Документација за проект: React + FastAPI Корисничка автентикација

Вовед

За овој проект направив веб апликација за регистрација и најава на корисници. Идејата беше да направам нешто функционално со современи технологии - React за frontend и FastAPI за backend. Системот користи JWT токени за автентикација и SQLite база за чување на податоците.

Технологии

Frontend:

- React со Vite
- JavaScript и CSS за стилизација

Backend:

- FastAPI (Python)
- SQLAlchemy за работа со база
- SQLite за складирање
- bcrypt за хеширање на лозинки
- JWT токени за автентикација
- python-dotenv за конфигурација

Безбедност:

- SSL сертификати (self-signed)
- JWT токени
- bcrypt хеширање на лозинки

Структура на проектот

```
/
├── frontend/
│   ├── src/
│   │   ├── components/
│   │   │   ├── SignUpForm.jsx
│   │   │   ├── LoginForm.jsx
│   │   │   └── ProtectedContent.jsx
│   │   ├── App.jsx
│   │   └── main.jsx
│   ├── index.css, App.css
│   ├── vite.config.js
│   └── SSL сертификати
│
└── backend/
    ├── main.py
    ├── models.py
    ├── auth.py
    ├── database.py
    ├── .env
    ├── .venv/
    ├── users.db
    └── SSL сертификати
```

Инсталација и подигање

Frontend

1. Инсталирај зависности:

npm install

2. Стартувај development сервер:

npm run dev

3. Генерирај SSL сертификати и стави ги во frontend и backend фолдерите.
4. Конфигурирај ги во vite.config.js.

Backend

1. Инсталирај потребни пакети:

pip install fastapi uvicorn sqlalchemy bcrypt python-dotenv pyjwt

2. Креирај .env фајл и додај:

SECRET_KEY=твојТаенКлуч

3. Стартувај сервер:

uvicorn main:app --reload --ssl-keyfile=path/to/key.pem --ssl-certfile=path/to/cert.pem

Базата users.db се креира автоматски кога ќе го стартуваш серверот прв пат.

Функционалности

Регистрација

Направив форма каде корисникот внесува корисничко име, email, лозинка и confirmation на лозинката. Има валидација - лозинката мора да содржи голема буква, мала буква, број и специјален знак. Ако сè е во ред, серверот зачувува корисничко име, email и лозинката како bcrypt хеш.

Најава

Корисникот се најавува со корисничко име и лозинка. Ако се точни податоците, серверот му враќа JWT токен кој го чувам во localStorage на прелистувачот.

Заштитена содржина

Оваа страна може да ја отвори само најавен корисник. Кога се праќа барање кон серверот, JWT токенот се испраќа во Authorization header. Ако токенот е невалиден или истечен, автоматски го одјавувам корисникот.

Одјавување

Едноставно се брише токенот од localStorage и корисникот се враќа на login страната.

Клучни фајлови

Frontend компоненти

App.jsx - Главната компонента што управува со целата апликација. Прикажува Login, Sign Up или заштитената содржина зависно дали корисникот е најавен.

SignUpForm.jsx - Регистрациска форма со валидација. Испраќа POST барање кон /signup endpoint.

LoginForm.jsx - Форма за најава. По успешна најава го зачувува JWT токенот.

ProtectedContent.jsx - Прикажува заштитена содржина. Прави fetch барање со JWT токен и ако не успее, корисникот се одјавува.

Backend компоненти

main.py - Главната FastAPI апликација со сите routes (/signup, /login, /protected).

auth.py - Содржи функции за работа со JWT токени - креирање, декодирање и верификација.

database.py - Setup за SQLAlchemy и конекција со базата.

models.py - User модел за ORM.

.env - Чува го SECRET_KEY за JWT токените.

Безбедносни мерки

- Лозинките не се чуваат како plain text, туку како bcrypt хешови
- Користам JWT токени за автентикација наместо сесии
- Имплементирав CORS за да дозволам само барања од frontend
- SSL сертификати за HTTPS комуникација
- Валидација на внесени податоци и на frontend и на backend

Тестирање

1. Стартувај го backend серверот
2. Стартувај го frontend
3. Регистрирај нов корисник со валидни податоци
4. Најави се со тие податоци
5. Провери дека можеш да пристапиш до заштитената страна
6. Тестирај одјавување и повторна најава

Забелешки

- Користам локална SQLite база users.db за чување на податоците
- SSL сертификатите се self-signed бидејќи ова е локален проект
- SECRET_KEY е во .env фајл кој не треба да се споделува
- Дизајнот можеше да биде подобар, ама фокусот беше на функционалност и безбедност

Автор: Јован Парлапанов
Факултет: Факултет за информатички науки и компјутерско инженерство
Предмет: Информациска безбедност
Година: 2025/2026