

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
SEMINARIO DE SISTEMAS 1
Primer Semestre 2024



Manual Tecnico

Nombre	Carne
Eduardo Isaf Ajsivinac Xico	201503584
Jonatan Leonel Garcia Arana	20200424
Kemel Josue Efrain Ruano Jeronimo	202006373
Estephanie Edelweiss Lemus Herrera	201603064

Objetivo General:

Facilitar a los usuarios la planificación de viajes ofreciendo una plataforma de turismo inteligente que integre tecnologías innovadoras para proporcionar información, recomendaciones y experiencias personalizadas.

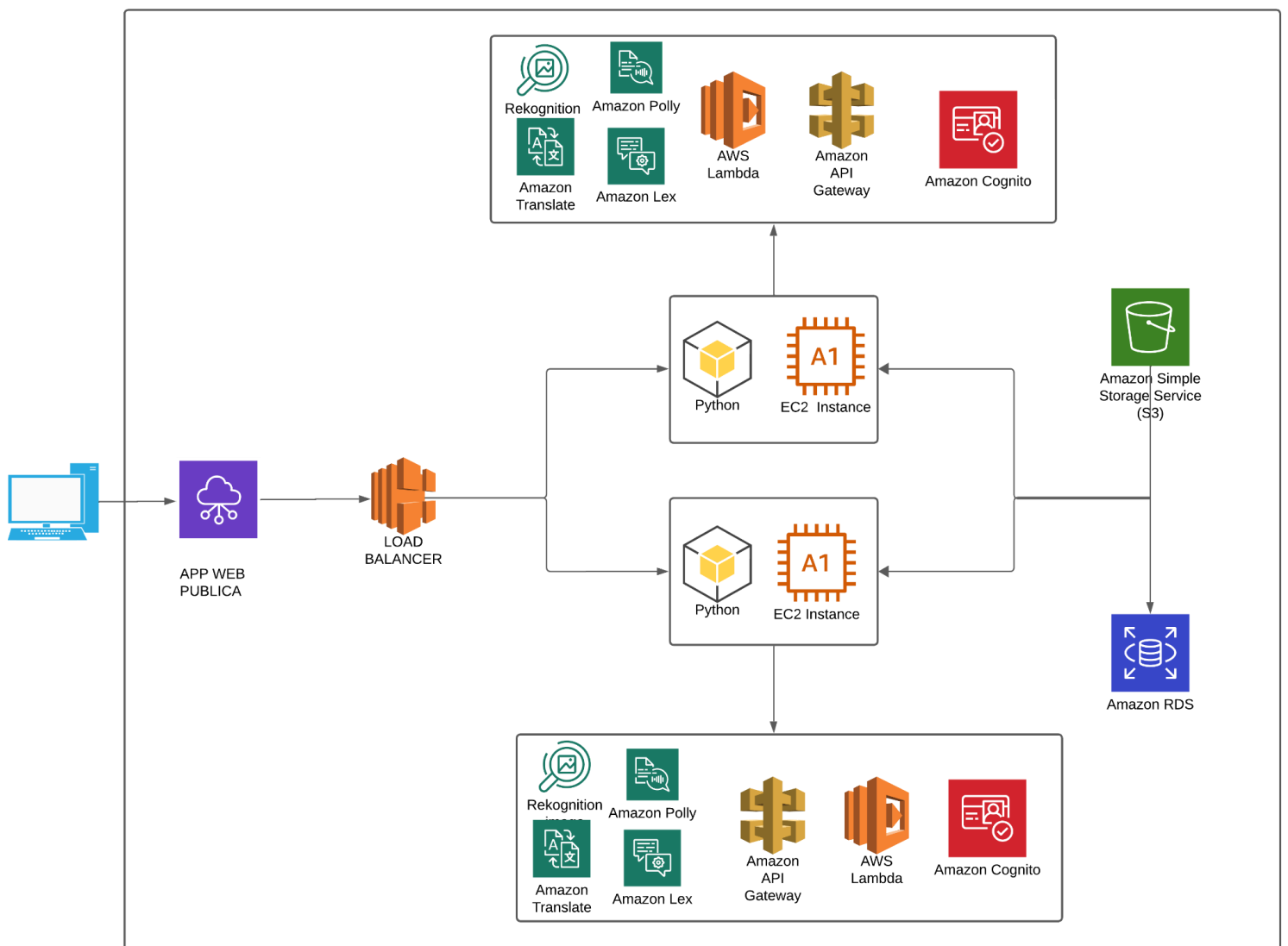
Objetivos Específicos:

1. Implementar un sistema de filtrado visual basado en etiquetas obtenidas mediante Amazon Rekognition para permitir a los usuarios buscar centros turísticos de acuerdo a sus preferencias visuales.
2. Mejorar la accesibilidad y la inclusión mediante la traducción automática de comentarios a otros idiomas utilizando Amazon Translate, garantizando que los usuarios de diferentes regiones puedan interactuar y contribuir a la comunidad de reseñas.
3. Ofrecer una experiencia de usuario interactiva y personalizada mediante un bot de recomendaciones impulsado por Amazon Lex, que recomienda destinos turísticos según las preferencias del usuario en términos de clasificación por estrellas y país de interés. Además, proporciona la opción de escuchar los comentarios de los usuarios convertidos en voz a través de Amazon Polly para una experiencia multimedia completa.

Descripción

Nuestra plataforma de Turismo Inteligente utiliza tecnologías innovadoras de Amazon Web Services para simplificar la planificación de viajes. Con un enfoque en la accesibilidad y la personalización, los usuarios pueden explorar destinos turísticos utilizando filtros visuales basados en etiquetas obtenidas mediante Amazon Rekognition. Además, Amazon Translate permite la traducción automática de comentarios, fomentando la inclusión global en nuestra comunidad de reseñas. Un bot de recomendaciones impulsado por Amazon Lex ofrece sugerencias personalizadas según las preferencias del usuario, mientras que Amazon Polly permite escuchar los comentarios convertidos en voz. Con una experiencia de usuario única y enriquecedora, nuestra plataforma está diseñada para hacer que descubrir el mundo sea más fácil y emocionante para todos los viajeros. ¡Únete a nosotros y comienza tu próxima aventura!

Arquitectura



Presupuesto

Amazon Cognto

Amazon Cognito	No se ha aplicado ningún grupo	US East (Ohio)	0,00 USD	10,00 USD
----------------	--------------------------------	----------------	----------	-----------

Estado: -

Descripción: Amazon Cognito

Resumen de la configuración: Número de usuarios activos mensuales (MAU) (200), Características de seguridad avanzadas (Habilitado)

Amazon EC2

Amazon EC2	No se ha aplicado ningún grupo	US East (Ohio)	0,00 USD	6,13 USD
------------	--------------------------------	----------------	----------	----------

Estado: -

Descripción: EC2 - servidores

Resumen de la configuración: Tenencia (Instancias compartidas), Sistema operativo (Linux), Carga de trabajo (Consistent, Número de instancias: 2), Instancia EC2 por adelantado (t4g.micro), Pricing strategy (Compute Savings Plans 3yr No Upfront), Habilitar la monitorización (desactivado), DT Entrada: Not selected (0 TB al mes), DT Salida: Not selected (0 TB al mes), DT Intra-región: (0 TB al mes)

AMAZON S3

Amazon Simple Storage Service (S3)	No se ha aplicado ningún grupo	US East (Ohio)	0,00 USD	0,48 USD
------------------------------------	--------------------------------	----------------	----------	----------

Estado: -

Descripción:

Resumen de la configuración: Almacenamiento de S3 Estándar (20 GB por mes), Solicitudes PUT, COPY, POST y LIST a S3 Standard (2000), Datos devueltos por S3 Select (4 GB por mes), Datos escaneados por S3 Select (3 GB por mes) DT Entrada: Internet (5 TB al mes), DT Salida: Not selected (0 TB al mes)

Balanceador De Carga

Elastic Load Balancing	No se ha aplicado ningún grupo	US East (Ohio)	0,00 USD	16,44 USD
------------------------	--------------------------------	----------------	----------	-----------

Estado: -

Descripción:

Resumen de la configuración: Número de balanceadores de carga de aplicaciones (1)

AWS LAMBDA

AWS Lambda	No se ha aplicado ningún grupo	US East (Ohio)	0,00 USD	0,02 USD
-------------------	--------------------------------	----------------	----------	----------

Estado: -

Descripción:

Resumen de la configuración: Arquitectura (x86), Arquitectura (x86), Modo de invocación (En búfer), Cantidad de solicitudes (10000 por mes), Cantidad de almacenamiento efímero asignado (512 MB)

AMAZON API GATEWAY

Amazon API Gateway	No se ha aplicado ningún grupo	US East (Ohio)	0,00 USD	1,00 USD
---------------------------	--------------------------------	----------------	----------	----------

Estado: -

Descripción:

Resumen de la configuración: Unidades de solicitud de la API REST (millones), Tamaño de memoria caché (GB) (Ninguno), Unidades de mensaje WebSocket (miles), Unidades de solicitudes de la API HTTP (millones), Tamaño promedio de cada solicitud (34 KB), Tamaño promedio del mensaje (32 KB), Solicitudes (1 por mes)

RDS

Amazon RDS for MySQL	No se ha aplicado ningún grupo	US East (Ohio)	0,00 USD	73,95 USD
-----------------------------	--------------------------------	----------------	----------	-----------

Estado: -

Descripción:

Resumen de la configuración: Almacenamiento para cada instancia RDS (SSD de uso general (gp2)), Cantidad de almacenamiento (20 GB), Cantidad (1), Tipo de instancia (db.t4g.small), Utilización (solo bajo demanda) (100 %Utilized/Month), Opción de implementación (Multi-AZ), Modelo de precios (OnDemand)

AMAZON REKOGNITION

Amazon Rekognition	No se ha aplicado ningún grupo	US East (Ohio)	0,00 USD	10,00 USD
---------------------------	--------------------------------	----------------	----------	-----------

Estado: -

Descripción:

Resumen de la configuración: Número de imágenes procesadas con llamadas API de etiquetas por mes (10000 por mes)

AMAZON TRANSLATE

Amazon Translate	No se ha aplicado ningún grupo	US East (Ohio)	0,00 USD	15,00 USD
-------------------------	--------------------------------	----------------	----------	-----------

Estado: -

Descripción:

Resumen de la configuración: Número de caracteres, incluidos espacios en blanco y signos de puntuación (traducción estándar en tiempo real) (1000000)

AMAZON POLLY

Amazon Polly	No se ha aplicado ningún grupo	US East (Ohio)	0,00 USD	4,00 USD
---------------------	--------------------------------	----------------	----------	----------

Estado: -

Descripción:

Resumen de la configuración: Número de solicitudes (conversión de texto a voz estándar) (10000 por mes), Número de caracteres por solicitud incluidos espacios en blanco, pero sin incluir SSML (100)

SERVICIOS UTILIZADOS

1. Amazon EC2 (Elastic Compute Cloud):

Amazon EC2 es un servicio web que proporciona capacidad informática escalable en la nube. Permite a los usuarios ejecutar aplicaciones en servidores virtuales, conocidos como instancias EC2. EC2 ofrece flexibilidad, escalabilidad y control completo sobre los recursos informáticos necesarios para ejecutar aplicaciones.

2. Amazon S3 (Simple Storage Service):

Amazon S3 es un servicio de almacenamiento en la nube altamente escalable y duradero. Ofrece almacenamiento de objetos para una amplia variedad de datos, incluidas imágenes, vídeos, archivos de texto y mucho más. S3 proporciona una infraestructura segura y de alta disponibilidad para almacenar y recuperar datos de manera eficiente.

3. Balanceador de carga de Elastic Load Balancing:

El balanceador de carga de Elastic Load Balancing es un servicio que distribuye automáticamente el tráfico de red entrante entre múltiples instancias EC2. Proporciona una alta disponibilidad y escalabilidad al dirigir el tráfico solo a instancias saludables y distribuir la carga de manera equitativa, lo que mejora el rendimiento y la tolerancia a fallos de las aplicaciones.

4. Amazon Cognito:

Amazon Cognito es un servicio de gestión de identidades y acceso que permite a los desarrolladores agregar fácilmente funciones de autenticación, autorización y gestión de usuarios a sus aplicaciones. Proporciona una forma segura de crear cuentas de usuario, iniciar sesión y acceder a recursos protegidos en aplicaciones web y móviles.

5. Amazon Rekognition:

Amazon Rekognition es un servicio de análisis de imágenes y videos basado en aprendizaje profundo. Permite analizar y extraer información valiosa de imágenes, como etiquetas, rostros, objetos y escenas, lo que facilita la búsqueda y el procesamiento de contenido visual en aplicaciones.

6. Amazon Translate:

Amazon Translate es un servicio de traducción automática de alta calidad que utiliza tecnologías de aprendizaje automático para traducir texto entre idiomas de manera rápida y precisa. Facilita la internacionalización de aplicaciones al permitir la traducción automática de contenido, como comentarios de usuarios, a múltiples idiomas.

7. Amazon Lex:

Amazon Lex es un servicio de inteligencia artificial para la creación de bots de conversación. Permite a los desarrolladores crear interfaces de conversación de voz y texto que pueden interactuar con los usuarios de manera natural, proporcionando respuestas automáticas y recomendaciones personalizadas.

8. Amazon Polly:

Amazon Polly es un servicio que convierte texto en voz utilizando tecnología de síntesis de voz avanzada. Permite agregar capacidades de voz a aplicaciones para mejorar la accesibilidad y la experiencia del usuario, convirtiendo texto en voz en tiempo real con una calidad de voz natural y realista.

CÓDIGO

```
1 def login_cognito(user,password):
2     try:
3         response = cognito.initiate_auth(
4             ClientId=client_id,
5             AuthFlow='USER_PASSWORD_AUTH',
6             AuthParameters={
7                 'USERNAME': user,
8                 'PASSWORD': password
9             }
10        )
11        access_token = response["AuthenticationResult"]["AccessToken"]
12        response = cognito.get_user(AccessToken=access_token)
13        return response
14    except Exception as e:
15        print("Error al iniciar sesión:", e)
16        return False
```

La función `login_cognito` facilita el proceso de autenticación de usuarios en Amazon Cognito mediante el flujo de autenticación basado en usuario y contraseña. Al recibir el nombre de usuario y la contraseña, esta función inicia la autenticación y obtiene un token de acceso que permite al usuario autorizar solicitudes subsiguientes a recursos protegidos por Cognito. Además, utilizando este token de acceso, la función recupera información adicional del usuario, como su perfil y atributos, proporcionando así una experiencia de inicio de sesión más completa. Si la autenticación es exitosa, la función devuelve esta información del usuario; de lo contrario, devuelve `False` para indicar un fallo en el proceso de inicio de sesión.



```
1  def singUp(user,name,email,password):
2      try:
3          response = cognito.sign_up(
4              ClientId=client_id,
5              Username=user,
6              Password=password,
7              UserAttributes=[
8                  {
9                      'Name': 'name',
10                     'Value': name
11                 },
12                 {
13                     'Name': 'email',
14                     'Value': email
15                 }
16             ]
17          )
18          return response
19      except Exception as e:
20          print("Error al registrar usuario:", e)
21          return False
```

Esta función singUp registra un nuevo usuario en Amazon Cognito con la información proporcionada, incluyendo nombre de usuario, nombre, dirección de correo electrónico y contraseña. Si el registro es exitoso, devuelve la respuesta del servicio Cognito. En caso de error, imprime un mensaje de error y devuelve False.

```

1  def Tranlatetext(texto , idioma):
2      load_dotenv()
3      translate = boto3.client(
4          'translate',
5          aws_access_key_id=os.environ.get('AWS_ACCESS_KEY_TRANSLATE'),
6          aws_secret_access_key=os.environ.get('AWS_SECRET_ACCESS_KEY_TRANSLATE'),
7          region_name=os.environ.get('AWS_REGION_NAME_TRANSLATE')
8      )
9      response = translate.translate_text(
10         Text=texto,
11         SourceLanguageCode="auto",
12         TargetLanguageCode=idioma
13     )
14
15     return response['TranslatedText']

```

Esta función TranslateText utiliza el servicio Amazon Translate para traducir un texto dado a un idioma especificado. Primero, se configura el cliente de Translate utilizando las credenciales y la región proporcionadas a través de variables de entorno. Luego, se llama al método translate_text con el texto a traducir, especificando el idioma de origen como "auto" para detectar automáticamente el idioma y el idioma de destino según el parámetro idioma. Finalmente, se devuelve el texto traducido obtenido de la respuesta del servicio Amazon Translate

```

1  def SubirS3(namefoto , foto):
2
3      load_dotenv()
4      s3 = boto3.resource(
5          's3',
6          aws_access_key_id=os.environ.get('ENV_AWS_ACCESS_KEY_ID'),
7          aws_secret_access_key=os.environ.get('ENV_AWS_SECRET_ACCESS_KEY'),
8          region_name=os.environ.get('ENV_AWS_REGION_NAME')
9      )
10
11     s3.Bucket(
12         os.environ.get('ENV_AWS_S3_BUCKET_NAME')
13     ).put_object(
14         Key=namefoto,
15         Body=foto,
16         ContentType='image'
17     )

```

Esta función SubirS3 permite cargar una imagen en Amazon S3. Primero, se configura el recurso S3 utilizando las credenciales y la región proporcionadas a través de variables de entorno. Luego, se utiliza el método put_object para cargar la imagen (foto) en el bucket de S3 especificado. La imagen se carga con un nombre definido por namefoto y se especifica el tipo de contenido como 'image'.

```
1 def TextToVoz(traductor , texto):
2     load_dotenv()
3     client = boto3.client(
4         'polly',
5         aws_access_key_id=os.environ.get('AWS_ACCESS_KEY_POLLY'),
6         aws_secret_access_key=os.environ.get('AWS_SECRET_ACCESS_KEY_POLLY'),
7         region_name=os.environ.get('AWS_REGION_NAME_POLLY')
8     )
9
10    response = client.synthesize_speech(
11        Text=texto,
12        OutputFormat='mp3',
13        VoiceId= traductor
14    )
15
16    body = response['AudioStream'].read()
17
18    with open('speech.mp3', 'wb') as file:
19        file.write(body)
20        file.close()
```

Esta función TextToVoz utiliza el servicio Amazon Polly para convertir un texto dado en voz. Primero, se configura el cliente de Polly utilizando las credenciales y la región proporcionadas a través de variables de entorno. Luego, se llama al método synthesize_speech para sintetizar el texto en voz, especificando el formato de salida como 'mp3' y el identificador de voz (VoiceId) que se pasa como parámetro. Finalmente, se lee el contenido del audio sintetizado y se guarda en un archivo llamado 'speech.mp3'.




```

1
2 def newAlbumntag (image_data):
3
4     response = rekognition.detect_labels(
5         Image={
6             'Bytes': image_data.read()
7         }
8     )
9     etiquetas = [label['Name'] for label in response['Labels']]
10    return etiquetas

```

Esta función newAlbumntag utiliza el servicio Amazon Rekognition para detectar etiquetas en una imagen dada. Recibe los datos de la imagen como entrada, los lee y los pasa al método detect_labels de Rekognition. Este método devuelve una respuesta que contiene una lista de etiquetas detectadas en la imagen. La función extrae y devuelve estas etiquetas como una lista.



```

1 import hashlib
2
3 def hash_password(password):
4
5     hash_md5 = hashlib.md5(password.encode())
6     return hash_md5.hexdigest()
7

```

Esta función hash_password utiliza el módulo hashlib de Python para calcular el hash MD5 de una contraseña dada. Primero, la contraseña se codifica en bytes utilizando el método encode(), luego se pasa al constructor md5() de hashlib para

crear un objeto de hash MD5. Finalmente, se llama al método `hexdigest()` para obtener la representación hexadecimal del hash y se devuelve como resultado.

```
1  def query(query , parms):
2      print (query)
3      connection = None
4      cursor = None
5      try:
6          load_dotenv()
7          connection = mysql.connector.connect(**db)
8          cursor = connection.cursor()
9          cursor.execute(query,parms)
10         results = cursor.fetchall()
11         idz = cursor.lastrowid
12         connection.commit()
13         return results , idz
14     except mysql.connector.Error as e:
15         print(e)
16         return None
17     finally:
18         if cursor:
19             cursor.close()
20         if connection and connection.is_connected():
21             connection.close()
```

Esta función `query` ejecuta una consulta en una base de datos MySQL utilizando los parámetros proporcionados. Primero, establece una conexión con la base de datos utilizando los valores de configuración almacenados en las variables de entorno. Luego, crea un cursor para ejecutar la consulta y pasa los parámetros a la consulta mediante el método `execute()`. Después de ejecutar la consulta, obtiene los resultados y el ID de la última fila insertada (si corresponde). Finalmente, confirma los cambios en la base de datos y devuelve los resultados y el ID de la fila insertada (si corresponde). Si se produce algún error durante el proceso, se imprime un mensaje de error y se devuelve `None`.

```

1  def conversa_bot(texto , idsesion):
2      load_dotenv()
3      print("en conversacion bot")
4      lexbot = boto3.client(
5          'lexv2-runtime',
6          aws_access_key_id=os.environ.get('AWS_ACCESS_KEY_LEX'),
7          aws_secret_access_key=os.environ.get('AWS_SECRET_ACCESS_KEY_LEX'),
8          region_name=os.environ.get('AWS_REGION_NAME_LEX')
9      )
10     print("conexion")
11     response = lexbot.recognize_text(
12         botId='TA0WVGJ87H',
13         botAliasId='TSTALIASID',
14         sessionId= idsesion,
15         localeId= 'es_US',
16         text= texto
17     )
18     #print("response before")
19     #print(response)
20     intent = response['sessionState']['intent']['name']
21     confirmationState = response['sessionState']['intent']['confirmationState']
22     slots = response['sessionState']['intent']['slots']
23     mensajeFinal = {
24         "contentType": "PlainText",
25         "content": f'Puedes hacer otra consulta si lo deseas.',
26     }
27
28     print(">>>", intent, confirmationState, slots)
29
30     if intent == "BuscarPorEstrellas":
31         if confirmationState == "Confirmed":
32             estrellas = slots['Estrellas']['value']['resolvedValues'][0]
33             print("Estrellas:", estrellas)
34
35             # buscar Lugares con esta cantidad de estrellas
36             lugares = busca_estrellas(estrellas)
37
38             # provisional
39             #lugares = [[1, "hotel1"], [2, "hotel2"], [3, "hotel3"]]
40
41             # mensaje inicial
42             response['messages'] = [{
43                 "contentType": "PlainText",
44                 "content": f'Los lugares con {estrellas} estrellas son:',
45             }]
46             # mensajes para cada Lugar
47             for i in lugares:
48                 response['messages'].append({
49                     "contentType": "PlainText",
50                     "content": f'{i[1]}',
51                 })
52             # mensaje end of chat
53             response['messages'].append(mensajeFinal)
54         elif intent=="Países":
55             if confirmationState == "Confirmed":
56                 #países = ["Pais1", "Pais2", "Pais3"]
57                 países = busca_países()
58                 # mensaje inicial
59                 response['messages'] = [{
60                     "contentType": "PlainText",
61                     "content": f'Los países mejor calificados son:',
62                 }]
63                 # mensajes para cada país
64                 for i in países:
65                     response['messages'].append({
66                         "contentType": "PlainText",
67                         "content": f'{i[0]} con {i[1]} estrellas',
68                     })
69                 # mensaje end of chat
70                 response['messages'].append(mensajeFinal)
71                 #response['sessionState']['intent']['state'] = "Fulfilled"
72             else:
73                 response['messages'] = [
74                     {"content": "No podemos atender tu solicitud, pero puedes:"},
75                     {"content": "Buscar lugares dependiendo de su numero de estrellas"},
76                     {"content": "Buscar los países mejor calificados"}
77                 ]
78
79     #print("mensajes en chatbot")
80     #print(response['messages'])
81     return response

```


Esta función `conversa_bot` permite interactuar con un bot utilizando el servicio Amazon Lex. Primero, se configura el cliente de Lexv2 utilizando las credenciales y la región proporcionadas a través de variables de entorno. Luego, se llama al método `recognize_text` para enviar el texto de entrada al bot y recibir la respuesta. Dependiendo del intento y el estado de confirmación obtenidos en la respuesta, se realizan diferentes acciones. Por ejemplo, si el intento es "BuscarPorEstrellas" y el estado de confirmación es "Confirmed", se busca lugares con la cantidad de estrellas especificada y se devuelve un mensaje con los resultados. Si el intento es "Paises" y el estado de confirmación es "Confirmed", se buscan los países mejor calificados y se devuelve un mensaje con los resultados. Si el intento no es reconocido, se devuelve un mensaje que proporciona opciones al usuario para continuar la conversación. Finalmente, se devuelve la respuesta del bot.