# Почивни бази ТУ-СофияТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ

## ФАКУЛТЕТ ПО КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ

**ДИПЛОМНА РАБОТА**

Тема: **„Онлайн магазин за дрехи с отделена административна система за управление"**

**Дипломен ръководител:**

проф. д-р инж. Румен Иванов Трифонов

**Изготвил:**

Християн Юриев Панов

**фак. номер: 121221139**

**ФКСТ, спец. КСИ, напр. ИТ**

**45 група**

София, 2025 г.

**СЪДЪРЖАНИЕ**

I. **Увод**  
1.1. Цели и обхват на разработката

II. **Анализ на изискванията**  
2.1. Функционални изисквания (каталог, продукт детайл, размери, количка, wishlist, поръчки, плащане със Stripe, ревюта, GET THE LOOK, профил, Forgot password, нотификации)  
2.2. Нефункционални изисквания (сигурност, производителност, UX)  
2.3. Потребителски роли (гост, регистриран клиент, администратор)

III. **Използвани технологии и инструменти**  
3.1. Backend: Python 3, Django (MVT архитектура)  
3.2. Frontend: HTML5, CSS3, JavaScript (Fetch API / AJAX), модали и динамични UI елементи  
3.3. База данни: **PostgreSQL**.env  
3.4. Плащания: Stripe API (ключове и webhook-и).env  
3.5. Имейл услуги: Gmail SMTP (за Forgot password).env  
3.6. Dev инструменти: VS Code, Git

IV. **Архитектура на системата и реализация**  
4.1. Общ преглед на архитектурата (MVT, приложения: users, main, products, cart, wishlist, orders, payment, reviews, outfits)  
4.2. Основни модели и връзки (ERD за нашата БД)  
4.3. Функционални модули:  
  4.3.1. Начална страница (мъжки/женски секции, случайни продукти, анимации)  
  4.3.2. Регистрация и вход (Login/Signup модали), Forgot password с имейл  
  4.3.3. Каталог и продукт детайл (избор на размер с radio бутони)  
  4.3.4. Количка (AJAX добавяне/премахване, динамично обновяване)  
  4.3.5. Wishlist (с избор на размер, модален прозорец)  
  4.3.6. GET THE LOOK (динамичен модал с AJAX зареждане и добавяне в количка)  
  4.3.7. Поръчка и Checkout (subtotal, shipping, discount, total)  
  4.3.8. Плащане със Stripe (интеграция и webhook-и)  
  4.3.9. Ревюта (ограничение до едно на потребител, визуализация, админ преглед)  
  4.3.10. Нотификации (toasts)  
  4.3.11. Админ панел (преглед на поръчки, ревюта)

V. **Тестване и оценка**  
5.1. Методология и тестова среда  
5.2. Функционални тестове на модулите  
5.3. UX и производителност  
5.4. Мерки за сигурност (CSRF, валидация, пароли)

VI. **Ръководство за потребителя**  
6.1. Регистрация и вход  
6.2. Търсене и преглед на продукти  
6.3. Добавяне в количка и wishlist  
6.4. Завършване на поръчка и плащане

VII. **Ръководство за администратор**  
7.1. Преглед на поръчки  
7.2. Управление на продукти  
7.3. Преглед на ревюта

VIII. **Инсталация и конфигурация**  
8.1. Локална инсталация (venv, зависимости, manage.pymanage, миграции)  
8.2. Конфигурация на .env (PostgreSQL, Stripe, Gmail).env  
8.3. Статични файлове и media  
8.4. Препоръки за production

IX. **Заключение и бъдещо развитие**  
9.1. Обобщение  
9.2. Насоки за разширяване

X. **Използвана литература и източници**

# 1.УВОД

С развитието на технологиите и навлизането на електронната търговия в ежедневието, потребителите все повече предпочитат да извършват своите покупки онлайн. Те очакват бърз достъп до желаните продукти, удобна навигация, сигурни разплащания и интуитивен потребителски интерфейс. Конкуренцията в този сектор е висока, а успешните онлайн магазини трябва да предлагат не само основни функции за пазаруване, но и допълнителни удобства, които подобряват цялостното потребителско изживяване.

Настоящият проект представлява уеб приложение за електронна търговия, разработено с помощта на Django и интегриращо ключови функционалности, които улесняват процеса на пазаруване. Системата обединява традиционните елементи на онлайн магазин с допълнителни модули като списък с желания, модални прозорци за бърз достъп до количката и акаунта, динамична секция „GET THE LOOK“ за пакетно добавяне на артикули, както и поддръжка на сигурни плащания чрез Stripe.

Разработката е мотивирана от стремежа да се създаде модерна, гъвкава и сигурна платформа, която предлага на потребителите удобство и бързина при пазаруване, а на администраторите – лесно управление на съдържанието и поръчките.

### 1.1. Цели и обхват на разработката

Основната цел на проекта е изграждането на модерен, динамичен и функционален онлайн магазин, който да предоставя на потребителите лесен и интуитивен начин за откриване, избиране и закупуване на продукти, без необходимост от постоянно презареждане на страниците.

За постигането на тази цел системата включва:

* **Интуитивен потребителски интерфейс** – ясно структурирано съдържание, бърз достъп до основните секции и минимален брой стъпки за завършване на поръчката.
* **Динамично зареждане на съдържанието** – чрез интеграция на AJAX и модални прозорци за интерактивно взаимодействие, което позволява обновяване на данните без презареждане на страницата.
* **Пазарска количка** – с поддръжка на избор на размер и управление на продуктите чрез модален прозорец.
* **Списък с желания** – възможност за запазване на избрани артикули с конкретен размер за бъдеща покупка.
* **„GET THE LOOK“ модул** – за пакетно добавяне на няколко свързани артикула към количката с предварително избрани размери.
* **Система за поръчки и плащания** – интеграция със Stripe API за обработка на плащания с дебитни и кредитни карти.
* **Потребителски акаунт** – регистрация, вход, възстановяване на забравена парола и преглед на историята на поръчките.
* **Секция за ревюта** – позволяваща на потребителите да оставят оценки и мнения за закупени продукти, с администраторски контрол.
* **Административен панел** – за управление на продукти, поръчки и ревюта.
* **PostgreSQL база данни** – за сигурно и ефективно съхранение на информацията за продукти, потребители, поръчки и плащания.

# 2.АНАЛИЗ НА ИЗИСКВАНИЯТА

### В тази глава се извършва анализ на функционалните и нефункционалните изисквания към системата, както и определяне на основните потребителски роли и сценариите за тяхната употреба. Целта е да се дефинира ясно какво трябва да може да изпълнява уеб приложението, какви са неговите технически и потребителски ограничения и какви потребителски взаимодействия трябва да бъдат поддържани. Анализът е основа за проектирането и реализацията на системата, като гарантира, че разработката ще отговаря на предварително поставените цели и ще предостави необходимата функционалност на целевите потребители.

### 2.1. Функционални изисквания

Системата трябва да реализира следните функционалности, разделени по основни модули:

 **Начална страница**

* Динамично показване на продукти от мъжка и женска категория.
* Промоционални секции с визуални елементи.

 **Каталог с продукти**

* Преглед на списък с продукти, филтрирани по категория и пол.
* Визуализация на основно изображение и допълнителни снимки.

 **Детайлна страница за продукт**

* Избор на размер чрез радио бутони.
* Добавяне в пазарска количка или списък с желания без презареждане на страницата.

 **Пазарска количка**

* Преглед и управление на избраните артикули чрез модален прозорец.
* Промяна на размер (при нужда) и премахване на артикули.
* Изчисляване на междинна сума, доставка, отстъпка и крайна цена.

 **Списък с желания**

* Запазване на избрани продукти с конкретен размер за бъдеща покупка.
* Преглед на съдържанието в модален прозорец.

 **„GET THE LOOK“ модул**

* Преглед на готови комбинации от продукти (outfits).
* Избор на размер за всеки продукт и пакетно добавяне в количката.

 **Поръчки и плащане**

* Попълване на форма с адрес и данни за доставка.
* Обработка на плащания чрез Stripe API.
* Потвърждение и запис на поръчката в базата данни.

 **Потребителски акаунт**

* Регистрация и вход чрез модални прозорци.
* Възстановяване на забравена парола с имейл.
* Преглед на историята на поръчките.

 **Ревюта на продукти**

* Потребителите могат да оставят едно ревю за даден продукт.
* Показване на всички ревюта в детайлната страница на продукта.

 **Администраторски панел**

* Управление на продукти, поръчки, потребители и ревюта.

## 2.2. Нефункционални изисквания

 **Сигурност** – защита на личните данни, пароли и транзакции; използване на CSRF защита и валидация на формите.

 **Производителност** – оптимизация на заявките към базата данни и минимизиране на презарежданията на страниците.

 **Мащабируемост** – възможност за добавяне на нови категории, методи на плащане и функционалности.

 **Удобство за потребителя (UX)** – интуитивна навигация, ясни бутони и бързо зареждане на съдържанието.

 **Поддръжка на различни устройства** – адаптивен дизайн за десктоп и мобилен телефон.

#### 2.3. Потребителски роли и сценарии на употреба

 **Гост**

* Разглежда продукти и категории.
* Може да добавя продукти в количката и списъка с желания (но запазването е временно).
* Може да се регистрира, за да запази историята на пазаруването.

 **Регистриран потребител**

* Всички права на гост.
* Запазване на количката и списъка с желания в профила.
* Извършване на поръчка и плащане.
* Оставяне на ревю за закупен продукт.
* Преглед на историята на поръчките.

 **Администратор**

* Пълен достъп до административния панел.
* Добавяне, редактиране и премахване на продукти.
* Преглед и управление на поръчки.
* Одобряване или премахване на ревюта.

# 3. ИЗПОЛЗВАНИ ТЕХНОЛОГИИ И ИНСТРУМЕНТИ

В тази глава се описват технологиите, програмните езици, библиотеките и инструментите, използвани при разработката на проекта. Подборът им е направен с оглед на нуждите на системата, като са взети предвид фактори като надеждност, производителност, лесна интеграция и възможности за бъдещо разширяване. Описани са както сървърната и клиентската част на приложението, така и базата данни, платежната система и инструментите за разработка и конфигурация.

**3.1. Backend: Python/Django (MVT архитектура)**

Сървърната част на системата е реализирана с помощта на **Python 3** и уеб фреймуърка **Django**, който следва архитектурния модел **MVT (Model–View–Template)**. Тази архитектура позволява ясна логическа структура на кода и разделяне на отговорностите между различните компоненти на системата.

**Python** е език за програмиране от високо ниво, известен със своята четимост, простота и богата екосистема от библиотеки. Благодарение на своята универсалност и изчистен синтаксис, Python е един от най-популярните езици за изграждане на уеб приложения, научни изчисления, изкуствен интелект и автоматизация. Неговата кросплатформеност и широката общност от разработчици го правят особено подходящ за бърза и стабилна разработка на софтуерни системи.

**Django** е един от водещите уеб фреймуърци за Python и е предпочитан за създаване на мащабируеми и сигурни уеб приложения. Основните му предимства включват:

* **„Batteries-included“ философия** – Django предоставя готови модули за работа с база данни (ORM), удостоверяване на потребители, формуляри, сесии, сигурност и административен панел. Това ускорява значително разработката.
* **Сигурност** – вградените защити срещу често срещани уязвимости (CSRF, SQL Injection, XSS) гарантират надеждност на системата.
* **Мащабируемост** – архитектурата позволява лесно надграждане и интегриране на нови модули.
* **Гъвкавост** – може да се използва както за малки приложения, така и за големи корпоративни системи.

В нашия проект Django е използван в комбинация с **PostgreSQL** като основна база данни и реализира следните основни слоеве на MVT архитектурата:

* **Model** – дефинира структурата на данните и връзките между тях чрез Django ORM. В проекта са реализирани модели за потребители, продукти, размери, изображения, пазарска количка, списък с желания, поръчки, плащания, ревюта и облекла („GET THE LOOK“). Това позволява бързо и сигурно взаимодействие с базата без нужда от писане на SQL заявки.
* **View** – обработва входящите заявки от клиента, изпълнява необходимата бизнес логика и подготвя данните за визуализация. В проекта са използвани както стандартни Django view функции, така и AJAX (Fetch API) за връщане на JSON отговори и динамично обновяване на съдържанието без презареждане на страниците.
* **Template** – осигурява визуалното представяне на данните чрез HTML, CSS и JavaScript. В системата са използвани шаблони с наследяване, което позволява поддържане на унифициран дизайн и лесно обновяване на интерфейса.

Освен това Django предоставя:

* **Административен панел**, който улеснява управлението на продукти, поръчки и потребители.
* **Система за удостоверяване и авторизация**, използвана за регистрация, вход и управление на потребителски профили.
* **Валидация на формуляри** и автоматично генериране на грешки при неправилно въведени данни.
* **CSRF защита и сигурно управление на сесии**, които са интегрирани в системата и гарантират безопасността на потребителските данни.

Благодарение на комбинацията от Python и Django, сървърната част на системата е едновременно **стабилна, сигурна и лесна за поддръжка**, което позволява бъдещо разширяване и интегриране на нови функционалности.

## 3.2. Frontend: HTML5/CSS3, JavaScript (Fetch API / AJAX), модали и динамични UI елементи

Клиентската част на системата е реализирана чрез трите основни уеб технологии – **HTML5**, **CSS3** и **JavaScript**, които съвместно изграждат визуалната и интерактивната част на онлайн магазина.

**HTML5**

HTML (HyperText Markup Language) е основният език за структуриране на съдържанието в уеб приложенията. Последната му версия – HTML5 – предлага редица подобрения, включително семантични тагове (<header>, <section>, <article>, <footer>), вградена поддръжка за мултимедия (<audio>, <video>) и по-добра интеграция с JavaScript API-та.  
В проекта HTML5 се използва за изграждане на:

* структурата на уеб страниците – хедър, меню, продуктови списъци, модали;
* формите за регистрация, вход, поръчка и оставяне на ревю;
* динамични секции като **GET THE LOOK** модала и checkout страницата.  
  Чрез семантичните елементи кодът е по-четим и по-лесен за поддръжка, а търсачките получават по-добър контекст за съдържанието.

**CSS3**

CSS (Cascading Style Sheets) е технологията, която определя визуалното оформление на уеб страниците. Версия 3 (CSS3) въвежда редица нови възможности – **Flexbox**, **Grid Layout**, анимации, преходи и медийни заявки за адаптивен дизайн.  
В рамките на проекта CSS3 е използван за:

* изграждане на адаптивна структура (responsive design), която гарантира правилно визуализиране както на настолни компютри, така и на мобилни устройства;
* прилагане на **плавни анимации и преходи** при отваряне и затваряне на модални прозорци;
* дефиниране на единна цветова схема и типография, която подсилва брандинга на сайта;
* изграждане на прецизни мрежови подредби за продуктови каталози чрез **CSS Grid** и **Flexbox**;
* стилизиране на toast-нотификациите, така че да се появяват ненатрапчиво и да изчезват автоматично.  
  Така CSS3 осигурява модерен, изчистен и консистентен дизайн на целия интерфейс.

**JavaScript**

JavaScript е основният език за програмиране от страна на клиента, който добавя интерактивност и динамично поведение към уеб страниците. За разлика от HTML и CSS, които дефинират структурата и визуалния облик, JavaScript позволява **асинхронна комуникация със сървъра**, промяна на съдържанието без презареждане и реализиране на сложна логика в браузъра.  
В системата JavaScript е използван за:

* управление на **модални прозорци** (login, регистрация, количка, списък с желания, GET THE LOOK);
* динамично добавяне и премахване на продукти в количката и списъка с желания чрез **AJAX (Fetch API)**;
* валидиране на данни при регистрация, вход и checkout, като се предотвратява изпращане на некоректна информация към сървъра;
* реализиране на **toast-нотификации**, които информират потребителя за извършени действия;
* логика за избиране на размери (вкл. помощната функция **Find My Size**).

JavaScript е използван в комбинация с Django backend-а чрез AJAX заявки, което гарантира **динамично обновяване на съдържанието без презареждане на страниците**. Това е ключов елемент от целта на проекта – изграждане на модерен, бърз и интерактивен онлайн магазин.

**3.3. База данни: PostgreSQL**

За управлението и съхранението на данните в системата е използвана релационната база данни **PostgreSQL**. Тя е една от най-популярните и мощни системи за управление на бази данни с отворен код и се отличава със своята надеждност, гъвкавост и богата функционалност.

**Характеристики на PostgreSQL**

PostgreSQL е релационна СУБД, която поддържа **стандарта SQL**, но разширява възможностите му чрез:

* **Сложни релации и връзки** – поддържа релационни връзки „един към много“, „много към много“ и вложени зависимости;
* **ACID съвместимост** – гарантира коректност и надеждност на транзакциите (Atomicity, Consistency, Isolation, Durability);
* **Разширяемост** – позволява дефиниране на собствени типове данни, функции и индекси;
* **Сигурност** – включва механизми за удостоверяване на потребители, контрол на достъпа и криптиране на връзката чрез SSL;
* **Мащабируемост и производителност** – оптимизирана е за работа с големи обеми от данни и множество едновременни заявки.

**Използване в проекта**

В рамките на системата PostgreSQL е интегрирана с Django чрез **Django ORM (Object-Relational Mapping)**, което позволява работа с данните чрез Python обекти вместо директни SQL заявки. Това улеснява поддръжката на кода и осигурява по-голяма сигурност срещу SQL Injection атаки.

В базата данни са реализирани следните основни модели:

* **Потребители** – информация за клиентите и администраторите, използвана за регистрация, вход и управление на профили.
* **Продукти и категории** – дефинират каталога на онлайн магазина, като включват информация за име, описание, цена, изображения и налични размери.
* **Количка (Cart)** – съхранява продуктите, добавени от потребителя преди финализиране на поръчка.
* **Списък с желания (Wishlist)** – позволява запазване на продукти за бъдещи покупки.
* **Поръчки и плащания** – съдържат информация за направените поръчки, тяхното съдържание, статус и съответните транзакции чрез Stripe.
* **Ревюта** – отразяват мненията и оценките на потребителите за закупените продукти.
* **Аутфити (GET THE LOOK)** – комбинирани продуктови предложения, съхранявани като отделни обекти с връзки към включените артикули.

**Предимства от използването на PostgreSQL в проекта**

* **Надеждност** – гарантира стабилност на данните дори при неочаквани сривове.
* **Гъвкавост** – лесно добавяне на нови модели и връзки между тях при бъдещо разширяване на системата.
* **Сигурност** – защитен достъп до базата и възможност за криптиране на комуникацията.
* **Интеграция с Django** – ORM слойът на Django позволява автоматична миграция на базата при промяна на моделите, което улеснява поддръжката.

**3.4. Плащания: Stripe API**

Системата поддържа онлайн плащания чрез интеграция с Stripe API – една от най-надеждните и широко използвани платформи за обработка на плащания. Изборът на Stripe е мотивиран от високото ниво на сигурност, лесната интеграция с Django и поддръжката на множество методи за разплащане.

Реализацията включва:

* Използване на Stripe Secret Key и Webhook Secret, съхранявани в .env файла за защита на чувствителните данни.
* Създаване на плащане от страна на клиента с въвеждане на данни за картата.
* Обработка на успешни и неуспешни транзакции.
* Потвърждаване и запис на плащането в базата данни при получаване на webhook събитие от Stripe.

По този начин системата гарантира сигурно извършване на финансовите операции и автоматично актуализиране на статуса на поръчките след плащане.

## 3.5. Имейл услуги: Gmail SMTP

Системата използва Gmail SMTP сървър за изпращане на автоматизирани имейли към потребителите. Основното му приложение в проекта е при функцията „Възстановяване на забравена парола“, която изисква изпращане на имейл с връзка за промяна на паролата.

Реализацията включва:

* Конфигуриране на SMTP параметрите (потребител, парола, хост, порт, протокол за сигурност) в .env файла, за да се защитят чувствителните данни.
* Използване на Django вградената система за изпращане на имейли, което улеснява интеграцията с Gmail.
* Автоматично генериране на имейл съобщения при заявка за възстановяване на парола.
* Гарантиране на надеждно доставяне чрез защитена връзка (TLS/SSL).

Тази интеграция осигурява бърза и сигурна комуникация между системата и потребителите, без нужда от външни платени имейл услуги.

**3.6. Dev инструменти: VS Code, Git**

Разработката и поддръжката на проекта се осъществяват с помощта на Visual Studio Code (VS Code) и Git. Комбинацията от двата инструмента осигурява бърза ежедневна работа, контрол на версиите и проследимост на всички промени по кода.

**Visual Studio Code (VS Code)**

VS Code е лек, но мощен редактор, който предоставя удобна среда за работа с Python, HTML, CSS и JavaScript. В проекта са използвани:

* Разширение „Python“ – подсветка на синтаксиса, IntelliSense, отладка (debug), стартиране на Django сървър през интегрирания терминал.
* Разширение „Django“ – открояване на шаблоните и помощ при работа с views.py, urls.py и templates/.
* Вграден Git панел – преглед на промените (diff), staging на файлове и commit-и директно от VS Code.

**Практики в редактора:**

* Интегриран терминал за изпълнение на чести команди:  
  python manage.py runserver, python manage.py makemigrations, python manage.py migrate, зареждане на фикстури, създаване на суперпотребител и др.
* Форматиране и linting – поддържане на четим стил на Python/JS кода и уеднаквена структура на шаблоните.
* Работни папки – ясна структура на Django приложенията (users/, main/, products/, cart/, wishlist/, orders/, payment/, reviews/, outfits/) и бърза навигация между модели, изгледи и шаблони.

#### ****Git (контрол на версиите)****

Git – система за контрол на версиите, която позволява проследяване на всички промени по проекта, работа по различни функционалности в отделни клонове и безопасно връщане към предишни версии при нужда. Използвана е и връзка с отдалечено хранилище за архивиране и споделяне на кода.

**Ползи от Git в проекта:**

* **Проследимост** – всяка промяна може да бъде върната или сравнена с предишно състояние.
* **Безопасни експерименти** – разработка в клонове без риск за стабилната версия.
* **Архивиране и споделяне** – връзка с отдалечено хранилище (например GitHub/GitLab/Bitbucket) за бекъп и сътрудничество.

#### ****Обобщение:****

VS Code предоставя бърза и удобна среда за писане, отладка и стартиране на Django приложението, а Git гарантира контрол на версиите и сигурна колаборация. Заедно те осигуряват **ефективен, надежден и проследим** процес на разработка на целия проект.

# 4. СРАВНИТЕЛЕН АНАЛИЗ НА СЪЩЕСТУВАЩИ РЕШЕНИЯ

## 4.1. Увод

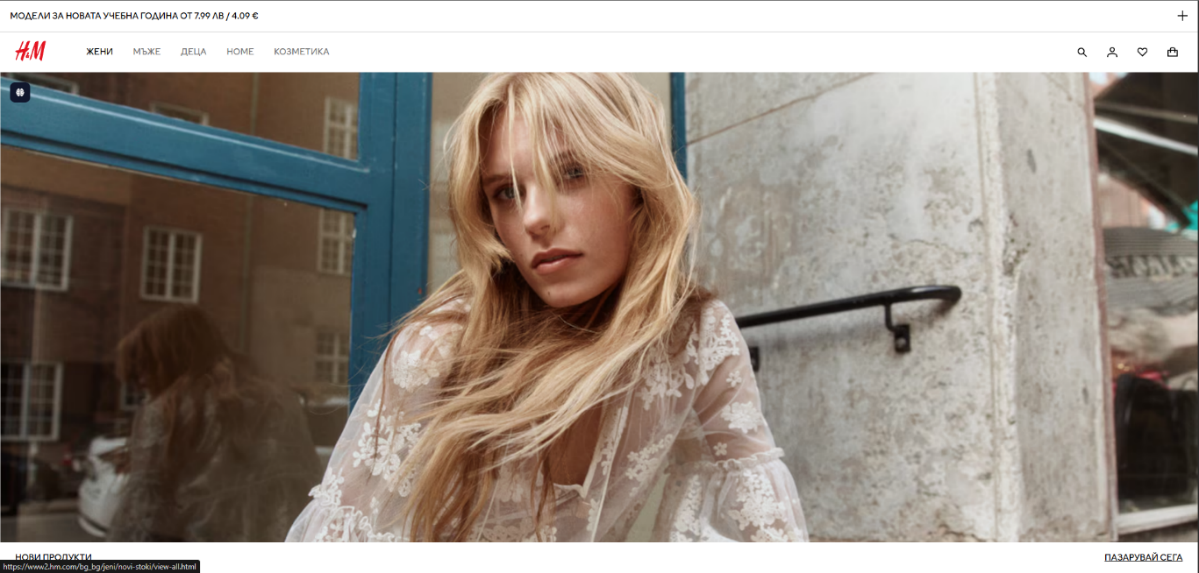
В тази глава се извършва сравнение между реализирания в проекта онлайн магазин и няколко популярни платформи за електронна търговия в сферата на модата. Целта е да се анализират прилики и разлики по отношение на функционалности, потребителско изживяване, технологии и подходи за представяне на продуктите. Това позволява да се оцени на какво ниво е реализираният проект спрямо вече утвърдените решения и да се откроят неговите уникални предимства.

**4.2. Избрани за сравнение платформи**

За целите на сравнителния анализ бяха избрани четири международно утвърдени онлайн магазина, които предлагат сходен тип продукти и са представителни за различни подходи при изграждане на e-commerce решения.

**Zara**  
Zara е един от най-популярните модни брандове в световен мащаб и поддържа добре развита онлайн платформа. Сайтът се отличава с минималистичен дизайн и силен визуален акцент върху продуктовите изображения, които са представени в голям формат и често съпроводени от художествени фотосесии. Навигацията е изчистена и лесна за използване, като целта е потребителят да се фокусира върху самите продукти. Zara използва динамични анимации и плавни преходи, които засилват усещането за премиум изживяване. Онлайн магазинът предлага и добре структурирани категории за мъжка, женска и детска мода, както и специални колекции.

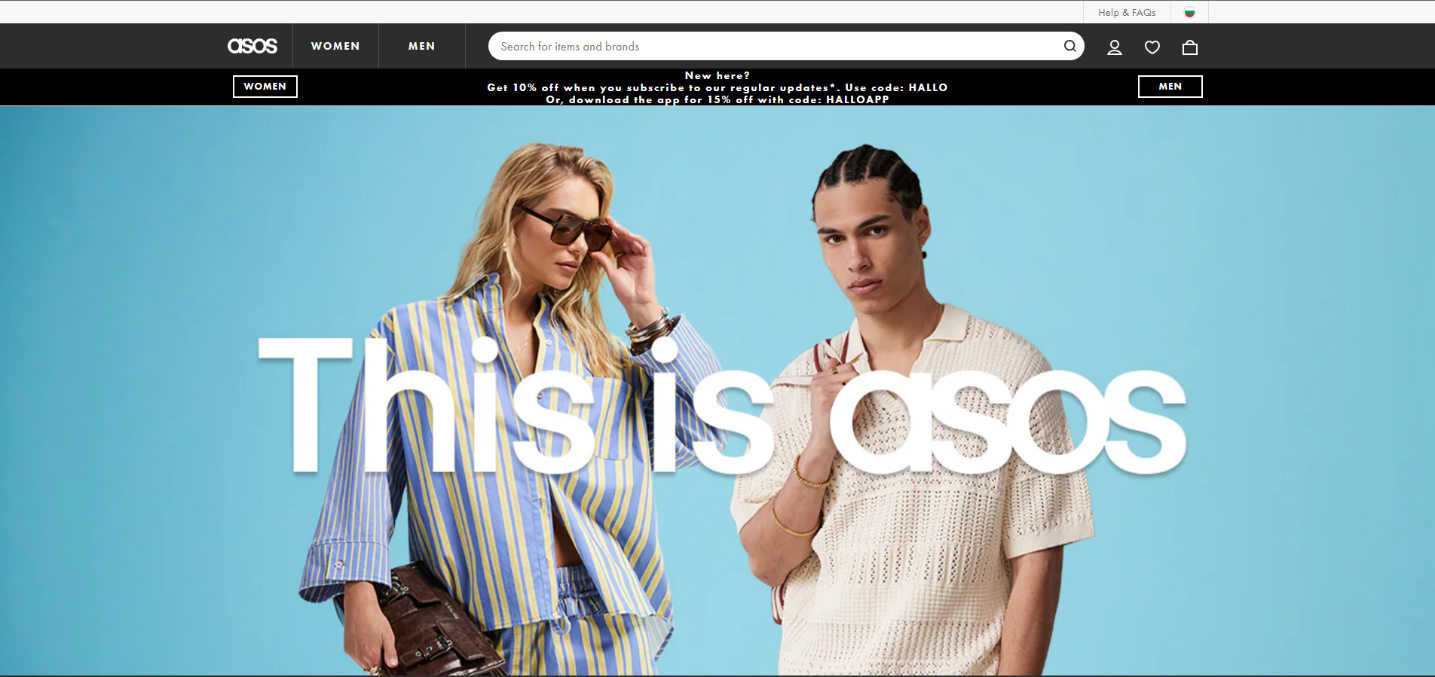
Фигура 1: Сайтът на Zara

**H&M**  
H&M разполага с онлайн магазин, който е насочен към широката аудитория и предлага богат избор от продукти във всички ценови сегменти. Платформата е изградена с акцент върху функционалността – налични са разнообразни филтри (по размер, цвят, стил, материя, цена), както и мощна търсачка, улесняваща откриването на продукти. H&M акцентира и върху устойчивата мода, като предлага специални продуктови линии и секции, свързани с еко инициативи. Дизайнът е опростен и лесен за навигация, а интеграцията на промоционални кампании и сезонни намаления е ясно видима още от началната страница.

Фигура 2: Сайтът на H&M

**Pull&Bear**  
Pull&Bear е моден бранд от групата Inditex, насочен към младежката аудитория. Онлайн магазинът има свеж и динамичен дизайн, който използва по-ярки цветове, графични елементи и неформално оформление. Платформата поставя акцент върху визуалното представяне на продуктите, като често включва lookbook секции и готови стилови комбинации. Навигацията е интуитивна и съобразена с навиците на младите потребители, които използват предимно мобилни устройства. Сайтът е интегриран с различни социални мрежи и често се обновява със сезонни колекции, промоции и мултимедийно съдържание.

Фигура 3: Сайтът на Pull&Bear

**ASOS**  
ASOS е една от най-големите международни платформи за онлайн търговия на дрехи и аксесоари. Тя се отличава с изключително богат продуктов каталог, който включва както собствените брандове на ASOS, така и продукти на външни марки. Системата за търсене и филтриране е сред най-развитите – позволява избор по десетки критерии и персонализирани препоръки на база предишни търсения и покупки. ASOS предлага и функционалности като визуално търсене (качване на снимка, за да се намери подобен продукт) и виртуална пробна за избор на размер. Интерфейсът е модерен, бърз и силно адаптиран за мобилни устройства, а платформата е интегрирана с международни системи за доставка и разнообразни методи за плащане.

Фигура 4: Сайтът на ASOS

## 4.3. Сравнение по функционалности

След извършения анализ се установява, че всички разгледани онлайн магазини – **Zara**, **H&M**, **Pull&Bear** и **ASOS** – предлагат базовите функционалности за електронна търговия като продуктов каталог, детайлна страница за продукт с избор на размер и пазарска количка. Те също поддържат динамично зареждане на съдържанието без презареждане на страницата, което осигурява по-бързо и плавно потребителско изживяване.

Съществени разлики се откриват при допълнителните модули. Разработеният от нас проект се отличава с **„Списък с желания“**, който позволява запазване на продукти с конкретен размер – функционалност, която липсва в Zara, H&M и Pull&Bear, а в ASOS е реализирана без избор на размер. Друг уникален елемент е **модулът „GET THE LOOK“**, който позволява пакетно добавяне на свързани продукти в количката, като подобна функция не е налична в останалите анализирани магазини.

По отношение на системата за ревюта, тя присъства при нашето решение, H&M и ASOS, но отсъства при Zara и Pull&Bear. Онлайн плащанията чрез **Stripe API** са налични единствено в нашия проект – другите магазини използват собствени или външни платежни решения. Всички разгледани платформи, включително нашата, предлагат възможност за възстановяване на забравена парола.

Административният панел за управление на продукти, поръчки и ревюта е реализиран чрез Django Admin и е достъпен само за потребители с администраторски права, по подобие на вътрешните системи на Zara, H&M, Pull&Bear и ASOS.

**Таблица 1: Табличен вид на 4.3 Сравнение по функционалности**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Функционалност | Наш проект | Zara | H&M | Pull&Bear | ASOS |
| Каталог с продукти | ✔ | ✔ | ✔ | ✔ | ✔ |
| Детайлна страница за продукт с размери | ✔ | ✔ | ✔ | ✔ | ✔ |
| Пазарска количка | ✔ | ✔ | ✔ | ✔ | ✔ |
| Списък с желания (с избор на размер) | ✔ | ✖ | ✖ | ✖ | ✔ |
| Модул „GET THE LOOK“ | ✔ | ✖ | ✖ | ✖ | ✖ |
| Динамично добавяне без презареждане | ✔ | ✔ | ✔ | ✔ | ✔ |
| Система за ревюта | ✔ | ✖ | ✔ | ✖ | ✔ |
| Онлайн плащания със Stripe | ✔ | ✖ | ✖ | ✖ | ✖\* |
| Възстановяване на забравена парола | ✔ | ✔ | ✔ | ✔ | ✔ |
| Админ панел за управление | ✔\*\* | ✔\*\* | ✔\*\* | ✔\*\* | ✔\*\* |

**\*** **ASOS предлага онлайн плащания, но чрез собствена интеграция, а не със Stripe.  
\*\* Наличен само за администратори (недостъпен за обикновени потребители).**

## 4.4. Анализ на предимствата на нашето решение

В резултат на сравнителния анализ с избраните за разглеждане онлайн магазини – Zara, H&M, Pull&Bear и ASOS – могат да бъдат откроени няколко ключови предимства на реализирания от нас проект:

**4.4.1. Списък с желания с избор на размер**

* Докато повечето разгледани платформи или не предлагат списък с желания, или той не поддържа избор на размер, нашата реализация позволява запазването на продукти с конкретна размерна опция. Това улеснява бъдещата покупка и елиминира риска потребителят да забрави кой размер е избрал.

**4.4.2. Модул „GET THE LOOK“**

* Уникална функционалност, позволяваща на потребителя да добави няколко свързани артикула в количката едновременно, като за всеки от тях може да избере размер. Това съкращава времето за пазаруване и стимулира продажбите на цели облекла.

**4.4.3. Динамично добавяне на продукти без презареждане на страницата**

* Използването на AJAX и модални прозорци прави процеса на пазаруване по-бърз и удобен, като премахва необходимостта от постоянно зареждане на нови страници.

**4.4.4. Интеграция със Stripe API**

* Гарантира сигурно и надеждно онлайн плащане, без необходимост от прехвърляне на потребителя към външна платежна страница. Това е предимство пред конкурентите, които използват външни решения или ограничени методи на плащане.

**4.4.5. Възможност за оставяне на ревюта с администраторски контрол**

* Потребителите могат да оставят само по едно ревю за продукт, което се одобрява или модерира от администратора. Това гарантира качеството и достоверността на отзивите.

**4.4.6. Интуитивен интерфейс и модулна структура**

* Всички основни действия – добавяне в количка, списък с желания, преглед на облекла и поръчка – могат да се извършват директно от модални прозорци, което ускорява процеса и намалява броя на стъпките до покупка.

# 5. АРХИТЕКТУРА НА СИСТЕМАТА И РЕАЛИЗАЦИЯ

## 5.1. Общ преглед на архитектурата

Реализираната система е изградена по архитектурния модел **MVT (Model–View–Template)**, който е вграден в уеб фреймуърка **Django**. Този модел позволява ясно разделение на отговорностите и улеснява поддръжката и разширяването на проекта.

**Model** – описва структурата на данните, връзките между тях и начините за достъп до тях чрез ORM (Object-Relational Mapping). В проекта са реализирани модели за потребители, продукти, категории, размери, количка, списък с желания, поръчки, плащания, ревюта и облекла („GET THE LOOK“).

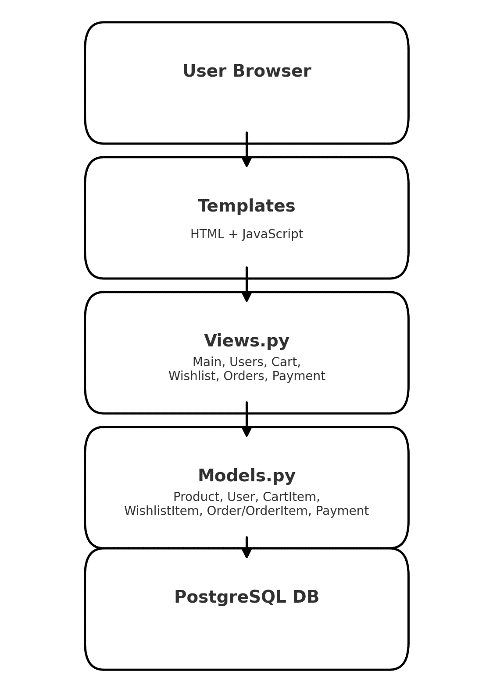
**View** – обработва заявките, получени от клиента, изпълнява необходимата бизнес логика и връща отговор към потребителя. В нашия проект са реализирани както стандартни Django view функции, така и AJAX endpoints за асинхронно обновяване на съдържанието без презареждане на страницата.

**Template** – представя данните във визуална форма чрез HTML, CSS и JavaScript. Използва се система за шаблони с наследяване, която осигурява единен дизайн на всички страници и улеснява поддръжката на интерфейса.

Проектът е организиран в **модулна структура** от отделни Django приложения, всяко от които изпълнява специфична роля:

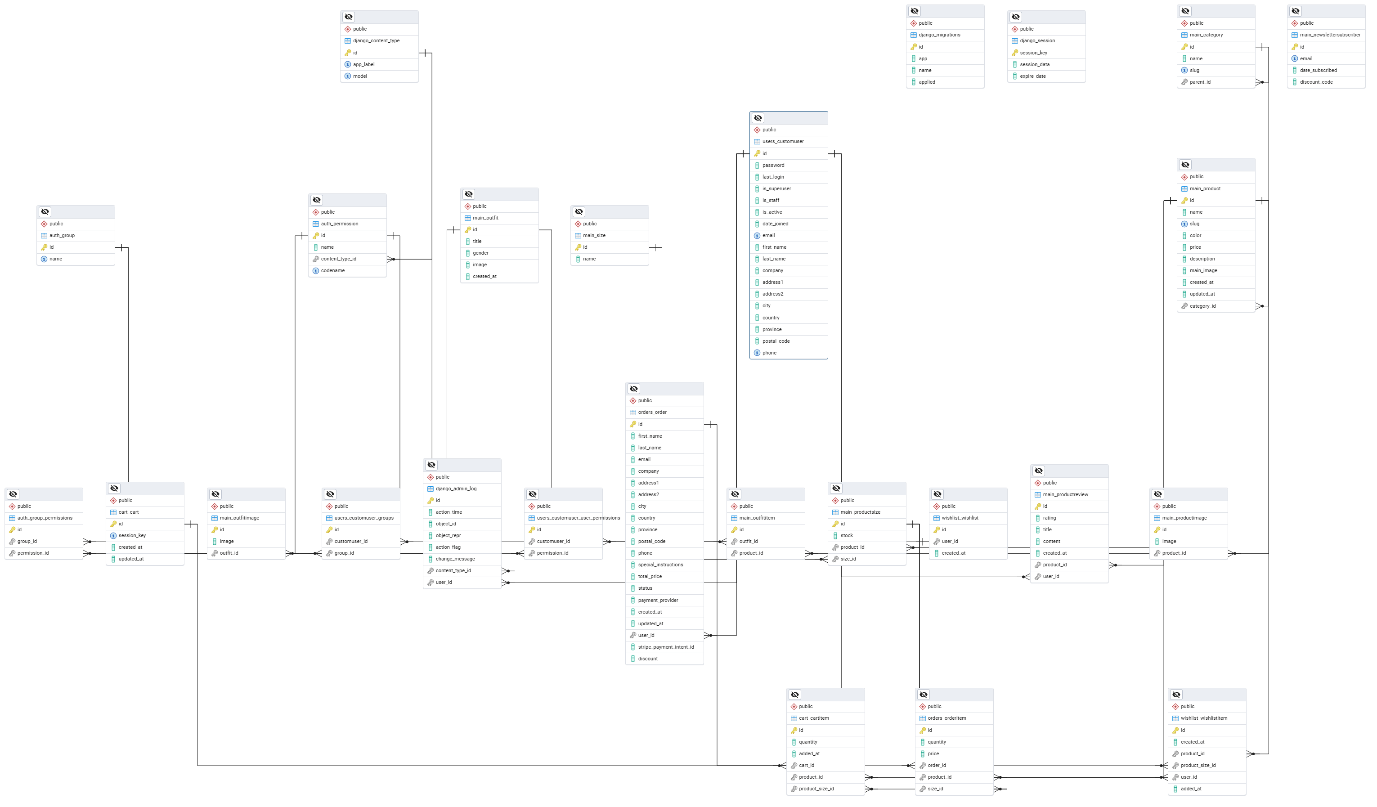
* users – управление на регистрация, вход, профили и възстановяване на парола.
* main – начална страница и общи изгледи.
* products – каталог, детайлна страница за продукт, размери и изображения.
* cart – пазарска количка и динамично управление на продуктите.
* wishlist – списък с желания.
* orders – обработка и запис на поръчки.
* payment – интеграция със Stripe за онлайн плащания.
* reviews – система за ревюта.
* outfits – модул „GET THE LOOK“.

Взаимодействието между компонентите е реализирано така, че **данните се обработват в Model слоя, логиката – във View слоя, а визуализацията – в Template слоя**, което осигурява яснота, гъвкавост и лесна поддръжка.



Фигура 2: Архитектурен модел MVT и модули на системата.

### 5.2. Основни модели и техните връзки (ERD)

Базата данни на системата е проектирана така, че да поддържа всички ключови процеси в един модерен онлайн магазин за дрехи – от управлението на потребителите и продуктовия каталог до пазарската количка, списъка с желания, поръчките, плащанията и ревютата. Архитектурата е реализирана в **PostgreSQL** и е интегрирана с Django ORM, което позволява удобна работа с данните чрез Python, без необходимост от директни SQL заявки.

Фигура 3: ERD диаграма на базата данни (основни таблици и връзки).

**5.2.1. Основни модели**

Моделът **User** е от ключово значение за системата. Той съхранява данните за всички регистрирани потребители – име, имейл, парола, дата на регистрация и други профилни настройки. Чрез него се изграждат връзките към количката, списъка с желания и историята на направените поръчки. Това позволява на всеки клиент да има персонализирано изживяване при пазаруване, запазвайки своите предпочитания и предишни покупки.

Структурата на каталога е реализирана чрез моделите **Category** и **Product**. **Category** групира продуктите по основни и подкатегории – например мъжко облекло, дамско облекло, аксесоари – което улеснява навигацията и филтрирането. Моделът **Product** съдържа детайлна информация за всеки артикул – име, описание, цена, дата на добавяне, както и връзки към категория, налични размери и продуктови изображения. Поддръжката на множество изображения дава възможност за по-добро визуално представяне на продуктите, а връзката с **Size** осигурява избора на конкретен размер при покупка.

Количката (**Cart**) и списъкът с желания (**Wishlist**) са индивидуални за всеки потребител. Те работят съвместно с междинните модели **CartItem** и **WishlistItem**, които съхраняват конкретните избрани артикули. Всеки ред в количката съдържа информация за продукта, избрания размер и количеството, докато елементите в списъка с желания съдържат продукт и избран размер, но без количество. Тази структура позволява прецизно управление на избраните продукти и улеснява потребителя при бъдещи покупки.

Процесът на поръчка е реализиран чрез моделите **Order** и **OrderItem**. **Order** съхранява общата информация за всяка поръчка – уникален номер, потребител, дата на създаване, адрес за доставка и статус. **OrderItem** съдържа детайлите за отделните артикули в поръчката – продукт, размер, единична цена и количество. Тази двустепенна структура позволява лесно управление и проследяване на поръчките, както от страна на потребителя, така и от страна на администратора.

Към всяка поръчка е свързан запис за плащане (**Payment**), който съдържа информация за транзакцията, извършена чрез **Stripe API** – идентификатор на плащането, статус и дата на извършване. Тази интеграция гарантира сигурността и надеждността на онлайн разплащанията.

За събиране на обратна връзка от клиентите е реализиран моделът **Review**. Той позволява на потребителите да оставят едно ревю за закупен продукт, което включва оценка и коментар. Това спомага за изграждане на доверие и информираност сред останалите клиенти.

Функционалността „**GET THE LOOK**“ е реализирана чрез моделите **Outfit** и **OutfitItem**. **Outfit** представлява готова комбинация от няколко продукта, а **OutfitItem** свързва конкретни артикули с дадено облекло. Потребителят може с един клик да добави цялата комбинация в количката, като за всеки продукт предварително се избира размер. Това значително ускорява процеса на пазаруване и насърчава покупките на цели комплекти.

### 5.2.2. Връзки между моделите

Връзките между моделите в системата са изградени така, че да осигуряват пълна функционалност на онлайн магазина и в същото време да поддържат логическа консистентност на данните. Те отразяват реалните зависимости между основните обекти – потребители, продукти, количка, списък с желания, поръчки и плащания.

Всеки **потребител** (**User**) може да има само **една активна количка** (**Cart**) и един **списък с желания** (**Wishlist**), които са персонализирани и съхраняват индивидуалните му избори. Връзката между тях е едно-към-едно (1–1). Когато потребителят добави продукт в количката или списъка с желания, се създава запис в съответната междинна таблица – **CartItem** или **WishlistItem**. Тези елементи съдържат референция към конкретен продукт (**Product**) и избран размер (**Size**). При количката се записва и желаното количество, докато при списъка с желания количеството липсва.

**Категориите** (**Category**) групират продуктите и имат връзка едно-към-много (1–N) с модела **Product**, което означава, че всяка категория може да съдържа множество продукти, но един продукт принадлежи само към една категория. Всеки продукт може да има няколко изображения (**ProductImage**) и множество налични размери (**Size**).

**Поръчките** (**Order**) са свързани с потребителя във връзка едно-към-много (1–N), тъй като един потребител може да направи множество поръчки във времето. Всяка поръчка съдържа един или повече артикули (**OrderItem**), които описват конкретен продукт с избран размер, единична цена и количество. Поръчката има връзка едно-към-едно (1–1) с плащането (**Payment**), което съдържа информация за извършената транзакция чрез Stripe API.

**Ревютата** (**Review**) също са свързани с потребителите и продуктите чрез връзки едно-към-много (1–N), но със специално ограничение – един потребител може да остави само едно ревю за даден продукт. Това се реализира чрез уникално условие върху комбинацията от потребител и продукт, което гарантира, че мнението на клиента е представено само веднъж.

Функционалността **„GET THE LOOK“** е реализирана чрез връзка много-към-много (N–N) между моделите **Outfit** и **Product**, осъществена чрез междинния модел **OutfitItem**. Това позволява към едно облекло да се свържат няколко продукта, както и един продукт да бъде част от повече от едно облекло.

Тази система от връзки позволява ефективно управление на данните, улеснява извличането на информация и поддържа висока степен на интегритет между отделните модули на приложението.

**5.3. Функционални модули**

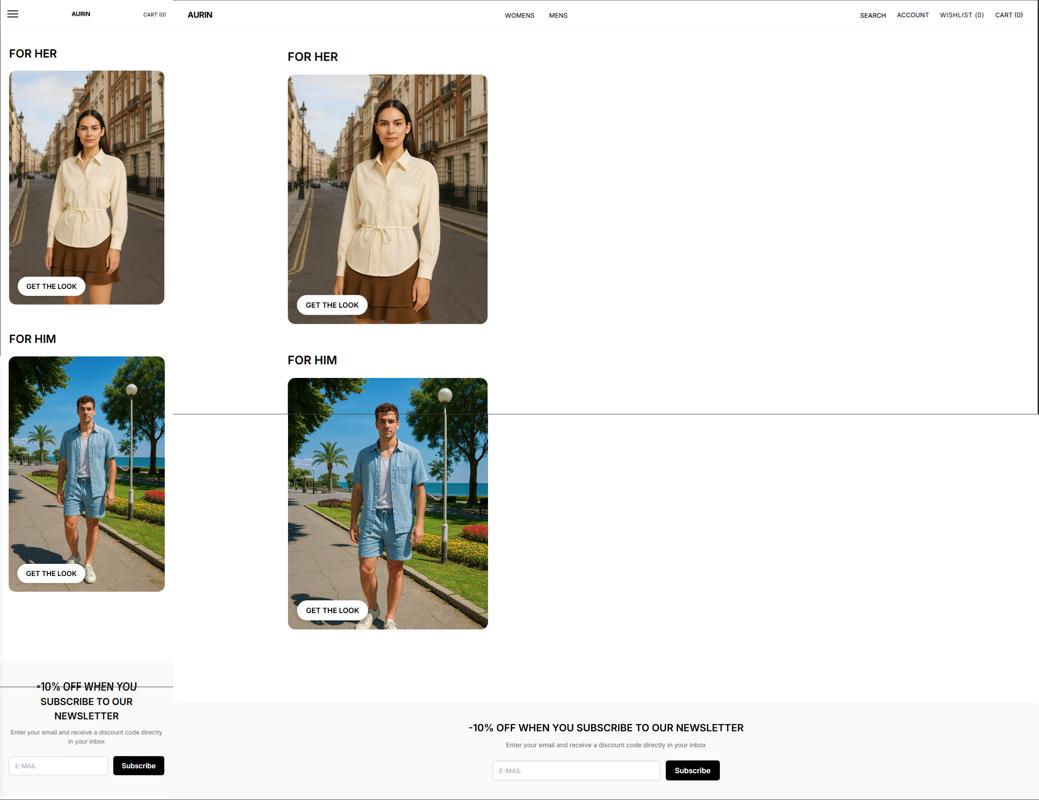
**5.3.1. Начална страница**

Началната страница на приложението служи като основна точка за влизане на потребителя в онлайн магазина. Нейната роля е да представи съдържанието по ясен и интуитивен начин, така че посетителят бързо да се ориентира и да започне пазаруването.

В горната част на страницата е разположен **хедър** с основното меню за навигация. То включва връзки към каталога на продуктите, пазарската количка, списъка с желания и профила на потребителя. В хедъра са позиционирани и бутоните за вход и регистрация, реализирани като модални прозорци.

Под хедъра са представени **продуктовите секции** за мъжка и женска мода. Във всяка секция се показват продукти с изображение, име и цена, като всяко изображение е линк към детайлната страница на продукта. Разположението е оптимизирано така, че потребителят бързо да прегледа наличните артикули и да избере интересуващия го продукт.

В долната част на страницата е разположена **форма за абонамент**. При успешно абониране системата предоставя **промоционален код за 10% отстъпка** от следваща покупка.

Дизайнът на началната страница е изчистен, с акцент върху продуктите. Структурата е адаптивна, което осигурява коректно визуализиране както на десктоп, така и на мобилни устройства.

Фигура 4: Начална страница на проекта(десктоп и мобилна адаптация).

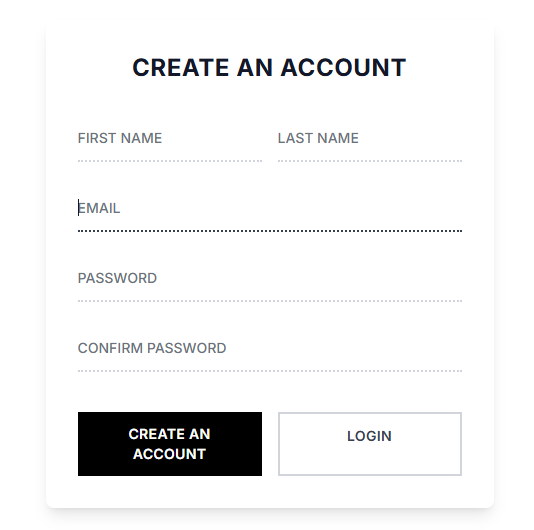
**5.3.2. Регистрация и вход + възстановяване на парола**

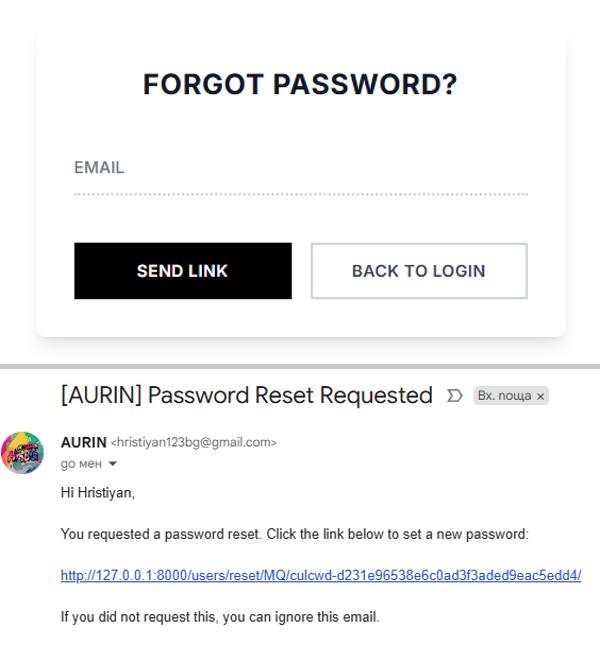
Достъпът до функциите за вход, регистрация и възстановяване на парола е реализиран чрез бутона **„ACCOUNT“** в хедъра на сайта. При натискане на бутона се отваря **модален прозорец** за вход в системата. Той съдържа поле за въвеждане на имейл адрес и парола, бутон **„LOG IN“** за вход, линк **„FORGOT PASSWORD?“** за възстановяване на парола и бутон **„CREATE AN ACCOUNT“** за преминаване към форма за регистрация.

Фигура 5: Форма за вход в сайта

Входът в системата се осъществява директно в този модален прозорец. При въвеждане на валидни данни потребителят получава достъп до своя профил, история на поръчките, пазарската количка и списъка с желания. При некоректни данни се показва съобщение за грешка.

Регистрацията се извършва след избор на бутона **„CREATE AN ACCOUNT“** в модала за вход. Отваря се форма, в която се въвеждат имена, имейл адрес, парола и потвърждение на паролата. След изпращане на формата системата създава нов акаунт и автоматично вписва потребителя или го пренасочва към формата за вход.

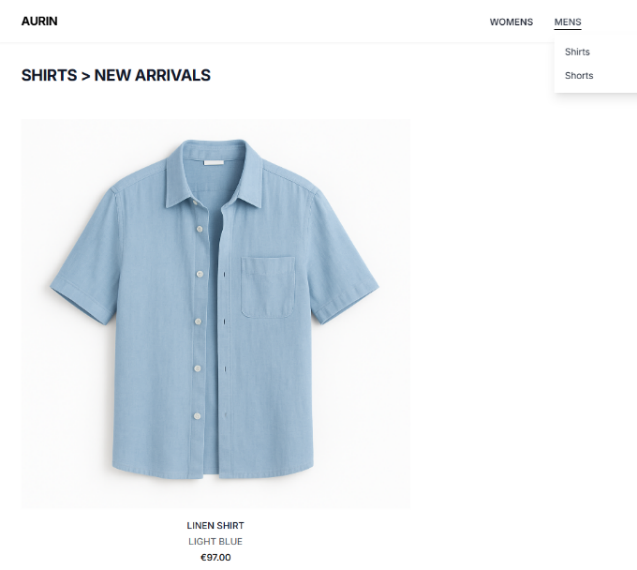
Фигура 6: Форма за регистрация в сайта

Функцията **„FORGOT PASSWORD?“** отваря форма за възстановяване на парола, където потребителят въвежда своя имейл адрес. На него автоматично се изпраща съобщение с линк за смяна на паролата, генериран от системата. Изпращането се извършва чрез конфигуриран **Gmail SMTP** сървър, като за сигурност се използва TLS криптиране. Линкът е валиден за ограничено време и води към защитена страница за задаване на нова парола.

Фигура 7: Форма за забравена парола и получен мейл

Тази централизирана реализация на вход, регистрация и възстановяване на парола в един модален прозорец осигурява бърз достъп до основните функции, без да е необходимо презареждане на страницата.

**5.3.3. Каталог и продукт детайл**

**Каталогът** на онлайн магазина е организиран по категории, позволявайки на потребителите лесно да откриват търсените продукти. Всяка категория съдържа списък с артикули, представени с основно изображение, име и цена. Изображението и името на продукта служат като връзка към **детайлната страница** на продукта.

Фигура 8: Категории в хедъра и каталог на артикулите

**Детайлната страница** показва пълната информация за избрания артикул – голямо основно изображение, допълнителни снимки (ако са налични), име, цена и подробно описание. Под основните данни се намира секция за **избор на размер**, реализирана чрез радио бутони. Потребителят трябва да избере размер, преди да добави продукта в пазарската количка или в списъка с желания.

Фигура 9: Детайлна страница на продукта

На същото място е разположена и функционалността **„Find My Size“**(виж Фиг.9). Тя представлява интерактивна форма, в която потребителят може да въведе своя ръст и тегло (задължителни полета) и, по желание, обиколка на талия и ханш. Системата извършва изчисление и препоръчва най-подходящия размер, като автоматично маркира съответния радио бутон. Това намалява риска от грешен избор и улеснява процеса на поръчка.

Под секцията с размери са разположени бутоните **„ADD TO CART“** и **„ADD TO WISHLIST“**, които чрез AJAX заявка добавят избрания продукт (с конкретния размер) в количката или в списъка с желания без презареждане на страницата.

В долната част на страницата е секцията с **ревюта**, където потребителите могат да видят мнения и оценки на други клиенти които вече са закупили продукта.

Фигура 10: Ревю на продукт

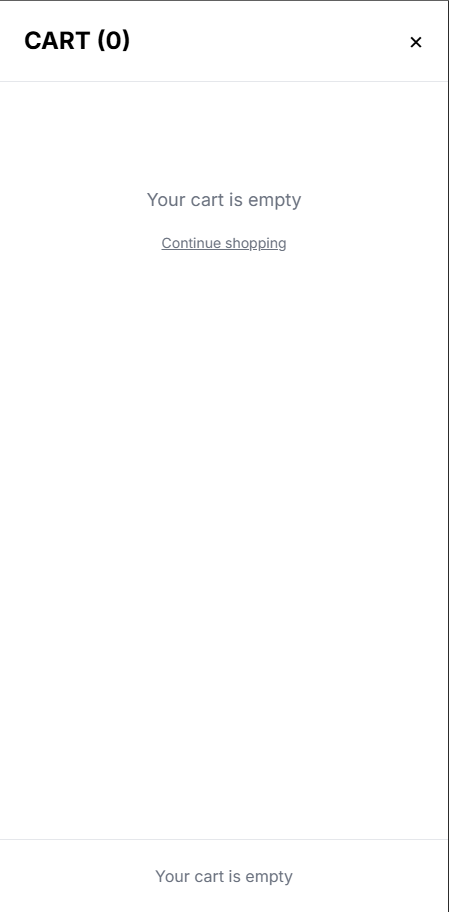
Тази комбинация от ясна продуктова информация, помощна функция за избор на размер и бързи бутони за добавяне в количката или списъка с желания прави детайлната страница удобна и интуитивна за потребителите.

**5.3.4. Пазарска количка**

Пазарската количка е реализирана като **модален прозорец**, който се отваря при натискане на иконата на количката в хедъра. Това позволява на потребителя да прегледа съдържанието на количката, без да напуска текущата страница.

Всеки ред в количката (**CartItem**) съдържа изображение на продукта, неговото име (линк към детайлната страница), избрания размер, единичната цена, количеството и общата стойност за този ред. Потребителят може да променя количеството чрез бутони „+“ и „–“, а също така да премахне продукта напълно от количката чрез бутон за изтриване.

Всички действия – добавяне на продукт, промяна на количество или премахване – се извършват чрез **AJAX заявки**, което означава, че модалът се обновява динамично, без презареждане на страницата. Това прави управлението на количката бързо и удобно.

Фигура 11: Модален прозорец на количката

В долната част на модалния прозорец е показана междинната сума (**Subtotal**), потребителят може да продължи директно към процеса на поръчка чрез бутона **„CHECKOUT“**, който го прехвърля към страницата за завършване на покупката.

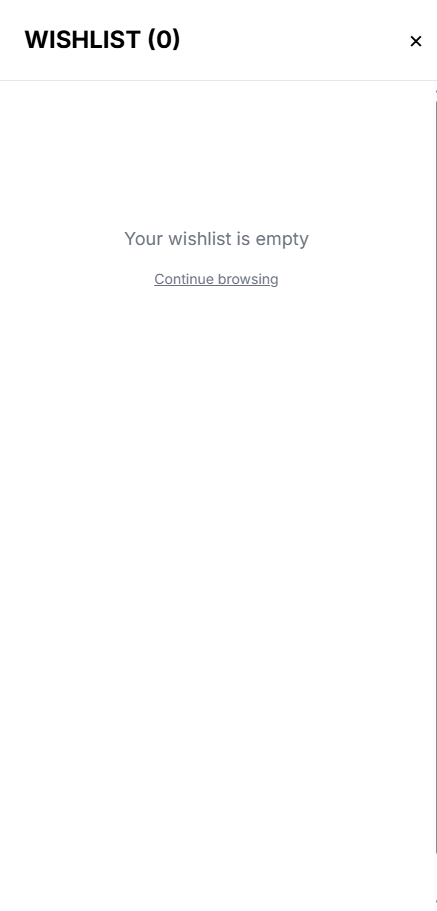
Дизайнът на количката е изчистен и следва същия визуален стил като останалата част от сайта, като акцентът е върху бързото и лесно управление на избраните продукти. разработка, улеснява бъдещото разширяване на системата и осигурява централизирано управление на бизнес логиката.

**5.3.5. Списък с желания**

Списъкът с желания е функционалност, която позволява на потребителите да запазят продукти за бъдеща покупка. Достъпът до него се осъществява чрез иконата на сърце в хедъра на сайта. При натискане се отваря **модален прозорец**, който показва всички добавени артикули.

Всеки елемент (**WishlistItem**) съдържа изображение на продукта, име (с линк към детайлната страница), избран размер и единична цена. За разлика от пазарската количка, тук **няма поле за количество**, тъй като списъкът с желания служи за запазване на предпочитани артикули, а не за директно финализиране на покупка.

Добавянето на продукт в списъка с желания става от детайлната страница на продукта чрез бутона **„ADD TO WISHLIST“**, разположен под избора на размер. Системата изисква задължително да е избран размер, преди продуктът да бъде добавен. Операцията се извършва чрез **AJAX заявка**, което позволява добавянето да стане без презареждане на страницата.

В модалния прозорец потребителят може да премахне артикул от списъка с желания или да го прехвърли директно в пазарската количка, като в този случай избраният размер се запазва автоматично.

Фигура 12: Модален прозорец на списъка с желания

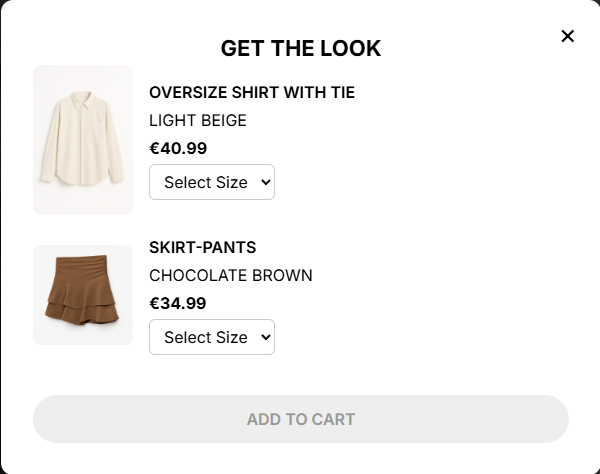
**5.3.6. GET THE LOOK**

Функционалността „GET THE LOOK“ позволява на потребителите с един клик да закупят цялостна стилова комбинация от няколко продукта, които се допълват визуално и тематично. Целта е да се улесни процесът на пазаруване и да се насърчат покупките на цели аутфити, вместо на единични артикули.

Достъпът до „GET THE LOOK“ се осъществява от началната страница. При избор на конкретно облекло се отваря модален прозорец, който показва всички продукти, включени в него. Всеки продукт е представен с изображение(линк към детайлната страница), име, цена и падащо меню за избор на размер.

Потребителят може да избере размер за всеки артикул от комбинацията и след това да добави всички избрани продукти в пазарската количка наведнъж чрез бутон „ADD TO CART“. Операцията се извършва чрез AJAX заявка, което позволява динамично обновяване на количката без презареждане на страницата.

Дизайнът на модалния прозорец е интуитивен и следва стилистиката на сайта – ясна подредба на продуктите, добре видими контроли за избор на размер и акцент върху бутона за добавяне в количката.

Тази функционалност значително съкращава времето за пазаруване, тъй като потребителят не е необходимо да посещава всяка продуктова страница поотделно, за да избере размер и да добави артикул в количката.

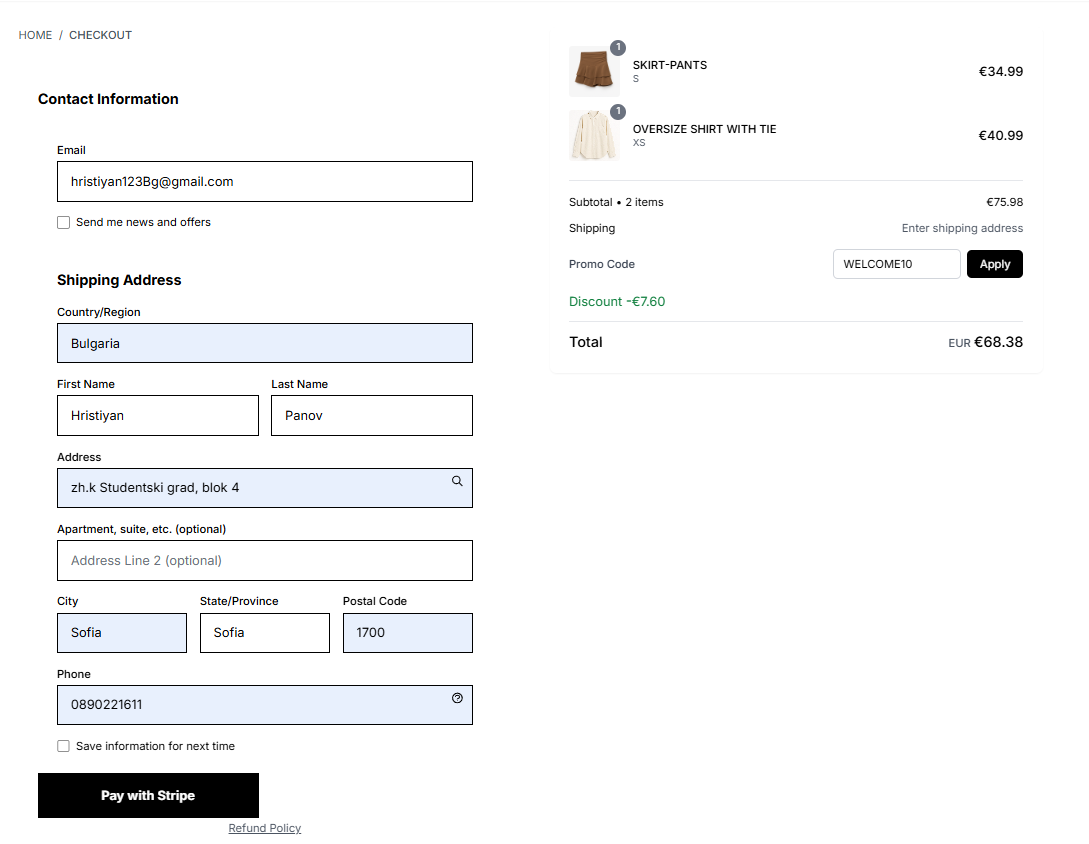
Фигура 13: Модален прозорец на “GET THE LOOK”

**5.3.7. Поръчка и Checkout**

Процесът на поръчка (**Checkout**) е ключов етап в работата на онлайн магазина, в който потребителят финализира избраните артикули от пазарската количка и извършва плащането. Достъпът до Checkout се осъществява чрез бутона **„CHECKOUT“** в модалния прозорец на количката.

Страницата за поръчка съдържа формуляр, разделен на няколко секции:

1. **Данни за доставка** – име и фамилия, адрес, град, пощенски код, телефон за контакт и имейл.
2. **Преглед на поръчката** – списък с продуктите, избраните размери, количествата, единичните цени и общата стойност за всеки ред.
3. **Обобщение на сумите** – междинна сума (**Subtotal**), разходи за доставка (**Shipping**), евентуална отстъпка (**Discount**) и крайна цена (**Total**).
4. **Метод на плащане** – избор на наличния метод (Stripe).

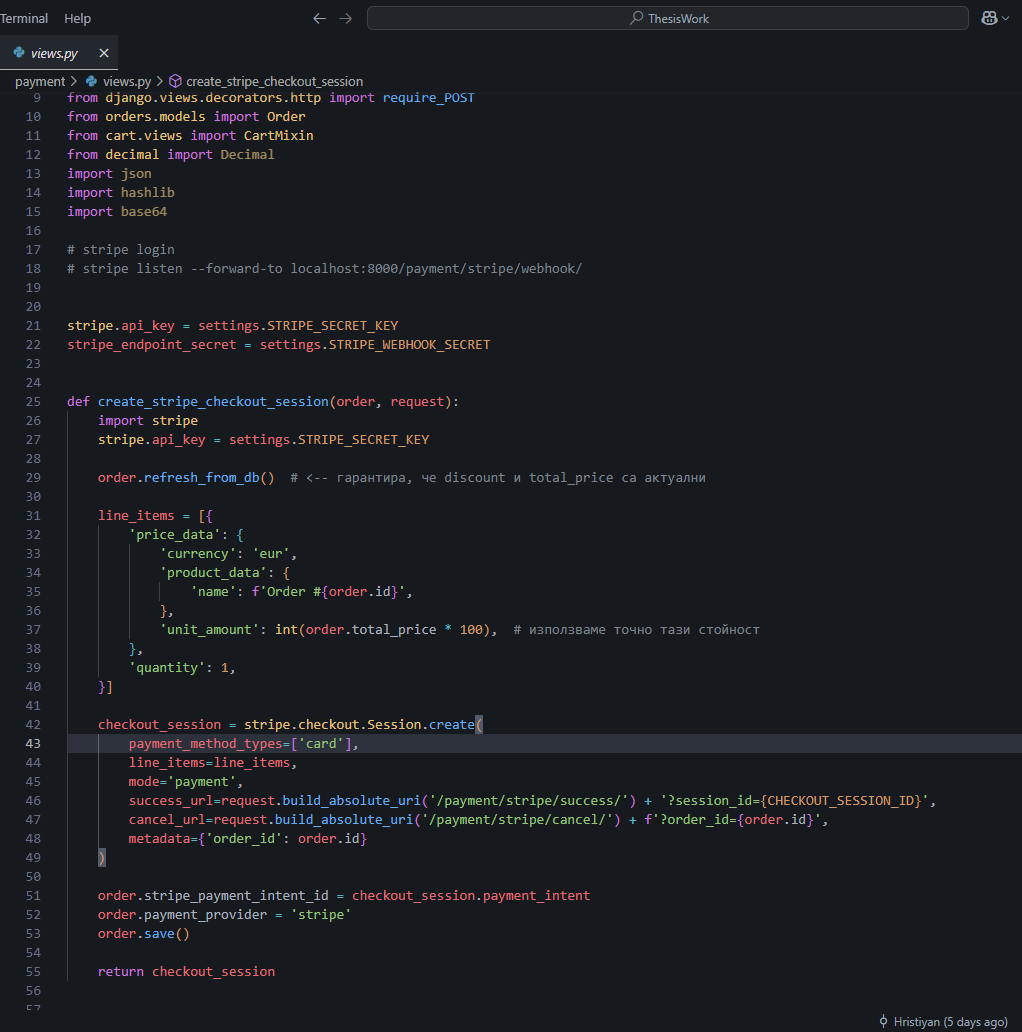
След попълване на данните и потвърждаване на поръчката, потребителят се пренасочва към процеса на плащане. Всички данни от формуляра се валидират, за да се гарантира коректното въвеждане и да се избегнат грешки при обработката на поръчката.

Фигура 14: Страница за финализиране на поръчката

**5.3.8. Плащане със Stripe**

Системата използва Stripe Checkout за обработка на онлайн плащания с дебитни и кредитни карти. Интеграцията е конфигурирана с Stripe Secret Key и Webhook Secret, съхранени в .env файла, което гарантира сигурност и лесно управление на настройките.

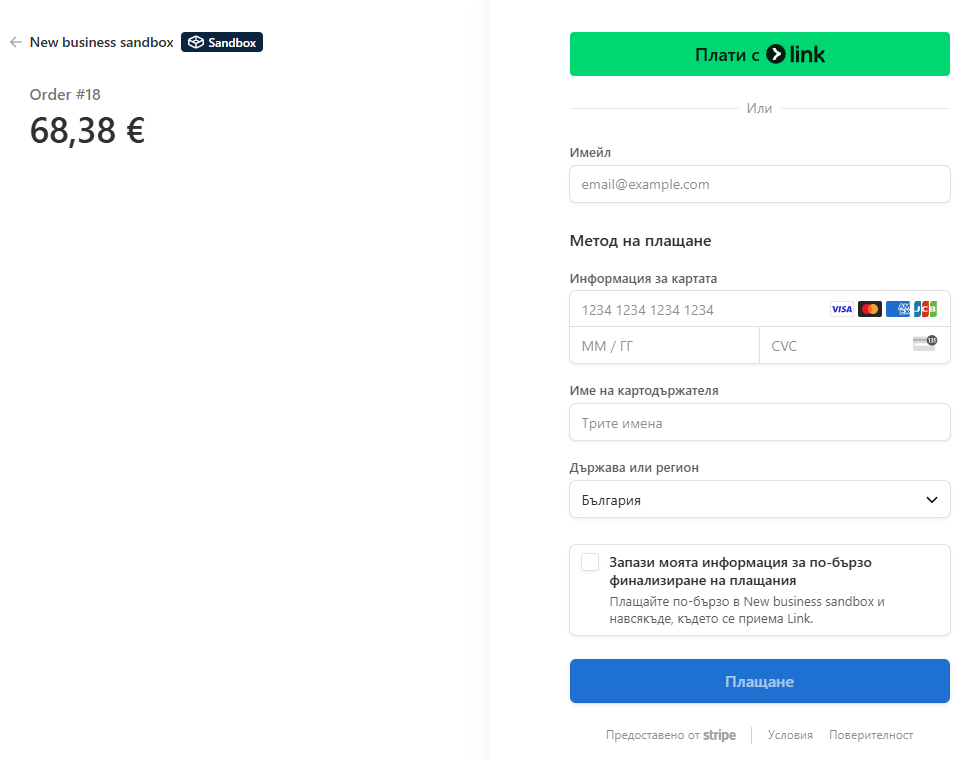
Процесът започва при финализиране на поръчка от Checkout страницата. Извиква се функцията create\_stripe\_checkout\_session(), която създава нова Stripe Checkout сесия и предава на Stripe данните за поръчката:

* Валутата (EUR)
* Името на поръчката
* Общата стойност на поръчката (в евро, конвертирана в центове)
* Линкове за успешно и неуспешно плащане
* Метаданни с идентификатора на поръчката

Фигура 15: Откъс от кода за създаване на Stripe Checkout сесия

След завършване на плащането Stripe изпраща **webhook** съобщение до нашия сървър. То се обработва от функцията **stripe\_webhook()**, която проверява подписа на съобщението и при потвърдено плащане актуализира статуса на поръчката в системата.

Фигура 16: Откъс от кода за обработка на Stripe webhook

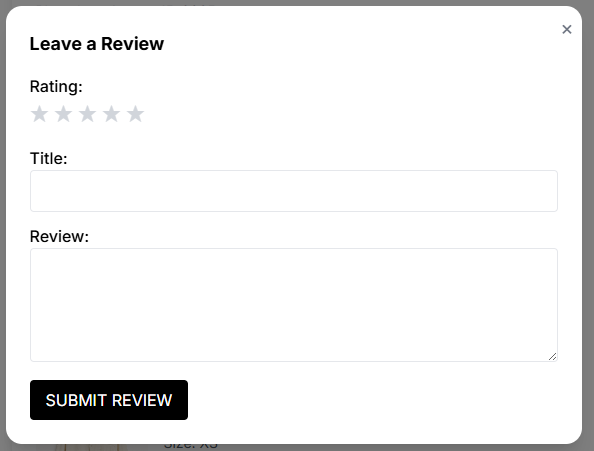
Системата разполага и с отделни изгледи за обработка на успешни и отказани плащания (**stripe\_success()** и **stripe\_cancel()**), които показват съответните страници и актуализират статуса на поръчката.

Фигура 17: Stripe Checkout страница

**5.3.9. Ревюта**

Модулът за ревюта позволява на регистрираните потребители да оставят оценка и кратък коментар за закупен продукт. Целта е да се предостави достоверна обратна връзка и да се подпомогне решението за покупка на други клиенти.

**Логика и правила:**

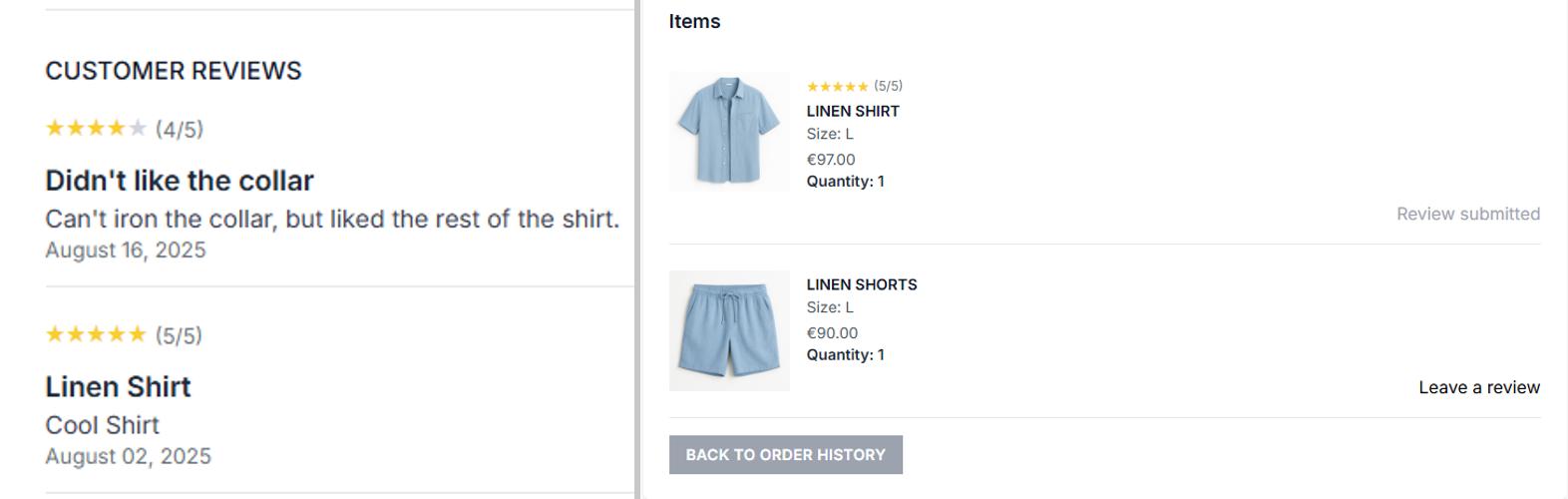
* **Едно ревю на потребител за продукт.** Ако потребителят вече е оставил ревю, бутонът „Leave a review“ в детайлната страница на поръчката(order\_detail) **не може да се използва** (скрит е за този потребител).

Фигура 18: Модален прозорец за оставяне на ревю към артикул

* **Визуализация в продукта.** В product\_detail се изписват наличните ревюта, дата, оценка (звезди) и текст на отзива.
* **Валидиране.** Формата валидира задължителните полета (най-малко оценка + текст/коментар според настройките).
* **Административен контрол.** Ревютата са видими в администрацията, където могат да се преглеждат и модерират при нужда.

**Потребителски поток:**

1. Потребителят отваря product\_detail.
2. Ако не е оставял ревю и е логнат, вижда бутон „Leave a review“.
3. Подаване на формата → запис в базата → връщане към детайла на продукта с актуализиран списък от ревюта.
4. При повторно посещение бутонът вече не се показва.

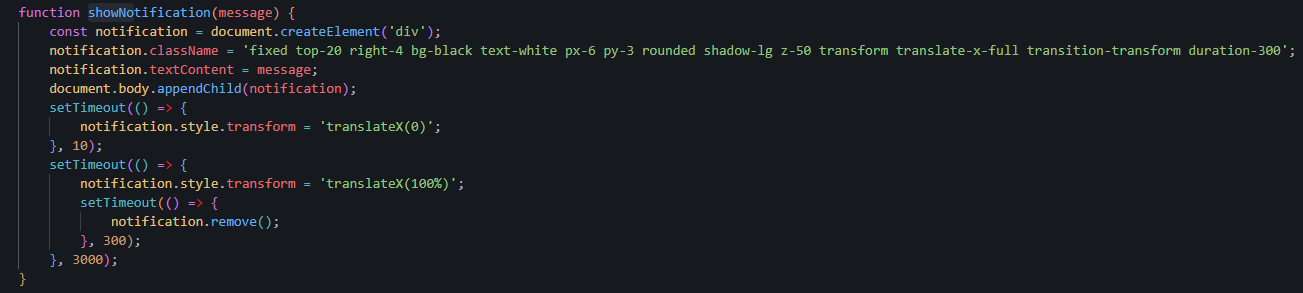
Фигура 19: Визуализация на ревютата в страницата на артикула, активен и неактивен бутон за оставяне на ревю

**5.3.10. Нотификации**

В системата са реализирани **визуални известия (toasts)**, които информират потребителя за успешни или неуспешни действия. Те се показват в горния десен ъгъл на екрана и автоматично изчезват след няколко секунди.

Нотификациите се използват при ключови събития, като:

* добавяне на продукт в пазарската количка;
* добавяне на продукт в списъка с желания;
* успешно завършване на ревю към артикул;
* грешки при избор на размер или при опит за добавяне на продукт без необходимите данни.

Функцията е реализирана чрез **JavaScript**, който създава динамично HTML елемент за съобщението, прилага му CSS класове за стил и анимация, след което го премахва от DOM дървото след изтичане на времето за показване. Това позволява лесна употреба и повторяемост за различни събития в системата.

Фигура 20: Откъс от JavaScript използван за нотификацията в проекта

По този начин потребителят получава незабавна обратна връзка за действията си, без да се налага презареждане на страницата или допълнителни потвърждения. визуализацията на данни, комуникацията със сървъра и динамичното рендиране на административните форми.

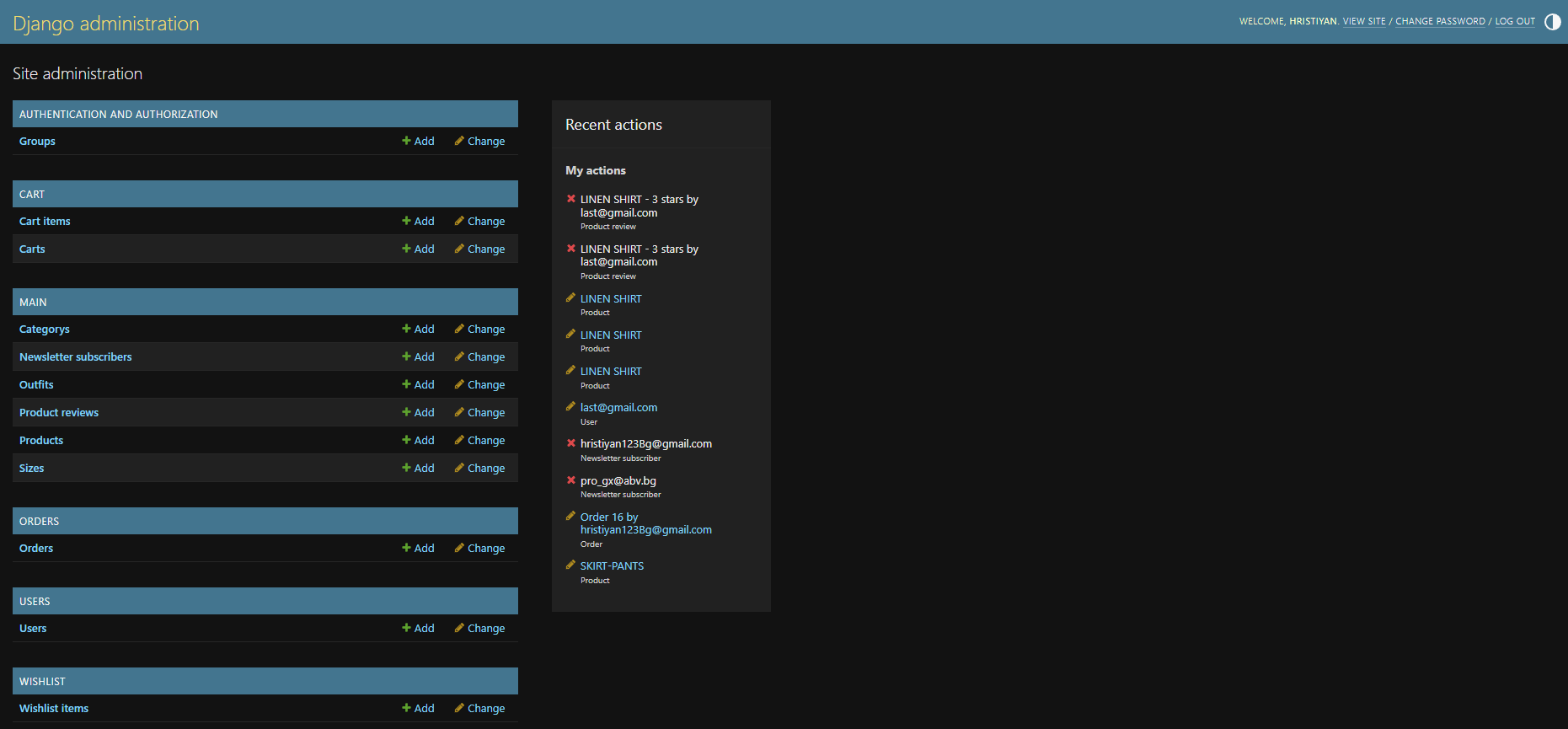
Фигура 21: Toast нотификация

**5.3.11. Админ панел**

Django предоставя вграден административен панел, който е разширен и конфигуриран в проекта за управление на основните данни в системата. Той е достъпен единствено за потребители с администраторски права и позволява ефективно администриране на онлайн магазина без необходимост от директна работа с базата данни.

**Функционалности на админ панела:**

* **Управление на продукти** – добавяне, редактиране и изтриване на артикули; качване на изображения; определяне на категории и размери.
* **Управление на поръчки** – преглед на направени поръчки, промяна на статуси (например „обработва се“, „завършена“, „отменена“), преглед на свързаните артикули и плащания.
* **Преглед на ревюта** – контрол върху потребителските мнения, с възможност за премахване на неподходящи или некоректни ревюта.
* **Управление на облекла (GET THE LOOK)** – създаване и редактиране на комбинации от продукти, които се показват в модула.
* **Потребители** – преглед и администриране на регистрирани клиенти, промяна на техните данни или права.

Тази конфигурация позволява на администратора бързо да намира продукти, поръчки и ревюта чрез търсене и филтриране, както и да управлява информацията в реално време.

Фигура 22: Начална страница на админ панела

**5.3.12. Сигурност**

Сигурността е един от основните приоритети при изграждането на системата. Django предоставя множество вградени механизми, които са използвани и разширени, за да гарантират защита на потребителските данни, плащанията и цялостната работа на приложението.

**Основни аспекти на сигурността:**

* **Удостоверяване и авторизация** – реализирани чрез вградената система на Django за управление на потребители и права на достъп. Административният панел и API крайните точки са достъпни единствено за упълномощени потребители.
* **Хеширане на пароли** – всички пароли се съхраняват в базата данни чрез силни алгоритми за хеширане (PBKDF2), предоставени от Django.
* **CSRF защита** – формите и заявките, които променят данни, са защитени чрез автоматично генерирани CSRF токени, което предпазва от Cross-Site Request Forgery атаки.
* **Валидация на входните данни** – при регистрация, вход, поръчка и плащане се извършва проверка за коректност и пълнота на въведената информация.
* **Сигурни плащания със Stripe** – интеграцията със Stripe API осигурява обработка на чувствителни картови данни извън нашия сървър, като информацията за плащането никога не се съхранява в базата данни на приложението. Всички транзакции се потвърждават чрез webhook-и, защитени със секретен ключ.
* **Защита на конфигурационни данни** – ключове, пароли и чувствителна информация са съхранени във .env файл и не се включват в публичното хранилище на кода.
* **Ограничения за ревюта** – всеки потребител може да остави само едно ревю за даден продукт, което предотвратява злоупотреби.

Тези мерки осигуряват надеждност и устойчивост на системата срещу често срещани заплахи в уеб приложенията и гарантират сигурността на потребителите по време на работа със сайта.

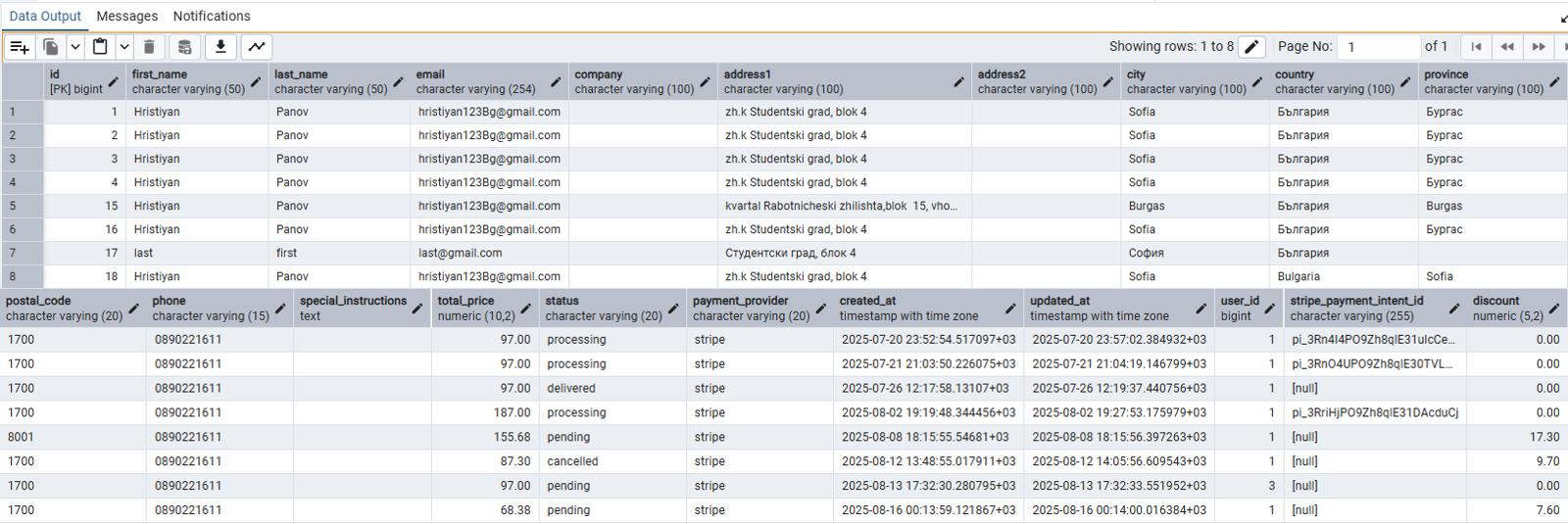
# ГЛАВА 6. ТЕСТВАНЕ И ОЦЕНКА

## 6.1. Подход и среда за тестване

За да се гарантира надеждността и правилното функциониране на системата, е приложен комбиниран подход за тестване, включващ:

* **Ръчно тестване** – извършвано от разработчика по време на имплементацията, с цел проверка на основните сценарии за работа на сайта.
* **Функционално тестване** – проверка на конкретните модули (количка, списък с желания, поръчки, плащания и др.), за да се уверим, че реализират описаните изисквания.
* **Интеграционно тестване** – оценка на взаимодействието между отделните модули, например: добавяне на продукт → поръчка → плащане → актуализация на статус.
* **Тестване на потребителското изживяване (UX)** – фокус върху удобството при използване на модали, бързината на AJAX операциите и визуалното представяне.

**Средата за тестване** включва:

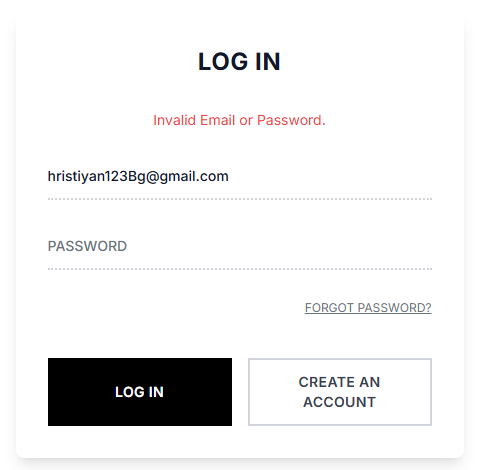
* Операционна система: Windows/Linux (локална разработка).
* Сървърна среда: Django с вграден дев-сървър.
* База данни: PostgreSQL.
* Платежна интеграция: Stripe тестов режим.
* Имейл услуга: Gmail SMTP със специален тестов акаунт.
* Браузъри: Chrome, Firefox, Edge.
* Тестови потребители: създадени специално за проверка на регистрация, вход, поръчки и ревюта.

Фигура 23: Примерни тестови данни от таблицата „orders\_order“ в pgAdmin

## 6.2. Функционални тестове

За проверка на коректността и надеждността на системата бяха проведени функционални тестове, които обхващат всички основни модули на онлайн магазина. Тестовете потвърдиха, че както при валидни действия, така и при грешно въведени данни, системата реагира правилно – показва предупреждение и не позволява извършване на невалидно действие.

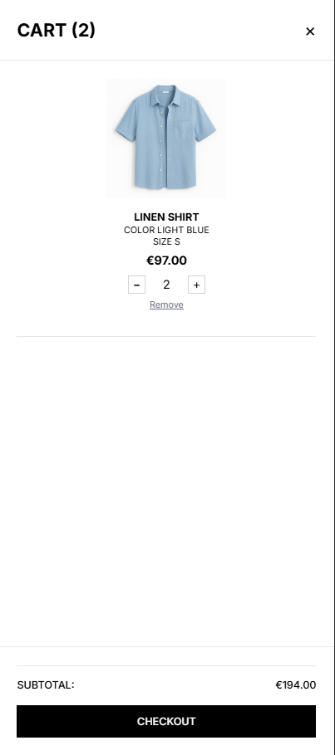
### 6.2.1. Потребители и автентикация

* Регистрация на нов потребител чрез модален прозорец – успешна при валидни данни.
* Вход с правилни данни – успешно пренасочване към началната страница.
* Вход с грешни данни (невалиден имейл или парола) – системата показва предупреждение и не допуска достъп.
* „Forgot password“ – при въвеждане на валиден имейл се изпраща съобщение за промяна на парола. При грешен имейл системата предупреждава, че акаунт с такъв имейл не съществува.

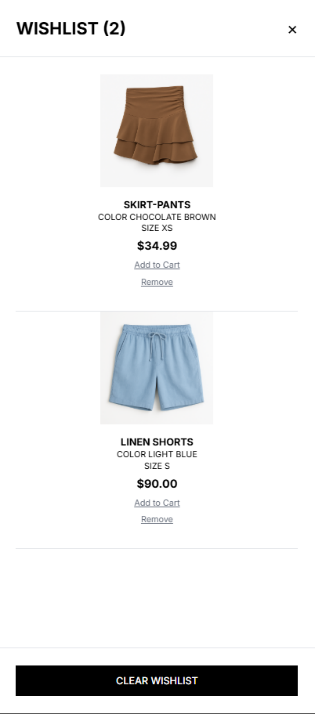
Фигура 23: Грешка при неправилно въведена парола при опит за вход

### 6.2.2 Пазарска количка (Cart)

* Добавяне на продукт с избран размер – броячът в количката се обновява динамично.
* Промяна на количеството – актуализира се автоматично цената.
* Опит за въвеждане на невалидно количество (напр. отрицателно число или текст) – системата показва предупреждение и не позволява действието.
* Премахване на продукт – модалът се обновява без презареждане.

Фигура 24: Модален прозорец на количка с артикули

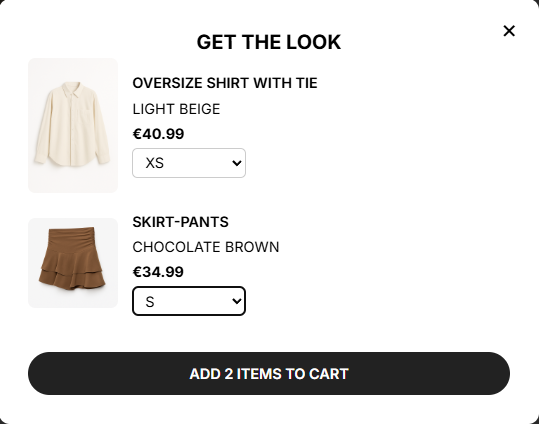
### 6.2.3. Списък с желания (Wishlist)

* Добавяне на продукт със зададен размер – появява се в модала „Wishlist“.
* Опит за добавяне на продукт без избран размер – системата изисква избор и не позволява добавяне.
* Премахване или прехвърляне на продукт в количката – действията се отразяват незабавно.

Фигура 25: Модален прозорец на списък с желания с артикули

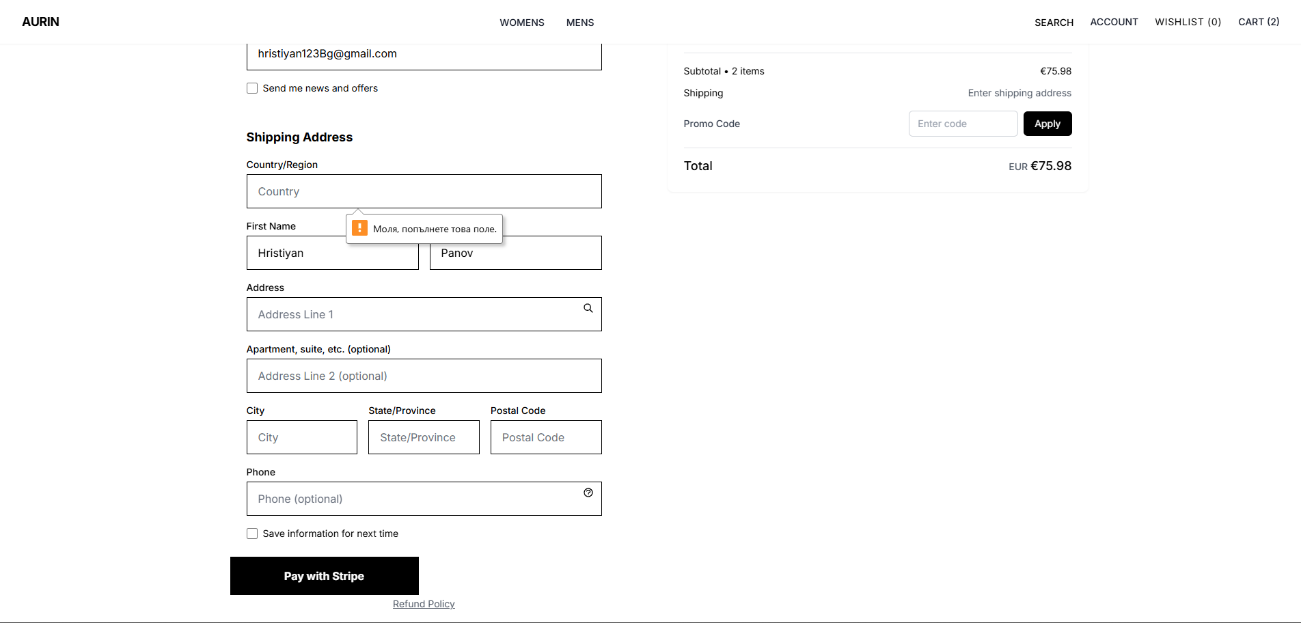
### 6.2.4. Модул „GET THE LOOK“

* Зареждане на аутфит – визуализират се всички продукти в комбинацията.
* Избор на размер за всеки продукт – задължителен преди добавяне в количката.
* Опит за добавяне без избран размер – системата предупреждава и не извършва действието.
* При правилно попълнени данни – всички артикули се добавят едновременно в количката.

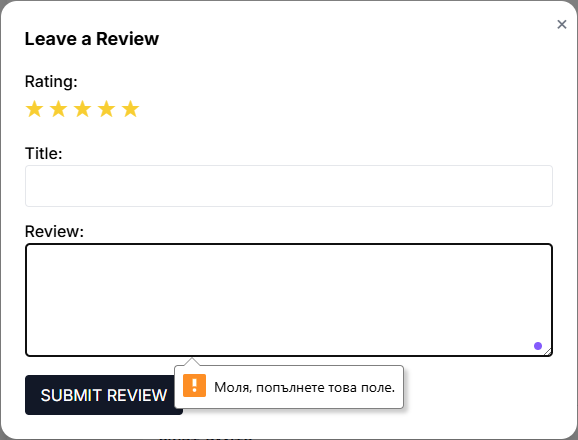
Фигура 26: Модален прозорец на “Get the look” преди добавяне на артикули в количката

### 6.2.5. Поръчки и плащания

* Преглед на количката и преминаване към checkout.
* Попълване на валидни данни за доставка – успешно преминаване към плащане.
* Пропуснато или грешно въведено поле – системата извежда грешка и не допуска финализиране на поръчката.
* Успешна интеграция със **Stripe** – при коректни данни плащането се осъществява. При грешни данни картата е отхвърлена и се показва съобщение за неуспешна транзакция.

Фигура 27: Предупреждение за непопълнено поле в checkout формата

### 6.2.6. Ревюта

* Добавяне на ревю с валидни данни – успешно визуализиране на страницата на продукта.
* Опит за оставяне на ревю без избрана оценка (звезди) или без текст – системата предупреждава за непопълнени полета и не приема ревюто.
* Втори опит за ревю на същия продукт от същия потребител – бутонът „Leave a review“ не е наличен.

Фигура 28: Предупреждение за непопълнено поле в review формата

## 6.3. Резултати и оценка на системата

Проведените тестове потвърждават, че реализираната система функционира коректно и отговаря на предварително заложените изисквания. Всички основни модули – регистрация и автентикация, пазарска количка, списък с желания, модул „GET THE LOOK“, управление на поръчки и плащания, както и модулът за ревюта – работят стабилно и осигуряват необходимата функционалност.

Системата показва правилно поведение както при коректно въведени данни, така и при опити за извършване на невалидни действия. При грешно въведени данни се извеждат подходящи съобщения за грешка, а операциите не се извършват. Това гарантира надеждност и сигурност на процесите, както и по-добро потребителско изживяване.

По време на разработката са използвани вградените защитни механизми на Django, които осигуряват стабилност и защита на приложението:

* **Хеширане на пароли** – паролите на потребителите не се съхраняват в чист вид, а се криптират чрез надеждни алгоритми (PBKDF2 със SHA256).
* **CSRF защита** – всяка форма и AJAX заявка използва уникален токен, който предотвратява Cross-Site Request Forgery атаки.
* **Защита срещу SQL Injection** – Django ORM автоматично създава заявки, които не позволяват инжектиране на зловреден SQL код.
* **Сигурна работа със сесии** – данните за сесиите се съхраняват на сървъра, а клиентът получава само уникален идентификатор.
* **Валидиране на формите** – при некоректно въведени данни потребителят получава предупреждение, а изпращането се блокира.
* **HTTPS и сигурна интеграция с външни услуги** – връзката със Stripe и Gmail SMTP протича по криптиран канал.

Тестовете на потребителския интерфейс също потвърждават, че системата предлага динамично и интуитивно изживяване. Действия като добавяне на продукти в количката или списъка с желания, избор на размери и финализиране на поръчка се извършват без презареждане на страницата, което създава усещане за бързина и лекота при работа.

В резултат от проведените тестове може да се заключи, че системата не само изпълнява своите функционални изисквания, но и осигурява високо ниво на сигурност и потребителска удовлетвореност. Разработената платформа отговаря на съвременните изисквания за онлайн магазин за дрехи и предоставя стабилна основа за бъдещо разширяване и надграждане.

# Глава 7. ЗАКЛЮЧЕНИЕ И БЪДЕЩО РАЗВИТИЕ

В резултат на проведената разработка е изградена и внедрена система за онлайн магазин за дрехи, която съчетава модерен интерфейс, динамично взаимодействие с потребителя и надеждна сървърна част. Основната цел – създаване на динамичен сайт, който не презарежда страниците при всяко действие – е успешно постигната чрез интеграция на **AJAX (Fetch API)** и модални прозорци.

Системата осигурява пълна функционалност, характерна за съвременна платформа за електронна търговия: регистрация и автентикация на потребители, пазарска количка и списък с желания, модул „GET THE LOOK“, ревюта и оценки, управление на поръчки и плащания чрез Stripe. Реализирани са и защити, предоставяни от Django, включително хеширане на пароли чрез **PBKDF2 със SHA256**, CSRF защита и сигурно управление на сесиите.

**Въпреки успешната реализация, системата има някои ограничения:**

* плащанията са ограничени единствено до Stripe в тестов режим;
* липсва мобилно приложение, което да осигури достъпност на по-широка аудитория;
* препоръчващ механизъм за персонализирани предложения не е реализиран;
* административният панел се базира на стандартния Django Admin и не е персонализиран.

**Възможностите за бъдещо развитие включват:**

* интеграция на допълнителни методи за плащане (PayPal, Google Pay, Apple Pay, наложен платеж);
* разработка на мобилно приложение за iOS и Android;
* изграждане на интелигентна система за препоръки, базирана на предишни покупки и поведение на потребителите;
* персонализиране на административния панел за по-удобно управление на продукти и поръчки;
* добавяне на многоезична поддръжка и оптимизация за международни пазари.

В заключение може да се посочи, че проектът изпълнява поставените цели и предлага стабилна основа за бъдещо развитие. Системата отговаря на съвременните изисквания за онлайн търговия, като същевременно предоставя възможност за надграждане и адаптация към нови технологии и потребителски нужди.

# Глава 8. ИЗПОЛЗВАНА ЛИТЕРАТУРА И ИЗТОЧНИЦИ

1. Django Documentation. The Web framework for perfectionists with deadlines.  
   URL: <https://docs.djangoproject.com/>
2. PostgreSQL Documentaion. The World's Most Advanced Open Source Relational Database

URL: <https://www.postgresql.org/docs/>

1. Stripe API reference – <https://stripe.com/docs/api>
2. Mozilla Developer Network (MDN) – Web Development Documentation  
   URL: <https://developer.mozilla.org/>
3. W3Schools – HTML, CSS, JavaScript Tutorials  
   URL: [https://www.w3schools.com/](https://developer.mozilla.org/)
4. GitHub – Примери и помощен код  
   URL: <https://github.com/>
5. Stack Overflow – Общност за въпроси и отговори  
   URL: <https://stackoverflow.com/>
6. Python Software Foundation – Официална документация  
   URL: <https://docs.python.org/3/>
7. ChatGPT – Генериране на текст и технически консултации  
   URL: <https://chat.openai.com/>
8. Zara – <https://www.zara.com/bg/>
9. H&M – <https://www2.hm.com/bg_bg/index.html>
10. Pull and Bear  
    URL:  [https://www.pullandbear.com/bg/%D0%BC%D1%8A%D0%B6%D0%BA%D0%BE-n6228?utm\_source=tradedoubler&utm\_medium=affiliate&utm\_campaign=fr\_branding&utm\_content=3350579&tduid=c8eaaa11dffd4677b472b90eb6018a2a](%20https://www.pullandbear.com/bg/%D0%BC%D1%8A%D0%B6%D0%BA%D0%BE-n6228?utm_source=tradedoubler&utm_medium=affiliate&utm_campaign=fr_branding&utm_content=3350579&tduid=c8eaaa11dffd4677b472b90eb6018a2a)
11. ASOS – <https://www.asos.com/>

## 3.5 Регистрация, вход и собствена captcha

В системата е реализиран модул за регистрация, логин, верификация на потребители и защита чрез собствен Captcha механизъм, който осигурява необходимата сигурност и контрол върху достъпа до клиентската част на онлайн магазина. Процесът по регистрация изисква от потребителя да въведе име, фамилия, имейл, парола, телефонен номер, адрес и пол. Подадената информация преминава валидация на сървърно ниво, като например се извършва проверка за уникалност на имейла в базата данни, а телефонният номер се обработва и проверява да съдържа точно девет цифри след премахване на водещата нула, ако такава е въведена.

Паролата се съхранява по сигурен начин чрез използване на криптографска хеш функция PBKDF2 със сол, генерирана чрез библиотеката crypto, което осигурява защита срещу brute-force атаки и т.н. След успешно завършване на регистрацията системата генерира уникален верификационен токен и изпраща имейл към потребителя, съдържащ линк за потвърждение на акаунта. Имейлът се изпраща с помощта на nodemailer, като съдържанието му се изгражда динамично на база предварително дефинирани шаблони.

Акаунтът на потребителя не може да бъде използван преди да бъде верифициран чрез получения линк, който съдържа генерирания токен. До верификация акаунтът остава със статус неактивен, което осигурява допълнително ниво на сигурност срещу неправомерна употреба.

При логин потребителят въвежда имейл и парола, като се извършва проверка за съществуването на въведения имейл, потвърждава се дали акаунтът е верифициран, а въведената парола се сравнява с хешираната стойност, съхранена в базата. След успешно удостоверяване се създава уникален сесиен токен, който се записва в таблицата със сесиите заедно със срока на валидност от 24 часа. Системата поддържа както активни потребителски сесии, така и анонимни сесии за клиенти, които разглеждат продукти без да са регистрирани.

За допълнителна защита срещу автоматизирани регистрации и атаки е създаден собствен Captcha механизъм. При генерирането на captcha се създава случайна аритметична задача със случайни числа и оператори събиране или изваждане. Текстът на задачата се визуализира графично чрез библиотеката canvas, като се добавят допълнителни визуални шумове, които затрудняват автоматизираното разчитане. Решението на задачата, заедно с генериран captcha token и IP адреса на потребителя, се съхранява временно в таблица в базата данни със срок на валидност от пет минути. При подаване на отговор от потребителя се проверява валидността на токена, съвпадението на IP адреса и коректността на отговора. След успешно валидиране записът се изтрива от базата.

Този цялостен механизъм за регистрация, логин, верификация и captcha защита осигурява надеждно управление на потребителските акаунти и предотвратява злоупотреби и автоматизирани опити за пробив.

## 3.6 Модул за управление на сесиите

В клиентската част на системата е реализиран модул за управление на сесиите, който позволява работа както с регистрирани, така и с анонимни потребители. При регистрираните потребители, след успешен логин, се създава нов запис в таблицата sessions, който съдържа уникален сесиен токен, валиден за определен период от време (24 часа), както и индикация дали сесията е активна. При последващи заявки към сървъра, сесията се валидира чрез middleware функция, която извлича сесийния токен от бисквитките, проверява валидността му в базата данни и асоциира потребителя към заявката. Ако токенът е невалиден или липсва, на потребителя се връща съобщение, че е необходимо да се логне, за да изпрати съответното съдържание.

Административната част използва отделна таблица за администраторските сесии (admin\_sessions), като логиката за валидиране е аналогична — сесиите се проверяват спрямо токените, записани в базата, като при успешно намерен запис се удостоверява самоличността на администратора.

Системата поддържа и функционалност за анонимни сесии, чрез които се реализира възможността за създаване и управление на количка от незалогнати потребители. При първоначално посещение на сайта, ако в бисквитките не съществува анонимен сесиен идентификатор, се генерира уникален идентификатор чрез UUID и се записва в бисквитка с ограничен срок на валидност. Този идентификатор се използва при асоцииране на избраните продукти в количката. В случай че по-късно потребителят се логне успешно, системата разполага с механизъм, който позволява обединяване на анонимната и потребителската количка, като съхранява всички избрани артикули и запазва логиката на пазаруването.

Този подход позволява едновременно работа с различни видове потребители, осигурява коректна асоциация на състоянието на количката и повишава удобството при използване на системата.

## 3.7 Забравена парола и промяна на данните на профила

Системата предоставя възможност за възстановяване на забравена парола и за редакция на профилните данни на потребителя след успешно влизане. При забравена парола потребителят подава своя имейл адрес, след което се извършва проверка дали съществува регистриран акаунт с този имейл. Ако такъв съществува, се генерира уникален токен за възстановяване и се записва в базата данни с ограничен срок на валидност. Потребителят получава имейл с линк за възстановяване, който съдържа токена. При отваряне на линка и въвеждане на нова парола, системата валидира токена, проверява неговата валидност и срок, и при успешно потвърждение обновява паролата в базата чрез новосъздаден хеш и сол. След успешна смяна на паролата, токенът за възстановяване се изтрива, за да не може да бъде използван повторно.

Промяната на профилните данни се извършва от вече логнат потребител. При промяна на имейл адреса се изисква повторна верификация на новия имейл, като за целта се генерира нов верификационен токен и се изпраща потвърждение на посочения нов имейл адрес. При подмяна на паролата се изисква въвеждане на текущата парола с цел потвърждение на самоличността. Валидират се също така останалите полета, като например телефонен номер, който трябва да съдържа точно деветцифрен формат. След преминаване на всички проверки, записите в базата данни се обновяват със съответните нови стойности.

## 3.8 Коментари и оценка на продукти

Системата предоставя възможност на потребителите да разглеждат детайлна информация за всеки продукт, да оставят коментари и оценки, както и да добавят продуктите в количката си. След като потребителят въведе желаните от него филтри в търсачката, се визуализира списък с продуктите, които отговарят на зададените критерии. За всеки от визуализираните продукти се предоставят два основни бутона – единият позволява директно добавяне на продукта в количката, а другият води към страница с детайлна информация за съответния продукт.

В страницата с подробности за продукта се извежда неговото описание, цена, изображения, както и текущото средно аритметично на направените до момента оценки. Ако потребителят е логнат в системата, той има възможност да публикува собствен коментар и да постави оценка на продукта. Оценките се съхраняват по такъв начин, че при повторно гласуване от същия потребител за същия продукт, неговата оценка се актуализира, вместо да се създава нов запис. Средната оценка се изчислява динамично чрез агрегиране на всички въведени оценки за дадения продукт. Публикуваните коментари се извличат от базата данни и се показват в хронологичен ред като допълнителна информация в детайлната страница на продукта.

В допълнение към това, при всяко извличане на продуктовите данни се извършва валидиране на информацията съобразно предварително зададен JSON Schema, което гарантира коректността на данните, преди те да бъдат визуализирани към потребителя.

## 3.9 Количка за избрани продукти

Системата разполага с пълноценна функционалност за управление на потребителска количка, като е реализирана поддръжка както за регистрирани потребители, така и за анонимни сесии. При добавяне на продукт в количката, системата първо проверява дали желаният брой бройки е наличен в склада. Ако продуктът съществува и количеството е достатъчно, тогава се извършва добавянето му в количката. При регистриран потребител, ако той вече има съществуваща количка в базата данни, новият продукт се добавя към нея. В случай че продуктът вече присъства в количката, количеството му се увеличава със зададената стойност. Ако потребителят все още няма създадена количка, се създава нов запис. За всяко добавяне в количката се намалява количеството на наличните продукти в склада.

В случай на анонимна сесия, за всеки потребител се създава уникален идентификатор на сесията, който се съхранява в cookie. Количката за анонимни потребители се управлява в отделна таблица и работи по аналогичен начин – при ново добавяне на продукт се проверява дали вече съществува запис за съответния продукт и ако да, количеството се увеличава, а в противен случай се създава нов запис. При промяна на количеството или премахване на продукт от количката, съответната операция актуализира или изтрива записите в базата според идентификатора на сесията или потребителя.

При влизане на потребител в акаунта си, системата извършва сливане на анонимната количка с тази на потребителя. Ако има продукти, които съществуват и в двете колички, системата обединява количествата. Така се осигурява безпроблемен преход между състоянието на количката при анонимно и регистрирано сърфиране.

При визуализиране на количката системата изчислява и показва крайната цена, като взема предвид евентуално активни промоции, които се прилагат върху цената на продукта. След това се изчислява стойността с приложена отстъпка и се начислява ДДС, като крайната сума с ДДС се визуализира на потребителя. Съществуват отделни функционалности за промяна на количествата, премахване на продукти от количката, както и изтриване на цялата анонимна количка при приключване на сесията.

Допълнително, при зареждане на количката за регистриран потребител, системата извлича и валидните за него ваучери на база кампании и таргет групи, в които участва, като показва само онези ваучери, които все още не е използвал.

## 3.10 Плащане

Процесът по финализиране на поръчка в системата започва след като потребителят е попълнил своята количка и е готов да завърши покупката. При изпращане на поръчката, системата първо извършва пълна проверка на всички артикули в количката, като се валидира дали има достатъчно налични количества за всеки от продуктите. Изчисляват се актуалните цени на базата на валидните към момента промоции, като се прави сравнение между цената, която е била при добавяне в количката, и актуалната стойност. Ако междувременно има промени в цените на някои от продуктите, системата актуализира количката и уведомява потребителя, че е необходимо да прегледа променените цени преди да продължи с поръчката. При успешно валидиране се изчисляват всички междинни суми като обща цена преди отстъпка, стойност на отстъпките, цена след отстъпки, ДДС, както и крайната сума с включено ДДС.

В случай че потребителят разполага с валиден ваучер, той може да го приложи в процеса на поръчката. За всеки ваучер се извършва допълнителна валидация, при която се проверява дали съответният потребител има право да използва конкретния ваучер, дали вече не го е използвал и дали е част от съответната таргет група на кампанията, към която е обвързан ваучерът. След прилагането на ваучера се преизчислява крайната сума по поръчката.

След като всички проверки преминат успешно, системата създава запис за новата поръчка в базата данни, като съхранява цялата изчислена информация за стойностите по нея, приложените ваучери и използваните кампании. След това се създават записите за всеки от продуктите в поръчката, като се актуализират и наличностите на продуктите в склада. Ако след приложението на ваучера крайната сума за плащане е станала нула, поръчката автоматично преминава в статус "платена" без необходимост от допълнително плащане.

Системата поддържа механизъм за извършване на плащания чрез PayPal API, като работи в PayPal Sandbox среда и позволява реална симулация на плащания. При плащане чрез PayPal системата първо генерира транзакция чрез PayPal API, връща на клиента URL за одобрение, а след това при завършване на плащането се извършва потвърждение чрез capture операция. След успешно плащане поръчката преминава в статус "progress" и се смята за потвърдена. Системата съхранява идентификаторите на PayPal транзакцията за бъдещи справки.

След успешно завършване на поръчката количката се изтрива, както от таблицата с артикулите, така и самият запис за количката на потребителя, като по този начин се гарантира, че същите артикули няма да останат налични в системата за следващи поръчки.

## 3.11 Нотификации

Системата разполага с модул за нотификации, който дава възможност за изпращане и управление на съобщения към потребителите. Всяка нотификация се съхранява в основна таблица с информация за съдържанието ѝ, след което чрез свързваща таблица се асоциира към конкретните потребители, за които тя е предназначена. При отваряне на страницата за нотификации от даден потребител, системата автоматично маркира всички негови съобщения със статус „pending“ като „delivered“, като записва и времето на доставяне. Всички вече доставени съобщения се визуализират подредени по дата на изпращане. При преглед на конкретна нотификация, тя допълнително се маркира със статус „seen“. По този начин системата осигурява проследимост на жизнения цикъл на всяко съобщение — от създаването му, през доставянето до окончателното му прочитане от потребителя. Освен обработката на самите нотификации, системата предоставя възможност и за зареждане на съдържанието на използваните шаблони за изпращане на email нотификации, като тези шаблони се съхраняват в отделна таблица и могат динамично да се използват при нужда.

## 3.12 Изводи към глава 3

1. В тази глава е реализирана цялостната система, обхващаща всички основни модули на платформата.

2. Разработени са отделните CRUD интерфейси, системата за роли и права, модулът за таргет групи, работа с ваучери и кампании, както и динамичното генериране на отчети чрез конфигурационни файлове.

3. Реализирана е регистрация с email верификация и captcha защита, управление на потребителски профили, възстановяване на парола и цялостен процес по създаване на поръчки с поддръжка на ваучери, промоции, плащания и известия към потребителите.

4. Системата е изградена така, че да може лесно да се разширява с нови модули и функционалности.

# ГЛАВА 4. ПРОВЕЖДАНЕ НА ТЕСТОВЕ

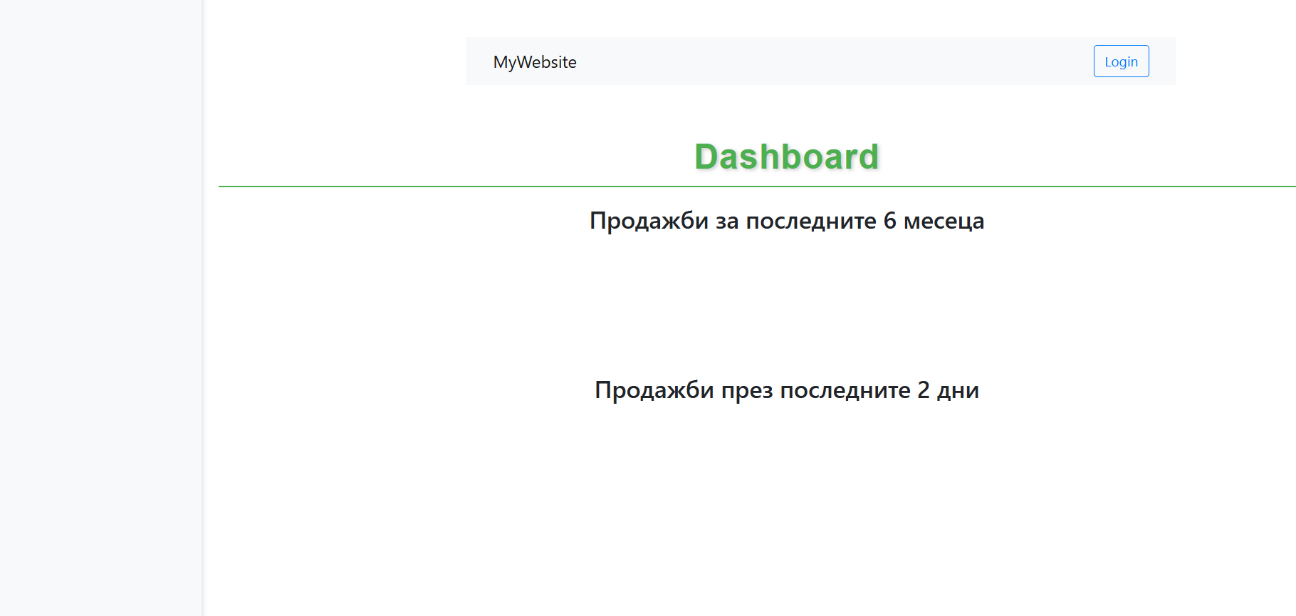
## 4.1. Системни изисквания за работа на приложението

За нормалната работа на системата е необходимо:

* Операционна система: Windows 8 или по-нова версия, macOS 10.13 (High Sierra) или актуална Linux дистрибуция със съвременни ядра.
* Процесор: Intel Core i3 или еквивалентен AMD процесор с минимална тактова честота от 2.0 GHz.
* Оперативна памет (RAM): минимум 4 GB.
* Свободно дисково пространство: поне 200 MB за инсталация, временни и работни файлове.
* Браузър: актуални версии на Google Chrome, Mozilla Firefox или Microsoft Edge.
* Интернет връзка: постоянна връзка с минимална скорост от 2 Mbps за стабилна комуникация със сървъра и зареждане на съдържанието.

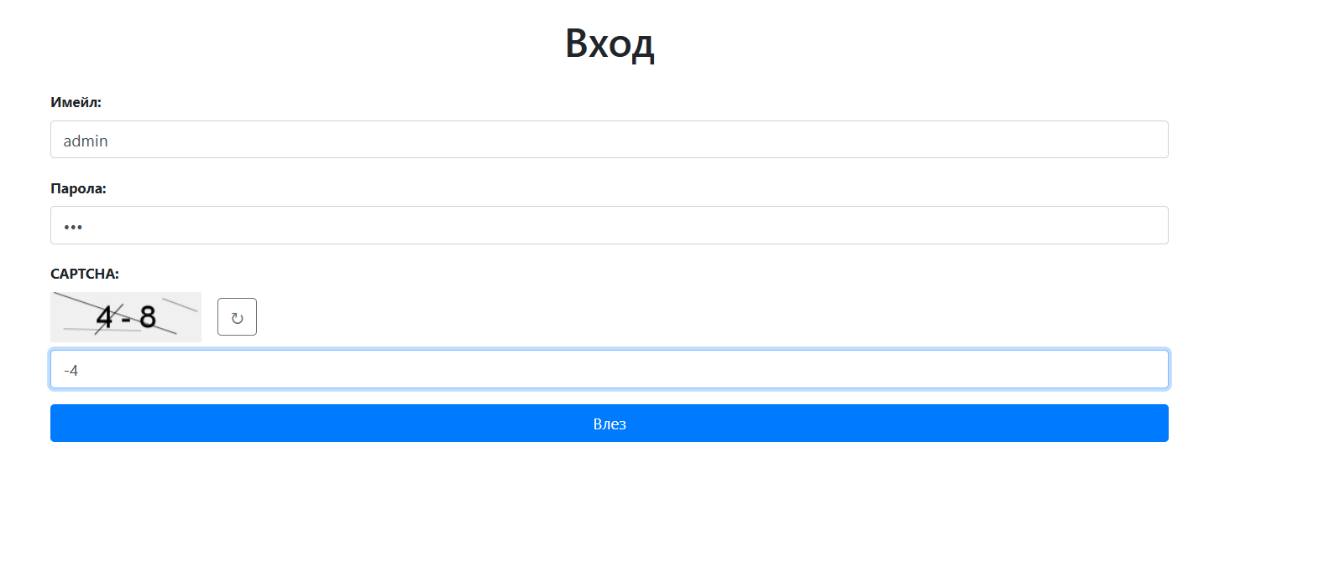
## 4.2 Резултати от тестване на приложението

### 4.2.1 Вход в административния панел (Backoffice)

****

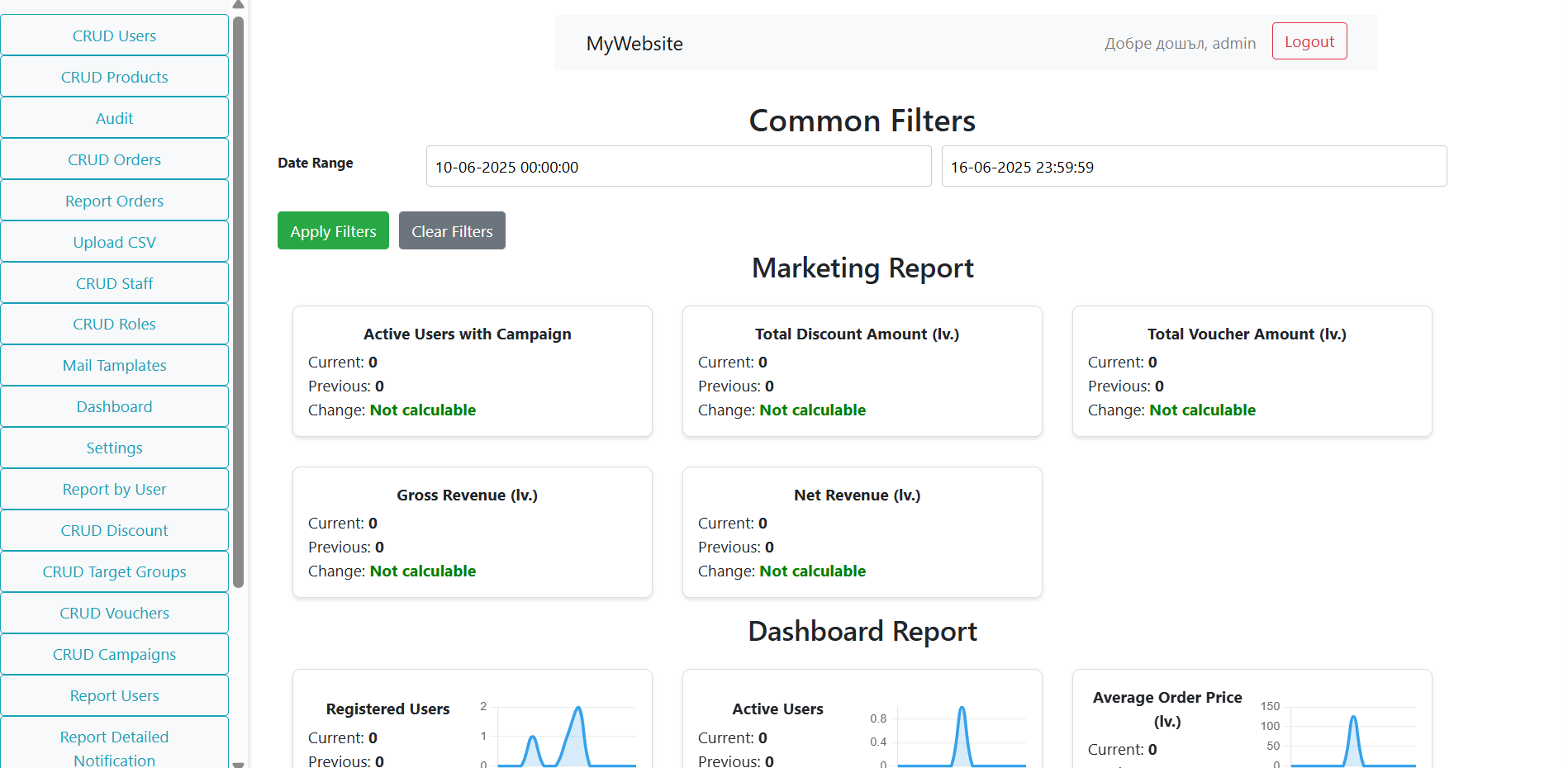
Фигура 5. Начална страница на админ панела.

Началната страница (Фигура 1) на административния панел съдържа форма за въвеждане на потребителско име (email) и парола (Фигура 2). След попълване на данните системата извършва проверка за валидност на въведената информация и статуса на акаунта. При успешно удостоверяване се зарежда основното меню с наличните функционалности, достъпни според правата на потребителя. В случай на неуспешен опит се показва съобщение с причината за отказания достъп.



Фигура 6. Вход на админ панела.

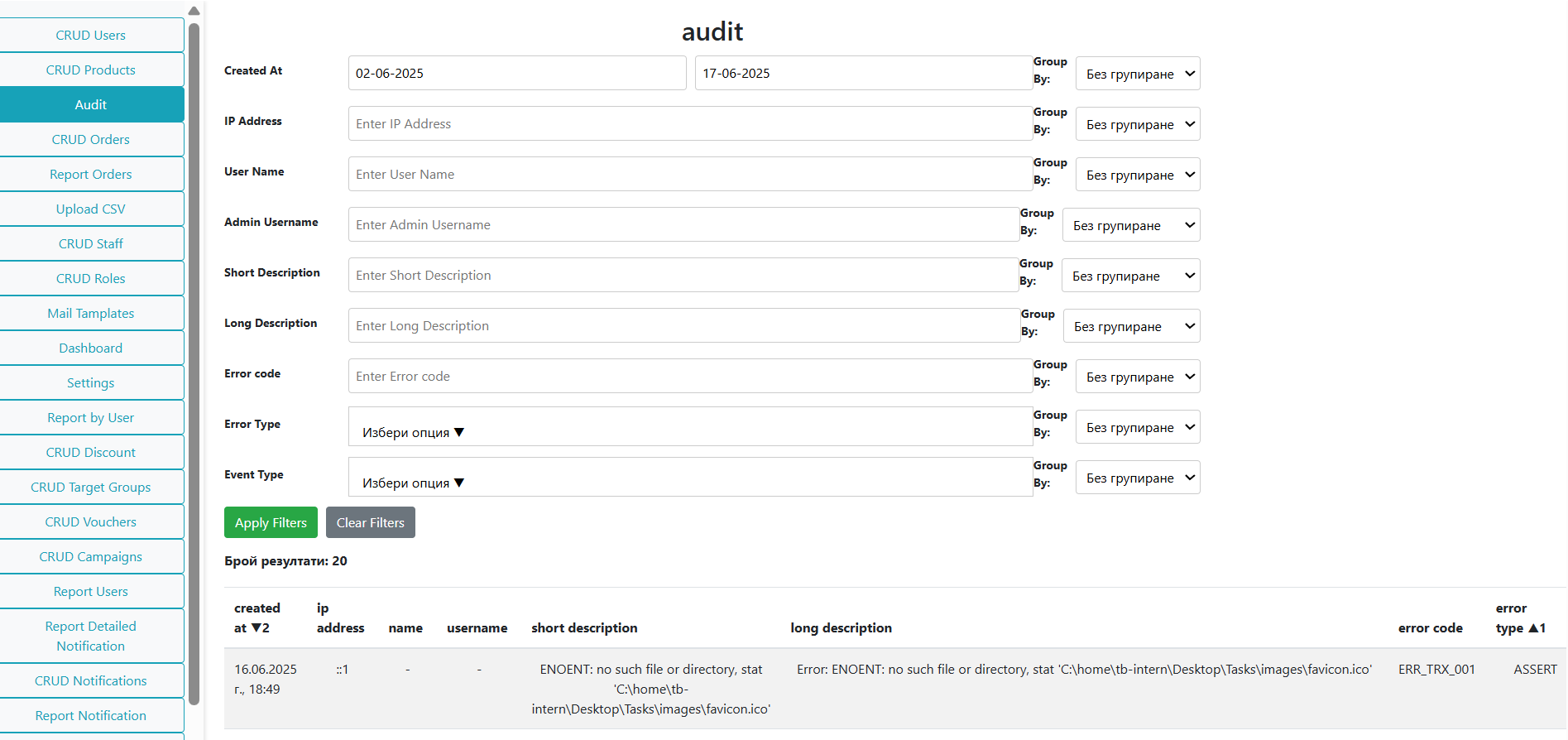
След като потребителят влезе в профила си, автоматично се зареждат начални статистики, които се генерират по подразбиране с предварително зададени филтри (Фигура 3). Тези статистики се отнасят до маркетинг секцията на приложението и включват информация като брой регистрирани потребители, активни потребители, средна стойност на поръчка, обща печалба на магазина и други ключови показатели. В лявата част на екрана се намира основното меню, от което може да се избира към кой модул или интерфейс да се премине според предоставените права на потребителя.



Фигура 7. Визуализация на първите статистики след успешен вход.

### 4.2.2 Интейфейс Audit (Репортите)

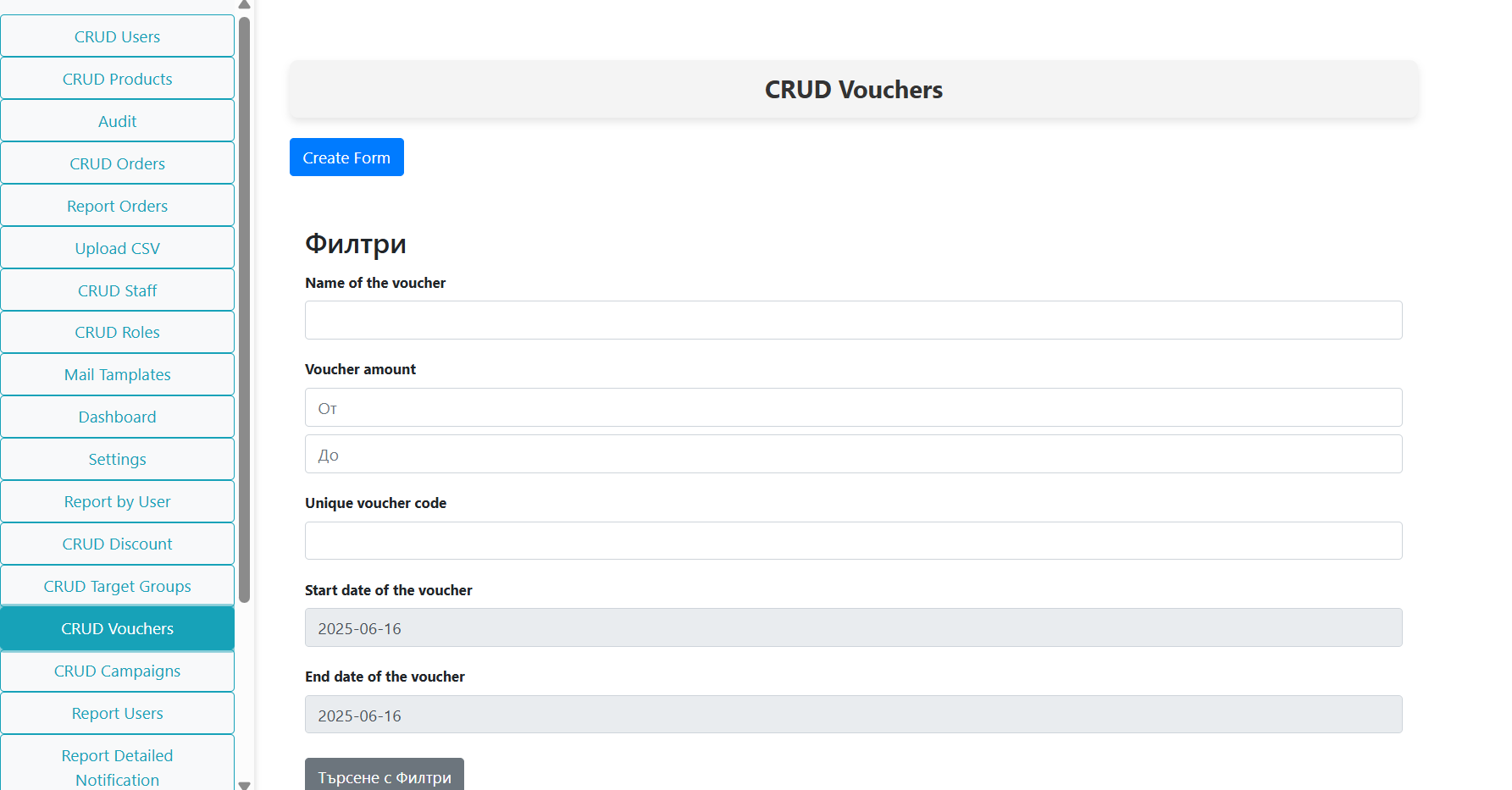
След отваряне на интерфейса за отчета Audit, системата автоматично зарежда наличните филтри, дефинирани в конфигурационния файл на репорта (Фигура 4). При самото зареждане на екрана, филтърът по създадена дата (Created At) се попълва автоматично в диапазона от днешната дата до утрешния ден. Благодарение на тази настройка веднага след отваряне на интерфейса се изпълнява автоматично търсене и потребителят вижда последно създадените записи, без да е необходимо първоначално ръчно въвеждане на филтри. След това потребителят има възможност да избере кои от останалите филтри желае да приложи или да промени вече зададените стойности, за да ограничи или разшири търсенето според нуждите си. Резултатите от търсенето се визуализират в табличен вид. Всеки един от резултатите може да бъде сортиран чрез кликване върху името на съответната колона. Системата поддържа мулти-сортиране, като при натискане на няколко различни колони се запазва реда на сортиране в зависимост от последователността на избора на колоните.



Фигура 8. Одит интерфейса(един от репортите).

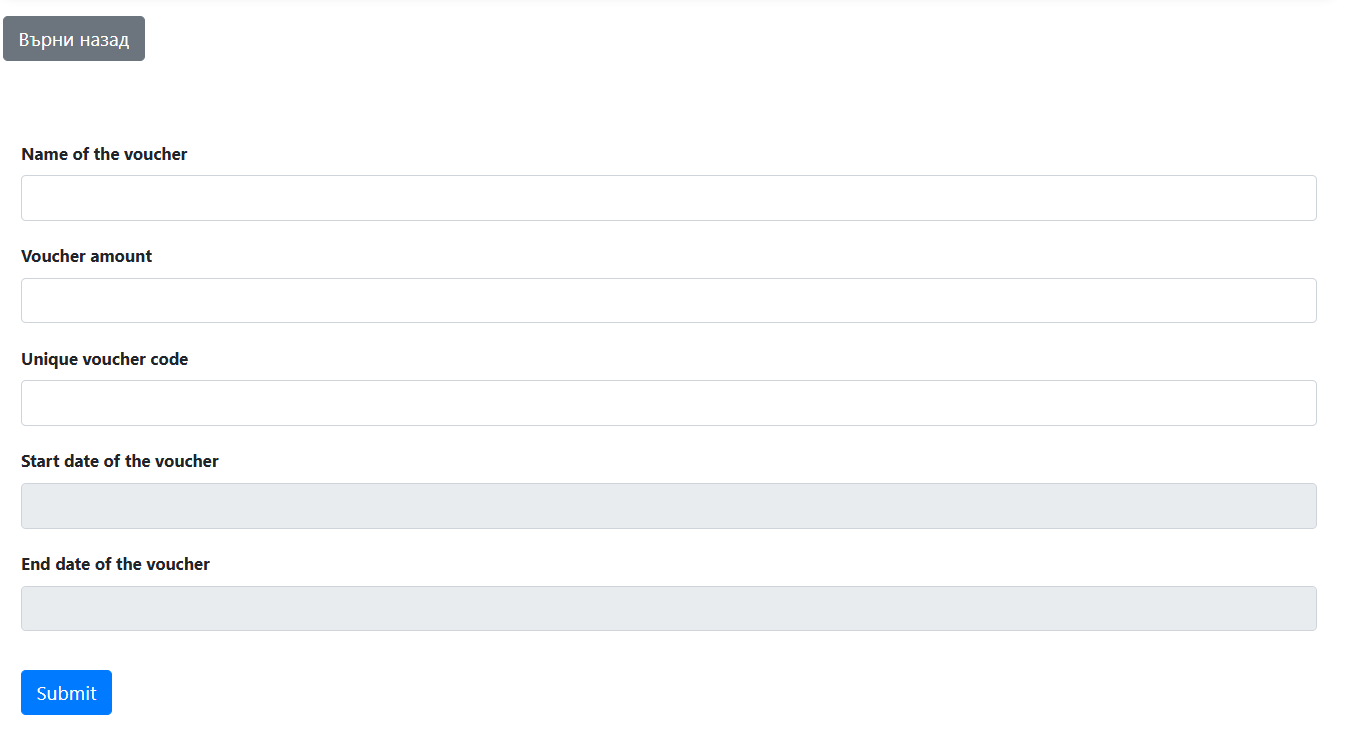
Всички останали отчети в системата работят по абсолютно идентичен начин със същия потребителски интерфейс. Единствената разлика между различните репорти се състои в наличните филтри, които се определят спрямо конфигурацията на съответния отчет. Така за всеки репорт се визуализират само релевантните полета за търсене, но логиката на зареждане, автоматичното попълване на филтри, изпълнението на заявката и визуализацията на резултатите с мулти-сортирането остава напълно еднаква.

### 4.2.3 Интейрейс Vouchers (CRUD модула)



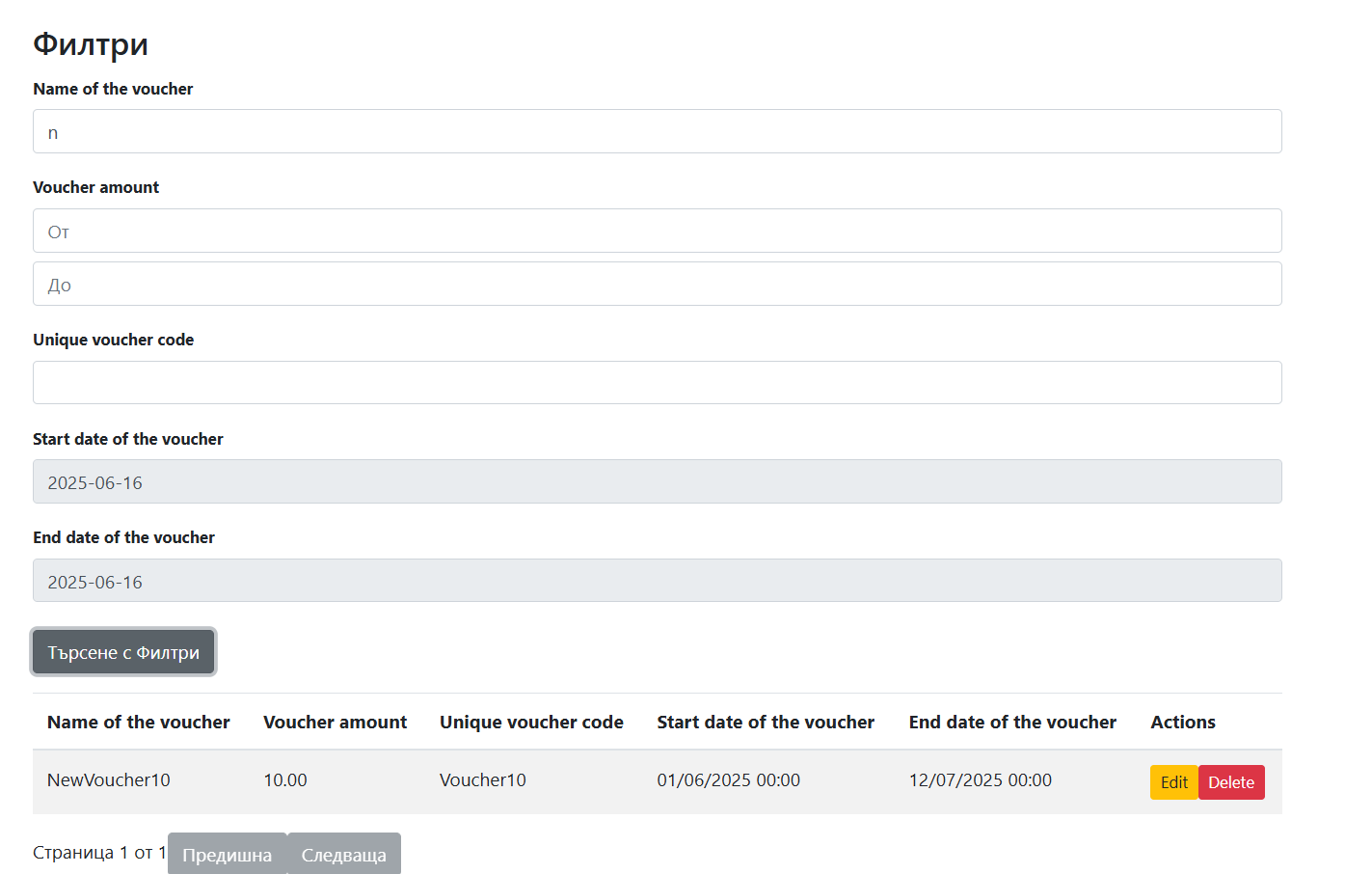
Фигура 9. Интерфейс CRUD Ваучери.

CRUD интерфейсите в системата следват единен шаблон на работа (Фигура 5). В горната част на всеки интерфейс се намира бутонът за създаване на нов запис (Create), който отваря формуляр с полетата, дефинирани в конфигурацията на съответния модул (Фигура 6). Под бутона за създаване се намира секцията с филтри, в която потребителят може да задава различни критерии за търсене. След попълване на желаните филтри, натискането на бутона за прилагане на филтрите зарежда таблица с резултатите, отговарящи на зададените условия.



Фигура 10. Create функционалността на CRUD модулите (CRUD Vouchers).

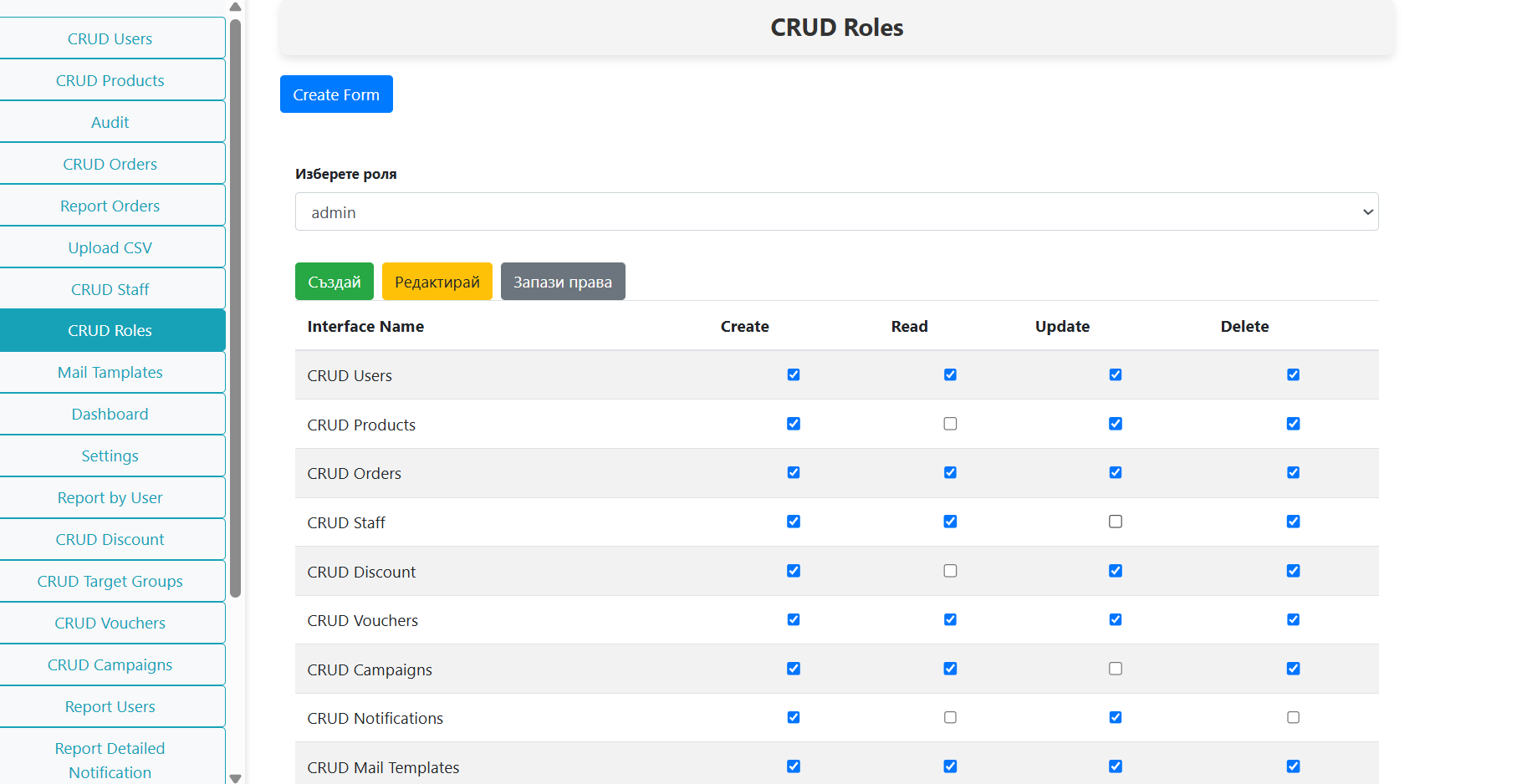
В самата таблица в края на всеки ред се намира колона **Actions**, която съдържа бутоните за редактиране (Edit) и изтриване (Delete) на съответния запис (Фигура 7). При натискане на бутона за редактиране, се отваря формуляр, в който полетата са предварително попълнени със съществуващата информация за избрания запис. Потребителят може да направи необходимите промени и да ги запамети. При натискане на бутона за изтриване, записът се премахва от системата след потвърждение.



Фигура 11. Филтри, резултат от търсене, Edit и Delete функционалности.

### 4.2.4 CRUD Roles (Роли и права)

Интерфейсът за управление на роли е реализиран като отделен CRUD модул, който не използва стандартния CRUD генератор, а има собствена логика, съобразена със специфичната структура на правата. В горната част на екрана е разположен падащ списък (дропдаун), от който потребителят избира съществуваща роля (Фигура 8). До този списък се намират бутони за създаване на нова роля или за редактиране на избраната роля.

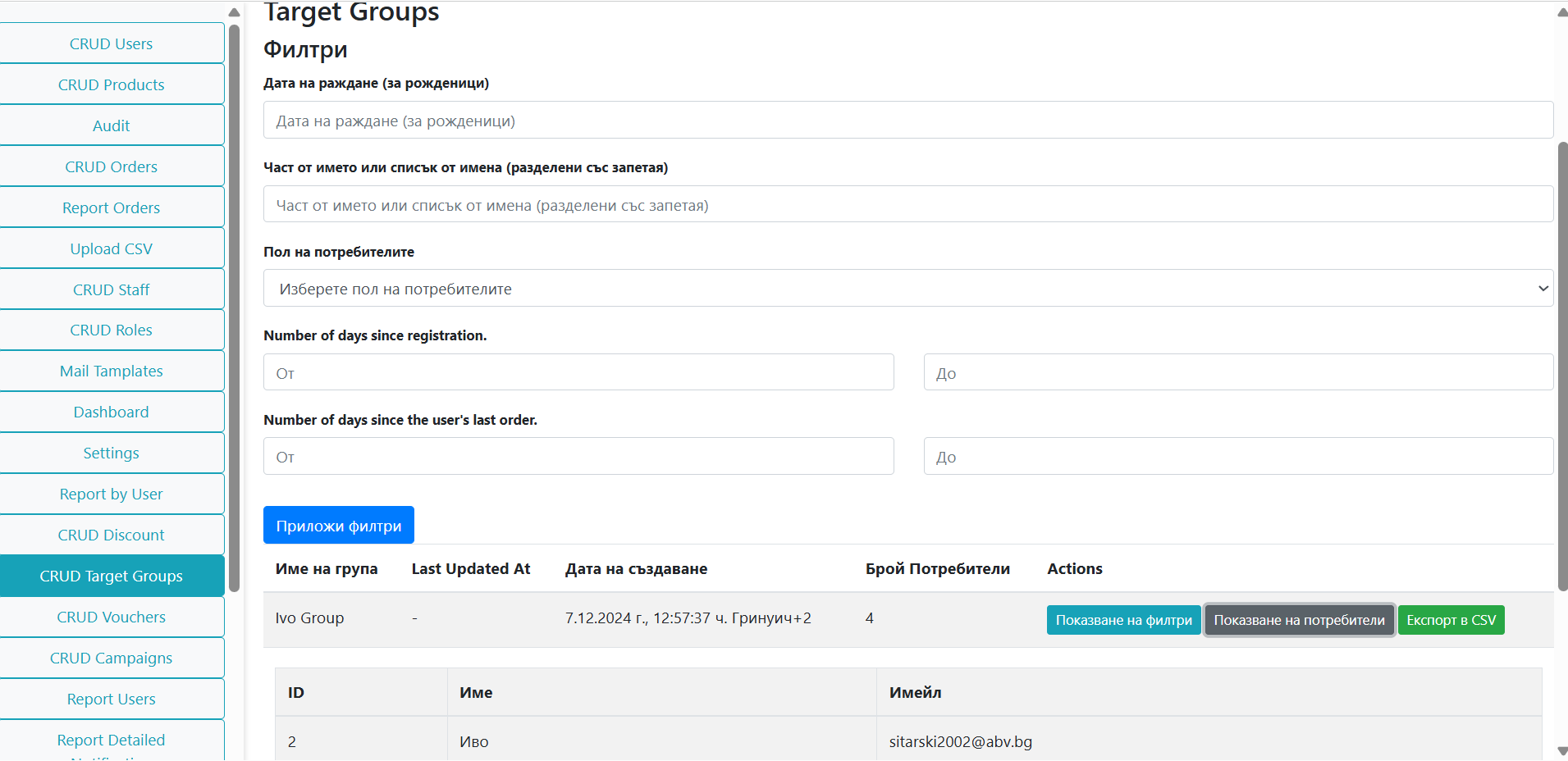


Фигура 12. CRUD Roles (Роли и права).

След избора на конкретна роля, под дропдауна се зарежда таблица, в която са изброени всички налични CRUD интерфейси в системата. Срещу всеки интерфейс са разположени четири чекбокса, отговарящи съответно на основните операции: **Create, Read, Update, Delete**. Чрез маркиране или демаркиране на съответните чекбоксове, администраторът определя какви права да има конкретната роля върху всеки отделен интерфейс. Всички промени по правата се запазват при натискане на съответния бутон за съхранение.

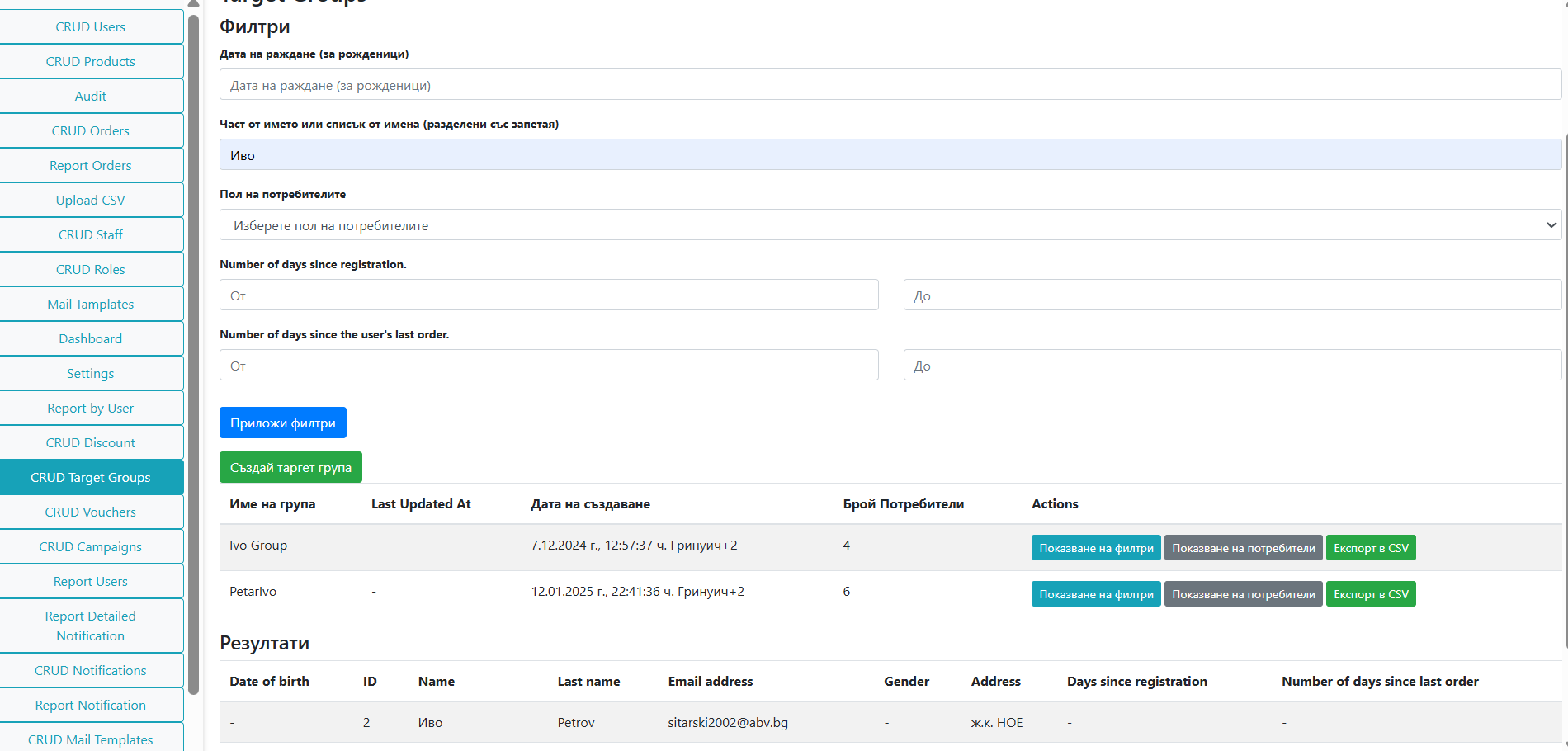
### 4.2.5 Интерфейс Target Groups (Таргет групи)

Интерфейсът за управление на Target Groups също е реализиран като отделен и специализиран CRUD модул със собствена логика, различна от стандартните CRUD форми. В долната част на екрана се зареждат вече създадените таргет групи, като за всяка от тях са налични три основни бутона — бутон за преглед на използваните при създаването ѝ филтри, бутон за визуализация на потребителите включени в групата и бутон за експорт на съдържанието в CSV файл (Фигура 9).



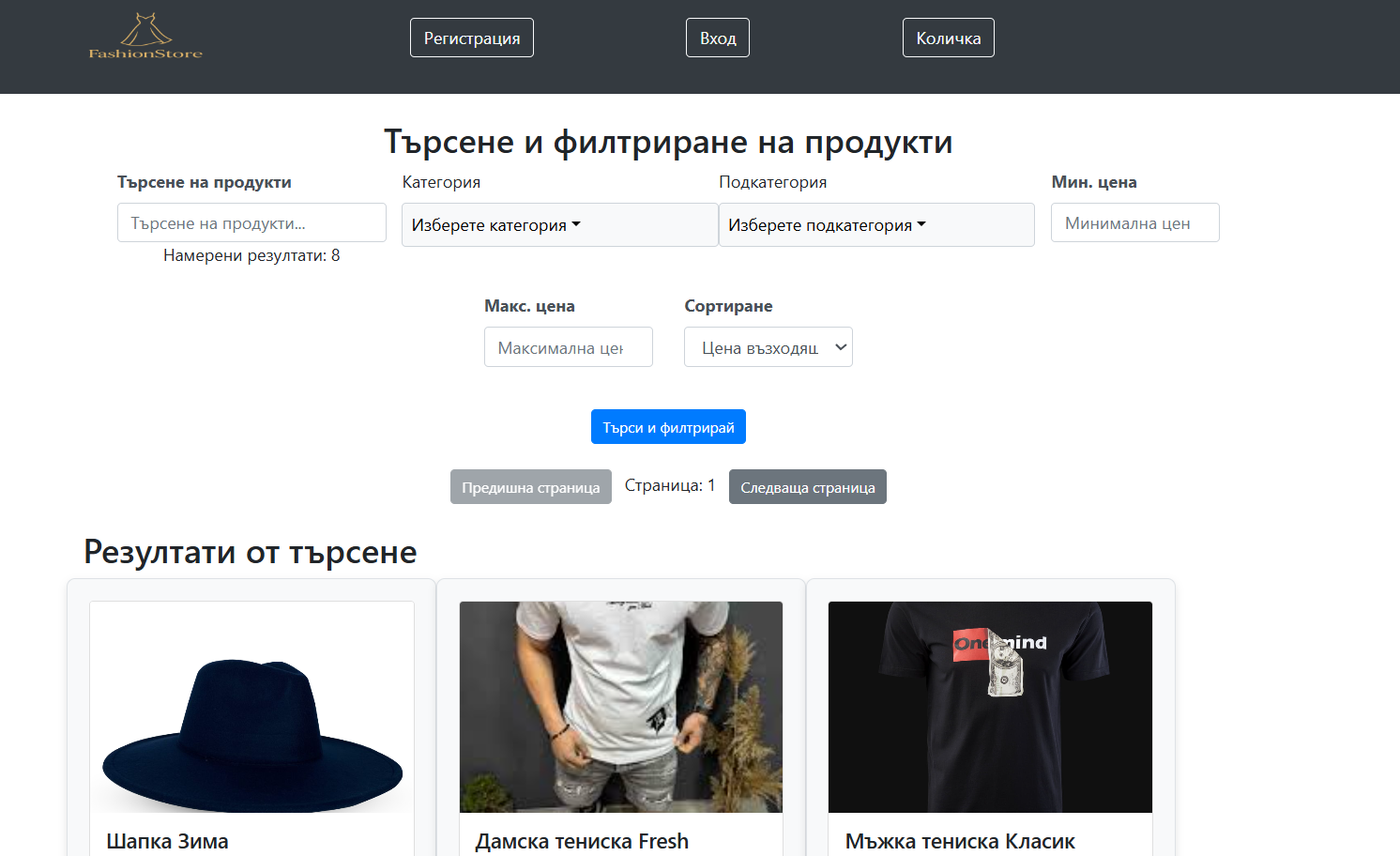
Фигура 13. Интерфейс таргет групи.

Над списъка с вече създадени групи се намира частта с филтрите. След попълването на желаните филтри и натискане на бутона за прилагането им, в долната част на интерфейса се зареждат като резултат всички потребители, които отговарят на зададените критерии. Над списъка с резултати се визуализира бутон за създаване на нова таргет група (Фигура 10). При натискането му се изисква единствено въвеждането на име на групата, след което тя автоматично се създава и включва всички потребители отговарящи на приложените към момента филтри.



Фигура 14. Резултат от търсене по филтри за създаване на таргет група.

### 4.2.6 Регистрация и вход на онлайн магазина



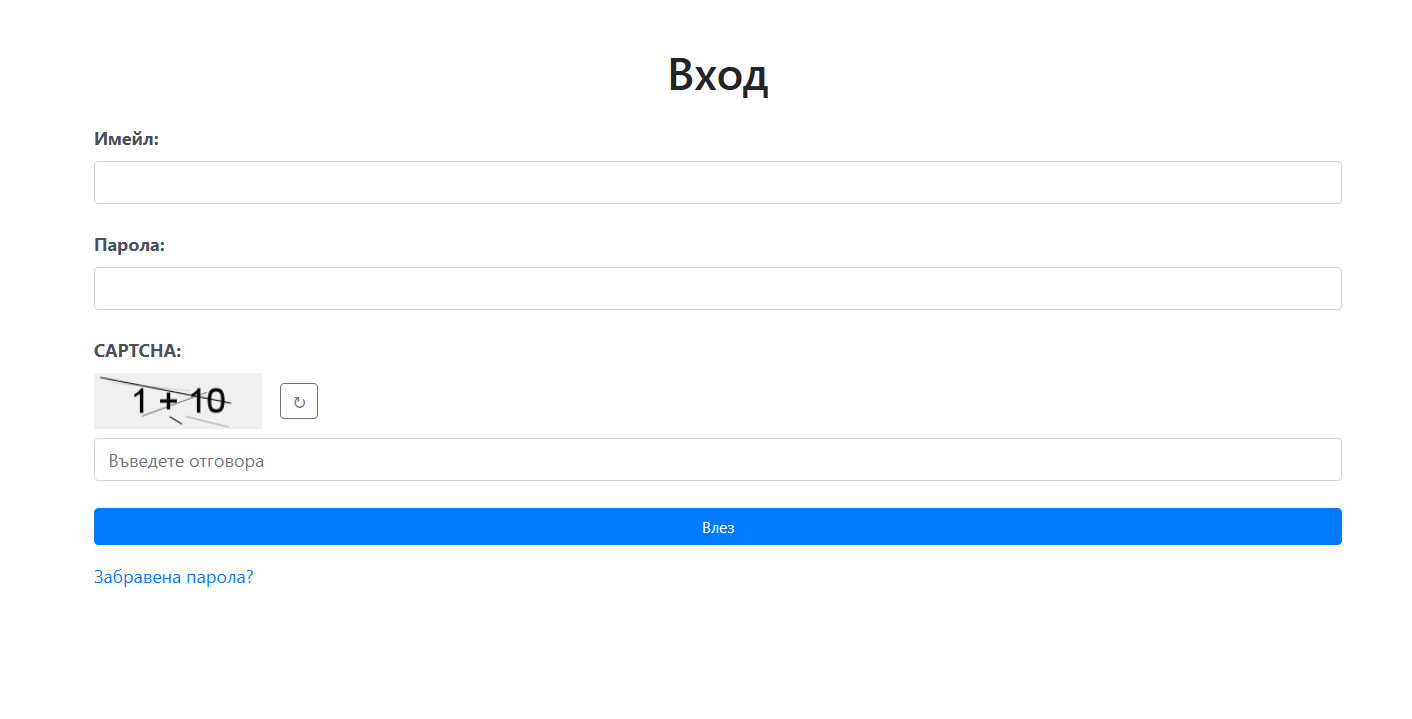
Фигура 15. Начална страница на онлайн магазина.

След зареждане на сайта на магазина на началната страница се визуализират три основни бутона – **Регистрация**, **Вход** и **Количка** (Фигура 11). Чрез бутона за регистрация се зарежда форма, в която потребителят въвежда своите данни като име, фамилия, имейл, телефон, парола, адрес и други необходими данни за създаване на профил (Фигура 12). След регистрацията се изпраща верификационен имейл, който потребителят трябва да потвърди. При натискане на бутона за вход се зарежда форма за логване, в която се въвеждат имейл, парола и captcha (Фигура 13).



Фигура 16. Форма за регистрация в онлайн магазина.

След успешно логване потребителят получава достъп до всички функционалности на системата, включително коментари, оценки и извършване на поръчки.

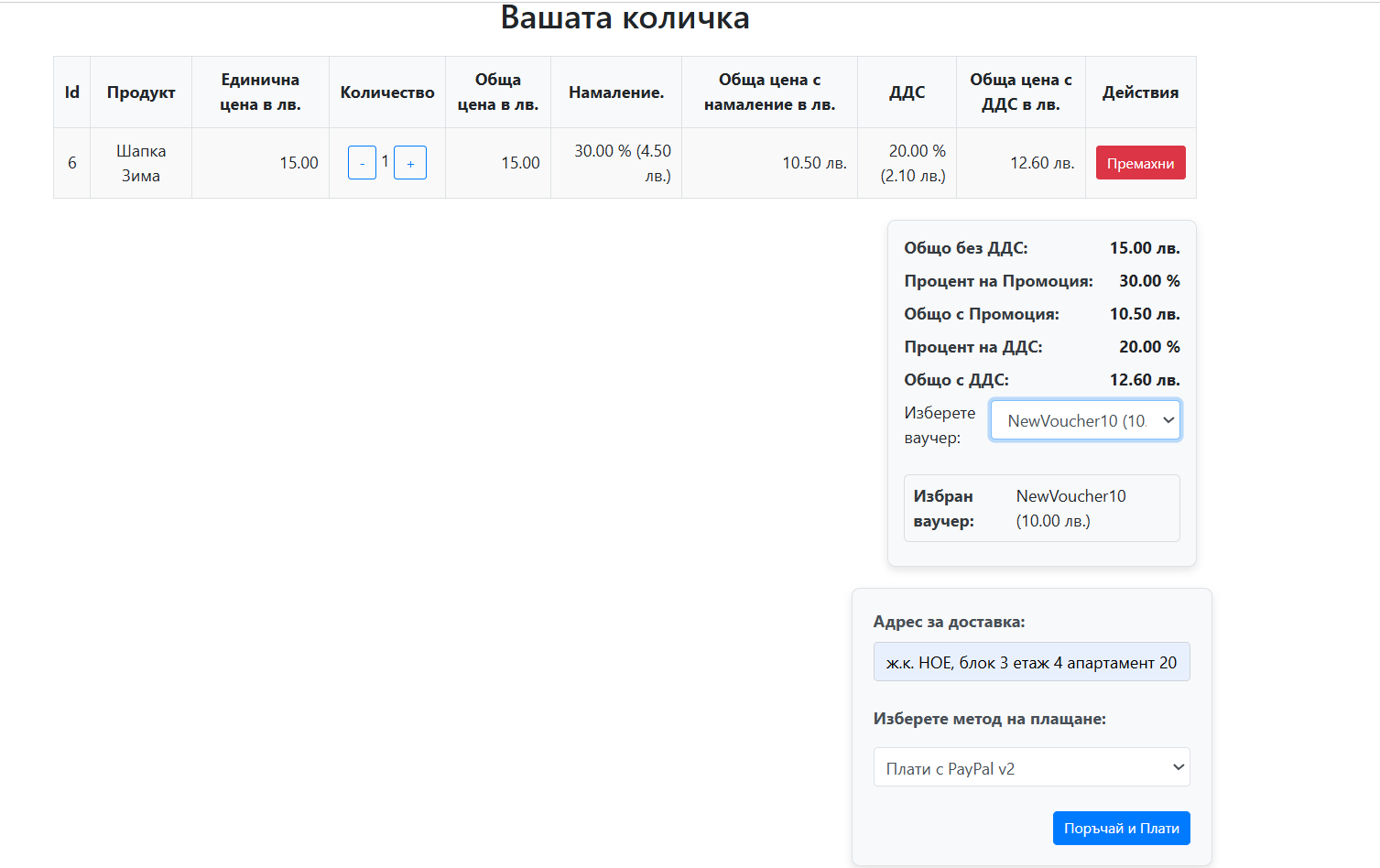


Фигура 17. Вход с captcha на онлайн магазина.

### 4.2.7 Количка за продукти от магазина

Под бутоните се зарежда основната част с наличните продукти. За всеки от тях са налични разнообразни филтри, чрез които може да се ограничат показваните резултати според категории, подкатегории, цена, промоции и други зададени критерии. След прилагането на желаните филтри системата зарежда списък с отговарящите продукти, като за всеки продукт има бутон за добавяне в количката и бутон за преглед. При натискане на бутона за преглед се зарежда детайлна страница за продукта, в която логнатите потребители могат да оставят коментари и да дават оценки. Системата поддържа оставяне на нов коментар, както и промяна на вече дадената оценка за конкретния продукт.

Количката позволява добавяне, редактиране и премахване на продукти преди финализиране на поръчката. В системата при добавяне на продукти в количката се поддържат както регистрирани потребители, така и анонимни посетители (Фигура 14). След като потребителят избере желаните продукти чрез зададените филтри, той може да ги добавя в своята количка. Количката се съхранява динамично — ако потребителят е логнат, тя се свързва с неговия акаунт, а ако не е логнат — се съхранява чрез уникален идентификатор за сесията. При логване на потребител с налична анонимна количка, двата вида колички се обединяват.

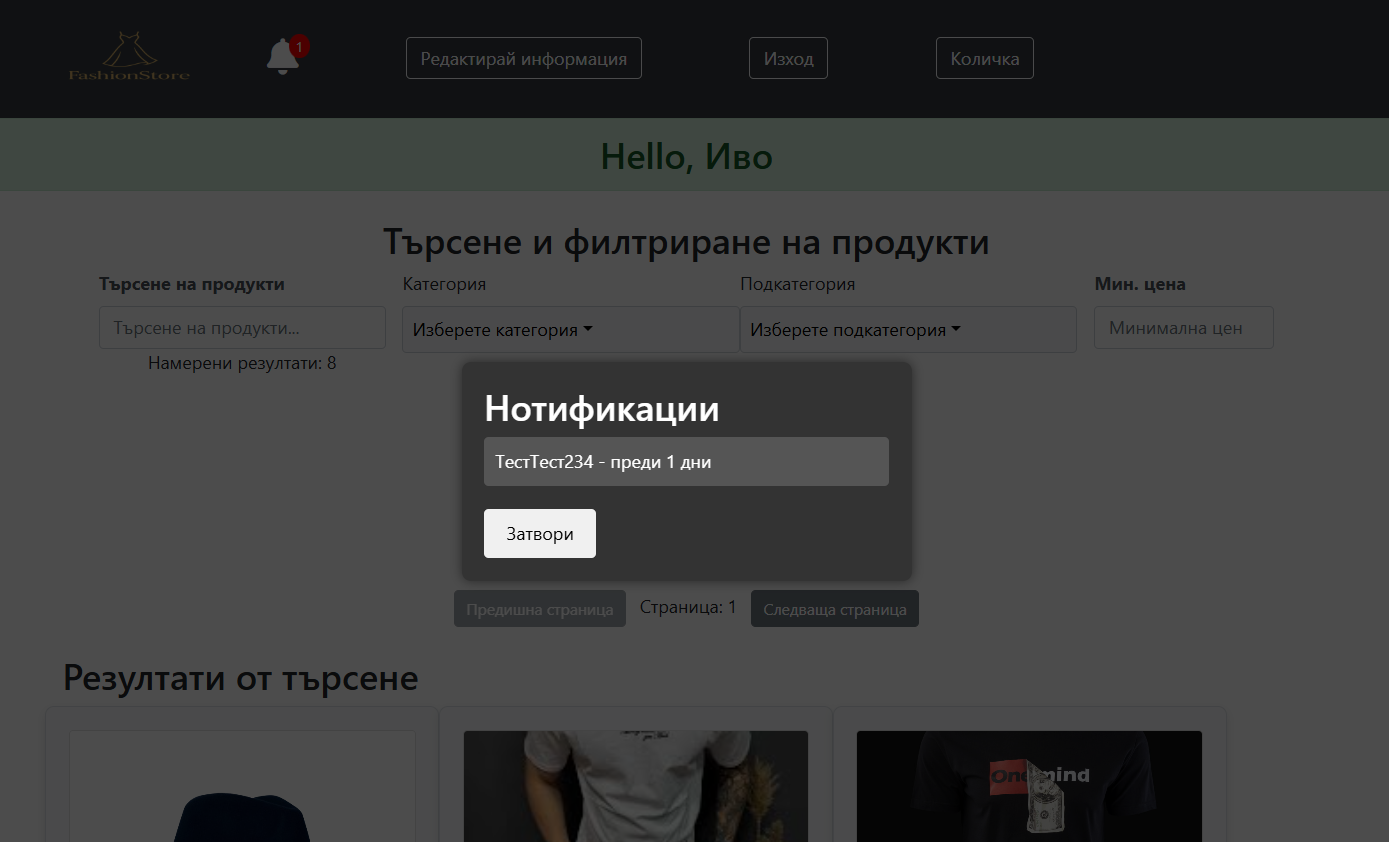


Фигура 18. Количка за избраните продукти с активен ваучер и зададена промоция 30%.

Освен стандартното добавяне и преглеждане на продукти в количката, системата предоставя възможност за използване на ваучери. Ако потребителят участва в активна кампания и е получил ваучер чрез нотификация, този ваучер автоматично става достъпен при финализиране на поръчката. Ваучерите са видими за потребителя в неговия профил, като системата следи дали ваучерът вече е използван. При наличие на валиден ваучер, крайната сума за плащане се преизчислява автоматично като се приспада съответната стойност на ваучера от общата сума.

### 4.2.8 Нотификации

Нотификациите в системата се изпращат до потребителите, като се използва механизъм за таргетиране чрез вече създадени таргет групи. При създаване на нова нотификация администраторът избира конкретна таргет група, към която да бъде изпратено съобщението. Целта на тези нотификации е да информират потребителите за различни събития, като например получен ваучер, нова промоция, потвърждение за участие в кампания и други. След изпращането всяка нотификация се запазва за съответния потребител, като той може да я види в профила си (Фигура 15). Системата поддържа проследяване на състоянието на нотификациите – нови, получени и прочетени.



Фигура 19. Нотификация на потребител, който е включен в таргет група.

### 4.2.9 Плащане

След като потребителят е избрал продуктите си, приложил е наличните ваучери и евентуални промоции, системата изчислява крайната сума за плащане. Ако след всички отстъпки крайната стойност остане по-голяма от 0, потребителят се пренасочва към PayPal, където се извършва самото плащане. При натискане на бутона за плащане се създава транзакция чрез PayPal API, като системата използва PayPal Sandbox среда за симулация. След успешното потвърждение на плащането от страна на PayPal, потребителят се връща обратно в сайта, където системата финализира поръчката и актуализира състоянието ѝ като платена.

В случай, че приложените ваучери и активните промоции намалят крайната сума до 0, системата автоматично пропуска процеса по прехвърляне към PayPal и директно финализира поръчката без да се изисква допълнително плащане от потребителя. Това осигурява гъвкавост и безпроблемно обслужване в зависимост от наличните отстъпки.

## 4.3 Изводи към глава 4

1. В тази глава е описана работата на разработената система със заложениете в нея функционалности.

2. Разгледани са процесите по регистрация, вход, управление на потребители и роли, както и динамичното създаване на отчети чрез конфигурационни файлове.

3. Обяснена е работата на таргет групите, ваучерите и нотификациите, както и процесът по добавяне на продукти в количката, използване на промоции и ваучери при поръчка и извършване на плащания през PayPal.

4. Системата коректно обработва и случаи, при които крайната стойност става нула. Проведените тестове потвърждават стабилната и правилна работа на всички модули.

# 5. ЗАКЛЮЧЕНИЕ

* + 1. Разработената система успешно реализира всички предварително заложени функционалности, обхващащи както административната, така и клиентската част на уеб- приложението. Чрез динамична конфигурационна архитектура се постигна висока гъвкавост при създаване на репорти и CRUD интерфейси, като системата позволява лесно разширяване с нови модули без необходимост от промяна в основния програмен код. Въведени бяха различни нива на валидация и верификация на данни, включително чрез Captcha механизъм, система за регистрация с имейл потвърждение и защита при възстановяване на парола.
    2. От страна на потребителите е реализирана пълна функционалност за регистрация, вход, избор и филтриране на продукти, коментари и оценки на продукти, както и управление на количка с възможност за използване на ваучери и промоции. При финализиране на поръчката потребителите могат да извършват реални или симулирани плащания чрез интеграция с PayPal. Реализирана е и допълнителна функционалност за изпращане на съобщения и известия към определени групи потребители.
    3. Цялостната архитектура на системата позволява надеждна, сигурна и гъвкава работа, като същевременно осигурява добра основа за бъдещо надграждане и внедряване на допълнителни функционалности при необходимост.
    4. Като възможна насока за развитие на приложението може да се предвиди интеграция с допълнителни разплащателни системи освен PayPal, като например Stripe или банкови карти чрез локални доставчици. Освен това, системата позволява лесно внедряване на модул за анализ на потребителското поведение, препоръки на продукти чрез алгоритми за машинно обучение, както и разширяване на ролевата система с по-фина настройка на правата за достъп. Това би допринесло за още по-висока степен на персонализация и адаптивност спрямо нуждите на бизнеса и крайните потребители.

# 6. ИЗПОЛЗВАНА ЛИТЕРАТУРА

[1] <https://www.shopify.com/>

[2] <https://business.adobe.com/products/magento/magento-commerce.html>

[3] <https://woocommerce.com/>

[4] <https://www.opencart.com/>

[5] Nielsen Norman Group - E-Commerce User Experience <https://www.nngroup.com/topic/e-commerce/>

[6] Baymard Institute - E-Commerce UX Research <https://baymard.com/research>

[7] MDN Web Docs — HTML: HyperText Markup Language <https://developer.mozilla.org/en-US/docs/Web/HTML>

[8] W3Schools - HTML Tutorial <https://www.w3schools.com/html/>

[9] MDN Web Docs — CSS: Cascading Style Sheets <https://developer.mozilla.org/en-US/docs/Web/CSS>

[10] CSS-Tricks — A Complete Guide to Flexbox <https://css-tricks.com/snippets/css/a-guide-to-flexbox/>

[11] MDN Web Docs — JavaScript <https://developer.mozilla.org/en-US/docs/Web/JavaScript>

[12] freeCodeCamp - JavaScript Full Course <https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures/>

[13] Node.js Official Documentation <https://nodejs.org/docs/latest/api/>

[14] Express.js Official Documentation <https://expressjs.com/>

[15] PostgreSQL Official Documentation <https://www.postgresql.org/docs/>

[16] DigitalOcean - PostgreSQL Tutorial <https://www.digitalocean.com/community/tutorial-series/postgresql>

[17] JSON Schema Official Website <https://json-schema.org/>

[18] AJV: Another JSON Schema Validator (официална документация) <https://ajv.js.org/>

# 7. ПРИЛОЖЕНИЕ

**Сорс код на разработената система е качен на:**

<https://github.com/IvoSitarski/webshop/tree/webstore_rc>

Акцентирано е върху отделни модули от него както следва.

**Код за сървърната логика на динамичните репорти:**

async function generateReport({ entity, searchParams }) {

const entityConfig = reportConfigurations[entity];

ASSERT(entityConfig, 'Report config not found', 'Report config not found', `Report configuration for "${entity}" not found.`, 'REPORT001');

let sql = entityConfig.sql;

const values = [];

let havingConditions = [];

const hasGroupings = searchParams.groupings && searchParams.groupings.length > 0;

entityConfig.reportFilters.forEach(filter => {

const fromValue = searchParams.filters?.[`${filter.key}\_from`];

const toValue = searchParams.filters?.[`${filter.key}\_to`];

const filterValue = searchParams.filters?.[filter.key];

if (fromValue && fromValue.startsWith("-")) {

ASSERT\_USER(false, 'Negative value is not allowed for "From".', 'Enter positive value for "From".', `The provided value for "${filter.key}\_from" is negative: ${fromValue}.`, 'REPORT002');

}

if (toValue && toValue.startsWith("-")) {

ASSERT\_USER(false, 'Negative value is not allowed for "To".', 'Enter positive value for "To".', `The provided value for "${filter.key}\_to" is negative: ${toValue}.`, 'REPORT003');

}

let filterExpression = filter.filter\_expression || '';

let havingExpression = filter.having\_expression || '';

if (filter.type === 'date-range' || filter.type === 'number-range') {

if (fromValue && toValue) {

const fromParam = `$${values.length + 1}`;

const toParam = `$${values.length + 2}`;

filterExpression = filterExpression.replace('$FILTER\_VALUE$\_from', fromParam).replace('$FILTER\_VALUE$\_to', toParam);

havingExpression = havingExpression.replace('$FILTER\_VALUE$\_from', fromParam).replace('$FILTER\_VALUE$\_to', toParam);

values.push(fromValue, toValue);

sql = sql.replaceAll(`$${filter.key}\_from$`, fromParam)

.replaceAll(`$${filter.key}\_to$`, toParam);

} else if (fromValue) {

const fromParam = `$${values.length + 1}`;

filterExpression = `AND ${filter.key} >= ${fromParam}`;

if (filter.having\_expression) {

havingExpression = 'AND ' + filter.having\_expression

.replace('BETWEEN $FILTER\_VALUE$\_from AND $FILTER\_VALUE$\_to', `>= ${fromParam}`)

.replace('$FILTER\_VALUE$\_from', fromParam);

}

values.push(fromValue);

} else if (toValue) {

const toParam = `$${values.length + 1}`;

filterExpression = `AND ${filter.key} <= ${toParam}`;

if (filter.having\_expression) {

havingExpression = 'AND ' + filter.having\_expression

.replace('BETWEEN $FILTER\_VALUE$\_from AND $FILTER\_VALUE$\_to', `<= ${toParam}`)

.replace('$FILTER\_VALUE$\_to', toParam);

}

values.push(toValue);

} else {

filterExpression = '';

havingExpression = '';

}

} else {

if (filterValue) {

const param = `$${values.length + 1}`;

filterExpression = filterExpression.replace('$FILTER\_VALUE$', param);

havingExpression = havingExpression.replace('$FILTER\_VALUE$', param);

values.push(filterValue);

} else {

filterExpression = '';

havingExpression = '';

}

}

sql = sql.replace(`$${filter.key}\_filter$`, filterExpression.trim());

sql = sql.replace(`$${filter.key}\_having$`, havingExpression.trim());

if (havingExpression) {

havingConditions.push(havingExpression.trim());

}

const columnPlaceholder = `$${filter.key}\_column$`;

if (hasGroupings) {

const selectedGrouping = searchParams.groupings.find(grouping =>

filter.groupingMap && filter.groupingMap[grouping]

);

if (selectedGrouping) {

sql = sql.replaceAll(columnPlaceholder, filter.groupingMap[selectedGrouping]);

} else {

sql = sql.replaceAll(columnPlaceholder, "'-'");

}

} else {

sql = sql.replaceAll(columnPlaceholder, `${filter.key}`);

}

});

if (sql.includes("'-' DESC")) {

sql = sql.replace(`ORDER BY '-' DESC`, '');

}

if (searchParams.sort && Array.isArray(searchParams.sort) && searchParams.sort.length > 0) {

const orderByClauses = searchParams.sort.map(sortItem => {

return `${sortItem.key} ${sortItem.order.toUpperCase()} NULLS LAST`;

}).join(', ');

sql = sql.replace(/\bORDER BY\s+\w+\s+(ASC NULLS LAST|DESC NULLS LAST)\b/i, `ORDER BY ${orderByClauses}`);

}

let limit;

const reportLimitResult = await dbClient.query('SELECT report\_limit FROM settings LIMIT 1');

if (reportLimitResult.rows.length > 0 && reportLimitResult.rows[0].report\_limit) {

limit = reportLimitResult.rows[0].report\_limit;

} else {

limit = '1000';

}

sql = sql.replace('$limit$', limit);

const result = await dbClient.query(sql, values);

return result.rows.map(row => {

for (const key in row) {

if (row[key] === null) row[key] = '-';

}

return row;

});

}

**Код за сървърната логика на динамичния CRUD модул :**

async function getEntitiesByFilters(schema, filters, limit = 100, offset = 0, randomizePage = false) {

if (randomizePage) {

const randomPage = Math.floor(Math.random() \* 10) + 1;

offset = (randomPage - 1) \* limit; }

const conditions = [];

const values = [];

const viewName = `${schema.tableName}\_view`;

Object.keys(filters).forEach((key) => {

const cleanKey = key.replace(/\_from|\_to$/, '');

const property = schema.properties[cleanKey];

if (property) {

if (property.format === 'date-time' || property.format === 'date') {

const fromKey = `${cleanKey}\_from`;

const toKey = `${cleanKey}\_to`;

if (filters[fromKey] && filters[toKey]) {

conditions.push(`${cleanKey} BETWEEN $${values.length + 1} AND $${values.length + 2}`);

values.push(filters[fromKey], filters[toKey]);

} else if (filters[fromKey]) {

conditions.push(`${cleanKey} >= $${values.length + 1}`);

values.push(filters[fromKey]);

} else if (filters[toKey]) {

conditions.push(`${cleanKey} <= $${values.length + 1}`);

values.push(filters[toKey]);

}

} else if (property.type === 'string' && filters[key]) {

conditions.push(`${cleanKey} ILIKE $${values.length + 1}`);

values.push(`%${filters[key]}%`);

} else if (property.type === 'number' || property.type === 'integer') {

if (property.type === 'integer' && filters[key]) {

conditions.push(`${cleanKey} = $${values.length + 1}`);

values.push(parseInt(filters[key], 10));

} else {

const fromKey = `${cleanKey}\_from`;

const toKey = `${cleanKey}\_to`;

if (filters[fromKey] && filters[toKey]) {

conditions.push(`${cleanKey} BETWEEN $${values.length + 1} AND $${values.length + 2}`);

values.push(parseFloat(filters[fromKey]), parseFloat(filters[toKey]));

} else if (filters[fromKey]) {

conditions.push(`${cleanKey} >= $${values.length + 1}`);

values.push(parseFloat(filters[fromKey]));

} else if (filters[toKey]) {

conditions.push(`${cleanKey} <= $${values.length + 1}`);

values.push(parseFloat(filters[toKey]));

}

}

}

}

});

const whereClause = conditions.length > 0 ? `WHERE ${conditions.join(' AND ')}` : '';

const sqlQuery = `

WITH filtered\_view AS (

SELECT \*

FROM ${viewName}

${whereClause}

)

SELECT \*

FROM filtered\_view

ORDER BY id ASC

LIMIT $${values.length + 1} OFFSET $${values.length + 2}

`;

const dataValues = [...values, limit, offset];

const result = await dbClient.query(sqlQuery, dataValues);

const countQuery = `

WITH filtered\_view AS (

SELECT \*

FROM ${viewName}

${whereClause}

)

SELECT count(\*) AS total\_count

FROM filtered\_view

`;

const countResult = await dbClient.query(countQuery, values);

const totalCount = parseInt(countResult.rows[0].total\_count, 10);

const totalPages = Math.ceil(totalCount / limit);

return {

rows: result.rows,

totalItems: totalCount,

totalPages,

};

}

async function deleteEntity(entityId, schema) {

const tableName = schema.tableName;

if (tableName === 'admin\_users') {

const result = await dbClient.query(`SELECT COUNT(\*) AS count FROM admin\_user\_roles WHERE admin\_user\_id = $1`, [entityId]);

ASSERT(parseInt(result.rows[0].count, 10) > 0, 'Cannot delete user with assigned roles.', 'Delete User Error', 'This user has roles assigned and cannot be deleted.', 'USER\_HAS\_ROLES');

await dbClient.query(`DELETE FROM admin\_user\_roles WHERE admin\_user\_id = $1`, [entityId]);

}

if (tableName === 'roles') {

await dbClient.query(`DELETE FROM role\_permissions WHERE role\_id = $1`, [entityId]);

}

try {

const testResult = await dbClient.query('SELECT 1');

} catch (error) {

console.error('Database connection failed:', error);

throw error;

}

const deleteResult = await dbClient.query(

`DELETE FROM ${tableName} WHERE id = $1 RETURNING \*`,

[entityId]

);

return { message: `${tableName} успешно изтрит` };

}

async function getEntityById(id, schema) {

const viewName = `${schema.tableName}\_view`;

const query = `

SELECT \* FROM ${viewName}

WHERE id = $1

`;

const values = [id];

const result = await dbClient.query(query, values);

if (result.rows.length === 0) {

return null;

}

return result.rows[0];

}