

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

		PRÁCTICA DE LABORATORIO	
CARRERA: INGENIERIA DE SISTEMA/COMPUTACION		ASIGNATURA: Inteligencia Artificial I	
NRO. PRÁCTICA:	2	TÍTULO PRÁCTICA: Informe Exámen	
OBJETIVO <ul style="list-style-type: none"> Consolidar los conocimientos adquiridos en clase sobre la IA en la nube (chatbot – IBM Watson). 			
<p>Ejercicios propuestos:</p> <p>Aunque existen muchos tipos de chatbots, si va a crear uno por primera vez, te recomendamos una de estas dos opciones:</p> <ul style="list-style-type: none"> Bots informativos <p>Tal como su nombre sugiere, estos bots proporcionan al usuario un nuevo formato para consumir información. Por ejemplo, los bots de noticias de último momento envían historias actuales a medida que se revela la información.</p> <ul style="list-style-type: none"> Bots de servicio <p>Estos bots están automatizados para completar tareas y responder preguntas. Dicho de otro modo, resuelven un problema o la inquietud de un usuario por medio de un chat. Tal vez estás pensando en bots de atención al cliente, pero cada vez hay más bots de servicio con fines como reservar citas o comprar en línea.</p> <p>En virtud de ello, algunas universidades han empleado el uso de chatbots para interactuar con los estudiantes o posibles candidatos, un ejemplo real es en la Universidad George Washington, después de poner a prueba su servicio de chatbot 24/7, MARTHA, el 89 % de los usuarios abogó por que la herramienta se convierta en un servicio permanente, entre otras más.</p> <p>En base a ello, se desea generar un chatbot informativo que sirva de soporte en la promoción de las carreras de grado(https://www.ups.edu.ec/es/web/guest/carreras-grado) de la Universidad Politécnica Salesiana, este chatbot deberá tener las siguientes características o servicios que brinde:</p> <ul style="list-style-type: none"> Inscripciones / Ficha Socio Economica Perfil de egreso Malla curricular Presentación Poder matricularse / inscripciones Informacion de los grupos de investigación pertenecientes a la carreras. Personal Docente Redes sociales Vida Estudiantil 			

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

- Contactanos
- Varios o información adicional
- Finalmente, utilizar al menos 3 servicios distintos de IBM Watson dentro del proyecto.

Problemática:

La Universidad Politécnica Salesiana, una de las múltiples instituciones ecuatorianas golpeadas por la pandemia mundial se ve en la necesidad de buscar nuevas formas de llegar a promocionar la oferta de carreras para el nuevo ciclo lectivo, para ello se pide a los estudiantes de Ingeniería de sistemas crear una aplicación para promocionar diferentes carreras que se ofertan dentro del campus de una forma innovadora, se aplicaran los conocimientos adquiridos en las materias de Sistemas Expertos e Inteligencia Artificial.


Propuesta de solución:

Para resolver la problemática, se plantea realizar una aplicación diseñada en Python, la cual consume servicios de IBM Watson, específicamente 3 los cuales detallaremos a continuación:

- Watson Assistant
- Watson Language Translator
- Watson Speech To Text

La aplicación tendrá un chatbot utilizando Watson Assistant, este bot responderá preguntas acerca de una de las carreras que oferta la universidad, la carrera de Electricidad. Así mismo podrá preguntar acerca del proceso de inscripciones, la malla curricular, el personal docente, los grupos de investigación entre otras opciones más. La siguiente parte que implementará la universidad será para estudiantes extranjeros, ya sea porque son de intercambio o porque vienen a vivir a Ecuador y aún tienen problemas con el idioma español, se implementará un menú donde la persona dirá su pregunta al micrófono en el idioma que domina (en este caso el inglés), la aplicación reconocerá la voz y la convertirá en texto. La última parte tomará este texto capturado y lo usará dentro del último servicio, el servicio de traductor de Watson se devolverá en pantalla una traducción al español de lo que quiso decir la persona, este pasará al chatbot y le devolverá la información que necesita, sin embargo, para esta opción, esta información será traducida para el usuario.

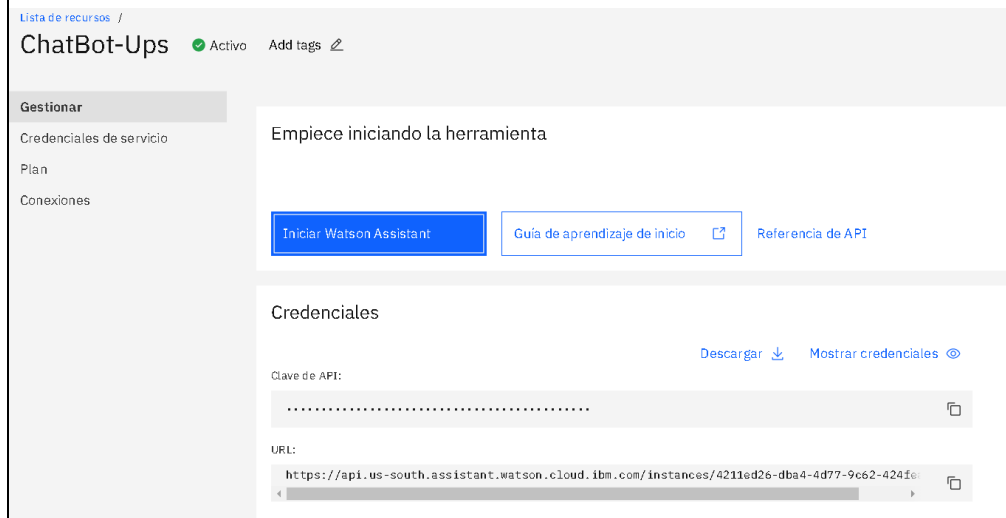
A continuación, se especifica el paso a paso de la construcción de la aplicación.

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

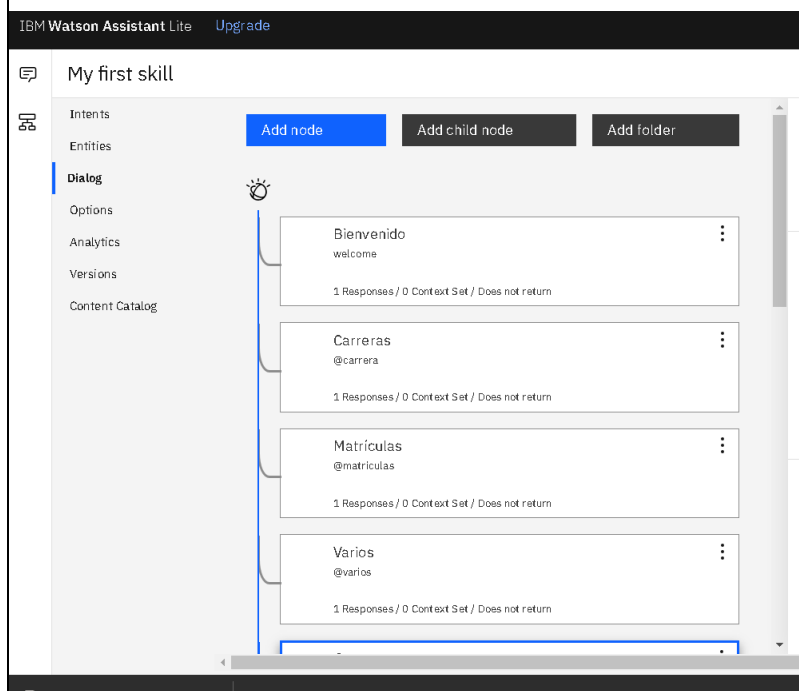
Desarrollo:


- Inteligencia Artificial

1. Generación de un ChatBot en ibm Watson Assistant: se crea el servicio dentro de los existentes en Watson y se inicia el Watson Assistant.



2. Generamos el diálogo que tendrá nuestro chatbot creando entidades, intentos y nodos que tendrán la información.



	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

3. Esta aplicación será consumirá el servicio de chatbot desde Python, sin embargo, estará conectado a Messenger donde también consumirá este servicio. Vamos a detallar ambas formas de consumo:

Captura de la voz por medio de la librería Pyaudio.

Esta función capturará la voz de la persona que necesita información (en inglés) y la guardará en formato .wav para ser usado en el siguiente método.

```
def voz():
    import pyaudio
    import wave

    FORMAT = pyaudio.paInt16
    CHANNELS = 2
    RATE = 44100
    CHUNK = 1024
    RECORD_SECONDS = 7
    WAVE_OUTPUT_FILENAME = "C:\\Users\\LENOVO\\Desktop\\Sonidos\\probando.wav"

    audio = pyaudio.PyAudio()

    # start Recording
    stream = audio.open(format=FORMAT, channels=CHANNELS,
                        rate=RATE, input=True,
                        frames_per_buffer=CHUNK)
    print ("recording begins..")
    frames = []

    for i in range(0, int(RATE / CHUNK * RECORD_SECONDS)):
        data = stream.read(CHUNK)
        frames.append(data)
    print ("finished recording")

    stream.stop_stream()
    stream.close()
    audio.terminate()

    waveFile = wave.open(WAVE_OUTPUT_FILENAME, 'wb')
    waveFile.setnchannels(CHANNELS)
    waveFile.setsampwidth(audio.get_sample_size(FORMAT))
    waveFile.setframerate(RATE)
    waveFile.writeframes(b''.join(frames))
    waveFile.close()
```

Servicio de voz a texto (speech to text)

Watson cuenta con un sencillo pero poderoso servicio de texto a voz que reconoce múltiples idiomas, la voz que fue capturada en el método anterior será enviada a este método para obtener la información de forma textual y trabajar con esta en el asistente de Watson.

```
def ServicioVozTexto():
    import json
    from os.path import join, dirname
    from ibm_watson import SpeechToTextV1
    from ibm_cloud_sdk_core.authenticators import IAMAuthenticator

    authenticator = IAMAuthenticator('RmV-HeURfV0wz8aWUKXRFnbEK3cf50cL2snrISOarrk8')
    speech_to_text = SpeechToTextV1(
        authenticator=authenticator
    )

    speech_to_text.set_service_url('https://api.us-south.speech-to-text.watson.cloud.ibm.com/instances/b8ac70ea-2ca3-4c95-9491-61372d')

    with open(join( "C:\\Users\\LENOVO\\Desktop\\Sonidos\\probando.wav"),
              'rb') as audio_file:
        speech_recognition_results = speech_to_text.recognize(
            audio=audio_file,
            content_type='audio/wav',
            word_alternatives_threshold=0.9,
            keywords=['colorado', 'tornado', 'tornadoes'],
            keywords_threshold=0.5
        ).get_result()
        #print(json.dumps(speech_recognition_results, indent=2))

    jsondecoded = json.loads(json.dumps(speech_recognition_results, indent=2))
    for entity in jsondecoded["results"]:

        for entityProperty in entity["alternatives"]:
            guardar = entityProperty["transcript"]
            #print("Propiedad " + entityProperty["transcript"] )

    print('You said: ', guardar)


    return guardar
```

Servicio de traductor (Watson Language Translate)

Este servicio traducirá la información obtenida en los métodos anteriores y permitirá enviarlas al asistente de Watson para que realice las acciones necesarias y devuelva la información que fue pedida al mismo.

El método traductor se usará más adelante para traducir de forma contraria, es decir, la primera vez se usa para traducir inglés - español, y la segunda vez para español – inglés.

```
def traductor():  
    import json  
    from ibm_watson import LanguageTranslatorV3  
    from ibm_cloud_sdk_core.authenticators import IAMAuthenticator  
  
    authenticator = IAMAuthenticator('mPTAi1x7n0mVZyio7UehAGmNwyixR-emINZYxvEzX0k4')  
    language_translator = LanguageTranslatorV3(  
        version='2018-05-01',  
        authenticator=authenticator  
    )  
  
    language_translator.set_service_url('https://api.us-south.language-translator.watson.cloud.ibm.com/instances/f9dd3377-792c-4d97-a1  
    translation = language_translator.translate(  
        text=ServicioVozTexto(),  
        model_id='en-es').get_result()  
    #print(json.dumps(translation, indent=2, ensure_ascii=False))  
  
    traducido = (translation.get("translations"))  
    for val in traducido:  
        #print(val["translation"])  
        datoEnviado = val["translation"]  
    print ("Text in spanish:" , datoEnviado)  
    return datoEnviado
```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

Implementación del ChatBot

El chatbot responderá a algunas preguntas comunes por los usuarios sobre temas como:

- Inscripciones / Ficha Socio Economica
- Perfil de egreso
- Malla curricular
- Presentación
- Poder matricularse / inscripciones
- Informacion de los grupos de investigación pertenecientes a la carreras.
- Personal Docente
- Redes sociales
- Vida Estudiantil
- Contactanos
- Varios o información adicional
-

OJO: La forma de conectarse es parecida a los otros servicios, tienen en común que se pasar las credenciales del servicio para iniciar la sesión. Los resultados serán presentados más adelante.

```
import json
from ibm_watson import AssistantV2
from ibm_cloud_sdk_core.authenticators import IAMAuthenticator

authenticator = IAMAuthenticator('btzGP76fH90liMxvnZfnimBTPGopc-ym32fZ2XNubGzk')
assistant = AssistantV2(
    version='2018-09-20',
    authenticator=authenticator)
assistant.set_service_url('https://api.us-south.assistant.watson.cloud.ibm.com/instances/4211ed26-dba4-4d77-9c62-424fea38972a')
assistant.set_disable_ssl_verification(False)
session = assistant.create_session("eelc54c4-c1d4-4da7-924e-871daf99054a").get_result()
print(json.dumps(session, indent=2))
print (session['session_id'])
message = assistant.message(
    "eelc54c4-c1d4-4da7-924e-871daf99054a",
    session['session_id'],
    input={
        'message_type': 'text',
        'text': 'y las inscripciones'
    }).get_result()
print(json.dumps(message, indent=2))
```

- Sistemas Expertos

1. Persistencia en la Base de Datos NEO4J

La base de datos se usará para persistir los datos para poder usarlos a futuro como información útil para la Universidad, de esta forma se puede trabajar con los algoritmos que implementa NEO4J.

Conexión con la base de datos

La conexión es directa con el localhost y sin autenticación.

```
from neo4j import GraphDatabase
|
uri="bolt://localhost:7687"
graphDB_Driver=GraphDatabase.driver(uri)
```

Creación de nuevos nodos a la base de datos

Aquí se realizará la persistencia de los datos, en una base previamente creada, se guardarán el nombre, el apellido y las carreras a las que tiene afinidad.

```
def crearNodo(nombre, acronimo):
    cqlCreate = """MERGE (""" + acronimo + """:Person {name: """ + nombre + """})"""

    with graphDB_Driver.session() as graphDB_Session:
        graphDB_Session.run(cqlCreate)

def crearRelacion(nombre, carrera1, carrera2):
    cqlCreate = """MATCH (a:Person) where a.name= """ + nombre + """,
        Match (b:Carrera) where b.name= """ + carrera1 + """,
        Match (c:Carrera) where c.name= """ + carrera2 + """,
        MERGE (a)-[:LIKES]->(b)
        MERGE (a)-[:LIKES]->(c)"""

    with graphDB_Driver.session() as graphDB_Session:
        graphDB_Session.run(cqlCreate)
```


Listado de los datos de la base de datos

En esta parte se visualizarán los datos de los usuarios en la base.

```
cqlNodeQuery = "Match (n) return n"
with graphDB_Driver.session() as graphDB_Session:
    nodes = graphDB_Session.run(cqlNodeQuery)
    for nodo in nodes:
        print(nodo)
```

```
<Record n=<Node id=0 labels=frozenset({'Carrera'}) properties={'name': 'Derecho'}>>
<Record n=<Node id=1 labels=frozenset({'Carrera'}) properties={'name': 'Literatura'}>>
<Record n=<Node id=2 labels=frozenset({'Carrera'}) properties={'name': 'Artes'}>>
<Record n=<Node id=3 labels=frozenset({'Carrera'}) properties={'name': 'Cultura Físc.
<Record n=<Node id=4 labels=frozenset({'Carrera'}) properties={'name': 'Administraci
<Record n=<Node id=5 labels=frozenset({'Carrera'}) properties={'name': 'Psicología'}>
<Record n=<Node id=6 labels=frozenset({'Person'}) properties={'name': 'David'}>>
<Record n=<Node id=7 labels=frozenset({'Person'}) properties={'name': 'Cristian'}>>
<Record n=<Node id=8 labels=frozenset({'Person'}) properties={'name': 'Joss'}>>
<Record n=<Node id=9 labels=frozenset({'Person'}) properties={'name': 'Bryan'}>>
```

Entrenamiento de ANN


El objetivo del entrenamiento ANN, es para ayudar al usuario a tener más opciones de carreras que podría seguir según sus gustos, por ejemplo: un estudiante quiere preguntar sobre la carrera de Electricidad, nuestro asistente al finalizar le hará unas pequeñas preguntas como: ¿qué otra carrera te gustaría seguir aparte de esta?, entonces la consulta en NEO4J le mostrará otras posibles carreras afines a las que ya eligió el usuario como recomendación, estas recomendaciones se basan en las materias que están por detrás en cada carrera, es decir, tenemos materias que pueden estar en ingeniería como también en derecho o administración, se hace una comparativa y se le recomienda.

Entrenamiento

```
def entrenamiento1():
    cqlNodeQuery = """MATCH (p:Person)-[:LIKES]->(Carrera)
    WITH {item:id(p), categories: collect(id(Carrera))} AS userData
    WITH collect(userData) AS data
    CALL gds.alpha.ml.ann.stream({
        nodeProjection: '*',
        relationshipProjection: '*',
        data: data,
        algorithm: 'jaccard',
        similarityCutoff: 0.1,
        concurrency: 1
    })
    YIELD item1, item2, similarity
    return gds.util.asNode(item1).name AS from, gds.util.asNode(item2).name AS to, similarity
    ORDER BY from"""
    with graphDB_Driver.session() as graphDB_Session:
        graphDB_Session.run(cqlNodeQuery)
```

```
def entrenamiento2():
    cqlNodeQuery = """
    MATCH (p:Person)-[:LIKES]->(Carrera)
    WITH {item:id(p), categories: collect(id(Carrera))} AS userData
    WITH collect(userData) AS data
    CALL gds.alpha.ml.ann.write({
        nodeProjection: '*',
        relationshipProjection: '*',
        algorithm: 'jaccard',
        data: data,
        similarityCutoff: 0.1,
        showComputations: true,
        concurrency: 1
    })
    YIELD nodes, similarityPairs, writeRelationshipType, writeProperty, min, max, mean, p95
    RETURN nodes, similarityPairs, writeRelationshipType, writeProperty, min, max, mean, p95"""
    with graphDB_Driver.session() as graphDB_Session:
        graphDB_Session.run(cqlNodeQuery)
```


```
def carrerasAfines(nombre):
    cqlNodeQuery = """MATCH (p:Person {name:'"' + nombre + '"'})-[:SIMILAR]->(other),
    (other)-[:LIKES]->(Carrera)
    WHERE not((p)-[:LIKES]->(Carrera))
    RETURN Carrera.name AS Carreras , count(*) AS Afinidad
    ORDER BY Afinidad DESC"""
    with graphDB_Driver.session() as graphDB_Session:
        nodes = graphDB_Session.run(cqlNodeQuery)
        for nodo in nodes:
            print(nodo)
```

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

2. Interfaz gráfica de la aplicación

Esta interfaz está diseñada en PyQt5, es sencilla e intuitiva para el usuario, desde ella podrá acceder a los 3 servicios que brindamos y llamar a nuestra página en Facebook, donde podrá interactuar con el bot usando solo el servicio de chat.



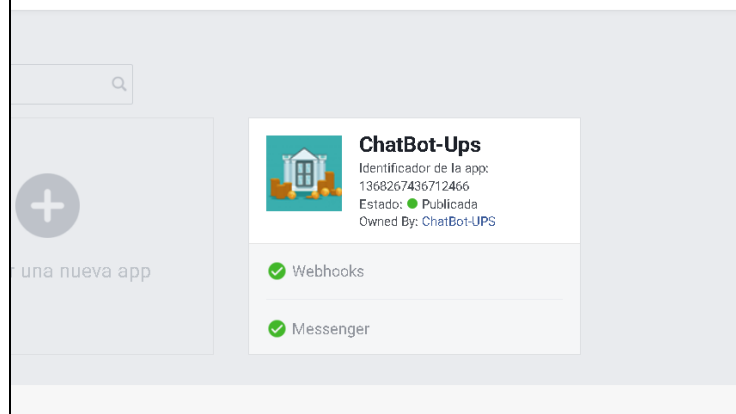
	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

3. Conexión a las Redes Sociales:

Por el momento solo contamos con el servicio de chat de Messenger consumiendo el servicio de asistente de Watson, sin embargo, se planea integrar los demás servicios a futuro.

- Se debe crear una aplicación en Facebook para desarrolladores e integrar el servicio de Watson.

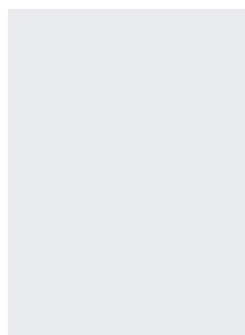
t have any required action items to display. If any of your apps need immediate attention in the future, an it



PRODUCTOS (+)

- ✓ Webhooks
- ✓ Messenger

Registro de actividad




ChatBot-Ups

Identificador de la app: 1368267436712466

0% del límite usado

[Ver detalles](#)

100% restante

	VICERRECTORADO DOCENTE	Código: GUIA-PRL-001
	CONSEJO ACADÉMICO	Aprobación: 2016/04/06
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		

- Luego de publicar la aplicación se puede interactuar con el ChatBot.



Conclusiones:

La herramienta Watson permitió desarrollar la aplicación optimizando el trabajo del programador a un gran nivel, el consumo de servicios de Watson fue rápido, óptimo y sencillo de implementar, asimismo consumimos servicios web desde una aplicación de escritorio, conectamos los servicios con una red social (Messenger) y logramos crear una aplicación bastante completa.

Recomendaciones:

Se recomienda realizar talleres a futuro sobre desarrollo de aplicaciones empresariales usando este tipo de servicios que brinda Watson.

Estudiante: David Egas.