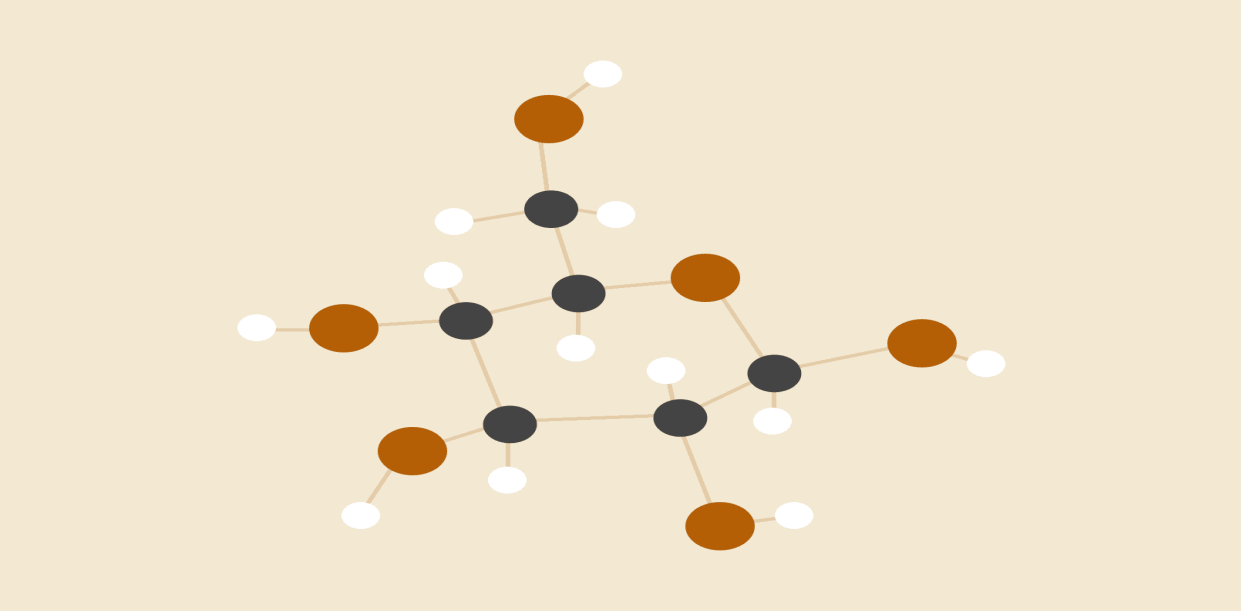
TRABAJO PRÁCTICO INTEGRADOR

IA3.5 Redes de Datos

Tecnicatura Universitaria en Inteligencia Artificial



Longo, Bruno y Donnarumma, César

02/07/2023

Universidad Nacional de Rosario

Facultad de Ciencias Exactas, Ingeniería y Agrimensura

INTRODUCCIÓN

Para el trabajo final de la materia hemos realizado una comunicación API entre servidor y cliente, con el propósito de manipular la base de datos de un catálogo de libros formato .json. Tanto el desarrollo del cliente API como del servidor API se realizaron con el sistema operativo Windows 10 con entorno físico.

DESCRIPCIÓN DEL ENTORNO DE TRABAJO DEL SERVIDOR

Para el desarrollo del servidor se utilizó el entorno de desarrollo integrado (IDE) Visual Studio Code (VSC), exclusivamente con lenguaje Python en su versión 3.10.4. Se utilizaron las librerías FastApi, os, Pandas y uvicorn.

La manipulacion de los objetos del .json se hizo a traves de la librería Pandas conviertiendolos a dataframe y luego se exporto el dataframe nuevamente a .json.

Los inconvenientes que se encontraron estuvieron relacionados más que nada a la hora de exportar el dataframe a .json con la inclusión de caracteres de escape que Pandas agregaba, pero nada que fuera muy difícil de sortear.

DESCRIPCIÓN DEL ENTORNO DE TRABAJO DEL CLIENTE

Para el desarrollo del cliente se utilizó el entorno de desarrollo integrado (IDE) Visual Studio Code (VSC), exclusivamente con lenguaje Python en su versión 3.10.4. Se utilizaron las librerías request, json y tkinter.

Los mayores inconvenientes fueron en torno a la manipulación del tipo de datos. Durante la visualización existieron modificaciones realizadas por funciones que no eran las deseadas y hubo que especificar en varios puntos la reconversión de los datos a .json, a fin de manipular fácilmente la base de datos como un diccionario.

Por otro lado, la interfaz al usuario a partir de la librería tkinter también propuso desafíos para su correcta y comprensible implementación. Por ejemplo, a la hora de que los prompts fueran claros.

DESCRIPCIÓN DE LA BASE DE DATOS ELEGIDA

Se utilizó la base de datos “books.json” provista por la cátedra, proveniente originalmente de 100-best-books/books.json at master · benoitvallon/100-best-books (github.com).

Contiene una lista con 100 elementos de tipo diccionario, cada uno con 8 pares key-value. Las 8 keys son:

- author : str, con el nombre del autor del libro.
- country : str, con el nombre del país originario.
- imageLink : str, con la url de la imagen .png.

- language : str, con la especificación del idioma.
- link : str, con un link.
- pages : int, con la cantidad de páginas.
- title : str, con el título.
- year : int, con el año de publicación.

PROCEDIMIENTOS

Los siguientes procedimientos fueron testeados a partir de http://localhost:8000/docs#/ gracias a la documentación generada por Fast API en Swagger UI.

- Método GET para obtener la lista completa.

```
request.get(f"http://127.0.0.1:8000/libreria_entera")
```

- Método GET para obtener un libro según su key y caracteres de valor.

```
request.get(f"http://127.0.0.1:8000//buscar_libro?columna=KEY&patron_busqueda=PATRON")
```

- Método POST para agregar un libro según sus valores respectivos a cada key.

```
request.post(f"http://127.0.0.1:8000//agregar_libro?autor={autor}&pais={pais}&link_imagen={link_imagen}&idioma={idioma}&link={link}&paginas={paginas}&titulo={titulo}&a%C3%B1o={anio}")
```

- Método PUT para modificar un libro según su título y valores.

```
request.put(f"http://127.0.0.1:8000//modificar_libro?eleccion={titulo_cambiar}&autor={autor}&pais={pais}&link_imagen={link_imagen}&idioma={idioma}&link={link}&paginas={paginas}&titulo={titulo}&a%C3%B1o={anio}”)
```

- Método DELETE para eliminar un libro a partir de su valor title.

```
request.delete(f"http://127.0.0.1:8000//eliminar_libro?eleccion={titulo_eliminar}")
```

INSTRUCCIONES DE USO DEL CLIENTE

Se requiere disponer de un entorno con las librerías mencionadas en la sección Cliente API y una versión actualizada de Python.

En caso de que el servidor no esté en el mismo dispositivo que el cliente, pero pertenezcan a la misma red local, es necesario modificar y agregar al inicio del código cliente:

```
server_ip = ""        #IP DEL SERVIDOR
```

Al ejecutar cliente_api_libreria.py, podrá seleccionar las funciones a partir de botones y líneas de texto a completar según las necesidades de su operación, mencionadas en cada ventana.

APÉNDICE A

SERVIDOR API

```
from fastapi import FastAPI

import pandas as pd

import os

import uvicorn


#desde el inicio host público


if __name__ == "__main__":

    uvicorn.run("APIbooks:app", host="0.0.0.0", port=8000)




app = FastAPI()


url = r"C:\Users\bruno\Downloads\books.json"    #CAMBIAR PARA QUE SE
```

```
UBIQUE EN EL SERVER


@app.get("/libreria_entera")
def libreria_entera():

    ''' Metodo GET que devuelve todos los libros de la libreria. '''

    libros = cargar_libreria()

    # Pasamos a formato .json para poder mandarlo en el return
    libros = libros.to_json(orient='records')

    return libros


@app.get("/buscar_libro")
def buscar_libro(columna: str, patron_busqueda : str):

    '''
    Metodo GET para buscar un determinado libro por: Autor, Pais, Idioma,
Titulo, Año.
    Busca si el patron esta contenido en la cadena.
    Recibe dos paremtros, columna es el atributo del .json, y
patron_busqueda es el
    patron que se busca.
    Ej:
http://localhost:8000/buscar_libro?columna=author&patron_busqueda=ch
```

```
    busca los autores que contengan en alguna parte de su nombre el patron
'ch'.
    '''

    libros = cargar_libreria()

    if columna == 'author':
        return busqueda_por_patron(libros, patron_busqueda, columna)
    elif columna == 'country':
        return busqueda_por_patron(libros, patron_busqueda, columna)
    elif columna == 'language':
        return busqueda_por_patron(libros, patron_busqueda, columna)
    elif columna == 'title':
        return busqueda_por_patron(libros, patron_busqueda, columna)
    elif columna == 'year':
        return busqueda_por_patron(libros, patron_busqueda, columna)
    else:
        return 'Opcion no valida'



@app.post("/agregar_libro")
def agregar_libro(autor: str, pais : str, link_imagen: str, idioma: str,
link: str, paginas: str, titulo: str,
                  año: str):

    '''
```

```
    Metodo POST que sirve para agregar un nuevo libro. Recibe los datos de
los atributos que tendra

    el nuevo libro y lo crea.

    '''



    libros = cargar_libreria()



    nuevo_libro = {'author': autor, 'country': pais, 'imageLink':
link_imagen, 'language': idioma,

                   'link': link, 'pages': paginas, 'title': titulo,
'year': año}



    # Coloca el nuevo libro en la ultima posicion del df.

    libros.loc[len(libros)] = nuevo_libro



    os.getcwd()



    # Exporta a .json sobreescribiendo la base de datos. orient='records'
cada fila del df es un objeto JSON

    # independiente

    libros.to_json('books.json', orient='records')



    return 'Libro agregado existosamente'




@app.put("/modificar_libro")
```

```
def modificar_libro(eleccion: str, autor: str, pais : str, link_imagen:
str, idioma: str, link: str,
                    paginas: str, titulo: str, año: str):


    '''
    Metodo PUT que modifica un libro a partir de un titulo, el parametro
eleccion. Este, debe coincidir
    exactamente con el titulo del libro en cuestion.
    Tambien recibe los nuevos valores de TODOS los atributos del libro,
para ser modificados.
    '''


    libros = cargar_libreria()


    libro_modificado = {'author': autor, 'country': pais, 'imageLink':
link_imagen, 'language': idioma,
                    'link': link, 'pages': paginas, 'title': titulo,
'year': año}


    # Verifica si existe al menos una fila con ese titulo.
    if libros['title'][ libros['title'] == eleccion ].empty == False:


        # Me quedo con la primera fila (podrian existir mas de una)
        fila = libros[ libros['title'] == eleccion ].iloc[0]


        # Busco su indice. Como ya la tengo identificada, en formato
series, lo paso a dataframe,
```

```
        # cambio filas por columnas para que vuelva a tener el indice en
donde va y elijo obtener
        # el indice del elemento que esta en la posicion 0.
        indice = fila.to_frame().transpose().index[0]

        # Modifico a partir del indice los diferentes atributos del libro.
        libros.loc[indice, 'author'] = libro_modificado['author']
        libros.loc[indice, 'country'] = libro_modificado['country']
        libros.loc[indice, 'imageLink'] = libro_modificado['imageLink']
        libros.loc[indice, 'language'] = libro_modificado['language']
        libros.loc[indice, 'link'] = libro_modificado['link']
        libros.loc[indice, 'pages'] = libro_modificado['pages']
        libros.loc[indice, 'title'] = libro_modificado['title']
        libros.loc[indice, 'year'] = libro_modificado['year']

        os.getcwd()

        # Sobreeescribo el archivo, una fila un objeto.
        libros.to_json('books.json', orient='records')

        return 'Libro modificado existosamente'

    # Si no existe ninguna fila con ese titulo..
    else:
```

```
        return 'El libro ' + str(eleccion) + ' no existe en la base de
datos'


@app.delete("/eliminar_libro")
def eliminar_libro(eleccion: str):


    '''
    Metodo DELETE que eliminar un libro a partir de un titulo, el
parametro eleccion. Este, debe coincidir
    exactamente con el titulo del libro en cuestion.
    '''


    libros = cargar_libreria()


    # Verifica si existe al menos una fila con ese titulo.
    if libros['title'][ libros['title'] == eleccion ].empty == False:


        # Me quedo con la primera fila (podrian existir mas de una)
        fila = libros[ libros['title'] == eleccion ].iloc[0]


        # Busco su indice. Como ya la tengo identificada, en formato
series, lo paso a dataframe,
        # cambio filas por columnas para que vuelva a tener el indice en
donde va y elijo obtener
        # el indice del elemento que esta en la posicion 0.
        indice = fila.to_frame().transpose().index[0]
```

```
        # Elimino el archivo en cuestion
        libros = libros.drop(indice)

        os.getcwd()

        # Sobreeescribo el archivo, una fila un objeto.
        libros.to_json('books.json', orient='records')

        return 'Libro eliminado existosamente'

    # Si no existe ninguna fila con ese titulo..
    else:

        return 'El libro ' + str(eleccion) + ' no existe en la base de
datos'


#-------------------------------------------------------------------------
---------------------------------------
# Funciones auxiliares:
#-------------------------------------------------------------------------
---------------------------------------

def cargar_libreria():

    ''' Funcion auxiliar utilizada para cargar la libreria cada vez que se
```

```
llama algun metodo'''

    # Usamos pandas para leer el archivo .json

    libros = pd.read_json(url)



    # Limpieza del dataframe en la columna link que al final del mismo
tenia la cadena "\n"

    libros['link'] = libros['link'].apply(lambda x : x.replace("\n", ''))

    # Conversion de tipos de datos

    libros['year'] = libros['year'].astype(str)

    libros['pages'] = libros['pages'].astype(str)

    return libros




def busqueda_por_patron(libros, patron_busqueda, columna):



    ''' Funcion auxiliar utilizada para buscar la/s fila/s del df que
contenga/n un patron en determinada

    columna. Es utilizada por el metodo GET /buscar libro  '''



    # Devuelve mascara booleana con valor True en las filas que contenga
la cadena buscada

    mascara = libros[columna].str.contains(patron_busqueda, case=False) #
Ignora may. y min.



    # Si al menos hay un True devuelve True, es decir hay resultados, sino
False, no hay resultados
```

```
    resultado_mascara = mascara.any()


    # En caso de haber resultados
    if resultado_mascara == True:


        # Buscamos los resultados
        resultado_busqueda = libros[mascara]


        # Convertimos a .json para poder devolverlo
        resultado_busqueda = resultado_busqueda.to_json(orient='records')


        return resultado_busqueda


    else:


        return 'No se encontro ningun resultado para el ' + columna + '
indicado'
```

APÉNDICE B

CLIENTE API

```
import tkinter as tk
from tkinter import scrolledtext, messagebox, simpledialog
import requests
```

```
import json



# Dirección IP y puerto del servidor FastAPI
server_ip = "127.0.0.1"       #IP DEL SERVIDOR, POR DEFAULT LOOPBACK

server_port = 8000

inicio = f"http://{server_ip}:{server_port}/libreria_entera"
requests.get(inicio)

# Variable para almacenar el endpoint
endpoint = ""

# Función para manejar los botones
def button_handler(endpoint_arg):
    global endpoint
    endpoint = endpoint_arg
    execute_action()

# Función para obtener la biblioteca completa
def get_libreria_entera():
    try:
        # Construir la URL completa con el endpoint seleccionado
        complete_url = f"http://{server_ip}:{server_port}{endpoint}"
```

```
        response = requests.get(complete_url)
        if response.status_code == 200:
            json_data = json.loads(response.json())

            # Limpiar el contenido actual del widget Text
            result_text.delete(1.0, tk.END)

            # Convertir los datos JSON a una cadena legible
            formatted_output = ""
            for item in json_data:
                for key, value in item.items():
                    formatted_output += f"{key}: {value}\n"
                formatted_output += "\n"

            # Insertar la cadena formateada en el widget Text
            result_text.insert(tk.END, formatted_output)
        else:
            result_text.delete(1.0, tk.END)
            result_text.insert(tk.END, "Error en la solicitud: " +
str(response.status_code))
    except requests.exceptions.RequestException:
        result_text.delete(1.0, tk.END)
        result_text.insert(tk.END, "Error en la solicitud")
```

```
def buscar_libro():
    parameter = simpledialog.askstring("Buscar libro", "Ingrese el parámetro a buscar (author, country, language, title o year)")
    value = simpledialog.askstring("Buscar libro", "Ingrese una parte del valor del parámetro")

    if parameter and value:
        try:
            # Construir la URL completa con el endpoint seleccionado y los parámetros
            complete_url = f"http://{server_ip}:{server_port}{endpoint}?columna={parameter}&patron_busqueda={value}"

            response = requests.get(complete_url)
            if response.status_code == 200:
                json_data = json.loads(response.json())

                # Limpiar el contenido actual del widget Text
                result_text.delete(1.0, tk.END)

                # Convertir los datos JSON a una cadena legible
                formatted_output = ""
                for item in json_data:
                    for key, value in item.items():
                        formatted_output += f"{key}: {value}\n"
                    formatted_output += "\n"
```

```
                # Insertar la cadena formateada en el widget Text
                result_text.insert(tk.END, formatted_output)
            else:
                result_text.delete(1.0, tk.END)
                result_text.insert(tk.END, "Error en la solicitud: " +
str(response.status_code))
        except requests.exceptions.RequestException:
            result_text.delete(1.0, tk.END)
            result_text.insert(tk.END, "Error en la solicitud")

# Función para agregar un libro
def agregar_libro():
    autor = simpledialog.askstring("Agregar libro", "Ingrese el autor:")
    pais = simpledialog.askstring("Agregar libro", "Ingrese el país:")
    link_imagen = simpledialog.askstring("Agregar libro", "Ingrese el
enlace de la imagen:")
    idioma = simpledialog.askstring("Agregar libro", "Ingrese el idioma:")
    link = simpledialog.askstring("Agregar libro", "Ingrese el enlace:")
    paginas = simpledialog.askstring("Agregar libro", "Ingrese el número
de páginas:")
    titulo = simpledialog.askstring("Agregar libro", "Ingrese el título:")
    anio = simpledialog.askstring("Agregar libro", "Ingrese el año:")

    try:
        complete_url =
```

```
f"http://{server_ip}:{server_port}{endpoint}?autor={autor}&pais={pais}&link_imagen={link_imagen}&idioma={idioma}&link={link}&paginas={paginas}&titulo={titulo}&a%C3%B1o={anio}"



        response = requests.post(complete_url)

        if response.status_code == 200:

            result_text.delete(1.0, tk.END)

            result_text.insert(tk.END, "Libro agregado exitosamente")

        else:

            result_text.delete(1.0, tk.END)

            result_text.insert(tk.END, "Error en la solicitud: " +
str(response.status_code))

    except requests.exceptions.RequestException:

        result_text.delete(1.0, tk.END)

        result_text.insert(tk.END, "Error en la solicitud")



# Función para modificar un libro

def modificar_libro():

    titulo_cambiar = simpledialog.askstring("Modificar libro", "Ingrese el
título del libro a modificar:")

    autor = simpledialog.askstring("Modificar libro", "Ingrese el autor:")

    pais = simpledialog.askstring("Modificar libro", "Ingrese el país:")

    link_imagen = simpledialog.askstring("Modificar libro", "Ingrese el
enlace de la imagen:")

    idioma = simpledialog.askstring("Modificar libro", "Ingrese el
idioma:")

    link = simpledialog.askstring("Modificar libro", "Ingrese el enlace:")
```

```
    paginas = simpledialog.askstring("Modificar libro", "Ingrese el número de páginas:")
    titulo = simpledialog.askstring("Modificar libro", "Ingrese el título:")
    anio = simpledialog.askstring("Modificar libro", "Ingrese el año:")

    try:
        complete_url = f"http://{server_ip}:{server_port}{endpoint}?eleccion={titulo_cambiar}&autor={autor}&pais={pais}&link_imagen={link_imagen}&idioma={idioma}&link={link}&paginas={paginas}&titulo={titulo}&a%C3%B1o={anio}"

        response = requests.put(complete_url)
        if response.status_code == 200:
            result_text.delete(1.0, tk.END)
            result_text.insert(tk.END, "Libro modificado exitosamente")
        else:
            result_text.delete(1.0, tk.END)
            result_text.insert(tk.END, "Error en la solicitud: " + str(response.status_code))
    except requests.exceptions.RequestException:
        result_text.delete(1.0, tk.END)
        result_text.insert(tk.END, "Error en la solicitud")

# Función para eliminar un libro
def eliminar_libro():
    titulo_eliminar = simpledialog.askstring("Eliminar libro", "Ingrese el título del libro a eliminar:")
```

```
    try:
        complete_url =
f"http://{server_ip}:{server_port}{endpoint}?eleccion={titulo_eliminar}"

        response = requests.delete(complete_url)
        if response.status_code == 200:
            result_text.delete(1.0, tk.END)
            result_text.insert(tk.END, "Libro eliminado exitosamente")
        else:
            result_text.delete(1.0, tk.END)
            result_text.insert(tk.END, "Error en la solicitud: " +
str(response.status_code))
    except requests.exceptions.RequestException:
        result_text.delete(1.0, tk.END)
        result_text.insert(tk.END, "Error en la solicitud")

# Función para ejecutar la acción correspondiente al botón presionado
def execute_action():
    if endpoint == "/libreria_entera":
        get_libreria_entera()
    elif endpoint == "/buscar_libro":
        buscar_libro()
    elif endpoint == "/agregar_libro":
        agregar_libro()
    elif endpoint == "/modificar_libro":
        modificar_libro()
```

```
    elif endpoint == "/eliminar_libro":
        eliminar_libro()


# Crear la ventana
window = tk.Tk()
window.title("BIBLIOTECA")


# Crear los botones
buttons = [
    {"text": "LIBRERIA ENTERA", "endpoint": "/libreria_entera"},
    {"text": "BUSCAR LIBRO", "endpoint": "/buscar_libro"},
    {"text": "AGREGAR LIBRO", "endpoint": "/agregar_libro"},
    {"text": "MODIFICAR LIBRO", "endpoint": "/modificar_libro"},
    {"text": "ELIMINAR LIBRO", "endpoint": "/eliminar_libro"},
]


for button_info in buttons:
    button = tk.Button(window, text=button_info["text"], command=lambda
endpoint=button_info["endpoint"]: button_handler(endpoint))
    button.pack(pady=10)


# Crear el widget Text con deslizador
result_text = scrolledtext.ScrolledText(window, width=60, height=20)
result_text.pack()
```

```
# Ejecutar el bucle principal de la ventana

window.mainloop()
```