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

**## FrontEnd**

- npx create-react-app frontend

- npm install axios

- npm install bootstrap

- npm install moment

- npm install react-bootstrap

- npm install react-chartjs-2

- npm install react-icons

Dockerfile

FROM python:3.9

EXPOSE 8000

ENV PYTHONDONTWRITEBYTECODE=1

ENV PYTHONUNBUFFERED=1

WORKDIR /

COPY requirements.txt requirements.txt

RUN pip install --no-cache-dir --upgrade -r /requirements.txt

COPY . .

CMD ["uvicorn", "app:app", "--host", "0.0.0.0", "--port", "8000"]

Docker-compose.yml

version: "3.9"

services:

  fastapitest:

    image: fastapitest

    build:

      context: .

      dockerfile: ./Dockerfile

    ports:

      - 8000:8000

app.py

from fastapi import FastAPI

from fastapi.middleware.cors import CORSMiddleware

from fastapi.staticfiles import StaticFiles

# esta es la ruta creada en el archivo user de rutas

from routes.monitor import monitor

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

)

# carga de la ruta

app.include\_router(monitor)

# despliegue de frontend

app.mount("/", StaticFiles(directory="frontend/build",

          html=True), name="frontend")

app.add\_middleware(

    CORSMiddleware,

    allow\_origins=["\*"],

    allow\_credentials=True,

    allow\_methods=["\*"],

    allow\_headers=["\*"],

)

docs.py

tags\_metadata = [{

    "name": "monitors",

    "description": "monitor\_routes"

}]

schemas/monitor.py

def monitorEntity(item) -> dict:

    # creacion de los esquemas para almacenar en mongo

    return {

        "\_id": str(item["\_id"]),

        "place": item["place"],

        "author": item["author"],

        "temperature": item["temperature"],

        "humidity": item["humidity"],

        "createdAt": item["createdAt"],

        "updatedAt": item["updatedAt"]

    }

def monitorsEntity(entity) -> list:

    # esquema de retorno de todos los datos

    return [monitorEntity(item) for item in entity]

models/monitor.py

from typing import Optional

from pydantic import BaseModel

import datetime

# creacion del modelo de conexion con la base de datos

class Monitor(BaseModel):  # Monitor hereda de BaseModel

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

    place: str

    author: str

    temperature: float

    humidity: float

    createdAt: Optional[datetime.datetime]

    updatedAt: Optional[datetime.datetime]

routes/monitor.py

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

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

from schemas.monitor import monitorEntity, monitorsEntity

from models.monitor import Monitor  # tipo de la entidad

from bson import ObjectId

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

import datetime

monitor = APIRouter()

# definicion de las rutas  
  
@monitor.get('/monitor', tags=["Monitors"])  
def find\_all\_monitor():  
    # cuando se acceda a la ruta monitor se retornara todos los monitors  
    # de la connexion a mondo busque en la coleccion monitor todos  
    return monitorsEntity(conn.monitor.monitor.find())  
  
@monitor.get('/monitor/last', tags=["Monitors"])  
def find\_last\_monitor():  
    # busqueda del ultimo dato de monitores  
    return monitorEntity(conn.monitor.monitor.find\_one({}, sort=[('$natural', -1)]))  
  
@monitor.get('/monitor/place/{place}', tags=["Monitors"])  
def find\_place\_monitor(place: str):  
    # busqueda del ultimo dato de monitores  
    return monitorsEntity(conn.monitor.monitor.find({'place': place}, sort=[('$natural', -1)]))  
  
@monitor.post('/monitor', response\_model=Monitor, tags=["Monitors"])  
def save\_monitor(monitor: Monitor):  
    # Crear nuevo dato de monitoreo  
    new\_monitor = dict(monitor)  
    new\_monitor["createdAt"] = datetime.datetime.utcnow()  
    new\_monitor["updatedAt"] = datetime.datetime.utcnow()  
    id = conn.monitor.monitor.insert\_one(new\_monitor).inserted\_id  
    # consulta en la base de datos el ultimo dato creado  
    monitor\_load = conn.monitor.monitor.find\_one({"\_id": id})  
    return monitorEntity(monitor\_load)  
  
@monitor.put('/monitor/{id}', response\_model=Monitor, tags=["Monitors"])  
def update\_monitor(id: str, monitor: Monitor):  
    # actualizar monitor  
    monitor = dict(monitor)  
    monitor["updatedAt"] = datetime.datetime.utcnow()  
    conn.monitor.monitor.find\_one\_and\_update(  
        {"\_id": ObjectId(id)}, {"$set": monitor}  
    )  
    return monitorEntity(conn.monitor.monitor.find\_one({"\_id": ObjectId(id)}))  
  
@monitor.delete('/monitor/{id}', status\_code=status.HTTP\_204\_NO\_CONTENT, tags=["Monitors"])  
def delete\_monitor(id: str):  
    # eliminar usuario  
    monitorEntity(conn.monitor.monitor.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, { useEffect, useState } from "react";

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

import { Container } from "react-bootstrap";

import DataTable from "./DataTable";

import "../styles/background.css";

function App() {

  // Hooks de datos

  //const [data, setData] = useState();

  const [monitorObj, setMonitorObj] = useState([]);

  // Carga de datos al inicio de la aplicacion

  useEffect(() => {

    loadDataFromDB();

  }, []);

  // carga de datos desde DB

  function loadDataFromDB() {

    axiosInstance

      .get(`/monitor`)

      .then((res) => {

        console.log(res.data);

        //setData(JSON.stringify(res.data));

        // Llenado del array con los datos

        setMonitorObj(res.data);

      })

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

  }

  return (

    <div className="App app-background">

      <Container className="p-5">

        <h3 className="header">Tabla de datos</h3>

        <DataTable data={monitorObj} />

      </Container>

    </div>

  );

}

export default App;

config/config.js

import axios from "axios";

export const axiosInstance = axios.create({

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

});

index.js

import React from "react";

import ReactDOM from "react-dom/client";

import App from "./components/App";

import "bootstrap/dist/css/bootstrap.min.css";

const root = ReactDOM.createRoot(document.getElementById("root"));

root.render(

  <React.StrictMode>

    <App />

  </React.StrictMode>

);

Styles/background.css

.app-background {

    background-image: linear-gradient(

        135deg,

        #4ab50c 5%,

        #454e69 10%,

        #4ab50c 90%

    );

    color: #ffffff;

}

.app-card {

    background-image: linear-gradient(to bottom, #1d2333, #0d6c15);

}

components/DataTable.js

import React from "react";

import { Table } from "react-bootstrap";

export default function DataTable(props) {

  const { data } = props;

  return (

    <div className="container">

      <div className="container p-5">

        <h2>Tabla de datos</h2>

      </div>

      <div className="container grey lighten-4">

        <Table responsive striped bordered hover variant="dark">

          <thead>

            <tr>

              <th>Fecha</th>

              <th>Lugar</th>

              <th>Autor</th>

              <th>Temperatura</th>

              <th>Humedad</th>

            </tr>

          </thead>

          <tbody>

            {data.map((monitorObj) => {

              return (

                <tr key={monitorObj.\_id}>

                  <td>{monitorObj.createdAt}</td>

                  <td>{monitorObj.place}</td>

                  <td>{monitorObj.author}</td>

                  <td>{monitorObj.temperature}</td>

                  <td>{monitorObj.humidity}</td>

                </tr>

              );

            })}

          </tbody>

        </Table>

      </div>

    </div>

  );

}