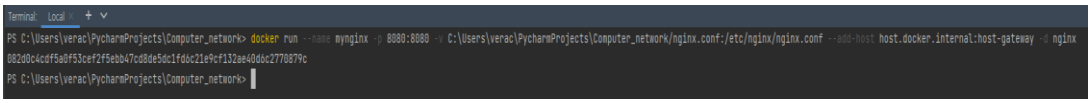
Web Service Deployment

제출일	2022.11.02	**전공**	컴퓨터공학과
과목	컴퓨터네트워크(02)	**학번**	201602037
담당교수	이영석 교수님	**이름**	이규정

① Nginx on Docker reverse proxy 설정



Docker를 실행시킬 때 -v 옵션을 사용하여 로컬에 작성한 nginx.conf 와 container를 연결시킵니다. 단순히 파이썬 workplace 파일을 경로로 설정해주었는데 제대로 실행이 되지 않았습니다. 전체 경로를 사용하여 docker run을 실행해야 제대로 container가 실행되었습니다.

```
PS C:\Users\verac\PycharmProjects\Computer_network> uvicorn --reload --host 0.0.0.0 --port 8889 --root-path /pastebin/api app.main:app
```

이번 실습에서 reverse proxy라는 개념이 나왔는데 간단히 말하면 웹 브라우저(클라이언트)가 웹 서버의 서비스에 접근하려고 할 때, 중간에서 경유지 역할을 하는 서버입니다. 이를 통해 과제에서 8889번 포트를 통해 접속해야 볼 수 있는 서비스들을 8080포트로 접속했을 때, 서비스에 접근할 수 있습니다.

1. Nginx.conf

```
    server {
        listen 8080;

        location /pastebin/api/ {
            proxy_pass http://host.docker.internal:8889/;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header Host $host;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        }
    }

}
```

Nginx.conf에서 변경한 부분입니다. Location을 /Pastebin/api/로 설정을 해줌으로서 URL이 /Pastebin/api/로 시작하는 주소로 접근하면 host.docker.internal:8889로 보내줍니다. Uvicorn을 8889 포트로 설정해주었기 때문에 이렇게 설정해주었습니다.

8080포트로 접속했을 때 8889 포트의 서비스를 이용할 수 있습니다.

2. Pastebin 지난주 과제

 # 코드에 대한 설명은 주석에 달아 놓았습니다. 실습코드에서 추가한 부분만 캡쳐 하였습니다.

1) Models.py

```python
from sqlalchemy import Boolean, Column, ForeignKey, Integer, String
from sqlalchemy.orm import relationship

from app.database import Base


class User(Base):
    __tablename__ = 'users'

    id = Column(Integer, primary_key=True, index=True)
    username = Column(String(length=128), unique=True, index=True)
    salt = Column(String(length=128))
    password = Column(String(length=128))

    pastes = relationship('Paste', back_populates='owner')


class Paste(Base):
    __tablename__ = 'pastes'

    id = Column(Integer, primary_key=True, index=True)
    # Paste 모델을 구현하는 부분입니다.
    # 과제에서 요구하는 title과 content를 추가해주었습니다.
    title = Column(String(length=128))
    content = Column(String(length=128))
    owner_id = Column(Integer, ForeignKey('users.id'))

    owner = relationship('User', back_populates='pastes')
```

2) Schema.py

```python
from typing import List, Union

from pydantic import BaseModel


class PasteBase(BaseModel):
    # Paste 의 BaseModel을 설정하는 부분
    # model에서 추가한 title과 content를 str로 설정해줍니다.
    title: str
    content: str

class PasteCreate(PasteBase):
    # PasteCreate, 즉 생성하는 부분입니다. BaseModel에서 따로 추가해줄 부분이 없기 때문에
    # PasteBase를 그대로 불러옵니다.
    pass

class Paste(PasteBase):
    id: int
    owner_id = int

    class Config:
        orm_mode = True
```

3) Crud.py

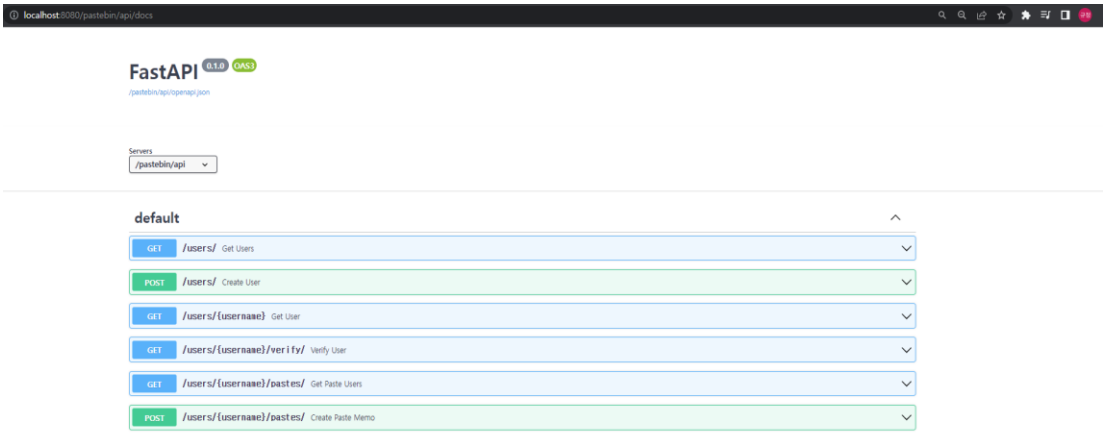
```python
# pasteuser 의 정보를 get 해주는 함수입니다.
# pastuser가 여러 user가 존재하므로 위의 get_users와 파라미터를 똑같이 설정해주었습니다.
# 또한 사용자별 메모를 열람하기 위해 사용자 이름이 필요하므로 username도 파라미터로 설정합니다.
def get_paste_users(db: Session, username: str):
    user_info = db.query(models.User).filter(models.User.username == username).first()
    # user의 info를 get_user 메소드를 불러와서 담아옵니다.
    if user_info:# user_info안에 paste.id와 일치하는 user.id가 있으면 그 paste를 반환합니다.
        # filter 함수는 일종의 조건문, Paste.owner_id와 user_info.id가 같은 paste를 모두(all) 불러옵니다.
        return db.query(models.Paste).filter(models.Paste.owner_id == user_info.id).all()
    else:
        return None


# 사용자별 메모 저장을 위한 함수입니다.
# 사용자 이름, 비밀번호로 사용자 인증 후 메모를 저장하기 때문에 파라미터들을 username, password를 추가해주었습니다.
def create_paste_memo(db: Session, username: str, password: str, paste: schemas.PasteCreate):
    # 검증하는 과정입니다. verify_user을 호출하여 제대로 검증이 됐다면 다음단계로 넘어가고, 검증이 되지 않았다면
    # None을 반환하여 오류 메시지를 나타냅니다.
    db_user = db.query(models.User).filter(models.User.username == username).first()

    m = hashlib.sha256()
    m.update(password.encode('utf-8'))
    m.update(bytes.fromhex(db_user.salt))
    password = m.hexdigest()
    if db_user.password != password:
        return None
    # 검증이 완료되었다면 메모저장을 시작합니다.
    memo = models.Paste(title=paste.title, content=paste.content, owner_id=db_user.id)
    db.add(memo)
    db.commit()
    db.refresh(memo)

    return memo
```

4) Main.py

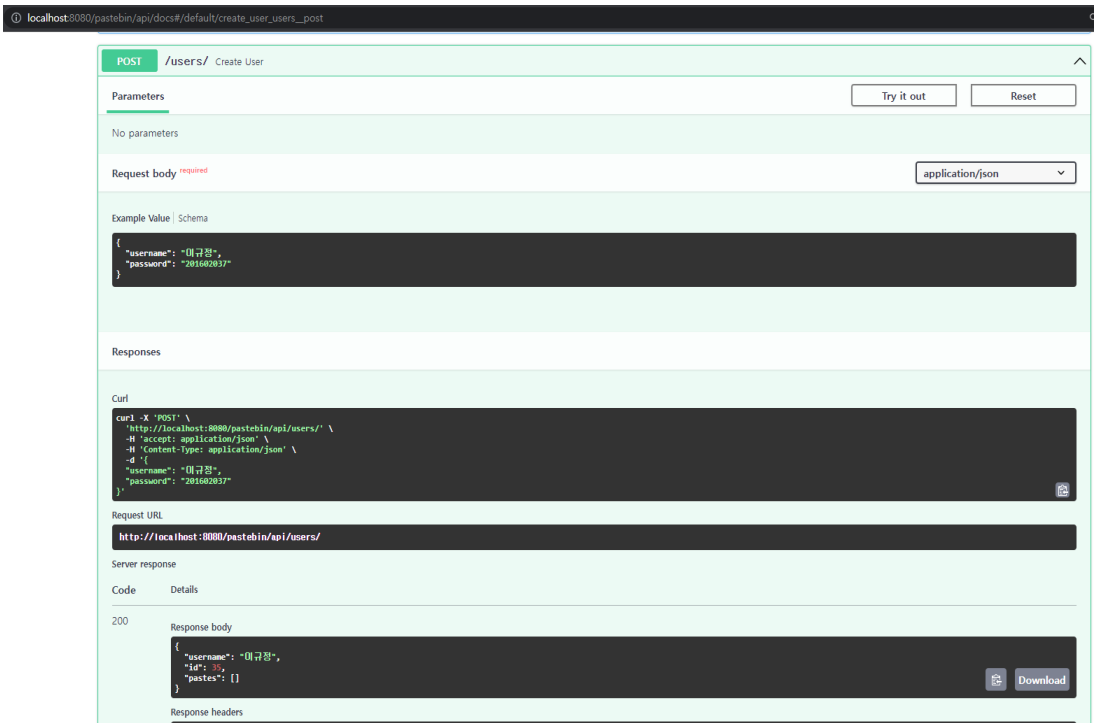
```python
# 메모를 열람하는 함수입니다.
# crud.py에서 작성한 get_paste_users를 불러와 user의 정보를 가져오고 정보가 없으면 404를 반환, 정보가 존재하면 메모를 보여줍니다.
@app.get('/users/{username}/pastes/', response_model=schemas.List[schemas.Paste])
def get_paste_users(username: str, db: Session = Depends(get_db)):
    db_paste = crud.get_paste_users(db, username=username)
    if db_paste is None:
        raise HTTPException(status_code=404, detail='User not found')
    return db_paste

# 메모를 저장하는 함수입니다.
# 열람과 마찬가지로 crud.py에서 작성한 create_paste_memo를 불러와서 저장하고 이 때는 password 정보도 필요하기 때문에 파라미터로 넣어
# 줍니다. create_paste_memo 함수에서 검증이 완료되고 성공적으로 메모저장이 완료되었다면 paste를 반환하고, 실패하면 404를 반환합니다.
@app.post('/users/{username}/pastes/', response_model=schemas.Paste)
def create_paste_memo(username: str, password: str, paste: schemas.PasteCreate, db: Session = Depends(get_db)):
    db_paste = crud.create_paste_memo(db, username=username, password=password, paste=paste)
    if db_paste is None:
        raise HTTPException(status_code=404, detail='User authentication failed')
    return db_paste
```

② 실행 결과 (docs)

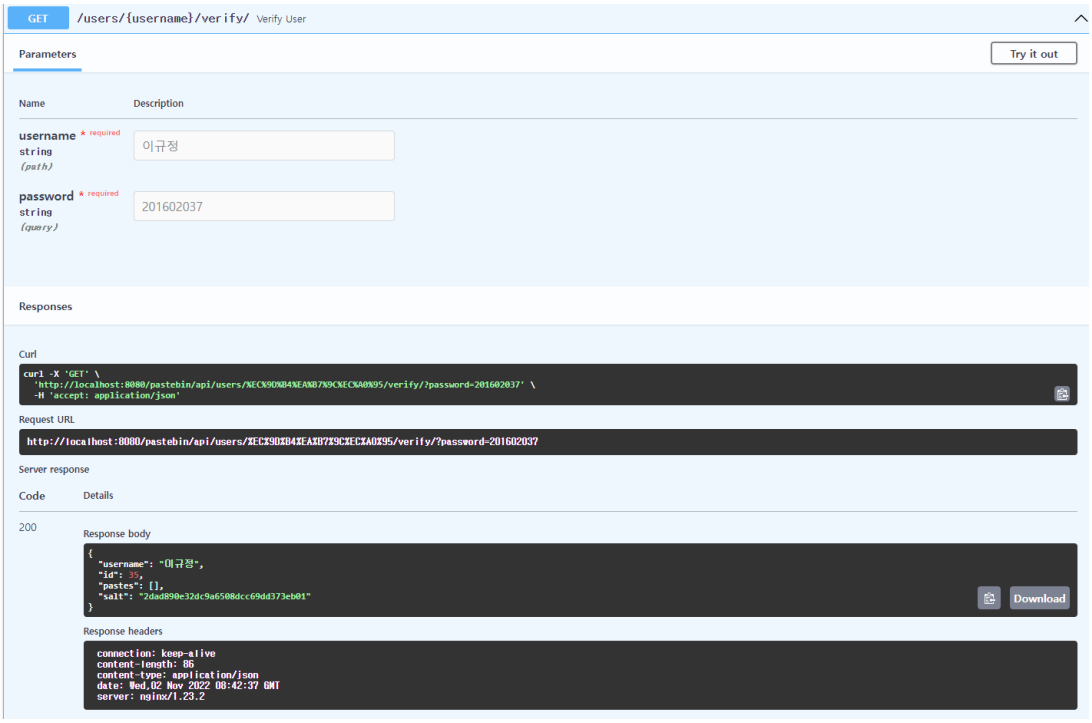
1. Localhost:8080/Pastebin/api/docs



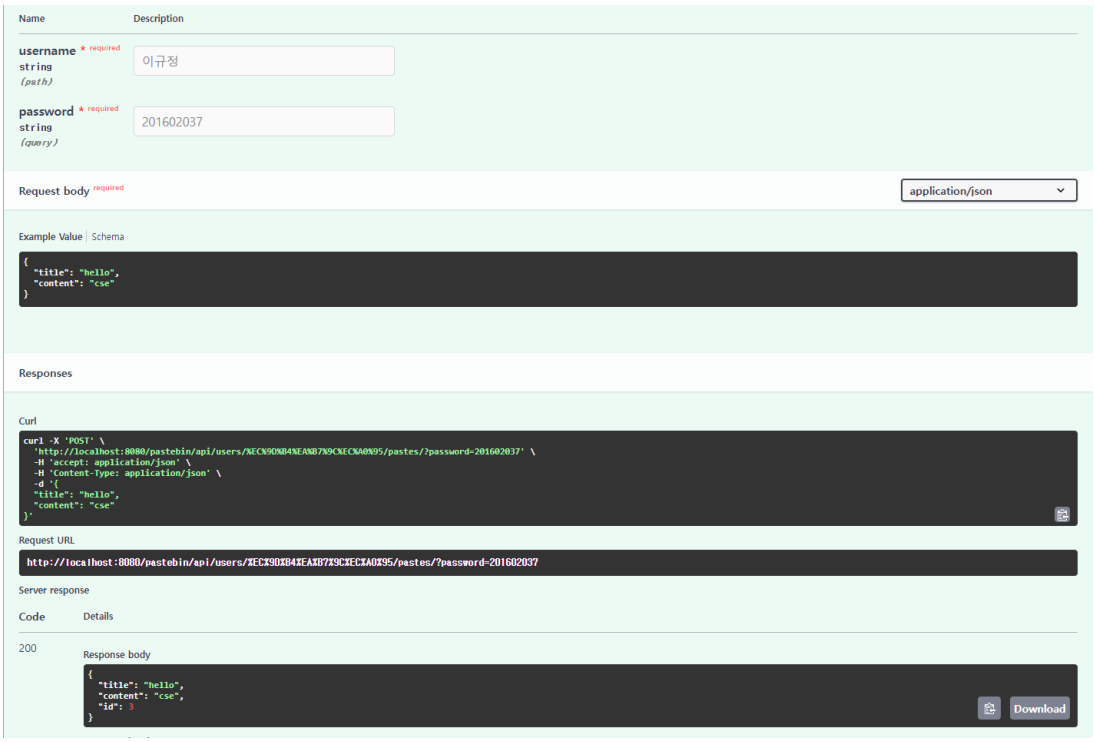
2. 유저 생성(POST)



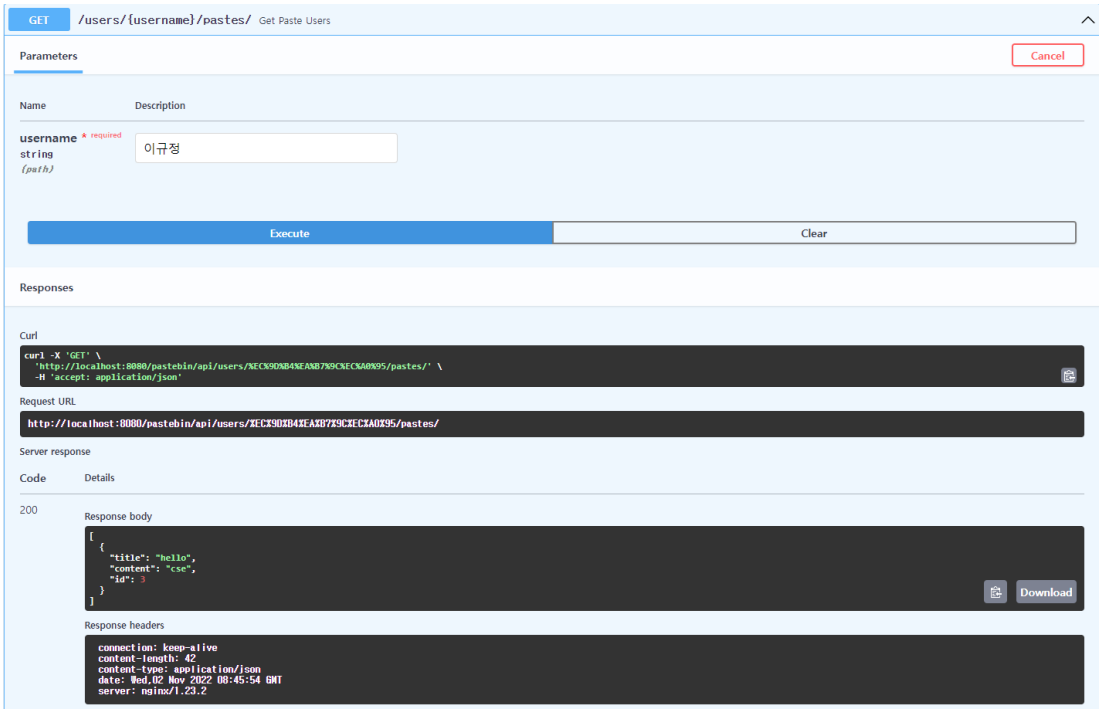
3. 유저 검증


GET /users/{username}/verify/ Verify User
Parameters
Try it out
Name
Description
username * required
string
(path)
이규정
password * required
string
(query)
201602037
Responses
Curl
curl -X 'GET' \
'http://localhost:8080/pastebin/api/users/%EC%9D%B4%EA%B7%9C%EC%A0%95/verify/?password=201602037' \
-H 'accept: application/json'
Request URL
http://localhost:8080/pastebin/api/users/%EC%9D%B4%EA%B7%9C%EC%A0%95/verify/?password=201602037
Server response
Code
Details
200
Response body
{
"username": "이규정",
"id": 35,
"pastes": [],
"salt": "2dad890e32dc9a6508dcc69dd373eb01"
}
Download
Response headers
connection: keep-alive
content-length: 86
content-type: application/json
date: Wed,02 Nov 2022 08:42:37 GMT
server: nginx/1.23.2

4. 메모 저장


Name
Description
username * required
string
(path)
이규정
password * required
string
(query)
201602037
Request body required
application/json
Example Value | Schema
{
"title": "hello",
"content": "cse"
}
Responses
Curl
curl -X 'POST' \
'http://localhost:8080/pastebin/api/users/%EC%9D%B4%EA%B7%9C%EC%A0%95/pastes/?password=201602037' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
"title": "hello",
"content": "cse"
}'
Request URL
http://localhost:8080/pastebin/api/users/%EC%9D%B4%EA%B7%9C%EC%A0%95/pastes/?password=201602037
Server response
Code
Details
200
Response body
{
"title": "hello",
"content": "cse",
"id": 3
}
Download

5. 메모 열람


GET
/users/{username}/pastes/
Get Paste Users
Parameters
Cancel
Name
Description
username
* required
string
(path)
이규정
Execute
Clear
Responses
Curl
curl -X 'GET' \
'http://localhost:8080/pastebin/api/users/%EC%9D%B4%EA%B7%9C%EC%A0%95/pastes/' \
-H 'accept: application/json'
Request URL
http://localhost:8080/pastebin/api/users/%EC%9D%B4%EA%B7%9C%EC%A0%95/pastes/
Server response
Code
Details
200
Response body
[
{
"title": "hello",
"content": "cse",
"id": 3
}
]
Download
Response headers
connection: keep-alive
content-length: 42
content-type: application/json
date: Wed,02 Nov 2022 08:45:54 GMT
server: nginx/1.23.2