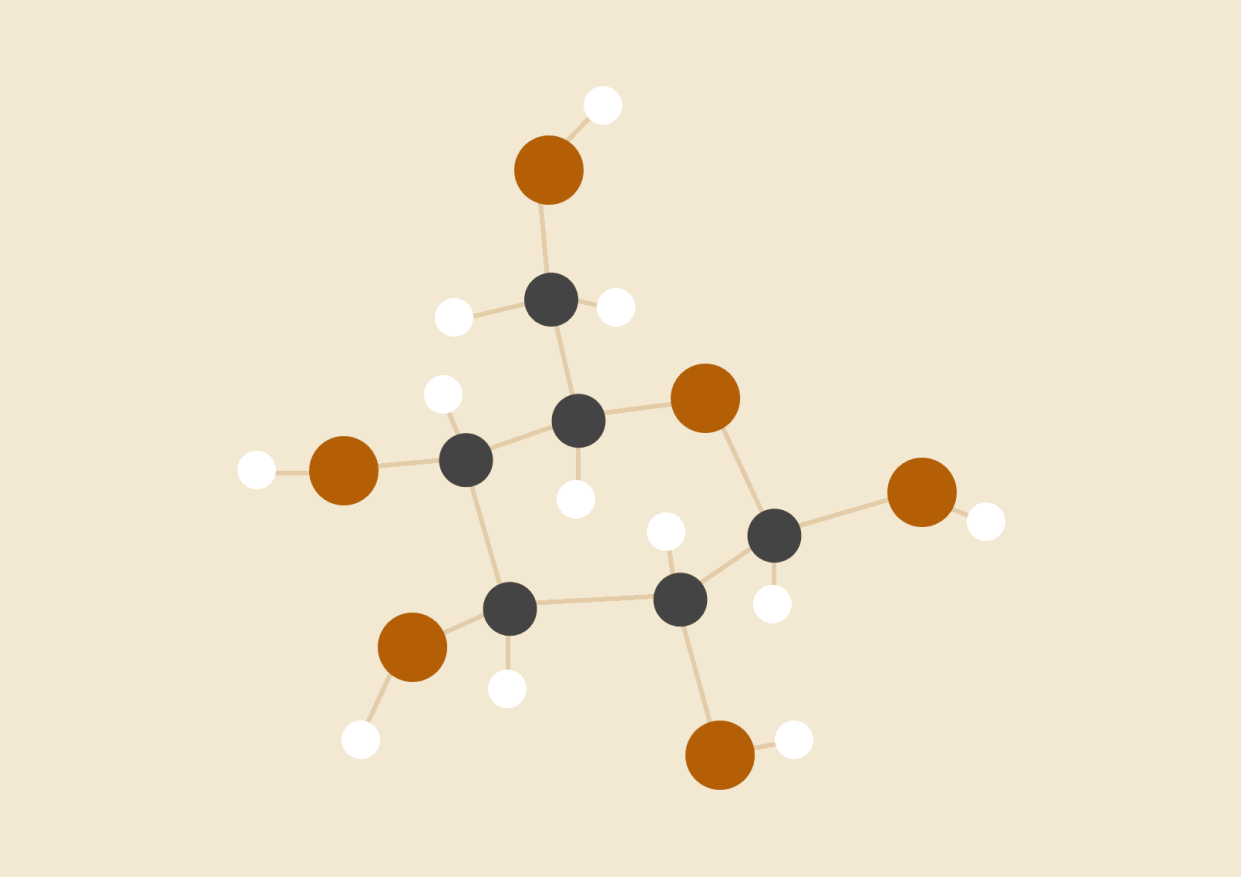
TRABAJO PRÁCTICO INTEGRADOR

IA3.5 Redes de Datos

Tecnicatura Universitaria en Inteligencia Artificial



Facundo López Crespo

22/02/2024

Universidad Nacional de Rosario

Facultad de Ciencias Exactas, Ingeniería y Agrimensura

INTRODUCCIÓN

El trabajo consiste en implementar una comunicación REST API Cliente-Servidor entre dos hosts para gestionar una base de datos de libros. Para el host del servidor, se utiliza un entorno físico con Microsoft Windows como sistema operativo. Para el host del cliente, se utiliza un entorno virtual la distribución Ubuntu de Linux como sistema operativo.

DESCRIPCIÓN DEL ENTORNO DE TRABAJO DEL SERVIDOR

Este entorno de trabajo está implementado a través del uso de Python. A su vez, se utilizan las siguientes librerías: fastapi, uvicorn, json, pydantic y socket.

Entre algunos de los problemas que debieron enfrentarse durante su implementación, puede identificarse:

(i) la definición del código necesario para captar la dirección IP del host donde correr el servidor;

(ii) la definición del código necesario para captar el body que el cliente incluye en la petición vinculada con la creación de un libro; y

(iii) la definición del código necesario para devolver un error con status code 404 cuando el libro que el cliente quiere consultar no existe en la base de datos.

DESCRIPCIÓN DEL ENTORNO DE TRABAJO DEL DEL CLIENTE

Este entorno de trabajo está implementado a través del uso de Python. A su vez, se utilizan las siguientes librerías: requests, time.

Entre algunos de los problemas que debieron enfrentarse durante su implementación, puede identificarse:

(i) la definición del código necesario para implementar el menú de opciones para vincularse con el servidor;

(ii) la definición del código necesario para que, ante la consulta del cliente por el listado de libros, la respuesta del servidor pueda mostrarse de manera amigable para un usuario final;

(iii) la definición del código necesario para que las peticiones POST y PUT incluyan el body necesario para que el servidor pueda procesarlas; y

(iv) la definición del código necesario para utilizar rutas parametrizadas.

DESCRIPCIÓN DE LA BASE DE DATOS ELEGIDA

Se eligió la base de datos provista por la Cátedra que contiene la lista de los mejores 100 libros. Al respecto, se realizan las siguientes dos aclaraciones:

(i) el objeto libro no contiene un atributo id, así está definido en la base de datos descargada, por lo cual, la búsqueda de un libro en particular se realiza consultando el atributo "title" del objeto.

(ii) a los efectos de hacer más amigable la visualización del listado de libros para el usuario final, se redujo el tamaño de la base de datos original.

Por el recorte mencionado en esta última aclaración, el tamaño de la base de datos es de 20 libros. A continuación, se listan las propiedades de este objeto con sus respectivos tipos de datos:

- author: str
- country: str
- imageLink: str
- language: str
- link: str
- pages: int
- title: str
- year: int

PROCEDIMIENTO

Descripciones y ejemplos de las instrucciones para comunicarse con la API.

El directorio *proyecto_final* cuenta con los archivos *main_servidor.py*, programa que debe ejecutarse en el host del servidor, y *main_cliente.py,* programa que debe ejecutarse en el host del cliente. A su vez, también cuenta con la base de datos de 20 libros en el archivo *books.json*.

Servidor:

Antes de iniciar el servidor, corresponde instalar las librerías fastapi y uvicorn, por lo cual, deben ejecutarse por separado los siguientes scripts en la terminal:

pip install fastapi
pip install uvicorn

Luego de instaladas estas librerías puede iniciarse prenderse el servidor. Para ello, posicionándose en el directorio referido, corresponde ejecutar el script:

python3 main_servidor.py

Este programa consulta la dirección IP del host, la cual es provista por terminal para ser utilizada en el programa del cliente.

Cliente:

Antes de iniciar el cliente, corresponde instalar la librería requests, por lo cual, debe ejecutarse el siguiente script:

pip install requests

Luego de instalada esta librería pueda iniciarse el programa cliente. Para ello, posicionándose en el directorio referido, corresponde correr el script:

python3 main_cliente.py

El primer dato que el programa va a solicitar para poder continuar con su ejecución es la dirección IP provista por el programa servidor.

INSTRUCCIONES DE USO DEL CLIENTE

Luego de ingresada la dirección IP del servidor el programa ingresará a un menú de opciones para ejecutar alguna de las siguientes instrucciones que equivalen a los endpoints implementados del lado del servidor:

- Consultar todos los libros contenidos en la base de datos.
- Crear un nuevo libro. El programa solicitará adicionalmente completar los campos necesarios para concretar la instrucción. Deben rellenarse todos los campos.
- Consultar un libro en particular. El programa solicitará adicionalmente ingresar el título del libro buscado.
- Consultar todos los libros referidos a un autor en particular. El programa solicitará adicionalmente ingresar el nombre del autor buscado.
- Actualizar los datos de un libro en particular. El programa solicitará adicionalmente completar los campos necesarios para concretar la instrucción. Deben rellenarse todos los campos. En caso que no quiera cambiarse el dato para un campo en particular, debe ingresarse el dato ya existente.
- Eliminar un libro en particular. El programa solicitará ingresar el título del libro a eliminar.

A los efectos de facilitar el uso del programa cliente, el directorio referido cuenta con el archivo *notas* que contiene información de ejemplo para completar los campos solicitados al usuario final en la formulación de las instrucciones.

REFERENCIAS

Para la realización del presente trabajo se consultaron las respectivas documentaciones de las librerías:

1. FastApi
2. Uvicorn
3. Requests

APÉNDICE A

Código del servidor

```
from fastapi import FastAPI,Query,HTTPException
from pydantic import BaseModel
import uvicorn, socket,json

hostname = socket.gethostname()
IPAddr = socket.gethostbyname(hostname)

app = FastAPI()

class Book(BaseModel):
    author: str
    country: str
    imageLink: str
    language: str
    link: str
    pages: int
    title: str
    year: int

with open ('books.json','r') as file:
    books = json.load(file)

@app.get("/api/v1/books",status_code=200)
def get_books()->list:
    if (len(books)>0):
        return books
    else:
        raise HTTPException(
            status_code=404,
            detail=f'There is no books'
        )
```

```
@app.get("/api/v1/books/search",status_code=200)
def search_books_author(author:str=Query(title='author',description='Looking up
books by author'))->list:
    books_by_author = []
    for book in books:
        if (book["author"] == author):
            books_by_author.append(book)
    if (len(books_by_author) > 0):
        return books_by_author
    else:
        raise HTTPException(status_code=404,detail=f'There is no book with
{author} as author')

@app.get("/api/v1/books/{title}")
def get_book_title(title:str)->dict:
    for book in books:
        if (book["title"] == title):
            return book
    raise HTTPException(status_code=404,detail=f'There is no book with {title} as
title')

@app.post("/api/v1/books",status_code=201)
def create_book(book:Book)->dict:
    new_book={
        "author": book.author,
        "country": book.country,
        "imageLink": book.imageLink,
        "language": book.language,
        "link": book.link,
        "pages": book.pages,
        "title": book.title,
        "year": book.year
    }
    books.append(new_book)
    with open("books.json",'w') as file:
        json.dump(books,file)
    return new_book

@app.put('/api/v1/books/{title}',status_code=200)
def update_book(title:str,book_updated:Book):
    for book in books:
        if (book['title'] == title):
            book["author"] = book_updated.author
            book["country"] = book_updated.country
            book["imageLink"] = book_updated.imageLink
```

```
            book["language"] = book_updated.language
            book["link"] = book_updated.link
            book["pages"] = book_updated.pages
            book["title"] = book_updated.title
            book["year"] = book_updated.year
            with open("books.json",'w') as file:
                json.dump(books,file)
            return book_updated
    raise HTTPException(status_code=404,detail=f'There is no book with {title} as
title')

@app.delete('/api/v1/books/{title}',status_code=200)
def delete_book(title:str)->dict:
    for book in books:
        if (book['title'] == title):
            books.remove(book)
            with open("books.json",'w') as file:
                json.dump(books,file)
            return {
                "message":"Book deleted"
            }
    raise HTTPException(status_code=404,detail=f'There is no book with {title} as
title')

if __name__ == "__main__":
    print(f"La dirección IP de este servidor es {IPAddr}. Esta dirección va a ser
solicitada por el programa del cliente")
    uvicorn.run("main_servidor:app", port=80, log_level="info",host=IPAddr)
```

APÉNDICE B

Código del cliente

```
print('¡Bienvenidos!')
servidor_direccion_ip = input("Ingrese la dirección IP del servidor: ")

def libro()->dict:
    author = input("Autor: ")
    country = input("País: ")
    imageLink = input("Foto: ")
    language = input("Idioma: ")
    link = input("Enlace: ")
    pages = int(input("Cantidad de páginas: "))
    title = input("Título: ")
    year = int(input("Año: "))
```

```
    libro = {
      "author": author,
      "country": country,
      "imageLink": imageLink,
      "language": language,
      "link": link,
      "pages": pages,
      "title": title,
      "year": year
    }
    return libro

def imprimir_libro(libro:dict)->None:
    autor = libro['author']
    pais = libro['country']
    imagen = libro['imageLink']
    idioma = libro['language']
    enlace = libro['link']
    paginas = libro['pages']
    titulo = libro['title']
    año = libro['year']
    print(f"* {titulo}")
    print(f"\t autor: {autor}")
    print(f"\t pais: {pais}")
    print(f"\t imagen: {imagen}")
    print(f"\t idioma: {idioma}")
    print(f"\t enlace: {enlace}")
    print(f"\t paginas: {paginas}")
    print(f"\t año: {año}")

def menu()->int:
    print("---------------------------------------")
    print("\nIngrese una de las siguientes opciones")
    print("1- Buscar todos los libros")
    print("2- Agregar un libro")
    print("3- Buscar un libro por título")
    print("4- Buscar libros por autor")
    print("5- Editar un libro")
    print("6- Eliminar un libro")
    print("0- Salir")
    op = input("Ingrese la opción deseada: ")
    while(not op.isdigit() or int(op)<0 or int(op)>6):
        op = (input("Error. Vuelva a ingresar una opción (0-6): "))
    return int(op)
```

```
def repetir_menu():
  repetir_menu = input("\n¿Quiere repetir el menú? (sí-no): ")
  if (repetir_menu=="no"):
    op=0
    return op

def buscar_libros()->None:
    respuesta = requests.get(f"http://{servidor_direccion_ip}:80/api/v1/books")
    if (respuesta.status_code==200):
      libros = respuesta.json()
      print(f'\nContamos con los siguiente {len(libros)} libros:\n')
      for libro in libros:
        imprimir_libro(libro)
    elif (respuesta.status_code==404):
        print("No tenemos libros para mostrar")


def crear_libro(nuevo_libro:dict)->None:
  respuesta =
requests.post(f"http://{servidor_direccion_ip}:80/api/v1/books/",json=nuevo_libro
)
  if (respuesta.status_code==201):
    libro_creado = respuesta.json()
    print("Agregamos el siguiente libro: \n")
    imprimir_libro(libro_creado)

def buscar_libro_titulo(titulo:str)->None:
    respuesta =
requests.get(f"http://{servidor_direccion_ip}:80/api/v1/books/{titulo}")
    if (respuesta.status_code==200):
        libro = respuesta.json()
        print(f'\nEste es el libro que encontramos con ese título:\n')
        imprimir_libro(libro)
    elif (respuesta.status_code==404):
        print("No tenemos ningún libro con ese título")

def buscar_libros_autor(autor:str)->None:
  autor_buscado = {"author":autor}
  #El argumento params almacena la query de la solicitud
  respuesta =
requests.get(f"http://{servidor_direccion_ip}:80/api/v1/books/search",params=auto
r_buscado)
  if (respuesta.status_code==200):
    libros = respuesta.json()
    print("Contamos con los siguientes libros de ese autor: \n")
```

```
    for libro in libros:
      imprimir_libro(libro)
  elif (respuesta.status_code==404):
      print("No tenemos libros de ese autor")


def actualizar_libro(titulo:str,libro_editado:str)->None:
  respuesta =
requests.put(f"http://{servidor_direccion_ip}:80/api/v1/books/{titulo}",json=libr
o_editado)
  if (respuesta.status_code==200):
      libro = respuesta.json()
      print("Libro actualizado: \n")
      imprimir_libro(libro)
  elif (respuesta.status_code==404):
      print("No tenemos ningún libro con ese título")

def eliminar_libro(titulo:str)->None:
  respuesta =
requests.delete(f"http://{servidor_direccion_ip}:80/api/v1/books/{titulo}")
  if (respuesta.status_code==200):
      print("Libro eliminado")
  elif (respuesta.status_code==404):
      print("No tenemos ningún libro con ese título")

def principal():
  opcion=1
  while opcion!=0:
    opcion = menu()
    if (opcion==1):
      buscar_libros()
      time.sleep(3)
      opcion = repetir_menu()
    if (opcion==2):
      nuevo_libro = libro()
      crear_libro(nuevo_libro)
      time.sleep(3)
      opcion = repetir_menu()
    if (opcion==3):
      titulo = input("Ingrese el título del libro que desea buscar: ")
      buscar_libro_titulo(titulo)
      time.sleep(3)
      opcion = repetir_menu()
    if (opcion==4):
      autor = input("Ingrese el autor que desea buscar: ")
```

```
      buscar_libros_autor(autor)
      time.sleep(3)
      opcion = repetir_menu()
    if (opcion==5):
      libro_actualizado = libro()
      titulo_libro_actualizado = libro_actualizado['title']
      actualizar_libro(titulo_libro_actualizado,libro_actualizado)
      time.sleep(3)
      opcion = repetir_menu()
    if (opcion==6):
      titulo_libro_eliminado = input("Ingrese el título del libro que desea
eliminar: ")
      eliminar_libro(titulo_libro_eliminado)
      time.sleep(3)
      opcion = repetir_menu()
  print("Fin del programa!")

principal()
```