fastapi + mongodb

▼ 간단한 user data 불러오는 것

0. 프로젝트 구조

project_directory/
│
├── myenv/ (가상환경 디렉토리)
│ ├── bin/
│ ├── include/
│ └── lib/
│
├──
app.py (FastAPI 애플리케이션 코드)
├── requirements.txt (프로젝트 의존성 목록)
└── ... (기타 프로젝트 파일들)

1. wsl에서 가상환경 구축

> sudo python3 -m venv myenv

2. 가상환경 활성화

> source myenv/bin/activate

3. 필요한 패키지 설치

> pip install fastapi uvicorn motor

4. 몽고db 설치 및 실행

> sudo apt update
> sudo apt upgrade
>
>
> curl -fsSL https://www.mongodb.org/static/pgp/server-6.0.asc | sudo apt-key add -

> echo "deb [arch=amd64,arm64] https://repo.mongodb.org/apt/ubuntu jammy/mongodb-org/6.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-6.0.list
>
> sudo apt update
>
> sudo apt install mongodb-org
>
> sudo systemctl start mongod
>
> sudo systemctl enable mongod
>
> sudo systemctl status mongod
>
> mongod

5. 간단한 app.py

```
from fastapi import FastAPI
from motor.motor_asyncio import AsyncIOMotorClient
from pydantic import BaseModel

app = FastAPI()

# MongoDB 연결
client = AsyncIOMotorClient("mongodb://localhost:27017")
db = client.userdb
```

```
collection = db.users

# 사용자 모델 정의
class User(BaseModel):
    nickname: str
    email: str

@app.on_event("startup")
async def startup_db_client():
    # 서버 시작시 샘플 사용자 추가
    await collection.insert_one({"nickname": "John Doe", "

@app.get("/user")
async def get_user():
    user = await collection.find_one({})
    if user:
        return {"nickname": user["nickname"], "email": use
    return {"error": "User not found"}

if __name__ == "__main__":
    import uvicorn
    uvicorn.run(app, host="0.0.0.0", port=8000)
```

6. fastapi를 실행

> python3 app.py

7. localhost:8000/user 으로 접속

▼ 간단한 회원가입, 로그인 페이지 및 기능 구현

redis : 세션관리

mongodb: 사용자 정보 저장

프로젝트 구조

project_directory/
│

```
├── myenv/          (가상환경 디렉토리)
│   ├── bin/
│   ├── include/
│   └── lib/
│
├──
app.py          (FastAPI 애플리케이션 코드)
├── static/index.html
```

▼ 1. redis 설치 및 실행

1. redis 설치 및 실행

```
# Redis 설치
sudo apt update
sudo apt install redis-server

# Redis 서비스 시작
sudo systemctl start redis-server

# Redis 서비스가 실행 중인지 확인
sudo systemctl status redis-server

# Redis가 부팅 시 자동으로 시작되도록 설정
sudo systemctl enable redis-server
```

2. (선택) redis 설정 확인 - redis를 로컬 호스트 말고 다른 곳에서도 접근하게 변경하고 싶을 때

```
sudo nano /etc/redis/redis.conf
```

3. redis-py 라이브러리 설치

> pip install redis

2. 필요 라이브러리 설치

> pip install fastapi uvicorn motor redis "pydantic[email]" "passlib[bcrypt]" "python-jose[cryptography]”

3. app.py 작성

```
import secrets
from fastapi import FastAPI, HTTPException, Depends
from fastapi.security import OAuth2PasswordBearer
from pydantic import BaseModel, EmailStr
from motor.motor_asyncio import AsyncIOMotorClient
from passlib.context import CryptContext
from jose import JWTError, jwt
from datetime import datetime, timedelta
import redis
import uuid
from fastapi.middleware.cors import CORSMiddleware
from fastapi import FastAPI
from fastapi.staticfiles import StaticFiles
from fastapi.responses import FileResponse

app = FastAPI()
# 정적 파일을 위한 디렉토리 설정
app.mount("/static", StaticFiles(directory="static"), name

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],  # 실제 운영 환경에서는 특정 출처만 허용
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

# MongoDB 연결
MONGO_URL = "mongodb://localhost:27017"
client = AsyncIOMotorClient(MONGO_URL)
db = client.userdb
users_collection = db.users
```

```
# Redis 연결
redis_client = redis.Redis(host='localhost', port=6379, db

# 비밀번호 해싱
pwd_context = CryptContext(schemes=["bcrypt"], deprecated=

# JWT 설정
SECRET_KEY = secrets.token_urlsafe(32)
ALGORITHM = "HS256"
ACCESS_TOKEN_EXPIRE_MINUTES = 30

oauth2_scheme = OAuth2PasswordBearer(tokenUrl="token")

class UserIn(BaseModel):
    email: EmailStr
    password: str

class UserOut(BaseModel):
    email: EmailStr

class Token(BaseModel):
    access_token: str
    token_type: str

def create_access_token(data: dict):
    to_encode = data.copy()
    expire = datetime.utcnow() + timedelta(minutes=ACCESS_
    to_encode.update({"exp": expire})
    encoded_jwt = jwt.encode(to_encode, SECRET_KEY, algori
    return encoded_jwt

async def get_current_user(token: str = Depends(oauth2_sch
    credentials_exception = HTTPException(
        status_code=401,
        detail="Could not validate credentials",
        headers={"WWW-Authenticate": "Bearer"},
    )
    try:
```

```
        payload = jwt.decode(token, SECRET_KEY, algorithms=
        email: str = payload.get("sub")
        if email is None:
            raise credentials_exception
    except JWTError:
        raise credentials_exception
    user = await users_collection.find_one({"email": email
    if user is None:
        raise credentials_exception
    return user

@app.get("/")
async def read_index():
    return FileResponse('static/index.html')

@app.post("/register", response_model=UserOut)
async def register(user: UserIn):
    existing_user = await users_collection.find_one({"emai
    if existing_user:
        raise HTTPException(status_code=400, detail="Email
    hashed_password = pwd_context.hash(user.password)
    new_user = {"email": user.email, "hashed_password": ha
    await users_collection.insert_one(new_user)
    return UserOut(**new_user)

@app.post("/login", response_model=Token)
async def login(user: UserIn):
    db_user = await users_collection.find_one({"email": us
    if not db_user or not pwd_context.verify(user.password
        raise HTTPException(status_code=400, detail="Incor

    # 세션 생성 및 Redis에 저장
    session_id = str(uuid.uuid4())
    redis_client.setex(f"session:{session_id}", ACCESS_TOK

    access_token = create_access_token(data={"sub": user.e
    return {"access_token": access_token, "token_type": "b
```

```
@app.get("/me", response_model=UserOut)
async def read_users_me(current_user: dict = Depends(get_c
    return UserOut(email=current_user["email"])

@app.post("/logout")
async def logout(token: str = Depends(oauth2_scheme)):
    try:
        payload = jwt.decode(token, SECRET_KEY, algorithms
        email: str = payload.get("sub")
        # Redis에서 해당 이메일의 모든 세션 삭제
        for key in redis_client.scan_iter(f"session:*"):
            if redis_client.get(key).decode() == email:
                redis_client.delete(key)
        return {"message": "Successfully logged out"}
    except JWTError:
        raise HTTPException(status_code=400, detail="Inval

if __name__ == "__main__":
    import uvicorn
    uvicorn.run(app, host="0.0.0.0", port=8000)
```

4. static/index.html 파일 작성

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, ini
    <title>User Authentication</title>
    <script src="https://cdn.jsdelivr.net/npm/axios/dist/a
    <style>
        body { font-family: Arial, sans-serif; max-width:
        h1 { text-align: center; }
        form { display: flex; flex-direction: column; }
        input, button { margin: 10px 0; padding: 5px; }
        #message { margin-top: 20px; padding: 10px; backgr
    </style>
</head>
```

```
<body>
    <h1>User Authentication</h1>

    <h2>Register</h2>
    <form id="registerForm">
        <input type="email" id="registerEmail" placeholder
        <input type="password" id="registerPassword" place
        <button type="submit">Register</button>
    </form>

    <h2>Login</h2>
    <form id="loginForm">
        <input type="email" id="loginEmail" placeholder="E
        <input type="password" id="loginPassword" placehol
        <button type="submit">Login</button>
    </form>

    <h2>User Info</h2>
    <button id="getUserInfo">Get User Info</button>

    <h2>Logout</h2>
    <button id="logout">Logout</button>

    <div id="message"></div>

    <script>
        const API_URL = '';
        let token = localStorage.getItem('token');

        function showMessage(msg) {
            document.getElementById('message').textContent
        }

        document.getElementById('registerForm').addEventLi
            e.preventDefault();
            const email = document.getElementById('registe
            const password = document.getElementById('regi
            try {
```

```
            const response = await axios.post(`${API_U
            showMessage(`Registered successfully: ${re
        } catch (error) {
            showMessage(`Registration failed: ${error.
        }
    });

    document.getElementById('loginForm').addEventListe
        e.preventDefault();
        const email = document.getElementById('loginEm
        const password = document.getElementById('logi
        try {
            const response = await axios.post(`${API_U
            token = response.data.access_token;
            localStorage.setItem('token', token);
            showMessage('Logged in successfully');
        } catch (error) {
            showMessage(`Login failed: ${error.respons
        }
    });

    document.getElementById('getUserInfo').addEventLis
        if (!token) {
            showMessage('Please login first');
            return;
        }
        try {
            const response = await axios.get(`${API_UR
                headers: { Authorization: `Bearer ${to
            });
            showMessage(`Current user: ${response.data
        } catch (error) {
            showMessage(`Failed to get user info: ${er
        }
    });

    document.getElementById('logout').addEventListener
        if (!token) {
```

```
                showMessage('No user is logged in');
                return;
            }
            try {
                await axios.post(`${API_URL}/logout`, {},
                    headers: { Authorization: `Bearer ${to
                });
                localStorage.removeItem('token');
                token = null;
                showMessage('Logged out successfully');
            } catch (error) {
                showMessage(`Logout failed: ${error.respon
            }
        });
    </script>
</body>
</html>
```

5. python3 app.py 로 서버 실행 → localhost:8000으로 접근