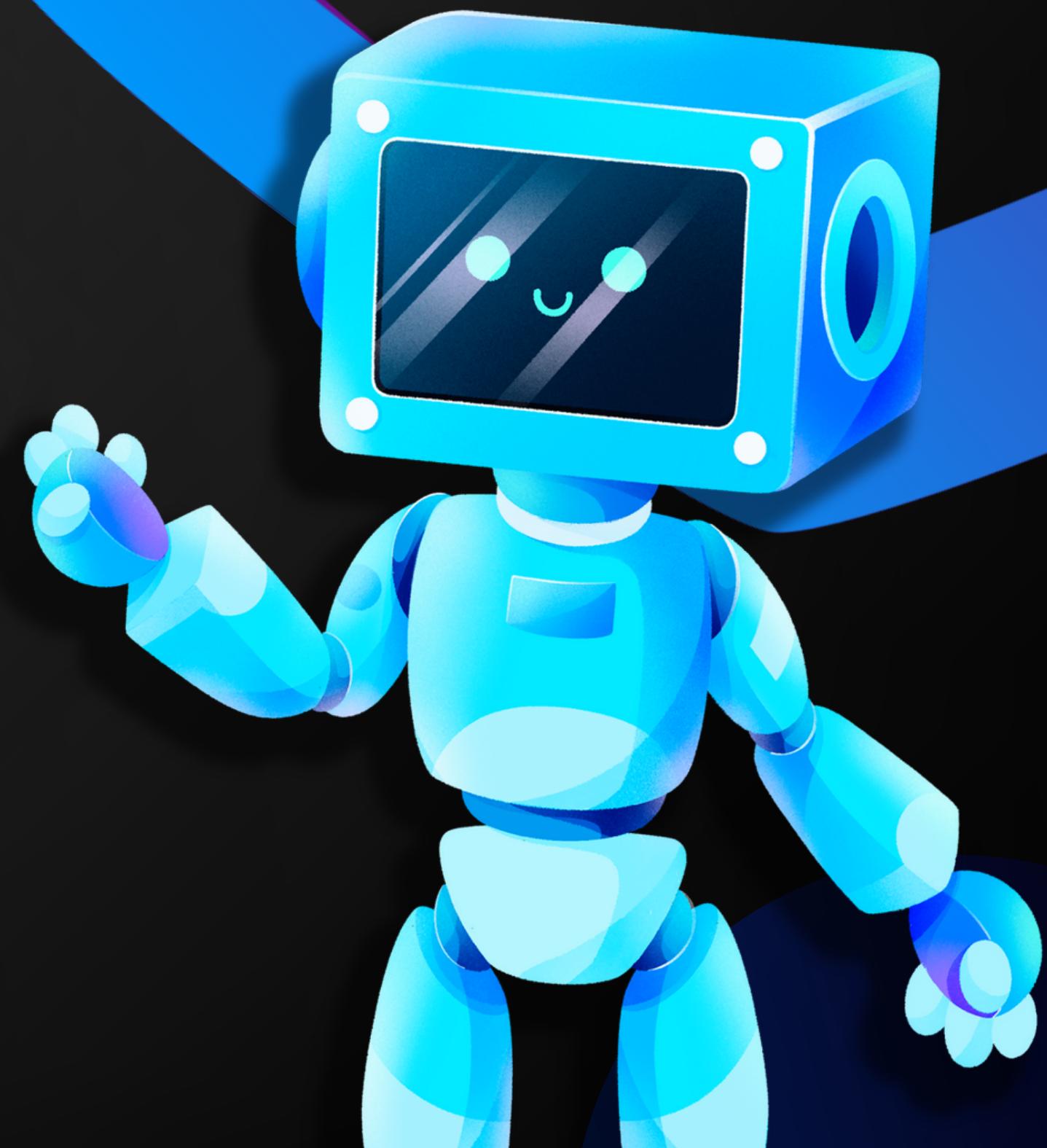


# NEOAPP PROJECT

by Лунтики



# Задачи



Работа в команде



Поиск решений



# Брейншторм

Анализ



Технологии



Идеи



Согласование



# Карта сайта



# Frontend

**ФОРМА  
АВТОРИЗАЦИИ**

**ЛИЧНЫЙ  
КАБИНЕТ**

**ФОРМА  
СОЗДАНИЯ  
ЗАПРОСА**

**ФОРМА  
СОГЛАСИЯ**

**HTML**

**CSS**

**JAVASCRIPT**

# BACKEND

## Структура проекта

Screenshot of a code editor showing the project structure and a Python script. The project structure includes files like crud.py, api.py, schemas.py, models.py, and main.py. The crud.py file contains code for database operations and API endpoints.

```
EXPLORER
...
HACK
> __pycache__
> env
neoapp
> __pycache__
> app
> __pycache__
> api
> endpoints
> __pycache__
api.py
crud.py
schemas.py
models.py
init.py
> db
> __pycache__
migrations
tests
> .env
alembic.ini
pyproject.toml
readme.txt
requirements.txt
main.py
requirements.txt
```

```
crud.py
neoapp > app > api > endpoints > api.py > get_owner_pending_requests
6     from schemas import LogInRequest, RequestCreate, RequestEdit, OwnerProfile, Employee
7     from fastapi import Header
8     from typing import List
9     from sqlalchemy.sql import text
10    from neoapp.app.api.models.employee import Сотрудники
11    from neoapp.app.api.models.request import Запросы
12    import jwt
13    from jwt import PyJWTError
14    from dotenv import load_dotenv
15    import os
16    import typing
17    from typing import Callable
18    # Загрузка переменных среды из файла .env
19    load_dotenv()
20
21    # Получение значения SECRET_KEY из переменных среды
22    SECRET_KEY = os.getenv("SECRET_KEY")
23
24    ALGORITHM = os.getenv("ALGORITHM", "HS256")
25
26
27    router = APIRouter()
28
29    # Эндпоинт для аутентификации пользователей
30    @router.post("/login/")
31    def login(request: schemas.LoginRequest, db: Session = Depends(get_db)):
32        user = crud.authenticate_user(db, request.Логин, request.Пароль)
33        if not user:
34            raise HTTPException(
35                status_code=status.HTTP_401_UNAUTHORIZED,
```

## Тесты

Screenshot of Postman showing a POST request to the login endpoint. The request body is a JSON object with 'Логин' (Login) and 'Пароль' (Password) fields. The response shows a generated access token.

POST аутентификацию

POST новые запросы

GET все сотрудники

GET Просмотр на рассмотрении

GET подключенность к бд

PUT редактировать запрос

GET история запросов

GET просмотр рассмотренных за...

POST наложение правил на запрос

Body

Params Authorization Headers (8) Body \* Pre-request Script Tests Settings

Body

JSON

1 {  
2 "Логин": "sasha@example.com",  
3 "Пароль": "password4"  
4 }  
5

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON

1 {  
2 "access\_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJQsInJvIgAIPfu0CK8KbjBt3gjBxQ\_vgFmi0JQmh67eXKmb8"  
3 }

Фамилия	Имя	Должность	Логин(почта)	Пароль	Роль_Б
Иванов	Иван	Директор	ivan@example.com	password1	dir
Петров	Петр	Директор	petr@example.com	password2	dir
Сидоров	Алексей	Директор	alex@example.com	password3	dir
Козлов	Александр	Менеджер	sasha@example.com	password4	admin
Морозова	Елена	Менеджер	elena@example.com	password5	user
Новиков	Дмитрий	Менеджер	dima@example.com	password6	user
Федорова	Анна	Программист	anna@example.com	password7	user
Кузнецов	Олег	Маркетолог	oleg@example.com	password8	user
Белов	Сергей	Бизнес-аналитик	sergei@example.com	password9	user
Васильев	Николай	Программист	nick@example.com	password10	user
	NULL	NULL	NULL	NULL	NULL

PYTHON

POSTMAN

MYSQL

# Команда

