TRABAJO PRÁCTICO INTEGRADOR

IA3.5 Redes de Datos

Tecnicatura Universitaria en Inteligencia Artificial

Tomás Navarro Miñón

Mateo Gravi Fiorino

17/06/2023

Universidad Nacional de Rosario

Facultad de Ciencias Exactas, Ingeniería y Agrimensura

INTRODUCCIÓN

En este trabajo realizamos una API la cual tiene varios métodos que consultan a una base de datos JSON

El primer método que tenemos se llama ***'pokemon':*** el mismo se le pasa un número de pokémon y la API devuelve: todas las estadísticas y la imagen del pokemon imitando lo que en la serie se conoce como una ***"pokédex"*** que es una herramienta que usan los personajes para saber sobre los pokemons en su mundo ficticio.
En segundo lugar tenemos el método ***'top5' :*** que devuelve los 5 pokémons más fuertes del universo pokémon.
El tercer método se llama ***'tipos':*** este método abre una ventana donde hay un gráfico de todos los tipos de pokémon y su cantidad.
También ofrecemos un método ***'put':*** que sirve para modificar el nombre de un pokémon. Esto lo utilizaremos más que nada a la hora de realizar traducciones.
Este trabajo lo realizamos sobre la base de datos de kaggle: "Pokemon Stats and Description with pictures (JSON)". La misma es una base de datos JSON donde se encuentra ID, nombre, tipo, altura, peso, etc.
El mismo fue realizado en el sistema operativo windows.

DESCRIPCIÓN DEL ENTORNO DE TRABAJO DEL SERVIDOR

El entorno de trabajo con el servidor fue la utilización de los framework uvicorn y fastAPI con el cual se realizó la conexión. Emulamos con 2 computadoras conectadas en la misma red para emular una respuesta servidor-cliente donde pudimos conectar el cliente al servidor. El lenguaje de programación utilizado fue python 3.10 y sus frameworks: FastAPI y json. También utilizamos el framework de matplotlib, que nos ayuda a realizar gráficos en Python.

DESCRIPCIÓN DEL ENTORNO DE TRABAJO DEL DEL CLIENTE

A la hora de armar el entorno nos encontramos con diversos problemas, la computadora que hacía de host, no podía correr uvicorn por una cuestión de permisos en el sistema ya que tiene windows 11. Por esto terminó haciendo de cliente. Problemas con el PATH, no reconocía donde estaba la carpeta de python, etc.

Tras su solución pudimos conectar el cliente con el servidor y de esta manera tener éxito con la respuesta que programamos. La programación fue realizada en python 3.10 con los frameworks: webbrowser y request que sirven para buscar por internet (utilizado en algunas imágenes) y el request para realizar pedidos al servidor respectivamente.

Cabe destacar que las imágenes las tomamos de la API de pokemon de su página web. El funcionamiento es muy similar al que habíamos planteado para el servidor local, con el beneficio de que las imágenes ya estaban cargadas ahí, aprovechando esta situación solamente nos quedaba conectar nuestra API con la de pokémon (web oficial)

DESCRIPCIÓN DE LA BASE DE DATOS ELEGIDA

La base de datos elegida se puede encontrar en el siguiente link:

https://www.kaggle.com/datasets/rajgaurav9/pokemon-stats-and-description-with-pictures-json?resource=download

La misma está compuesta por: 800 pokemons donde los mismos poseen los atributos de: 'id', 'name', 'species', 'type', 'height', 'weight', 'abilities', 'stats', 'evolution', 'description', 'gen'.

Id: Número del Pokémon en Pokédex
name: Nombre del Pokémon
species: Especie del Pokémon
type: Tipo del Pokémon
height: Altura en metros
weight: Peso en KG
abilities: Habilidades que tiene el Pokémon
stats: Estadísticas del Pokémon
evolution : Siguiente evolución del Pokémon (Si posee)
description: Descripción del Pokémon
gen: generación del pokémon (1era gen, 2da, etc)

PROCEDIMIENTO

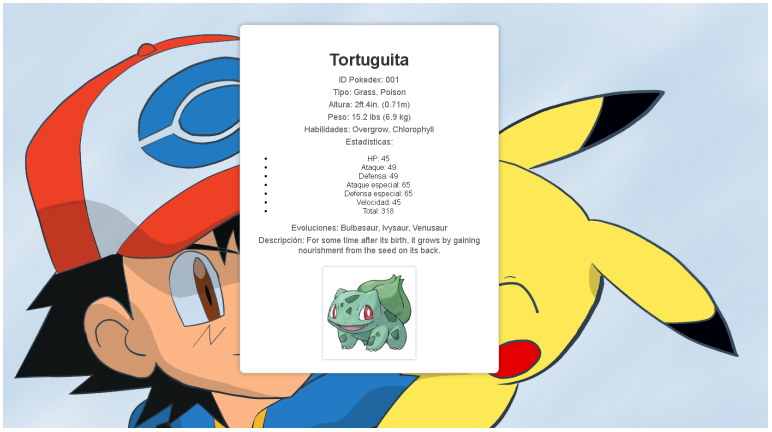
Primero deberemos ejecutar el código del cliente, a continuación, por consola, se nos pedirá que elijamos una “pregunta” para hacerle el servidor, se vería de la siguiente forma:

```
Nombre del Metodo a utilizar:
 -pokemon (Muestra estadisticas de un Pokemon)
 -top5 (Muestra los 5 Pokemons mas fuertes)
 -tipos (Grafico de los tipos con mas Pokemons)
 -put (Modifica el Nombre de un Pokemon)
 Metodo: pokemon
```

Una vez elegido un método, deberemos ingresar ciertos datos, dependiendo de lo que hayamos elegido. Por ejemplo si escribimos pokemon se verá lo siguiente:

```
Escribe el ID de un Pokemon: 001
Página HTML generada correctamente.
```

A continuación se abrirá una página como esta:



REFERENCIAS

1. https://stackoverflow.com/questions/65427501/how-to-send-a-picture-from-server-to-client-in-python
2. https://www.youtube.com/watch?v=TMWWpjFXiCc&t=584s
3. https://www.w3big.com/es/python/att-string-zfill.html
4. Material de cátedra

APÉNDICE A

Código del servidor:

```python
from fastapi import FastAPI, HTTPException
from fastapi.responses import JSONResponse
import json
import matplotlib.pyplot as plt

app = FastAPI()

# Ruta al archivo JSON
JSON_FILE_PATH = "pokemon_full.json"

# Cargar datos del archivo JSON
def load_pokemon_data():
    with open(JSON_FILE_PATH, "r") as json_file:
        return json.load(json_file)

def save_pokemon_data(data):
    with open(JSON_FILE_PATH, "w") as json_file:
        json.dump(data, json_file, indent=4)
```

```python
pokemon_data = load_pokemon_data()

@app.get("/pokemon/{pokemon_id}")
def get_pokemon(pokemon_id: int):
    # Obtener la información del Pokémon del archivo JSON
    pokemon_info = pokemon_data[pokemon_id - 1]

    # Obtener la ruta de la imagen del Pokémon
    image_path = get_image_path(pokemon_id)

    # Agregar la ruta de la imagen al diccionario de información del
Pokémon
    pokemon_info["image_path"] = image_path

    # Devolver la información del Pokémon como respuesta JSON
    return JSONResponse(content=pokemon_info, status_code=200)

@app.put("/put/{pokemon_id}")
def update_pokemon_name(pokemon_id: int, updated_data: dict):
    updated_name = updated_data.get("updated_name")
    # Verificar si el ID del Pokémon es válido
    if pokemon_id < 1 or pokemon_id > len(pokemon_data):
        raise HTTPException(status_code=404, detail="El ID del Pokémon
no es válido")

    # Actualizar el nombre del Pokémon en la lista de datos en memoria
    pokemon_data[pokemon_id - 1]["name"] = updated_name

    # Guardar los datos actualizados en el archivo JSON
    save_pokemon_data(pokemon_data)

    # Obtener la información actualizada del Pokémon
    updated_pokemon_info = pokemon_data[pokemon_id - 1]

    # Obtener la ruta de la imagen del Pokémon
    image_path = get_image_path(pokemon_id)

    # Agregar la ruta de la imagen al diccionario de información del
Pokémon
    updated_pokemon_info["image_path"] = image_path

    # Devolver la información actualizada del Pokémon como respuesta
JSON
```

```
    return JSONResponse(content=updated_pokemon_info, status_code=200)

@app.get("/plot")
def plot_pokemon_types():
    # Contar la cantidad de Pokémon por tipo
    type_counts = {}
    for pokemon in pokemon_data:
        pokemon_types = pokemon["type"]
        for pokemon_type in pokemon_types:
            if pokemon_type in type_counts:
                type_counts[pokemon_type] += 1
            else:
                type_counts[pokemon_type] = 1

    # Generar el gráfico de barras
    types = list(type_counts.keys())
    counts = list(type_counts.values())

    plt.bar(types, counts)
    plt.xlabel("Tipo de Pokémon")
    plt.ylabel("Cantidad")
    plt.title("Cantidad de Pokémon por Tipo")
    plt.tick_params(axis='x', rotation=90)
    plt.show()

@app.get("/strongest")
def get_strongest_pokemon():
    # Ordenar los Pokémon por su estadística de ataque (attack) en
orden descendente
    sorted_pokemon = sorted(pokemon_data, key=lambda p:
p["stats"]["attack"], reverse=True)

    # Obtener los 5 Pokémon más fuertes
    strongest_pokemon = sorted_pokemon[:5]

    # Devolver los 5 Pokémon más fuertes como respuesta JSON
    return JSONResponse(content=strongest_pokemon, status_code=200)

def get_image_path(pokemon_id: int) -> str:
    pokemon_id_str = str(pokemon_id).zfill(3)
    return f'{pokemon_id_str}.png'
```

(PD: se ve asi porque lo copie y pegue de visual studio code)

APÉNDICE B

```
import webbrowser
import requests
# URL del servidor FastAPI
SERVER_URL = "http://192.168.0.24:8080"
# Archivo HTML para el formulario
HTML_FILE_PATH = "form.html"
# Cargar el archivo HTML del formulario
def load_html_file(file_path):
    with open(file_path, "r") as file:
        return file.read()
# Ejecutar la solicitud al servidor FastAPI
def execute_api_request(method):
    if method == "pokemon":
        pokemon_id = input("Escribe el ID de un Pokemon: ")
        response = requests.get(f"{SERVER_URL}/pokemon/{pokemon_id}")
        pokemon_info = response.json()
        image_path =
f"https://assets.pokemon.com/assets/cms2/img/pokedex/full/{pokemon_id}.
png"
        html = generate_html_response(pokemon_info, image_path)
        with open("pokemon.html", "w") as html_file:
            html_file.write(html)
        print("Página HTML generada correctamente.")
        webbrowser.open("pokemon.html")
```

```
    elif method == "tipos":
        response = requests.get(f"{SERVER_URL}/plot")
        print("Plot generated.")
    elif method == "top5":
        response = requests.get(f"{SERVER_URL}/strongest")
        strongest_pokemon = response.json()

        # Obtener los 5 primeros Pokémon más fuertes
        top5_pokemon = strongest_pokemon[:5]
        # Generar el HTML del top 5 de Pokémon
        html = generate_top5_html_response(top5_pokemon)
        with open("top5.html", "w") as html_file:
            html_file.write(html)
        print("Página HTML generada para el top 5 de Pokémon más
fuertes.")
        webbrowser.open("top5.html")
    elif method == "put":
        pokemon_id = input("Escribe el ID de un Pokemon: ")
        updated_name = input("Nuevo nombre para el Pokemon: ")
        # Crear un diccionario con los datos actualizados del Pokémon
        updated_data = {"updated_name": updated_name}
        response = requests.put(f"{SERVER_URL}/put/{pokemon_id}",
json=updated_data)
        if response.status_code == 200:
            print("Pokémon updated successfully.")
        else:
            print("Failed to update Pokémon.")
    else:
```

```
        print("Invalid method.")

# Generar la respuesta HTML con la información del Pokémon
def generate_html_response(pokemon_info: dict, image_path: str) -> str:
    css_styles = """
    <style>
    body {
        font-family: 'Arial', sans-serif;
        background-color: #f0f0f0;
        background-image:
url('https://images5.alphacoders.com/109/1092473.png');
        background-size: cover;
        background-position: center;
        text-align: center;
        margin-top: 50px;
    }
    .pokemon-card {
        display: inline-block;
        border: 1px solid #ccc;
        border-radius: 10px;
        box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.3);
        padding: 30px;
        background-color: #fff;
        max-width: 500px;
        margin: 0 auto;
    }
    h1 {
        font-size: 36px;
```

```
        color: #333;
        margin-bottom: 10px;
    }
    p {
        font-size: 18px;
        color: #666;
        margin: 5px;
    }
    .pokemon-image {
        width: 200px;
        height: auto;
        margin-top: 20px;
        box-shadow: 0px 0px 5px rgba(0, 0, 0, 0.3);
    }
    </style>
    """
    # Formatear el tipo del Pokémon sin corchetes y comillas
    pokemon_type = ", ".join(pokemon_info['type'])
    # Obtener las estadísticas del Pokémon
    stats = pokemon_info['stats']
    # Obtener las evoluciones del Pokémon
    evolutions = pokemon_info['evolution']
    html_response = f"""
    <html>
    <head>
        <title>{pokemon_info['name']} - Pokedex</title>
        {css_styles}
```

```
    </head>
    <body>
        <div class="pokemon-card">
            <h1>{pokemon_info['name']}</h1>
            <p>ID Pokedex: {pokemon_info['id']}</p>
            <p>Tipo: {pokemon_type}</p>
            <p>Altura: {pokemon_info['height']}</p>
            <p>Peso: {pokemon_info['weight']}</p>
            <p>Habilidades: {', '.join(pokemon_info['abilities'])}</p>
            <p>Estadísticas:</p>
            <ul>
                <li>HP: {stats['hp']}</li>
                <li>Ataque: {stats['attack']}</li>
                <li>Defensa: {stats['defense']}</li>
                <li>Ataque especial: {stats['sp.atk']}</li>
                <li>Defensa especial: {stats['sp.def']}</li>
                <li>Velocidad: {stats['speed']}</li>
                <li>Total: {stats['total']}</li>
            </ul>
            <p>Evoluciones: {', '.join(evolutions)}</p>
            <p>Descripción: {pokemon_info['description']}</p>
            <img src="{image_path}" class="pokemon-image">
        </div>
    </body>
    </html>
    """
    return html_response
```

```
def generate_top5_html_response(pokemon_list):
    css_styles = """
    <style>
    body {
        font-family: 'Arial', sans-serif;
        background-color: #f0f0f0;
        background-image:
url('https://images5.alphacoders.com/109/1092473.png');
        background-size: cover;
        background-position: center;
        text-align: center;
        margin-top: 50px;
    }

    .pokemon-card {
        display: inline-block;
        border: 1px solid #ccc;
        border-radius: 10px;
        box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.3);
        padding: 30px;
        background-color: #fff;
        max-width: 500px;
        margin: 0 auto;
    }
    h1 {
        font-size: 36px;
        color: #333;
```

```
        margin-bottom: 10px;
    }
    p {
        font-size: 18px;
        color: #666;
        margin: 5px;
    }
    .pokemon-image {
        width: 200px;
        height: auto;
        margin-top: 20px;
        box-shadow: 0px 0px 5px rgba(0, 0, 0, 0.3);
    }
    </style>
    """
    pokemon_cards = ""
    for i, pokemon in enumerate(pokemon_list, start=1):
        image_path =
f"https://assets.pokemon.com/assets/cms2/img/pokedex/full/{pokemon['id'
]}.png"
        pokemon_card = f"""
        <div class="pokemon-card">
            <h1>{pokemon['name']}</h1>
            <img src="{image_path}" class="pokemon-image">
            <p>Puesto: {i}</p>
        </div>
        """
        pokemon_cards += pokemon_card
```

```
    html_response = f"""
    <html>
    <head>
        <title>Top 5 Pokémon más fuertes - Pokedex</title>
        {css_styles}
    </head>
    <body>
        <h1>Top 5 Pokémon más fuertes</h1>
        {pokemon_cards}
    </body>
    </html>
    """
    return html_response
if __name__ == "__main__":
    method = input("Nombre del Metodo a utilizar: \n -pokemon (Muestra estadisticas de un Pokemon) \n -top5 (Muestra los 5 Pokemons mas fuertes) \n -tipos (Grafico de los tipos con mas Pokemons) \n -put (Modifica el Nombre de un Pokemon) \n Metodo: ")
    execute_api_request(method)
```