컴퓨터 네트워크

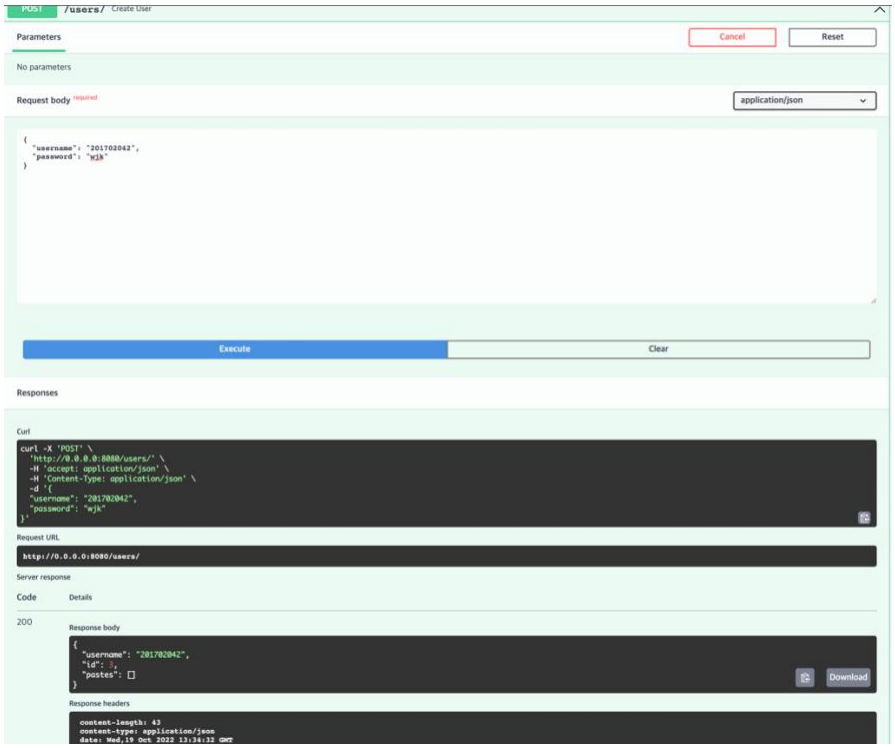
[실습 7]

컴퓨터공학과

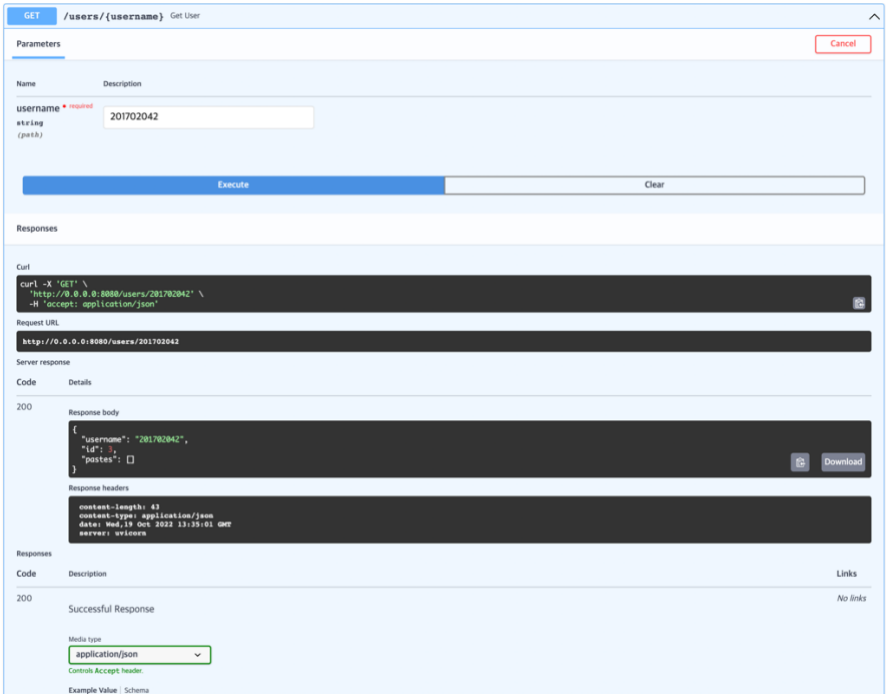
201702042 우정균

사용자 등록 및 검증

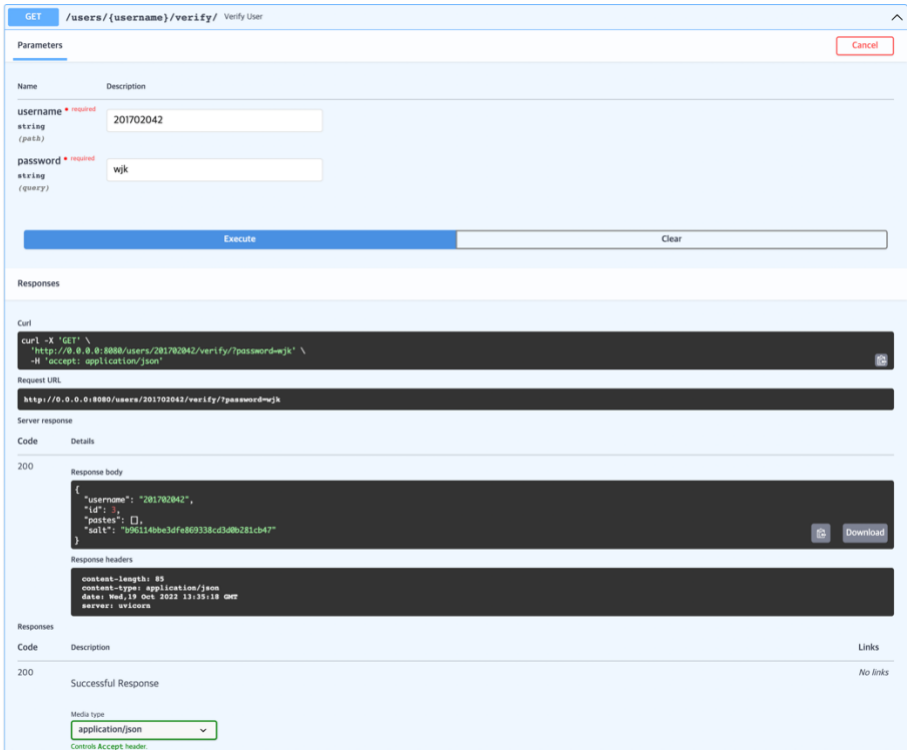
사용자 등록



사용자 조회



사용자 검증


GET
/users/{username}/verify/
Verify User
Parameters
Cancel
Name
Description
username * required
string
(path)
201702042
password * required
string
(query)
wjk
Execute
Clear
Responses
Curl
curl -X 'GET' \
'http://0.0.0.0:8080/users/201702042/verify/?password=wjk' \
-H 'accept: application/json'
Request URL
http://0.0.0.0:8080/users/201702042/verify/?password=wjk
Server response
Code
Details
200
Response body
{
"username": "201702042",
"id": 3,
"postes": [],
"salt": "b96114bbe3dfe869338cd3d0b281cb47"
}
Download
Response headers
content-length: 85
content-type: application/json
date: Wed,19 Oct 2022 13:35:18 GMT
server: uvicorn
Responses
Code
Description
Links
200
Successful Response
No links
Media type
application/json
Controls Accept header.

schemas.py

```python
class PasteBase(BaseModel):
    title: str
    content: str

class PasteCreate(PasteBase):
    pass

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

    class Config:
        orm_mode = True
```

Paste 계열의 클래스 완성.

PasteBase에 title과 content 속성을 추가해줬고, 메모 생성에 사용하는 PasteCreate는 요구하는 정보가 PasteBase의 요소만 필요하므로 그대로 pass로 둔다.

models.py

```python
class Paste(Base):
    __tablename__ = 'pastes'

    id = Column(Integer, primary_key=True, index=True)
    title = Column(String(128))
    content = Column(String(128))
    owner_id = Column(Integer, ForeignKey('users.id'))

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

Paste의 모델 완성

title과 content열을 추가 해줬다.

사용자별 메모 저장

crud.py 소스코드

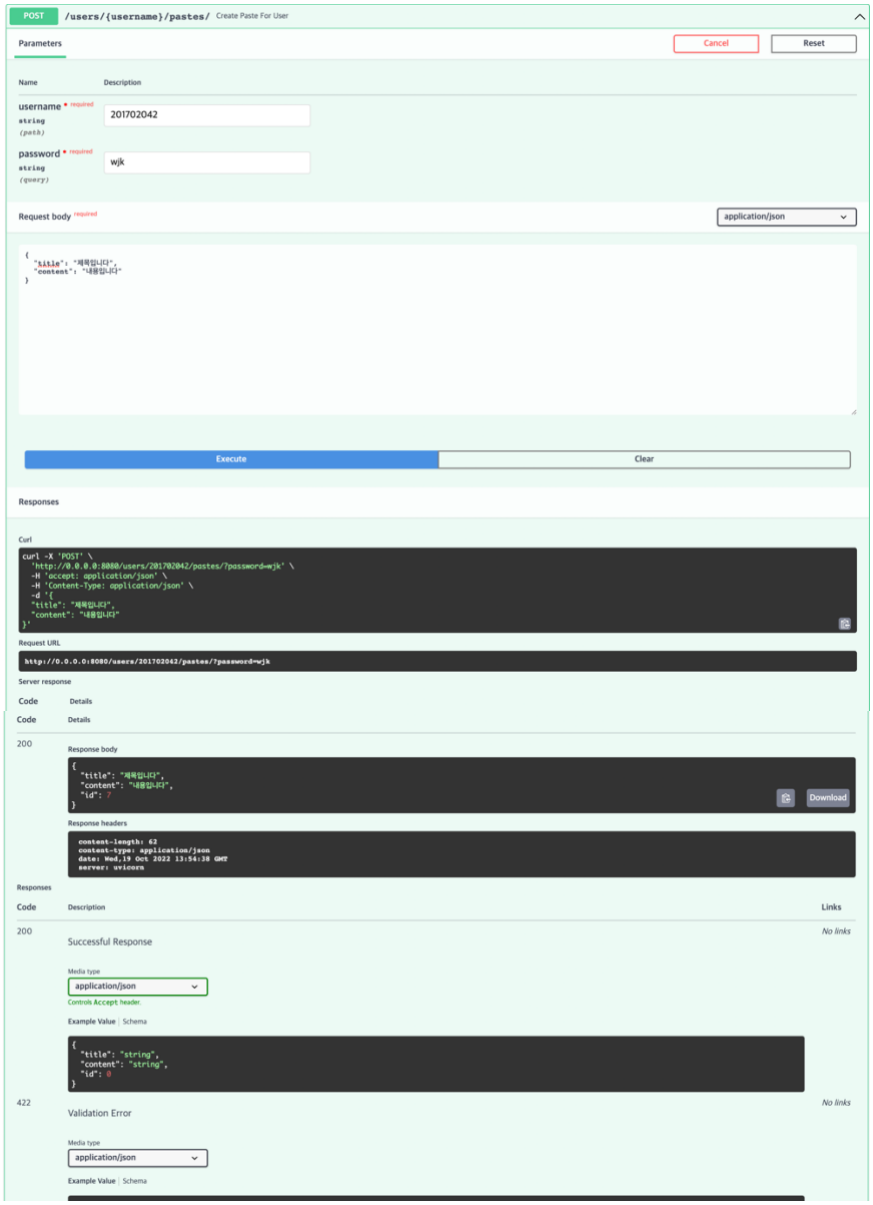
```
def create_paste_for_user(db: Session, username: str, password:str, paste: schemas.
PasteCreate):
    db_user = db.query(models.User).filter(models.User.username == username).first()
    if(db_user):
        m = hashlib.sha256()
        m.update(password.encode('utf-8'))
        m.update(bytes.fromhex(db_user.salt))
        password = m.hexdigest()
        if db_user.password != password:
            return None
    else:
        return None

    db_paste = models.Paste(title=paste.title, content=paste.content, owner_id=db_user.id)
    db.add(db_paste)
    db.commit()
    db.refresh(db_paste)
    return db_paste
```

우선 아이디와 패스워드를 통해 DB의 사용자 정보를 가져오고, 사용자 검증을 진행한다. db_user에 값이 있는지 없는지 if문으로 확인하는 코드는 타입 에러를 방지하기 위해 넣은 코드이다.
사용자 검증이 실패하거나 사용자 정보가 없다면 None을 반환한다.
사용자 검증에 성공하면 Paste 객체를 생성해서 db에 넣어준다.

main.py 소스코드

```
@app.post('/users/{username}/pastes/', response_model=schemas.Paste)
def create_paste_for_user(username: str, password: str, paste: schemas.PasteCreate,
db:Session = Depends(get_db)):
    db_paste = crud.create_paste_for_user(db, username=username, password=password,
    paste=paste)
    if db_paste is None:
        raise HTTPException(status_code=404, detail='User authentication failed')
    return db_paste
```

username은 경로로, password는 쿼리로 받는다. paste 정보는 스키마의 PasteCreate 객체 형태로 받는다.
crud.py파일의 create_paste_for_user() 함수를 실행하고, 반환 값이 None이면 404를 전달하고 값이 있다면 응답 값으로 보내준다.
응답 값의 타입은 schemas의 Paste 클래스 형태이다.



docs에서 실행한 모습

사용자별 메모 열람

crud.py 소스코드

```
def get_pastes_for_user(db: Session, username: str, skip: int = 0, limit: int = 100):
    db_user = db.query(models.User).filter(models.User.username == username).first()
    if db_user:
        return db.query(models.Paste).filter(models.Paste.owner_id == db_user.id).offset
        (skip).limit(limit).all()
    else:
        return None
```

매개변수로 username과 skip, limit을 받는다. username으로 db의 유저 테이블에서 해당 유저 정보를 가져온다. 가져온 유저 정보의 id를 사용해서 해당 id값을 owner_id로 갖는 paste들을 모두 가져온다. 이 때 skip과 limit을 활용해서 그 사이 수만큼의 데이터만 가져온다.

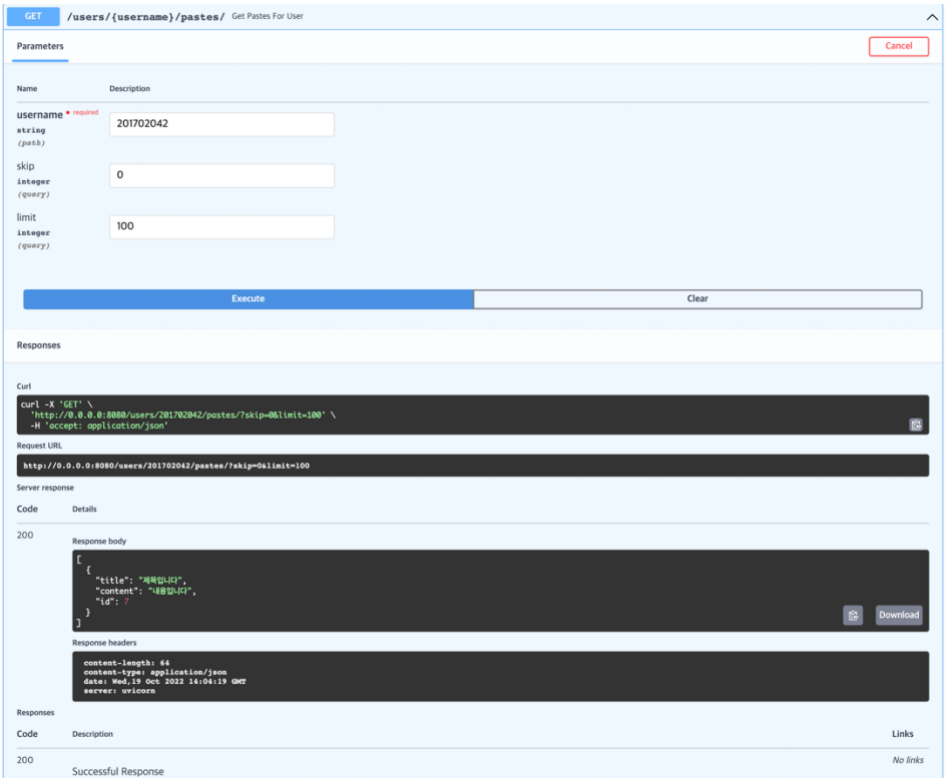
main.py 소스코드

```
@app.get('/users/{username}/pastes/', response_model=List[schemas.Paste])
def get_pastes_for_user(username: str, skip: int = 0, limit: int = 100, db: Session = Depends
(get_db)):
    db_pastes = crud.get_pastes_for_user(db, username=username, skip=skip, limit=limit)
    if db_pastes is None:
        raise HTTPException(status_code=404, detail='User not found')
    return db_pastes
```

username을 경로로 받고, skip과 limit을 쿼리로 받는다.

crud의 get_pastes_for_user() 함수를 사용해서 pastes 값들을 모두 가져오고 값이 없다면 404 에러를 전송, 값이 있다면 응답 값으로 보내준다.

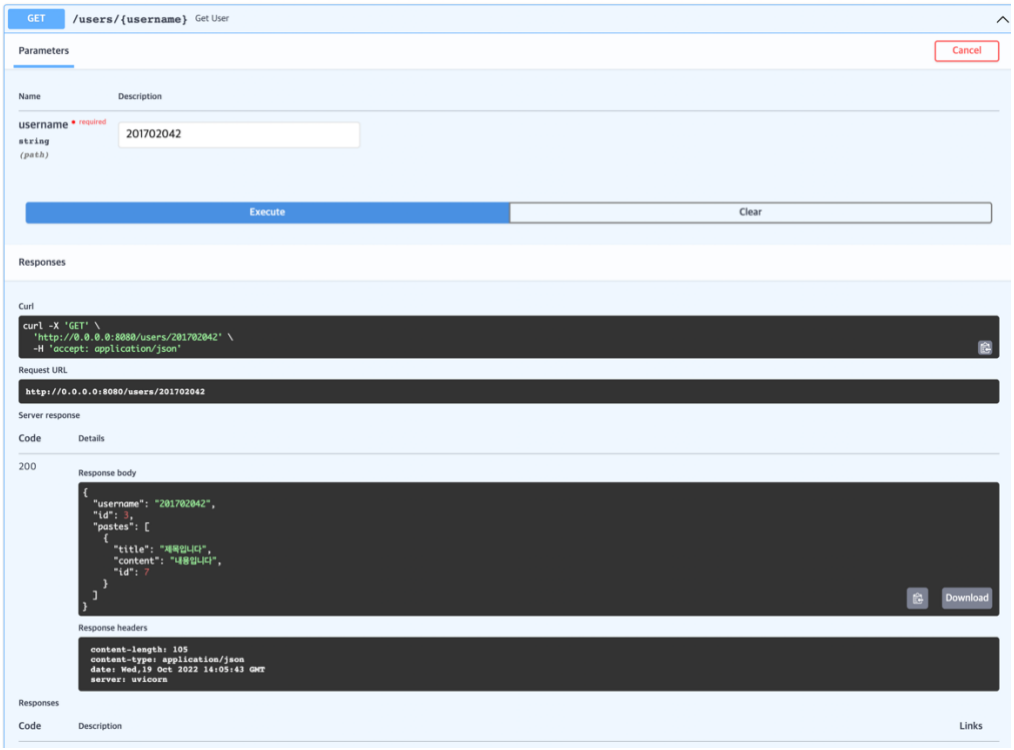
응답 타입은 Paste의 리스트 형태이다.



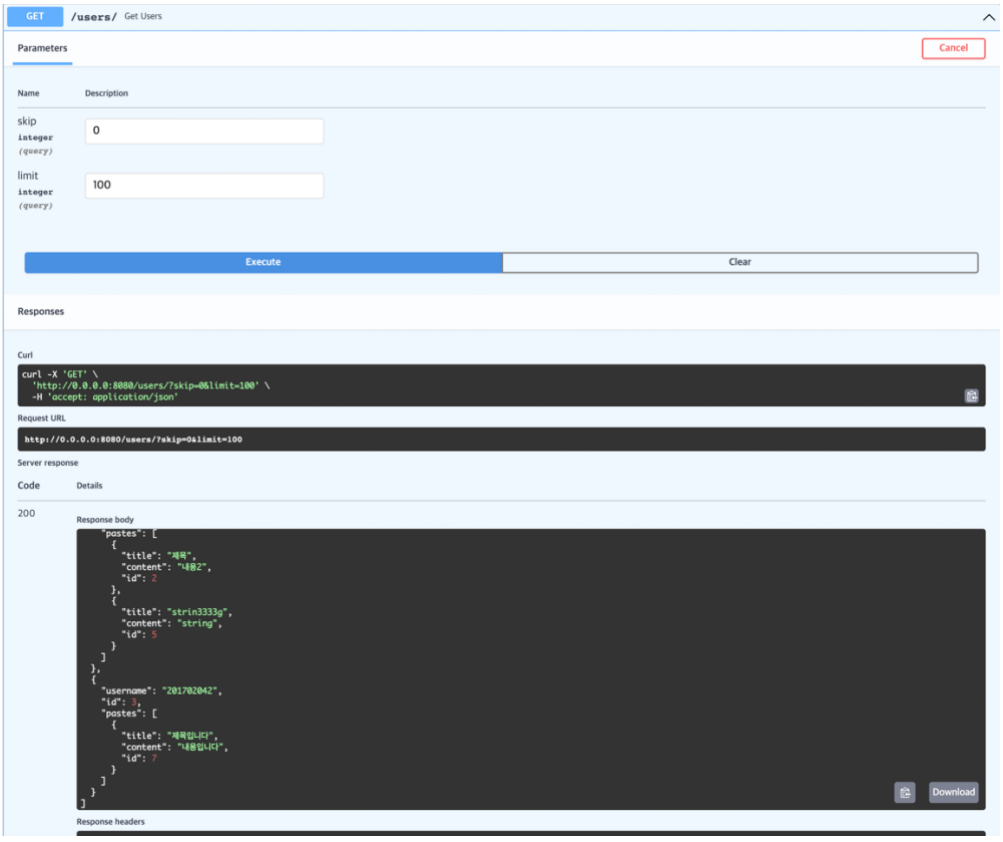
docs에서 실행한 모습

사용자 정보 열람시 사용자 메모도 함께 반환

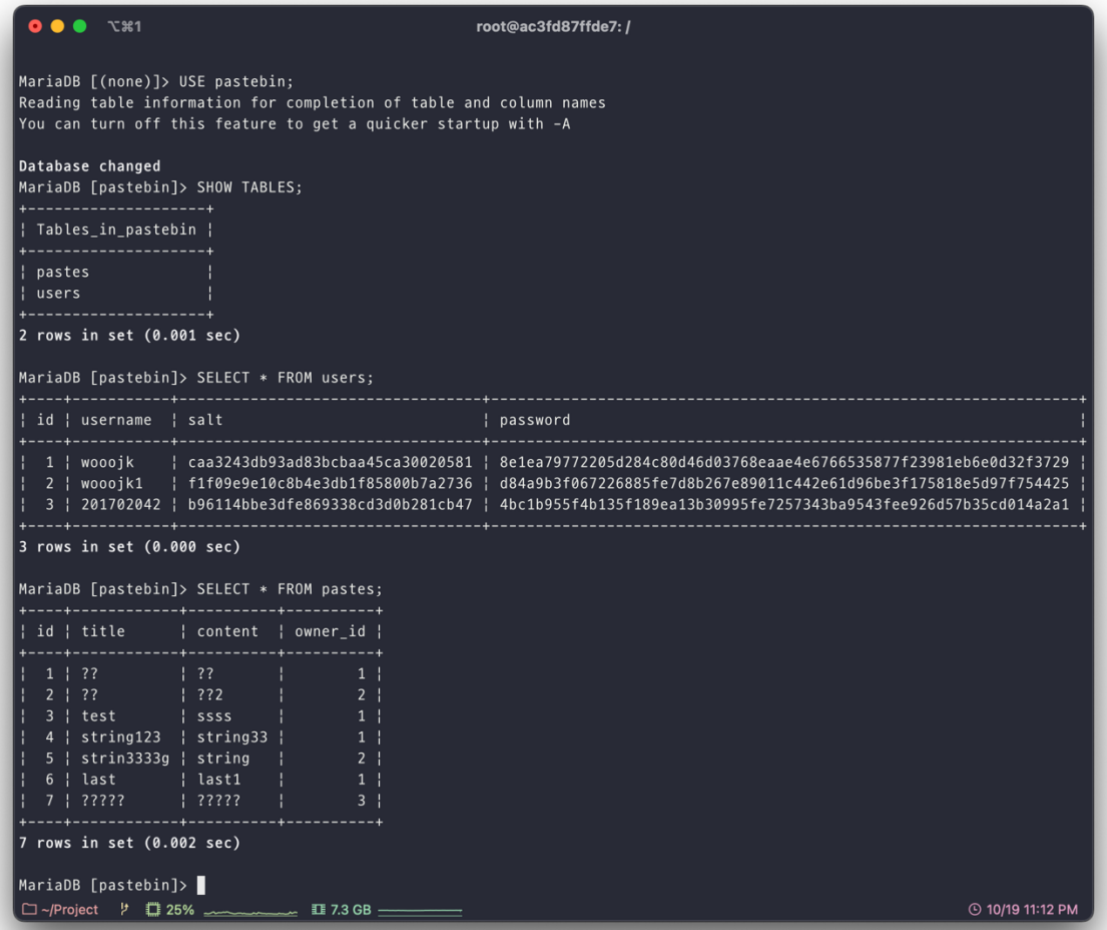
GET /users/{username} 메소드



GET /users/ 메소드



MariaDB 에 저장 캡쳐



파이썬과 MariaDB 연결

```python
from curses import echo
import os
from sqlalchemy import create_engine
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker

# user_id = os.environ.get("MYSQL_USER")
# user_pw = os.environ.get('MYSQL_PASSWORD')
# port = os.environ.get('MYSQL_PORT')
# db = os.environ.get('MYSQL_DATABASE')

# SQLALCHEMY_DATABASE_URL = 'mariadb+mariadbconnector://pasteuser:user-secret-pw@127.0.0.
1:3306/pastebin'
SQLALCHEMY_DATABASE_URL = "mysql+mysqlconnector://pasteuser:user-secret-pw@127.0.0.1:3306/
pastebin"

engine = create_engine(SQLALCHEMY_DATABASE_URL, echo=True)
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)

Base = declarative_base()
```

기존의 mariadb connector 를 사용하면 파이썬과 mariadb 가 연결되지 않는다.
대안으로 mysql-connector 를 사용해서 실행하니 잘 연결되었다.