Софийски университет „Св. Климент Охридски“

Факултет по математика и информатика

Курсов проект по

Системи за е-бизнес

зимен семестър 2025/2026

Проект:

Онлайн магазин за спортни стоки

Изготвил:

Владимир Григоров, 9MI8000007

1. Описание на учебния проект

Настоящият курсов проект представлява разработка на уеб базирана система за електронна търговия, ориентирана към продажба на спортна екипировка и аксесоари. Проектът е реализиран като пълнофункционално клиент–сървър приложение, включващо бекенд, фронтенд и релационна база данни.

Основната цел на проекта е да се демонстрира:

- работа с база от данни
- управление на каталог от продукти
- реализация на потребителска кошница
- обработка на поръчки
- автентикация и авторизация
- интеграция с външни системи

2. Архитектура и дизайн на системата

2.1 Общ архитектурен модел

Проектът е реализиран по тристепенна архитектура:

1. Presentation Layer (Frontend)
2. Business Logic Layer (Backend)
3. Data Layer (Database)

Този подход осигурява:

- добра поддръжка

- лесно разширяване
- ясно разделение на отговорностите

2.2 Backend архитектура

Бекендът е реализиран чрез ASP.NET Core Web API, използващ REST архитектурен стил.

Основни технологии:

- ASP.NET Core
- Entity Framework Core (Code First)
- PostgreSQL
- JWT (JSON Web Tokens)
- Stripe API

Основните отговорности на бекенда са:

- обработване на HTTP заявки
- бизнес логика
- валидация
- комуникация с базата данни
- контрол на достъпа (ролите User / Admin)

2.3 Frontend архитектура

Фронтендът е реализиран като SPA (Single Page Application) с:

- HTML
- CSS
- TypeScript
- Vite
- Fetch API

Frontend-ът отговаря за:

- визуализация на данните
- изпращане на заявки към бекенда
- управление на JWT токена
- потребителско взаимодействие

3. Описание на е-бизнеса

Проектът моделира реален онлайн магазин за спортна екипировка, който предлага:

- фитнес оборудване
- кардио уреди
- тренировъчни аксесоари
- продукти за възстановяване

Основни бизнес цели:

- лесна навигация в каталога
- бързо търсене
- персонализирани препоръки
- сигурно плащане
- администраторски контрол

4. Функционалност, обекти и потребителски случаи

4.1 Основни актьори

Актьор	Описание
Гост	Разглежда продукти
Потребител	Купува, прави поръчки
Администратор	Управлява съдържанието

4.2 Потребителски случаи (Use Cases)

За потребител:

- Регистрация
- Вход
- Разглеждане на каталог
- Търсене на продукти
- Попълване на въпросник
- Добавяне в кошница

- Създаване на поръчка
- Плащане
- Преглед на поръчки

За администратор:

- Добавяне на продукти
- Изтриване на продукти
- Добавяне на категории
- Изтриване на категории

5. Описание на базата от данни

Проектът използва релационна база данни PostgreSQL. Моделът е създаден чрез подход Code First.

Основни таблици:

Users

- Id
- Username
- Email
- Password (hash)
- FirstName
- LastName

- IsAdmin

Categories

- Id
- Name

Products

- Id
- Name
- Description
- Price
- CategoryId
- ImageUrl

CartItems

- Id
- UserId
- ProductId
- Quantity

Orders

- Id
- UserId

- CreatedAt

OrderItems

- Id
- OrderId
- ProductId
- Quantity

Tags

- Id
- Name

ProductTags

- ProductId
- TagId

UserProfiles

- Id
- UserId
- Age
- Weight
- Height

UserProfileTags

- UserProfileId
- TagId

6. Реализация на каталог, категории и търсене

Каталогът е структуриран йерархично:

- категории
- продукти

Реализирано е:

- CRUD за категории
- CRUD за продукти
- търсене по име
- администраторски контрол

В базата от данни има:

- 9 категории
- 72 продукта
- минимум 8 продукта във всяка категория

7. Потребителска кошница и поръчки

Кошница:

- отделна таблица
- асоциирана с потребителя
- поддържа редактиране и изтриване

Поръчки:

- създават се от кошницата
- запазват се в БД
- достъпни чрез “My Orders”

Препоръки:

- на база тагове
- броят на съвпаденията определя реда

8. Потребителски сметки и плащания

Автентикация:

- JWT
- защитени API endpoints
- роли

Плащане:

- Stripe Checkout
- Test Mode
- Redirect след плащане

9. Иновативност

- персонализиран въпросник
- таг-базирани препоръки
- реална платежна система
- admin панел

10. Интеграция с външни системи

- Stripe (плащания)
- PostgreSQL

12. Подробна Use Case секция (Сценарии)

UC-01: Регистрация на потребител

Актьор: Гост (не логнат потребител)
Цел: Създаване на нов акаунт.

Предусловия: Няма активен login.
Основен поток:

1. Потребителят отваря страницата за регистрация.

2. Попълва: username, email, password, firstName, lastName.
3. Frontend изпраща `POST /api/auth/register`.
4. Backend валидира дали username/email са свободни.
5. Backend записва потребителя (паролата се хешира).
6. Frontend показва успешно съобщение и пренасочва към login.

Алтернативи:

- Ако email вече съществува → backend връща 409 (Conflict)
- Ако username вече съществува → backend връща 409 (Conflict)

UC-02: Вход (Login)

Актьор: Потребител

Цел: Получаване на JWT token и достъп до защитени ресурси.

Основен поток:

1. Потребителят въвежда username и password.
2. Frontend изпраща `POST /api/auth/login`.
3. Backend валидира паролата.
4. Backend генерира JWT token и връща `{ token, isAdmin }`.
5. Frontend записва token и isAdmin в LocalStorage.
6. Потребителят се пренасочва към Home.

Алтернативи:

- грешна парола или username → 401 Unauthorized

UC-03: Разглеждане на каталог

Актьор: Логнат потребител

Цел: Вижда списък с продукти.

Основен поток:

1. Frontend извиква `GET /api/products`.
2. Backend връща списък с продукти.
3. Frontend визуализира продуктите.

UC-04: Търсене на продукт

Актьор: Логнат потребител

Цел: Търсене по име.

Основен поток:

1. Потребителят пише в search полето.
2. Frontend извиква `GET /api/products/search?name=...`
3. Backend връща филтриран списък.
4. Frontend показва резултатите.

UC-05: Управление на кошница

Актьор: Логнат потребител

Цел: Добавяне/промяна на количества/премахване.

Основен поток:

1. Потребителят натиска Add to Cart.
2. Frontend прави `POST /api/cart/add`.
3. Backend добавя/увеличава quantity.
4. Потребителят отваря cart page.
5. `GET /api/cart` връща текущите items.
6. Потребителят променя quantity (+/-).
7. Frontend праща `PUT /api/cart/update` или `DELETE /api/cart/remove`.
8. Backend обновява кошницата.

UC-06: Checkout (симулирано плащане)

Актьор: Логнат потребител

Цел: Симулирано плащане чрез Stripe и приключване на поръчка.

Основен поток:

1. Потребителят натиска “Proceed to Checkout”.
2. Frontend вика `POST /api/checkout/create-session`.
3. Backend създава Stripe Checkout Session и връща `url`.
4. Frontend redirect-ва към Stripe Checkout.
5. Потребителят използва test карта (4242...).
6. Stripe връща към `checkout-success` страница.
7. Frontend извиква `POST /api/orders`.
8. Backend създава order и изчиства cart.
9. Потребителят вижда потвърждение.

UC-07: Преглед на поръчки

Актьор: Логнат потребител

Цел: Вижда списък със свои поръчки.

Основен поток:

1. Frontend извиква `GET /api/orders`.
2. Backend връща поръчките на текущия user.
3. Frontend ги показва.

UC-08: Попълване на въпросник (персонализация)

Актьор: Логнат потребител

Цел: Запис на профил + тагове за препоръки.

Основен поток:

1. Потребителят отваря Questionnaire.
2. Отговаря на въпроси.
3. Frontend изпраща резултата към backend.
4. Backend записва/обновява `UserProfile`.
5. Backend записва `UserProfileTags` според отговорите.

UC-09: Препоръчани продукти

Актьор: Логнат потребител

Цел: Получава персонализирани продукти.

Основен поток:

1. Frontend извиква `GET /api/products/recommended`.
2. Backend намира тагове на потребителя.
3. Backend намира продуктите с най-много съвпадащи тагове.
4. Връща топ N резултати.
5. Frontend ги показва на Home.

UC-10: Admin – управление на продукти (CRUD)

Актьор: Администратор

Цел: Добавяне/триене/редакция.

Предусловия: User е логнат и `IsAdmin=true`.

Основен поток:

1. Администраторът отваря Admin Panel.
2. Вижда списък с продукти (`GET /api/products`).
3. Създава нов продукт (`POST /api/products`).
4. Трие продукт (`DELETE /api/products/{id}`).

Алтернативи:

- ако не е администратор → 403 Forbid

UC-11: Admin – upload на снимка

Актьор: Администратор

Цел: Качване на product image.

Основен поток:

1. Администраторът избира файл от компютъра.
2. Frontend изпраща `POST /api/uploads/product-image` (multipart/form-data).
3. Backend записва файла в `wwwroot/images/products/`.
4. Връща `imageUrl`.
5. Създаденият продукт използва този `imageUrl`.

Заключение

Проектът SportsGearStore представлява пълноценна уеб система, която покрива всички изисквания за курсов проект, демонстрирайки реални бизнес процеси, модерни технологии и добра архитектура.