SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

DOKUMENTÁCIA K INTERNETOVÉMU OBCHODU S ELEKTRONIKOU

SEMESTRÁLNY PROJEKT

2024 **Eshop**

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

DOKUMENTÁCIA K INTERNETOVÉMU OBCHODU S ELEKTRONIKOU

SEMESTRÁLNY PROJEKT

Študijný program:	Aplikovaná informatika
Predmet:	I–ASOS – Architektúra softvérových systémov
Garant predmetu:	Ing. Viliam Hromada, PhD.
Cvičiaci:	Ing. Stanislav Marochok

Bratislava 2024 **Eshop**

Obsah

Zoznam výpisov

1 Štruktúra projektu

Vytvorili sme jedoduchú webovú aplikáciu, ktorá ponúka zjednodušené funkcionality internetového obchodu. Štruktúra projektu pozostáva z troch hlavných častí: backend, frontend a infraštruktúra.

1.1 Backend

Backendová časť projektu je zameraná na spracovanie API požiadaviek a databázovú logiku. Je implementovaná v Python-e a využíva nástroje ako Alembic pre databázové migrácie.

Štruktúra aplikácie:

Logika aplikácie sa nachádza v adresári `app/`, ktorý má nasledujúcu štruktúru:

- `config.py`: Konfigurácia aplikácie.
- `main.py`: Vstupný bod pre backend.
- `middlewares/`: Implementuje middleware, napr. kontrolu prihlásenia.
- **routers/**: API routy (napr. autentifikácia, správa produktov, košík).
- `models/`: Databázové modely entít (používateľ, produkt, objednávka).
- `migrations/`: Skripty pre databázové migrácie pomocou Alembic.

Okrem adresára `app/` sa v adresári `backend` nachádzajú aj súbory:

- `Dockerfile`: Definuje obraz pre backend kontajner.
- `alembic.ini`: Konfigurácia pre Alembic.
- `requirements.txt`: Zoznam potrebných Python knižníc.

1.2 Frontend

Frontendová časť projektu je implementovaná v TypeScript a Vue.js. Poskytuje užívateľské rozhranie pre interakciu s aplikáciou.

Štruktúra aplikácie:

Zdrojové súbory sa nachádzajú v adresári `src/`, ktorý má nasledujúcu štruktúru:

- `App.vue`: Hlavná komponenta aplikácie.
- `components/`: Znovupoužiteľné komponenty, ako tlačidlá a formuláre.
- `pages/`: Jednotlivé stránky aplikácie (autentifikácia, administrácia, produkty).
- `stores/`: Manažment stavov pre autentifikáciu a košík.
- `locales/`: Prekladové súbory.

Okrem adresára `src/` sa v adresári `frontend` nachádzajú aj súbory:

- `Dockerfile`: Definuje obraz pre frontend kontajner.
- `index.html`: Koreňový HTML súbor pre aplikáciu.
- `package.json`: Zoznam závislostí a skriptov projektu.

1.3 Infraštruktúra

Infraštruktúrna časť projektu zabezpečuje správu sieťových požiadaviek a ukladanie súborov.

- `nginx/`: Konfigurácia webového servera Nginx.
  - `nginx.conf`: Hlavná konfigurácia.
  - `conf.d/`: Dodatočné konfigurácie.
- `seaweedfs/`: Konfigurácia pre distribuovaný systém súborov SeaweedFS.

2 Kontajnerizácia

Kontajnerizácia v projekte je riešená prostredníctvom Dockeru. Projekt využíva Docker Compose na orchestráciu viacerých služieb: backendu, frontendu, databázy a webového servera Nginx. Každá komponenta má vlastný Dockerfile, ktorý špecifikuje postup zostavenia obrazu:

- **Backend:** Obsahuje Python aplikáciu, pričom Dockerfile definuje inštaláciu závislostí zo súboru requirements.txt, nastavenie pracovného adresára, a spustenie aplikácie. Počas build procesu sa tiež aplikujú databázové migrácie pomocou Alembic.
- **Frontend:** Dockerfile zahŕňa inštaláciu balíkov definovaných v package.json a následne kompiláciu Vue.js aplikácie. Hotová aplikácia je optimalizovaná a nasadená do produkčného obrazu.
- **Databáza a infraštruktúrne komponenty:** V Docker Compose sú zahrnuté služby ako Nginx a SeaweedFS, kde sa používajú preddefinované oficiálne obrazy s vlastnými konfiguráciami.

2.1 Build a Deploy

Build a nasadenie aplikácie prebieha pomocou jednoduchých príkazov:

- `docker-compose build`: Zostaví všetky potrebné obrazy.
- `docker-compose up`: Spustí celú aplikáciu vrátane všetkých služieb.

3 Databáza

Databáza je implementovaná pomocou SQLLite

3.1 Tabuľky

Tabuľka users: slúži na ukladanie údajov o používateľoch

- **id**: Primárny kľúč (Integer, Auto Increment).
- **username**: Používateľské meno (String, Unique).
- **email**: E-mailová adresa (String, Unique).
- **hashed_password**: Heslo uložené v hashovanom formáte (String).
- **is_active**: Príznak aktivity používateľa (Boolean, default = True).
- **is_admin**: Príznak administrátorského prístupu (Boolean, default = False).

Tabuľka products: Obsahuje informácie o produktoch dostupných v aplikácii.

- **id**: Primárny kľúč (Integer, Auto Increment).
- **name**: Názov produktu (String).
- **description**: Popis produktu (Text).
- **price**: Cena produktu (Float).
- **category_id**: Cudzí kľúč odkazujúci na kategóriu (Integer).
- **stock**: Počet dostupných kusov na sklade (Integer).

Tabuľka categories:

- **id**: Primárny kľúč (Integer, Auto Increment).
- **name**: Názov kategórie (String, Unique).
- **description**: Popis kategórie (Text).

Tabuľka orders:

- **id**: Primárny kľúč (Integer, Auto Increment).
- **user_id**: Cudzí kľúč odkazujúci na používateľa (Integer).

- **total_price**: Celková cena objednávky (Float).
- **status**: Stav objednávky (String, napr. "pending", "completed").
- **created_at**: Dátum a čas vytvorenia (Datetime).

Tabuľka cart: Slúži na ukladanie položiek v nákupnom košíku.

- **id**: Primárny kľúč (Integer, Auto Increment).
- **user_id**: Cudzí kľúč odkazujúci na používateľa (Integer).
- **product_id**: Cudzí kľúč odkazujúci na produkt (Integer).
- **quantity**: Počet kusov daného produktu (Integer).

3.2 Integrita tabuľky

3.3 Relácie

- **users -> orders**: Jeden používateľ môže mať viacero objednávok.
- **categories -> products**: Každý produkt patrí do jednej kategórie.
- **users -> cart**: Každý používateľ má svoj nákupný košík.

3.4 Indexy a unikátnosť

Tabuľky obsahujú unikátne kľúče pre polia ako username a email. Indexy sú automaticky generované pre primárne a cudzie kľúče.

4 Základné funkcionality

4.1 Registrácia a Prihlásenie

Backend

Backend je implementovaný pomocou FastAPI a podporuje autentifikáciu OAuth2 s heslami. Nasledujúce API endpointy sa nachádzajú v súbori `backend/app/routers/auth.py`:

- `POST /auth/login`
  - Slúži na prihlasovanie užívateľov.
  - Očakáva formulár s parametrami username a password.
  - Validuje užívateľa pomocou hesla uloženého v databáze.
  - Generuje JWT token pri úspešnom prihlásení.
- `POST /auth/register`
  - Slúži na registráciu nových užívateľov.
  - Očakáva informácie o užívateľovi (napr. meno, e-mail, heslo).
  - Hashuje heslo pomocou passlib a ukladá užívateľa do databázy.
- `POST /auth/logout` : Ruší aktuálnu reláciu užívateľa odstránením tokenu.

Použité knižnice a technológie

- FastAPI: Framework pre tvorbu REST API.
- Passlib: Hashovanie hesiel pomocou algoritmu bcrypt
- JWT: Generovanie a validácia tokenov pre autentifikáciu.
- SQLModel: Práca s databázou a modelovanie užívateľských objektov.

```
@router.post("/login")
def login(form_data: OAuth2PasswordRequestForm, db: DbSession):
    user = db.query(User).filter(User.username == form_data.username).first()
    if not user or not pwd_context.verify(form_data.password, user.hashed_password):
        raise HTTPException(status_code=400, detail="Incorrect username or password")

    token = jwt.encode({"sub": user.username}, settings.secret_key, algorithm="HS256")
```

```
return {"access_token": token, "token_type": "bearer"}
```

Výpis 1: Zjednodušená ukážka loginu

Frontend Frontend je implementovaný pomocou Vue.js. V adresári `frontend/src/pages/auth/` sa nachádzajú nasledujúce komponenty:

- `login.vue`
  - Poskytuje formulár na zadanie mena a hesla.
  - Odošle údaje na endpoint `/auth/login`.
  - Pri úspechu uloží token do lokálneho úložiska.
- `register.vue`
  - Poskytuje formulár na vytvorenie nového účtu.
  - Odošle údaje na endpoint `/auth/register`.
  - Validuje heslo a e-mail pred odoslaním.

4.2 Produkty

Produkty je možné si prehliadať, filtrovať na základe typu a značky, vyhľadať podľa názvu, pridať do košíka a objednať. Pridať nový produkt, značku alebo typ je možné v admin paneli na endpointe /admin.

Backend na zobrazenie produktu

Endpointy:

- GET /products: Poskytuje zoznam všetkých produktov.
- GET /products/id: Poskytuje detailné informácie o konkrétnom produkte.
- Implementované v súbore: `backend/app/routers/products.py`

Model produktu:

- Nachádza sa v súbore `backend/app/models/product.py`
- Využíva SQLAlchemy na mapovanie vlastností produktu, ktoré sú popísané v kapitole 3.

Seedovanie dát: V súbore `backend/app/seed.py` sú ukážkové produkty inicializované do databázy pre testovanie.

Validácia: Pydantic modely v `backend/app/routers/products.py` validujú vstupy a výstupy.

Frontend: Zobrazenie produktov

Hlavné komponenty:

- `ProductSearch.vue`: Komponenta umožňuje vyhľadávanie produktov podľa názvu.
- `ProductList.vue` (pravdepodobne, podľa konvencie): Zobrazuje zoznam produktov.
- `ProductDetail.vue`: Detailný pohľad na produkt (pravdepodobne odkazuje na /products/id).

Store a typy:

- Store `frontend/src/stores/cart.ts` obsahuje manipuláciu s produktmi v košíku.
- Typy pre produkty sú definované v frontend/src/types/api.ts.

Zobrazenie produktu v systéme

Frontend cez endpoint `/products` načíta zoznam produktov. Tieto produkty sa zobrazia v zozname alebo ako výsledky vyhľadávania. Po kliknutí na konkrétny produkt frontend načíta údaje z endpointu `/products/id`. Detailný pohľad obsahuje informácie, ako sú názov, popis, cena, skladová dostupnosť, a kategória. Komponenta `ProductSearch.vue` umožňuje používateľovi filtrovať produkty podľa názvu. StreamingResponse sa používa na poskytovanie dát v reálnom čase alebo na streamovanie údajov o produktoch (stav zásob).

5 Endpointy

Dokumentácia k endpointom je popísaná v Swaggeri. Niektoré časti sú popísané aj v tejto kapitole.

Hlavné endpointy:

- Autentifikácia
  - POST /auth/login
    - **Popis**: Umožňuje používateľovi prihlásiť sa do systému.
    - **Vstup**:

```
{
  "username": "string",
  "password": "string"
}
```

    - **Výstup**:

```
{
  "access_token": "string",
  "refresh_token": "string"
}
```

    - **Oprávnenie**: Verejný endpoint (bez autentifikácie).
  - POST /auth/register
    - **Popis**: Registrácia nového používateľa.
    - **Vstup**:

```
{
  "username": "string",
  "email": "string",
  "password": "string"
}
```

    - **Výstup**:

```
{
  "message": "User registered successfully."
}
```

        * **Oprávnenie**: Verejný endpoint.
    - POST /auth/refresh
        * **Popis**: Obnovenie access tokenu pomocou refresh tokenu.
        * **Vstup**:

```
{
  "refresh_token": "string"
}
```

        * **Výstup**:

```
{
  "access_token": "string"
}
```

        * **Oprávnenie**: Vyžaduje platný refresh token.
- Produkty
    - GET /products
        * **Popis**: Získanie zoznamu všetkých produktov.
        * **Výstup**:

```
[
  {
    "id": "int",
    "name": "string",
    "description": "string",
    "price": "float",
    "stock": "int",
    "category_id": "int"
  }
]
```

        * **Oprávnenie**: Verejný endpoint.
    - POST /products
        * **Popis**: Pridanie nového produktu.
        * **Vstup**:

```
{
  "name": "string",
```

```
  "description": "string",
  "price": "float",
  "stock": "int",
  "category_id": "int"
}
```


    * **Výstup**:


```
{
  "message": "Product created successfully."
}
```


    * **Oprávnenie**: Vyžaduje oprávnenie administrátora.

- Košík
  - GET /cart
    * **Popis**: Získanie obsahu košíka aktuálneho používateľa.
    * **Výstup**:


```
[
  {
    "product_id": "int",
    "quantity": "int"
  }
]
```


    * **Oprávnenie**: Vyžaduje autentifikáciu.
  - POST /cart
    * **Popis**: Pridanie produktu do košíka.
    * **Vstup**:


```
{
  "product_id": "int",
  "quantity": "int"
}
```


    * **Výstup**:


```
{
  "message": "Product added to cart."
}
```

        * **Oprávnenie**: Vyžaduje autentifikáciu.
    - DELETE /cart/product_id
        * **Popis**: Odstránenie produktu z košíka.
        * **Výstup**:

```
{
  "message": "Product removed from cart."
}
```

        * **Oprávnenie**: Vyžaduje autentifikáciu.

- Objednávky
    - GET /orders
        * **Popis**: Získanie histórie objednávok aktuálneho používateľa.
        * **Výstup**:

```
[
  {
    "id": "int",
    "total_price": "float",
    "status": "string",
    "created_at": "datetime"
  }
]
```

        * **Oprávnenie**: Vyžaduje autentifikáciu.
    - POST /orders
        * **Popis**: Vytvorenie novej objednávky.
        * **Vstup**:

```
{
  "cart": [
    {
      "product_id": "int",
      "quantity": "int"
    }
  ]
}
```

* **Výstup**:

```
{
  "message": "Order created successfully."
}
```

* **Oprávnenie**: Vyžaduje autentifikáciu.

6 PWA

PWA časť projektu je integrovaná vo frontend aplikácii. **Prvky PWA**

- `manifest.json`: Tento súbor definuje základné parametre PWA, ako sú názov aplikácie, ikony, farebná schéma a spôsob spúšťania aplikácie. Obsahuje tieto hlavné parametre:
  - **name**: Názov aplikácie.
  - **short_name**: Skrátený názov aplikácie pre zariadenia.
  - **icons**: Sada ikon v rôznych veľkostiach pre podporu rôznych zariadení.
  - **start_url**: Počiatočná URL aplikácie.
  - **display**: Typ zobrazenia, napr. standalone pre natívny vzhľad.
  - **background_color** a **theme_color**: Farby pre splash obrazovky a UI.
- `service-worker.js`: Implementovaný na správu vyrovnávacej pamäte (caching) a offline prístupu.
  - Caching statických súborov (HTML, CSS, JavaScript) pri prvej návšteve.
  - Network-first stratégia pre dynamické dáta, kde sa najskôr skúsi sieť, a ak nie je dostupná, použijú sa dáta z cache.
- Kompatibilita a inštalácia: Aplikácia je navrhnutá tak, aby fungovala na moderných prehliadačoch podporujúcich PWA. Po pridaní na plochu sa spúšťa ako natívna aplikácia bez prehliadačovej lišty.
- Offline režim: Pomocou service workera aplikácia zabezpečuje, že základná funkcionalita (napr. prezeranie produktov alebo kategórií) je dostupná aj bez internetového pripojenia. Dáta, ktoré sa často menia (napr. košík), môžu byť synchronizované neskôr, keď je internet opäť dostupný.

Pri zostavovaní (build) aplikácie sa PWA funkcionalita pridáva automaticky. Po nasadení na server (napr. cez Nginx alebo Docker kontajner) je PWA dostupná koncovým používateľom. Prehliadače, ktoré podporujú PWA, rozpoznajú manifest a umožnia používateľovi aplikáciu inštalovať.

7 Testovanie

Projekt poskytuje prístup k testovaniu, ktorý zahŕňa unit testy, load testy a integračné testy.

7.1 Unit testy

Unit testy sú zamerané na testovanie jednotlivých funkcií alebo metód v izolácii. Cieľom je overiť, či každá komponenta funguje správne bez ohľadu na ostatné časti aplikácie. Unit testy vo frontendovej časti sú implementované pomocou knižnice **@vue/test-utils**.

Príklad testovaných oblastí v ProductSearch.vue

- **Rendrovanie**: Overenie, či sa vstupné pole pre vyhľadávanie správne vykreslí.
- **Interakcia používateľa**: Simulácia zadávania textu do vyhľadávacieho poľa. Overenie, či sa správne spúšťajú udalosti, ako napríklad vyhľadávanie pri odoslaní formulára.
- **Validácia vstupov**: ontrola, či sú nesprávne hodnoty správne odmietnuté.

```
import { test, expect } from '@playwright/test';

test('login test', async ({ page }) => {
    await page.goto('http://localhost:3000/auth/login');
    await page.fill('#username', 'test');
    await page.fill('#password', 'password');
    await page.click('button[type="submit"]');
    await expect(page).toHaveURL('http://localhost:3000/dashboard');
});
```

Výpis 2: Príklad integračných testov

7.2 Load testy

Load testy simulujú vysokú záťaž na aplikáciu, aby sa overila jej stabilita a výkonnosť pod ťažkým zaťažením. Sú implementované pomocou nástroja Locust a knižnica gevent sa používa na spracovanie asynchrónnych požiadaviek.. Nachádzajú sa v súbore `backend/locustfile.py`. Testy simulujú správanie používateľov, napríklad: prihlásenie, prehliadanie produktov, vytváranie objednávok.

```
from locust import HttpUser, task, between

class WebsiteUser(HttpUser):
    wait_time = between(1, 5)

    @task
    def view_products(self):
        self.client.get("/products")

    @task
    def login(self):
        self.client.post("/auth/login", json={"username": "test", "password": "password"})
```

Výpis 3: Príklad load testov

7.3 Integračné testy

Testy overujú správnu funkčnosť aplikácie ako celku, od frontendového rozhrania po backend a databázu. Sú implementované pomocou nástroja Playwright. Nachádzajú sa v adresári `frontend/e2e`. Testy pokrývajú:

- Interakcie používateľov s rozhraním (napr. prihlásenie, prehliadanie produktov).
- Korektné zobrazenie prvkov na stránke.
- Overenie navigačných tokov (napr. prechod z košíka na dokončenie objednávky).

```
from locust import HttpUser, task, between

class WebsiteUser(HttpUser):
    wait_time = between(1, 5)

    @task
    def view_products(self):
        self.client.get("/products")

    @task
    def login(self):
        self.client.post("/auth/login", json={"username": "test", "password": "password"})
```

Výpis 4: Príklad load testov

7.3.1 Testované oblasti

- **Autentifikácia**: Prihlásenie, registrácia, odhlásenie.

- **Produkty**: Prehliadanie produktov, filtrovanie, vyhľadávanie.
- **Košík**: Pridávanie a odstraňovanie produktov, zobrazenie ceny.
- **Objednávky**: Dokončenie objednávky, sledovanie stavu.
- **Administračné funkcie**: Správa produktov a kategórií.

8 Bezpečnosť

8.1 Autentifikácia a autorizácia

Autentifikácia

- Používa sa JWT (JSON Web Token) na zabezpečenie relácií používateľov.
- Tokeny obsahujú údaje o identite a platnosti relácie.
- Implementácia je v súboroch:
  - `backend/app/config.py`
  - `backend/app/dependencies.py`
  - `backend/app/middlewares/is_admin.py`
  - `backend/app/routers/auth.py`

Autorizácia

- Implementovaná pomocou rolí (napr. administrátor, bežný používateľ).
- Každá API cesta overuje, či používateľ má potrebné oprávnenia.

8.2 Hashovanie hesiel

- Heslá používateľov sú uložené v hashovanom formáte pomocou algoritmu **bcrypt**.
- V súbore `backend/app/routers/auth.py`.
- Hashovanie zahŕňa používanie **salt**, aby sa predišlo útokom bruteforce a rainbow table.

8.3 Bezpečná komunikácia

- Celá aplikácia vyžaduje šifrované pripojenie pomocou protokolu HTTPS.

8.4 Logovanie

- Aplikácia zaznamenáva všetky dôležité udalosti, napr. prihlásenia, zmeny údajov a chyby.
- Implementácia logovania je v `backend/app/migrations/env.py`.
- Logy obsahujú minimálne citlivé údaje a sú šifrované.

8.5 Validácia vstupov

Validácia vstupov je v projekte implementovaná pomocou ORM a knižnice Pydantic.