Technicka Dokumentace - E-Shop

Zakladni informace

Polozka	Hodnota
Nazev projektu	E-Shop
Oznaceni	D1 - Repository Pattern
Autor	Robin Zajicek (zajicek3)
Kontakt	zajicek3@spsejecna.cz
Skola	SPSE Jecna, Praha
Predmet	Databaze
Datum vypracovani	Leden 2026
Typ projektu	Skolni projekt

Obsah

---

1. Popis projektu

1.1 Ucel aplikace

E-Shop je webova aplikace pro spravovani internetoveho obchodu. Umoznuje:

- Prohlizeni a spravovani produktu
- Vytvareni objednavek
- Spravovani uzivatelu a jejich kreditu
- Generovani reportu a statistik

- Import dat z JSON formatu

1.2 Pouzite technologie

Vrstva	Technologie
Frontend	Next.js 16, React
Backend	Python 3.13, Flask
Databaze	Microsoft SQL Server
Komunikace	REST API, JSON

1.3 Hlavni funkce

- CRUD operace nad produkty, kategoriemi, uzivateli
- Vytvareni objednavek (transakce pres vice tabulek)
- Prevod kreditu mezi uzivateli (transakce)
- Agregovany report z vice tabulek
- Import produktu a kategorii z JSON

---

2. Pozadavky a Use Cases

2.1 Funkcni pozadavky (Functional Requirements)

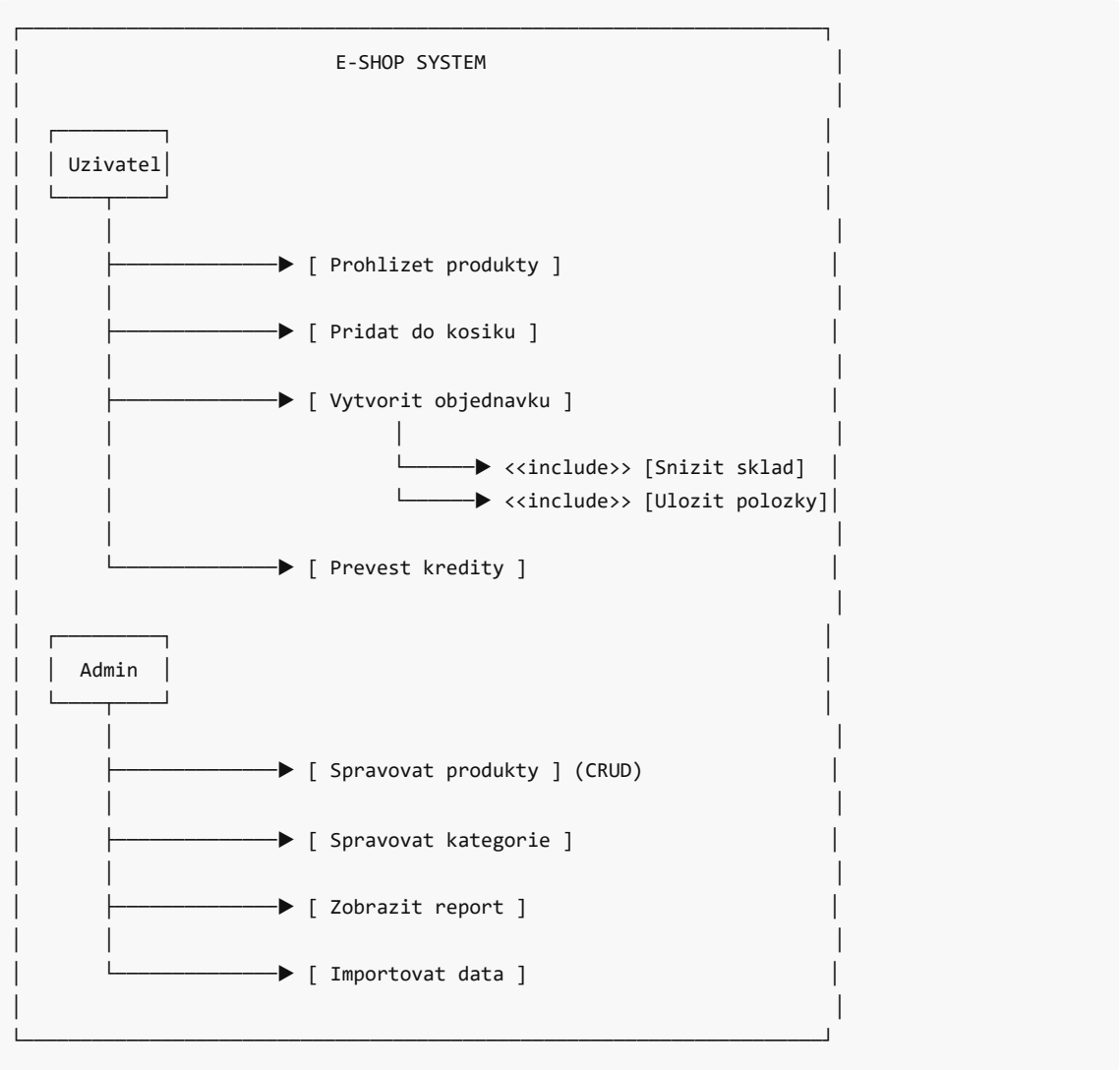
ID	Pozadavek	Priorita
FR01	Uzivatel muze prohlizet produkty	Vysoka
FR02	Uzivatel muze pridat produkt do kosiku	Vysoka
FR03	Uzivatel muze vytvorit objednavku	Vysoka
FR04	Admin muze pridavat/editovat/mazat produkty	Vysoka
FR05	Admin muze pridavat kategorie	Stredni
FR06	Admin muze videt souhrnny report	Stredni
FR07	Admin muze importovat data z JSON	Stredni
FR08	Uzivatel muze prevest kredity jinemu uzivateli	Nizka

2.2 Nefunkcni pozadavky (Non-Functional Requirements)

ID	Pozadavek
NFR01	Aplikace musi pouzivat relacni databazi (SQL Server)
NFR02	Databaze musi obsahovat min. 5 tabulek
NFR03	Databaze musi obsahovat min. 2 views
NFR04	Musi byt implementovan Repository pattern (D1)

NFR05	Konfigurace pomoci .env souboru
NFR06	Osetreni chyb s vhodnymi hlaskami

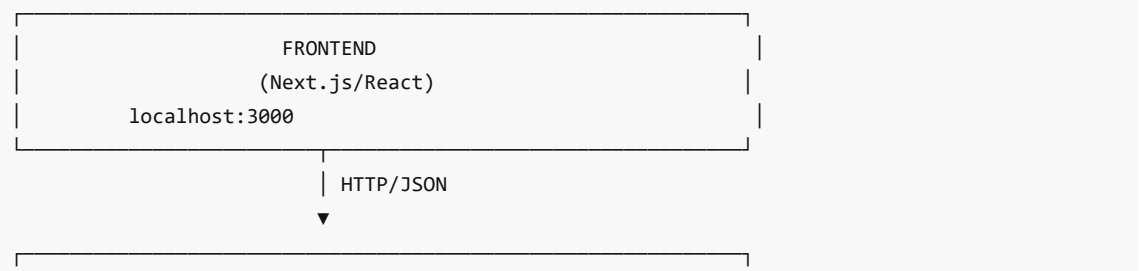
2.3 Use Case Diagram

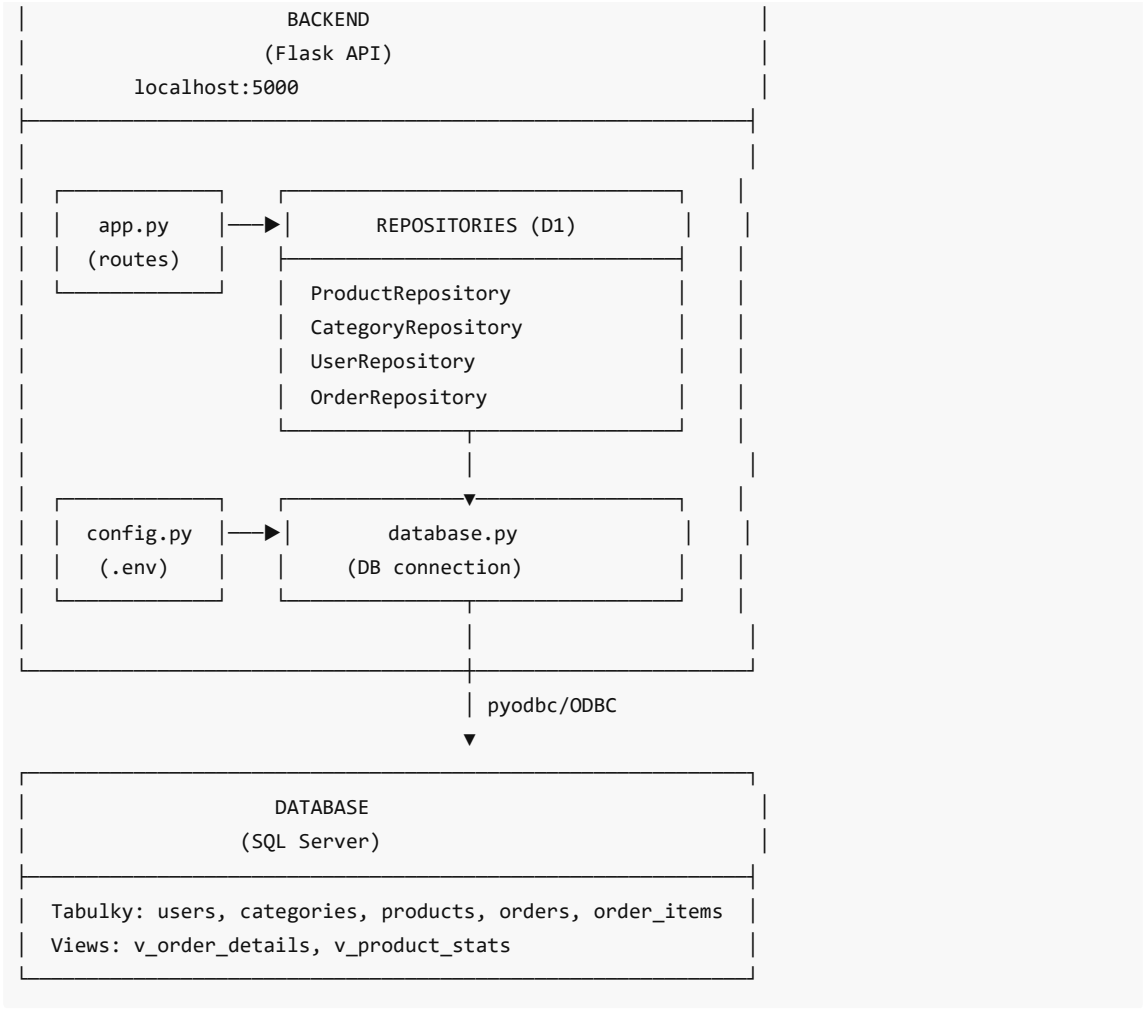


---

3. Architektura systemu

3.1 Vrstvy aplikace





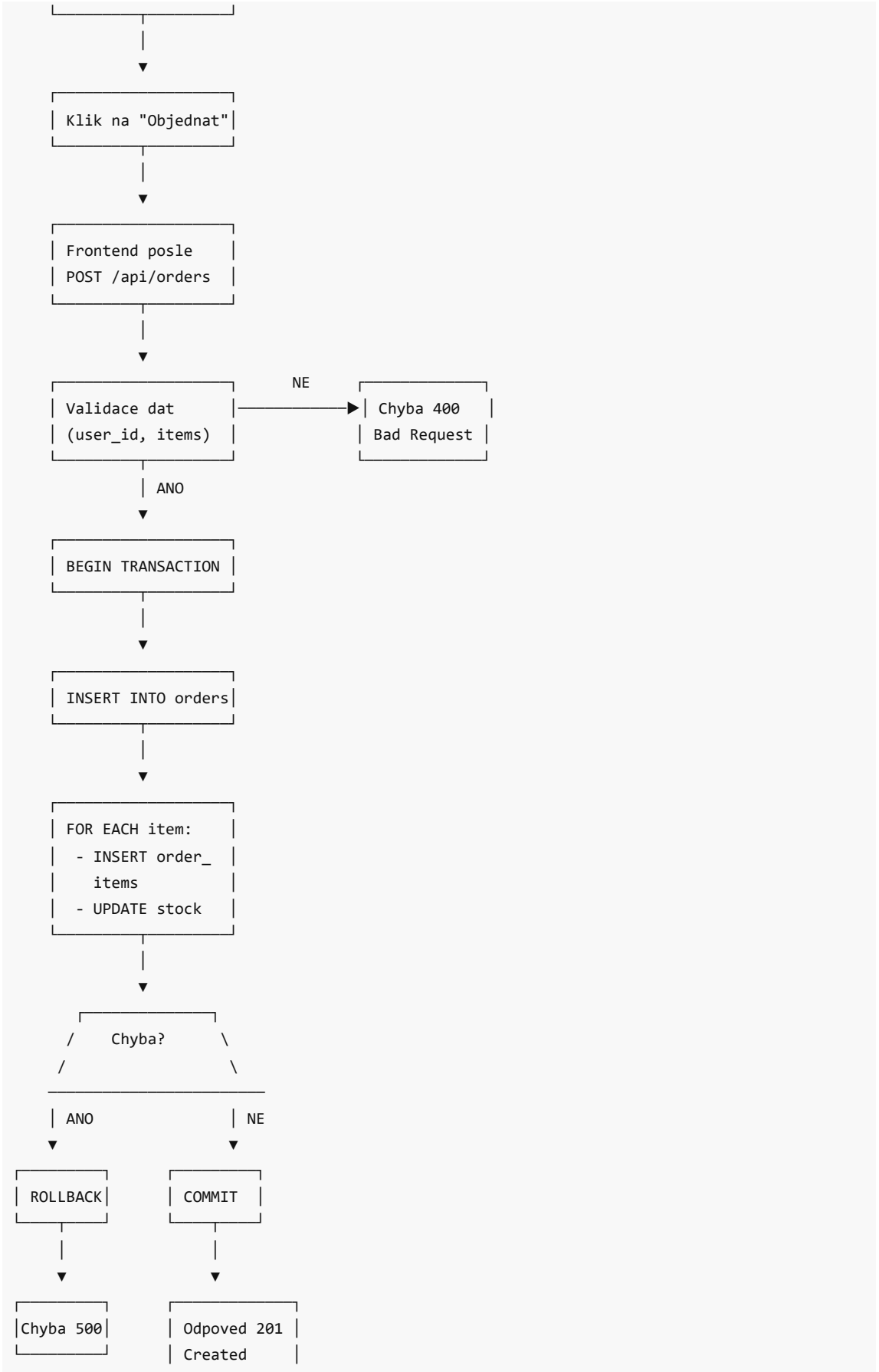
3.2 Tok dat

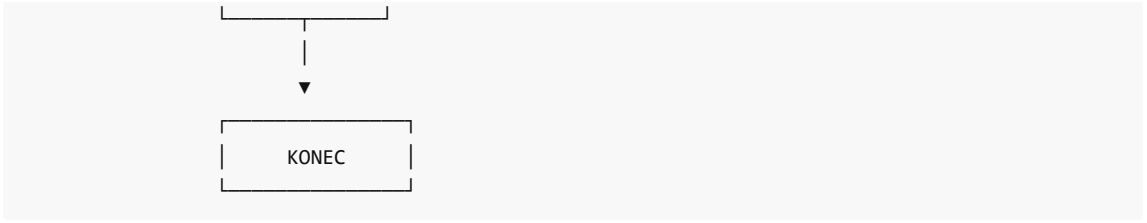
1. Uzivatel interaguje s frontendem (prohlizec)
2. Frontend posila HTTP requesty na backend API
3. Backend zpracuje request pres prislusny endpoint
4. Endpoint vola metody z Repository
5. Repository provadi SQL dotazy pres Database tridu
6. Vysledky se vraci zpet ve formatu JSON

---

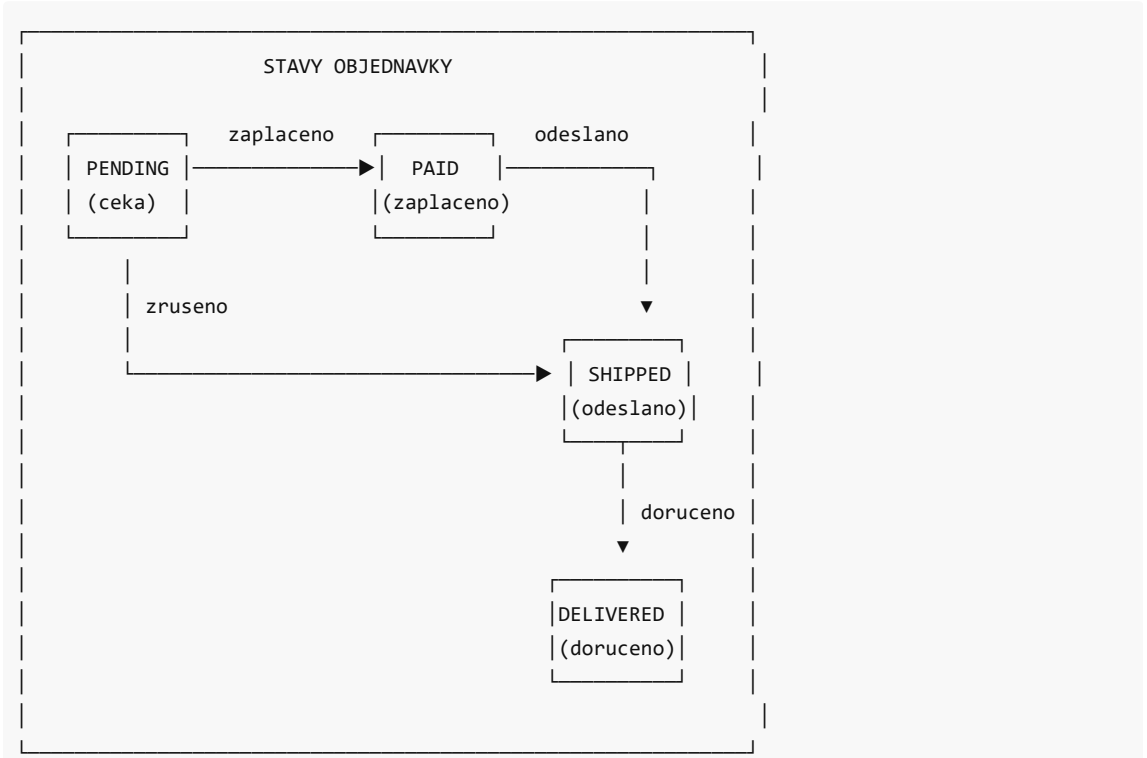
4. Beh aplikace

4.1 Activity Diagram - Vytvoreni objednavky


Klik na "Objednat"
Frontend posle
POST /api/orders
Validace dat
(user_id, items)
NE
Chyba 400
Bad Request
ANO
BEGIN TRANSACTION
INSERT INTO orders
FOR EACH item:
- INSERT order_
items
- UPDATE stock
Chyba?
ANO
NE
ROLLBACK
COMMIT
Chyba 500
Odpoved 201
Created

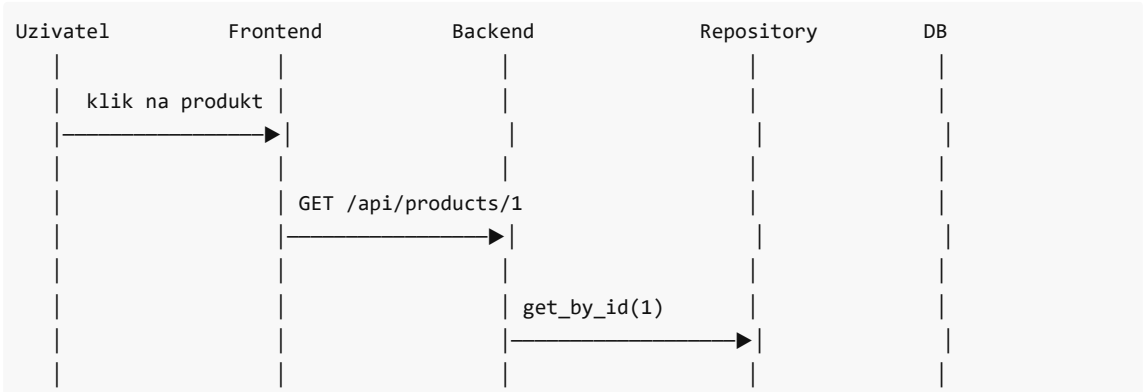


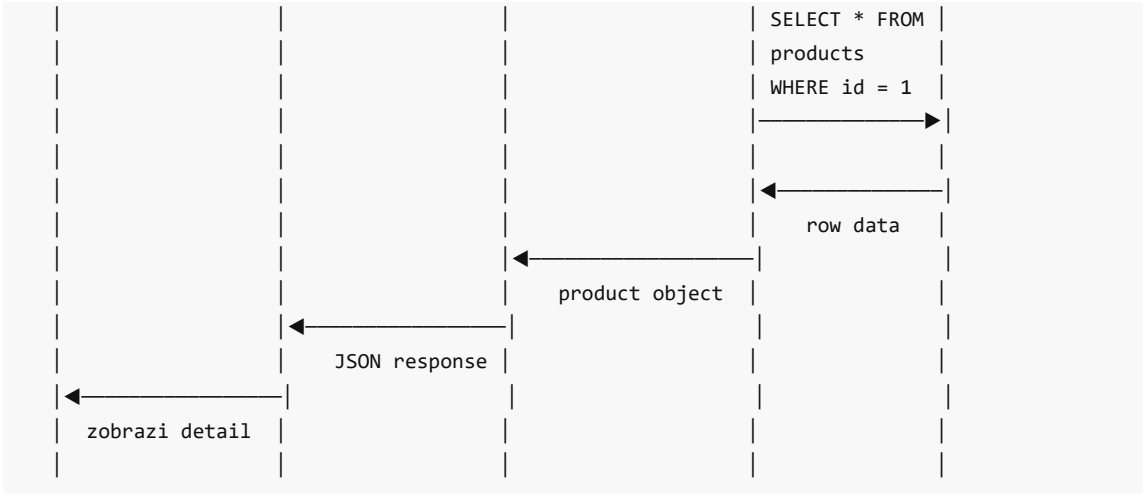
4.2 State Diagram - Stav objednavky




```
Povolene stavy v DB (CHECK constraint):
- 'pending'   - objednavka ceka na zaplaceni
- 'paid'      - zaplaceno
- 'shipped'   - odeslano
- 'delivered' - doruceno
```

4.3 Sequence Diagram - API Request

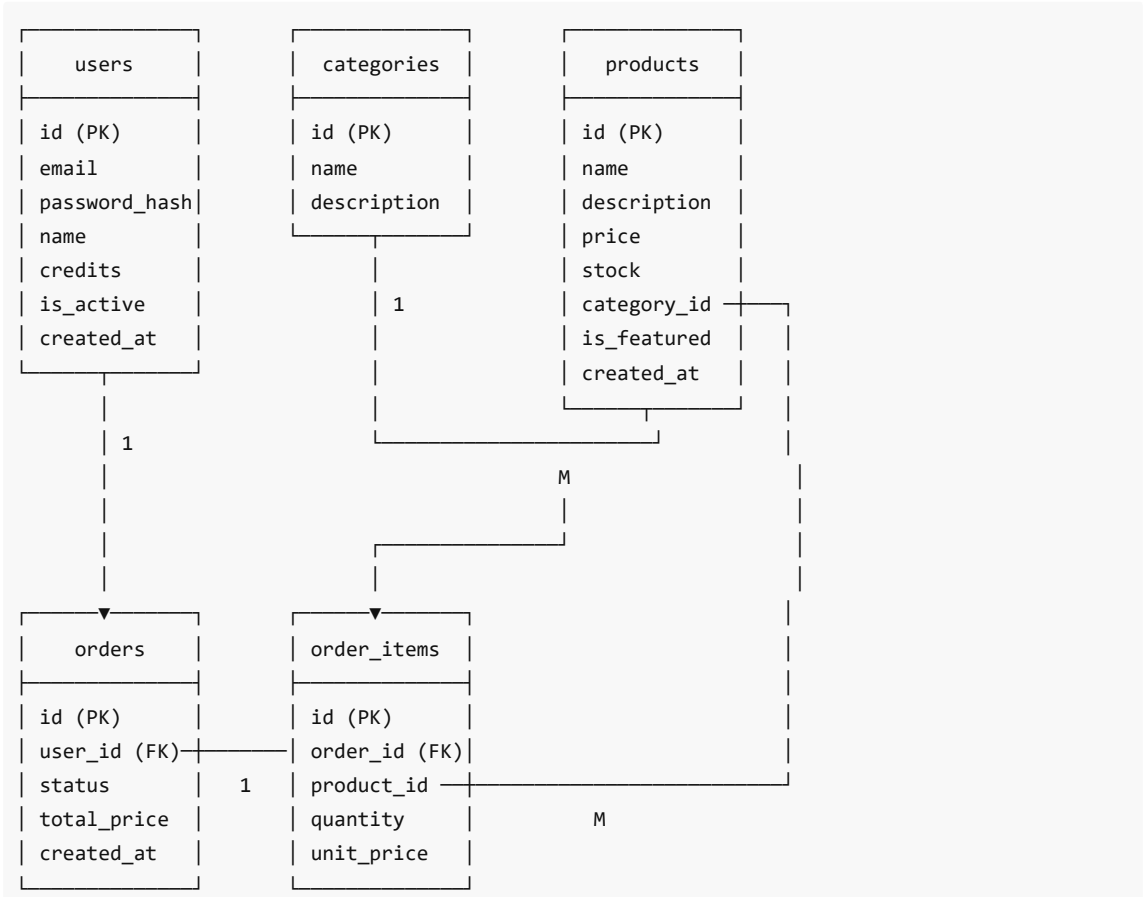




---

5. Databazove schema

5.1 ER Diagram



Vazby:
- users 1:N orders (uzivatel ma vice objednavek)
- categories 1:N products (kategorie obsahuje vice produktu)
- orders 1:N order_items (objednavka ma vice polozek)

```
- products M:N orders (produkt muze byt ve vice objednavkach)
  → reseno pres vazebni tabulku order_items
```

5.2 Popis tabulek

users

Sloupec	Typ	Popis
id	INT IDENTITY	Primarni klic
email	VARCHAR(255) UNIQUE	Email uzivatele
password_hash	VARCHAR(255)	Hashovane heslo
name	VARCHAR(100)	Jmeno uzivatele
credits	DECIMAL(10,2)	Kredit uzivatele
is_active	BIT	Aktivni ucet (0/1)
created_at	DATETIME	Datum vytvoreni

categories

Sloupec	Typ	Popis
id	INT IDENTITY	Primarni klic
name	VARCHAR(100)	Nazev kategorie
description	TEXT	Popis kategorie

products

Sloupec	Typ	Popis
id	INT IDENTITY	Primarni klic
name	VARCHAR(255)	Nazev produktu
description	TEXT	Popis produktu
price	DECIMAL(10,2)	Cena
stock	INT	Pocet na sklade
category_id	INT (FK)	ID kategorie
is_featured	BIT	Doporuceny produkt
created_at	DATETIME	Datum pridani

orders

Sloupec	Typ	Popis

id	INT IDENTITY	Primarni klic
user_id	INT (FK)	ID uzivatele
status	VARCHAR(20) CHECK	Stav objednavky
total_price	DECIMAL(10,2)	Celkova cena
created_at	DATETIME	Datum vytvoreni

Povolene hodnoty status: 'pending', 'paid', 'shipped', 'delivered'

order_items

Sloupec	Typ	Popis
id	INT IDENTITY	Primarni klic
order_id	INT (FK)	ID objednavky
product_id	INT (FK)	ID produktu
quantity	INT	Mnozstvi
unit_price	DECIMAL(10,2)	Cena v dobe nakupu

5.3 Views

v_order_details

Zobrazuje objednavky s detaily polozek, zakaznika a produktu.

```
SELECT o.id, o.status, o.total_price, o.created_at,
       u.name, u.email, p.name, oi.quantity, oi.unit_price
FROM orders o
JOIN users u ON o.user_id = u.id
JOIN order_items oi ON o.id = oi.order_id
JOIN products p ON oi.product_id = p.id
```

v_product_stats

Zobrazuje produkty se statistikami prodeje.

```
SELECT p.id, p.name, c.name, p.price, p.stock,
       SUM(oi.quantity), SUM(oi.quantity * oi.unit_price)
FROM products p
LEFT JOIN categories c ON p.category_id = c.id
LEFT JOIN order_items oi ON p.id = oi.product_id
GROUP BY p.id, p.name, c.name, p.price, p.stock
```

5.4 Datove typy

Pozadavek	Datovy typ	Kde pouzito

Realne cislo	DECIMAL(10,2)	price, credits, total_price
Logicka hodnota	BIT	is_active, is_featured
Vycet (enum)	VARCHAR + CHECK	status
Retezec	VARCHAR	name, email, description
Datum/cas	DATETIME	created_at

---

6. Repository Pattern (D1)

6.1 Co je Repository Pattern

Repository pattern je navrhovy vzor, ktery zapouzdruje pristup k datum. Vytvari abstrakci mezi aplikaci a databazi.

Princip: Kazda tabulka ma svou Repository tridu, ktera obsahuje vsechny metody pro praci s daty.

6.2 Implementace

Hierarchie trid

```
BaseRepository (abstraktni)
    │
    ├── ProductRepository
    ├── CategoryRepository
    ├── UserRepository
    └── OrderRepository
```

BaseRepository

Obsahuje spolecne metody pro vsechny repository:

Metoda	SQL	Popis
get_all()	SELECT * FROM {table}	Vsechny zaznamy
get_by_id(id)	SELECT * WHERE id = ?	Jeden zaznam
delete(id)	DELETE WHERE id = ?	Smazani
count()	SELECT COUNT(*)	Pocet zaznamu

ProductRepository

Dedi z BaseRepository + vlastni metody:

Metoda	Popis
create()	Vytvori novy produkt
update()	Aktualizuje produkt
get_by_category()	Produkty podle kategorie

get_featured()	Doporucene produkty
update_stock()	Zmena stavu skladu

UserRepository

Metoda	Popis
create()	Registrace uzivatele
get_by_email()	Hledani podle emailu
update_credits()	Zmena kreditu
transfer_credits()	**TRANSAKCE** - prevod kreditu

OrderRepository

Metoda	Popis
create_order()	**TRANSAKCE** - vytvoreni objednavky (3 tabulky)
get_order_with_items()	Objednavka s polozkammi
get_user_orders()	Objednavky uzivatele
update_status()	Zmena stavu

6.3 Transakce

transfer_credits()

```
BEGIN TRANSACTION
  1. UPDATE users SET credits = credits - amount WHERE id = from_user
  2. UPDATE users SET credits = credits + amount WHERE id = to_user
  3. IF error THEN ROLLBACK ELSE COMMIT
END TRANSACTION
```

create_order()

```
BEGIN TRANSACTION
  1. INSERT INTO orders (nova objednavka)
  2. FOR EACH item:
     a. INSERT INTO order_items
     b. UPDATE products SET stock = stock - quantity
  3. IF error THEN ROLLBACK ELSE COMMIT
END TRANSACTION
```

6.4 Vyhody Repository Pattern

1. **Oddeleni zodpovednosti** - SQL logika je oddelena od API
2. **Znovupouzitelnost** - metody lze volat z vice mist
3. **Testovatelnost** - lze snadno mockovat

4. **Udrzba** - zmeny SQL na jednom miste

---

7. API Dokumentace

7.1 Zakladni informace

- **Base URL:** `http://localhost:5000`
- **Format:** JSON
- **Autentizace:** Zadna (zjednoduseno pro skolni projekt)

7.2 Endpointy

Health Check

Metoda	URL	Popis
GET	/api/health	Stav API a databaze

Odpoved:

```
{"status": "ok", "database": "connected"}
```

Produkty

Metoda	URL	Popis
GET	/api/products	Seznam vsech produktu
GET	/api/products/{id}	Detail produktu
POST	/api/products	Vytvoreni produktu
PUT	/api/products/{id}	Uprava produktu
DELETE	/api/products/{id}	Smazani produktu
GET	/api/products/featured	Doporucene produkty

POST /api/products - telo:

```
{
  "name": "Nazev",
  "description": "Popis",
  "price": 999.00,
  "stock": 10,
  "category_id": 1,
  "is_featured": false
}
```

Kategorie

Metoda	URL	Popis

GET	/api/categories	Seznam kategorii
POST	/api/categories	Vytvoreni kategorie
GET	/api/categories/with-counts	Kategorie s poctem produktu

Uzivatele

Metoda	URL	Popis
POST	/api/users	Registrace
GET	/api/users/{id}	Detail uzivatele
POST	/api/users/transfer-credits	Prevod kreditu

POST /api/users/transfer-credits:

```
{
  "from_user_id": 1,
  "to_user_id": 2,
  "amount": 100
}
```

Objednavky

Metoda	URL	Popis
POST	/api/orders	Vytvoreni objednavky
GET	/api/orders/{id}	Detail objednavky
PUT	/api/orders/{id}/status	Zmena stavu
GET	/api/users/{id}/orders	Objednavky uzivatele

POST /api/orders:

```
{
  "user_id": 1,
  "items": [
    {"product_id": 1, "quantity": 2, "unit_price": 299.00}
  ]
}
```

Report

Metoda	URL	Popis
GET	/api/report	Agregovana data

Odpoved:

```
{
  "total_revenue": 50000,
  "total_orders": 15,
  "total_users": 10,
  "avg_order_value": 3333.33,
  "top_products": [...],
  "category_sales": [...]
}
```

Import

Metoda	URL	Popis
POST	/api/import/products	Import produktu z JSON
POST	/api/import/categories	Import kategorii z JSON

POST /api/import/products:

```
{
  "products": [
    {"name": "...", "price": 100, "category_id": 1}
  ]
}
```

7.3 HTTP Status kody

Kod	Vyznam
200	Uspech
201	Vytvoreno
400	Chybny pozadavek (validace)
404	Nenalezeno
500	Chyba serveru

---

8. Frontend

8.1 Stranky

URL	Komponenta	Popis
/	page.js	Homepage, doporucene produkty
/products	products/page.js	Seznam produktu
/cart	cart/page.js	Nakupni kosik
/admin	admin/page.js	Admin panel, report

8.2 Komponenty

Komponenta	Popis
Navbar	Navigacni lista
ProductCard	Karta produktu

8.3 API komunikace

Soubor `lib/api.js` obsahuje helper funkce:

- fetchAPI() - GET requesty
- postAPI() - POST requesty
- putAPI() - PUT requesty
- deleteAPI() - DELETE requesty

---

9. Konfigurace

9.1 Konfiguracni soubor

Aplikace pouziva `.env` soubor pro konfiguraci:

Promenna	Popis	Vychozi
DB_SERVER	Adresa SQL serveru	localhost
DB_NAME	Nazev databaze	eshop_db
DB_USER	Uzivatelske jmeno	-
DB_PASSWORD	Heslo	-
DB_DRIVER	ODBC driver	ODBC Driver 17
FLASK_HOST	Adresa Flask	127.0.0.1
FLASK_PORT	Port Flask	5000
FLASK_DEBUG	Debug mod	True

9.2 Nacitani konfigurace

1. `config.py` nacita promenne z `.env` pomoci `python-dotenv`
2. Pokud promenna chybi, pouzije se vychozi hodnota
3. Connection string se sestavuje dynamicky

---

10. Chybove stavy

10.1 HTTP chybove kody

Kod	Nazev	Pricina	Reseni

400	Bad Request	Chybne vstupni data	Zkontrolovat format a povinne polozky
404	Not Found	Zaznam neexistuje	Overit ID zaznamu
500	Server Error	Chyba na serveru/DB	Zkontrolovat logy, pripojeni k DB

10.2 Databazove chyby

Chyba	Pricina	Reseni
18456	Spatne prihlasovaci udaje	Zkontrolovat DB_USER a DB_PASSWORD v .env
Connection refused	Server nebezi	Spustit SQL Server
Database not found	Databaze neexistuje	Vytvorit databazi a spustit SQL skripty
ODBC Driver not found	Chybi driver	Nainstalovat ODBC Driver 17 for SQL Server

10.3 Aplikacni chyby

Chyba	Pricina	Reseni
"Failed to fetch"	Backend nebezi	Spustit `python app.py`
"Nedostatek skladu"	Stock = 0	Doplnit sklad produktu
"Nedostatek kreditu"	credits < amount	Navysit kredit uzivatele
"Email jiz existuje"	Duplicitni email	Pouzit jiny email

---

11. Testovani

11.1 Testovaci scenare

Scenar	Soubor	Popis
1	test_scenario_1_installation.md	Instalace a nastaveni
2	test_scenario_2_functions.md	Funkcni pozadavky
3	test_scenario_3_errors.md	Chybove stavy

11.2 Jak testovat

1. Postupujte dle scenare 1 pro instalaci
2. Spuste backend i frontend
3. Projdete scenar 2 pro overeni funkci
4. Projdete scenar 3 pro overeni chybovych stavu

11.3 Vysledky testovani

Vsechny testy byly provedeny autorem pred odevzdanim:

- ✅ Instalace funguje dle navodu
- ✅ CRUD operace funguji

- ✅ Transakce funguji (rollback i commit)
- ✅ Report vraci spravna data
- ✅ Import z JSON funguje
- ✅ Chybove stavy jsou spravne osetreny

---

12. Zavislosti a knihovny

12.1 Backend (Python)

Knihovna	Verze	Ucel	Licence
Flask	3.0+	Web framework	BSD
pyodbc	5.0+	SQL Server driver	MIT
python-dotenv	1.0+	Nacitani .env	BSD

12.2 Frontend (Node.js)

Knihovna	Verze	Ucel	Licence
Next.js	16+	React framework	MIT
React	19+	UI knihovna	MIT

12.3 Externi sluzby

Sluzba	Popis	Konfigurace
SQL Server	Databazovy server	Promenne v .env
ODBC Driver 17	Driver pro pripojeni	Musi byt nainstalovany

12.4 Instalace zavislosti

```
# Backend
cd src/backend
pip install flask pyodbc python-dotenv

# Frontend
npm install
```

---

13. Import a Export

13.1 Import produktu

Endpoint: `POST /api/import/products`

JSON schema:

```
{
  "products": [
    {
      "name": "string (povinne)",
      "description": "string (volitelne)",
      "price": "number (povinne)",
      "stock": "integer (volitelne, default 0)",
      "category_id": "integer (povinne)",
      "is_featured": "boolean (volitelne, default false)"
    }
  ]
}
```

Priklad:

```
{
  "products": [
    {
      "name": "Novy produkt",
      "price": 599.00,
      "category_id": 1
    }
  ]
}
```

13.2 Import kategorii

Endpoint: `POST /api/import/categories`

JSON schema:

```
{
  "categories": [
    {
      "name": "string (povinne)",
      "description": "string (volitelne)"
    }
  ]
}
```

13.3 Pravidla importu

- Pokud zaznam jiz existuje (podle name), je preskocen nebo aktualizovan
- Povinne polozky musi byt vyplneny
- Neplatne zaznamy jsou ignorovany s chybovou zpravou
- Import probiha v transakci - bud vsechno nebo nic

---

14. Licence a pravni aspekty

14.1 Licence projektu

Tento projekt je **skolni prace** a je urcen pouze pro vzdelavaci ucely.

14.2 Licence pouzitych knihoven

Knihovna	Licence	Odkaz
Flask	BSD-3-Clause	https://flask.palletsprojects.com/
pyodbc	MIT	https://github.com/mkleehammer/pyodbc
python-dotenv	BSD-3-Clause	https://github.com/theskumar/python-dotenv
Next.js	MIT	https://nextjs.org/
React	MIT	https://react.dev/

14.3 Autorskoprávní omezeni

- Kod neni urcen pro komercni vyuziti
- Pri dalsim pouziti je nutne uvest puvod
- Testovaci data jsou fiktivni

---

15. Verze a zname problemy

15.1 Historie verzi

Verze	Datum	Zmeny
1.0.0	Leden 2026	Prvni verze, vsechny pozadavky splneny

15.2 Zname problemy (Known Issues)

ID	Problem	Stav	Workaround
#1	Ceske znaky v JSON odpovedi	Znamo	Backend vraci UTF-8, funguje spravne
#2	Bez autentizace	Zamerne	Skolni projekt - zjednoduseno

15.3 Planovane vylepseni (Future)

- Implementace JWT autentizace
- Skutecne hashovani hesel (bcrypt)
- Pagination pro velke seznamy
- Frontend kosik v local storage

---

16. Splneni pozadavku

16.1 Hlavni ukol D1

Pozadavek	Splneno	Kde

Repository pattern	✅	src/backend/repositories/

16.2 Databazove pozadavky

Pozadavek	Splneno	Kde
5 tabulek	✅	users, categories, products, orders, order_items
2 views	✅	v_order_details, v_product_stats
M:N vazba	✅	order_items
DECIMAL	✅	price, credits
BIT (bool)	✅	is_active, is_featured
CHECK (enum)	✅	status
VARCHAR	✅	name, email
DATETIME	✅	created_at

16.3 Funkcni pozadavky

Pozadavek	Splneno	Kde
CRUD vice tabulek	✅	OrderRepository.create_order()
Transakce	✅	transfer_credits(), create_order()
Report 3+ tabulek	✅	/api/report
Import JSON	✅	/api/import/*
Config soubor	✅	.env + config.py
Error handling	✅	Validace, try/catch

16.4 Dokumentace

Pozadavek	Splneno	Kde
README	✅	README.md
Dokumentace	✅	doc/DOKUMENTACE.md
Test scenar 1	✅	doc/test_scenario_1_installation.md
Test scenar 2	✅	doc/test_scenario_2_functions.md
Test scenar 3	✅	doc/test_scenario_3_errors.md

16.5 Checklist dle Prilohy 1

#	Pozadavek	Splneno

1	Nazev, autor, kontakt, skola, datum	✅
2	Specifikace pozadavku / Use Cases	✅
3	Architektura (diagramy)	✅
4	Beh aplikace (Activity/State diagram)	✅
5	Zavislosti a knihovny	✅
6	Licence a pravni aspekty	✅
7	Konfigurace	✅
8	Instalace (README)	✅
9	Chybove stavy	✅
10	Testovani	✅
11	Verze a known issues	✅
12	ER diagram databaze	✅
13	Schema site	N/A (neni potreba)
14	Konfigurace sluzeb	✅ (SQL Server v .env)
15	Import/export schema	✅
16	Jeden .md soubor	✅

Zaver

Tento projekt demonstruje implementaci Repository patternu (D1) v kontextu e-shopove aplikace. Vsechny pozadavky zadani byly splneny, vcetne vsech bodu z Prilohy 1 - Checklist.

Autor: Robin Zajicek (zajicek3)
Kontakt: zajicek3@spsejecna.cz
Skola: SPSE Jecna, Praha
Datum: Leden 2026