ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UN PROTOTIPO DE CHATBOT PARA RESPONDER PREGUNTAS FRECUENTES DE LA CARRERA DE TECNOLOGÍAS DE LA INFORMACIÓN

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

GISELA ELIZABETH OSORIO TIBÁN

gisela.osorio@epn.edu.ec

DIRECTOR: ING. RAÚL DAVID MEJÍA NAVARRETE, MSc.

david.mejia@epn.edu.ec

Quito, Agosto 2019

AVAL

Certifico que el presente trabajo fue desarrollado por Gisela Elizabeth Osorio Tibán, bajo mi supervisión.

ING. RAÚL DAVID MEJÍA NAVARRETE, MSc.
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, Gisela Elizabeth Osorio Tibán, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.

_____

Gisela Elizabeth Osorio Tibán

DEDICATORIA

A mis papis, Laura y Guillermo, por cuidarme, guiarme y llenarme de amor.

AGRADECIMIENTO

Agradezco a Dios y a la Virgen de Guadalupe.

A mi mami Laura por su amor; por la alegría que me transmite; por acompañarme en las madrugadas y en los desvelos, siempre atenta, siempre motivándome a marcar la diferencia y a ser cada día mejor.

A mi papi Guillermo por su amor; por su entusiasmo; por su apoyo incondicional y por estar siempre dispuesto a ayudarme sin importar el día, la hora o el lugar.

A mi hermano Rafael, eres un ángel en mi vida.

A mis amigos Homerito, Dariito, Ericksito, Sofi, Jimmy, Abuelo, Nico y Alarmitas por compartir conmigo tantos y tantos momentos; todos ustedes hicieron que estudiar en la poli sea lindo y memorable. A Mauricito por su alegría y por el cariño que me brinda.

Un especial agradecimiento al Ingeniero David Mejía por el tiempo y energía invertidos en guiar, leer y corregir este Trabajo de Titulación. Su apoyo fue esencial para la culminación de este proyecto.

A los profesores de la Escuela Politécnica Nacional.

Giss Osorio

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE CÓDIGOS

RESUMEN

El presente Trabajo de Titulación se centra en el desarrollo de un prototipo de Chatbot para responder preguntas frecuentes de la Carrera de Tecnologías de la Información; por lo menos referentes a los procesos de matriculación, titulación y otros temas determinados en la etapa de análisis. El chatbot podrá incluirse en cualquier página web, consumirá servicios WCF y usará una base de datos donde se almacenará su información.

En el primer capítulo se describen a los chatbots; se presentan tecnologías como AIML, HTML, JavaScript, AJAX y servicios WCF. Además se presentan conceptos sobre herramientas como Grampal y Selenium WebDriver. Después, se detalla la metodología ágil Kanban.

El segundo capítulo, plantea tres etapas: análisis, diseño e implementación. En el análisis se especifican los requerimientos funcionales y no funcionales de la aplicación, y se definen los temas específicos que el chatbot será capaz de responder. En la etapa de diseño se presenta la arquitectura del chatbot y los diferentes diagramas. En la etapa de Implementación se presenta la codificación.

El tercer capítulo describe los resultados tanto de las pruebas de integración de componentes como de las pruebas de funcionalidad. Además, se corrigen los errores presentados.

En el cuarto capítulo se presentan las conclusiones y recomendaciones generadas durante el desarrollo de este Trabajo de Titulación.

Finalmente, los anexos incluyen el modelo de entrevista inicial y final, el *script* de base de datos, el manual de usuario de la aplicación de administración, el proyecto de la aplicación de administración y del chatbot.

PALABRAS CLAVE: chatbot, servicios WCF, AIML, JavaScript, Grampal.

ABSTRACT

The current project focuses on the development of a Chatbot's prototype to answer frequently asked questions of the Information Technologies Career; at least about the academic enrollment, academic degree and other topics defined in the requirements analysis. The chatbot can be included in any web page, consumes WCF services and uses a database where it information will be stored.

The first chapter describes chatbots or conversational agents; presents Technologies such as AIML, HTML, JavaScript, AJAX and WCF services. In addition, this chapter presents concepts about tools such as Grampal and Selenium WebDriver. Afterwards, the agile Kanban methodology is detailed.

The second chapter proposes three phases: analysis, design and implementation. The analysis specifies the functional and non-functional requirements and defines the frequent questions that the chatbot will be able to answer. The design phase presents the Chabot's architecture and the different diagrams. Then, in the Implementation phase the coding is presented.

The third chapter describes the results the integration tests and the results of the functionality tests. In addition, presents the correction of the errors.

The fourth chapter presents the conclusions and recommendations generated during the development of this project.

Finally, the attachments include the initial interview model, the database script, the administration application project, the chatbot project, the final interview model and the user manual.

KEYWORDS: chatbot, WCF services, AIML, JavaScript, Grampal.

1. INTRODUCCIÓN

En la actualidad las instituciones educativas están haciendo uso de herramientas de inteligencia artificial para ofrecer apoyo estudiantil al agilizar las interacciones entre los estudiantes y el servicio administrativo, brindando asistencia oportuna. Un chatbot es un canal de mensajería instantánea que permite al usuario comunicarse de forma directa e inmediata e intercambiar amplia información para solventar dudas en cualquier momento y desde cualquier lugar [1], [2].

Hoy en día los estudiantes y el público en general deben acercarse a la Coordinación de la Carrera de Electrónica y Redes de Información para realizar preguntas referentes a los procesos de matriculación, titulación, entre otros.

Por este motivo, tanto el personal administrativo de la Coordinación de la Carrera así como los estudiantes, utilizan un periodo de su tiempo para solventar preguntas que podrían ser resueltas inmediatamente por medio de un chatbot; optimizando así su tiempo y recursos en otras actividades y servicios.

Con el desarrollo de este Trabajo de Titulación se plantea contar con un canal de comunicación directo y frecuente entre la Coordinación de la Carrera, los estudiantes y el público en general.

El enfoque de este Trabajo de Titulación es desarrollar un chatbot que permita responder preguntas frecuentes de la carrera de Tecnologías de la Información, al menos referentes a los procesos de matriculación y titulación. Además, de contar con una aplicación de administración que permita que el chatbot brinde información actualizada conforme el periodo académico en curso.

En este capítulo se describirán el objetivo general, los objetivos específicos, el alcance y el marco teórico. En el marco teórico se detallarán algunos conceptos referentes a la Inteligencia Artificial como el Procesamiento de Lenguaje Natural y la Generación de Lenguaje Natural. Además, se presentarán las tecnologías y herramientas usadas en el desarrollo de este Trabajo de Titulación. Finalmente, se describirá la metodología ágil Kanban.

1.1 OBJETIVOS

El objetivo general de este Trabajo de Titulación es desarrollar un prototipo de Chatbot para responder preguntas frecuentes de la Carrera de Tecnologías de la Información.

Los objetivos específicos de este Trabajo de Titulación son:

- Analizar el funcionamiento de algunas técnicas de inteligencia artificial.
- Diseñar los módulos y base de datos necesarios para la implementación del chatbot.
- Implementar el diseño propuesto para el chatbot.
- Analizar los resultados obtenidos en las pruebas realizadas a personal administrativo y a estudiantes.

1.2 ALCANCE

Este Trabajo de Titulación propone el desarrollo de un complemento para una página web, el cual se lo denominará chatbot y permitirá responder preguntas frecuentes.

El chatbot se implementará mediante JavaScript[1] y podrá ser embebido o incluido en cualquier página web. Además, se comunicará por medio de solicitudes y respuestas con un servidor web y usará una base de datos, en la cual se almacenará la información que el chatbot emplea.

El chatbot permitirá al usuario establecer una conversación con el fin de realizar preguntas y obtener respuestas, por lo menos referentes a los procesos de: matriculación, titulación y otros temas que se determinen en el análisis de requerimientos.

Para que el chatbot realice estas tareas se necesita implementar cuatro módulos principales. Un ejemplo del esquema que se plantea realizar se presenta en la Figura 1.1. El primer módulo permitirá el entendimiento del lenguaje, procesando el mensaje introducido por el usuario, analizando su composición y significado.

El segundo módulo guardará el registro de la conversación, lo que permitirá al chatbot entender nuevos mensajes.

El tercer módulo permitirá seleccionar la respuesta más adecuada conforme el mensaje recibido. En este módulo se realizarán las consultas a la base de datos.

[1] JavaScript: Lenguaje de programación orientado a objetos e interpretado.

Finalmente, el módulo generador de lenguaje natural seleccionará la respuesta más adecuada conforme el contexto de la conversación y políticas de diálogo [1].

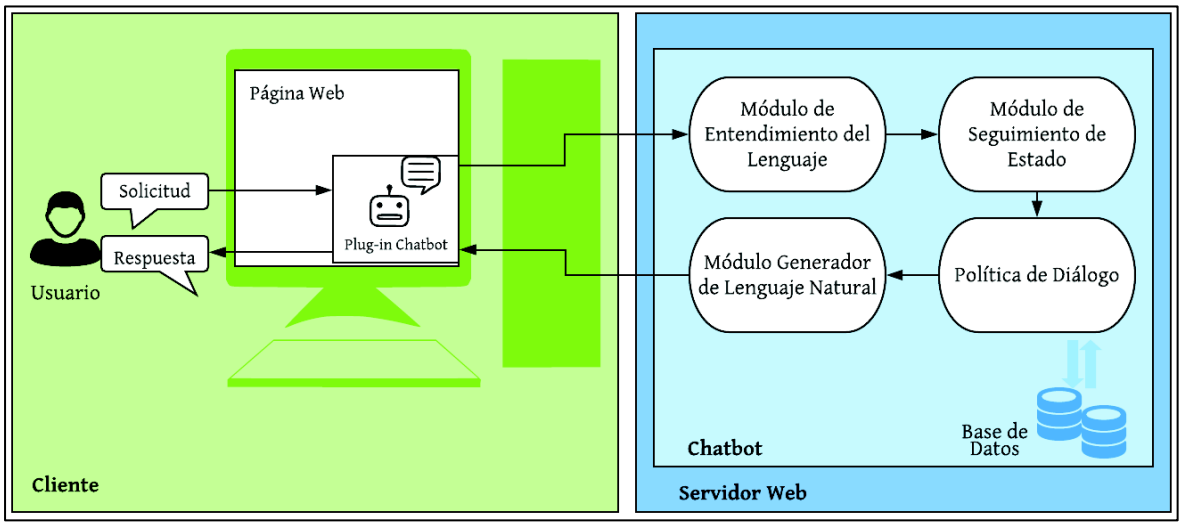


Figura 1.1 Esquema del chatbot propuesto

1.3 MARCO TEÓRICO

1.3.1 CHATBOT

Un Chatbot o agente conversacional es un programa capaz de mantener un diálogo con una persona mediante el uso de lenguaje natural [2]. Existen dos modelos de agentes conversacionales principales: modelos basados en la recuperación y modelos generativos.

En los modelos basados en la recuperación, se selecciona la respuesta conforme la mejor coincidencia con los patrones de la base de datos de conocimiento del chatbot; mientras que los modelos generativos crean nuevas respuestas con base en la recopilación de información, entrenamiento y pruebas [4].

1.3.1.1 Modelo de un chatbot basado en la recuperación

El programa recibe la oración enviada por el usuario, la procesa y busca una coincidencia entre dicha oración y los patrones almacenados en una base de datos de conocimiento. Finalmente, el chatbot envía al usuario la respuesta más apropiada, cabe recalcar que conforme más patrones y respuestas se tengan almacenados en la base de datos, más certeras serán las respuestas del chatbot [3]. Además, estos sistemas no crean respuestas nuevas por lo que si un usuario envía un mensaje que no coincida con ningún patrón, se responderá con un texto previamente definido. En la Figura 1.2 se puede apreciar un modelo de un chatbot basado en la recuperación que consta de tres

elementos: interfaz de usuario, motor de inferencia y base de conocimiento. El motor de inferencia realiza el Procesamiento de Lenguaje Natural para interpretar el mensaje del usuario y elegir la respuesta más acertada (generación de lenguaje natural). Además, en esta arquitectura están presente dos actores: el experto humano (quién introduce la información a la base de conocimiento del chatbot) y el usuario (quién interactuará con la aplicación) [4].

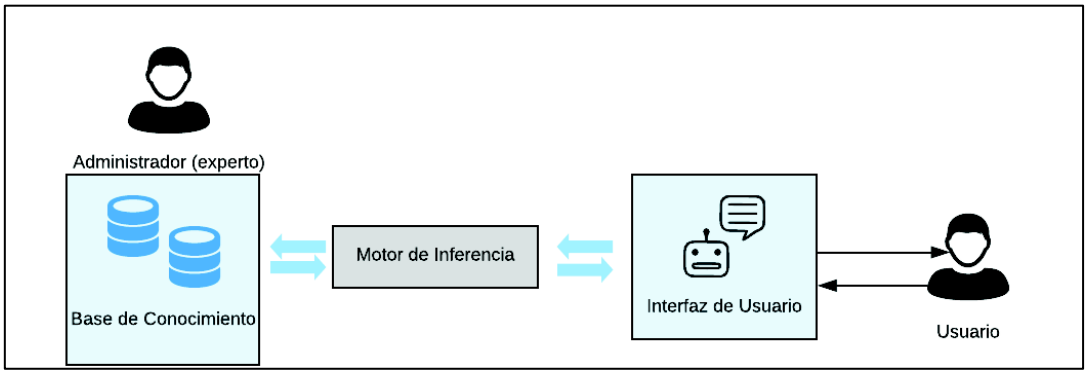


Figura 1.2 Modelo de un chatbot basado en la recuperación

1.3.1.2 Procesamiento de Lenguaje Natural

El Procesamiento del Lenguaje Natural es un campo de conocimiento de la Inteligencia Artificial[2], cuyo objetivo es la comunicación entre programas informáticos y personas mediante lenguajes naturales [3]. Lenguaje natural es el lenguaje que las personas usan diariamente para comunicarse.

Los componentes del Procesamiento de Lenguaje Natural son: el análisis morfológico (léxico) encargado de analizar cada una de las palabras que forman la oración, extrayendo lemas, rasgos, categorías; el análisis sintáctico que analiza la estructura de la oración conforme el modelo gramatical seleccionado; el análisis semántico que analiza el significado de las palabras de la oración y el análisis pragmático que analiza el contexto de la oración. El objetivo de realizar este análisis es interpretar el lenguaje humano.

1.3.1.3 Generación de Lenguaje Natural

Los chatbots presentan información a los usuarios en lenguaje natural. Existen diferentes métodos para generar mensajes en lenguaje natural, con distintos grados de complejidad. Entre los principales métodos se tiene a la Heurística basada en patrones, que son patrones respuestas previamente definidos, y, el método de generación de lenguaje

[2] Inteligencia Artificial: Programa informático que realiza tareas propias de la inteligencia humana.

natural basado en el conocimiento, donde el sistema genera todos los mensajes de forma dinámica basándose en la recopilación de información, entrenamiento y pruebas [6].

1.3.2 TECNOLOGÍAS

1.3.2.1 AIML (Artificial Intelligence Mark-up Language)

Es un lenguaje de programación basado en XML[3] que permite crear aplicaciones con capacidad de conversación [4]. AIML usa patrones, respuestas, etiquetas y caracteres comodín para emular una conversación proveyendo las respuestas que una persona proporcionaría, además no distingue entre mayúsculas y minúsculas. Los caracteres comodín (asterisco y guion bajo) permiten coincidir un patrón con cualquier texto introducido por el usuario.

En la Tabla 1.1 se muestran las etiquetas más usadas en los documentos AIML.

Tabla 1.1 Algunas etiquetas AIML

Etiqueta	Descripción
`<aiml>`	Indica el inicio de un documento AIML
`<category>`	Establece la unidad de conocimiento
`<pattern>`	Establece los patrones que el usuario podrá ingresar
`<template>`	Define las respuestas a las entradas del usuario
`<random>`	Permite tener respuestas aleatorias
`<li>`	Permite representar múltiples respuestas

En el Código 1.1 se presenta el contenido de un documento AIML, en la primera línea con la etiqueta `<aiml>` se indica el inicio del documento. En la tercera línea dentro de la etiqueta `<pattern>` se establece el patrón exacto que el usuario deberá escribir para que el chatbot responda, en este caso el patrón es `"hola"`. En la línea cinco por medio de la etiqueta `<random>` se proporciona respuestas aleatorias y en las líneas seis y siete se establecen las respuestas del chatbot; cabe recalcar que todas las etiquetas deben ser cerradas. Entonces si el usuario escribe `"hola"` se responderá aleatoriamente `"hola humano"` o `"Lindo día humano"`; en el caso que el usuario escriba otro patrón el chatbot no responderá. En la Figura 1.3 se muestra un ejemplo de una conversación con el chatbot basado en el documento AIML del Código 1.1.

[3] XML (*Extensible Markup Language*): Lenguaje de marcado usado para almacenar y transportar datos.

En el Código 1.2 se indica un ejemplo de documento AIML con caracteres comodín, si el usuario escribe "hola" seguido de cualquier texto el chatbot responderá aleatoriamente "hola humano" o "Lindo día humano". En la Figura 1.4 se muestra una conversación con el chatbot basado en el documento AIML del Código 1.2.

```
<aiml>
  <category>
    <pattern>hola</pattern>
    <template>
      <random>
        <li>hola humano</li>
        <li>Lindo día humano</li>
      </random>
    </template>
  </category>
</aiml>
```

Código 1.1 Ejemplo de documento AIML

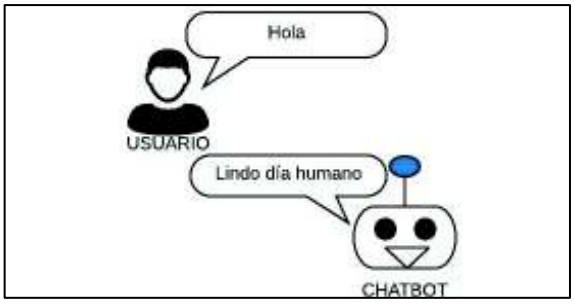


Figura 1.3 Conversación programada

```
<aiml>
  <category>
    <pattern>hola *</pattern>
    <template>
      <random>
        <li>hola humano</li>
        <li>Lindo día humano</li>
      </random>
    </template>
  </category>
</aiml>
```

Código 1.2 Ejemplo de documento AIML

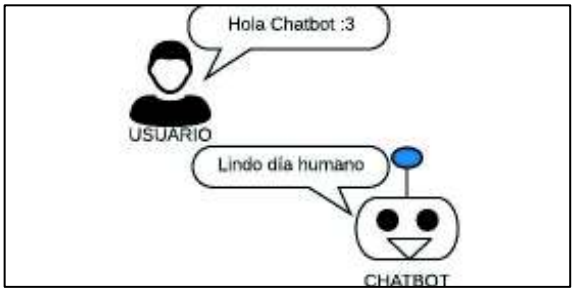


Figura 1.4 Conversación programada

1.3.2.2 Métodos del protocolo HTTP (Hypertext Transfer Protocol)

Es un protocolo cliente-servidor que mediante peticiones y respuestas permite intercambiar datos y recursos en la web [7]. Cuando un cliente (explorador web) desea comunicarse con un servidor: abre una conexión TCP[4] luego realiza la petición HTTP, recibe la respuesta enviada por el servidor y cierra la conexión TCP. La Tabla 1.2 muestra los campos de una petición HTTP y en la Tabla 1.3 se indican los campos de una respuesta HTTP.

Tabla 1.2 Campos de una petición HTTP

Campo	Descripción
Método HTTP	Define la operación que el cliente quiere realizar como: GET[5], POST[6]
URI[7]	Indica la dirección del recurso del pedido
Versión	Indica la versión del protocolo HTTP
Cabeceras HTTP	Cadena de caracteres que poseen un valor. Se pueden clasificar en tres grupos cabeceras: generales que afectan al mensaje completo, de petición que especifican detalles y de entidad que se aplican al cuerpo de la petición [8]
Cuerpo del mensaje	No todas las peticiones tienen cuerpo. La petición POST puede mandar información al servidor para actualizarlo [8]. Existen cuerpos con un único dato o con múltiples datos

Tabla 1.3 Campos de una respuesta HTTP

Campo	Descripción
Versión	Indica la versión del protocolo HTTP
Código de estado	Indica de forma abreviada la respuesta HTTP (respuestas afirmativas, correctas, redirecciones, errores del cliente o servidor)
Mensaje Estado	Corta descripción del código de estado
Cabecera HTTP	Cadena de caracteres que poseen un valor. Se pueden clasificar en tres grupos cabeceras: generales que afectan al mensaje completo, de petición que especifican detalles y de entidad que se aplican al cuerpo de la petición [8]
Cuerpo del mensaje	Representa la información solicitada, no todas las respuestas tienen cuerpo. Existen cuerpos con dato único o con múltiples datos [8]

[4] TCP (*Transport Control Protocol*): Protocolo confiable de la capa de transporte.
[5] GET: Método HTTP que solicita un recurso especificado.
[6] POST: Método HTTP que envía datos al servidor.
[7] URI (*Uniform Resource Identifier*): Formato estándar para identificar recursos en la red.

1.3.2.3 HTML (HyperText Markup Language)

Es un lenguaje que sirve para crear y representar páginas web, determina el contenido de la página más no su funcionalidad. Se utiliza el lenguaje HTML para crear páginas web y los navegadores web como: Chrome, Opera, entre otros; interpretan el contenido y lo muestran al usuario.

La sintaxis de HTML se basa en etiquetas, atributos y elementos. Las etiquetas son instrucciones escritas entre paréntesis angulares (< >). El Código 1.3 muestra una línea de código HTML, se puede ver que la etiqueta <h2> da inicio a la instrucción y para finalizar se escribe la misma etiqueta de inicio pero añadiéndole una barra diagonal </h2>; cabe resaltar que no todas las etiquetas tienen principio y fin. Las etiquetas HTML se pueden anidar es decir se puede usar una etiqueta dentro de otra [8].

```
<h2>Tecnologías de la Información</h2>
```

Código 1.3 Línea de código HTML

Para definir la estructura de un documento HTML se usa la etiqueta obligatoria <HTML>, que define el inicio y fin de un documento, y las etiquetas opcionales <HEAD> y <BODY>. En la Tabla 1.4 se describen algunas etiquetas HTML.

Tabla 1.4 Algunas etiquetas HTML

Etiqueta	Descripción
<HTML>	Define el inicio y fin de un documento HTML
<HEAD>	Define el área de cabeceras del documento HTML, la información que no se visualiza (título, estilo, etc.)
<BODY>	Incluye el contenido que el usuario visualiza en su pantalla (texto, imágenes, etc.)
<H*>	Párrafo encabezado, numerados del 1 al 6 por orden de relevancia
<P>	Párrafo normal
<TITLE>	Define el título de la ventana del navegador web

En el Código 1.4 se aprecia un ejemplo de documento HTML estructurado con las etiquetas básicas: <html>, <head> y <body>. En la cabecera del documento se usa la etiqueta <script> para hacer referencia al *script*[8] de jQuery[9], también se utiliza la

[8] *Script*: Conjunto de comandos guardados en un archivo de texto.

etiqueta `<title>` para colocar un título a la página web (título que aparecerá en la pestaña del navegador), además se aplican estilos CSS (*Cascading Style Sheets*) a través de la etiqueta `<link>` que vincula un archivo externo (`.css`) al documento HTML. Cabe mencionar que CSS describe la apariencia de la página web y JavaScript determina su funcionalidad. Finalmente en el cuerpo se indica la estructura y contenido de la página web, es decir se establecen los párrafos, botones, cajas de texto, etc. con sus respectivos nombres, identificadores y contenido. En la Figura 1.5 se muestra la página web que se ha creado mediante el código HTML del Código 1.4.

```
1   <!DOCTYPE html>
2   <!--ServiceClientPage.html-->
3   <html>
4   <head>
5       <script src="jquery-3.3.1.js" type="text/javascript"></script>
6       <title>gissOso.com</title>
7       <link rel="stylesheet" type="text/css" href="/Style.css" media="screen" /
8   </head>
9   <body>
10
11      <h2>Tecnologías de la Información</h2>
12      <div class="chatbotsito" id="idChatbotsito">
13          <h2>Chat</h2>
14          <input type="text" id="conversacion_pregunta" />
15          <input type="text" id="conversacion_respuesta" />
16          <input type="button" id="btnChatear" value="Chatear" />
17          <input type="button" id="btnGrandpa" value="Grandpa" />
18      </div>
19      <div class="grampal" id="idGrampal">
20          <h2>Grampal</h2>
21      </div>
22
23  </body>
24  </html>
```

Header

Body

Código 1.4 Estructura de un documento HTML

Figura 1.5 Página web HTML

[9] jQuery: Biblioteca de JavaScript.

1.3.2.4 JavaScript

JavaScript es un lenguaje de programación orientado a objetos, ligero e interpretado que permite crear contenido dinámico en una página web. Se lo utiliza del lado del cliente y del lado del servidor [10].

Los programas en JavaScript son conocidos como *scripts* que se ejecutan como texto plano y no necesitan ser compilados para ejecutarse. Los *scripts* se pueden escribir directamente dentro del código HTML de la página web o en un archivo externo con extensión `.js`. El Código 1.5 muestra un ejemplo de código JavaScript, en la línea 8 la etiqueta `<script>` indica el inicio del código JavaScript, en la línea 9 se observa el comando `$(document).ready()` que asegura que el documento se encuentre listo para cargar el código JavaScript pues no es seguro manipularlo si el DOM[10] (*Document Object Model*) no se encuentra listo [12], en la línea 10 se indica que se realizará una acción si se da clic en el botón llamado `btnChatear`.

```
8   <script>
9       $(document).ready(function () {
10          $("#btnChatear").click(function () {
11
12
13          });
14
15      });
16  </script>
```

Código 1.5 Ejemplo de código JavaScript

1.3.2.5 AJAX (Asynchronous JavaScript And XML)

Describe el modo de usar un conjunto de tecnologías para desarrollar aplicaciones web que puedan actualizarse sin necesidad de cargar la página completa [5].

En una aplicación web tradicional el cliente envía una petición HTTP al servidor; el servidor procesa la solicitud y devuelve una respuesta; durante este proceso la página web permanece bloqueada; finalmente al recibir la respuesta el navegador carga nuevamente toda la página [6]. En una aplicación web AJAX al realizar una solicitud asíncrona[11] la página web se va actualizando conforme se reciben las respuestas; en todo momento el usuario puede interactuar con la página web sin necesidad de

[10] DOM (*Document Object Model*): Interfaz de programación para documentos HTML, define la estructura lógica y el modo en que se accede y manipula el documento.
[11] Solicitud asíncrona: Operación que mientras se procesa no bloquea otras operaciones.

recargarla proporcionando interactividad a la misma. La Figura 1.6 muestra la estructura del modelo tradicional y del modelo AJAX de una aplicación web.

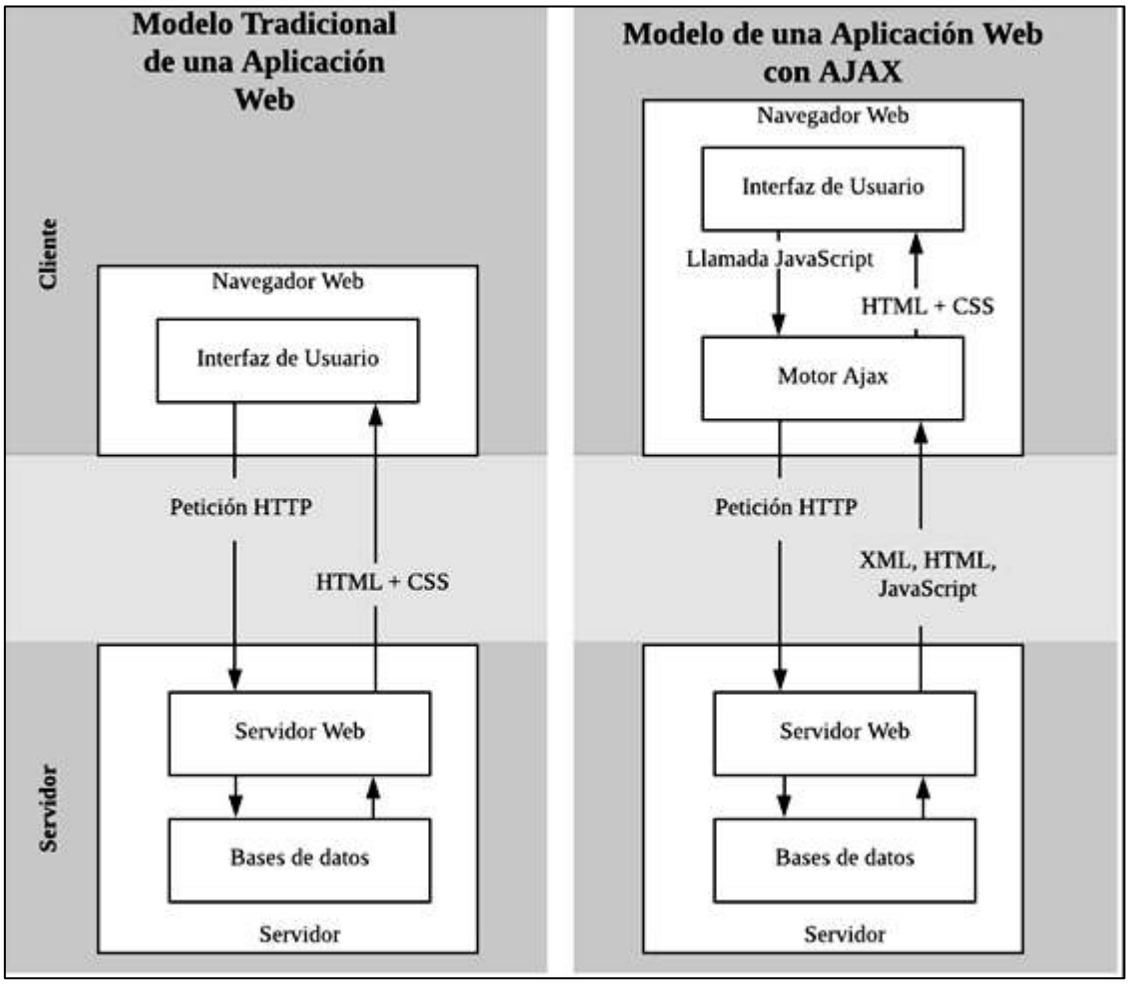


Figura 1.6 Estructura del modelo tradicional y del modelo AJAX de una aplicación web [6]

1.3.2.6 jQuery

Es una librería de JavaScript libre y de código abierto que facilita la manipulación de documentos HTML, el manejo de eventos, desarrollar animaciones y proporcionar interactividad a páginas web mediante AJAX. jQuery posee un API[12] compatible con la mayoría de los navegadores web [11].

El *framework*[13] de jQuery se lo puede descargar desde la página web: `https://jquery.com/download/` y se lo debe incluir dentro del directorio de la

[12] API (*Application Programming Interface*): Código que permite que dos programas se comuniquen.
[13] *Framework:* Conjunto de archivos que proveen una funcionalidad específica, engloba librerías, API, etc.

aplicación. En el Código 1.6 se observa la instrucción que se añade dentro del código de la página HTML para usar jQuery, esta línea de código llama al documento `jquery-3.3.3.js` que contiene código JavaScript.

```
<script src="jquery-3.3.1.js" type="text/javascript"></script>
```

Código 1.6 Llamada a script jQuery

En la Tabla 1.5 se muestran algunos parámetros de una solicitud AJAX mediante jQuery [7].

Tabla 1.5 Algunos parámetros de una solicitud AJAX

Parámetro	Definición
`Asyn`	Es posible realizar solicitudes asíncronas o solicitudes no asíncronas cambiando el valor de este parámetro de `true` a `false` respectivamente
`ContentType`	Se indica el tipo de contenido de los datos que se enviarán al servidor
`Method- Type`	Establece el método HTTP usado para la solicitud (POST, GET)
`DataType`	Establece el tipo de datos que se espera recibir desde el servidor. Los tipos de datos disponibles son: XML, HTML, script, JSON[14], JSONP[15] y texto. Es posible colocar valores espaciados y separados para conversiones entre tipos de datos. Por ejemplo: Si se requiere que un tipo de dato texto sea tratado como XML se usaría "`TEXT XML`"
`Data`	Son los datos por enviar al servidor, se adjunta a la URL[16] para las peticiones GET. El objeto debe ser un par clave-valor
`Url`	Cadena de caracteres que contienen la URL a donde la petición se envía
`ProcessData`	Permite que los datos del parámetro `Data` se procesen y transformen en una cadena de consulta
`Success`	Es una función que se ejecuta si la solicitud es satisfactoria

[14] JSON (JavaScript Object Notation): Formato para el intercambio de datos estructurados.
[15] JSONP (*JSON Padding*): Técnica de JavaScript para realizar llamadas asíncronas a dominios diferentes.
[16] URL (*Uniform Resource Locator*): Secuencia para denominar y localizar recursos en una red.

En el Código 1.7 se presenta un ejemplo del uso de AJAX, para lo cual en la línea 30 se define la URL del servicio (`Chat.svc/FuncionChat`), en la línea 31 se especifica el método a usar (POST). En la línea 32 se escriben los datos que se van a enviar al servidor, estos datos se encuentra en formato JSON (especificado en la línea 33). Además se define en la línea 34 que se espera recibir del servidor datos de tipo JSON. Finalmente si la petición se realiza con éxito ejecutarán las líneas 37 a 39, donde se cambiará el contenido de los objetos `#idGrampal` y `#conversacion_respuesta` conforme la respuesta obtenida.

```
29  $.ajax({
30      url: "Chat.svc/FuncionChat",
31      type: "POST",
32      data: '{"nombre":"' + $("#conversacion_pregunta").val() + '" }',
33      contentType: "application/json",
34      dataType: "json",
35      processData: true,
36      success: function (response) {
37          var foo = response;
38          $("#conversacion_respuesta").val(foo.Respuesta);
39          $("#idGrampal").html("<h3>" + foo.Respuesta + "</h3>");
40      }
41  });
```

Código 1.7 Ejemplo de petición AJAX

1.3.2.7 JSON (JavaScript Object Notation)

Es un formato para el intercambio de datos que permite representar datos estructurados. JSON está construido sobre dos estructuras la primera es una colección de pares clave/valor llamado objeto; mientras que la segunda es una lista ordenada de valores denominado *array*[17].

Un objeto JSON se encuentra delimitado por llaves, para establecer los pares clave valor se utilizan *strings* dentro de comillas dobles y son separados entre sí por comas `{"nombre1":"valor1", "nombre2": "valor2"}`. Mientras que para representar *arrays* se utiliza corchetes [11]. La Figura 1.7 muestra un ejemplo de un objeto JSON.

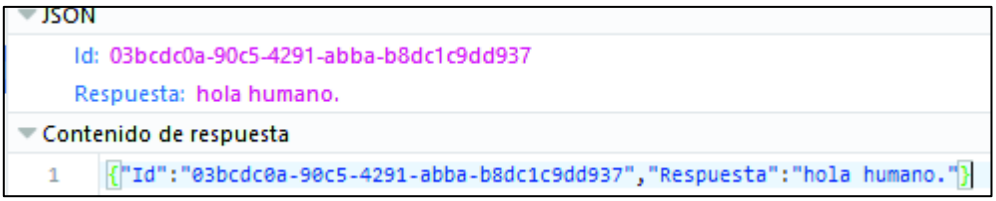


Figura 1.7 Ejemplo de objeto JSON

[17] *Array*: Conjunto de objetos del mismo tipo.

1.3.2.8 Servicios WCF (Windows Communication Foundation)

Es un *framework* para crear aplicaciones que envían mensajes entre clientes y servicios web[18]. Los clientes son aplicaciones que inician la comunicación, mientras que los servicios son aplicaciones que escuchan las peticiones, las procesan y retornan un resultado al cliente, cabe mencionar que una única aplicación puede actuar como cliente y servicio [17]. Algunas de las características de WCF son:

- Aplicaciones orientadas a servicios: servicios web débilmente acoplados para enviar y recibir datos, es decir que cualquier plataforma puede conectar con cualquier servicio siempre que se cumplan los contratos.
- Contratos de datos: presenta un completo sistema para trabajar con datos, cuando se crean las clases el servicio WCF genera de forma automática los metadatos[19] que permiten a los clientes acoplarse a los tipos de datos diseñados.
- Compatibilidad con AJAX Y REST: Se puede configurar WCF para procesar datos XML sin formato que no se acoplan a SOAP[20] o para admitir formatos no XML como JSON.
- Patrones de mensajes: Se pueden intercambiar mensajes con varios patrones como solicitud/respuesta, unidireccionales e intercambio dúplex.

La arquitectura de WCF está compuesta de cuatro capas principales, tal como se puede observar en la Figura 1.8. Estas capas son las siguientes: de contratos, de tiempo de ejecución de servicio, de mensajería y la capa de alojamiento y activación; las mismas que se detallan a continuación [17]:

Contratos: Esta capa define el sistema de mensajes. En la Figura 1.8 se puede apreciar que dentro de esta capa existen cuatro módulos como: el contrato de datos que describe cada parámetro de los mensajes que un servicio puede crear o utilizar; el contrato de servicios que define las firmas de los métodos del servicio; las directivas y enlaces que indican los requisitos para comunicarse con el servicio como protocolo de transporte, codificación y seguridad.

Tiempo de ejecución de servicio: Esta capa contiene los comportamientos que se producen solo durante la ejecución del servicio. Está compuesta de nueve módulos, entre los principales se tiene: la limitación de peticiones que establece el número de mensajes

[18] Servicio Web: Clase que permite que sus métodos sean llamados a través de la red.
[19] Metadatos: Datos que describen a otros datos.
[20] SOAP: Protocolo que emplea XML para realizar llamadas a procedimientos remotos.

que se procesan; el comportamiento de error que indica la acción que se debe tomar al producirse un error interno en el servicio; y el comportamiento de la instancia que especifica cuántas instancias del servicio se pueden ejecutar.

Mensajería: Esta capa muestra los posibles formatos y patrones de intercambio de los datos. Está compuesta de canales que procesan el mensaje, estos canales son de dos tipos: de transporte y de protocolo. Los canales de transporte leen y escriben mensajes desde la red, como por ejemplo: canal HTTP o canal TCP que especifican que se utilice el protocolo correspondiente para la entrega de mensajes. Los canales de protocolo leen y escriben encabezados adicionales en los mensajes, como por ejemplo: `WS-Security` que habilita la seguridad en la capa mensajería o `WS-Realiable` que habilita la garantía de entrega del mensaje.

Alojamiento y activación: Los servicios pueden ejecutarse de diversas formas como, por ejemplo: mediante un ejecutable (servicio auto hospedado) o pueden ejecutarse en un ejecutable administrado por un agente externo como IIS[21] (servicio hospedado).

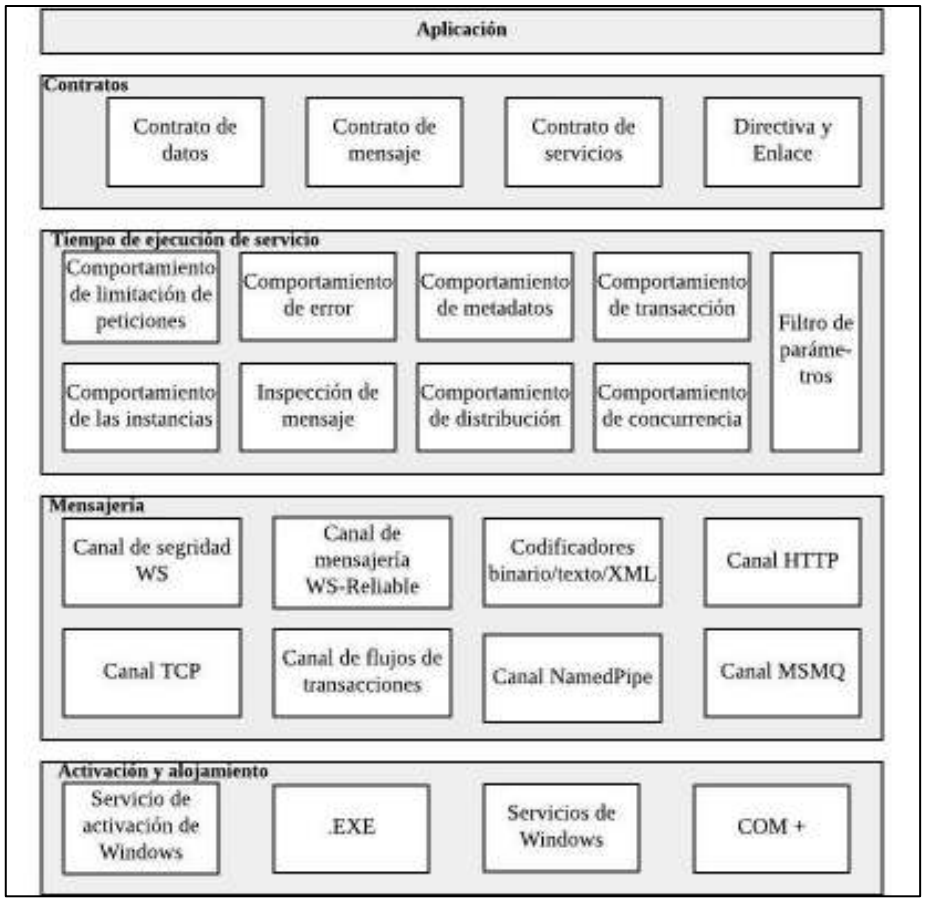


Figura 1.8 Arquitectura de WCF [17]

[21] IIS (*Internet Information Services*): Servidor web para el sistema operativo Windows.

Toda la comunicación con un servicio WCF se produce a través de puntos de conexión, estos proporcionan acceso a la funcionalidad de un servicio. Esto quiere decir que los mensajes se envían entre puntos de conexión, los mismos que definen la información necesaria para el intercambio de mensajes. Un servicio web expone uno o más puntos de conexión y el cliente genera un punto de conexión compatible con el servicio que desea contactar [17].

Cada punto de conexión está compuesto de [17]:

- Dirección: Indica la ubicación del punto de conexión del servicio web, está compuesto de una propiedad `URI` que representa la dirección del servicio (formado por el mecanismo de transporte (*binding*), el nombre del equipo, el puerto y la ubicación del servicio), una propiedad `Identity` que representa la identidad de seguridad del servicio y encabezados opcionales que brindan información de direccionamiento adicional para identificar e interactuar con el punto de conexión.
- Asociación (*binding*): indica la forma de comunicarse con un punto de conexión. Incluye la codificación (texto, binaria) y el mecanismo de transporte (HTTP, TCP) utilizado en la comunicación entre el cliente y el servicio web. También puede indicar otros parámetros como mecanismos de seguridad.
- Contrato de servicios: es una interfaz que especifica la firma de los métodos del servicio, los tipos de datos aceptados y de retorno, la ubicación de los métodos, los protocolos y formatos de serialización admitidos [18].
- Comportamientos: especifican los detalles de implementación local del punto de conexión y se los puede usar para personalizar el comportamiento del punto de conexión.

Estructura de una dirección: El URI de la mayoría de protocolos de transporte tiene cuatro partes. El esquema, el equipo, el puerto (opcional) y la ruta de acceso. La Figura 1.9 se presenta una dirección de un servicio web compuesta de:

- Esquema: `HTTP`
- Equipo: `localhost`
- Puerto: `55954`
- Ruta de acceso*:* `Chat.svc/FuncionChat`

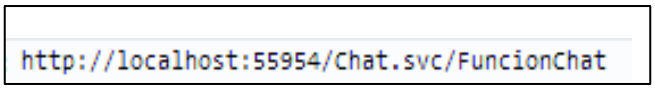


Figura 1.9 Dirección de un servicio web

Un servicio WCF se puede configurar a través de la tecnología de configuración de .NET Framework[22]. Si el servicio WCF se hospeda en un servidor IIS el archivo de configuración se llama `Web.config`, mientras que si el servicio WCF se hospeda en un entorno diferente el archivo de configuración se llamará `App.config`.

Este archivo de configuración tiene formato XML y sus principales etiquetas son las siguientes [19]:

`<service>`: Contiene las especificaciones para todos los servicios que la aplicación hospeda, posee dos atributos. El atributo `Name` indica el nombre junto al espacio de nombres donde se encuentra el servicio, en la línea 12 de la Figura 1.10 se puede observar que el espacio de nombres es: `WCFService` y el nombre del servicio es: `Chat`. El atributo `behaviorConfiguration` establece el nombre de los comportamientos.

`<endpoint>`: Cada punto de conexión posee cuatro atributos: `address` que especifica el URI del servicio; `binding` que especifica un tipo de enlace definido por el usuario o por el sistema (se define transporte, seguridad y codificación); el atributo `bindingConfiguration` que especifica el nombre del enlace y el atributo `contract` que especifican la interfaz donde se define el contrato. En la línea 13 de la Figura 1.10 se observa esta etiqueta junto a sus atributos.

`<behavior>`: Indica el comportamiento de un servicio, por lo que se asocia con un punto de conexión y es posible que algunos servicios compartan el mismo comportamiento. Posee un atributo `name` que identifica cada comportamiento. Además, puede tener etiquetas secundarias como: `<serviceMetadata>` que especifica la publicación de metadatos e información asociada al servicio; `<serviceDebug>` que especifica la forma de depuración y ayuda para el servicio WCF.

En la línea 25 de la Figura 1.10 se observa que la etiqueta `<serviceMetadata>` posee los atributos `httpGetEnabled` y `httpsGetEnabled` con valor `true` lo que indica que se van a publicar los metadatos del servicio en el URI del servicio seguido de "`?wsdl`". En la línea 26 se aprecia que la etiqueta `includeExceptionDetailInFaults` tiene

[22] .NET Framework: Es un entorno de trabajo de Microsoft que permite el desarrollo de aplicaciones.

un valor de `false` lo que significa que no se va a permitir el flujo de información de excepciones o la publicación de archivos HTML al momento de examinar el servicio en un explorador web ya que exponer la implementación de un servicio es un riesgo de seguridad.

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <appSettings>
    <add key="aspnet:UseTaskFriendlySynchronizationContext" value="true"/>
  </appSettings>
  <system.web>
    <compilation debug="true" targetFramework="4.6.1"/>
    <httpRuntime targetFramework="4.6.1"/>
  </system.web>
  <system.serviceModel>
    <services>
      <service name="WCFService.Chat">
        <endpoint address="" behaviorConfiguration="WCFService.ChatAspNetAjaxBehavior
      </service>
    </services>
    <behaviors>
      <endpointBehaviors>
        <behavior name="WCFService.ChatAspNetAjaxBehavior">
          <enableWebScript/>
         <webHttp defaultOutgoingResponseFormat="Json" defaultBodyStyle="WrappedReque
        </behavior>
      </endpointBehaviors>
      <serviceBehaviors>
        <behavior name="">
          <serviceMetadata httpGetEnabled="true" httpsGetEnabled="true"/>
          <serviceDebug includeExceptionDetailInFaults="false"/>
        </behavior>
      </serviceBehaviors>
    </behaviors>
```

Figura 1.10 Archivo `Web.config`

Compatibilidad de los servicios WCF: La compatibilidad de WCF, AJAX y JSON permite que los clientes (páginas web habilitadas para JavaScript) puedan acceder a los servicios WCF mediante solicitudes HTTP. Existe tres pasos para crear un servicio WCF AJAX:

- Crear un punto final AJAX que pueda ser accedido desde el navegador.
- Crear un contrato de servicio compatible con AJAX.
- Acceder a los servicios WCF AJAX.

Un contrato de servicio está adornado de:

- El atributo `ServiceContract` que define que una clase o interfaz sea un contrato de servicio.

- El atributo `OperationContract` especifica los métodos que se van a exponer como operaciones del servicio web.

Un contrato de datos está adornado de:

- El atributo `DataContract` especifica que se implementa un contrato de datos serializable
- El atributo `DataMember` define los campos o valores que se serializarán.

En el Código 1.8 se tiene la implementación de un método que se desea exponer en el servicio web, el mismo que recibe un *string*, lo procesa y retorna un objeto de la clase `Chatear`. Para lo cual se implementa un contrato de servicios con los adornos mencionados anteriormente. En la línea 14 se tiene el uso del atributo `ServiceContract` y en la línea 18 se expone el método `FuncionChat` por medio del atributo `OperationContract`.

```
14      [ServiceContract(Namespace = "")]
15      [AspNetCompatibilityRequirements(RequirementsMode = AspNetCompatibilityRequirementsMode.Allowed)]
        0 referencias
16      public class Chat
17      {
18          [OperationContract]
19          [WebInvoke(Method = "POST", ResponseFormat = WebMessageFormat.Json)]
            0 referencias
20          public Chatear FuncionChat(string nombre)
21          {
22              return new Chatear() { Id = Guid.NewGuid(), Respuesta= Respuesta.Responder(nombre) };
23          }
```

Código 1.8 Contrato de Servicio

En el Código 1.9 se observa un contrato de datos, y la línea 40 presenta el atributo `DataContract` que indica que se tiene un contrato serializable y se observa que en la línea 43 el atributo `DataMember` especifica los campos a serializar.

```
40      [DataContract]
        2 referencias
41      public class Chatear
42      {
43          [DataMember]
            1 referencia
44          public Guid Id { get; set; }
45          [DataMember]
            1 referencia
46          public string Respuesta { get; set; }
47      }
```

Código 1.9 Contrato de Datos

Cabe resaltar que por defecto los contratos de servicio compatibles con AJAX devuelven datos en formato XML, pero se puede modificar, para usar formato de datos JSON [15].

Para lo cual se debe localizar y modificar las siguientes tres etiquetas del archivo `Web.config` [21].

La etiqueta `<behavior>` describe el comportamiento del punto de conexión, su atributo `name` es una cadena única que contiene el nombre de la configuración del comportamiento. En la línea 19 de la Figura 1.11, se puede observar que en este archivo de configuración el nombre de este comportamiento es `WCFService.ChatAspNetAjaxBehavior`.

La etiqueta `<enableWebScript>` habilita el comportamiento del punto de conexión que permite utilizar AJAX en una página web. En la línea 20 de la Figura 1.11 se encuentra habilitado.

La etiqueta `<webHttp>` especifica un comportamiento en un punto de conexión a través de la configuración, dentro de esta etiqueta se puede especificar varios atributos. El atributo `DefaultOutgoingResponseFormat` establece el formato predeterminado de las respuestas de los mensajes. Existen dos métodos para determinar el formato correcto: el formato automático que elige con base en el contenido y cabecera del mensaje de solicitud y el formato explícito, donde el desarrollador establece un formato fijo como JSON. El atributo `defaultBodyStyle` especifica el estilo del cuerpo de los mensajes recibidos, donde `WrappedRequest` indica que se ajustan las solicitudes pero no las respuestas. Lo indicado aparece en la línea 21 de la Figura 1.11.

```
18      <endpointBehaviors>
19        <behavior name="WCFService.ChatAspNetAjaxBehavior">
20          <enableWebScript/>
21         <webHttp defaultOutgoingResponseFormat="Json" defaultBodyStyle="WrappedRequest"/>
22        </behavior>
23      </endpointBehaviors>
24      <serviceBehaviors>
```

Figura 1.11 Archivo `Web.config` con AJAX

1.3.3 HERRAMIENTAS

1.3.3.1 Grampal

Es un analizador morfosintáctico del español con una base de datos léxica de 50.000 entradas, que asigna una etiqueta a cada palabra analizada [6]. La etiqueta posee la siguiente información [5]:

- Categoría sintáctica: indica si es verbo, preposición, sustantivo, etc.

- Lema: término que representa todas las variaciones morfológicas de la palabra. Por ejemplo: el lema de la palabra cocinando es: cocinar.
- Rasgos morfosintácticos: género, número, persona y tiempo.

En la Figura 1.12 se muestra un ejemplo de uso de Grampal. Al ingresar la oración: "Espero que tengas un bien día" se obtuvo el análisis de cada palabra, por ejemplo se observa que la palabra "tengas" es de categoría V (verbo), su lema es tener y se encuentra en singular presente subjuntivo. Estos datos resultan útiles para diferentes fines prácticos como obtener el lema de los verbos de cualquier oración, entonces sin importar en que tiempo o modo se encuentre la oración se podrá conocer los verbos en infinitivo.

Figura 1.12 Ejemplo de uso de Grampal

1.3.3.2 Selenium WebDriver

Es una API de código abierto que permite realizar pruebas de automatización web a páginas dinámicas o estáticas, admite múltiples navegadores web y sistemas operativos. Además para codificar estas pruebas automatizadas permite utilizar múltiples lenguajes de programación como Java, C#, Python, etc. Para su uso se selecciona un controlador web según el explorador web en el que se desee realizar las pruebas.

Los controladores se comunicarán con la página web solicitada e interactúan con los elementos (`WebElement`) de la misma: cuadros de texto, botones, enlaces, etc. Para interactuar con un elemento de la página web en primer lugar se lo debe encontrar, una vez localizado se procede a realizar la acción [20]. En la Tabla 1.6 se muestran algunos

comandos de la librería Selenium WebDriver usando el lenguaje de programación C#[23] [21].

Tabla 1.6 Algunos comandos de Selenium WebDriver

Comando	Tipo	Descripción
`click`	`WebElement`	Permite hacer clic en un elemento que se encuentre visible en una página web como botones, casillas de verificación, botones de radio.
`SendKeys`	`WebElement`	Permite introducir un parámetro en los cuadros de texto.
`TagName`	`WebElement`	Retorna el documento HTML de un elemento web como un *string*.
`GoToUrl`	`Browser`	Carga la página web deseada en el explorador web.
`FindElement`	`WebElement`	Permite encontrar el elemento en la página web.

1.3.3.3 SQL Server

Permite gestionar bases de datos relacionales. En la Tabla 1.7 se presentan algunas instrucciones básicas de manipulación de datos SQL [28].

Tabla 1.7 Algunas instrucciones SQL

Sentencia	Tareas
`Insert`	Inserta filas en una tabla
`Select`	Selecciona los datos almacenados en una base de datos
`Delete`	Borra las filas de una tabla

La sintaxis básica de una consulta `SELECT` está presente en la Figura 1.13. Se puede apreciar que `SELECT` permite seleccionar las columnas que se van a mostrar, puede ser todas las columnas o específicas, además `SELECT` está acompañado de `ALL` que indica que se obtendrá todas las filas incluyendo las duplicadas o de `DISTINC` que retornará solo filas únicas. A continuación, mediante `FROM` se especifica el nombre de las tablas o vistas de las que se obtendrá la información. `WHERE` especifica condiciones para que se retorne solo las filas cumplan ciertas expresiones lógicas como mayor, menor, igual, entre

[23] C#: Lenguaje de programación orientado a objetos.

otros. Finalmente se puede presentar los resultados de la consulta ordenados mediante una columna de forma ascendente o descendente [28].

```
SELECT [ALL /DISTINC] [*] / [ListaColumnas]

FROM Nombre_de_la_Tabla

WHERE Condiciones
```

Figura 1.13 Estructura sentencia `SELECT`

1.3.4 KANBAN

Kanban es una metodología ágil[24] que permite visualizar y gestionar el flujo de trabajo. La palabra Kanban se deriva de dos palabras japonesas *kan* y *ban* cuyo significado es "tarjetas visuales". Kanban se considera el enfoque de Lean al desarrollo de software.

Kanban implementa completamente los principios de un sistema Lean:

- Alta calidad: Busca eliminar los defectos, detectar y corregir problemas desde el origen. Es mejor hacerlo bien que hacerlo rápido.
- Reducción del desperdicio: Busca eliminar las actividades que no presentan valor y optimizar el uso de recursos.
- Mejora continua: Busca reducir los costos, aumentar calidad y productividad.
- Flexibilidad: Producción rápida de variedad de productos, priorizar y condicionarlas tareas de la cola de tareas.
- Productos *pull*: Los productos deben ser solicitados por el usuario final.

"Lean es básicamente todo lo concerniente a obtener las cosas correctas en el lugar correcto, en el momento correcto, en la cantidad correcta, minimizando el desperdicio, siendo flexible y estando abierto al cambio" [27].

Kanban se basa en el desarrollo incremental, dividiendo el trabajo en partes y permite visualizar el flujo de trabajo y para esto divide el trabajo en bloques (ítems) y lo representa usando tarjetas y tableros. Además, establece un tiempo de plazo que es el tiempo promedio para completar un ítem, se procura que este valor sea pequeño. Kanban indica que todo el desarrollo debe optimizarse y presenta como objetivos:

[24] Metodología ágil: Proceso adaptable que permite dar respuestas rápidas, cooperativas.

- Balancear la demanda con la capacidad.
- Limitar el trabajo en proceso, detectar rápidamente los problemas.
- Coordinar y sincronizar el trabajo, descubrir cuellos de botella.
- Equipos autoorganizados.

1.3.4.1 Tarjetas Kanban

Cada ítem de trabajo se representa con una tarjeta en el tablero Kanban. Cada tarjeta posee información sobre la persona responsable, la descripción del trabajo y el tiempo estimado de realización del ítem. En cualquier punto de tiempo, la tarjeta representa el estado actual del trabajo [30].

1.3.4.2 Tablero Kanban

Sirve para visualizar, optimizar y estandarizar el flujo de trabajo; ayuda a identificar cuellos de botella, bloqueos, dependencias para su pronta resolución. Un tablero Kanban básico posee tres estados: por hacer, en progreso y listo; se pueden añadir nuevos estados dependiendo de las necesidades y requerimientos propios de un proyecto. Cuando un ítem se completa, se mueve la tarjeta que lo representa al siguiente estado del flujo de trabajo. Es decir, el objetivo de un tablero Kanban es tener siempre presente el trabajo que se debe realizar; así nadie se quedará sin tareas por hacer y las tareas más importantes se realizarán primero [31].

Como se observa en la Figura 1.14 se tiene un tablero Kanban con los tres estados básicos. Además están dos tarjetas en el estado por hacer y una en el estado en progreso. Cada tarjeta representando un ítem de trabajo con su descripción, encargado y tiempo estimado.

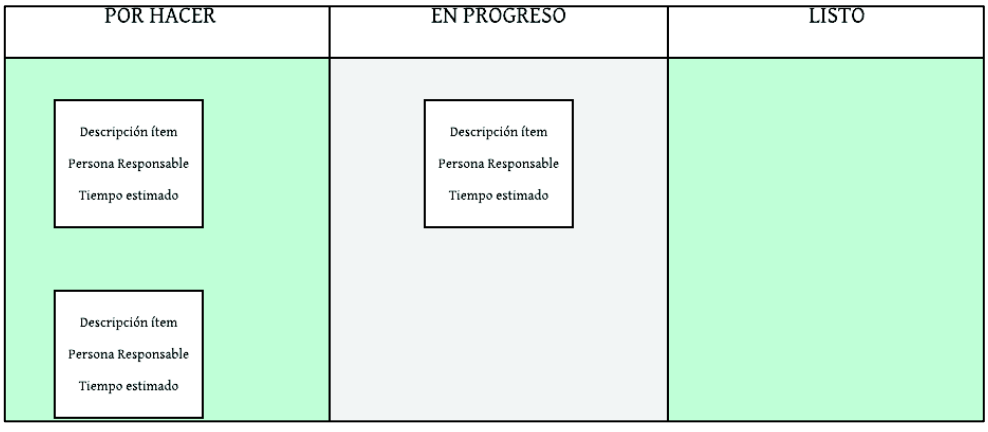


Figura 1.14 Ejemplo tablero y tarjetas Kanban

2. METODOLOGÍA

El presente Trabajo de Titulación propone desarrollar un prototipo de chatbot mediante el uso de la metodología ágil Kanban. Esta metodología plantea dividir el trabajo en distintos bloques que serán llamados ítems y se representará cada ítem en una tarjeta Kanban que se colocará en el tablero Kanban, con el fin de visualizar el flujo de trabajo en todo momento.

En este capítulo se planteará tres etapas: análisis, diseño e implementación. Cada una de estas etapas estará formada por un conjunto de ítems (tareas a realizar). Además, se usará un tablero Kanban para cada etapa compuesto por tres estados: *por hacer*, *en progreso* y *listo*. Es importante mencionar que se realizaron pruebas al chatbot que serán detalladas en el capítulo 3.

La primera etapa es el análisis, esta etapa contará con cuatro ítems en la columna *por hacer* del tablero Kanban. Se iniciará con el levantamiento de información por medio de entrevistas presenciales a veinte personas entre personal administrativo y estudiantes; a partir de la información obtenida en estas entrevistas se generará historias de usuario con el fin de especificar los requerimientos funcionales y no funcionales de la aplicación. Para finalmente definir las preguntas frecuentes que el chatbot será capaz de responder.

La segunda etapa es el diseño, esta etapa contará con cinco ítems en la columna *por hacer* del tablero Kanban. Se diagramará la arquitectura del chatbot con sus respectivas clases e interfaces para brindar la funcionalidad requerida. Además, se generará el diagrama de casos de uso, el diagrama de actividades, el diagrama de clases y el diagrama relacional de base de datos. Luego se bosquejará el *sketch* de la interfaz del chatbot.

La tercera etapa es la Implementación, esta etapa contará con siete ítems en la columna *por hacer* del tablero Kanban. En primer lugar, se codificarán las clases conforme los diagramas planteados en la etapa de diseño, estas clases proveerán la funcionalidad del chatbot. A continuación, se creará la base de datos, que se poblará con las preguntas y respuestas previamente definidas y se la conectará con la aplicación. También se crearán los servicios web WCF y se generará una página web HTML de prueba, donde se incluirá al chatbot.

En el planteamiento de este Trabajo de Titulación no se consideró la implementación de un mecanismo de administración; sin embargo si se codificará un mecanismo que permita

gestionar las respuestas del chatbot, es importante que el mismo brinde información actualizada conforme al periodo académico vigente.

2.1 ANÁLISIS

En la etapa de análisis se establecen cuatro ítems de trabajo. Se han planificado tareas para realizar el levantamiento de información y poder definir los requerimientos funcionales y no funcionales de la aplicación. Después de este proceso se establecerán los temas específicos sobre los que el chatbot brindará información.

En la Figura 2.1 se muestra el tablero Kanban de esta etapa, cada ítem de trabajo planteado se encuentra representado como una tarjeta Kanban en la columna *por hacer*. Se observa que existen cuatro tarjetas *por hacer* y que las columnas *en progreso* y *listo* se encuentran vacías debido a que ninguna tarea ha iniciado.

Figura 2.1 Tablero Kanban para la Etapa de Análisis

A continuación, se procede con el desarrollo de los ítems de trabajo planteados.

2.1.1 ENTREVISTAS

Se diseñó una entrevista formada por diecisiete preguntas. El modelo de la entrevista se encuentra en el ANEXO A.

Los resultados de las entrevistas son los siguientes: La primera pregunta menciona: ¿Piensas que es importante que el chatbot conozca en donde se puede consultar el turno

de la matrícula?, mediante esta pregunta el chatbot responderá: “Tu turno para matrícula lo puedes consultar en la página web del SAEW. Ingresa a www.saew.epn.edu.ec dirígete al módulo INFORMACIÓN ESTUDIANTIL y en la pestaña INFORMACIÓN GENERAL encontrarás TURNO PARA MATRÍCULA”. En la Figura 2.2 se observa que el 100% de los entrevistados respondió que SI es importante.

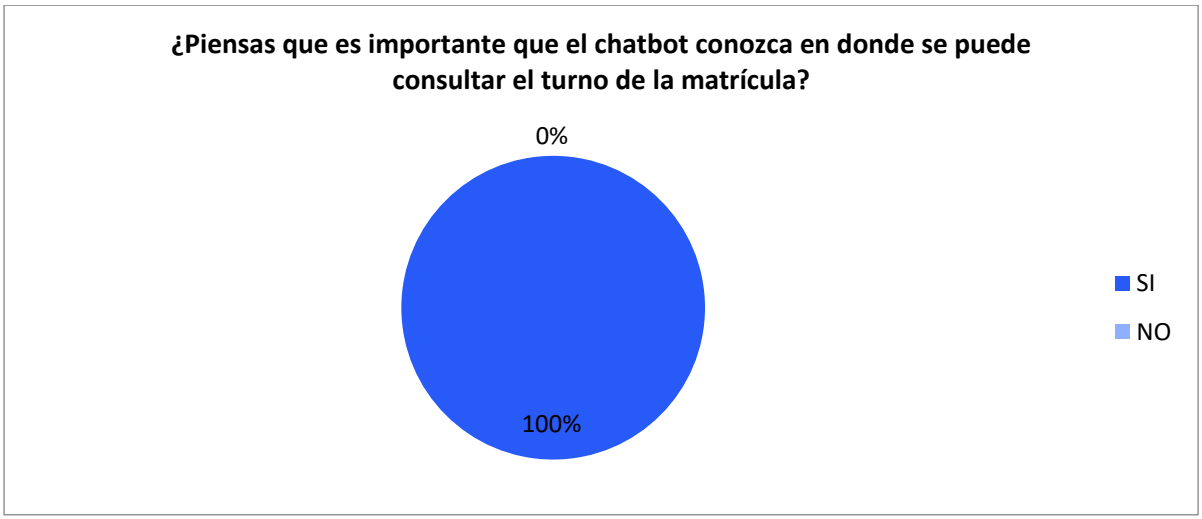


Figura 2.2 Resultados de la pregunta 1

La segunda pregunta establece ¿Crees conveniente que el chatbot conozca las fechas de matriculación?, cuya respuesta será “Las matrículas ordinarias para el semestre 2019-A se realizarán desde el 7 de marzo del 2019 hasta el 13 de marzo del 2019”. En la Figura 2.3 se observa que el 100% de los entrevistados contestó que SI es conveniente.

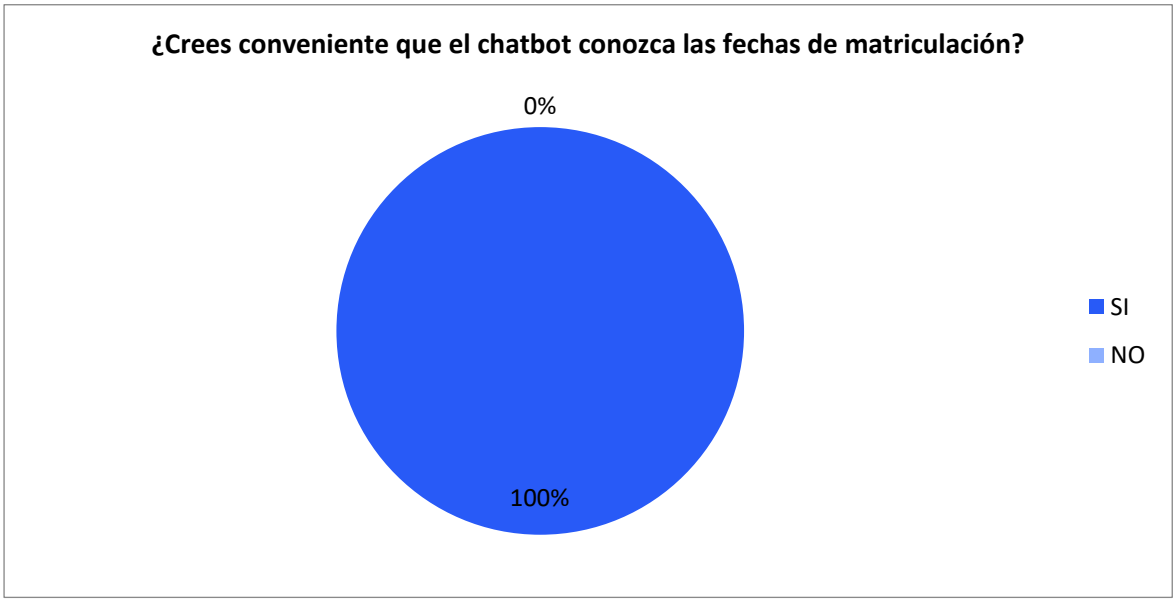


Figura 2.3 Resultados de la pregunta 2

La tercera pregunta indica ¿Consideras valioso que el chatbot conozca las fechas de reinscripciones?, cuya respuesta será: “Las reinscripciones para el semestre 2019-A se realizarán desde el 8 de abril del 2019 hasta el 9 de abril del 2019”. En la Figura 2.4 se observa que el 100% de los entrevistados respondió que SI considera valiosa esta información.

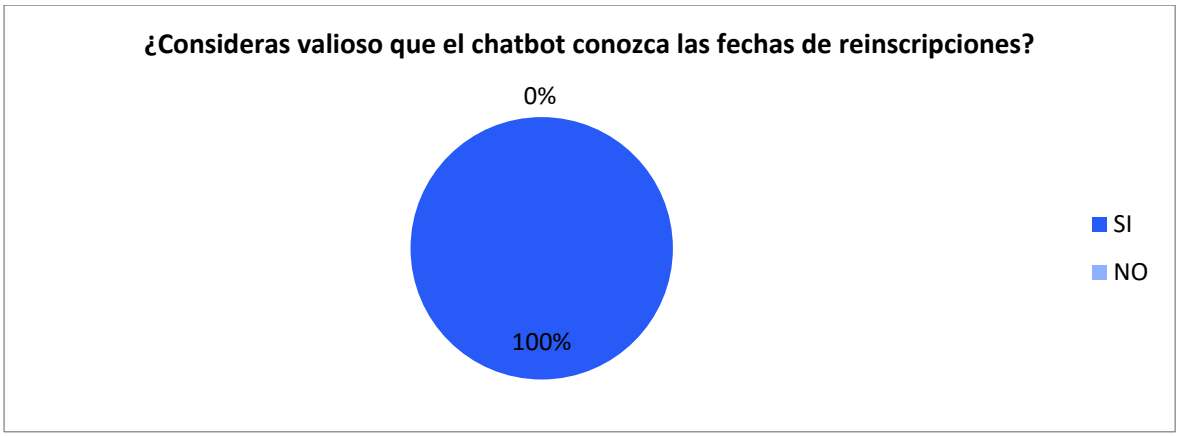


Figura 2.4 Resultados de la pregunta 3

La cuarta pregunta indica ¿Crees importante que el chatbot conozca en dónde se puede consultar el horario de materias?, por medio de esta pregunta el chatbot responderá: “Los horarios de materias lo puedes consultar en la página web del SAEW. Ingresa a www.saew.epn.edu.ec dirígete al módulo INFORMACIÓN ESTUDIANTIL y en la pestaña INFORMACIÓN GENERAL encontrarás HORARIO DE MATERIAS. Además, ¿quieres que te envíe por medio de correo electrónico el horario de materias?”. En la Figura 2.5 se observa que el 90% de los entrevistados contestó que SI es importante.

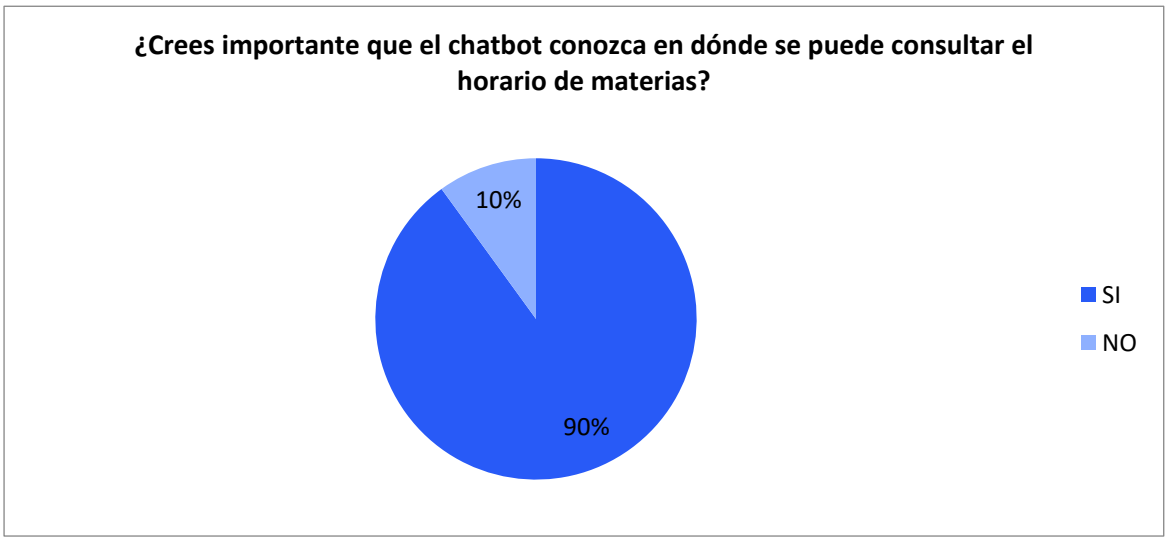


Figura 2.5 Resultados de la pregunta 4

La quinta pregunta menciona ¿Piensas que es importante que el chatbot proporcione información sobre los tipos de becas institucionales?, por medio de esta pregunta el chatbot responderá: “La institución otorga los siguientes tipos de becas: -Vulnerabilidad por situación Económica -Mérito Cultural. Para solicitar las becas los estudiantes deben utilizar el módulo académico en el SAEW”. En la Figura 2.6 se observa que el 85% de los entrevistados respondió que SI es importante.

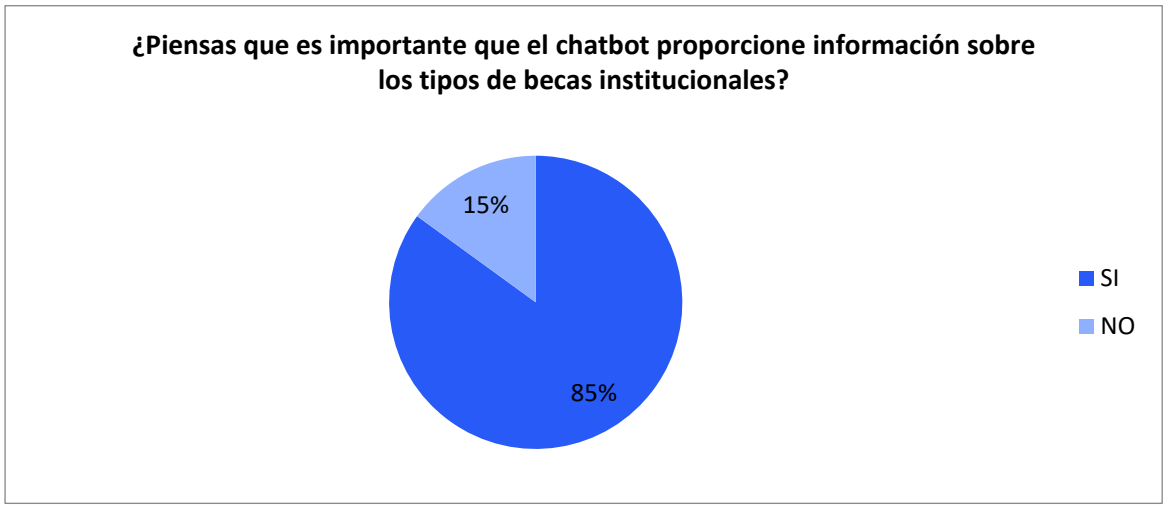


Figura 2.6 Resultados de la pregunta 5

La pregunta seis manifiesta ¿Consideras valioso que el chatbot conozca los plazos para aplicar a las becas institucionales?, por medio de esta pregunta el chatbot responderá: “Los estudiantes de la EPN pueden aplicar a las becas otorgadas por la institución (Situación Económica y Mérito Cultural) desde el 6 de marzo del 2019 hasta el 10 de marzo del 2019”. En la Figura 2.7 se observa que el 85% de los entrevistados indicó que SI es valioso.

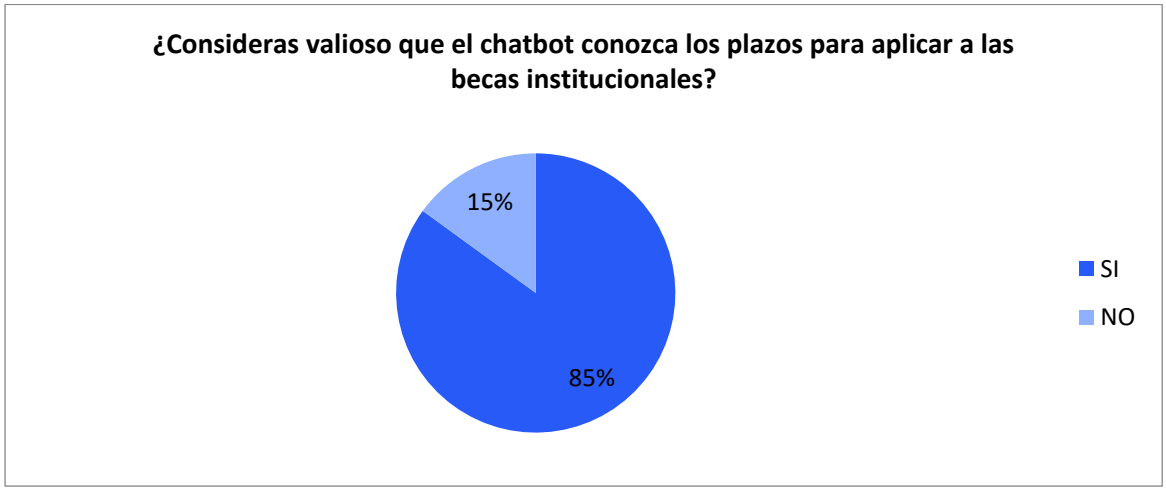


Figura 2.7 Resultados de la pregunta 6

La séptima pregunta indica ¿Consideras valioso que el chatbot conozca cuáles son las opciones de titulación?, se responderá: “Las opciones de titulación son: desarrollar un trabajo de titulación o la aprobación de un examen complexivo. Se consideran trabajos de titulación: Artículo Académico, Proyecto de Investigación, Proyecto Integrador”. En la Figura 2.8 se observa que el 90% de los entrevistados respondió que SI es valioso.

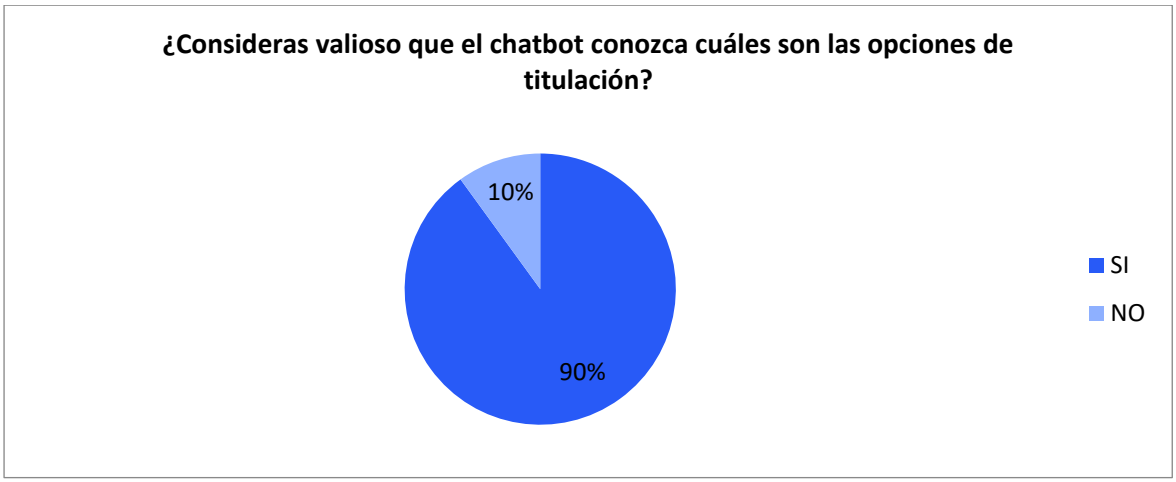


Figura 2.8 Resultados de la pregunta 7

La pregunta ocho menciona ¿Consideras valioso que el chatbot conozca sobre los plazos para cambiarse de opción de titulación en Unidad de Titulación?, se responderá: “Puedes cambiarte de opción de titulación por una sola vez hasta el período de culminación de tu plan de estudios. Este cambio lo puedes hacer en matrículas ordinarias o extraordinarias”. En la Figura 2.9 se observa que el 90% de los entrevistados respondió que SI es valioso.

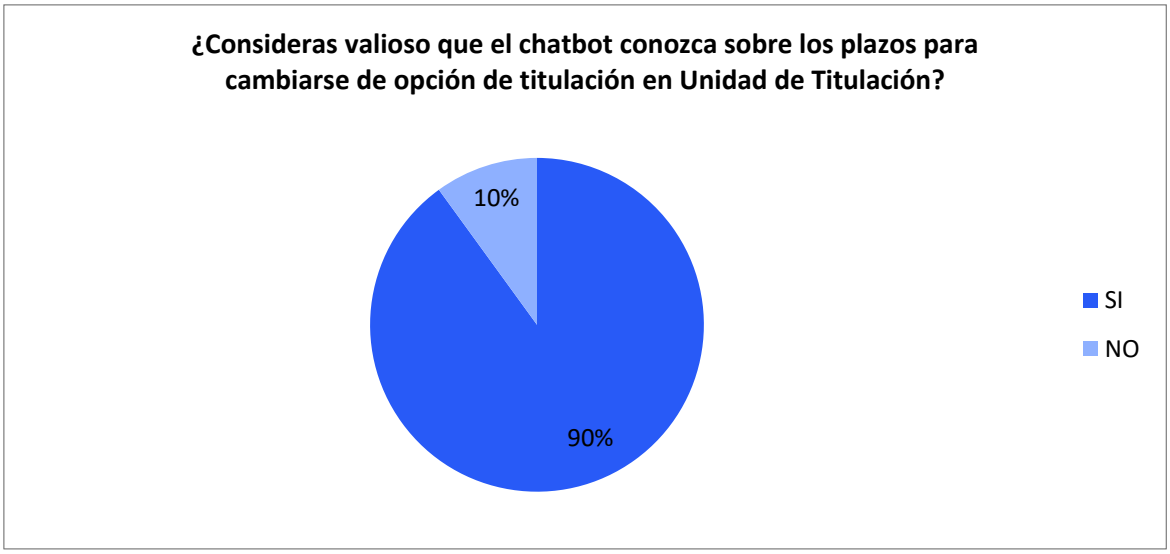


Figura 2.9 Resultados de la pregunta 8

La pregunta nueve manifiesta ¿Crees conveniente que el chatbot proporcione una descripción de la unidad de titulación: Examen Complexivo?, cuya respuesta será: “Es un examen teórico práctico que consta de 100 preguntas que tiene una duración de 8 horas. Si optas por este examen puedes rendirlo hasta por dos ocasiones en los períodos académicos consecutivos al de culminación del plan de estudios”. En la Figura 2.10 se observa que el 80% de los entrevistados indicó que SI es conveniente.

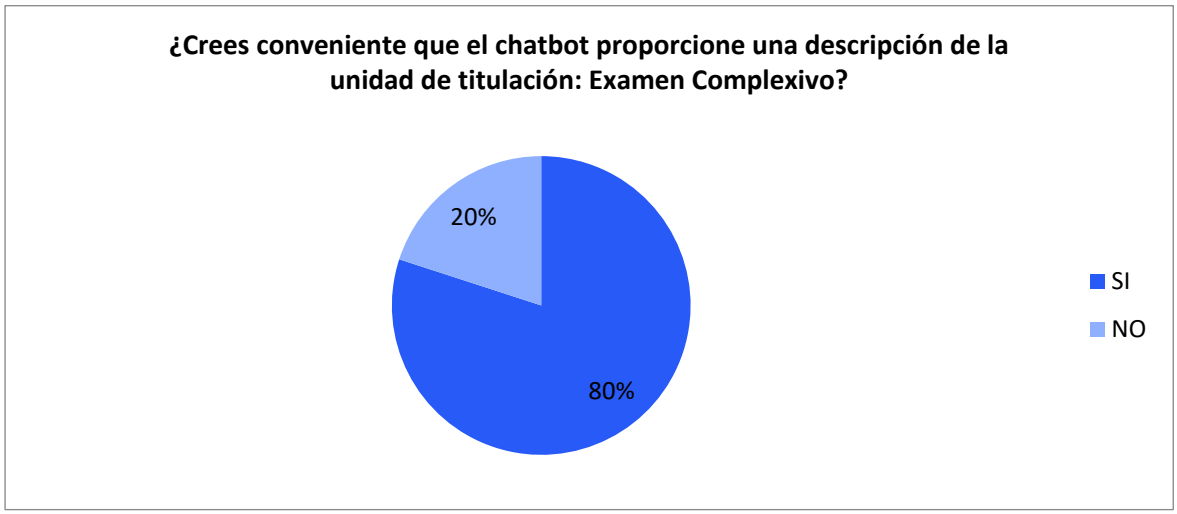


Figura 2.10 Resultados de la pregunta 9

La pregunta diez establece ¿Crees importante que el chatbot indique los plazos para el Examen Complexivo?, se responderá: “Se aplicará el examen complexivo en el segundo bimestre de cada periodo académico, una vez que los estudiantes hayan finalizado el proceso de preparación. En el semestre 2019-A el examen complexivo se aplicará del 8 al 20 de Julio del 2019”. En la Figura 2.11 se observa que el 85% de los entrevistados contestó que SI es importante.

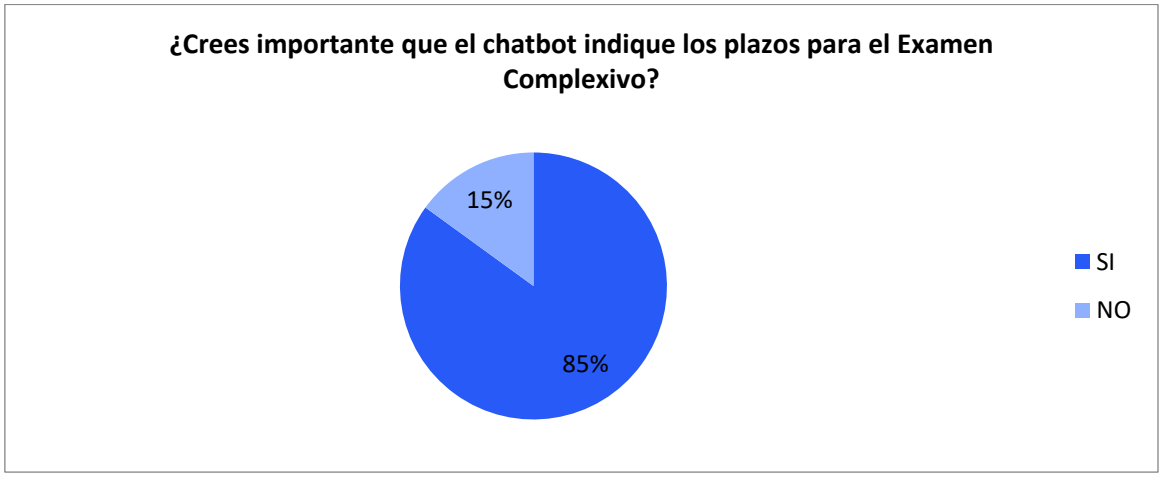


Figura 2.11 Resultados de la pregunta 10

La pregunta once menciona ¿Crees importante que el chatbot provea una descripción de la unidad de titulación: Proyectos Técnicos?, se responderá a esta pregunta mencionando “Un Proyecto Técnico es un Trabajo de Titulación que deberá tener un componente de investigación básica o aplicada. Deberá contener como mínimo, la determinación del tema o problema, el marco teórico referencial, la metodología, la interpretación de los resultados y las conclusiones”. En la Figura 2.12 se observa que el 75% de los entrevistados respondió que SI es importante.

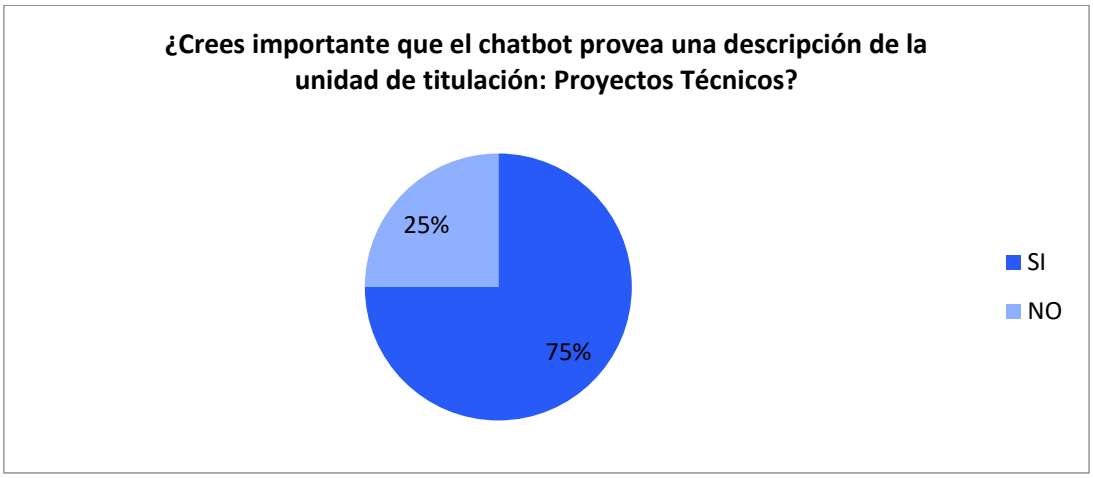


Figura 2.12 Resultados de la pregunta 11

La pregunta doce indica ¿Consideras conveniente que el chatbot brinde una descripción de la unidad de titulación: Proyecto de Investigación?, se responderá “Un proyecto de investigación es un procedimiento científico destinado a recabar información de un fenómeno para generar conocimiento o dar solución a un fenómeno”. En la Figura **2.13** se observa que el 75% de los entrevistados indicó que SI es conveniente.

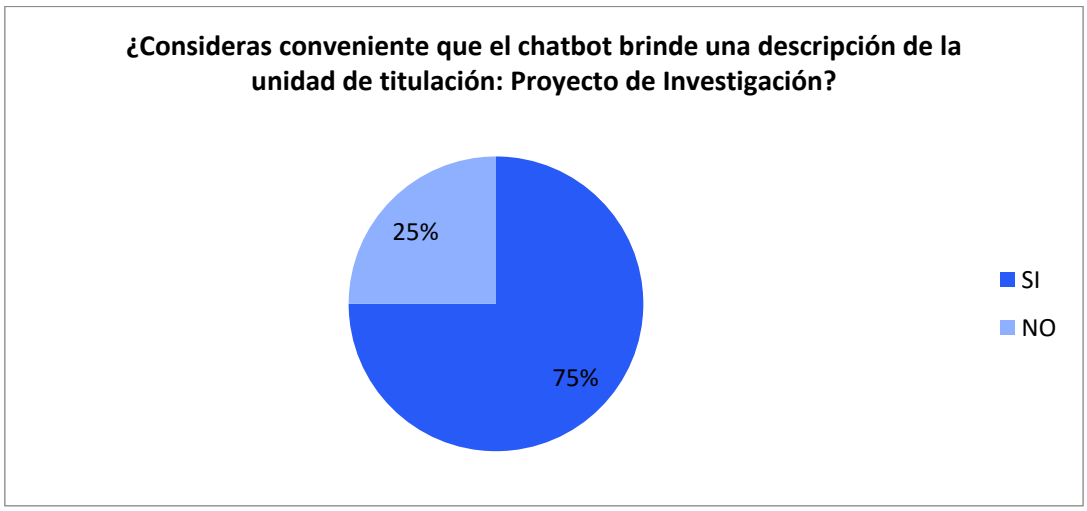


Figura 2.13 Resultados de la pregunta 12

La pregunta trece expresa ¿Crees valioso que el chatbot proporcione información sobre los requisitos para presentar anillados de Trabajo de Titulación?, el chatbot responderá: “Requieres: Matrícula vigente en Unidad de Titulación, tener plan aprobado y vigente registrado en el SAEW, certificado de aprobación de plan de estudios otorgado por la secretaría de la carrera, certificado de finalización del Trabajo de Titulación y certificado de plagio otorgados por el tutor del Trabajo de Titulación”. En la Figura 2.14 se observa que el 100% de los entrevistados contestó que SI es valioso.

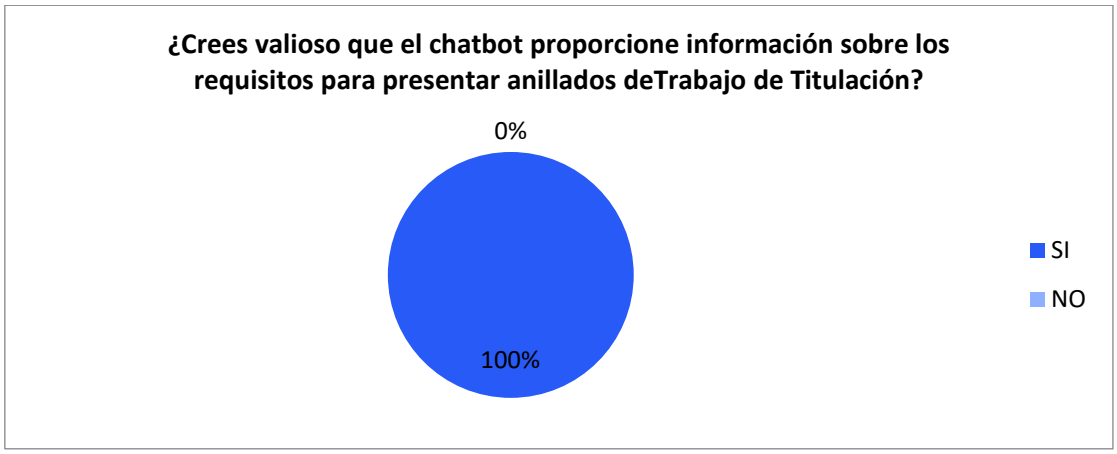


Figura 2.14 Resultados de la pregunta 13

La pregunta catorce manifiesta ¿Crees valioso que el chatbot conozca cómo ser declarado apto para la defensa de grado oral de un Trabajo de Titulación?, se responderá “Para ser declarado apto para la defensa de grado oral, el estudiante debe verificar y actualizar datos en el SAEW, completar la documentación, tener el plan de Trabajo de Titulación aprobado, inglés aprobado, haber completado las prácticas preprofesionales, y tener calificado los anillados de Trabajo de Titulación”. En la Figura 2.15 se observa que el 95% de los entrevistados contestó que SI es valioso.

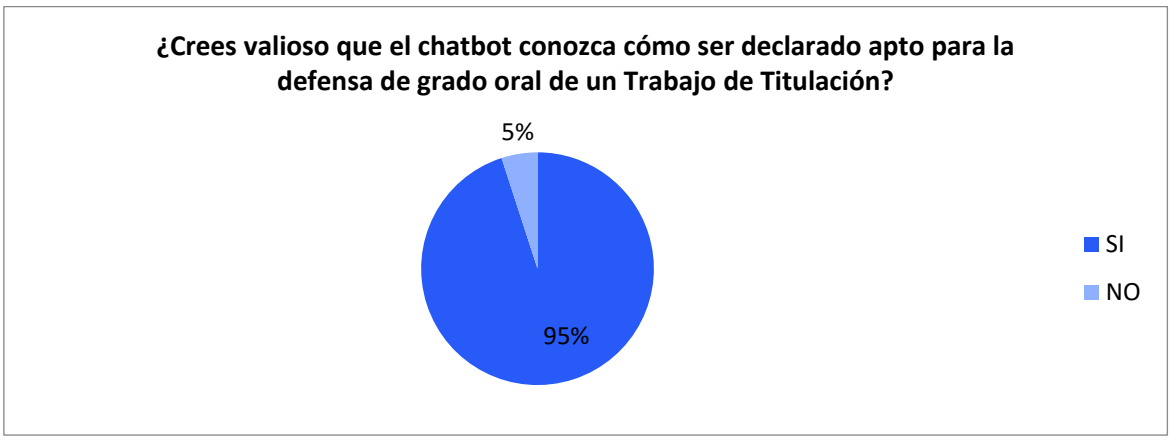


Figura 2.15 Resultados de la pregunta 14

La pregunta quince establece ¿Piensas que es importante que el chatbot informe sobre los plazos de designación del Tribunal de Trabajos de Titulación?, se responderá “Una vez entregado los ejemplares del trabajo de titulación, el decano en siete días calendario entregará a los miembros del tribunal el documento escrito, para que en un plazo de quince días calendario cada miembro presente un informe con sugerencias y la calificación sobre diez puntos”. En la Figura 2.16 se observa que el 90% de los entrevistados respondió que SI es importante.

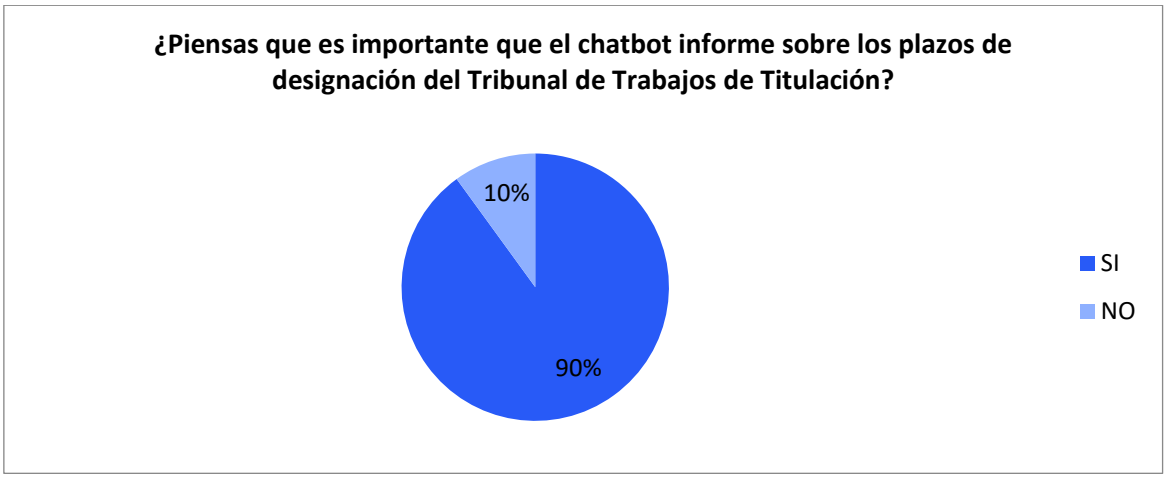


Figura 2.16 Resultados de la pregunta 15

La pregunta dieciséis indica ¿Consideras importante que el chatbot conozca sobre la conformación del Tribunal de Trabajos de Titulación?, se responderá a esta pregunta “Concluido el trabajo de titulación, el estudiante debe solicitar por escrito al decano la designación del Tribunal, el que se conformará por el director del trabajo de titulación y dos miembros del personal académico”. En la Figura 2.17 se observa que el 85% de los entrevistados contestó que SI es importante.

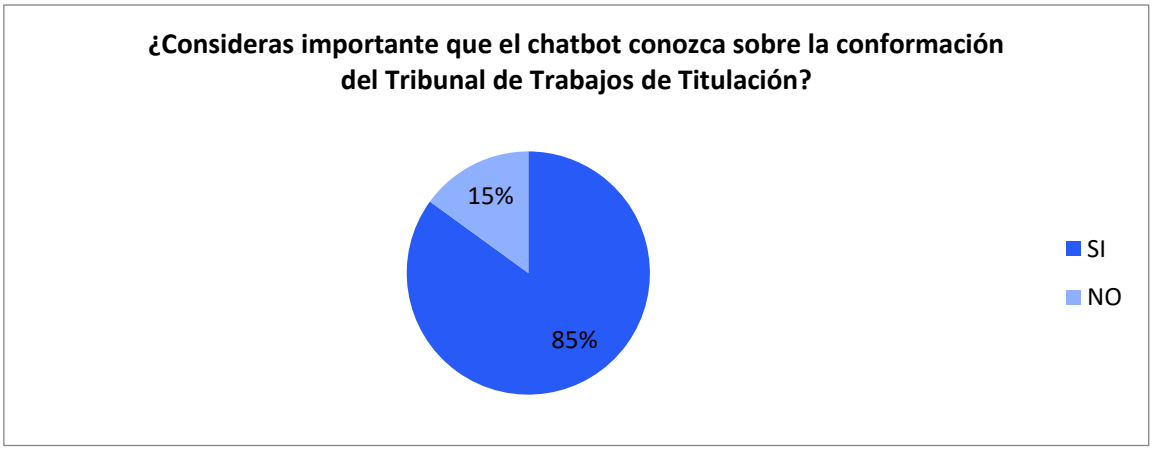


Figura 2.17 Resultados de la pregunta 16

La última pregunta de la entrevista establece ¿Piensas que el chatbot debería brindar información sobre otros temas?, es una pregunta abierta y permite conocer temas que los potenciales usuarios de la aplicación consideran importantes y no han sido considerados en las preguntas de la entrevista.

En la Figura 2.18 se observa que el 65% de los entrevistados propuso nuevos temas. En la Tabla 2.1 se muestra los temas propuestos con sus respectivas frecuencias, se puede observar que “Información sobre pasantías”, “Información sobre los docentes de la carrera” e “Información sobre la secretaría de la carrera” son los temas con mayor repetición entre los entrevistados con una frecuencia de 4, 3 y 2 respectivamente.

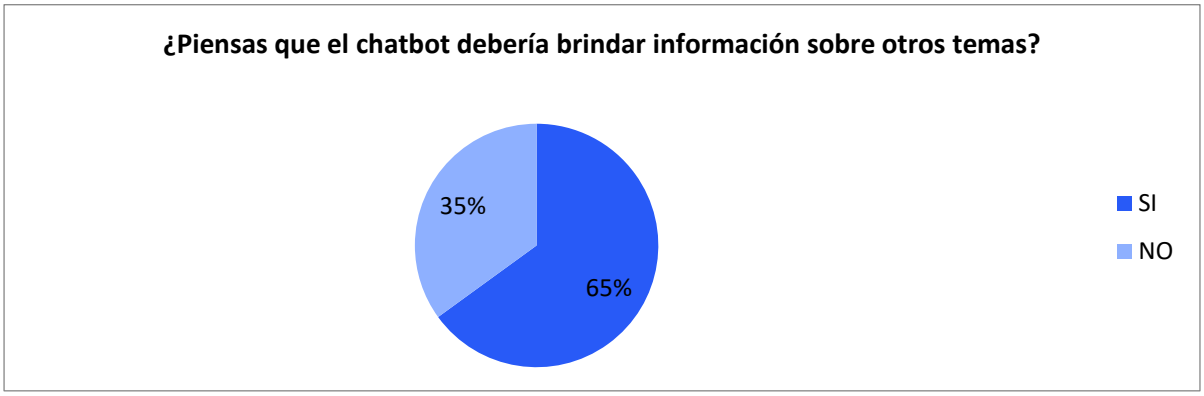


Figura 2.18 Resultados de la pregunta 17

Tabla 2.1 Temas propuestos

Temas	Frecuencia
Inicio y Cierre del periodo académico en curso	1
Fechas para cambios de carrera	1
Fechas de entrega de solicitud para pago en partes	1
Fechas de revisión de documentos	1
Fechas de extensión de entrega de anillados	1
Horarios de Supletorios	1
Información sobre pasantías	4
Información sobre la coordinación de la carrera	2
Información sobre los docentes de la carrera	3
Normativas generales, malla curricular	1
Información acerca de los seminarios de la carrera	1

2.1.2 HISTORIAS DE USUARIO

Las entrevistas realizadas permitieron identificar las expectativas que tienen los usuarios para interactuar con un chatbot que responderá preguntas frecuentes de la carrera de Tecnologías de la Información. Se agrupó la información provista por los entrevistados en categorías; cada categoría posee una descripción de la información que el usuario requiere obtener y cada una de estas categorías representa una historia de usuario.

En la Tabla 2.2 se observan las siete historias de usuario creadas. Las historias de usuario con ID 1 y 2 se les propuso a los entrevistados por medio de las preguntas uno a diecinueve de la entrevista, mientras que las historias de usuario con ID 3, 4, 5 y 6 se generaron a través de las respuestas de los entrevistados a la pregunta número veinte.

Tabla 2.2 Historias de Usuario

ID	Título	Descripción
1	Preguntas sobre el proceso de Matriculación	El usuario requiere saber información general sobre el proceso de matriculación como fechas, procedimiento, becas
2	Preguntas sobre el proceso de Titulación	El usuario requiere conocer información general del proceso de titulación como unidades de titulación y normativa
3	Preguntas sobre prácticas pre-profesionales	El usuario requiere información acerca del proceso de prácticas pre-profesionales y lugares con convenio
4	Preguntas sobre los docentes	El usuario requiere información sobre los teléfonos y oficinas de los docentes de la carrera de Tecnologías de la Información
5	Preguntas sobre fechas	El usuario requiere saber fechas de inicio y fin del semestre, fechas para cambios de carrera, revisión de documentos, solicitudes para pago en partes
6	Preguntas sobre la coordinación de la carrera	El usuario requiere información sobre el horario de atención de la coordinación de la carrera, los miembros que la conforman y su contacto
7	Preguntas de información general	El usuario requiere conocer la malla curricular, el horario de los supletorios, los seminarios de la carrera y el horario de materias

2.1.3 REQUERIMIENTOS DE LA APLICACIÓN

Una vez realizado el análisis de las entrevistas y las historias de usuario, se han identificado los requerimientos funcionales y no funcionales de este Trabajo de Titulación.

2.1.3.1 Requerimientos funcionales

- Disponer de un chatbot que responda preguntas frecuentes de la coordinación de la carrera de Tecnologías de la Información. Estas preguntas estarán divididas en seis categorías. Se requiere que el chatbot responda preguntas al menos sobre: el proceso de matriculación brindando información sobre fechas, procedimiento y becas; proceso de titulación facilitando información sobre unidades de titulación y normativa; proceso de prácticas preprofesionales ofreciendo información sobre el procedimiento y los lugares con convenio; preguntas sobre los docentes brindando información sobre el horario de atención y oficinas de los mismos; preguntas sobre fechas que faciliten información sobre el inicio y fin del semestre, fechas de revisión de documentos, solicitud de pago en partes, fechas para cambios de carrera; preguntas generales del semestre que brinden información sobre la malla curricular, el horario de supletorios, el horario de seminarios; preguntas sobre la coordinación de la carrera facilitando información sobre el horario de atención y su contacto.
- Disponer de un mecanismo de administración para actualizar las respuestas del chatbot.

2.1.3.2 Requerimientos no funcionales

- Disponer de un complemento para una página web que permita obtener respuestas a preguntas frecuentes de la coordinación de la carrera de Tecnologías de la información.
- Contar con un manual de usuario para el mecanismo de administración del chatbot.

2.1.4 TEMAS ESPECÍFICOS

Finalmente se ha definido un listado de treinta temas específicos sobre los cuales el chatbot brindará información. En la Tabla 2.3 se aprecian los temas agrupados por categorías. Cabe mencionar que las respuestas se han identificado conforme normativa e información verídica para el semestre 2019-A.

Tabla 2.3 Temas específicos

Categoría	N°	Temas Específicos
Proceso de Matriculación	1	Lugar de consulta del turno de la matrícula
	2	Fechas de matriculación
	3	Fechas de reinscripciones
	4	Lugar de consulta en el SAEW del horario de materias
	5	Tipos de becas institucionales
	6	Plazos para aplicar a las becas institucionales
Proceso de Titulación	7	Opciones de titulación
	8	Plazos para cambiarse de opción de titulación en Unidad de Titulación
	9	Descripción de la unidad de titulación: Examen Complexivo
	10	Plazos para el Examen Complexivo
	11	Descripción de la unidad de titulación: Proyectos Técnicos
	12	Descripción de la unidad de titulación: Proyecto de Investigación
	13	Requisitos para presentar anillados de Trabajo de Titulación
	14	Cómo ser declarado apto para la defensa de un Trabajo de Titulación
	15	Plazos de designación del Tribunal de Trabajos de Titulación
	16	Fechas de extensión de entrega de anillados
	17	Conformación del Tribunal de Trabajos de Titulación
Prácticas Preprofesionales	18	Proceso de prácticas pre-profesionales
	19	Ofertas de prácticas preprofesionales
Docentes	20	Teléfonos
	21	Oficinas
Fechas	22	Fechas de inicio y fin del semestre
	23	Fechas para cambios de carrera
	24	Fechas de entrega de solicitudes para pago en partes
	25	Fechas para presentar documentos
Coordinación de la carrera	16	Horario de atención
	27	Información del Personal de la coordinación
Información General	28	Malla curricular
	29	Horario de los supletorios
	30	Seminarios de la carrera

2.2 DISEÑO

Se utilizarán diferentes diagramas UML (*Unified Modeling Language*) para el diseño de la estructura y comportamiento del prototipo de chatbot propuesto en este Trabajo de Titulación; adicionalmente, se presentará a breves rasgos el diseño de la aplicación de administración.

En la etapa de diseño se han definido cinco ítems de trabajo, se los puede observar en la columna *por hacer* del tablero Kanban en la Figura 2.19. Además, se observan las columnas *en progreso* y *listo* vacías.

Figura 2.19 Tablero Kanban para la Etapa de Diseño

A continuación, se procede con el desarrollo de los ítems de trabajo planteados para esta etapa.

2.2.1 DIAGRAMA DE LA ARQUITECTURA DEL CHATBOT

En el diagrama de la arquitectura del chatbot se aprecian las interacciones entre los elementos, se tiene llamadas a procedimientos, comportamientos, peticiones, respuestas y transmisión asíncrona.

En la Figura 2.20 se detalla la arquitectura del prototipo de chatbot. Existen cuatro módulos: Módulo de Entendimiento del Lenguaje, Módulo de Seguimiento de Estado, Política de Diálogo y Módulo Generador de Lenguaje Natural. Además, esta arquitectura está formada por cuatro elementos: (1) método principal, (2) servicios WCF, (3) base de datos y (4) administración.

- El primer módulo llamado Entendimiento del Lenguaje está formado por tres partes: la autenticación del usuario, el procesamiento del mensaje del usuario y el análisis morfológico de este mensaje. Este módulo estará formado por dos servicios web (2) y por el método principal(1).

  El primer servicio web(2) expone tres métodos y se encontrará identificado por `localhost:55803/Chat.svc/`. El primer método se llamará `Autenticar` que validará las credenciales introducidas por los usuarios. El chatbot será utilizado tanto por personas particulares como por estudiantes de la EPN; pero el chatbot solo compartirá información privada como números de teléfono de los profesores con los estudiantes autenticados. El segundo método se llamará `NoAutenticar` y asignará un identificador a los usuarios particulares que ingresen al chat sin autenticación. El tercer método `FuncionChat` se encargará de realizar una petición (POST) enviando los mensajes introducidos por el usuario hasta el servidor. Cuando el mensaje llega al servidor llamará al método estático `Respuesta` de la clase `Program`(1) que contiene la lógica del procesamiento del mensaje hasta obtener la respuesta.

  El segundo servicio web(2) expone un solo método y se encontrará identificado por `localhost:55803/Grampal.svc/Postear`, se encargará de realizar el análisis morfológico del mensaje enviado por el usuario a través de peticiones a la herramienta externa Grampal.
- El segundo módulo llamado Seguimiento de Estado guardará el registro de la conversación, lo que permitirá al chatbot entender nuevos mensajes. Esta acción se realiza por medio del método estático `Respuesta` de la clase `Program(1)`.
- El tercer módulo se llama Política de Diálogo que permitirá seleccionar la respuesta más adecuada conforme el mensaje recibido. Este módulo estará compuesto de dos bibliotecas: Paquete de Información y Proveedor de Servicio, que permitirán gestionar la base de datos.

  En la Figura 2.20 a la base de datos se la puede observar con el identificador (3).

- El cuarto módulo llamado Generador de Lenguaje Natural retornará la respuesta más adecuada. Este módulo está formado por el método `FuncionChat` del servicio WCF(2) `localhost:55803/Chat.svc/` que retornará la respuesta a la petición (POST) que el cliente realizó al enviar un mensaje en el chatbot (primer módulo).

Finalmente, el elemento aplicación de administración (4) es una interfaz gráfica que permitirá gestionar las respuestas del chatbot. Esta aplicación de escritorio consumirá los métodos del servicio WCF `localhost:54787/Service.svc`, que permitirán manipular los datos de la base de datos. Es importante contar con una aplicación de administración para que el chatbot brinde información actualizada de acuerdo al periodo académico.

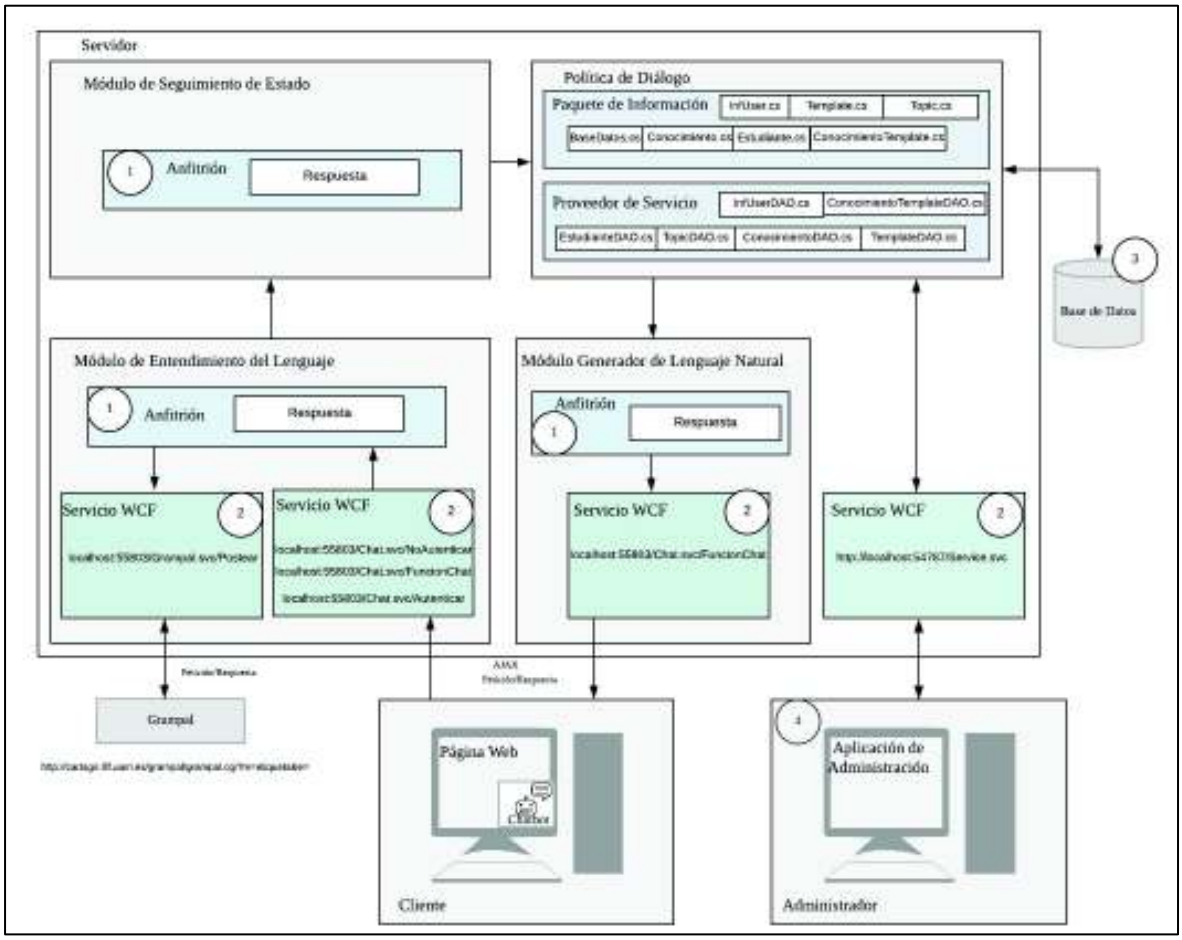


Figura 2.20 Arquitectura detallada del chatbot

2.2.2 DIAGRAMA DE CASOS DE USO

El diagrama de casos de uso representa la forma en que el sistema interactúa con las entidades externas, permite modelar el comportamiento de una aplicación con el fin de

comprender el uso de la misma. Un diagrama de casos de uso está formado de tres elementos: actores, casos de uso y relaciones de uso [24].

Un actor es una persona, organización o componente externo que interactúa con el sistema [25]. Para este prototipo de chatbot se tienen dos actores: el usuario, quien puede autenticarse o no para entablar una conversación y el administrador del chatbot, quien gestiona las respuestas del chatbot.

Un caso de uso representa las acciones ejecutadas por los actores para conseguir un objetivo específico [25]. Para esta aplicación se presentan cuatro casos de uso generales.

El administrador de la aplicación realiza un caso de uso al gestionar las respuestas del chatbot.

El usuario puede ejecutar tres casos de uso; autenticarse, no autenticarse y entablar una conversación con el chatbot para obtener cierta información. En la Figura 2.21 se observa el diagrama de casos de uso conformado por dos actores y cuatro casos de uso, se observa que para entablar una conversación con el chatbot se necesita haberse autenticado o no autenticado.

Figura 2.21 Diagrama de casos de uso

2.2.3 DIAGRAMA DE ACTIVIDADES

Un diagrama de actividades permite visualizar, especificar, construir y documentar la dinámica de un conjunto de objetos, es decir modelar el flujo de la aplicación. Dentro del flujo se puede observar pasos secuenciales, concurrentes y condiciones [26].

Se realizarán dos diagramas de actividades correspondientes a los casos de uso definidos en la sección anterior.

El primer caso de uso es ejecutado por el administrador al gestionar las respuestas del chatbot, en la Figura 2.22 se observa el diagrama de actividades de este caso de uso donde intervienen el administrador y la aplicación de administración.

Para modificar una respuesta del chatbot el administrador ingresará a la aplicación de administración, seleccionará el periodo académico a gestionar, seleccionará un parámetro, gestionará su valor y procederá a guardar los cambios.

A continuación, en el servidor se validará el valor del parámetro ingresado y se gestionará la base de datos. Si la gestión fue incorrecta se mostrará un mensaje de error, y si la gestión fue exitosa se mostrará un mensaje de éxito.

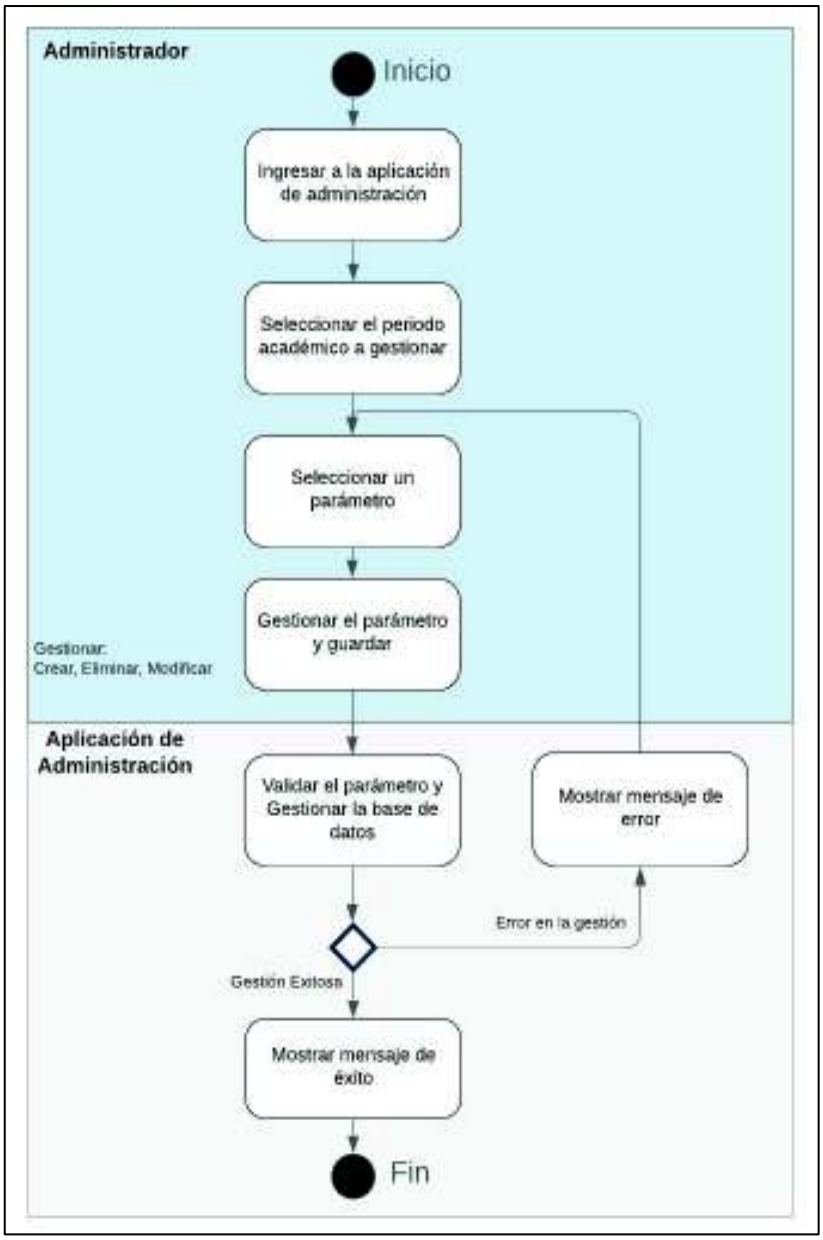


Figura 2.22 Diagrama de actividades del primer caso de uso

Los siguientes casos de uso: autenticarse, no autenticarse y entablar una conversación con el chatbot son ejecutado por el usuario. En la Figura 2.23 se muestra el diagrama de actividades para estos casos de uso donde intervienen el usuario y el servidor.

En primer lugar, el usuario ingresará a una página web donde se encontrará el chatbot y lo seleccionará. Si el usuario es estudiante de la EPN podrá autenticarse usando su correo institucional y contraseña; así accederá a ciertos datos privados como números de teléfono de los profesores. Si el usuario es una persona particular no realizará el proceso de autenticación y será dirigido inmediatamente al chat donde podrá enviar mensajes.

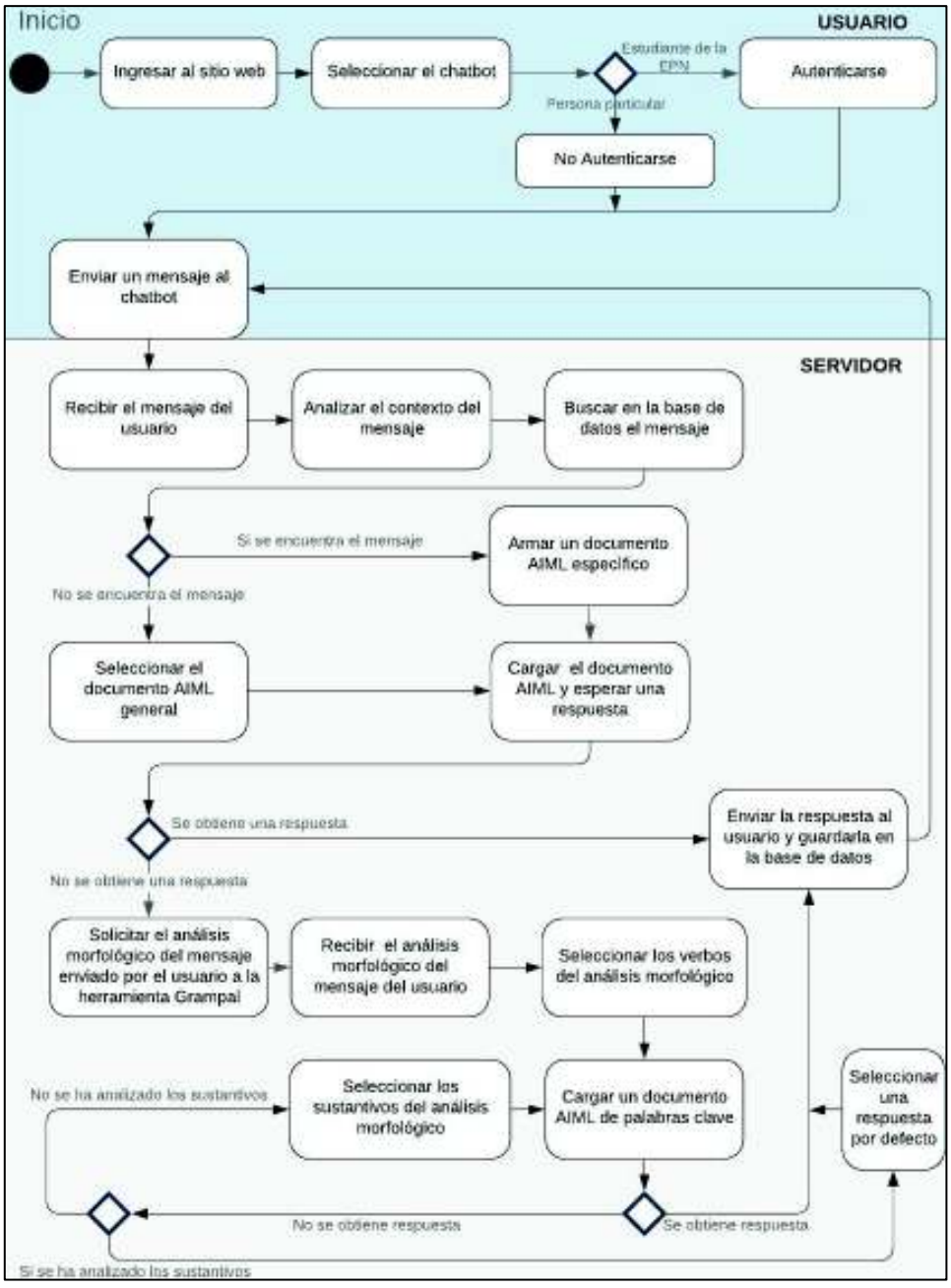


Figura 2.23 Segundo diagrama de actividades

Cuando un usuario envíe un mensaje al chatbot, el servidor recibirá el mensaje a través de un servicio WCF y analizará el contexto del mensaje.

A continuación, se buscará en la base de datos este mensaje; si se lo encuentra se armará un documento AIML específico y si no se lo encuentra se arma un documento AIML general. Cabe mencionar que un documento AIML general posee elementos con comodines del lenguaje AIML (asterisco o guion bajo), mientras que un documento AIML específico posee elementos sin comodines.

Después, se cargará el documento AIML y se obtendrá una respuesta. Si esta respuesta es válida se la enviará al usuario y se guardará en la base el identificador del usuario y el tema de la conversación.

Si la respuesta obtenida es inválida se utilizará otro servicio WCF para analizar morfológicamente el mensaje enviado por el usuario a través de la herramienta Grampal. Inicialmente se seleccionarán los verbos de la respuesta del análisis morfológico, se cargará un documento AIML de palabras clave y se esperará obtener una respuesta. Si no se obtiene una respuesta se realizará el proceso previamente indicado seleccionando los sustantivos de la respuesta del análisis morfológico. Finalmente, si no se obtiene ninguna respuesta se enviará una respuesta por defecto.

2.2.4 DIAGRAMA DE CLASES

Un diagrama de clases permite visualizar las relaciones entre las clases que conforman la aplicación. Un diagrama de clases está formado de clases y relaciones.

Clases: Para UML una clase es “una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, métodos y semántica” [26]. Una clase se representa mediante un rectángulo dividido en tres secciones: la sección superior indica el nombre de la clase, la segunda sección los atributos que caracterizan la clase y la sección inferior indica los métodos con los que el objeto interactúa. Cabe recalcar que en el diagrama de clases no se indican todos los atributos o métodos, solo los más representativos. Los atributos y los métodos de una clase pueden ser especificados mediante su nombre, tipo, visibilidad [26].

Las relaciones representan las conexiones entre las clases, en el diagrama se representa mediante una línea conectando las clases.

La cardinalidad de las relaciones indica el grado y nivel de dependencia, se anotan en cada extremo de la relación. En la Tabla 2.4 se muestran ejemplos de cardinalidades [27].

Tabla 2.4 Cardinalidad

Cardinalidad	Significado
1.. *	Uno a muchos
0..1	Cero a uno
m	Representa un número

El diagrama de clases del prototipo de chatbot que se ha planteado está formado por ocho clases.

La clase `Conocimiento` representa los mensajes que el usuario puede enviar al chatbot y este los reconoce.

La clase `Template` representa las respuestas que el chatbot tiene almacenadas para enviar al usuario.

La clase `Topic` indica el tema de la conversación del chatbot.

La clase `InfUser` representa al usuario que entabla una conversación con el chatbot.

La clase `Estudiante` representa a los alumnos de la EPN que podrán acceder a información privada.

La clase `Publico` representa a las personas que no son estudiantes de la EPN y acceden al chat sin autenticarse.

La clase `ConocimientoTemplate` representa la asociación entre los mensajes que el chatbot reconoce y las respuestas que envía.

Finalmente, la clase `Semestre` permitirá asociar las respuestas del chatbot a un periodo académico, permitiendo así que el chatbot pueda ser útil en varios periodos académicos.

Como se observa en la Figura 2.24, la clase `Conocimiento` se encuentra relacionada con las clases: `ConocimientoTemplate` y `Topic`; la relación hacia la clase `ConocimientoTemplate` presenta una cardinalidad de uno a muchos debido a que un mensaje (clase `Conocimiento`) puede tener asociadas muchas respuestas (clase `ConocimientoTemplate`). La clase `Template` se encuentra relacionada con las clases `Conocimiento`, `Topic` y `Semestre`; esto se debe a que una respuesta (clase `Template`) a un mensaje, corresponde a un tema y a un semestre específico. La clase `InfUser` se encuentra relacionada con la clase `Topic` lo que permitirá que se asocie un

usuario del chatbot con un tema de conversación, ayudando así a mantener el contexto de la conversación.

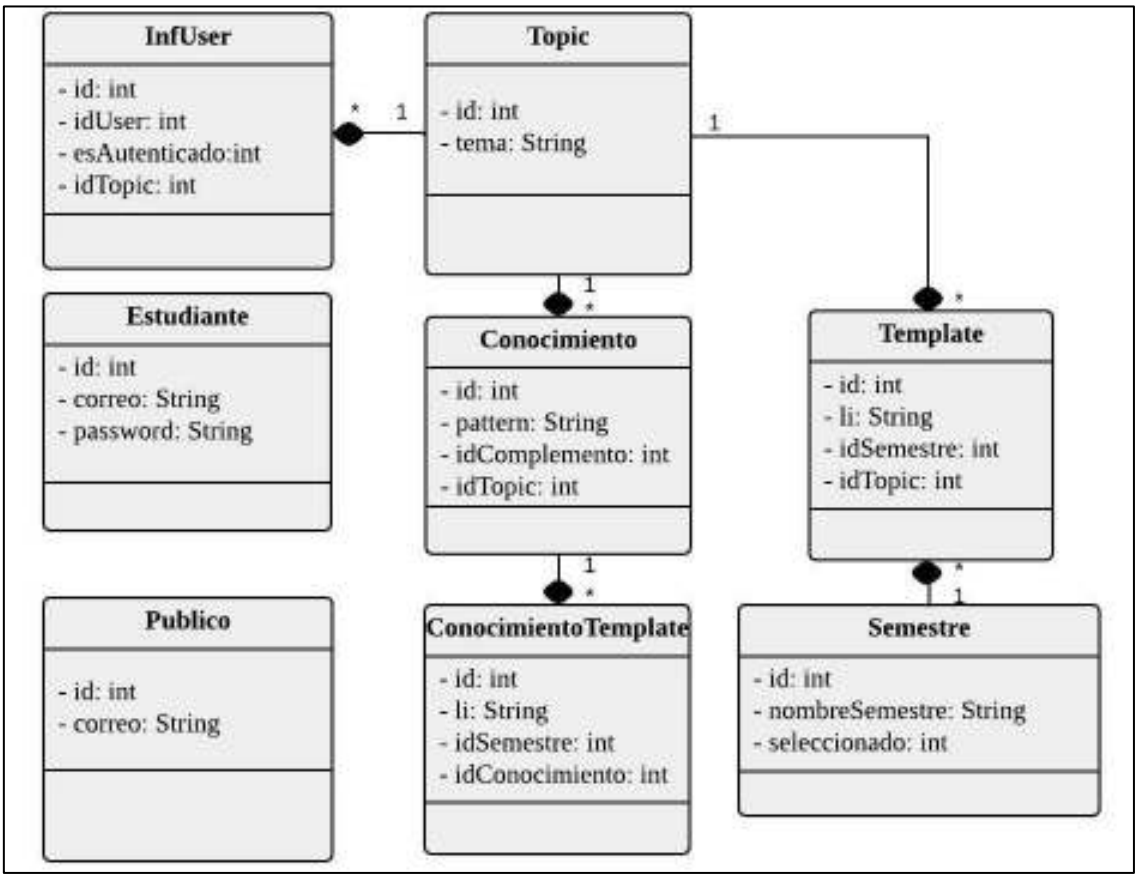


Figura 2.24 Diagrama de clases

2.2.5 DIAGRAMA RELACIONAL DE BASE DE DATOS

En la Figura 2.25 se observa el diagrama relacional que consta de ocho tablas con sus respectivos atributos y relaciones. A continuación, se enlista el nombre de las diferentes tablas que conforman la base de datos de la aplicación junto a una descripción.

`tblConocimiento`: En la columna `pattern` de esta tabla se almacenan los mensajes que el usuario puede enviar al chatbot, con sus respectivos identificadores de tema. Cabe mencionar que esta columna almacena dos tipos de mensajes los que poseen comodines del lenguaje AIML (guion bajo y asterisco) y los que no poseen comodines. Los elementos con comodines permitirán armar un documento AIML general, mientras que los elementos sin comodines permitirán armar un documento AIML específico.

`tblTemplate`: En esta tabla se almacenan las respuestas que el chatbot es capaz de proporcionar, cada respuesta está categorizada dentro de un tema por medio del `idTopic` y asociada a un periodo académico por medio de `idSemestre`.

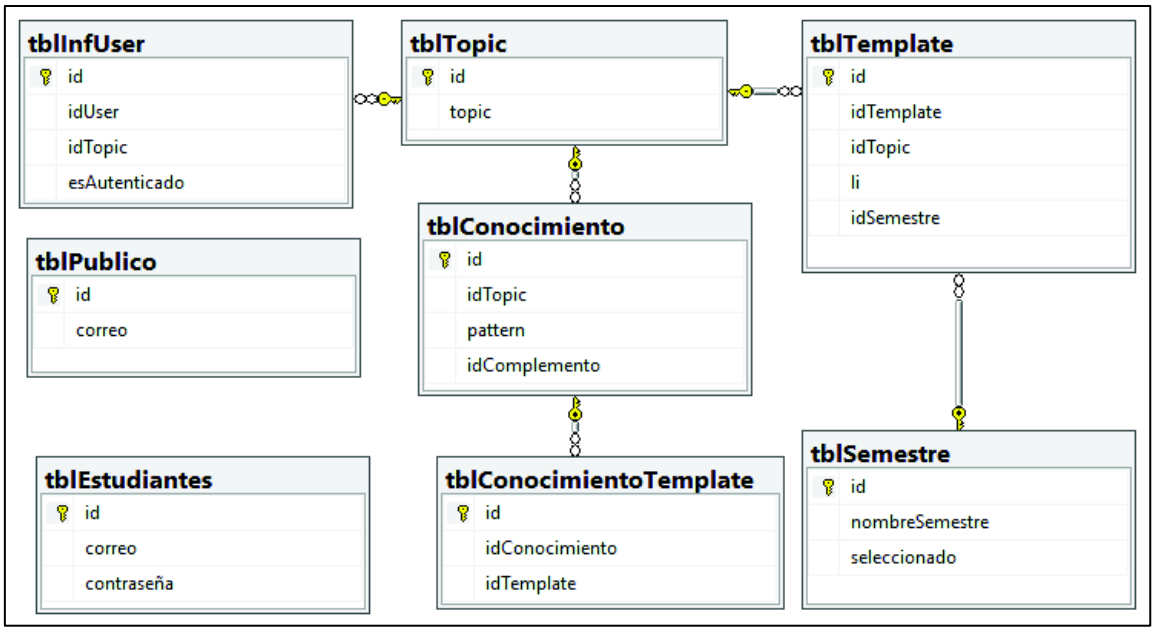


Figura 2.25 Diagrama relacional

`tblConocimientoTemplate`: Esta es una tabla intermedia que permite asociar los mensajes que el chatbot reconoce con las respuestas que enviará al usuario.

`tblInfUser`: Mantiene información de los usuarios y los temas de conversación para mantener el contexto, cabe recalcar que esta tabla es temporal y será borrada cada cierto tiempo.

`tblTopic`: En esta tabla se tienen los temas sobre los cuales el chatbot puede proporcionar información los mismos que se definieron en la etapa de análisis.

`tblEstudiantes`: Esta tabla almacena el listado de los estudiantes con su respectivo correo institucional y contraseña. Debido a que cierta información privada será únicamente compartida con estudiantes.

`tblPublico`: Esta tabla almacena el listado del público que ingresó al chatbot sin autenticación con su identificador y correo.

`tblSemestre`: En esta tabla se almacenarán periodos académicos, lo que permitirá que el chatbot pueda ser útil en varias ocasiones y brinde información actualizada.

2.2.6 SKETCH

A continuación, se muestran las diferentes interfaces del chatbot y de la aplicación de administración. En la Figura 2.26 se tiene la pestaña colapsada del chatbot, la misma que se encontrará ubicada en la esquina inferior derecha de la página web.

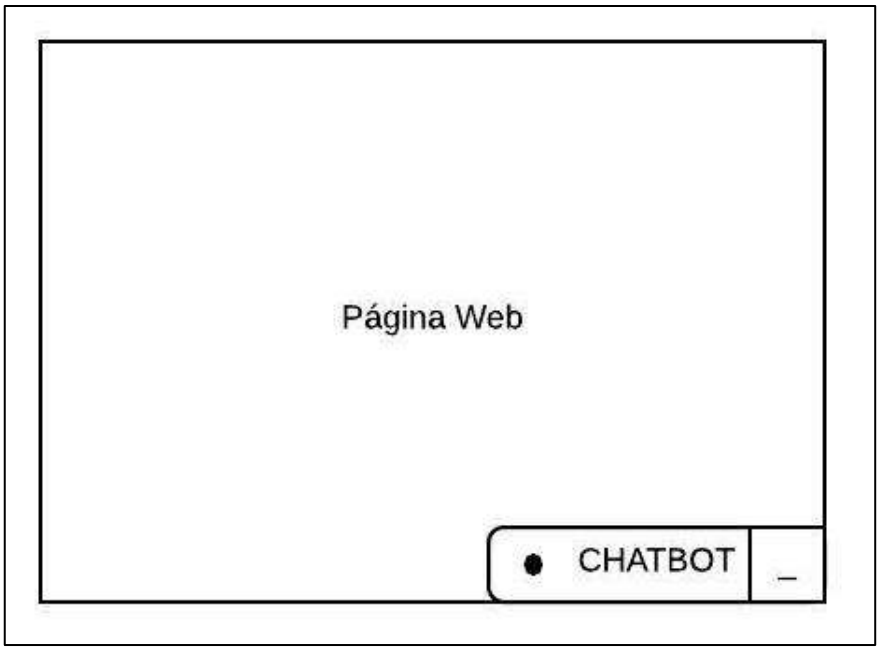


Figura 2.26 Pestaña colapsada del chatbot

En la Figura 2.27 se muestra la interfaz de autenticación del chatbot, en donde el usuario deberá ingresar su correo institucional y su contraseña.

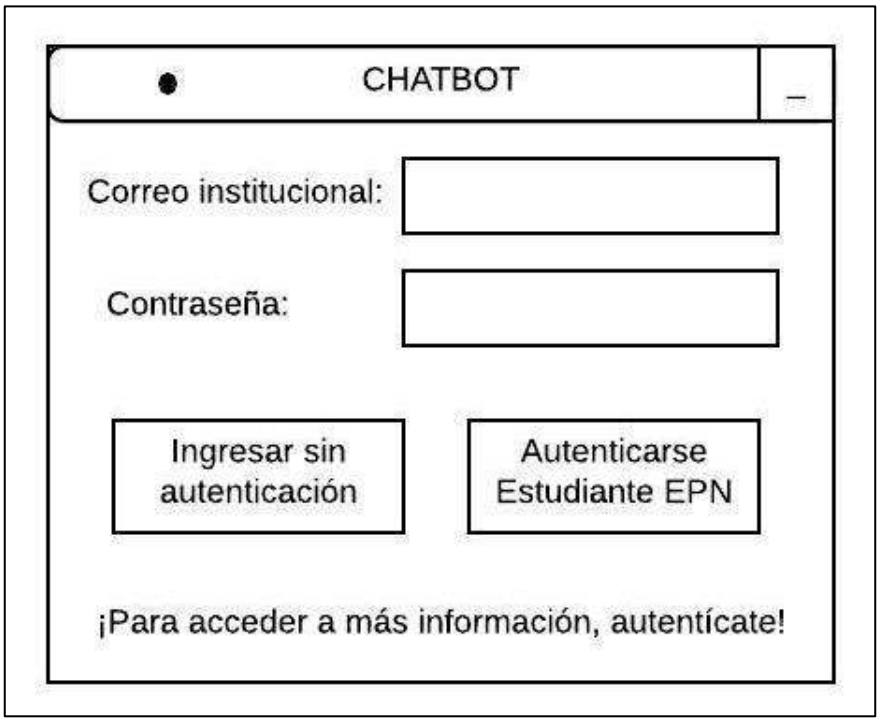


Figura 2.27 Interfaz de autenticación

En la Figura 2.28 se observa la interfaz del chat, donde el usuario podrá enviarle un mensaje al chatbot y recibir una respuesta.

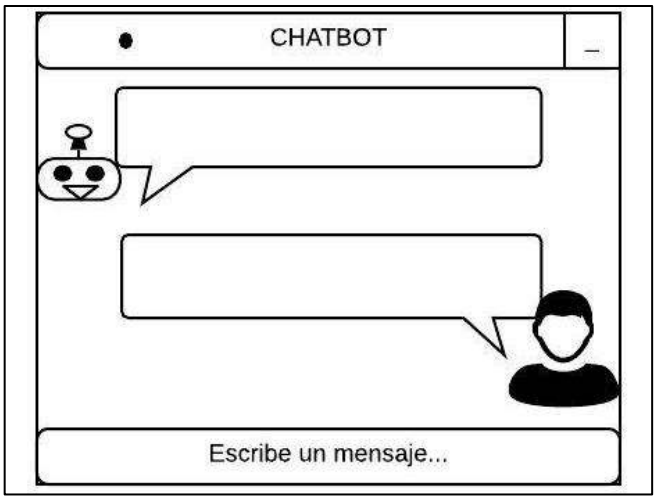


Figura 2.28 Interfaz del chat

En la Figura 2.29 se observa a breves rasgos la interfaz de administración, donde el administrador seleccionará un periodo académico, elegirá la pestaña que corresponda a los parámetros que desee gestionar y procederá a realizar los respectivos cambios.

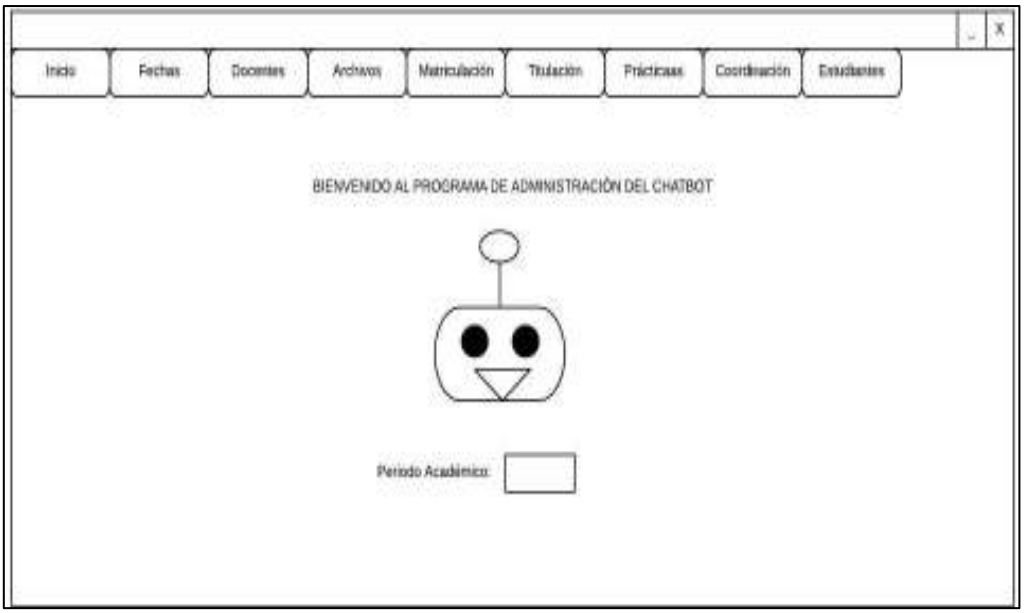


Figura 2.29 Interfaz de la aplicación de administración

2.3 IMPLEMENTACIÓN

En esta etapa se codificarán las clases de acuerdo a los diagramas planteados en la etapa de diseño. Se poblará la base de datos, se crearán los servicios web WCF y se levantará una página web HTML de prueba. En la página web HTML de prueba se incluirá el código JavaScript y HTML correspondiente al chatbot. Además, se codificará la aplicación de administración que permitirá modificar las respuestas del chatbot para que este brinde información actualizada conforme el periodo académico.

En la etapa de implementación se han definido seis ítems de trabajo, se los puede observar en la columna *por hacer* del tablero Kanban de la Figura 2.30. Además, se observan las columnas *en progreso* y *listo* vacías.

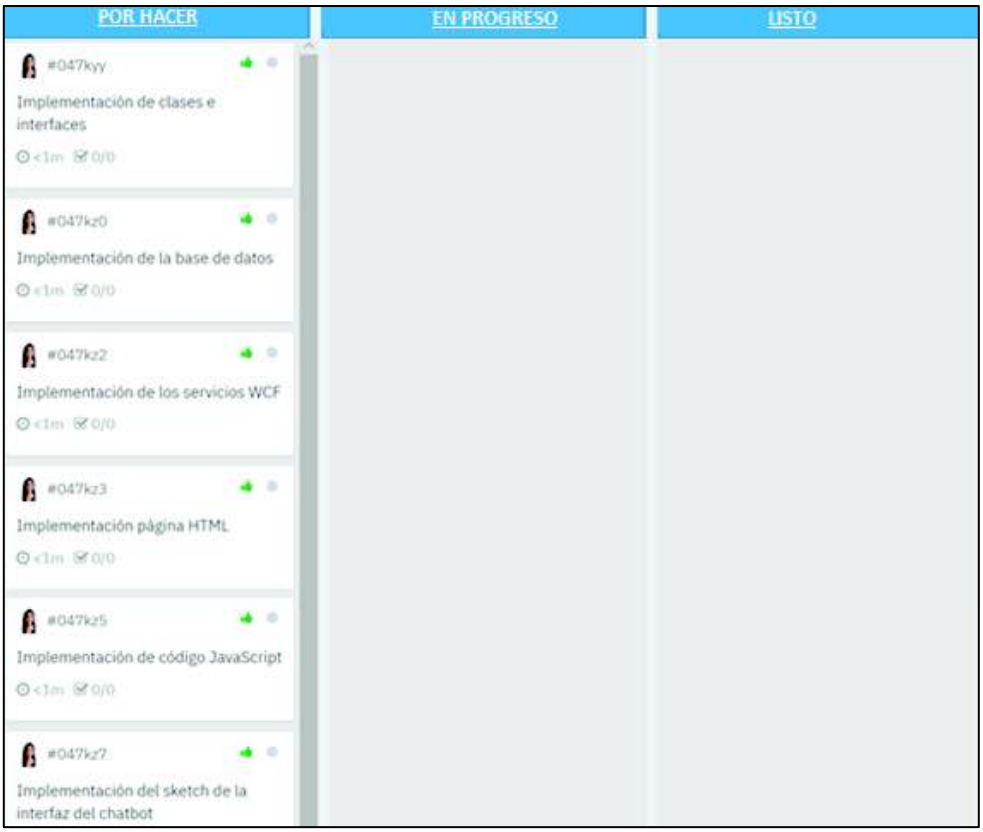


Figura 2.30 Tablero Kanban para la etapa de Implementación

A continuación, se desarrollará cada ítem definido en el tablero Kanban.

2.3.1 IMPLEMENTACIÓN DE CLASES

La etapa de desarrollo se inició implementando las librerías `Paquete de Información` y `Proveedor de Servicio`. Estas librerías contienen las clases de la aplicación. En la Figura 2.31 se observa la implementación de la primera librería llamada `Paquete de Información` que está formada por nueve clases.

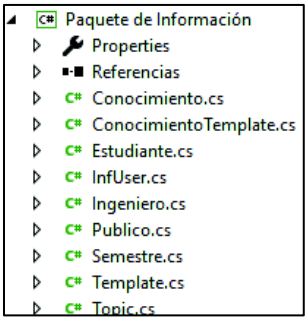


Figura 2.31 Librería de clases `Paquete de Información`

La clase `Conocimiento` representará los mensajes que el usuario puede enviar al chatbot, posee cuatro atributos como se aprecia en el Código 2.1.

```
public class Conocimiento
{
    private int id;
    private int idTopic;
    private String pattern;
    private int idComplemento;
```

Código 2.1 Clase `Conocimiento`

En el Código 2.2 se observa la clase `ConocimientoTemplate` con tres atributos, esta clase nace debido a la cardinalidad muchos a muchos entre las clases `Conocimiento` y `Template`.

```
public class ConocimientoTemplate
{
    private int id;
    private int idConocimiento;
    private int idTemplate;
```

Código 2.2 Clase `ConocimientoTemplate`

En el Código 2.3 se aprecia la clase `InfUser` que representará los temas de conversación entre los usuarios y el chatbot, posee tres atributos.

```
public class InfUser
{
    private int id;
    private int idUser;
    private int idTopic;
    private int esAutenticado;
```

Código 2.3 Clase `InfUser`

En el Código 2.4 se tiene la clase `Template` que representa las respuestas que el chatbot es capaz de enviar al usuario. Esta clase posee cuatro atributos.

```
public class Template
{
    private int id;
    private int idTopic;
    private String li;
    private int idSemestre;
```

Código 2.4 Clase `Template`

El Código 2.5 muestra la clase `Topic` que representa los temas sobre los que el chatbot proporciona información, estos temas se definieron en la etapa de análisis. Esta clase posee dos atributos.

```
public class Topic
{
    private int id;
    private String tema;
```

Código 2.5 Clase `Topic`

En el Código 2.6 se aprecia la clase `Estudiante` que representa a los estudiantes de la EPN con su respectivo correo institucional y contraseña.

```
public class Estudiante
{
    private String correo;
    private String password;
```

Código 2.6 Clase `Estudiante`

En el Código 2.7 se tiene la clase `Semestre` que representa los periodos académicos permitiendo que el chatbot brinde información actualizada. Esta clase posee tres atributos.

```
public class Semestre
{
    private int id;
    private String nombreSemestre;
    private int seleccionado;
```

Código 2.7 Clase `Semestre`

En el Código 2.8 se aprecia la clase `Ingeniero` que facilitará el manejo de información sobre los profesores en la aplicación de administración.

```
public class Ingeniero
{
    private int idTemplate;
    private String nombre;
    private String oficina;
    private String telefono;
```

Código 2.8 Clase `Ingeniero`

En el Código 2.9 se aprecia la clase `Publico` que representará a los usuarios que ingresaron al chat sin autenticación.

```
public class Publico
{
    private int id;
    private string correo;
```

Código 2.9 Clase `Publico`

La segunda librería implementada es: `Proveedor de Servicios`, en la Figura 2.32 se observa que esta librería está formada por ocho clases. La clase `BaseDatos` contiene la cadena de conexión con la base de datos, mientras que las demás clases permiten gestionar la base de datos.

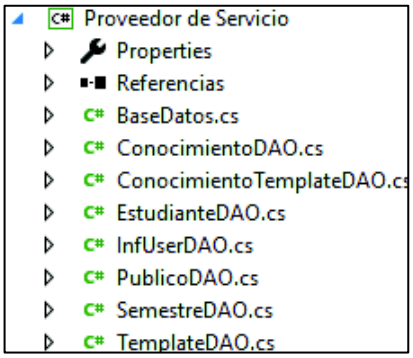


Figura 2.32 Librería de clases `Proveedor de Servicio`

La clase pública no estática `BaseDatos` posee el método público estático `obtenerConexion()`, en el Código 2.10 se indica la implementación de este método. Cabe mencionar que una clase no estática puede contener métodos estáticos, este método estático puede ser invocado aun cuando no se ha instanciado un objeto de la clase, se accede a este método por el nombre de la clase y no por el nombre de la instancia. Además, los métodos estáticos permiten almacenar un valor que puede compartirse entre varias instancias [29].

La clase `SqlConnection` gestiona una base de datos de SQL Server. Cuando se crea una instancia de esta clase, todos los parámetros se establecen en sus valores iniciales, en este caso como se puede observar en las líneas 14 a 16 solo se ha especificado los valores de la cadena de conexión y el tiempo de espera para establecer una conexión antes de generar una excepción. Este método abre una conexión con la base de datos.

```
12  public static SqlConnection obtenerConexion()
13  {
14      SqlConnection con = new SqlConnection(@"Data Source=DESKTOP-G5A74E7\SQLEXPRESS;
15                                              Initial Catalog=pruebaBase2;Integrated Security=True");
16      con.Open();
17      return con;
```

Código 2.10 Clase `BaseDatos`

Las clases `ConocimientoDAO`, `ConocimientoTemplateDAO`, `InfUserDAO`, `EstudianteDAO`, `TemplateDAO`, `PublicoDAO` y `TopicDAO` son clases públicas que poseen métodos estáticos para leer y escribir sobre la base de datos. A continuación, se detallarán algunos métodos modelos que están presentes en estas clases.

En el Código 2.11 se presenta el método `obtenerId()` que retorna un identificador de objeto (clave primaria) en una tabla de la base de datos. En la línea 16 se abre la conexión con la base de datos mediante el método `obtenerConexion()`. La línea 17 define un procedimiento almacenado que se ejecutará en la base de datos [30]. En la línea 18 mediante la clase `SqlCommand` se ejecuta la sentencia. En las líneas 19 y 20 mediante `SqlDataReader` se lee el flujo de filas que se han recibido como respuesta de la sentencia SQL. Las líneas 21 a 28 validan el valor obtenido y la línea 29 cierra la conexión SQL antes de retornar un valor.

```
13  private static int obtenerId()
14  {
15      int id;
16      SqlConnection con = BaseDatos.obtenerConexion();
17      String consulta = "SELECT MAX(id) FROM tblInfUser;";
18      SqlCommand comando = new SqlCommand(consulta, con);
19      SqlDataReader reader = comando.ExecuteReader();
20      reader.Read();
21      if (reader.IsDBNull(0))
22      {
23          id = 0;
24      }
25      else
26      {
27          id = reader.GetInt32(0);
28      }
29      con.Close();
30      return id + 1;
```

Código 2.11 Método `obtenerId()`

En el Código 2.12 se puede apreciar el método estático `buscarFilaPorPattern(String pattern)`, este método retorna un objeto de la clase `Conocimiento` cuando el texto enviado como argumento de entrada se encuentra en la base de datos. La estructura de este método es la misma que la del método `obtenerId()` detallado anteriormente. La diferencia radica en la cadena de consulta SQL presente en la línea 17. Es decir se implementó distintos métodos que interactúen con la base de datos conforme se requirió en la lógica del programa.

```
13  public static Conocimiento buscarFilaPorPattern(String pattern)
14  {
15      Conocimiento auxiliar = new Conocimiento();
16      SqlConnection con = BaseDatos.obtenerConexion();
17      String consulta = "SELECT * FROM tblConocimiento where pattern='" + pattern + "' ";
18      SqlCommand comando = new SqlCommand(consulta, con);
19      SqlDataReader reader = comando.ExecuteReader();
20      if (reader.HasRows)
21      {
22          while (reader.Read())
23          {
24              auxiliar.Id = reader.GetInt32(0);
25              auxiliar.IdTopic = reader.GetInt32(1);
26              auxiliar.Pattern = reader.GetString(2);
27              auxiliar.IdComplemento = reader.GetInt32(3);
28          }
29      }
30      con.Close();
```

Código 2.12 Método `buscarFilaPorPattern`

La tercera librería implementada es: `Anfitrión`. En la Figura 2.33 se observa que esta librería está formada por una clase llamada `Respuesta`, donde radica toda la lógica de la aplicación. Esta clase maneja al lenguaje AIML.

Figura 2.33 Librería de clases: `Logica`

La clase pública estática `Respuesta` está formada por varios métodos públicos y privados. Los principales métodos son: `Responder`, `Autenticar` y `NoAutenticar`.

El método `Autenticar` recibe como argumento de entrada el correo institucional y la contraseña del estudiante. Si la autenticación es correcta el estudiante podrá chatear con el chatbot y acceder a información privada, caso contrario se indicará falla de autenticación. En el Código 2.13 se aprecia la implementación de este método. La línea 377 llama al método estático `Autenticar` de la clase `EstudianteDAO` que interactuará con la base de datos.

```
375    public static int Autenticar(String correo, String password)
376    {
377        int i = EstudianteDAO.Autenticar(correo, password);
378        return i;
379    }
```

Código 2.13 Implementación del método `Autenticar`

El método `NoAutenticar` permite que se genere un identificador para el usuario que desea ingresar al chat sin autenticación. En el Código 2.14 se aprecia la implementación de este método. La línea 529 llama al método estático `NoAutenticar` de la clase `PublicoDAO` que interactuará con la base de datos.

```
527    public static int NoAutenticar()
528    {
529        int i = PublicoDAO.NoAutenticar();
530        return i;
```

Código 2.14 Implementación del método `NoAutenticar`

En el Código 2.15 se presenta el método `Responder` que recibe tres argumentos de entrada: el mensaje enviado por el usuario, el identificador del usuario y un identificador

de usuario que indica si autenticó o no se autenticó. Este método retorna un `string` con la respuesta del chatbot.

```
public static string Responder(String mensaje, Int32 identificador,Int32 esAutenticado)
```

Código 2.15 Firma del método `Responder`

Para temas como: Horario de materias, Horario de supletorios, Horario de seminarios y Malla curricular; el chatbot propondrá enviar la información mediante correo electrónico, entonces el usuario podrá aceptar o rechazar esta propuesta, para usuarios que se han autenticado se enviará la información al correo institucional y para usuarios que no se han autenticado se solicitará que ingresen un correo para enviar la información.

El método `Responder` inicia revisando el contexto de la conversación. En la Figura 2.34 se observa un ejemplo donde el chatbot mantiene el contexto de la conversación.

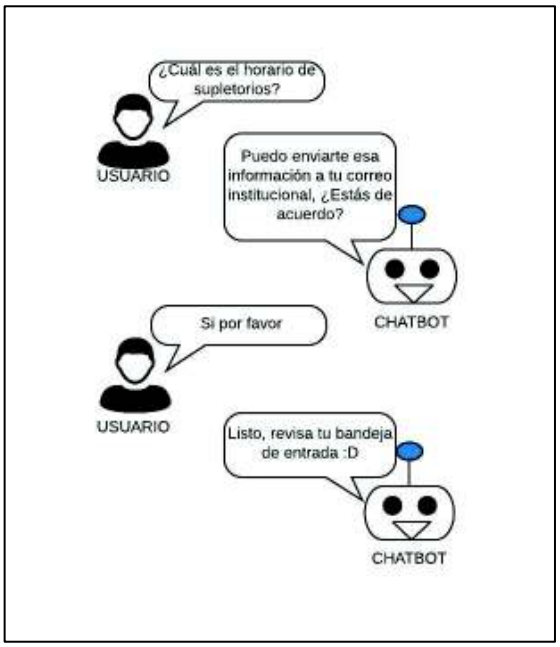


Figura 2.34 Ejemplo de conversación con contexto

Para que el chatbot sepa cuando el usuario desea que le envíen información sobre un tema se implementarán dos acciones. La primera acción es analizar el mensaje que el usuario acaba de enviar, se chequeará si el mensaje es una afirmación o es una negación; es decir si contiene las palabras: `sí`, `claro`, `por supuesto` o `no`. La segunda acción es consultar en la base de datos el tema del último mensaje enviado por el usuario. Una vez realizadas estas acciones se puede validar si el usuario desea recibir un correo electrónico con la información.

En el Código 2.16 se presenta el método `enviarCorreo` que se ejecutará si el usuario acepta que el chatbot le envíe un correo. Este método recibe tres argumentos de entrada: el correo del usuario, el identificador del contenido que se enviará y el identificador del semestre para enviar el contenido conforme el periodo académico actual. En la línea 383 se aprecia la clase `SmtpSection` que proporciona acceso a la información almacenada en los archivos de configuración, por ejemplo acceso a la cuenta de correo, contraseña, al puerto SMTP, y al *host*. Es importante mencionar que se modificó el archivo de configuración `web.config`, añadiendo las líneas de código que se aprecian en la Figura 2.35.

```
<system.net>
 <mailSettings>
   <smtp>
     <network host="smtp.gmail.com" password="alogreen16" port="587" userName="chatbot.responde@gmail.com" />
   </smtp>
 </mailSettings>
</system.net>
```

Figura 2.35 SMTP en el archivo `web.config`

En la línea 541 del Código 2.16 se crea un objeto `SmtpClient` con el *host* y el puerto SMTP. En la línea 542 se obtiene las credenciales del correo electrónico desde donde se enviará la información, para esto se creó la cuenta de Gmail `chatbot.responde@gmail.com`. En la línea 545 se crea el correo electrónico a través de la clase `MailMessage` y en las líneas siguientes se definen el cuerpo, el asunto, los archivos adjuntos y se lo envía.

Anteriormente, se describió el procedimiento que se realizará cuando el usuario pregunta sobre temas como: Horario de materias, Horario de supletorios, Horario de seminarios y Malla curricular. Ahora se describirá el procedimiento implementado cuando el usuario pregunta por temas como: Fechas de inicio y fin de semestre, Fechas de matrículas, Pasantías, Profesores, entre otros.

```
533     private static int enviarCorreo(int contenidoCorreo, string correo, int idSemestre)
534     {
535         var smtpSection = (SmtpSection)ConfigurationManager.GetSection("system.net/mail
536         string strHost = smtpSection.Network.Host;
537         int port = smtpSection.Network.Port;
538         string strUserName = smtpSection.Network.UserName;
539         string strFromPass = smtpSection.Network.Password;
540
541         SmtpClient smtp = new SmtpClient(strHost, port);
542         NetworkCredential cert = new NetworkCredential(strUserName, strFromPass);
543         smtp.EnableSsl = true;
544
545         MailMessage msg = new MailMessage(smtpSection.From,correo);
546         msg.Subject = "Respuesta a tu pregunta";
547         msg.IsBodyHtml = true;
548         Attachment adjunto;
```

Código 2.16 Implementación del método `enviarCorreo`

Una vez recibido el mensaje nuevo, se lo buscará en la base de datos por medio de la función estática `ConocimientoDAO.buscarFilaPorPattern` que obtiene datos desde la tabla `tblConocimiento` en la base de datos.

En el Código 2.17 se observa la implementación de esta función, en la línea 39 se define la sentencia SQL, mientras que desde la línea 41 en adelante se lee el resultado y se lo almacena en un objeto del tipo `Conocimiento`.

```
35  public static Conocimiento buscarFilaPorPattern(String pattern)
36  {
37      Conocimiento auxiliar = new Conocimiento();
38      SqlConnection con = BaseDatos.obtenerConexion();
39      String consulta = "SELECT * FROM tblConocimiento where pattern='" + pattern + "' ";
40      SqlCommand comando = new SqlCommand(consulta, con);
41      SqlDataReader reader = comando.ExecuteReader();
42      if (reader.HasRows)
43      {
44          while (reader.Read())
45          {
46              auxiliar.Id = reader.GetInt32(0);
47              auxiliar.IdTopic = reader.GetInt32(1);
48              auxiliar.Pattern = reader.GetString(2);
49              auxiliar.IdComplemento = reader.GetInt32(3);
50          }
51      }
52      con.Close();
53      return auxiliar;
54  }
```

Código 2.17 Implementación del método `buscarFilaPorPattern`

En este punto se presentan dos opciones: haber encontrado en la base de datos el mensaje que el usuario envió y no haberlo encontrado. Si no se encontró el mensaje se arma un documento AIML general y si fue encontrado se arma un documento AIML específico.

Cabe recordar que un documento AIML general posee elementos con comodines del lenguaje AIML (asterisco o guion bajo), mientras que un documento AIML específico no posee elementos con comodines.

Para armar un documento AIML que prácticamente es un documento XML con las correspondientes etiquetas de AIML, se usó la clase `XElement` que crea un árbol XML.

En el Código 2.18 se observa el código que permite armar un documento XML con las etiquetas `<aiml>`, `<category>`, `<pattern>`, `<template>` y `<random>`. En las líneas 106 a 110 se añaden las posibles respuestas encontradas en la base de datos.

Finalmente, en las líneas 113 y 114 se guarda el documento.

```
100     XElement xmlConocimiento = new XElement("aiml",
101     new XElement("category",
102     new XElement("pattern", conocimiento.Pattern),
103     new XElement("template", new XElement("random"))));
104
105     //Insertar respuestas
106     foreach (var item in templates)
107     {
108         XElement child1 = xmlConocimiento.Element("category").Element("template").Element("random");
109         child1.Add(new XElement("li", item));
110     }
111
112     //Guardar xml
113     doc.LoadXml(xmlConocimiento.ToString());
114     doc.Save("C:\\Users\\Giselita\\AppData\\Local\\Temp\\Chatbot\\doc.aiml");
```

Código 2.18 Código para armar documento XML

Una vez que se tiene guardado el documento XML, se procede a usar la librería `AIML.bot` para C#. Esta librería se la puede descargar desde la página web: `https://sourceforge.net/projects/aimlbot/files/`. Una vez descargada la librería `AIMLbot.dll`, se la debe agregar a la carpeta `Debug` dentro del proyecto y añadir la referencia correspondiente. Por medio de las clases y métodos de esta librería se vinculará el mensaje enviado por el usuario y el documento AIML construido para así obtener una respuesta, que será enviada al usuario. En la línea 117 del Código 2.19 se instancia un objeto de la clase `Bot`, que permitirá vincular el documento AIML. En la línea 129 se construye una nueva petición mediante la clase `Request` y en la línea 130 se obtiene una respuesta.

```
117     Bot AI = new Bot(); //Define el objeto AI, para guardar la información del bot
118     string configPath = Directory.GetCurrentDirectory();
119
120     AI.loadSettings(); // Cragar la configuración del chatbot, desde la carpeta config
121
122     //terminar de cargar aiml
123     AI.loadAIMLFromXML(doc, "C:\\Users\\Giselita\\AppData\\Local\\Temp\\Chatbot\\doc.aiml");
124     AI.isAcceptingUserInput = false; // Desactivar  la entrada de texto del usuario mientras
125     User myUser = new User("Giss", AI); // Crear un nuevo usuario con la información cargada
126     AI.isAcceptingUserInput = true; // Cuando el chatbot se ha cargado se permite la entrada
127
128
129     Request r = new Request(mensaje, myUser, AI); // Lee la consola y ao
130     Result res = AI.Chat(r); //  Envía la solicitud en el objeto resultado, la respuesta est
```

Código 2.19 Uso de la librería `AIMLbot`

La respuesta obtenida puede ser válida o inválida (nula), dependiendo de la información de la base de datos. Cuando la respuesta es nula, se realizará un análisis morfológico. En el Código 2.20 se tiene la implementación de una petición POST a `localhost:55803/Grampal.svc/Postear`, para interactuar con la herramienta Grampal. En la línea 161 se define la URL del servicio WCF y en la línea 162 se indica que se utilizará el método POST. En las líneas 163 y 164 se definen las cabeceras de la petición (indican que se trabajará con JSON y sin cache). En la línea 166 se arma la consulta adjuntando el mensaje para analizar. En la línea 167 se realiza la petición y en la

línea 168 se obtiene la respuesta. El resultado de esta petición es un objeto JSON que será procesado mediante el método `AnalisisPalabra` para obtener solo los datos de interés.

```
161     var client = new RestClient("http://localhost:55803/Grampal.svc/Postear");
162     var request = new RestRequest(Method.POST);
163     request.AddHeader("cache-control", "no-cache");
164     request.AddHeader("Content-Type", "application/json");
165     const string quote = "\"";
166     string consulta = "{\n" + quote + "data" + quote + ":" + quote + mensajeInicial + quote + ",\n}";
167     request.AddParameter("undefined", consulta, ParameterType.RequestBody);
168     IRestResponse response = client.Execute(request);
169     string cadenaTexto = response.Content;
170     string loQueSeAnalizará = AnalisisPalabra(cadenaTexto,identificador,idSemestre);
171     return loQueSeAnalizará;
```

Código 2.20 Petición POST al servicio `Grampal`

En la Figura 2.36 se presenta un ejemplo de respuesta de una petición POST a `Grampal.svc/Postear`. El texto analizado fue "`Quiero saber cómo`" y la respuesta se encuentra en formato JSON. En esta figura se puede observar la categoría, el lema y el rasgo de cada palabra del texto analizado. A continuación, mediante el método `AnalisisPalabra` se extraerá únicamente el lema de los verbos y los sustantivos.

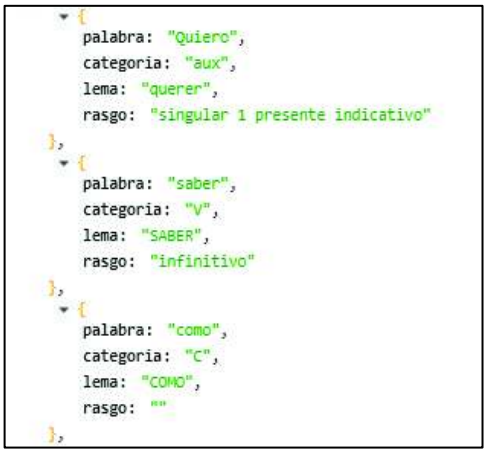


Figura 2.36 Respuesta en formato JSON de la petición POST al servicio WCF `Grampal`

En el Código 2.21 se presenta la implementación del método `AnalisisPalabra` que recibe como argumento de entrada el análisis morfológico. En la línea 173 se valida si existen verbos, si es así se extraerá el lema del primer verbo y se lo analizará mediante AIML esperando obtener una respuesta válida. Si la respuesta está vacía, se analizará el lema del segundo verbo mediante AIML. Si no existen verbos o si se obtuvo una respuesta inválida en el análisis, se procederá a extraer el primer sustantivo y analizarlo con AIML, tal como se observa en la línea 199. Si después de realizar estos análisis aún no se obtiene una respuesta válida, se procederá a enviar una respuesta por defecto. El

método `AyudaGrampaVerbo` permite realizar el análisis AIML para cada verbo o sustantivo encontrado.

```
164     private static string AnalisisPalabra(string cadenaTexto)
165     {
166         string cadenaTexto2 = cadenaTexto;
167         string primerSustantivo = "";
168         string primerVerbo = "";
169         string resultado = "";
170         string resultado2 = "";
171         const string quote = "\"";
172 
173         if (cadenaTexto.Contains(quote + "V" + quote))
174         {
175             //busca el primer verbo en el analisis morfologico
176             resultado2 = cadenaTexto2.Substring(cadenaTexto2.IndexOf(quote + "V" + quote)
177             primerVerbo = resultado2.Substring(0, resultado2.IndexOf(quote));
178             //Busca el primer verbo en aiml
179             string primeraBusqueda = AyudaGrampaVerbo(primerVerbo);
180 
181             //Busca el segundo verbo en el analisis morfologico
182             if (primeraBusqueda == "" && resultado2.Contains(quote + "V" + quote))
183             {
184                 string resultado3 = resultado2.Substring(resultado2.IndexOf(quote + "V" +
185                 string segundoVerbo = resultado3.Substring(0, resultado3.IndexOf(quote));
186 
187                 //Busca segundo verbo en aiml
188                 string segundaBusqueda = AyudaGrampaVerbo(segundoVerbo);
189 
190                 if (segundaBusqueda != "")
191                 {
192                     return segundaBusqueda;
193                 }
194             }
195             else if (primeraBusqueda!="")
196             {
197                 return primeraBusqueda;
198             }
199             else if (cadenaTexto.Contains(quote + "N" + quote))
200             {
201                 resultado = cadenaTexto.Substring(cadenaTexto.IndexOf(quote + "N" + quote) + 12,
202                 primerSustantivo = resultado.Substring(0, resultado.IndexOf(quote));
203                 return AyudaGrampaVerbo(primerSustantivo);
204             }
205         }
206             return "ni idea";
207 
```

Código 2.21 Implementación del método `AnalisisPalabra`

Una vez que se haya obtenido una respuesta se guardará en la base de datos el identificador del usuario, el tema asociado a la respuesta y un identificador que indicará si el usuario al ingreso al chatbot se autenticó.

En el Código 2.22 se muestran las líneas de código que almacenarán estos datos en la tabla `tblInfUser` de la base de datos. En la línea 177 se busca el tema de la respuesta. En la línea 178 se guarda el identificador del usuario, en la línea 179 se guarda el identificador de autenticación y en la línea 180 se inserta la información en la tabla `InfUser`. A demás, en la línea 182 se verifica si el usuario puede acceder a la información que solicita.

Finalmente, en la línea 187 se retorna el mensaje que será enviado al usuario.

```
177  informacionUsuario.IdTopic = TemplateDAO.buscarIdTopicPorRespuesta(res.Output,idSemestre);
178  informacionUsuario.IdUser = identificador;
179  informacionUsuario.EsAutenticado = esAutenticado;
180  InfUserDAO.insertarInfUser(informacionUsuario);
181
182  if (informacionUsuario.IdTopic==18 && identificador==0)
183  {
184      return "La información solicitada es privada. Favor ¡Autentícate!";
185  }
186
187  return res.Output; // Mostrar por consola el resultado
```

Código 2.22 Código para almacenar el tema de la conversación y el identificador del usuario

2.3.2 IMPLEMENTACIÓN DE LA BASE DE DATOS

En la etapa de diseño se estableció el diagrama relacional de la aplicación, donde se presentó la base de datos formada de ocho tablas: `tblConocimiento`, `tblConocimientoTemplate`, `tblEstudiante`, `tblInfUser`, `tblTemplate`, `tblTopic`, `tblSemestre` y `tblPublico`. A continuación, se indicarán algunas sentencias usadas para la creación y población de la base de datos.

El Código 2.23 indica la sentencia que permitirá crear una base de datos llamada `baseChatbot`.

```
CREATE DATABASE baseChatbot
```

Código 2.23 Creación de la base de datos

El Código 2.24 presenta la creación de la tabla `tblConocimiento` que consta de cuatro columnas. En la línea 34 se crea el campo `id` como clave primaria que identificará de forma única a cada fila dentro de la tabla. En la línea 35 se observa la creación de la columna `idTopic` la misma que es clave foránea y vincula a esta tabla con la tabla `tblTopic`. En la línea 36 se crea la columna `pattern` que almacenará cadenas de caracteres. Finalmente, en la línea 37 se crea la columna `idComplemento` que identificará si el elemento contiene comodines del lenguaje AIML (asterisco o guion bajo).

```
33  CREATE TABLE tblConocimiento (
34  id INT NOT NULL PRIMARY KEY,
35  idTopic INT NOT NULL FOREIGN KEY REFERENCES [tblTopic] (id) ON DELETE CASCADE,
36  pattern VARCHAR(200),
37  idComplemento INT
```

Código 2.24 Creación de la tabla `tblConocimiento`

En el Código 2.25 se tiene un ejemplo para insertar una fila de valores en una tabla. En primer lugar se especifica la tabla con la que se quiere trabajar, en este caso

tblEstudiantes. Después, se indican los valores que se quieren añadir de acuerdo a las columnas de la tabla; estos valores deben estar separados por comas y si son cadenas de caracteres se los debe poner entre comillas.

```
insert into tblEstudiantes
values (1,'gisela.osorio@epn.edu.ec','2013')
```

Código 2.25 Ejemplo de población de la tabla tblEstudiantes

En el ANEXO B se encuentra el *script* en lenguaje SQL que se usó en esta aplicación.

2.3.3 IMPLEMENTACIÓN DE LOS SERVICIOS WCF

Se crearon dos servicios web WCF: WCFGram y WCFService. En primer lugar, se describirá el servicio WCFGram, en la Figura 2.37 se muestra la carpeta de archivos de este servicio.

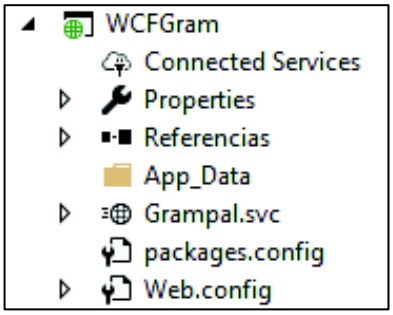


Figura 2.37 Servicio WCF WCFGram

Se implementó el servicio Grampal.svc que se encuentra adornado con el atributo ServiceContract que lo define como un contrato de servicio, tal como se observa en la línea 17 del Código 2.26, mientras que en la línea 18 de esta figura se habilita la compatibilidad con ASP.NET.

```
17    [ServiceContract(Namespace = "")]
18    [AspNetCompatibilityRequirements(RequirementsMode = AspNetCompatibilityRequirementsMode.Allowed)]
```

Código 2.26 Contrato de servicio de Grampal.svc

El servicio Grampal.svc expone el método Postear que acepta como parámetro de entrada una cadena de caracteres y retorna una lista de objetos del tipo Estructura. En el Código 2.27 se observa la primera parte de esta implementación. En la línea 21 mediante el atributo OperationContract se indica que el método Postear se expondrá como operación del servicio web. La línea 22 establece el tipo de solicitud

HTTP y el formato de respuesta que aceptará, en este caso es POST y JSON. Este método interactuará con la herramienta Grampal y retornará el análisis morfológico del texto enviado.

Para realizar el análisis morfológico se utilizará Selenium WebDriver que es una librería que permite realizar pruebas de páginas web. La librería `Selenium.WebDriver` y el controlador correspondiente al navegador seleccionado (`Selenium.Chrome.WebDriver` para Chrome) pueden ser descargados usando *Nuget package manager*.

Estas librerías permiten interactuar con páginas web por medio de líneas de código. Para la implementación de esta aplicación se realizará una consulta a la siguiente dirección: `http://cartago.lllf.uam.es/grampal/grampal.cgi?m=etiqueta&e=`.

En la línea 28 del Código 2.27 se establece el navegador que se utilizará para acceder a la página web, en este caso es Chrome. Mientras que en la línea 30 se crea un objeto `IWebDriver` que establece la URL de la herramienta Grampal. En la línea 33 se asocia el elemento de la página web donde se ingresará el texto para analizar. En la línea 34 se da clic en el botón “Etiqueta” y finalmente se obtiene la respuesta en la línea 37.

```
21      [OperationContract]
22      [WebInvoke(Method = "POST", ResponseFormat = WebMessageFormat.Json)]
        0 referencias
23      public List<Estructura> Postear(string data)
24      {
25      string oracion = data;//oracion gramatical a analizar
26      
27      /*configuraicon de modo headless*/
28      ChromeOptions configuracion = new ChromeOptions();
29      configuracion.AddArgument("--headless");//sin ventanas graficas
30      IWebDriver navegador = new ChromeDriver(configuracion);
31      
32      navegador.Navigate().GoToUrl("http://cartago.lllf.uam.es/grampal/grampal.cgi?m=etiqueta&e=");
33      navegador.FindElement(By.Name("e")).SendKeys(oracion);//escribir oracion deseada en linea de
34      navegador.FindElement(By.CssSelector("input[type='submit']")).Click();//clic en boton para ar
35      
36      /*el elemento navegador en este punto contiene toda la pagina con los resultados del analisis
37      List<IWebElement> filas = navegador.FindElements(By.TagName("tr")).ToList();//obtengo filas d
38      
```

Código 2.27 Primera parte del método `Postear`

En el Código 2.28 se observa la segunda parte de la implementación del método `Postear` que indica el proceso que se realizará al haber obtenido el análisis morfológico.

Desde la línea 48 se forma la respuesta que retornará el método `Postear`, eliminando el texto innecesario y colocando la palabra, la categoría, el lema y los rasgos de cada palabra del texto analizado.

```
41      foreach (IWebElement fila in filas)
42      {
43          List<IWebElement> cols = fila.FindElements(By.TagName("td")).ToList();/
44          Estructura item = new Estructura();
45          int contador = 0;
46          foreach (IWebElement col in cols)
47          {
48              if (contador == 0)
49              {
50                  item.palabra = col.Text;
51                  contador++;
52              }
53              else if (contador == 1)
54              {
55                  item.categoria = col.Text.Replace("categoría ", "").Trim();//ar
56                  contador++;
57              }
58              else if (contador == 2)
59              {
60                  item.lema = col.Text.Replace("lema ", "").Trim();
61                  contador++;
62              }
63              else if (contador == 3)
64              {
65                  item.rasgo = col.Text.Replace("rasgos ", "").Trim();
66                  contador = 0;
67              }
68          }
69          datos.Add(item);
```

Código 2.28 Segunda parte del método `Postear`

En la Figura 2.38 se presenta la carpeta de archivos del servicio `WCFService`, que está formado por el servicio web `Chat.svc`, la página web `VentanaChat.html` donde se añadirá el chatbot, el código JavaScript `codigoJSc.js` que permitirá la interactividad con el chatbot y la comunicación asíncrona entre la página web y el servicio web, la hoja de estilos CSS `Styles.css` que otorgará un buen aspecto al chatbot, la biblioteca de JavaScript `jquery-3.3.1.js` que facilitará el uso de JavaScript y dos imágenes que identificarán al chatbot y al usuario en la ventana de chat.

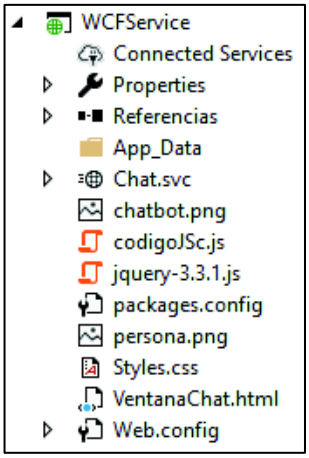


Figura 2.38 Servicio WCF `WCFService`

En segundo lugar se describirá el servicio `Chat.svc` que es un servicio WCF con AJAX habilitado, se encuentra adornado con el atributo `ServiceContract` que lo define como

un contrato de servicio; tal como se observa en la línea 15 del Código 2.29, mientras que en la línea 18 se habilita la compatibilidad con ASP.NET.

```
15    [ServiceContract(Namespace = "")]
16    [AspNetCompatibilityRequirements(RequirementsMode = AspNetCompatibilityRequirementsMode.Allowed)]
```

Código 2.29 Contrato de servicio `Chat.svc`

El servicio `Chat.svc` expone tres métodos: `FuncionChat`, `Autenticar` y `NoAutenticar`. En el Código 2.30 se observa la primera parte del método `FuncionChat`. En la línea 19 mediante el atributo `OperationContract` se indica que el método `FuncionChat` se expondrá como operación del servicio web. La línea 20 establece el tipo de solicitud HTTP y el formato de respuesta que aceptará, en este caso es POST y JSON.

Este método emplea el método estático `Respuesta.Responder` que recibe como argumentos de entrada el mensaje y el identificador del usuario, y retorna un objeto `Respuesta` que es un `string` con la respuesta del chatbot.

```
19    [OperationContract]
20    [WebInvoke(Method = "POST", ResponseFormat = WebMessageFormat.Json)]
      0 referencias
21    public Chatear FuncionChat(string nombre,Int32 identificador)
22    {
23        return new Chatear() {
24            Respuesta = Respuesta.Responder(nombre,identificador)
```

Código 2.30 Método `FuncionChat`

El Código 2.31 presenta el método `Autenticar`, la línea 28 mediante el atributo `OperationContract` indica que este método se expondrá como operación del servicio web. La línea 29 establece el tipo de solicitud HTTP y el formato de respuesta que aceptará, en este caso es POST y JSON. El método `Autenticar` recibe como argumento de entrada el correo y la contraseña del estudiante que interactuará con el chatbot, además llama al método estático `Respuesta.Autenticar` y retorna un objeto del tipo `Autenticado` que indicará si la autenticación fue exitosa o falló.

```
28    [OperationContract]
29    [WebInvoke(Method = "POST", ResponseFormat = WebMessageFormat.Json)]
      0 referencias
30    public Autenticado Autenticar(String correo, String password)
31    {
32        return new Autenticado()
33        {
34            idAutenticado = Respuesta.Autenticar(correo, password)
```

Código 2.31 Método `Autenticar`

El Código 2.32 presenta la implementación del método `NoAutenticar` que no recibe argumentos de entrada, llama al método estático `Respuesta.NoAutenticar` y retorna un objeto del tipo `Autenticado` que proveerá un identificador al usuario que ingresa al chatbot sin autenticación.

```
[OperationContract]
[WebInvoke(Method = "POST", ResponseFormat = WebMessageFormat.Json)]

public Autenticado NoAutenticar()
{
    return new Autenticado()
    {
        idAutenticado = Respuesta.NoAutenticar()
```

Código 2.32 Método `NoAutenticar`

A continuación, se establecen los contratos de datos para dos clases: `Chatear` y `Autenticado`. En la línea 39 del Código 2.33 se observa el atributo `DataContract` que especifica un contrato de datos para la clase `Chatear`. En la línea 42 por medio del atributo `DataMember` se indica que el atributo `Respuesta` se serializará.

Además, en la línea 46 se observa el atributo `DataContract` que especifica un contrato de datos para la clase `Autenticado`. En la línea 49 por medio del atributo `DataMember` se indica que el atributo `idAutenticado` se serializará.

```
39      [DataContract]
        2 referencias
40      public class Chatear
41      {
42          [DataMember]
            1 referencia
43          public string Respuesta { get; set; }
44      }
45
46      [DataContract]
        2 referencias
47      public class Autenticado
48      {
49          [DataMember]
            1 referencia
50          public int idAutenticado { get; set; }
```

Código 2.33 Contrato de Datos

2.3.4 IMPLEMENTACIÓN DE LA PÁGINA HTML

Se implementó una página HTML donde se añadirán las líneas de código de la interfaz del chatbot. En el Código 2.34 se tiene el código HTML de esta página web, cuya línea 1 indica el tipo de documento. En la línea 2 mediante la etiqueta `<html>` da inicio al documento. Las etiquetas `<head>` y `<body>` indican la cabecera en las líneas 3 a 8 y el cuerpo de la página HTML en las líneas 9 a 11. En la cabecera se especifican los *scripts*,

título y hojas de estilo y dentro del cuerpo se tendrá el contenido que el usuario visualizará.

```
2   <html>
3   <head>
4       <script src="jquery-3.3.1.js" type="text/javascript"></script>
5       <title>Carrera de Tecnologías de la Información</title>
6       <link rel="stylesheet" type="text/css" href="/Styles.css" media="screen" />
7       <script src="codigoJSc.js"></script>
8   </head>
9   <body>
10
11  </body>
12  </html>
```

Código 2.34 Página HTML de prueba

2.3.5 IMPLEMENTACIÓN DEL CÓDIGO JAVASCRIPT

Se implementó un *script* que permitirá interactuar con los métodos `Autenticar`, `NoAutenticar` y `FuncionChat` del servicio WCF `Chat.svc`.

En el Código 2.35 se observa el código JavaScript con AJAX para interactuar con el método `Autenticar` del servicio web `Chat.svc`, este método permitirá que un estudiante de la EPN se autentique y pueda acceder a más información. La línea 18 indica que se ejecutará el código JavaScript cuando se presione en el botón `idCredenciales`. La petición AJAX inicia en la línea 20, en la línea 21 se indica la URL del servicio, la línea 22 indica el tipo de petición HTTP, la línea 23 define los datos que serán enviados (el correo y la contraseña del estudiante). Las líneas 24 y 25 establecen el tipo de dato que se enviará y que se recibirá, en los dos casos es JSON. La línea 26 señala que se va a procesar los datos. Finalmente, las línea 28 a 32 se ejecutan solo si la petición fue realizada con éxito, si es así se almacenará el identificador del usuario autenticado y se le permitirá ingresar al chat.

```
18  $("#idCredenciales").click(function (e) {
19      e.preventDefault();
20      $.ajax({
21          url: "Chat.svc/Autenticar",
22          type: "POST",
23          data: '{"correo":"' + $("#inputEmail").val() + '","password":"' + $("#inputName").val() + '" }',
24          contentType: "application/json",
25          dataType: "json",
26          processData: true,
27          success: function (response) {
28              var foo = response;
29              if (foo.idAutenticado > 0) {
30                  $("#identificador").val(foo.idAutenticado);
31                  $("#esAutenticado").val(1);
32                  $chatbox.removeClass('chatbox--empty');
```

Código 2.35 Código de interacción con el método `Autenticar` del servicio `Chat.svc`

En el Código 2.36 se observa el código JavaScript con AJAX para interactuar con el método `NoAutenticar` del servicio web `Chat.svc`, este método permitirá que personas que no sean parte de la EPN utilicen el chatbot. La línea 37 indica que se ejecutará el código JavaScript cuando se presione en el botón `idSinCredenciales`.

La petición AJAX inicia en la línea 39, en la línea 40 se indica la URL del servicio, la línea 41 indica el tipo de petición HTTP. Las líneas 42 y 43 establecen el tipo de dato que se enviará y que se recibirá, en los dos casos es JSON. La línea 45 señala que se va a procesar los datos. Finalmente, las línea 46 a 50 se ejecutan solo si la petición fue realizada con éxito, si es así se almacenará el identificador del usuario no autenticado y se le permitirá al usuario acceder al chat.

```
37  $("#idSinCredenciales").click(function (e) {
38      e.preventDefault();
39      $.ajax({
40          url: "Chat.svc/NoAutenticar",
41          type: "POST",
42          contentType: "application/json",
43          dataType: "json",
44          processData: true,
45          success: function (response) {
46              var foo = response;
47              if (foo.idAutenticado > 0) {
48                  $("#identificador").val(foo.idAutenticado);
49                  $("#esAutenticado").val(0);
50                  $chatbox.removeClass('chatbox--empty');
```

Código 2.36 Código de interacción con el método `NoAutenticar` del servicio `Chat.svc`

En el Código 2.37 se tiene el código JavaScript con AJAX para interactuar con el método `FuncionChat` del servicio web `Chat.svc`. Este código se ejecutará cada vez que el usuario envíe un mensaje al chatbot. La petición AJAX inicia en la línea 20, en la línea 21 se establece la URL del servicio web, la línea 22 establece el tipo de solicitud HTTP, la línea 23 señala los datos que se enviarán (el mensaje que el usuario envió y su identificador). Las líneas 24 y 25 establecen el tipo de dato que se enviará y que se recibirá, en los dos casos es JSON. La línea 26 señala que se va a procesar los datos. Finalmente, las línea 28 y 29 se ejecutan solo si la petición fue realizada con éxito, si es así se mostrará la respuesta al usuario.

```
19  $("#btnChatear").click(function () {
20      $.ajax({
21          url: "Chat.svc/FuncionChat",
22          type: "POST",
23          data: '{"nombre":"' + $("#conversacion_pregunta").val() + '","identificador":"' + $("#identificador").val() + '"}',
24          contentType: "application/json",
25          dataType: "json",
26          processData: true,
27          success: function (response) {
28              var foo = response;
29              $("#conversacion_respuesta").val(foo.Respuesta);
```

Código 2.37 Código de interacción con el método `FuncionChat` del servicio `Chat.svc`

2.3.6 IMPLEMENTACIÓN DEL SKETCH DE LA INTERFAZ DEL CHATBOT

Para implementar la interfaz del chatbot, se descargaron las plantillas "*chat box Costumer care*" de Bootstrap desde la página web: `https://bootsnipp.com/snippets/eomDM`. Después, se modificó tanto el documento HTML como el JavaScript y el .CSS para adaptarlos conforme los requerimientos del chatbot que se está implementando.

La interfaz del chatbot está dividida en tres bloques: título, cuerpo y credenciales. El Código 2.38 muestra el primer bloque, en la línea 14 se define el inicio de esta sección lo que permitirá aplicar diferentes estilos y operaciones solo a esta sección. Aquí se especifican el nombre chatbot y el botón de minimizar. En la Figura 2.39 se observa gráficamente el título del chatbot.

```
15      <div class="chatbox__title" id="inicioChat">
16          <h5><a href="#">Chatbot</a></h5>
17          <button class="chatbox__title__tray">
18              <span></span>
19          </button>
20      </div>
```

Código 2.38 Documento HTML de la interfaz del chatbot

Figura 2.39 Pestaña colapsada del chatbot

El Código 2.39 muestra el bloque de credenciales, en la línea 28 se muestra un mensaje al usuario indicando las ventajas que tendrá al autenticarse. Las líneas 31 a 40 muestran los campos donde el usuario podrá ingresar su correo y contraseña. En las líneas 44 a 46 se establecen los botones que el usuario usará para ingresar al chat, ya sea con autenticación o sin autenticación. En la Figura 2.39 se observa gráficamente el bloque de credenciales del chatbot.

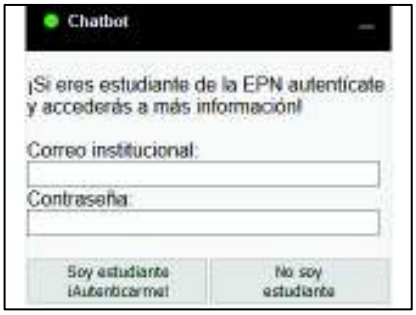


Figura 2.40 Interfaz de autenticación

```
27  <form class="chatbox__credentials">
28      <label class="introduccion"><br /> ¡Si eres estudiante de la EPN autentícate y accederás a más inf
29      <br />
30      <br />
31      <div class="form-group">
32          <label for="inputEmail">Correo institucional:</label>
33          <br />
34          <input type="email" class="form-control" id="inputEmail" required>
35      </div>
36      <div class="form-group">
37          <label for="inputName">Contraseña:</label>
38          <br />
39          <input type="password" class="form-control" id="inputName" required>
40      </div>
41      <div class="form-group">
42          <br />
43      </div>
44      <div class="form-group" id="botones">
45          <button type="button" id="idCredenciales" class="form">Soy estudiante ¡Autenticarme!</button>
46          <button type="button" id="idSinCredenciales" class="form">No soy<br /> estudiante</button>
```

Código 2.39 Documento HTML de la interfaz del chatbot

El Código 2.40 muestra el bloque denominado cuerpo del chatbot. En la línea 21 inicia este bloque, aquí se especifican los mensajes que el usuario y el chatbot van a intercambiar con las respectivas imágenes que los identifican. En la Figura 2.41 se aprecia esta interfaz.

```
21  <div id="body" class="chatbox__body">
22      <div id="primerChat" class="chatbox__body__message chatbox__body__message--left">
23          <img src="/chatbot.png" alt="Picture">
24          <p id="1Chatbot">Hola! Soy el chatbot de la carrera de Tecnologías de la Información ¿En qué te puedo ayudar?</p>
25      </div>
```

Código 2.40 Documento HTML de la interfaz del chatbot

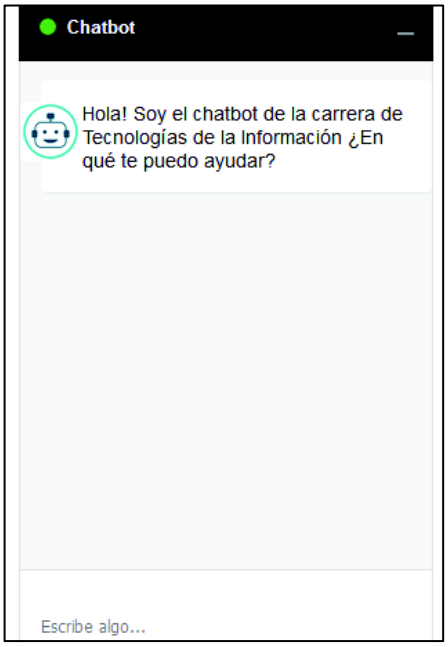


Figura 2.41 Interfaz del chat

En el Código 2.41 se aprecia el código JavaScript que permite anidar los mensajes enviados por el chatbot y por el usuario para presentarlos en forma de conversación. Las líneas 56 y 57 facilitan que el usuario envíe los mensajes al chatbot teclee `Enter`. Las líneas 58 a 66 realizan la petición AJAX al servicio web y obtienen una respuesta. Las líneas 67 a 70 crean un nuevo elemento HTML que contendrá la respuesta del chatbot, con la foto y estilo correspondiente. Las líneas 71 y 72 asignan los valores correspondientes a la estructura HTML creada previamente. Las líneas 73 a 76 añaden esta estructura HTML al código previo de la página HTML. Al realizar este proceso se podrá tener los mensajes que el chatbot y el usuario han intercambiado previamente.

```
55  function myFunction(e) {
56      tecla = (document.all) ? e.keyCode : e.which;
57      if (tecla == 13) {
58          $.ajax({
59              url: "Chat.svc/FuncionChat",
60              type: "POST",
61              data: '{"nombre":"' + $("#mensajeAE").val() + '","identificador":"' + $("#identifi
62              contentType: "application/json",
63              dataType: "json",
64              processData: true,
65              success: function (response) {
66                  var foo = response;
67                  var divChatbot = document.createElement("div");
68                  divChatbot.className = "chatbox__body__message chatbox__body__message--left";
69                  var p = document.createElement("p");
70                  var img = document.createElement("img");
71                  img.src = "/chatbot.png"
72                  p.innerHTML = foo.Respuesta;
73                  divChatbot.appendChild(img);
74                  divChatbot.appendChild(p);
75                  var capa = document.getElementById("body");
76                  capa.appendChild(divChatbot);
77              }
78          });
79          var div = document.createElement("div");
80          div.className = "chatbox__body__message chatbox__body__message--right";
81          var p = document.createElement("p");
82          var img = document.createElement("img");
83          img.src = "/persona.png"
84          p.innerHTML = $("#mensajeAE").val();
85          div.appendChild(img);
86          div.appendChild(p);
```

Código 2.41 Documento JavaScript de la interfaz del chatbot

En el ANEXO C se puede observar toda la solución de este Prototipo de Chatbot.

2.3.7 APLICACIÓN DE ADMINISTRACIÓN

Es una aplicación de escritorio que permitirá al chatbot proporcionar información verídica de acuerdo al periodo académico. El administrador podrá añadir periodos académicos permitiendo que el chatbot pueda ser usado en varios semestres y en cada periodo académico tendrá la posibilidad de gestionar los parámetros. La interfaz posee ocho pestañas, la primera pestaña es el inicio de la aplicación mientras que las demás

pestañas contienen todos los parámetros que podrán ser gestionados; estos parámetros han sido agrupados en categorías como Fechas, Docentes, Archivos, entre otros.

Esta aplicación interactúa con el servicio `localhost:54787/ServicioAdministrar` que a su vez gestiona la base de datos.

La Figura 2.42 muestra el inicio de la Aplicación de Administración, se puede ver en la parte superior las distintas ocho pestañas que agrupan los parámetros a gestionar, en el centro se seleccionará el periodo académico a gestionar y en la parte inferior se podrá añadir un nuevo periodo académico.

Al añadir un nuevo periodo académico las respuestas de los parámetros se copian del último periodo académico para facilitar al administrador la tarea de gestión, así solo modificará los valores que han cambiado.

En el ANEXO D se encuentra el código completo de esta Aplicación de Administración y en el ANEXO E se localiza su manual de usuario.

Figura 2.42 Interfaz de inicio de la Aplicación de Administración

A continuación, se indicarán dos ejemplos de pestañas de la aplicación de administración. La Figura 2.43 muestra la pestaña de Fechas de la aplicación, donde el

administrador seleccionará las fechas de inicio y fin de los parámetros con ayuda del control Calendario facilitando esta tarea.

Figura 2.43 Pestaña de Fechas de la Aplicación de Administración

La Figura 2.44 muestra la pestaña Docentes de la aplicación, aquí el administrador podrá crear, modificar o eliminar la información de un profesor.

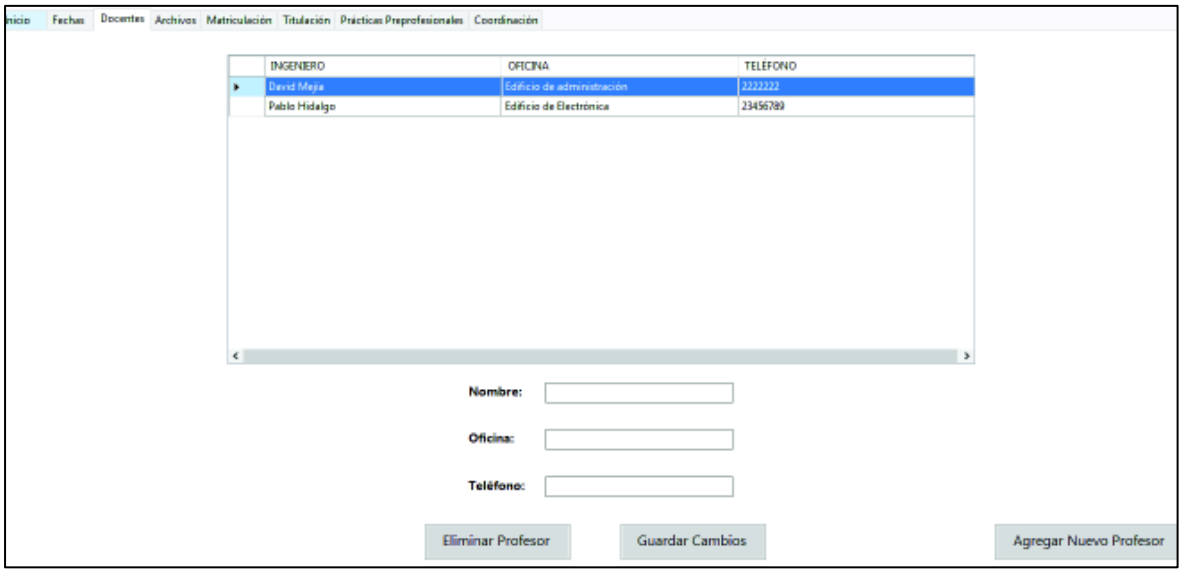


Figura 2.44 Pestaña de Docentes de la Aplicación de Administración

3. RESULTADOS Y DISCUSIÓN

En el presente capítulo se describen las pruebas realizadas a la aplicación de administración y al chatbot, se presentará los resultados obtenidos y con base en los errores encontrados se realizarán las correcciones.

Inicialmente, se presentan pruebas de integración para verificar el correcto funcionamiento de los distintos módulos del chatbot. Aquí se presentan las peticiones HTTP a los servicios WCF y la conexión a la base de datos. A continuación, se realizan pruebas de funcionalidad a la aplicación de administración donde se gestionarán diferentes parámetros del chatbot. Después, se ejecutará pruebas de funcionalidad al chatbot; se establece un modelo de entrevista y se selecciona a 20 personas quienes interactuarán con el chatbot. Finalmente, se presentan los errores corregidos con base en las entrevistas realizadas.

3.1 PRUEBAS DE INTEGRACIÓN DE MÓDULOS

Para las pruebas de integración de módulos se presentarán las peticiones HTTP a los diferentes servicios WCF y la conexión a la base de datos. El chatbot implementado en este Trabajo de Titulación consta de cuatro módulos: Módulo de Entendimiento del Lenguaje, Módulo de Seguimiento de Estado, Módulo Política de Diálogo y Módulo Generador de Lenguaje Natural.

El primer módulo llamado Entendimiento del Lenguaje está formado por dos servicios web. El servicio web `localhost:55803/Chat.svc` expone tres métodos, el primer método se llama `Autenticar` que valida las credenciales introducidas por los usuarios. En la Figura 3.1 se verifica la correcta ejecución de esta petición HTTP. Es una petición POST al método `Autenticar` cuya respuesta presenta un código 200 `OK` que indica éxito.

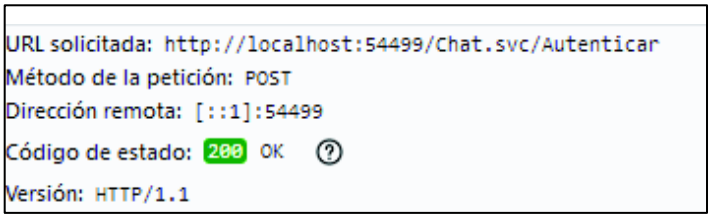


Figura 3.1 Petición HTTP al método `Autenticar`

En la Figura 3.2 se aprecian los datos en formato JSON enviados en la petición POST al método `Autenticar`, cabe indicar que este método recibe el correo electrónico y la contraseña del estudiante.

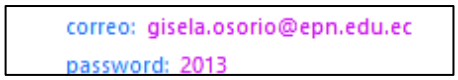


Figura 3.2 JSON enviado al método `Autenticar`

En la Figura 3.3 se muestra la respuesta del método `Autenticar` en formato JSON, el haber obtenido un número 1 significa que la autenticación fue correcta y el usuario accederá a información privada, caso contrario se recibía un 0.

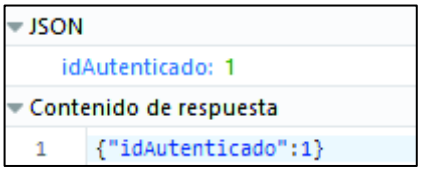


Figura 3.3 Respuesta del método `Autenticar`

El segundo método se llama `NoAutenticar` que asigna un identificador a los usuarios particulares que ingresan al chatbot sin autenticarse. En la Figura 3.4 se verifica la correcta ejecución de esta petición HTTP. Es una petición POST al método `NoAutenticar` cuya respuesta presenta un código 200 `OK` que indica éxito.

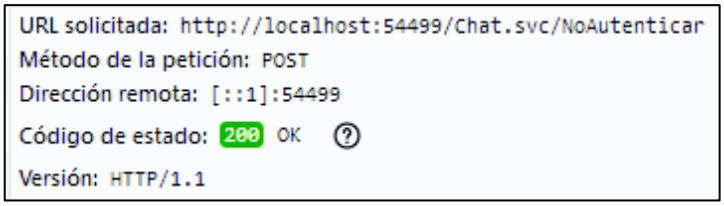


Figura 3.4 Petición HTTP al método `NoAutenticar`

La Figura 3.5 muestra la respuesta del método `NoAutenticar` en formato JSON y representa el identificador de un usuario no autenticado.

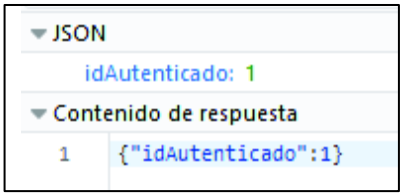


Figura 3.5 Respuesta del método `NoAutenticar`

El tercer método `FuncionChat` envía los mensajes introducidos por el usuario desde la ventana del chatbot hasta el servidor y retorna una respuesta. En la Figura 3.6 se aprecia la correcta ejecución de esta petición POST cuya respuesta presenta un código 200 `OK` que indica éxito.

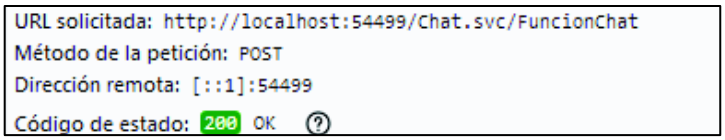


Figura 3.6 Petición HTTP al método `FuncionChat`

La Figura 3.7 muestra los datos en formato JSON enviados en la petición POST al método `Autenticar`. Se enviaron tres parámetros, el primero indica si el usuario se autenticó al momento de ingresar al chatbot, el segundo parámetro es el identificador del usuario y el tercer parámetro es el mensaje introducido por el usuario.

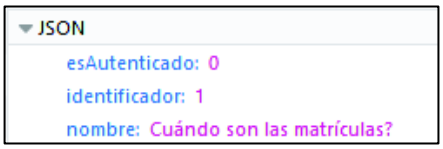


Figura 3.7 JSON enviado al método `FuncionChat`

La Figura 3.8 indica la respuesta en formato JSON proporcionada por el método `FuncionChat`. En la ventana del chatbot se desplegará esta respuesta. La acción de entregar una respuesta al usuario es parte del módulo Generador de Lenguaje Natural.

Figura 3.8 Respuesta del método `FuncionChat`

El segundo servicio web identificado por `localhost:55803/Grampal.svc/Postear` expone un solo método que realiza el análisis morfológico del mensaje enviado por el usuario. La Figura 3.9 muestra una petición GET a la herramienta Grampal, donde se envían tres parámetros, la etiqueta que define el tipo de análisis a realizar, un identificador del usuario y el mensaje que se desea analizar.

Figura 3.9 Petición GET al método `Postear`

La Figura 3.10 muestra la respuesta de la petición GET al método `Postear`, se aprecia la correcta ejecución de esta petición con un código 200 `OK` que indica éxito. Además, se obtiene la página HTML de la herramienta Grampal que en su cuerpo contiene el resultado del análisis morfológico.

```
HTTP/1.1 200 OK\r\n
  [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
      [HTTP/1.1 200 OK\r\n]
      [Severity level: Chat]
      [Group: Sequence]
    Response Version: HTTP/1.1
    Status Code: 200
    [Status Code Description: OK]
    Response Phrase: OK
  Date: Mon, 03 Jun 2019 22:52:45 GMT\r\n
  Server: Apache/2.2.16 (Debian)\r\n
  Set-Cookie: CGISESSID=ffe10941abe42b63786f9b9070fcd28c; path=/; expires=Mon, 03-Jun-2019 22:53:45 GMT\r\n
  Vary: Accept-Encoding\r\n
  Content-Encoding: gzip\r\n
  Content-Length: 1382\r\n
  Keep-Alive: timeout=15, max=99\r\n
  Connection: Keep-Alive\r\n
  Content-Type: text/html; charset=utf-8\r\n
  \r\n
  [HTTP response 2/2]
  [Time since request: 0.460361000 seconds]
  [Prev request in frame: 203]
  [Prev response in frame: 208]
  [Request in frame: 210]
  Content-encoded entity body (gzip): 1382 bytes -> 3492 bytes
  File Data: 3492 bytes
Line-based text data: text/html (119 lines)
  <html xmlns="http://www.w3.org/TR/REC-html40">\n
  <head>\n
  <title>Grampal </title>\n
  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">\n
  <meta name="Content-Language" content="EN">\n
  <meta name="author" content="jmguirao@ugr.es">\n
```

Figura 3.10 Respuesta de la petición GET al método `Postear`

El módulo Política de Diálogo permite seleccionar la respuesta más adecuada conforme el mensaje enviado por el usuario, este módulo interactúa con la base de datos. En la Figura 3.11 se aprecia la conexión con la base de datos `pruebaBase2` desde el Explorador de servidores en Visual Studio.

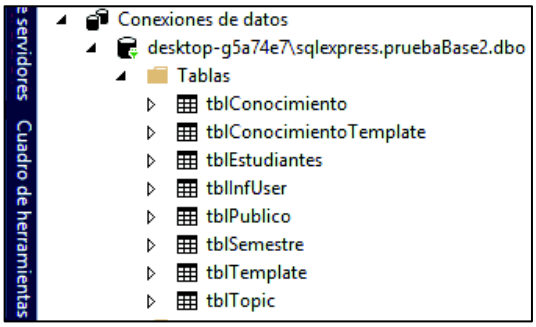
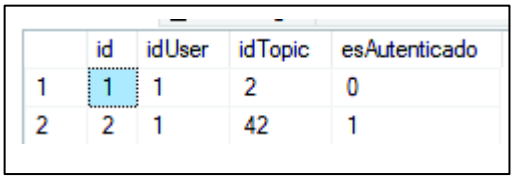


Figura 3.11 Conexión a la base de datos desde Visual Studio

El módulo Seguimiento de Estado guarda el registro de la conversación; para esto almacena en la tabla `tblInfUser` el identificador del usuario, el tema de la conversación y un identificador que indica si el usuario se autenticó.

En la Figura 3.12 se muestran los datos de un usuario autenticado que realizó dos preguntas al chatbot que están almacenados en la tabla `tblInfUser`.



	id	idUser	idTopic	esAutenticado
1	1	1	2	0
2	2	1	42	1

Figura 3.12 Tabla `tblInfUser`

3.2 PRUEBAS DE FUNCIONALIDAD

En este trabajo de titulación se desarrolló un prototipo de chatbot que responde las preguntas frecuentes de la carrera de Tecnologías de la Información y también se desarrolló una aplicación de administración para gestionar las respuestas de este chatbot.

A continuación, se describen las pruebas de funcionalidad de la aplicación de administración y del chatbot.

3.3.1 PRUEBAS DE FUNCIONALIDAD DE LA APLICACIÓN DE ADMINISTRACIÓN

La aplicación de administración es una interfaz gráfica que permite gestionar las respuestas del chatbot. Esta aplicación de escritorio consume los métodos del servicio `localhost:54787/Service.svc` que permiten gestionar la base de datos.

Gracias a la aplicación de administración el chatbot podrá ser usado en varios periodos académicos y brindará información actualizada.

Esta aplicación consta de varias pestañas, cada pestaña agrupa parámetros a gestionar.

Inicialmente el chatbot proveerá información referente al semestre 2019-A; por esto al ingresar por primera vez a la aplicación de administración, en la pestaña `Inicio` se mostrará únicamente el semestre 2019-A y en las siguientes pestañas se gestionará la información del chatbot referente a este semestre.

En la Figura 3.13 se aprecia la pestaña `Inicio`, en (1) se observa solo el semestre 2019-A, además en la parte superior de esta ventana se observan las diferentes pestañas que conforman esta aplicación.

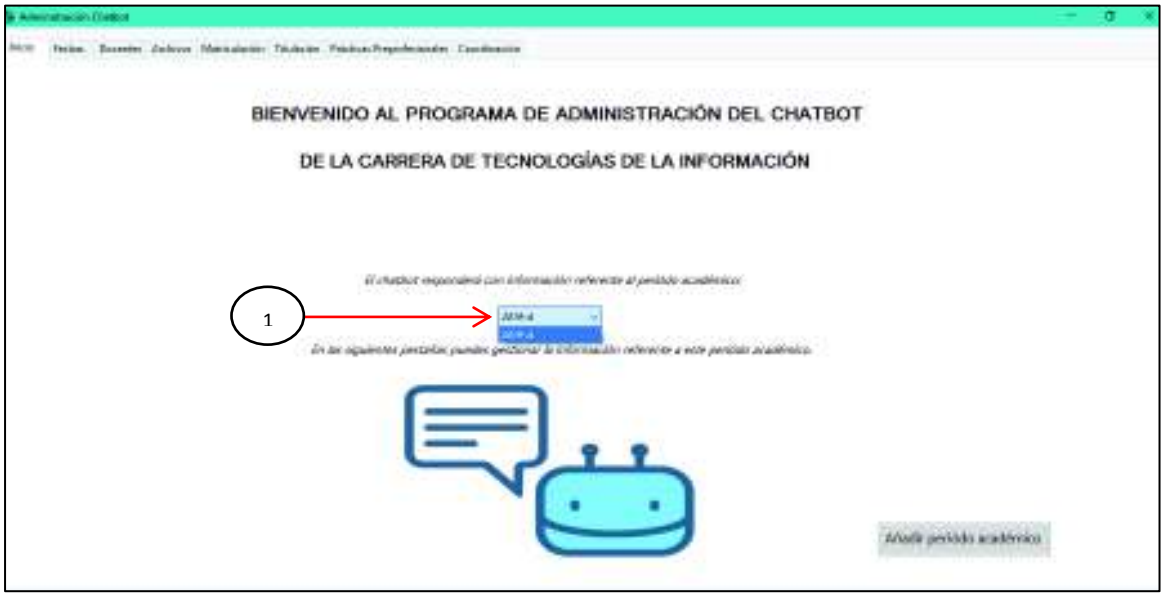


Figura 3.13 Pestaña de `Inicio` de la Aplicación de Administración

La Figura 3.14 presenta el comportamiento de la aplicación al añadir un nuevo periodo académico. Por ejemplo si termina el semestre 2019-A, el administrador seleccionará el botón `Añadir periodo académico` (1) y creará el semestre 2019-B. Cuando se crea un nuevo semestre, se copia la información del último periodo académico guardado evitando que el administrador tenga que añadir la información del chatbot desde cero.

Figura 3.14 Añadir un nuevo periodo académico

Cuando existen varios periodos académicos en la pestaña `Inicio` tal como se aprecia en la Figura 3.15, el administrador podrá seleccionar un semestre de la lista y gestionarlo. En (1) de la Figura 3.15 se ha seleccionado el semestre 2019-B, esto quiere decir que se gestionará la información referente a este periodo académico y además el chatbot responderá con esta información.

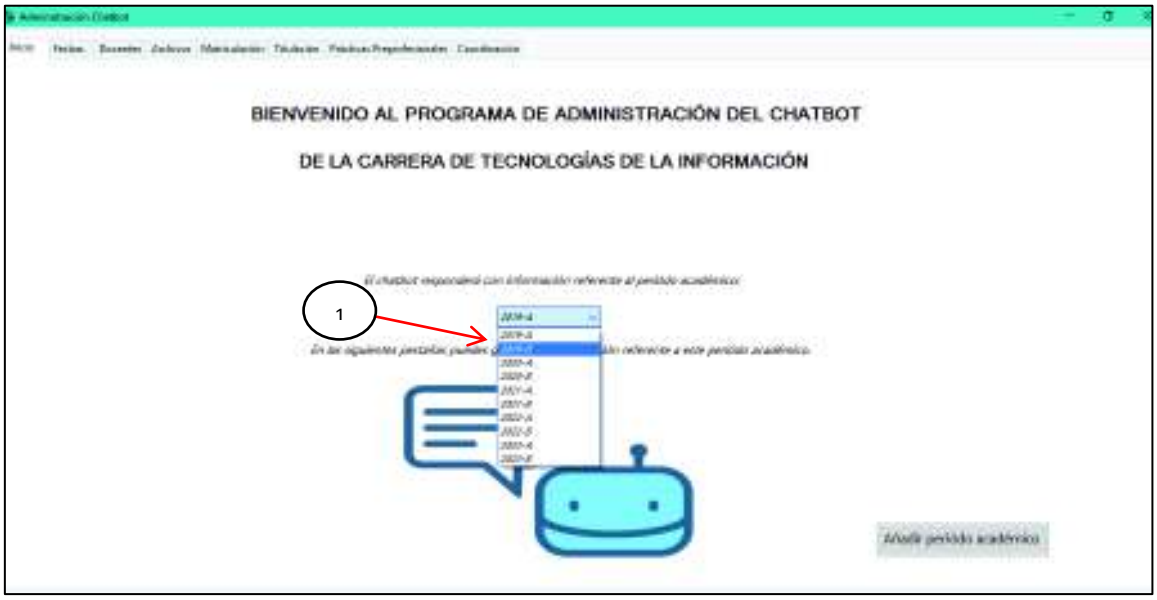


Figura 3.15 Selección del periodo académico

El chatbot implementado proveerá información desde el periodo académico 2019-A hasta el periodo 2023-B. La Figura 3.16 presenta el error generado al tratar de ingresar el periodo académico 2024-A.

Figura 3.16 Error al ingresar nuevo periodo académico

La Figura 3.17 muestra la pestaña `Fechas` donde se puede gestionar datos como inicio y fin de semestre, fechas de matrículas ordinarias, extraordinarias, reinscripciones, entre otros. Además, se aprecia el mensaje de éxito al gestionar estos datos de forma correcta y guardarlos.

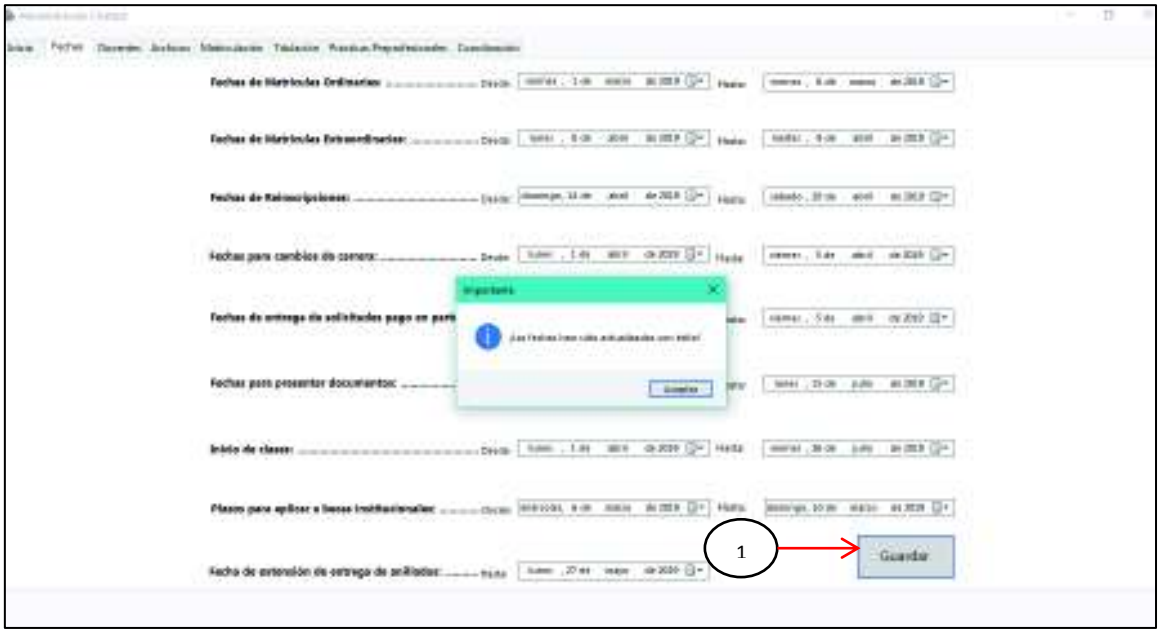


Figura 3.17 Pestaña de `Fechas`

La Figura 3.18 muestra el mensaje de error presentado al momento de guardar datos no coherentes o inválidos en la pestaña `Fechas`.

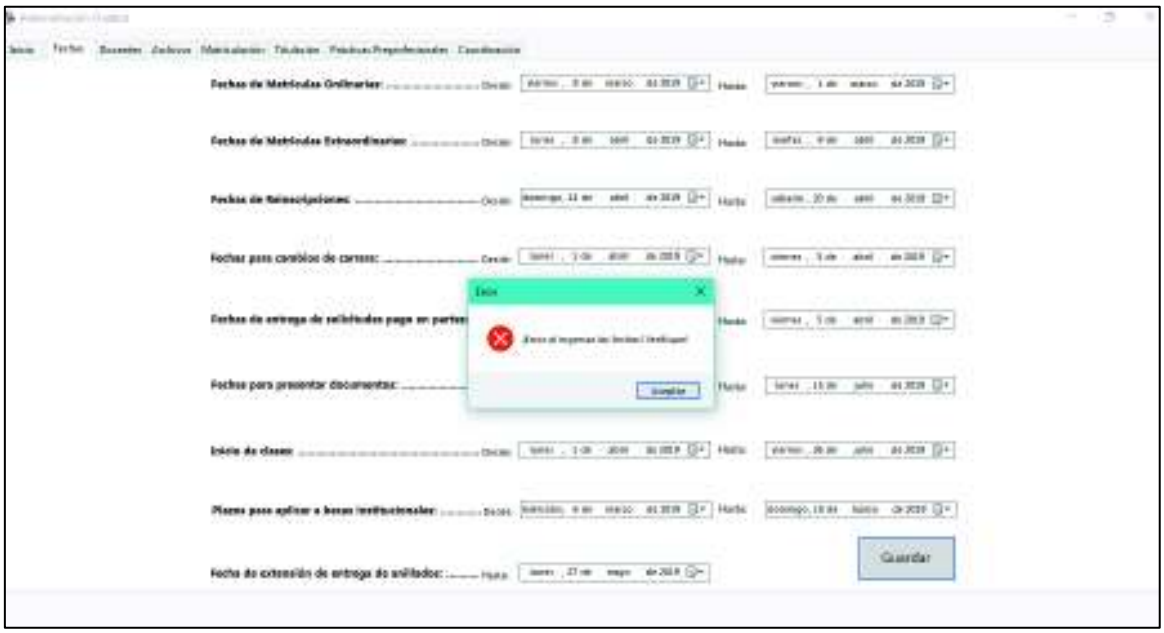


Figura 3.18 Error en la pestaña `Fechas`

La pestaña `Docentes`, permite gestionar datos como nombre, oficina y número de teléfono de los profesores de la carrera de Tecnologías de la Información. En la Figura 3.19 se aprecia el mensaje de éxito al haber ingresado correctamente la información de un nuevo profesor. Además, se pueden editar los datos o eliminar a un profesor conforme requiera el administrador.

Figura 3.19 Insertar nuevo Profesor

La pestaña `Archivos` permite gestionar los documentos que se envían al usuario cuando pregunta sobre el horario de materias, la malla curricular, el horario de supletorios y los seminarios. En la Figura 3.20 se aprecian las rutas de los archivos cuyo contenido se enviará a los usuarios, se puede actualizar estos documentos por medio del botón `Cambiar Archivo`.

Figura 3.20 Gestionar archivos

La pestaña `Matriculación` permite gestionar el proceso de consulta de turnos de matrícula y los tipos de becas institucionales. En la Figura 3.21 se puede observar los parámetros antes mencionados y el botón que permite guardar los cambios realizados, es importante mencionar que estas respuestas pueden tener una longitud máxima de 400 caracteres.

Figura 3.21 Pestaña `Matriculación`

La pestaña `Titulación` permite gestionar 10 parámetros referentes al proceso de titulación. En la Figura 3.22 se puede observar estos parámetros y el botón que permite guardar los cambios realizados.

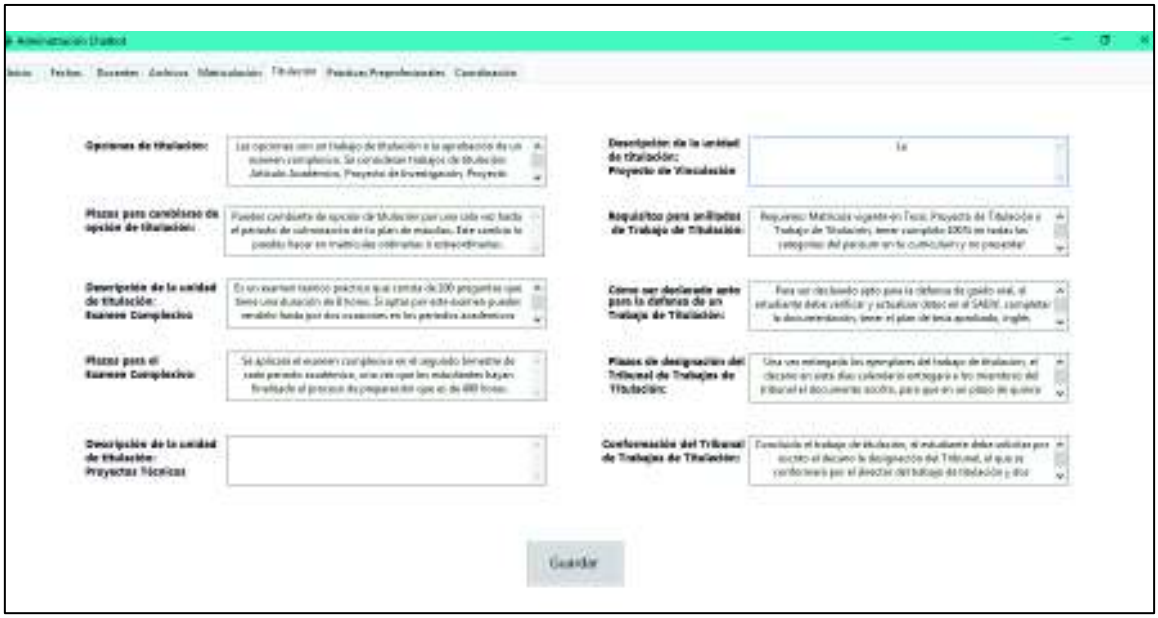


Figura 3.22 Pestaña `Titulación`

La pestaña `Practicas Preprofesionales` permite gestionar el proceso de prácticas preprofesionales y el listado de las empresas que tienen convenio con la facultad. En la Figura 3.23 se puede observar estos parámetros y el botón que permite actualizar la información.

Figura 3.23 Pestaña `Prácticas Preprofesionales`

La pestaña `Coordinación` permite gestionar el horario de atención de la coordinación de la carrera, los datos del coordinador y del respectivo secretario. En la Figura 3.24 se puede observar estos parámetros y el botón que permite guardar los cambios.

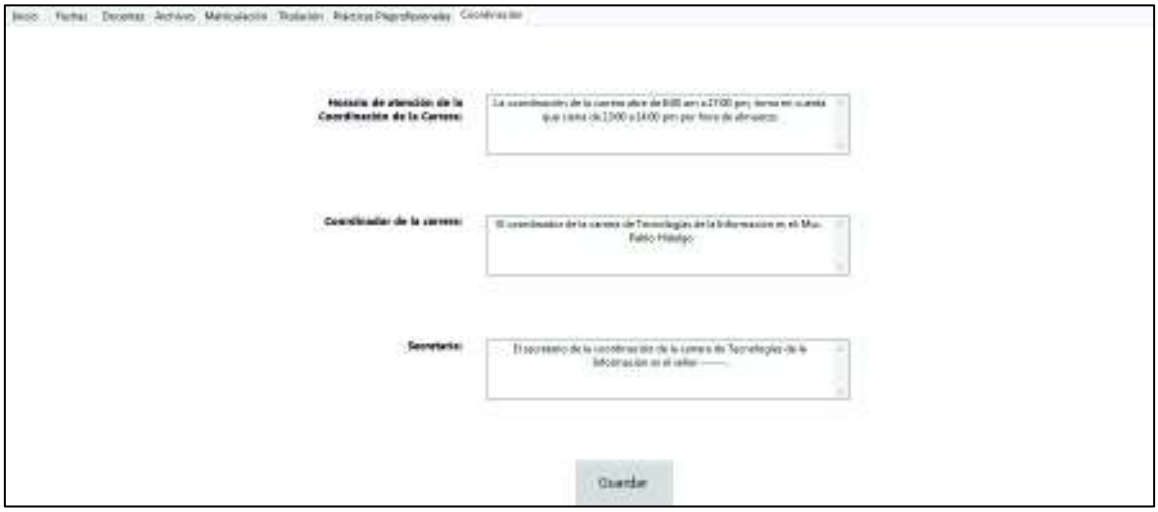


Figura 3.24 Pestaña `Coordinación`

3.3.2 PRUEBAS DE FUNCIONALIDAD DEL CHATBOT

El chatbot implementado en este Trabajo de Titulación puede ser añadido a cualquier página web, en este caso se lo añadió a una página web en blanco. Para realizar la

prueba de funcionalidad se abrirá cualquier navegador que tenga JavaScript habilitado y se dirigirá a la siguiente URL: `http://localhost:54499/VentanaChat.html`. En la Figura 3.25 se aprecia la página web `VentanaChat.html` que contiene únicamente la pestaña del chatbot en la parte inferior derecha, esta pestaña al ser seleccionada se maximizará.

Figura 3.25 Chatbot en una página web

El chatbot puede ser utilizado por estudiantes de la EPN y por el público en general, debido a esto en el ingreso se presentan dos opciones: autenticarse usando el correo institucional y contraseña o acceder directamente al chat, cabe recalcar que al autenticarse el usuario puede acceder a más información. En la Figura 3.26 se aprecia el ingreso al chatbot usando autenticación. En la Figura 3.27 se aprecia la alerta que se presenta cuando el usuario desea autenticarse pero no ingresa de forma correcta sus credenciales.

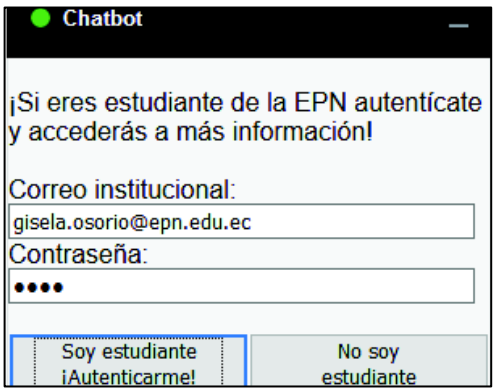


Figura 3.26 Ingreso al chatbot con autenticación

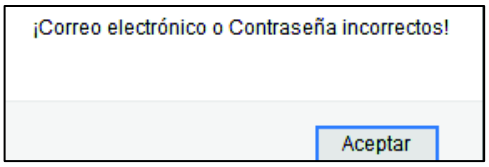


Figura 3.27 Ingreso al chatbot con autenticación incorrecta

Una vez que el usuario se ha autenticado podrá enviar mensajes al chatbot. En la Figura 3.28 se observa un ejemplo de conversación entre un usuario autenticado y el chatbot.

Figura 3.28 Ejemplo de una conversación con el chatbot de un usuario autenticado

Cuando un usuario se autentica puede acceder a los datos de los profesores de la carrera de Tecnologías de la Información, como su número de teléfono y oficina. La Figura 3.29 muestra un ejemplo de conversación cuando un usuario autenticado le pregunta al chatbot acerca de un profesor, se puede observar que se le proporciona la información sin ningún tipo de restricción.

Figura 3.29 Usuario autenticado pregunta acerca de un profesor

En el momento que un usuario pregunta por temas como: malla curricular, horario de materias, horario de supletorios, horario de seminarios; el chatbot le propone al usuario enviar la información por medio de correo electrónico. La Figura 3.30 muestra un ejemplo de conversación donde un usuario requiere información del horario de supletorios, el chatbot le propone enviar la información al correo institucional, el usuario acepta y el chatbot ejecuta la acción de enviar el correo electrónico. La Figura 3.31 presenta el correo electrónico enviado por el chatbot al usuario, se puede observar que tiene un documento adjunto con la información solicitada.

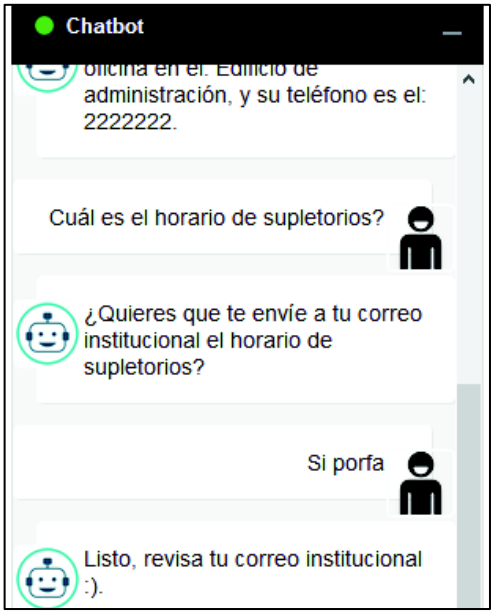


Figura 3.30 Usuario autenticado pregunta acerca del horario de supletorios

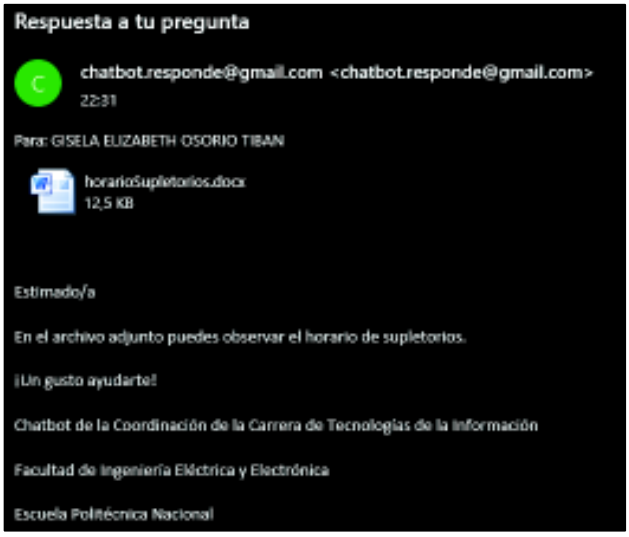


Figura 3.31 Correo electrónico enviado por el chatbot con la información del horario de supletorios

Anteriormente, se presentó un ejemplo de conversación de un usuario autenticado. Ahora se presentará un ejemplo de conversación de un usuario no autenticado para apreciar las diferencias, el usuario realizará las mismas preguntas.

En la Figura 3.32 se aprecia el ingreso al chatbot únicamente seleccionando el botón cuya leyenda dice No soy estudiante.

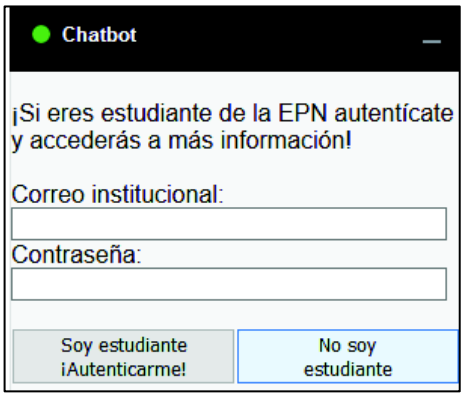


Figura 3.32 Ingreso al chatbot sin autenticación

La Figura 3.33 presenta una conversación de un usuario no autenticado, se puede apreciar que la información que el chatbot envió anteriormente al usuario autenticado es la misma que envió al usuario no autenticado. Información como fechas de matrículas no tiene restricción y puede ser accedida por estudiantes de la EPN como también por el público en general.

Figura 3.33 Ejemplo de una conversación con el chatbot de un usuario no autenticado

En la Figura 3.34 se aprecia que el usuario no autenticado solicita información acerca de un profesor, como se había mencionado anteriormente esta información es privada y no puede ser compartida con usuarios no autenticados.

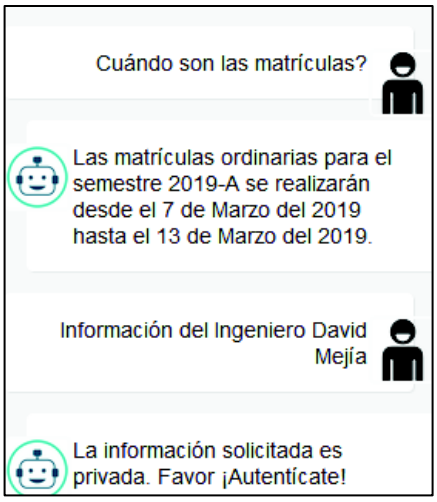


Figura 3.34 Usuario no autenticado pregunta acerca de un profesor

La Figura 3.35 muestra un ejemplo en el que el usuario no autenticado solicita información acerca del horario de supletorios. Esta información no es privada y si puede ser compartida con el público en general, pero el chatbot no tiene almacenado la dirección de correo electrónico del usuario, entonces le solicita al usuario que indique un correo electrónico a donde enviarle la información. La Figura 3.36 presenta el correo electrónico enviado por el chatbot a la dirección de correo proporcionada por el usuario durante la conversación.

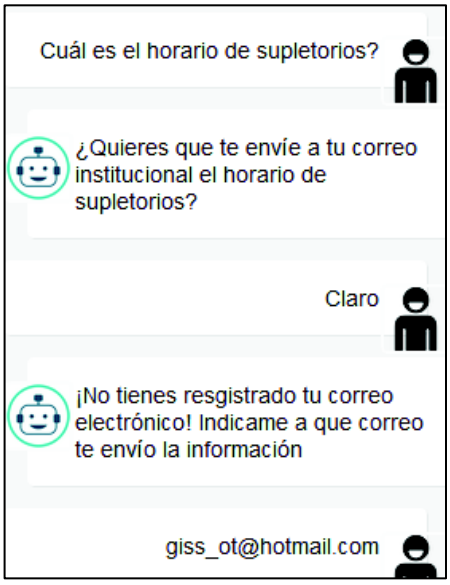


Figura 3.35 Usuario no autenticado pregunta acerca del horario de supletorios

Figura 3.36 Correo electrónico enviado por el chatbot con la información del horario de supletorios

En general, las respuestas del chatbot se categorizaron en tres grupos. El primer grupo llamado “Éxito” agrupa las respuestas que proporcionaron la información solicitada por el usuario, estas respuestas se generan cuando el usuario envía textos que se encuentran en la base de datos. En la Figura 3.37 se aprecia un ejemplo de una respuesta del grupo “Éxito”.

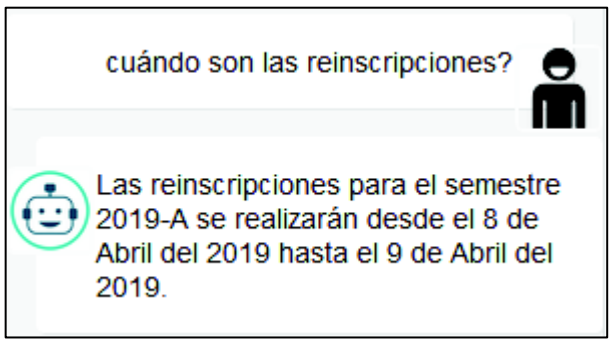


Figura 3.37 Respuesta del grupo "Éxito"

El segundo grupo de respuestas se llama “Comodín” este tipo de respuesta se genera cuando el usuario envía un mensaje que no está en la base de datos y al realizar el análisis morfológico se determina que tiene palabras clave, cada palabra clave está asociada a una respuesta que ayuda al usuario a obtener información. En la Figura 3.38 se observa un ejemplo, el usuario menciona “¿Cómo me cambiaría de carrera?” y el chatbot ayuda al usuario a obtener la información requerida.

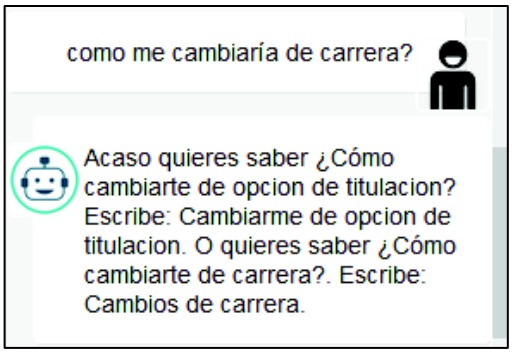


Figura 3.38 Respuesta del grupo "Comodín"

El tercer grupo de respuestas llamadas “Error” se genera cuando el usuario envía un mensaje que no está en la base de datos y al realizar el análisis morfológico no se encuentra ninguna palabra clave. En la Figura 3.39 se aprecia que el usuario envió un mensaje que el chatbot no lo encontró en la base de datos ni en el análisis morfológico por lo tanto envió una respuesta por defecto.

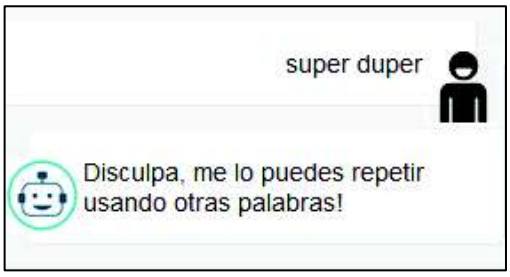


Figura 3.39 Respuesta del grupo "Error"

3.3.3 ENTREVISTAS

Mediante entrevistas a veinte personas se identificarán las debilidades y los errores presentes en el chatbot con el propósito de corregirlos. En la etapa de Análisis se definió treinta temas específicos sobre los que el chatbot brinda información, estos temas forman parte de esta entrevista.

El modelo de la entrevista se encuentra en el ANEXO F.

La dinámica de la entrevista consistió en indicarle al usuario los temas de las preguntas a formular. El usuario formuló estas preguntas, las envió al chatbot y recibió una respuesta.

A continuación se presentarán los resultados de las entrevistas.

La primera pregunta menciona ¿El chatbot te contestó al preguntarle sobre el lugar dentro del SAEW donde consultar el turno de la matrícula? En la Figura 3.40 se aprecian los resultados obtenidos.

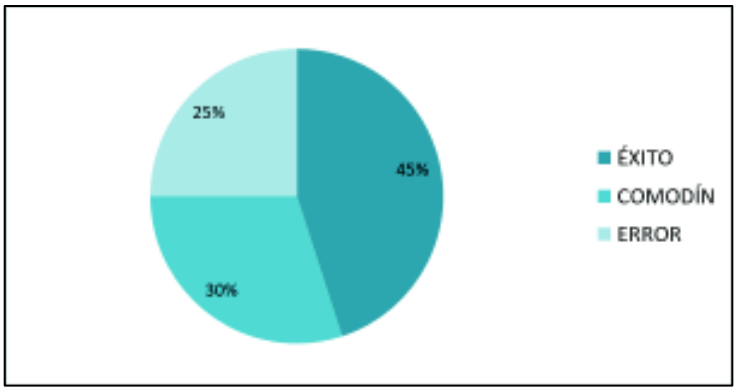


Figura 3.40 Resultados de la pregunta 1

La segunda pregunta indica ¿El chatbot te respondió al preguntarle sobre las fechas de matrículas ordinarias? En la Figura 3.41 se aprecian los resultados obtenidos.

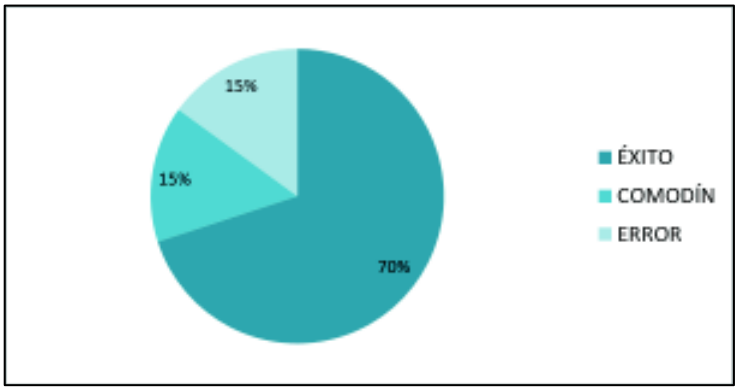


Figura 3.41 Resultados de la pregunta 2

La tercera pregunta establece ¿El chatbot te respondió al preguntarle sobre las fechas de matrículas extraordinarias? En la Figura 3.42 se aprecian los resultados obtenidos.

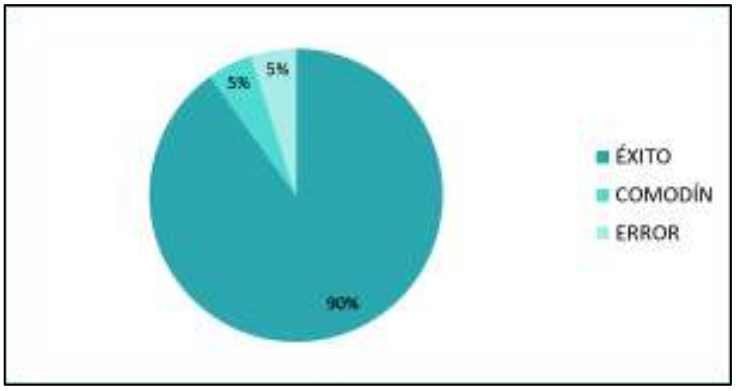


Figura 3.42 Resultados de la pregunta 3

La cuarta pregunta señala ¿El chatbot te contestó al preguntarle sobre las fechas de reinscripciones? En la Figura 3.43 se aprecian los resultados obtenidos.

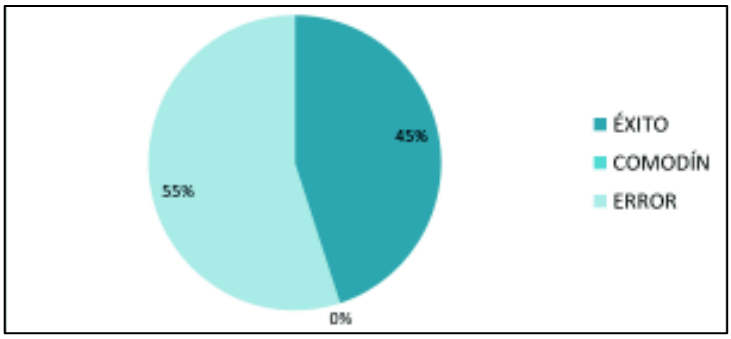


Figura 3.43 Resultados de la pregunta 4

La quinta pregunta menciona ¿El chatbot te respondió al preguntarle sobre el horario de materias? En la Figura 3.44 se aprecian los resultados obtenidos.

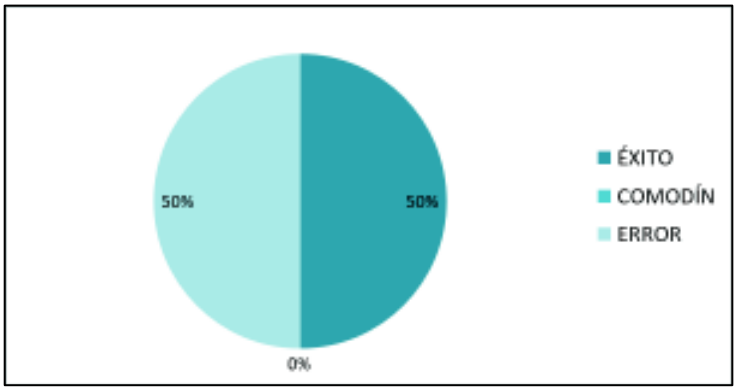


Figura 3.44 Resultados de la pregunta 5

La sexta pregunta establece ¿El chatbot te respondió al preguntarle sobre cuáles son las opciones de titulación? En la Figura 3.45 se aprecian los resultados obtenidos.

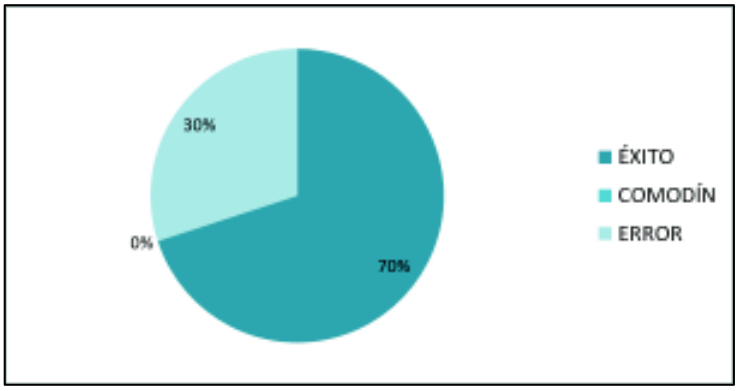


Figura 3.45 Resultados de la pregunta 6

La séptima pregunta indica ¿El chatbot te contestó al preguntarle sobre los plazos para cambiarse de opción de titulación en Unidad de Titulación? En la Figura 3.46 se aprecian los resultados obtenidos.

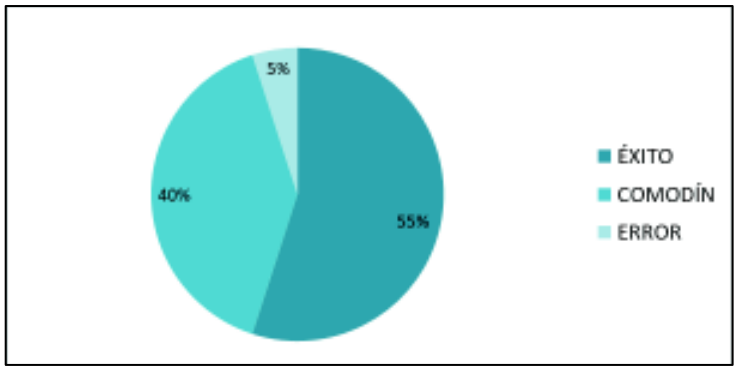


Figura 3.46 Resultados de la pregunta 7

La octava pregunta señala ¿El chatbot te respondió al preguntarle sobre la unidad de titulación: Examen Complexivo? En la Figura 3.47 se aprecian los resultados obtenidos.

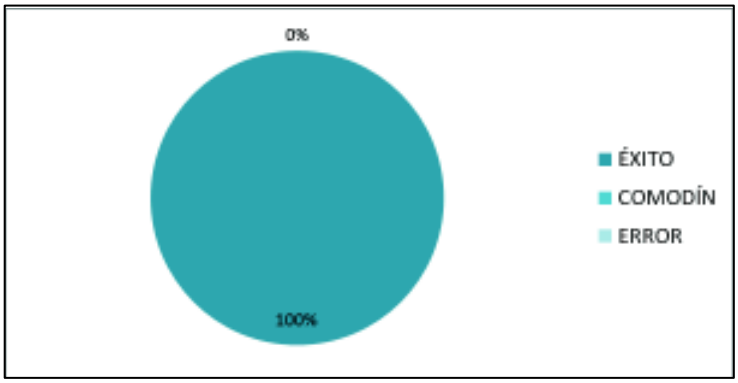


Figura 3.47 Resultados de la pregunta 8

La novena pregunta menciona ¿El chatbot te contestó al preguntarle sobre los plazos para el Examen Complexivo? En la Figura 3.48 se aprecian los resultados obtenidos.

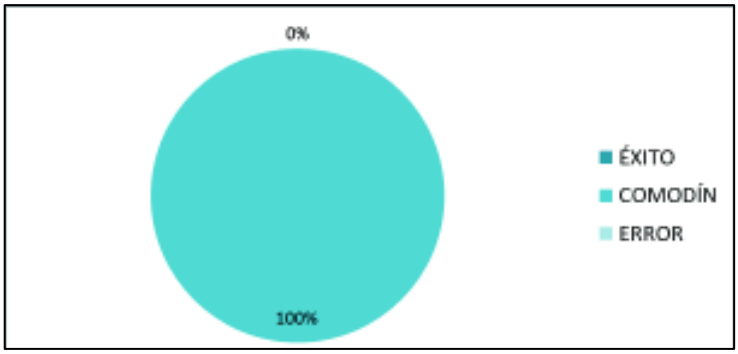


Figura 3.48 Resultados de la pregunta 9

La décima pregunta establece ¿El chatbot te respondió al preguntarle al chatbot sobre la unidad de titulación: Proyectos Técnicos? En la Figura 3.49 se aprecian los resultados obtenidos.

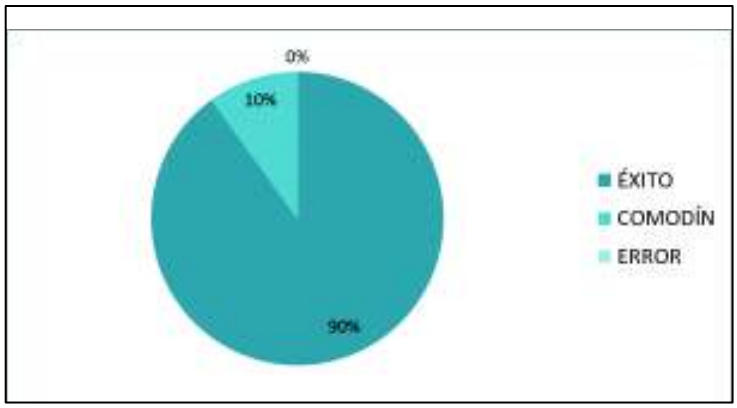


Figura 3.49 Resultados de la pregunta 10

La pregunta once indica ¿El chatbot te respondió al preguntarle sobre la unidad de titulación: Proyecto de Investigación? En la Figura 3.50 se aprecian los resultados obtenidos.

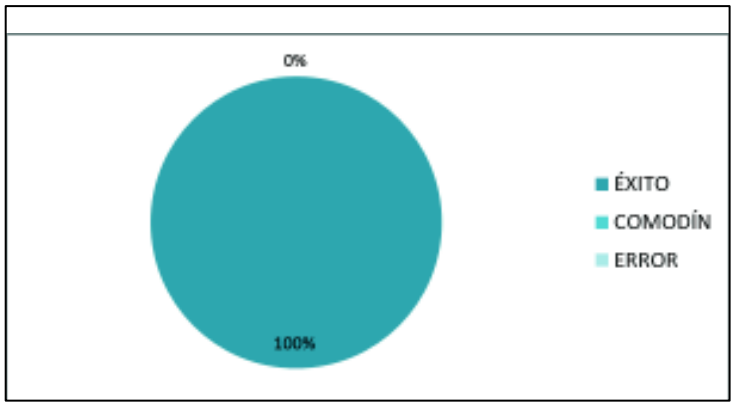


Figura 3.50 Resultados de la pregunta 11

La pregunta doce señala ¿El chatbot te respondió al preguntarle sobre los requisitos para presentar anillados de Proyecto de Titulación? En la Figura 3.51 se aprecian los resultados obtenidos.

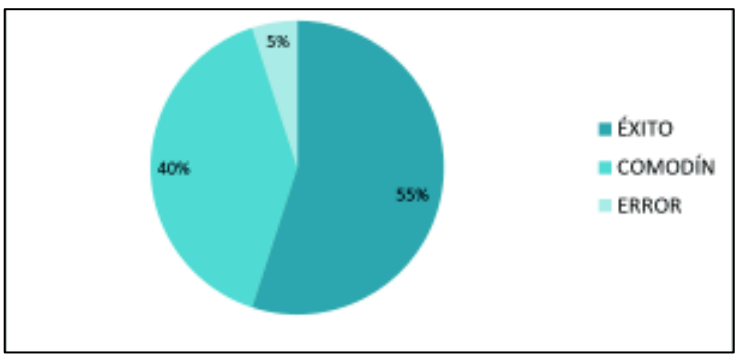


Figura 3.51 Resultados de la pregunta 12

La pregunta trece menciona ¿El chatbot te contestó al preguntarle sobre cómo declararse apto para la defensa de grado oral de un Trabajo de Titulación? En la Figura 3.52 se aprecian los resultados obtenidos.

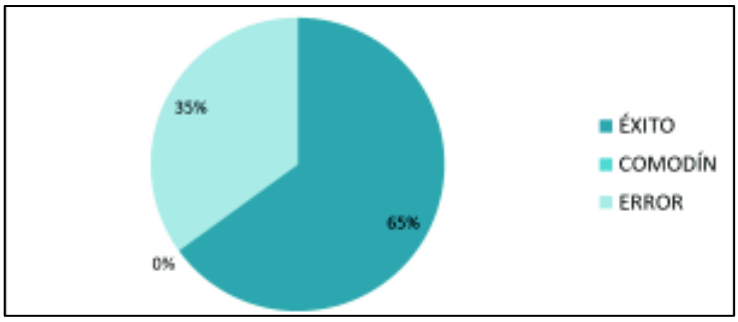


Figura 3.52 Resultados de la pregunta 13

La pregunta catorce establece ¿El chatbot te respondió al preguntarle sobre los plazos de designación del Tribunal de Trabajos de Titulación? En la Figura 3.53 se aprecian los resultados obtenidos.

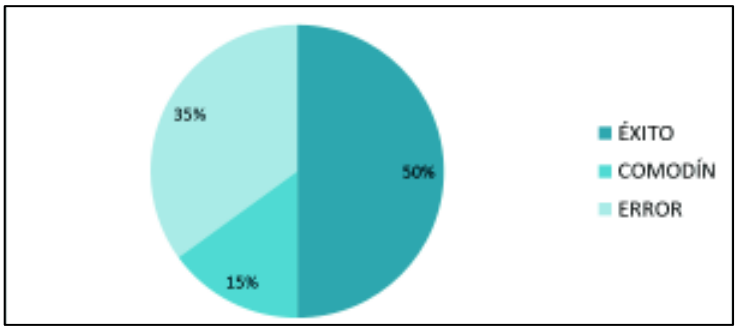


Figura 3.53 Resultados de la pregunta 14

La pregunta quince indica ¿El chatbot te contestó al preguntarle sobre la conformación del Tribunal de Trabajos de Titulación? En la Figura 3.54 se aprecian los resultados obtenidos.

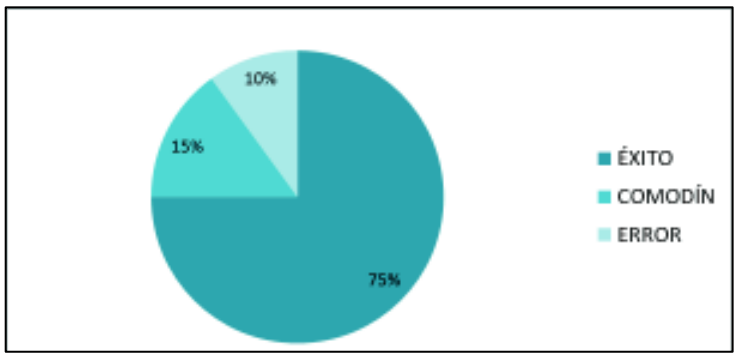


Figura 3.54 Resultados de la pregunta 15

La pregunta dieciséis establece ¿El chatbot te contestó al preguntarle sobre las fechas de extensión de entrega de anillados? En la Figura 3.55 se aprecian los resultados obtenidos.

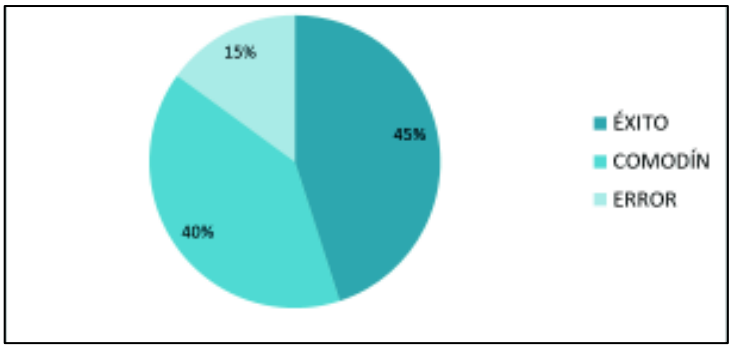


Figura 3.55 Resultados de la pregunta 16

La pregunta diecisiete señala ¿El chatbot te respondió al preguntarle sobre el proceso de prácticas preprofesionales? En la Figura 3.56 se aprecian los resultados obtenidos.

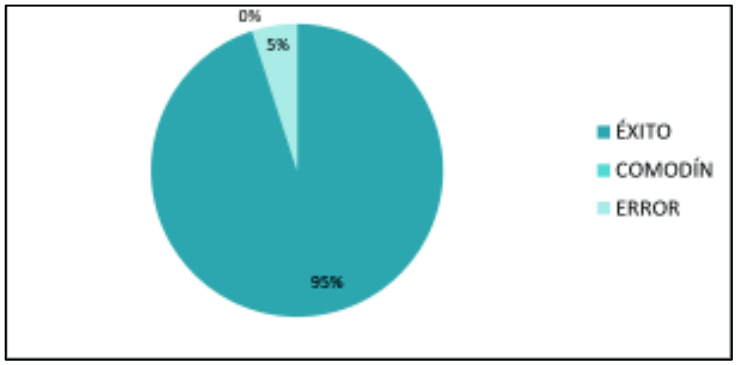


Figura 3.56 Resultados de la pregunta 17

La pregunta dieciocho menciona ¿El chatbot te contestó al preguntarle al chatbot sobre ofertas de prácticas preprofesionales? En la Figura 3.57 se aprecian los resultados obtenidos.

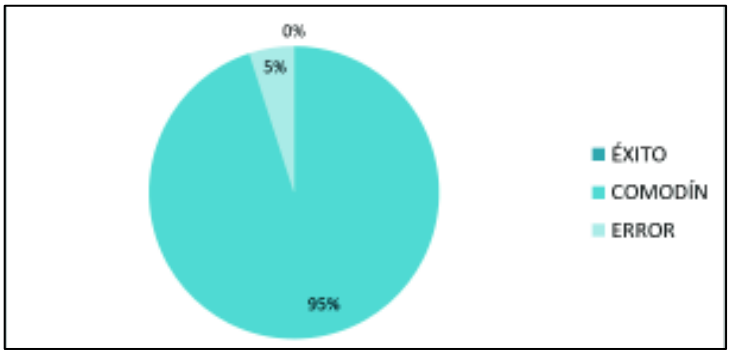


Figura 3.57 Resultados de la pregunta 18

La pregunta diecinueve establece ¿El chatbot te envió una respuesta válida al preguntarle sobre la información de contacto de un docente de la carrera de tecnologías de la información? En la Figura 3.58 se aprecian los resultados obtenidos.

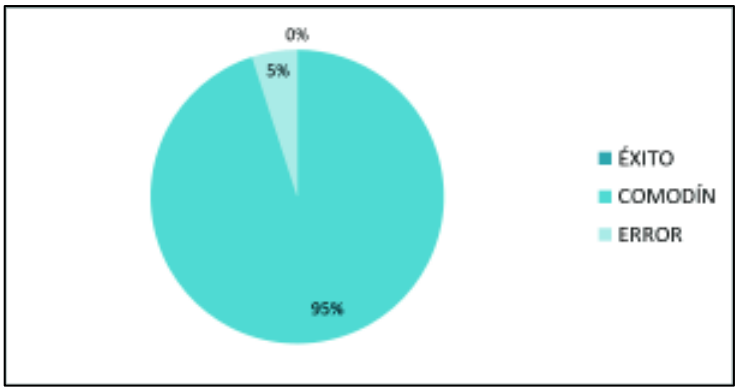


Figura 3.58 Resultados de la pregunta 19

La pregunta veinte indica ¿El chatbot te respondió al preguntarle sobre las fechas de inicio y fin de semestre? En la Figura 3.59 se aprecian los resultados obtenidos.

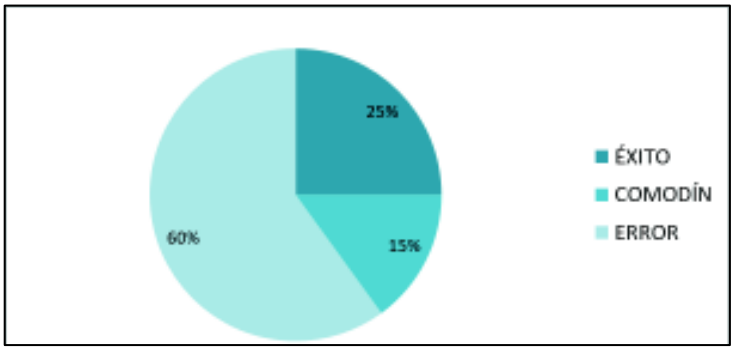


Figura 3.59 Resultados de la pregunta 20

La pregunta veinte y uno señala ¿El chatbot te respondió al preguntarle sobre las fechas para cambios de carrera? En la Figura 3.60 se aprecian los resultados obtenidos.

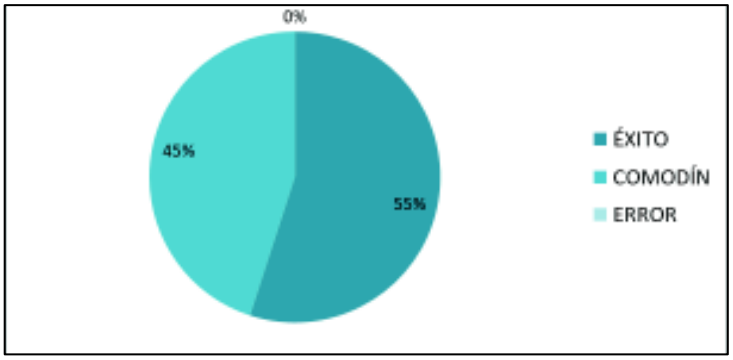


Figura 3.60 Resultados de la pregunta 21

La pregunta veinte y dos menciona ¿El chatbot te contestó al preguntarle sobre las fechas de entrega de solicitud para pago en partes? En la Figura 3.61 se aprecian los resultados obtenidos.

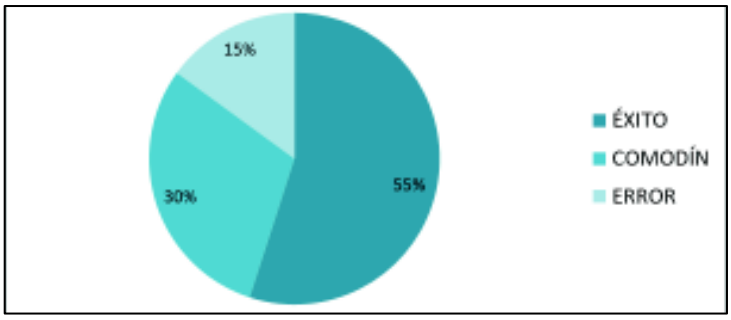


Figura 3.61 Resultados de la pregunta 22

La pregunta veinte y tres establece ¿El chatbot te respondió al preguntarle sobre las fechas para presentar documentos? En la Figura 3.62 se aprecian los resultados obtenidos.

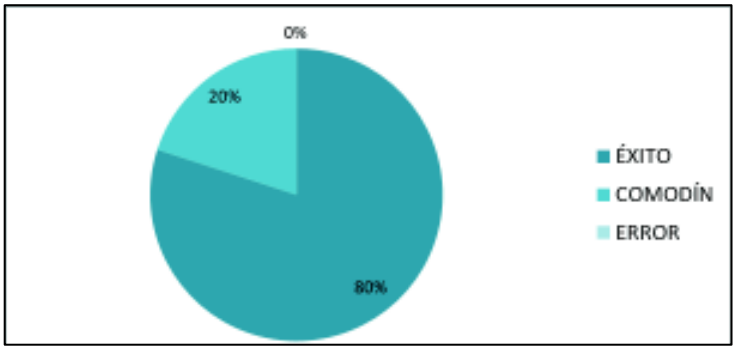


Figura 3.62 Resultados de la pregunta 23

La pregunta veinte y cuatro indica ¿El chatbot te respondió al preguntarle sobre el horario de atención de la coordinación de la carrera? En la Figura 3.63 se aprecian los resultados obtenidos.

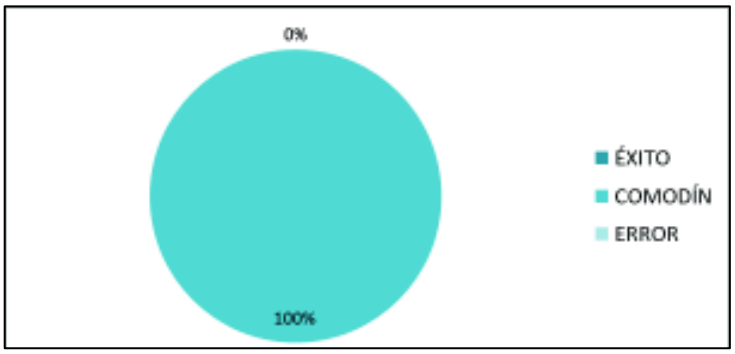


Figura 3.63 Resultados de la pregunta 24

La pregunta veinte y cinco señala ¿El chatbot te contestó al preguntarle sobre la malla curricular de la carrera? En la Figura 3.64 se aprecian los resultados obtenidos.

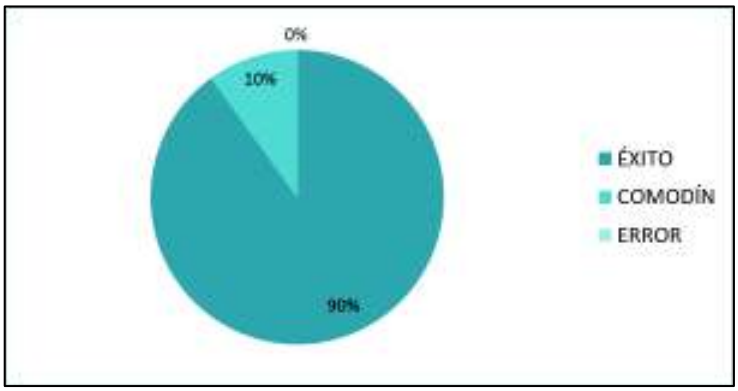


Figura 3.64 Resultados de la pregunta 25

La pregunta veinte y seis menciona ¿El chatbot te respondió al preguntarle sobre el horario de supletorios? En la Figura 3.65 se aprecian los resultados obtenidos.

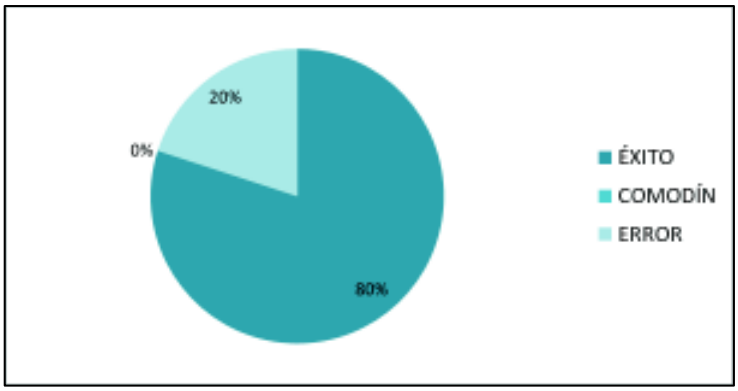


Figura 3.65 Resultados de la pregunta 26

La pregunta veinte y siete establece ¿El chatbot te contestó al preguntarle al chatbot los seminarios de la carrera? En la Figura 3.66 se aprecian los resultados obtenidos.

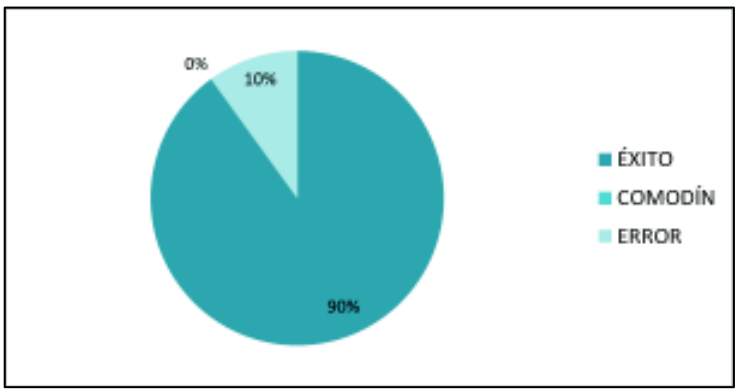


Figura 3.66 Resultados de la pregunta 27

La pregunta veinte y ocho menciona ¿El chatbot te respondió al preguntarle sobre el personal de la coordinación de la carrera? En la Figura 3.67 se aprecian los resultados obtenidos.

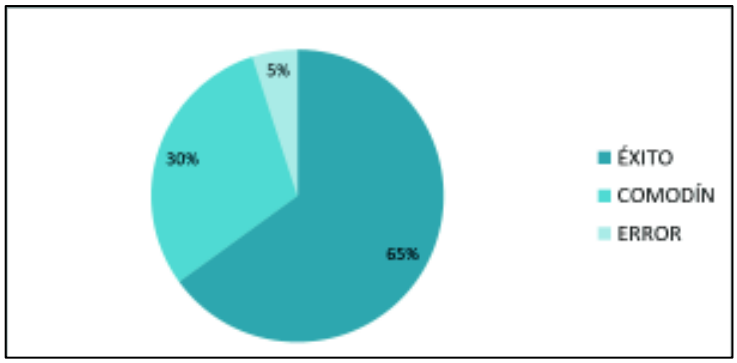


Figura 3.67 Resultados de la pregunta 28

La pregunta veinte y nueve establece ¿El chatbot te contestó al preguntarle sobre los tipos de becas institucionales? En la Figura 3.68 se aprecian los resultados obtenidos.

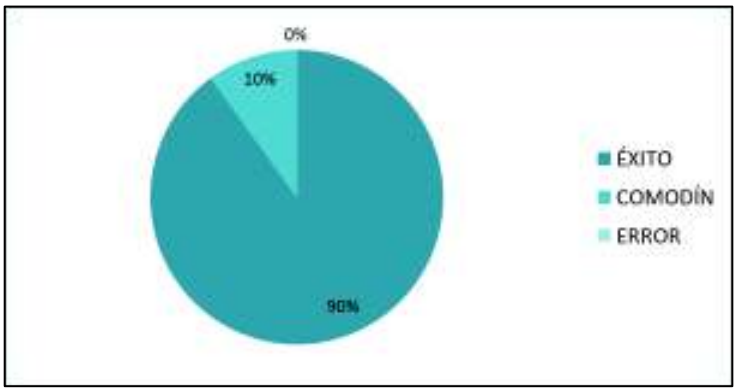


Figura 3.68 Resultados de la pregunta 29

La pregunta treinta menciona ¿El chatbot te respondió al preguntarle sobre los plazos para aplicar a becas institucionales? En la Figura 3.69 se aprecian los resultados obtenidos.

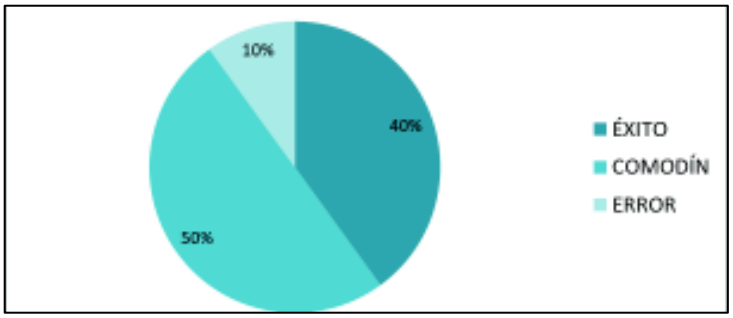


Figura 3.69 Resultados de la pregunta 30

3.3 CORRECCIÓN DE ERRORES

Al realizar las entrevistas se determinaron algunas falencias y errores tanto de la información proporcionada por el chatbot como de su funcionalidad.

3.3.1 CORRECCIÓN DE ERRORES DE FUNCIONALIDAD

El primer error se presentó cuando un usuario envió al chatbot mensajes que contenían comillas (“) y/o barras invertidas (\), en estos casos el chatbot no respondió. Para resolver este error se inspeccionó la petición enviada al servidor. La Figura 3.70 presenta los datos enviados en la petición POST que al añadir comillas corrompe el formato JSON.

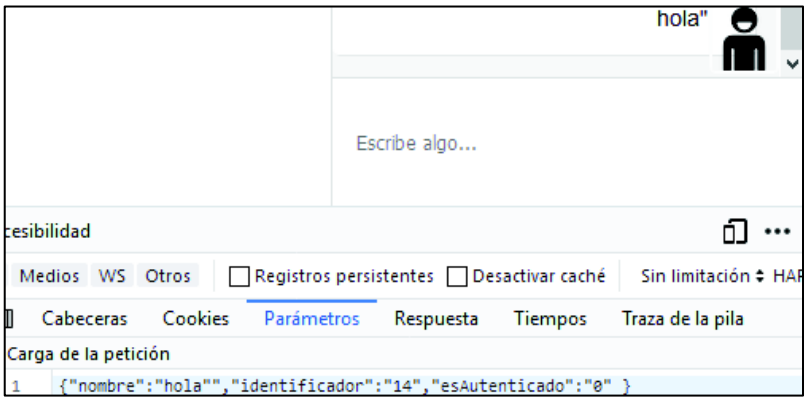


Figura 3.70 Mensaje con caracteres especiales

La Figura 3.71 muestra el error generado al enviar un mensaje que contiene comillas en la petición POST al método `FuncionChat`. Se observa un código de error 400 que indica que el servidor no pudo procesar la petición por un error de sintaxis.

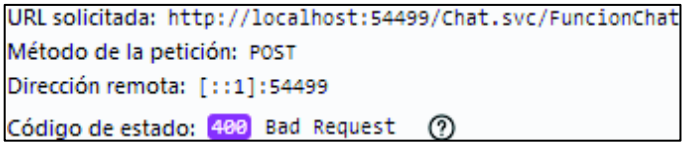


Figura 3.71 Petición HTTP al método `FuncionChat`

La solución a este problema fue eliminar las comillas y barras invertidas de los mensajes que el usuario envía al chatbot. En las líneas 61 y 62 del Código 3.1 se eliminan las comillas y barras invertidas antes de enviarlos en la petición POST.

```
57  function myFunction(e) {
58      tecla = (document.all) ? e.keyCode : e.which;
59      if (tecla == 13) {
60          var str = $("#mensajeAE").val();
61          var res = str.replace(/['"]+/g, '');
62          res = res.replace(/\\/g, '');
```

Código 3.1 Código que elimina caracteres especiales

La Figura 3.72 muestra un mensaje enviado por el usuario que contiene barras invertidas. Los datos enviados en la petición POST se encuentran en formato JSON, sin errores y sin barras invertidas, además se observa la respuesta del chatbot.

Figura 3.72 Mensaje con caracteres especiales

El segundo error se presentó el momento en que un usuario no escribió ningún mensaje y presionó la tecla `Enter`, esto ocasionó que se envíe una línea vacía. La Figura 3.73 muestra el envío de una línea vacía.

Figura 3.73 Error al enviar líneas vacías

Para solucionar este problema se modificó el archivo JavaScript. En la línea 67 del Código 3.2 se remplazan todos los espacios en blanco por un solo espacio en blanco y en la línea 68 se verifica que el mensaje no se encuentre en blanco para poder ser enviado en la petición POST.

```
64  function myFunction(e) {
65      tecla = (document.all) ? e.keyCode : e.which;
66      var str = $("#mensajeAE").val();
67      str = str.replace(/[' ']+/g, ' ');
68      if (tecla == 13 && str!= "" && str!=" ") {
```

Código 3.2 Código para no enviar líneas vacías

El tercer error se presentó cuando un usuario envió un texto con espacios en blanco al inicio o al final de la frase, el chatbot no reconoció el mensaje y no envió una respuesta válida. La Figura 3.74 presenta un ejemplo de mensaje con espacios en blanco al inicio y al final de la oración. En la Figura 3.75 se aprecia el mensaje enviado en la petición POST, se puede ver claramente los espacios en blanco.

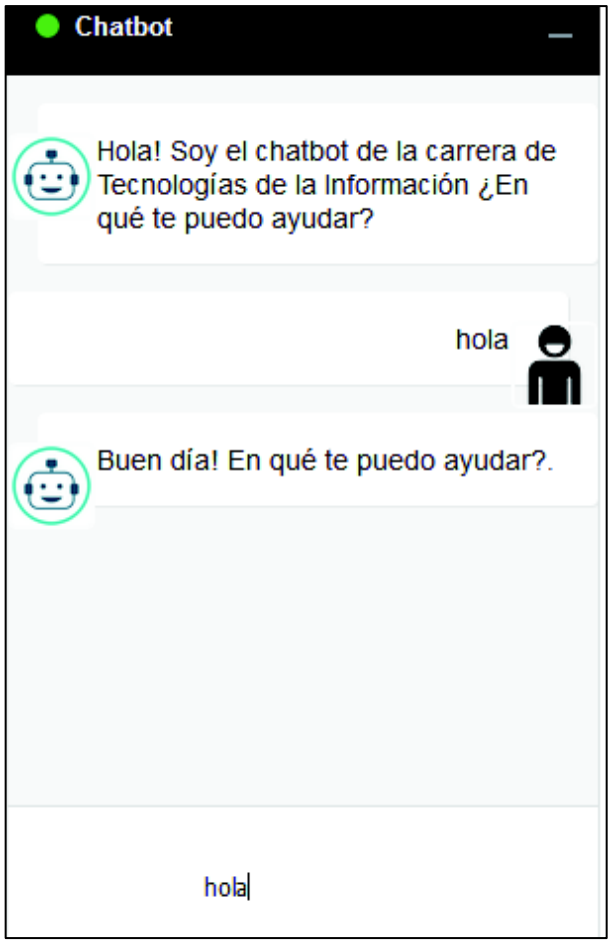


Figura 3.74 Mensaje con espacios en blanco

Figura 3.75 Parámetros enviados en la petición POST

Este error se solventó con la solución del error número dos, sin embargo también se lo puede solucionar agregando una línea de código, como se puede ver en la línea 70 del Código 3.3. Entonces en la petición POST se enviará la oración sin espacios tal como se aprecia en la Figura 3.76.

```
64  function myFunction(e) {
65      tecla = (document.all) ? e.keyCode : e.which;
66      if (tecla == 13 && $("#mensajeAE").val()!="") {
67          var str = $("#mensajeAE").val();
68          var res = str.replace(/['"]+/g, '');
69          res = res.replace(/\\/g, '');
70          res = res.trim();
```

Código 3.3 Código que soluciona el error

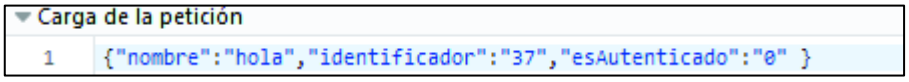


Figura 3.76 Parámetros enviados en la petición POST

El error número cuatro se presentó cuando los usuarios recibieron o enviaron nuevos mensajes y la barra de desplazamiento permaneció inmóvil. La Figura 3.77 muestra un ejemplo de conversación donde el usuario envió mensajes y el chatbot respondió pero la barra de desplazamiento no se desplazó automáticamente y generó confusión.

Figura 3.77 Chatbot sin desplazamiento automático

La solución de este problema fue añadir las tres líneas de código de JavaScript que se aprecian en el Código 3.4, estas líneas desplazan el contenido del chat y lo ubican en el último mensaje enviado o recibido.

```
108    VentanaChat = document.getElementById("body");
109    var nuevo = VentanaChat.scrollHeight;
110    VentanaChat.scrollTo(0, nuevo);
```

Código 3.4 Código para desplazar el contenido del chat

3.3.2 CORRECCIÓN DE ERRORES DE LA INFORMACIÓN DEL CHATBOT

Una vez realizadas las entrevistas se apreció claramente las distintas maneras que las personas expresan una misma idea. A continuación, con base en los resultados obtenidos en las entrevistas se indicarán las acciones correctivas que se realizarán.

- Añadir información a la base de datos para aumentar el porcentaje de respuestas del grupo “Éxito”.
- Definir palabras clave para aumentar el porcentaje de respuestas del grupo “Comodín”.

Inicialmente se tenía una base de datos con 350 elementos del lenguaje AIML. Para corregir este error y aumentar el porcentaje de respuestas del grupo “Éxito”, se añadieron nuevos elementos del lenguaje AIML a la base de datos y actualmente se tienen 450 elementos. Estos nuevos elementos se definieron con base en las preguntas que los usuarios plantearon al chatbot en las entrevistas. En la Tabla 3.1 se observa un ejemplo de algunas preguntas formuladas por los usuarios en las entrevistas junto con el nuevo elemento AIML que se añadió a la base de datos.

Tabla 3.1 Nuevo elemento AIML en la base de datos

Preguntas formuladas por los usuario en las entrevistas	Nuevo elemento AIML
¿Qué plazo tengo para **cambiar la opción de titulación**?	_ cambiar la opción de titulación *
¿Hasta cuándo se puede **cambiar la opción de titulación**?	
¿Cuándo es la fecha máxima para **cambiar la opción de titulación**?	

Para determinar las palabras clave se analizaron las treinta preguntas que el chatbot es capaz de responder. Inicialmente se definieron 20 palabras clave, después de corregir este error se tienen un total de 50 palabras clave.

Si se determina por medio del análisis morfológico que el mensaje enviado por el usuario contiene una palabra clave se enviará una respuesta del grupo “Comodín”. En la Tabla 3.2 se aprecian todas las palabras clave junto a sus correspondientes respuestas, estas respuestas procuran ser lo más cercanas a la petición del usuario.

Tabla 3.2 Palabra clave y respuesta

PALABRA CLAVE	RESPUESTA
CAMBIAR	Acaso quieres saber ¿Cómo cambiarte de opción de titulación? Puedes escribir: **Cambiarme de opción de titulación**. O quieres saber ¿Cómo cambiarte de carrera? Puedes escribir: **Cambiarme de carrera**
HORARIO	Acaso quieres saber sobre el horario de materias, escribe: **Horario de materias**. O quieres saber el horario de atención de la coordinación, entonces escribe: **Coordinación de la carrera**
ANILLADOS	Al parecer quieres saber sobre anillados, puedes escribir: **Requisitos para presentar anillados**, o puedes escribir: **Extensión para entregar anillados**
ENTREGAR	Al parecer quieres entregar anillados, puedes escribir: **Requisitos para presentar anillados**, o puedes escribir: **Extensión para entregar anillados**
PRESENTAR	Al parecer quieres saber ¿Cuándo se debe presentar documentos? Escribe: **Presentar documentos**. O quieres entregar anillados, escribe: **Requisitos para presentar anillados** o **Extensión para presentar anillados**
OPCION	Me parece que quieres saber las opciones de unidad de titulación, escribe: **Opciones de titulación**
DESIGNAR	Me parece que quieres saber acerca de la designación del tribunal, escribe: **Designación del tribunal**
SUPLETORIO	Al parecer quieres saber sobre supletorios, puedes escribir: **Supletorios**
CULMINAR	Talvez quieres saber sobre culminar tu plan de estudios. Escribe: **culminación del plan de estudios**
CONFORMAR	Me parece que quieres saber acerca de la conformación del tribunal, escribe: **conformación del tribunal**
MATRÍCULA	Acaso quieres saber: ¿Cuándo son las matrículas? Escribe: **Matrículas ordinarias**
MATRÍCULAR	Acaso quieres saber: ¿Cuándo son las matrículas? Escribe: **Matrículas ordinarias**
MATRÍCULACIÓN	Me parece que quieres saber dónde consultar tu turno de matrícula, escribe: **Turno de matrícula**

TURNO	Me parece que quieres saber dónde consultar tu turno de matrícula, escribe: **Turno de matrícula**
PERIODO	Acaso quieres saber: ¿Cuándo son las matrículas? Escribe: **Matrículas ordinarias**
REINSCRIBIR	Acaso quieres saber: ¿Cuándo son las reinscripciones? Escribe: **Reinscripciones**
HORARIO	Acaso quieres saber sobre el horario de materias, escribe: **Horario de materia**s. O quieres saber el horario de atención de la coordinación, entonces escribe: **Coordinación de la carrera**
OPCIÓN	Me parece que quieres saber las opciones de unidad de titulación, escribe: **Opciones de titulación**
TITULACIÓN	Me parece que quieres saber las opciones de unidad de titulación, escribe: **Opciones de titulación**. O acaso quieres saber ¿Cómo cambiarte de opción de titulación? Escribe: **Cambiarme de opción de titulación**.
GRADUAR	Me parece que quieres saber las opciones de unidad de titulación, escribe: **Opciones de titulación**
DEFENDER	Me parece que quieres saber ¿Cómo ser declarado apto para la defensa oral de Trabajo de Titulación? Escribe: **Apto para la defensa oral**
TRIBUNAL	Me parece que quieres saber acerca de la conformación del tribunal, escribe: **conformación del tribunal**. O acaso quieres saber sobre los plazos de designación de tribunal Escribe: **Plazos de designación de tribunal**
PAGAR	Me parece que quieres saber ¿Hasta cuándo se puede presentar la solicitud de pago en partes? Escribe: Pago en partes
SOLICITUD	Me parece que quieres saber ¿Hasta cuándo se puede presentar la solicitud de pago en partes? Escribe: **Pago en partes**
INICIAR	Talvez quieres saber ¿Cuándo inicia el semestre? Escribe: **Inicio del semestre**
TERMINAR	Talvez quieres saber ¿Cuándo termina el semestre? Escribe: **Termina el semestre**
FINALIZAR	Talvez quieres saber ¿Cuándo termina el semestre? Escribe: **Termina el semestre**
PRÁCTICA	Talvez quieres saber sobre el proceso de prácticas preprofesionales, escribe: **Prácticas Preprofesionales**. O quieres saber sobre las empresas que tienen convenio con la facultad, escribe: **Convenios prácticas preprofesionales**
MALLA	Talvez quieres saber sobre la malla curricular, escribe: **Malla curricular**
PRACTICAR	Talvez quieres saber sobre el proceso de prácticas preprofesionales, escribe: **Prácticas Preprofesionales**. O quieres saber sobre las empresas que tienen convenio con la facultad, escribe: **Convenios prácticas preprofesionales**
TURNO	Me parece que quieres saber dónde consultar tu turno de matrícula, escribe: **Turno de matrícula**

MATERIA	Acaso quieres saber sobre el horario de materias, escribe: **Horario de materias**
EXAMEN	Talvez quieres saber sobre el examen complexivo, escribe: **Examen Complexivo.** O quieres saber sobre los Plazos del Examen Complexivo escribe: **Plazos examen complexivo**
COMPLEXIVO	Talvez quieres saber sobre el examen complexivo, escribe: **Examen Complexivo**. O quieres saber sobre los Plazos del Examen Complexivo escribe: **Plazos examen complexivo**
TÉCNICO	Talvez quieres saber sobre la unidad de titulación Proyectos Técnicos, escribe: **Proyecto Técnico**.
VINCULACIÓN	Talvez quieres saber sobre la unidad de titulación Proyectos de Vinculación, escribe: **Proyecto de Vinculación**.
REQUISITO	Talvez quieres entregar anillados, puedes escribir: **Requisitos para presentar anillados**,
APTO	Me parece que quieres saber ¿Cómo ser declarado apto para la defensa oral de Trabajo de Titulación? Escribe: **Apto para la defensa oral**
DESIGNAR	Me parece que quieres saber sobre los plazos de designación de tribunal Escribe: **Plazos de designación de tribunal**
CONFORMAR	Me parece que quieres saber acerca de la conformación del tribunal, escribe: **conformación del tribunal**.
EXTENSIÓN	Al parecer quieres saber sobre anillados puedes escribir: **Extensión para entregar anillados**
PLAZO	Me parece que quieres saber sobre quieres saber sobre anillados, puedes escribir: **Extensión para entregar anillados**
OFERTA	Talvez quieres saber sobre las empresas que tienen convenio con la facultad para realizar prácticas preprofesionales, escribe: **Convenios prácticas preprofesionales**
PASANTÍA	Talvez quieres saber sobre el proceso de prácticas preprofesionales, escribe: **Prácticas Preprofesionales**. O quieres saber sobre las empresas que tienen convenio con la facultad, escribe: **Convenios prácticas preprofesionales**
COORDINACIÓN	Talvez quieres saber sobre la coordinación de la carrera, puedes escribir: **Coordinación de la carrera**. O quieres saber sobre el personal de la coordinación de la carrea entonces escribe: **Personal de la coordinación**
CARRERA	Talvez quieres saber sobre la coordinación de la carrera, puedes escribir: **Coordinación de la carrera**. O quieres saber sobre el personal de la coordinación de la carrea entonces escribe: **Personal de la coordinación**
SEMINARIO	Me parece que quieres saber sobre los seminarios que este semestre son ofertados, puedes escribir: **Seminarios de la carrera**
BECA	Talvez quieres saber sobre la coordinación de la carrera, puedes escribir: **Coordinación de la carrera**. O quieres saber sobre el personal de la coordinación de la carrea entonces escribe:

	Personal de la coordinación
APLICAR	Talvez quieres saber sobre ¿Cómo aplicar a las becas institucionales? puedes escribir: **Tipos de becas**. O quieres saber sobre los plazos para aplicar a becas institucionales, entonces escribe: **Plazos becas**
FECHA	Talvez quieres saber sobre las fechas de matrículas ordinarias, puedes escribir: **Matrículas ordinarias**. Si quieres saber sobre las fechas de reinscripciones, puedes escribir: **Reinscripciones.** Si quieres saber sobre las fechas del examen complexivo, puedes escribir: **Plazos examen complexivo.** Si quieres saber sobre las fechas de reinscripciones, puedes escribir: **Reinscripciones.** Si quieres saber sobre las fechas de inicio y fin de semestre, puedes escribir: **Inicio y fin de semestre.** Si quieres saber sobre las fechas para cambios de carrera, puedes escribir: **Cambiarme de carrera.** Si quieres saber sobre las fechas para presentar la solicitud de pago en partes, puedes escribir: **Pago en partes.** Y si quieres saber sobre las fechas para presentar documentos, puedes escribir: **Presentar documentos.**

Además, se modificó la aplicación de administración para que se puedan gestionar las palabras clave. Se añadió la Pestaña `Gestionar Palabras Clave` con el listado de estas palabras, al seleccionarlas se aprecia la respuesta que se enviará al usuario. Además, se pueden agregar nuevas palabras clave, eliminarlas o modificarlas.

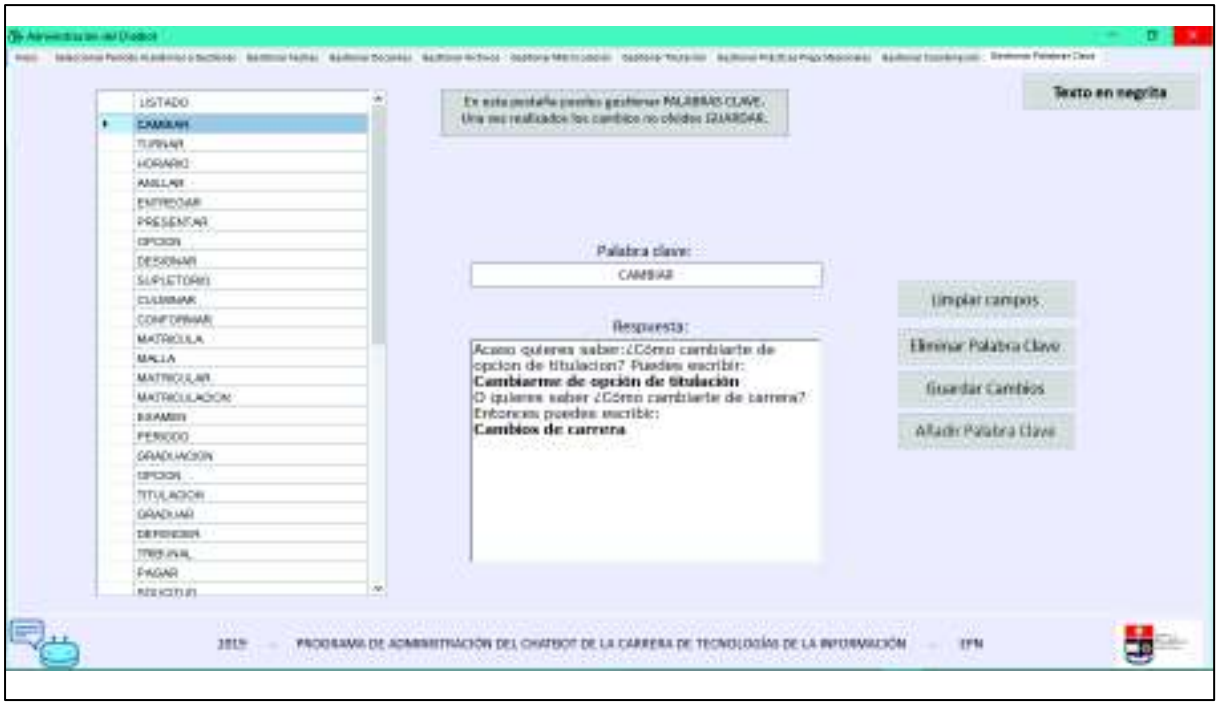


Figura 3.78 Pestaña Gestionar Palabras Clave de la Aplicación de Administración

El siguiente error presentado se aprecia en la Figura 3.79. Se observa una respuesta que el chatbot envió al usuario, la misma que se encuentra con un solo formato de texto y al ser una respuesta de múltiples líneas puede confundir al usuario.

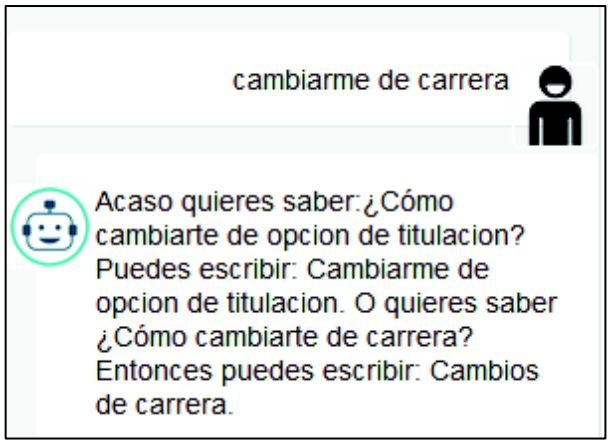


Figura 3.79 Respuesta del chatbot con un solo formato de texto

Para corregir este error se modificó la aplicación de administración. En primer lugar se cambió los `TextBox` por `RichTextBox` para manejar formatos de texto. En segundo lugar se añadió un botón que maneja el formato de texto negrillas.

En el Código 3.5 se aprecian las líneas de código que colocan o eliminan negrillas en el texto seleccionado. En la línea 1408 se define el texto a modificar. El elemento `txtRespuestaClave` es un `RichTextBox`. En las líneas 1412 a 1419 se establece si el texto seleccionado se encontraba en negrillas para eliminar las negrillas o si el texto seleccionado se encontraba en formato normal para colocar negrillas.

Finalmente, en la línea 1420 se coloca el nuevo formato al texto seleccionado.

```
1406    private void button5_Click_2(object sender, EventArgs e)
1407    {
1408        if (txtRespuestaClave.SelectionFont != null)
1409        {
1410            System.Drawing.Font currentFont = txtRespuestaClave.SelectionFont;
1411            System.Drawing.FontStyle newFontStyle;
1412            if (txtRespuestaClave.SelectionFont.Bold == true)
1413            {
1414                newFontStyle = FontStyle.Regular;
1415            }
1416            else
1417            {
1418                newFontStyle = FontStyle.Bold;
1419            }
1420            txtRespuestaClave.SelectionFont = new Font(currentFont.FontFamily,currentFont.Size,newFontStyle);
1421        }
1422    }
```

Código 3.5 Líneas de código del botón `Negrilla`

La Figura 3.80 muestra el botón `Negrilla` en la interfaz de la aplicación de administración. Gracias a este botón se puede modificar el formato del texto seleccionado en el `RichTextBox`. En la Figura 3.81 se puede apreciar la respuesta enviada por el chatbot con el nuevo formato para facilitar al usuario la lectura del mensaje.

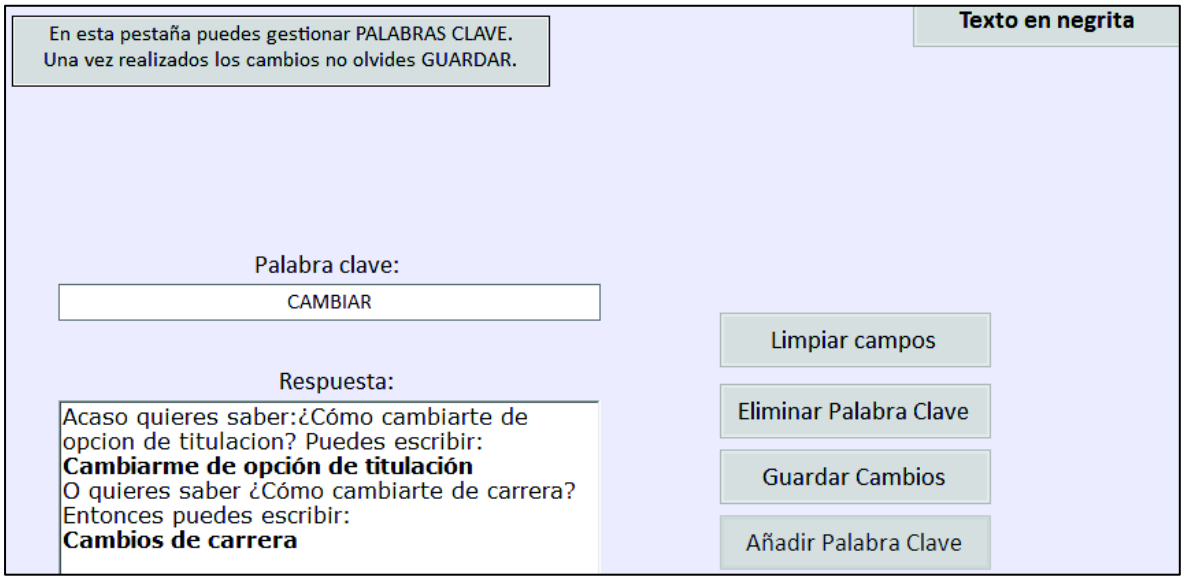


Figura 3.80 Botón `Negrilla`

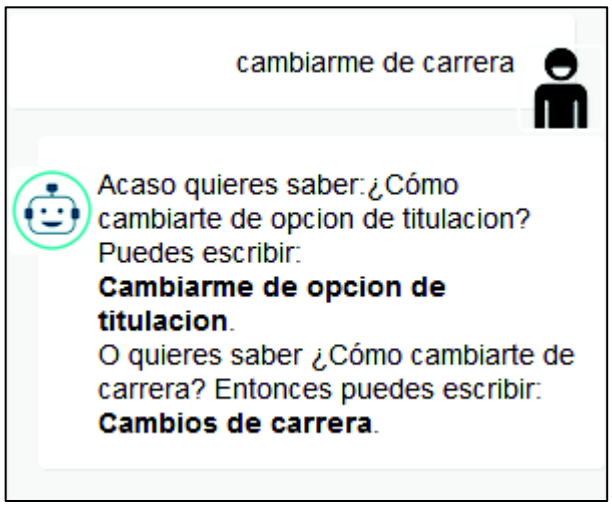


Figura 3.81 Respuesta con varios formatos de texto

Para modificar el formato de texto en la aplicación de administración se utilizó el elemento `RichTextBox`, el mismo que trabaja con el formato RTF (*Rich Text Format*) que es un formato de archivo para el intercambio de documentos multiplataforma. Gracias a esto se pudo almacenar el texto en la base de datos y presentarlo en la interfaz del chatbot. En el Código 3.6 se puede apreciar la conversión de HTML a RTF para presentar el texto en el

`RichTextBox` de la aplicación de administración. En la línea 1301 se observa el uso de la clase `HtmlToRtf` de la librería `SautinSoft`. La línea 1306 muestra la conversión a RTF mediante el método `ToRtf`.

```
1301    SautinSoft.HtmlToRtf h = new SautinSoft.HtmlToRtf();
1302    string html = texto;
1303    string rtf = String.Empty;
1304    if (h.OpenHtml(html))
1305    {
1306        rtf = h.ToRtf();
1307    }
1308    txtRespuestaClave.Rtf = rtf;
```

Código 3.6 Conversión de HTML a RTF

En el Código 3.7 se aprecia la conversión de RTF a HTML para presentar el texto en la interfaz del chatbot. En la línea 1377 se observa el uso del método `ToHtml` de la clase `Rtf` de la librería `RtfPipe`.

```
1376    string rtf = txtRespuestaClave.Rtf;
1377    var html = Rtf.ToHtml(rtf);
1378    template.Li = html;
```

Código 3.7 Conversión de RTF a HTML

El último error detectado indica que la aplicación de administración inicialmente tenía un solo botón para realizar dos acciones: seleccionar el periodo académico que se iba a gestionar y para seleccionar el periodo académico referente al que el chatbot brindaba información.

Para corregir este error se implementó una pestaña para cada acción. En la Figura 3.82 se aprecia la Pestaña `Inicio` donde se selecciona el periodo académico del cual el chatbot brinda información.

Figura 3.82 Pestaña `Inicio` de la Aplicación de Administración

En la Figura 3.83 se aprecia la pestaña `Seleccionar Periodo Académico` donde se selecciona el periodo académico a gestionar.

Figura 3.83 Pestaña `Seleccionar Periodo Académico a Gestionar` de la Aplicación de Administración

4. CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

- El objetivo de este Trabajo de Titulación fue desarrollar un prototipo de Chatbot para responder preguntas frecuentes de la Carrera de Tecnologías de la Información. Al concluir este Trabajo de Titulación se dispone de un prototipo que responde a treinta temas específicos referentes a los procesos de matriculación, titulación y otros temas definidos en la etapa de análisis. El chatbot consta de cuatro módulos principales: Módulo de Entendimiento del Lenguaje, Módulo de Seguimiento de Estado, Módulo de Política de Diálogo y Módulo Generador de Lenguaje Natural. Además, se desarrolló un mecanismo de administración para gestionar las respuestas del chatbot.

- Las entrevistas iniciales que se realizaron al personal administrativo y a los estudiantes permitieron generar historias de usuario con el fin de especificar los requerimientos funcionales y no funcionales de la aplicación así como definir los temas específicos sobre los que el chatbot es capaz de brindar información. Se definieron 30 temas.

- El prototipo de chatbot implementado proporciona información sobre 30 temas específicos, para agregar nuevos temas de conocimiento se requeriría modificar considerablemente la base de datos y el código del mecanismo de administración. Este prototipo presenta una escalabilidad limitada para agregar nuevos temas de conocimiento.

- El chatbot se lo implementó con referencia al modelo de chatbot basado en la recuperación; por tal motivo se construyó una base de datos con gran cantidad de información tanto de patrones como de respuestas, para que el chatbot conozca sobre 30 temas específicos se implementaron alrededor de 500 elementos en la base de datos.

- Las entrevistas finales permitieron detectar los errores del prototipo de chatbot. Además, se apreció claramente las distintas maneras que las personas expresan una misma idea. Existieron errores de funcionalidad del chatbot y errores en la información que brindó. Con estos resultados se corrigieron estos errores.

- Se consideró importante implementar un mecanismo de administración que permita gestionar la información proporcionada por el chatbot para que esta información corresponda al periodo académico actual y para que el chatbot pueda ser usado en más de un semestre. Este mecanismo de administración proporciona escalabilidad para agregar nuevas respuestas. Se codificó una aplicación de escritorio con diez pestañas, donde se pueden gestionar los periodos académicos y las respuestas a los treinta temas que el chatbot es capaz de responder. Esta aplicación interactúa con un servicio web WCF.

- El módulo Entendimiento del Lenguaje permite al usuario ingresar al chatbot por medio de autenticación o ingresar al chatbot sin autenticarse. También procesa los mensajes que los usuarios envían al chatbot. Además, se encarga de gestionar el análisis morfológico de estos mensajes. Este módulo está conformado por dos servicios web WCF.

- El módulo Seguimiento de Estado almacena en la base de datos el registro de las conversaciones que el chatbot entabla con los usuarios. Por lo tanto, el chatbot entiende el contexto de los nuevos mensajes recibidos.

- El módulo Política de Diálogo selecciona la respuesta que el chatbot enviará al usuario conforme el mensaje recibido. Este módulo utiliza lenguaje AIML.

- El módulo Generador de Lenguaje Natural se encarga de entregar la respuesta para el usuario en la interfaz gráfica del chatbot. Este módulo está formado por un servicio web WCF.

- El prototipo de chatbot que se implementó puede estar embebido o contenido en cualquier página web. Para este Trabajo de Titulación se lo incluyó en la página web `VentanaChat.html`.

- Para incluir este prototipo de chatbot dentro de una página web se debe: modificar el código HTML que describe la estructura de la página; e incluir: los archivos que contienen el código JavaScript que proveen la funcionalidad del chatbot y el archivo del código CSS que establece su apariencia.

- El chatbot se comunica con el servicio WCF a través de peticiones asíncronas AJAX. Estas peticiones permiten que el usuario interactúe en todo momento con la página web donde se encuentra contenido el chatbot. Por esto, no se necesita recargar la página web para enviarle mensajes al chatbot o recibir respuestas del mismo.

- En este prototipo de chatbot se intercambian mensajes entre cliente y servidor mediante servicios web WCF; el cliente es el chatbot contenido en la página web debido a que inicia la comunicación.
- Para tener acceso a la funcionalidad de los servicios WCF se crearon puntos de conexión compatibles con los servicios que se requirió contactar. En estos puntos de conexión se definió la información necesaria para el intercambio de mensajes.
- La página web donde se integró el chatbot (cliente) accede a los servicios web WCF (servidor) mediante solicitudes HTTP de tipo GET y POST.
- Para el intercambio de datos se empleó el formato de datos JSON por ser ligero, no verboso y por su rápida integración con JavaScript.
- Por defecto los servicios web WCF compatibles con AJAX devuelven datos en formato XML, en este Trabajo de Titulación se modificó el archivo `Web.config` del servicio web para usar el formato de datos JSON.
- Para la implementación de este chatbot se empleó el lenguaje de programación AIML que permitió crear una aplicación con capacidad de conversación de forma simple y entendible. Se utilizó la librería `AIMLbot` para C# la misma que presenta métodos y clases para procesar documentos AIML.
- Este prototipo de chatbot utilizó el análisis morfológico que es un componente del Procesamiento de Lenguaje Natural, con el objetivo de interpretar el lenguaje humano. Se empleó la herramienta Grampal para realizar este análisis, la herramienta asigna una etiqueta a cada palabra analizada con su categoría sintáctica y su lema. Esto permite determinar si los mensajes contienen palabras clave para que el chatbot envíe una determinada respuesta.
- Se empleó la herramienta Selenium WebDriver para hacer uso de la herramienta Grampal y por medio de líneas de código realizar el análisis morfológico de algunos mensajes que los usuarios envían al chatbot.
- Por medio de los métodos de la librería `RestSharp` se consumió el servicio WCF `Grampal.svc` lo que agilizó la creación y ejecución de peticiones web; además, de facilitar el procesamiento de respuestas.
- Para la implementación de la interfaz del chatbot se tomó como base las plantillas "*chat box Costumer care*" de Bootstrap de la página web: `https://bootsnipp.com/snippets/eomDM`. Después, se modificaron los

archivos HTML, JavaScript y CSS para adaptarlos a los requerimientos del chatbot de este Trabajo de Titulación.

- El chatbot desarrollado en este Trabajo de Titulación implementa la Generación de Lenguaje Natural mediante la heurística basada en patrones. Por tanto, se tienen patrones y respuestas previamente definidos.

4.2 RECOMENDACIONES

- Para implementar un chatbot escalable se recomienda implementarlo a partir de alguna de las plataformas cognitivas que existen en el medio, como Microsoft Azure Cognitive Services, IBM Watson, entre otras. Caso contrario se deberá codificar una base de datos sumamente grande empleando gran cantidad de tiempo y energía que se podría emplear para proporcionarle alguna funcionalidad al chatbot.

- El prototipo de chatbot implementado en este Trabajo de Titulación responde a treinta preguntas frecuentes de la carrera de Tecnologías de la Información. Para futuros desarrollos se recomienda que el chatbot proporcione información sobre otros temas de interés de los estudiantes como formatos de solicitudes, normativa, aulas, laboratorios.

- Para futuros desarrollos se puede incluir un chatbot en alguna página web institucional como en la página del SAEw, en la página web del DETRI, en la página web del Aula Virtual o inclusive integrarlo en la página web de la EPN. Además, se recomienda vincular la información que estas plataformas proveen para que el chatbot brinde datos como horario académico, supletorios, notas, estado de plan académico, tribunal designado de cada estudiante.

- En un futuro se puede desarrollar un chatbot para dispositivos móviles Android, o inclusive incluirlo en la aplicación móvil de la Escuela Politécnica Nacional.

- La aplicación de administración de este Trabajo de Titulación fue implementada como aplicación de escritorio. Para un futuro se puede implementar una aplicación web para gestionar el chatbot.

- Todos los datos que viajan entre el navegador web y los servicios WCF se envían en texto plano debido a que el alcance de este Trabajo de Titulación no consideró mecanismos de privacidad. En un futuro se puede proporcionar privacidad al usuario cifrando estos datos.

- Para el análisis morfológico del mensaje enviado por el usuario se utilizó la herramienta externa Grampal. En un futuro se puede realizar el análisis morfológico de los mensajes de forma local.
- Este prototipo de chatbot utilizó el análisis morfológico que es un componente del Procesamiento de Lenguaje Natural. Para futuros desarrollos se puede emplear otros componentes del Procesamiento de Lenguaje Natural como: el análisis sintáctico o el análisis semántico o el análisis pragmático de la oración.
- El chatbot desarrollado en este Trabajo de Titulación implementa la Generación de Lenguaje Natural mediante el método de textos enlazados. En un futuro se puede implementar la Generación de Lenguaje Natural basado en el conocimiento, donde el sistema genera todos los mensajes de forma dinámica basándose en la recopilación de información, entrenamiento y pruebas.
- Este trabajo de titulación implementa la lógica en lenguaje C#. Para futuros trabajos de titulación se puede implementar un chatbot en diferentes lenguajes de programación como Python que posee diversas herramientas cognitivas que pueden facilitar la implementación de un asistente virtual.
- Se utilizó la herramienta Selenium WebDriver para realizar consultas a la herramienta Grampal. Como alternativa en un futuro se puede usar `RestSharp` para realizar estas peticiones.
- En este Trabajo de Titulación se gestionó la base de datos a través de un cliente SQL. En un futuro se puede gestionar la base de datos por medio de sentencias LINQ.
- En este Trabajo de Titulación debido a que es un prototipo se empleó la base de datos Microsoft SQL Server. Si en un futuro se lanzara a producción este chatbot se requeriría licenciamiento, por lo tanto se recomienda comparar los precios de licenciamiento para elegir la mejor opción.
- Los servicios web de este Trabajo de Titulación se desplegaron de forma local. En un futuro para lanzar en producción este chatbot se lo puede publicar a través de alguna plataforma como Google Cloud Platform o Amazon Web Services.
- En este Trabajo de Titulación se implementó un chatbot que brinda información, en futuros trabajos de titulación se puede implementar chatbots o asistentes virtuales que ejecuten alguna funcionalidad y automaticen algún proceso.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] C. Duchanoy, Chatbots: Conversación natural con robots, Ciudad de México, 2018.

[2] M. Dorfman, N. Mazza , A. Grondona y P. Mazza, Asistentes Virtuales de Clase como complemento a la educación universitaria presencial, Buenos Aires, 2011.

[3] A. F. Toledo, «Desarrollo de un chatbot que ayude a responder a preguntas frecuentes referentes a becas en la Universidad Técnica Particular de Loja.,» 2018.

[4] A. Van Woudenberg, «Chatbots and dialogue management,» de *A Chatbot Dialogue Manager*, 2014, p. 1.

[5] A. Moreno, «Procesamiento del lenguaje natural,» 17 Octubre 2017. [En línea]. Available: http://www.iic.uam.es/inteligencia/que-es-procesamiento-del-lenguaje-natural/. [Último acceso: 6 Enero 2018].

[6] Chatcompose, «Comprendiendo la Arquitectura de Chatbots,» [En línea]. Available: https://www.chatcompose.com/arquitectura.html. [Último acceso: 15 Julio 2019].

[7] TutorialsPoint, «AIML Tutorial,» 2018. [En línea]. Available: https://www.tutorialspoint.com/aiml/. [Último acceso: 15 Enero 2019].

[8] Mozilla, «Generalidades del protocolo HTTP,» 2005-2019. [En línea]. Available: https://developer.mozilla.org/es/docs/Web/HTTP/Overview. [Último acceso: 20 Enero 2019].

[9] Mozilla, «Mensajes HTTP,» 2019. [En línea]. Available: https://developer.mozilla.org/es/docs/Web/HTTP/Messages. [Último acceso: 9 Abril 2019].

[10] «¿Qué es HTML?,» de *Designing Internet Home Pages Made Simple*, Barcelona, Marcombo, 1999.

[11] Mozilla, «Acerca de JavaScript | MDN,» [En línea]. Available: https://developer.mozilla.org/es/docs/Web/JavaScript/Acerca_de_JavaScript. [Último acceso: 9 Enero 2019].

[12] La Fundación jQuery, «Documentation,» 2019. [En línea]. Available: https://learn.jquery.com/using-jquery-core/document-ready/. [Último acceso: 3

Febrero 2019].

[13] «¿Qué es Ajax?,» 22 Abril 2014. [En línea]. Available: https://www.ibm.com/support/knowledgecenter/es/SS8PJ7_9.6.1/com.ibm.etools.webtoolscore.doc/topics/cajax.html. [Último acceso: 8 Enero 2019].

[14] «Manual de Ajax,» Anyelguti, 13 Julio 2018. [En línea]. Available: https://aprende-web.net/progra/ajax/ajax_1.php. [Último acceso: 8 Enero 2019].

[15] La fundación jQuery, «jQuery,» 2019. [En línea]. Available: https://jquery.com/. [Último acceso: 20 Enero 2019].

[16] «The jQuery Foundation,» 2019. [En línea]. Available: http://api.jquery.com/jquery.ajax/. [Último acceso: 8 Enero 2019].

[17] D. Crockford, «Introducing JSON,» [En línea]. Available: https://www.json.org/. [Último acceso: 20 Enero 2019].

[18] Microsoft, «Conceptos básicos de Windows Communication Foundation,» 29 Marzo 2017. [En línea]. Available: https://docs.microsoft.com/es-es/dotnet/framework/wcf/fundamental-concepts. [Último acceso: 3 Febrero 2019].

[19] Microsoft, «Arquitectura de Windows Communication Foundation,» [En línea]. Available: https://docs.microsoft.com/es-mx/dotnet/framework/wcf/architecture. [Último acceso: 10 Marzo 2019].

[20] Microsoft, «Diseño e Implementación de servicios,» 29 Marzo 2017. [En línea]. Available: https://docs.microsoft.com/es-es/dotnet/framework/wcf/designing-and-implementing-services. [Último acceso: 3 Enero 2019].

[21] Microsoft, «Configuración de servicios mediante archivos de configuración,» [En línea]. Available: https://docs.microsoft.com/es-mx/dotnet/framework/wcf/configuring-services-using-configuration-files. [Último acceso: 10 Marzo 2019].

[22] Microsoft, «Creating WCF AJAX Services without ASP.NET,» [En línea]. Available: https://docs.microsoft.com/en-us/dotnet/framework/wcf/feature-details/creating-wcf-ajax-services-without-aspnet. [Último acceso: 22 Enero 2019].

[23] Microsoft, «EndpointBehavior,» [En línea]. Available: https://docs.microsoft.com/es-

mx/dotnet/framework/configure-apps/file-schema/wcf/behavior-of-endpointbehaviors. [Último acceso: 10 Marzo 2019].

[24] Moreno y Guirao, «Grampal,» [En línea]. Available: http://www.lllf.uam.es/ESP/Grampal.html. [Último acceso: 1 Enero 2019].

[25] A. M. Sandoval y J. M. Guirao Miras , «Frecuencia y distintividad en el uso lingüístico: casos tomados de la,» Universidad Autónoma de Madrid, Universidad de Granada , [En línea]. Available: https://www.um.es/lacell/aelinco/contenido/pdf/14.pdf. [Último acceso: 9 Enero 2019].

[26] L. SHARM, «Selenium C Sharp,» 2013-2018. [En línea]. Available: https://www.toolsqa.com/selenium-c-sharp/. [Último acceso: 3 Febrero 2019].

[27] «Selenium C# Webdriver Tutorial,» [En línea]. Available: https://www.guru99.com/selenium-csharp-tutorial.html. [Último acceso: 3 Febrero 2019].

[28] J. Vegas, «Introdución al SQL,» [En línea]. Available: https://www.infor.uva.es/~jvegas/cursos/bd/sqlplus/sqlplus.html. [Último acceso: 13 3 2019].

[29] A. M. Godínez González y G. Hernández Moreno, EL GRAN LIBRO de los Procesos Esbeltos: Los Principios ACTUALES de Manufactura Esbelta y Mejora Continua, Guanajuato: Ignius Media Innovation, 2014.

[30] Kanbanize, «¿Qué es una tarjeta Kanban? Definición y explicación de los detalles,» [En línea]. Available: https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-tarjeta-kanban/. [Último acceso: 13 3 2019].

[31] Kanbanize, «Qué es un tablero Kanban: Fundamentos,» [En línea]. Available: https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-tablero-kanban/. [Último acceso: 13 3 2019].

[32] «Diagramas de Casos de Uso,» UNAM, [En línea]. Available: http://www.mcc.unam.mx/~cursos/Objetos/Cap17/cap17.html. [Último acceso: 23 Febrero 2019].

[33] Microsoft, «Diagramas de casos de uso de UML,» 2017. [En línea]. Available: https://docs.microsoft.com/es-es/visualstudio/modeling/uml-use-case-diagrams-guidelines?view=vs-2015. [Último acceso: 23 Febrero 2019].

[34] Microsoft, «UML,» [En línea]. Available: https://docs.microsoft.com/es-es/previous-versions/bb972214(v=msdn.10). [Último acceso: 3 Marzo 2019].

[35] F. J. García Peñalvo, «Diagramas de Clase en UML 1.1,» [En línea]. Available: https://repositorio.grial.eu/bitstream/grial/353/1/DClase.pdf. [Último acceso: 2019 Febrero 23].

[36] J. Camacho Romero, «Diagrama de clases, diagrama de casos de uso,» [En línea]. Available: http://cic.puj.edu.co/wiki/lib/exe/fetch.php?media=materias:pis:diagramas_de_clases_y_casos_de_uso.pdf. [Último acceso: 23 Febrero 2019].

[37] Microsoft, «Clases estáticas y Miembros de clases estáticas,» [En línea]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/static-classes-and-static-class-members. [Último acceso: 2019 Marzo 4].

[38] Microsoft, «SqlCommand,» [En línea]. Available: https://docs.microsoft.com/en-us/dotnet/api/system.data.sqlclient.sqlcommand?view=netframework-4.7.2#definition. [Último acceso: 4 Marzo 2019].

[39] L. Carvajal, Metodología de la Investgación Científica. Curso general y aplicado, 28 ed., Santiago de Cali: U.S.C., 2006, p. 139.

[40] L. R. Aguado, A. García y P. Martínez, Planteamiento semántico y pragmático para gestión de diálogos en asistentes virtuales, 2002.

[41] M. D. Arango Serna, L. F. Campuzano Zapata y J. A. Zapata Cortes, «Mejoramiento de procesos de manufactura utilizando Kanban,» 2015.

[42] The OMG, «UNIFIED MODELING LANGUAGE,» 1997-2019. [En línea]. Available: http://www.uml.org/what-is-uml.htm. [Último acceso: 2019 Febrero 23].

ANEXOS

ANEXO A. Modelo de la entrevista inicial.

ANEXO B. *Script* en lenguaje SQL del Chatbot.

ANEXO C. Solución del Chatbot.

ANEXO D. Proyecto de la Aplicación de Administración.

ANEXO E. Manual de usuario de la Aplicación de Administración del Chatbot.

ANEXO F. Modelo de la entrevista final.

ORDEN DE EMPASTADO