficiente y modular a los distintos tipos de consultas que pueden surgir durante una conversación. La estructura también permite su mantenimiento, así como escalabilidad en futuras versiones del sistema.

14.1 Problemas de diseño

Durante el desarrollo del algoritmo, pueden surgir diversos retos técnicos y funcionales. A continuación, se presentan los principales problemas de diseño identificados y las medidas adoptadas para abordarlos:

Eficiencia

Uno de los principales retos es asegurar que el chatbot procese cada solicitud en el menor tiempo posible, incluso cuando el número de usuarios concurrentes aumenta o se consultan múltiples módulos. Para ello, se ha optimizado el flujo de decisiones mediante árboles de decisión, manejo asincrónico de solicitudes, y uso de estructuras condicionales ligeras, evitando sobrecargas innecesarias.

Además, las respuestas del sistema se generan en tiempo real gracias al uso de filtros inteligentes que agrupan las intenciones comunes, evitando múltiples evaluaciones.

Escalabilidad

El algoritmo está estructurado con un enfoque modular, lo que permite extender fácilmente su funcionalidad sin necesidad de modificar la lógica principal. Por ejemplo, si se desea agregar un nuevo módulo de orientación académica o de contacto con departamentos específicos, basta con definir nuevas funciones de manejo de intenciones y conectarlas al sistema mediante reglas conversacionales.

La arquitectura del chatbot está preparada para manejar más temas, más usuarios simultáneamente y un mayor volumen de peticiones, sin comprometer la estabilidad del sistema.

Complejidad

Aunque el sistema está diseñado para abordar consultas variadas, además de potencialmente complejas, la lógica interna se ha mantenido legible por lo que es clara. Se han utilizado funciones con responsabilidades específicas (principio de responsabilidad única), lo que facilita el mantenimiento del código, las pruebas y futuras actualizaciones.

Cada componente del algoritmo está documentado para que otros desarrolladores puedan entenderlo y adaptarlo a nuevos requerimientos, sin comprometer el flujo original del chatbot.

14.2 Elementos de diseño

A continuación, se describen los principales elementos que componen el diseño algorítmico del chatbot, organizados por funcionalidad y jerarquía de ejecución:

1. Funciones Principales del Chatbot

- capturarEntradaUsuario()**

Función encargada de recibir el texto ingresado por el usuario desde la interfaz y enviarlo al motor de procesamiento. Incluye validaciones para evitar entradas vacías o maliciosas.

- analizarIntencion(mensaje)**

Utiliza reglas semánticas predefinidas y procesamiento básico del lenguaje

natural (NLP) para determinar la intención de la pregunta. Clasifica entradas en categorías como: horarios, ubicación de departamentos, contacto con personal, entre otros.

- **generarRespuesta(intención)**

Con base en la intención detectada, genera una respuesta estructurada. En algunos casos, redirige la conversación hacia un nuevo módulo o estado.

- **renderizarRespuesta ()**

Muestra la respuesta generada en la interfaz del usuario y, si es necesario, incluye botones de acción para continuar la conversación de forma interactiva.

2. Funciones Auxiliares

- **validarLenguajeOfensivo ()**

Escanea las entradas del usuario en busca de términos inapropiados. Si detecta lenguaje ofensivo, bloquea la conversación y emite una advertencia.

- **gestionarErrores ()**

Maneja situaciones donde el sistema no puede identificar una intención. Ofrece opciones como: repetir la pregunta, acceder a ayuda o consultar con personal humano.

- **transicionarEstado (actual, evento)**

Lógica que cambia de un estado conversacional a otro en función del evento recibido. Por ejemplo, del estado de *espera* al estado de *respuesta directa*.

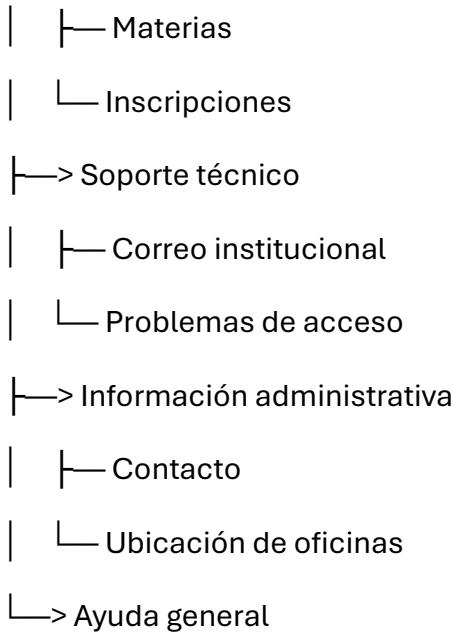
3. Estructura de Decisión (Árbol de Intenciones)

Inicio

|—> Saludo

|—> Consulta académica

| |—> Horarios



Este árbol se recorre dinámicamente según la entrada del usuario, lo que permite mantener una lógica conversacional fluida, por lo que no tiene necesidad de repetir información.

4. Condiciones de Transición y Respuesta

Cada transición entre estados dentro del flujo conversacional está gobernada por condiciones lógicas como:

- Palabras clave encontradas.
- Ausencia de información.
- Comportamiento anterior del usuario (sin almacenarlo).
- Selección de botones interactivos.

Además, la salida de cada función se adapta al estado anterior, lo que permite continuidad conversacional y evita repeticiones innecesarias.

5. Restricciones de Seguridad y Privacidad

- **No se almacena ningún dato del usuario.**

Todas las interacciones se procesan en tiempo real y se destruyen tras

finalizar la conversación. Esto asegura la privacidad total de los estudiantes y cumple con principios de diseño ético.

- **Control de sesiones anónimas.**

Cada usuario interactúa con el sistema sin autenticación obligatoria, salvo que se acceda a módulos internos protegidos. En este caso, la verificación se realiza sin retener información personal.

14.3 Idiomas de ejemplos

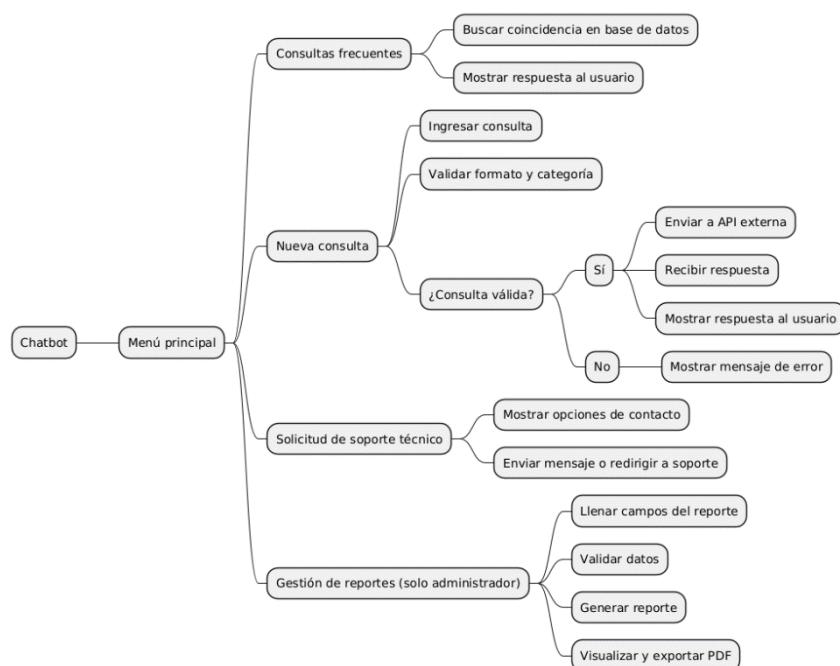


ILUSTRACIÓN 28 DIAGRAMA DE DECISIONES

1. Estructura Principal y Flujo de Decisión

El chatbot sigue un **enfoque modular** con un árbol de decisiones jerárquico:

- **Entrada del usuario:** La función `capturarEntradaUsuario()` recibe y sanitiza el texto, evitando entradas vacías o maliciosas.
- **Procesamiento de intención:** `analizarIntencion(mensaje)` clasifica la consulta usando reglas semánticas o NLP básico (ej: "horarios" → *Consulta académica*).

- **Generación de respuestas:** Según la intención, se activan rutinas específicas:
 - **Consultas frecuentes:** Búsqueda en una base de datos local (eficiente para respuestas rápidas).
 - **Nuevas consultas:** Validación de formato y envío a una API externa si son válidas (manejo asincrónico para evitar cuellos de botella).
 - **Soporte técnico:** Transición a un estado conversacional con opciones de contacto humano.
 - **Reportes (admin):** Funciones restringidas con validación de datos antes de generar PDFs.

2. Mecanismos Clave del Algoritmo

- **Validaciones y control de errores:**
 - validarLenguajeOfensivo() filtra contenido inapropiado.
 - gestionarErrores() redirige consultas no reconocidas a ayuda o soporte humano.
- **Transiciones de estado:**
 - La función transicionarEstado(actual, evento) cambia entre estados (ej: de *menú principal* a *nueva consulta*) basándose en palabras clave o acciones del usuario.
- **Eficiencia y escalabilidad:**
 - **Árbol de intenciones:** Organiza las opciones en una jerarquía (ej: *Consulta académica* → *Horarios*), reduciendo la carga de procesamiento.
 - **Módulos independientes:** Cada función (ej: generarRespuesta(), renderizarRespuesta()) sigue el *principio de responsabilidad única*, facilitando actualizaciones.

- **Seguridad:**
 - **Sin almacenamiento de datos:** Las interacciones se procesan en tiempo real y se descartan.
 - **Acceso diferenciado:** Los módulos de administrador requieren autenticación sin retener información personal.
-

3. Problemas de Diseño Resueltos

- **Eficiencia:** Uso de árboles de decisión y procesamiento asincrónico para manejar múltiples usuarios.
- **Escalabilidad:** Arquitectura modular que permite añadir nuevas funcionalidades (ej: un módulo de "pagos") sin alterar el núcleo.
- **Complejidad:** Documentación clara y funciones especializadas para mantener el código mantenable.

Conclusión: El algoritmo del chatbot está diseñado para ser **rápido, seguro y adaptable**, priorizando la experiencia del usuario, la escalabilidad futura. Su estructura refleja un equilibrio entre complejidad funcional y simplicidad en el mantenimiento.