README.md

**# Use of FastAPI as REST API**

-   Link: https://www.youtube.com/watch?v=4e2VW3Nu-64

create a virtual environment

-   creacion del entorno virtual `virtualenv -p python3 python-mongo`

-   Activar el entorno virtual `.\python-mongo\Scripts\activate`

-   para exportar los paquetes del env `pip freeze > requirements.txt`

-   para instalar de nuevo los paquetes `pip install -r .\requirements.txt`

-   para desactivar el entorno `deactivate`

**## Correr el servidor**

-   `uvicorn app:app` dentro del archivo app correr el objeto app

-   `uvicorn app:app --reload` dentro del archivo app correr el objeto app para recargar

**## Python packages**

-   pip install fastapi

-   pip install uvicorn

-   pip install pymongo

-   pip install python-dotenv

-   pip install passlib

app.py

from fastapi import FastAPI

from fastapi.middleware.cors import CORSMiddleware

from routes.user import user  # esta es la ruta creada en el archivo user de rutas

from docs import tags\_metadata

app = FastAPI(

    title="REST API with FastAPI and MongoDB",

    description="This is a simple REST API using FastAPI and MongoDB",

    version="0.0.1",

    openapi\_tags=tags\_metadata

)

app.add\_middleware(

    CORSMiddleware,

    allow\_origins=["\*"],

    allow\_credentials=True,

    allow\_methods=["\*"],

    allow\_headers=["\*"],

)

# carga de la ruta

app.include\_router(user)

docs.py

tags\_metadata = [{

    "name": "users",

    "description": "user\_routes"

}]

schemas/user.py

def userEntity(item) -> dict:

    # creacion de los esquemas para almacenar en mongo

    return {

        "\_id": str(item["\_id"]),

        "name": item["name"],

        "email": item["email"],

        "password": item["password"]

    }

def usersEntity(entity) -> list:

    # esquema de retorno de todos los usuarios

    return [userEntity(item) for item in entity]

models/user.py

from typing import Optional

from pydantic import BaseModel

# creacion del modelo de conexion con la base de datos

class User(BaseModel):  # User hereda de BaseModel

    \_id: Optional[str]  # el id es opcional

    name: str

    email: str

    password: str

routes/user.py

from fastapi import APIRouter, Response, status  # Definir todas las rutas

from config.db import conn  # importar el objeto de conexion

from schemas.user import userEntity, usersEntity

from models.user import User  # tipo de la entidad

from passlib.hash import sha256\_crypt

from bson import ObjectId

from starlette.status import HTTP\_204\_NO\_CONTENT

user = APIRouter()

@user.get('/users', response\_model=list[User], tags=["Users"])

def find\_all\_user():

    # cuando se acceda a la ruta users se retornara todos los usuarios

    # de la connexion a mondo busque en la coleccion user todos

    return usersEntity(conn.remote.user.find())

@user.post('/users', response\_model=User, tags=["Users"])

def find\_all\_user(user: User):

    # crear nuevo usuario

    new\_user = dict(user)

    new\_user["password"] = sha256\_crypt.encrypt(new\_user['password'])

    del new\_user['\_id']

    id = conn.remote.user.insert\_one(new\_user).inserted\_id

    # consulta en la base de datos el ultimo dato creado

    user = conn.remote.user.find\_one({"\_id": id})

    return userEntity(user)

@user.get('/users/{id}', response\_model=User, tags=["Users"])

def find\_user(id: str):

    # buscar un unico usuario

    return userEntity(conn.remote.user.find\_one({"\_id": ObjectId(id)}))

@user.put('/users/{id}', response\_model=User, tags=["Users"])

def update\_user(id: str, user: User):

    # actualizar usuario

    conn.remote.user.find\_one\_and\_update(

        {"\_id": ObjectId(id)}, {"$set": dict(user)})

    return userEntity(conn.remote.user.find\_one({"\_id": ObjectId(id)}))

@user.delete('/users/{id}', status\_code=status.HTTP\_204\_NO\_CONTENT, tags=["Users"])

def delete\_user(id: str):

    # eliminar usuario

    userEntity(conn.remote.user.find\_one\_and\_delete({"\_id": ObjectId(id)}))

    return Response(status\_code=HTTP\_204\_NO\_CONTENT)

config/db.py

from pymongo import MongoClient  # conexion a una base de datos

from pymongo.server\_api import ServerApi

from dotenv import load\_dotenv

import os

load\_dotenv()

uri = os.getenv("MONGO\_URI")

conn = MongoClient(uri)  # , server\_api=ServerApi('1'))

# Send a ping to confirm a successful connection

try:

    conn.admin.command('ping')

    print("Pinged your deployment. You successfully connected to MongoDB!")

except Exception as e:

    print(e)

.env

#.env file

MONGO\_URI = "mongodb+srv://fabian:fabian@instrumentation.uqtyxfa.mongodb.net/?retryWrites=true&w=majority"

Frontend

components/App.js

import React, { useState } from "react";

import { axiosInstance } from "../config/config";

function App() {

    const [text, setText] = useState();

    function loadDataFromDB() {

        axiosInstance

            .get(`/users`)

            .then((res) => {

                console.log(res);

                setText(JSON.stringify(res.data));

            })

            .catch((err) => console.error(err));

    }

    function eventHandler() {

        loadDataFromDB();

    }

    return (

        <div className="App">

            <h1>Hello World</h1>

            <p>{text}</p>

            <button onClick={eventHandler}>Presioname</button>

        </div>

    );

}

export default App;

config/config.js

import axios from "axios";

export const axiosInstance = axios.create({

    baseURL: "http://localhost:8000/",

});