# Почивни бази ТУ-СофияТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ

## ФАКУЛТЕТ ПО КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ

**ДИПЛОМНА РАБОТА**

Тема: **„Онлайн магазин за дрехи с административен панел 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

[ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ 1](#_Toc207909676)

[1.УВОД 5](#_Toc207909680)

[1.1. Цели и обхват на разработката 5](#_Toc207909681)

[2.АНАЛИЗ НА ИЗИСКВАНИЯТА 7](#_Toc207909682)

[2.1. Функционални изисквания 7](#_Toc207909683)

[2.2. Нефункционални изисквания 9](#_Toc207909684)

[2.3. Потребителски роли и сценарии на употреба 9](#_Toc207909685)

[3. СРАВНИТЕЛЕН АНАЛИЗ НА СЪЩЕСТУВАЩИ РЕШЕНИЯ 11](#_Toc207909686)

[3.1. Увод 11](#_Toc207909687)

[3.2. Избрани за сравнение платформи 11](#_Toc207909688)

[3.3. Сравнение по функционалности 13](#_Toc207909689)

[3.4. Анализ на предимствата на нашето решение 15](#_Toc207909690)

[4. ИЗПОЛЗВАНИ ТЕХНОЛОГИИ И ИНСТРУМЕНТИ 16](#_Toc207909691)

[4.1. Backend: Python/Django (MVT архитектура) 16](#_Toc207909692)

[4.2. Frontend: HTML5/CSS3, JavaScript (Fetch API / AJAX), модали и динамични UI елементи 17](#_Toc207909710)

[4.3. База данни: PostgreSQL 19](#_Toc207909711)

[4.4. Плащания: Stripe API 20](#_Toc207909712)

[4.5. Имейл услуги: Gmail SMTP 21](#_Toc207909713)

[4.6. Dev инструменти: VS Code, Git 21](#_Toc207909714)

[5. АРХИТЕКТУРА НА СИСТЕМАТА И РЕАЛИЗАЦИЯ 23](#_Toc207909715)

[5.1. Общ преглед на архитектурата 23](#_Toc207909716)

[5.2. Основни модели и техните връзки (ERD) 24](#_Toc207909717)

[5.2.1. Основни модели 25](#_Toc207909718)

[5.2.2. Връзки между моделите 26](#_Toc207909719)

[5.3. Функционални модули 27](#_Toc207909720)

[5.3.1. Начална страница 27](#_Toc207909721)

[5.3.2. Регистрация и вход + възстановяване на парола 28](#_Toc207909722)

[5.3.3. Каталог и продукт детайл 30](#_Toc207909723)

[5.3.4. Пазарска количка 31](#_Toc207909724)

[5.3.5. Списък с желания 32](#_Toc207909725)

[5.3.6. GET THE LOOK 33](#_Toc207909726)

[5.3.7. Поръчка и Checkout 34](#_Toc207909727)

[5.3.8. Плащане със Stripe 35](#_Toc207909728)

[5.3.9. Ревюта 38](#_Toc207909729)

[5.3.10. Нотификации 39](#_Toc207909730)

[5.3.11. Админ панел 40](#_Toc207909731)

[5.3.12. Сигурност 41](#_Toc207909732)

[6. ТЕСТВАНЕ И ОЦЕНКА 42](#_Toc207909733)

[6.1. Подход и среда за тестване 42](#_Toc207909734)

[6.2. Функционални тестове 43](#_Toc207909735)

[6.2.1. Потребители и автентикация 43](#_Toc207909736)

[6.2.2 Пазарска количка (Cart) 43](#_Toc207909737)

[6.2.3. Списък с желания (Wishlist) 44](#_Toc207909738)

[6.2.4. Модул „GET THE LOOK“ 45](#_Toc207909739)

[6.2.5. Поръчки и плащания 45](#_Toc207909740)

[6.2.6. Ревюта 46](#_Toc207909741)

[6.3. Резултати и оценка на системата 46](#_Toc207909742)

[7. ЗАКЛЮЧЕНИЕ 48](#_Toc207909743)

[8. ИЗПОЛЗВАНА ЛИТЕРАТУРА И ИЗТОЧНИЦИ 49](#_Toc207909744)

[8.1. Приложение 50](#_Toc207909745)

[8.1.1. Приложение A: База данни – таблици и модели 51](#_Toc207909746)

[8.1.2. Приложение Б: Извадки от сорс кода 53](#_Toc207909747)

# 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 [1], [3].

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. СРАВНИТЕЛЕН АНАЛИЗ НА СЪЩЕСТУВАЩИ РЕШЕНИЯ

## 3.1. Увод

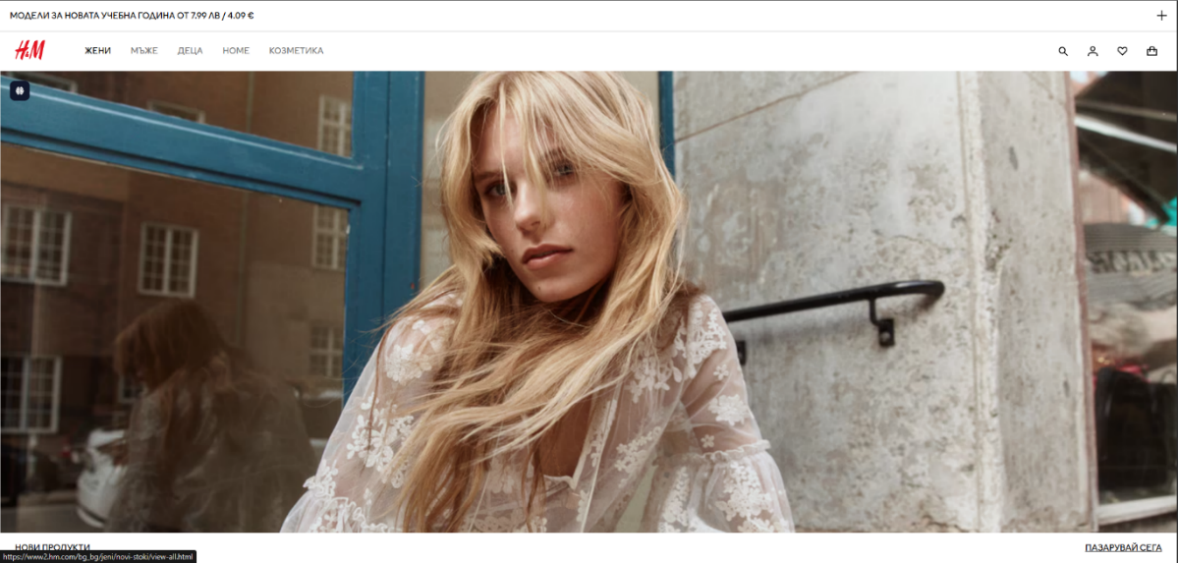
В тази глава се извършва сравнение между реализирания в проекта онлайн магазин и няколко популярни платформи за електронна търговия в сферата на модата. Целта е да се анализират прилики и разлики по отношение на функционалности, потребителско изживяване, технологии и подходи за представяне на продуктите.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

## 3.2. 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 [11].

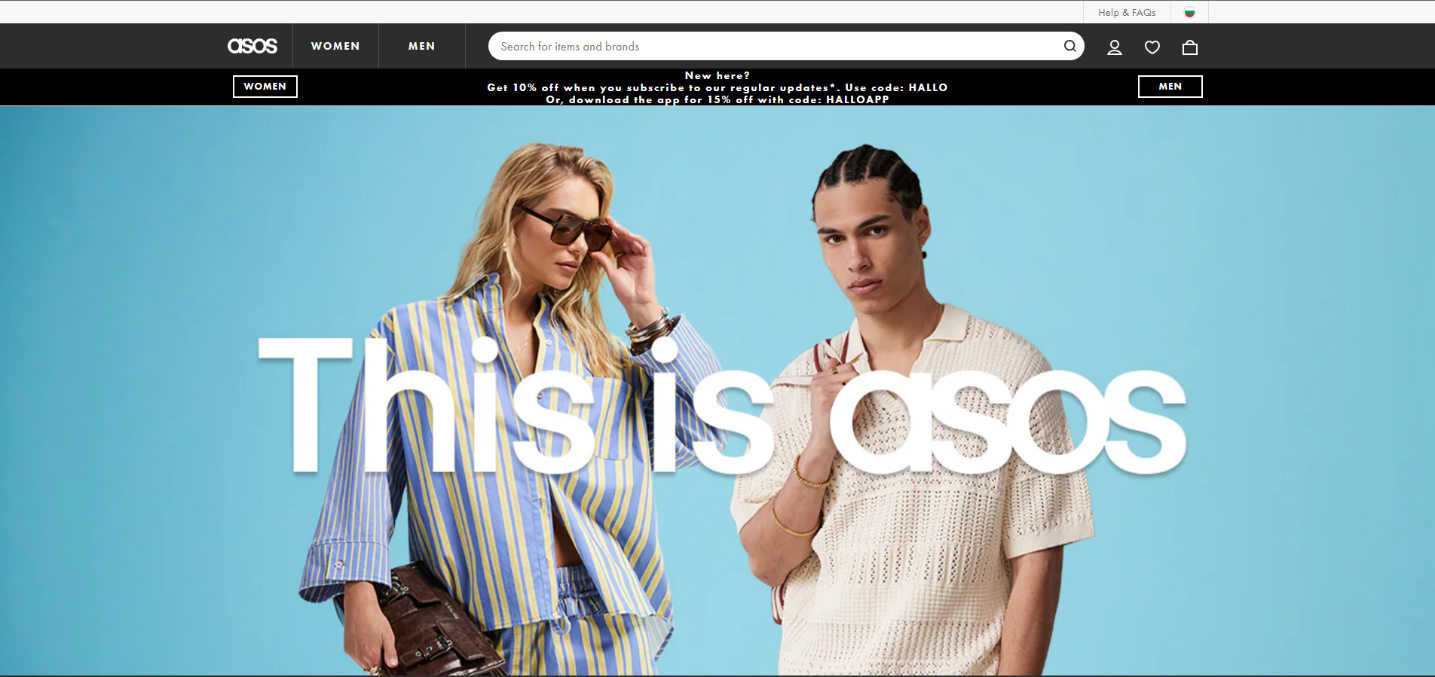
**Фигура 1: Сайтът на Zara**

**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 [12].

**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 [13].

**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 [14].

**Фигура 4: Сайтът на ASOS**

## 3.3. 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).**

## 3.4. 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:

**3.4.1. 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.

**3.4.2. 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.

**3.4.3. 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.

**3.4.4. 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.

**3.4.5. 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.

**3.4.6. 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.

# 4. 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.

## 4.1. 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 [1].

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 [8].

Django е един от водещите уеб фреймуърци за Python и е предпочитан за създаване на мащабируеми и сигурни уеб приложения. Основните му предимства включват [1]:

* **„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“). ORM автоматизира работата с базата и елиминира нуждата от директно писане на SQL заявки. Основните операции се осъществяват чрез CRUD подхода (Create, Read, Update, Delete), който позволява добавяне, извличане, редактиране и изтриване на данни по стандартизиран и защитен начин.
* **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.

## 4.2. 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 [5].  
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 [5].  
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 [4], [5].  
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.

## 4.3. 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 [2].

**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.

## 4.4. 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 [3].

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.

## 4.5. 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 [7], [10].

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.

## 4.6. 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 [6].

**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.

# 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 [1].

**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“).

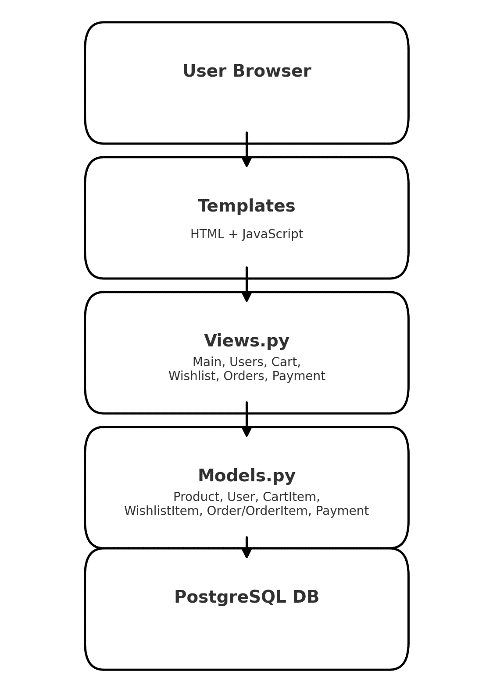
**View** – обработва заявките, получени от клиента, изпълнява необходимата бизнес логика и връща отговор към потребителя. В нашия проект са реализирани както стандартни Django view функции, така и AJAX endpoints за асинхронно обновяване на съдържанието без презареждане на страницата.

**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[4], [5].

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

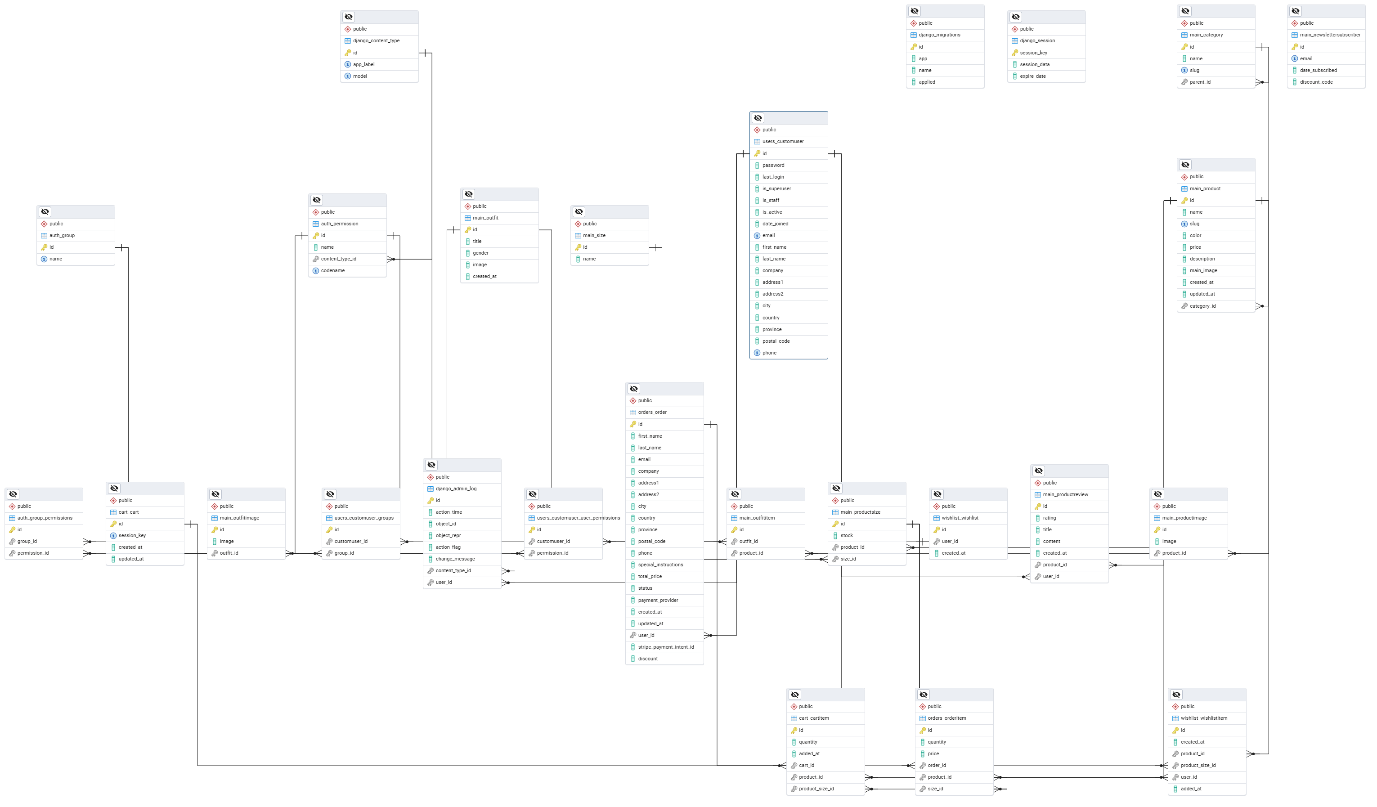
* users – управление на регистрация, вход, профили и възстановяване на парола.
* main – начална страница и общи изгледи.
* products – каталог, детайлна страница за продукт, размери и изображения.
* cart – пазарска количка и динамично управление на продуктите.
* wishlist – списък с желания.
* orders – 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 [2], [8].

**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

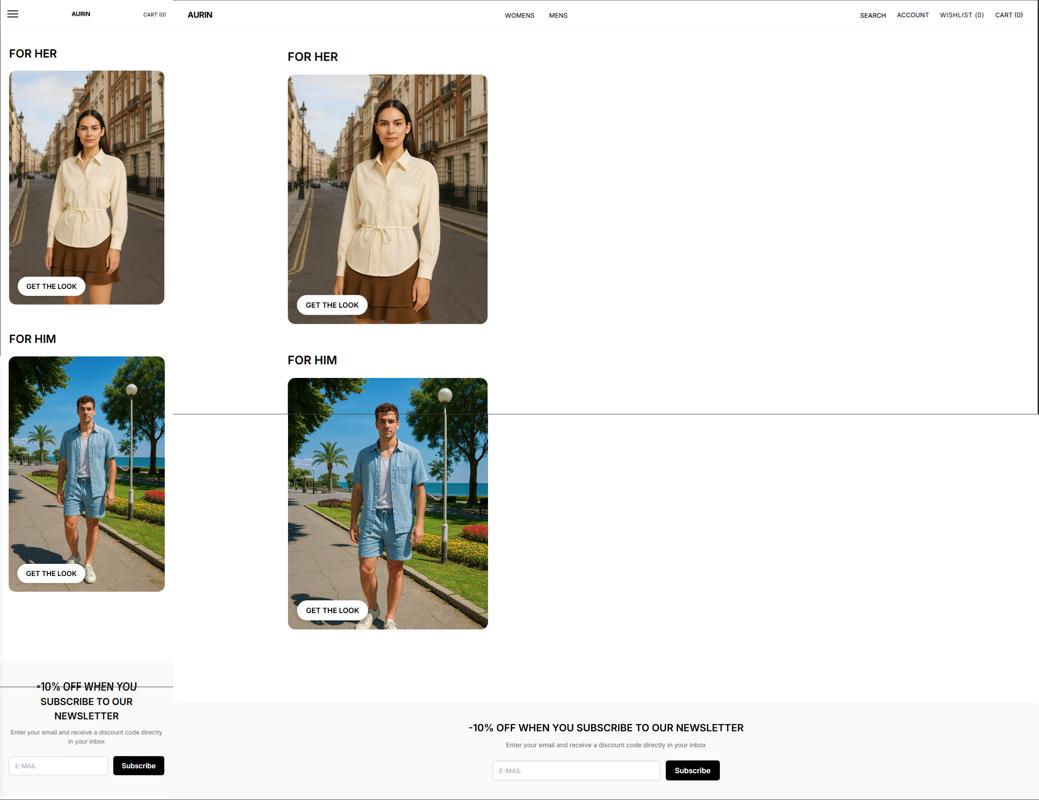
**5.3.1. Начална страница**

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).**

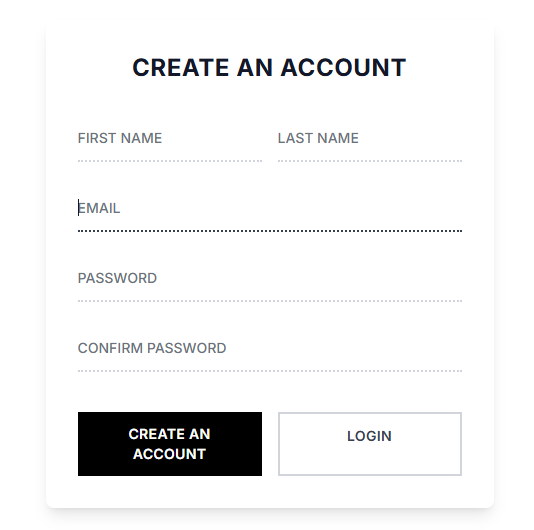
**5.3.2. Регистрация и вход + възстановяване на парола**

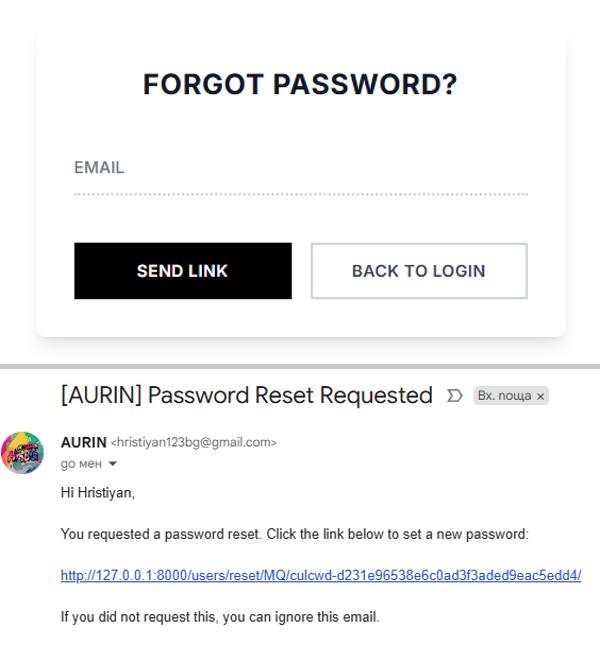
Достъпът до функциите за вход, регистрация и възстановяване на парола е реализиран чрез бутона **„ACCOUNT“** в хедъра на сайта. При натискане на бутона се отваря **модален прозорец** за вход в системата. Той съдържа поле за въвеждане на имейл адрес и парола, бутон **„LOG IN“** за вход, линк **„FORGOT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

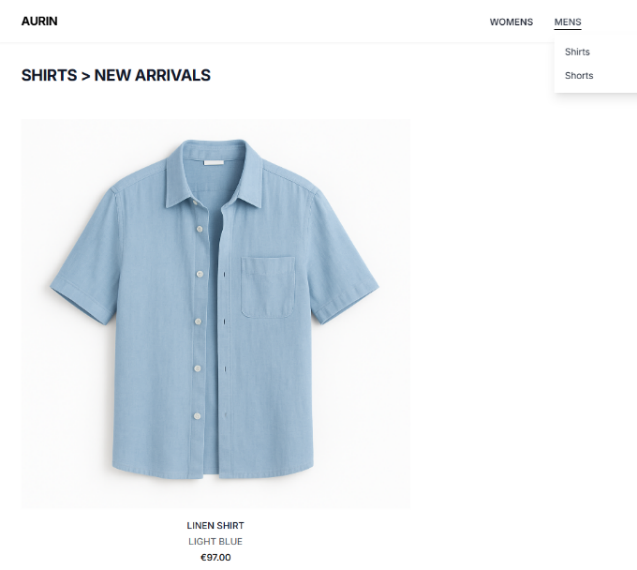
**Фигура 6: Форма за регистрация в сайта**

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 [7], [10].

**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**

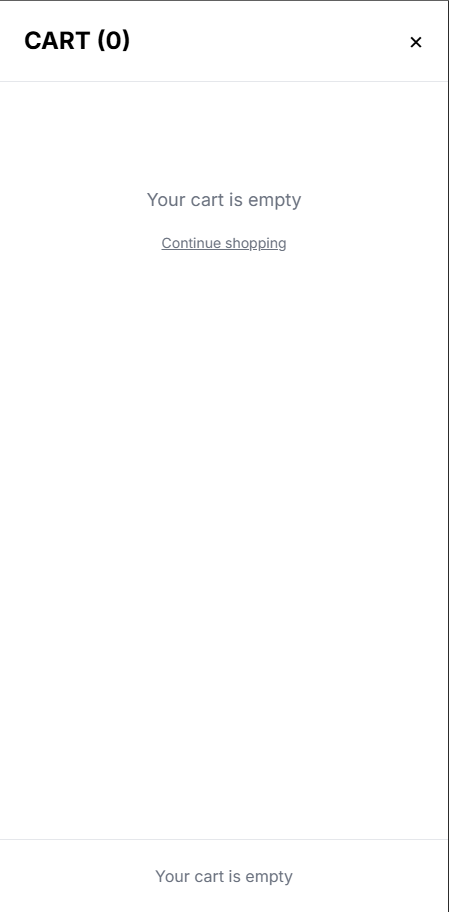
Тази комбинация от ясна продуктова информация, помощна функция за избор на размер и бързи бутони за добавяне в количката или списъка с желания прави детайлната страница удобна и интуитивна за потребителите.

**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.

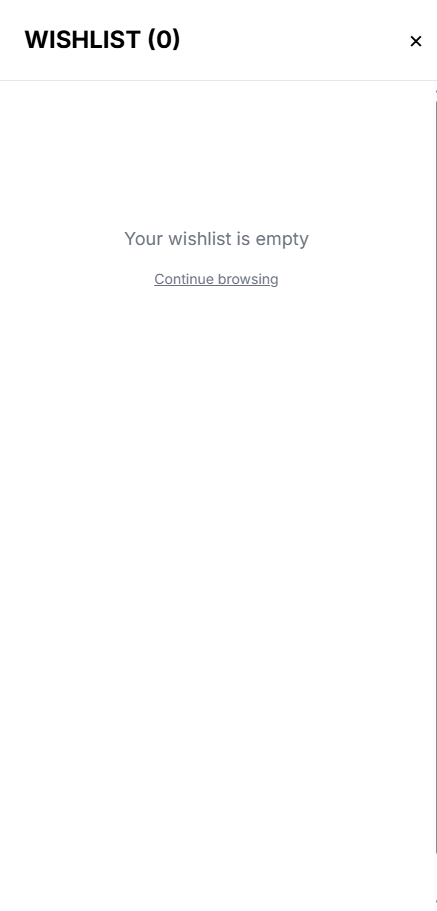
Дизайнът на количката е изчистен и следва същия визуален стил като останалата част от сайта, като акцентът е върху бързото и лесно управление на избраните продукти. разработка, улеснява бъдещото разширяване на системата и осигурява централизирано управление на бизнес логиката.

### 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**

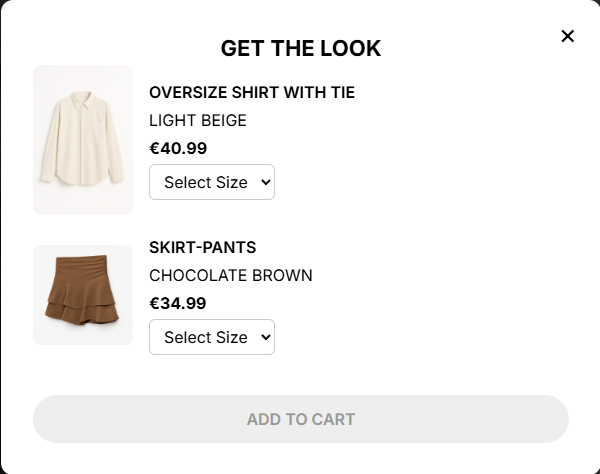
### 5.3.6. GET THE LOOK

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.

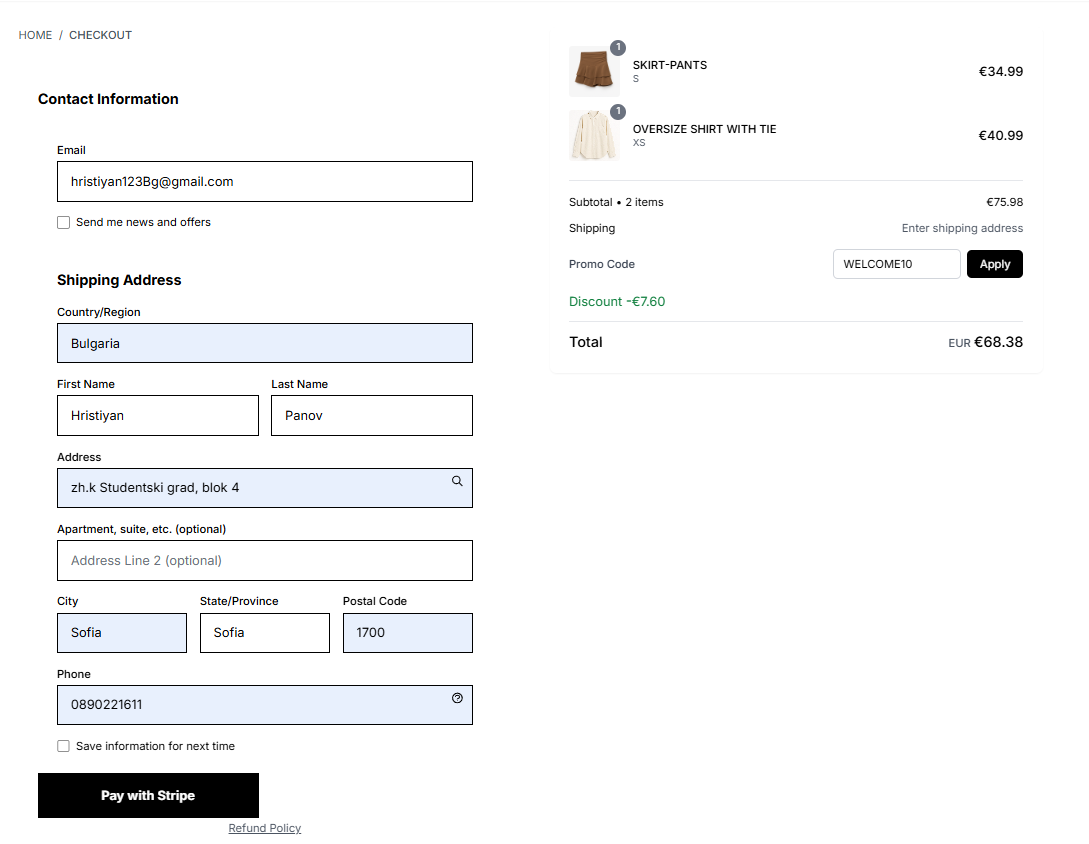
**Фигура 13: Модален прозорец на “GET THE LOOK”**

### 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).

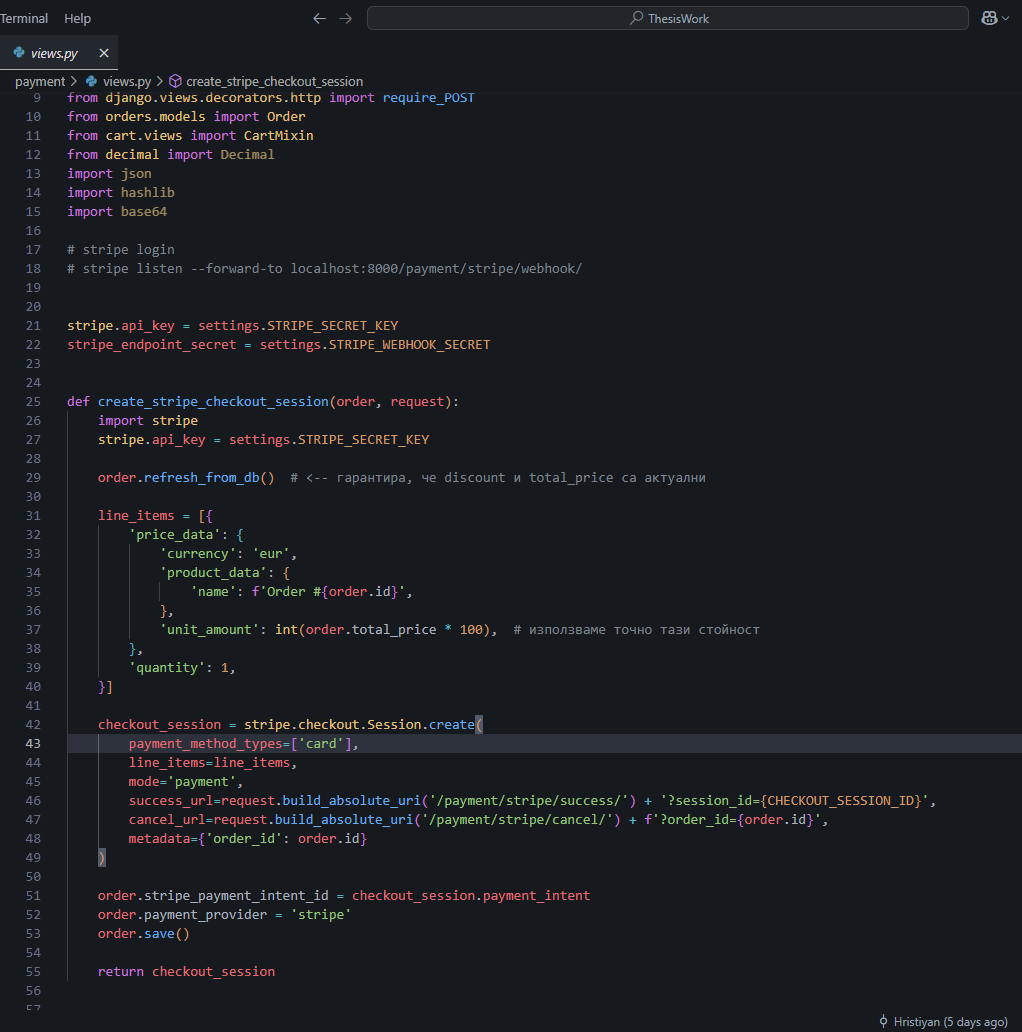
След попълване на данните и потвърждаване на поръчката, потребителят се пренасочва към процеса на плащане. Всички данни от формуляра се валидират, за да се гарантира коректното въвеждане и да се избегнат грешки при обработката на поръчката.

**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 [3].

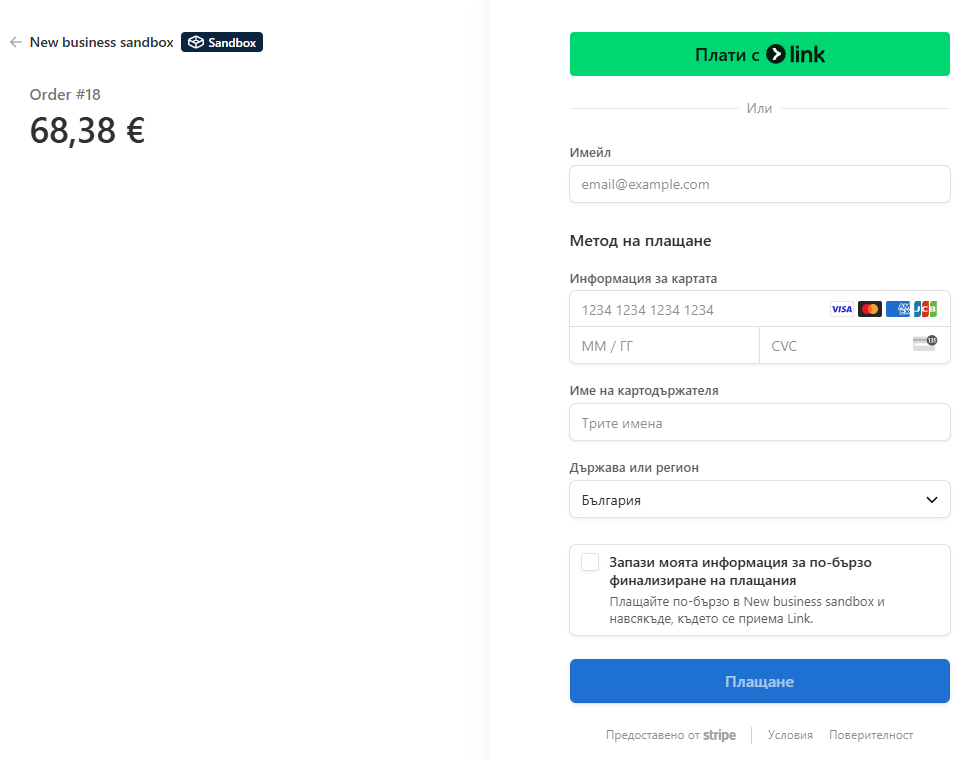
Процесът започва при финализиране на поръчка от Checkout страницата. Извиква се функцията create\_stripe\_checkout\_session(), която създава нова Stripe Checkout сесия и предава на Stripe данните за поръчката:

* 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**

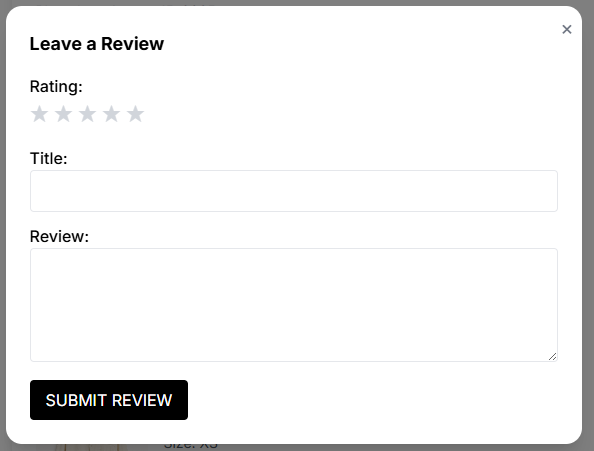
Системата разполага и с отделни изгледи за обработка на успешни и отказани плащания (**stripe\_success()** и **stripe\_cancel()**), които показват съответните страници и актуализират статуса на поръчката.

**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:**

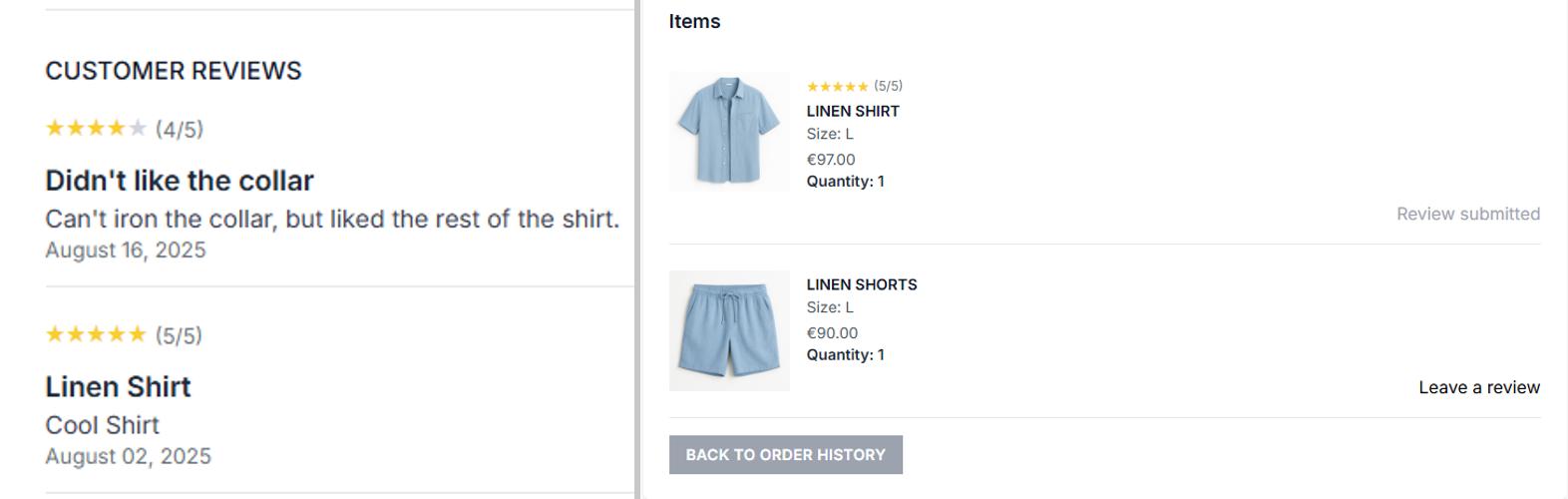
* **Едно ревю на потребител за продукт.** Ако потребителят вече е оставил ревю, бутонът „Leave a review“ в детайлната страница на поръчката(order\_detail) **не може да се използва** (скрит е за този потребител).

**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.

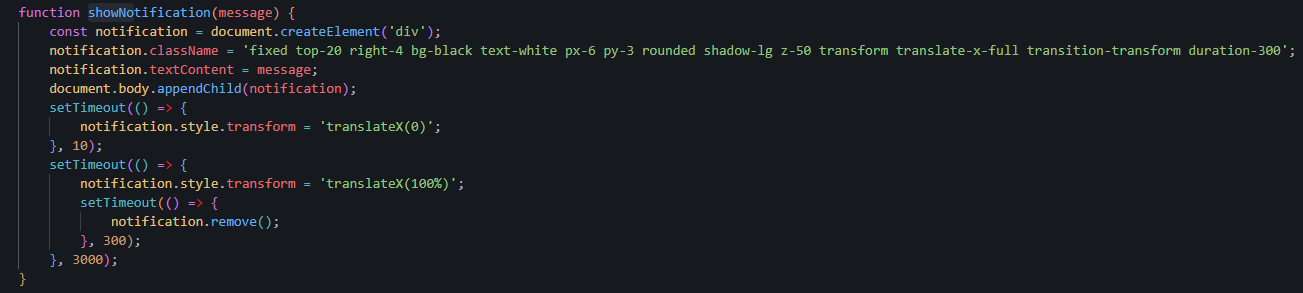
**Фигура 19: Визуализация на ревютата в страницата на артикула, активен и неактивен бутон за оставяне на ревю**

### 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.

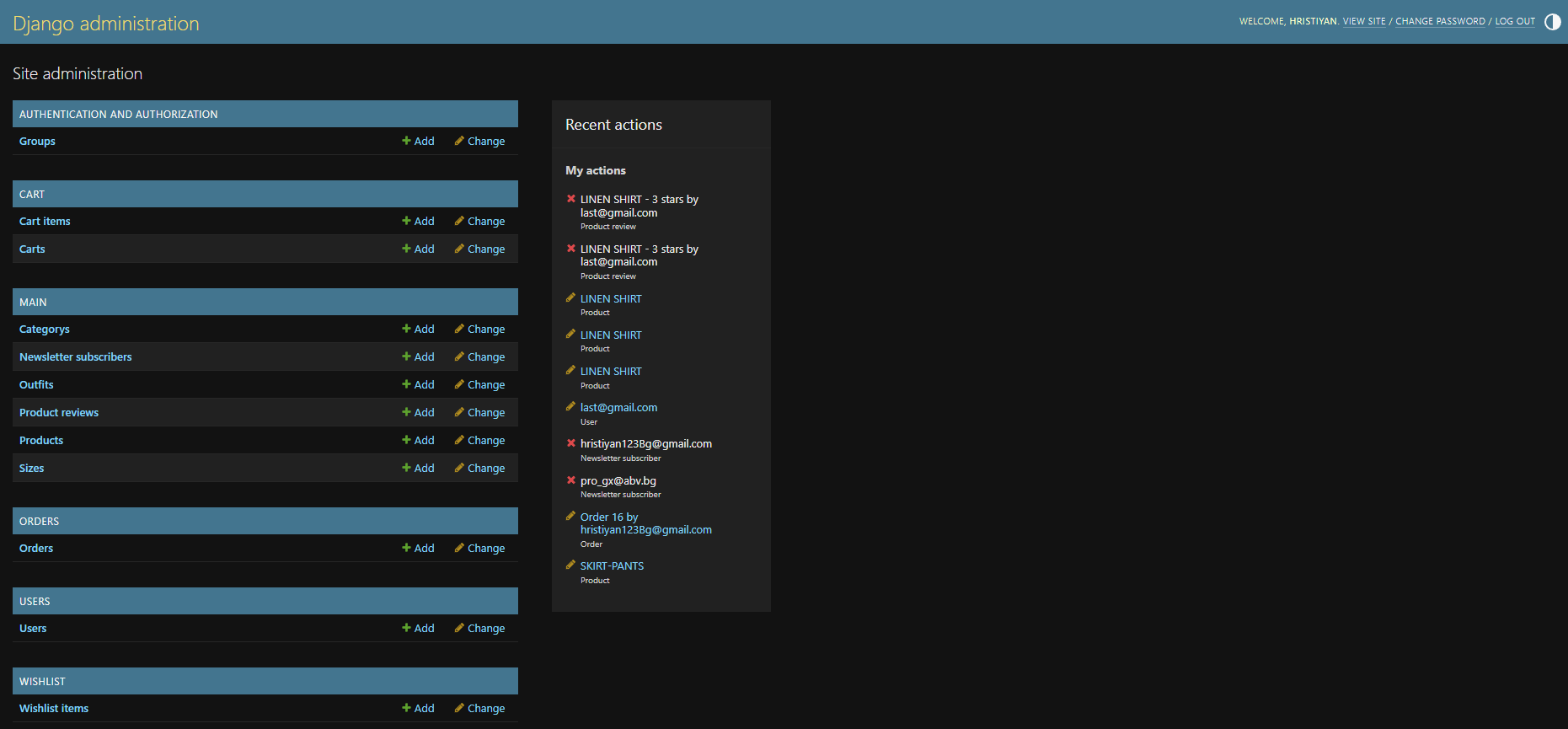
**Фигура 21: Toast нотификация**

### 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 [1], [4].

**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.

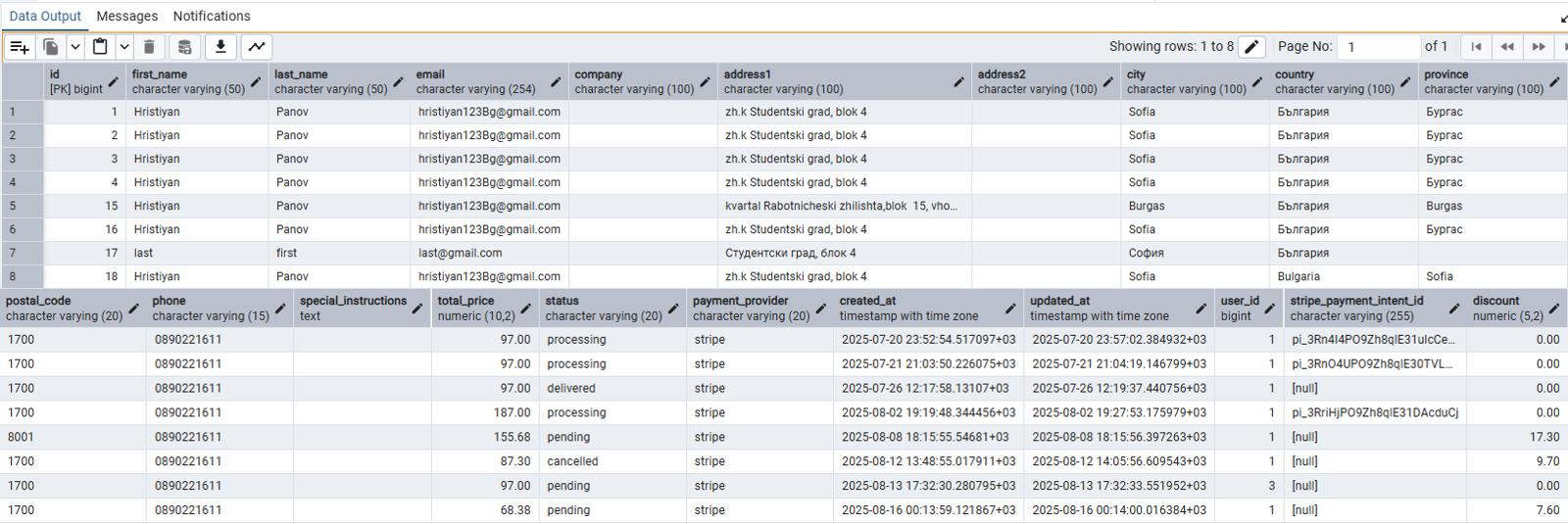
# 6. ТЕСТВАНЕ И ОЦЕНКА

## 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:

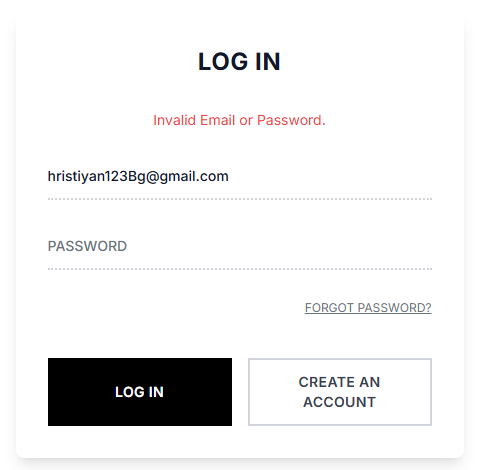
* Операционна система: Windows/Linux (локална разработка).
* Сървърна среда: Django с вграден дев-сървър.
* База данни: PostgreSQL.
* Платежна интеграция: Stripe тестов режим.
* Имейл услуга: Gmail SMTP със специален тестов акаунт.
* Браузъри: Chrome, Firefox, Edge.
* Тестови потребители: създадени специално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

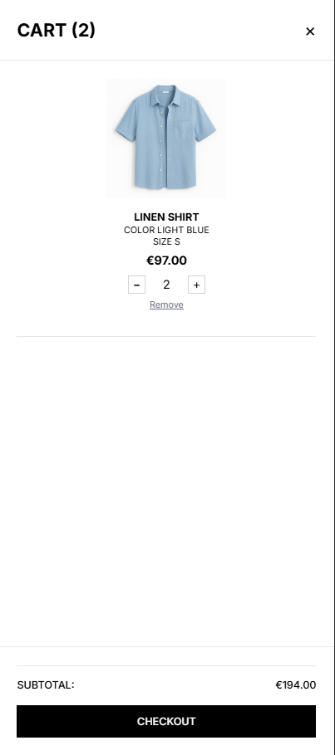
### 6.2.1. Потребители и автентикация

* 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.

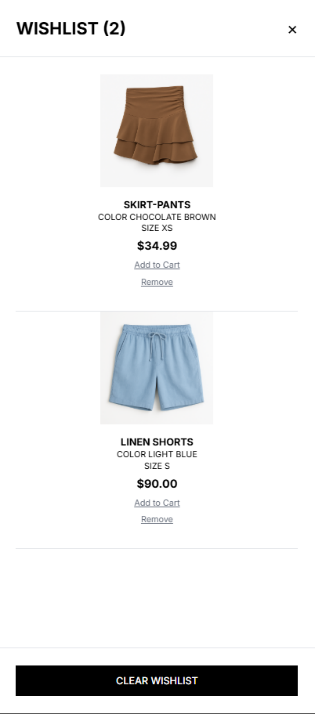
**Фигура 23: Грешка при неправилно въведена парола при опит за вход**

### 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**

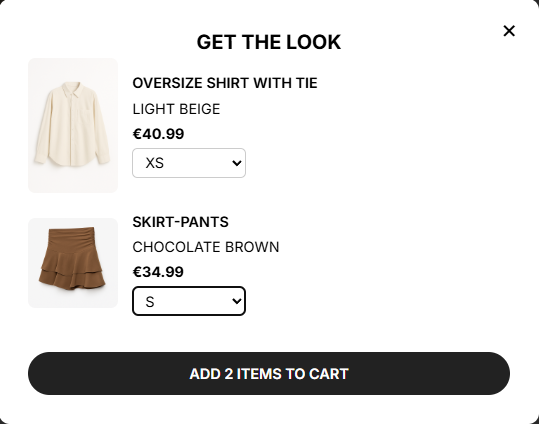
### 6.2.3. Списък с желания (Wishlist)

* 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**

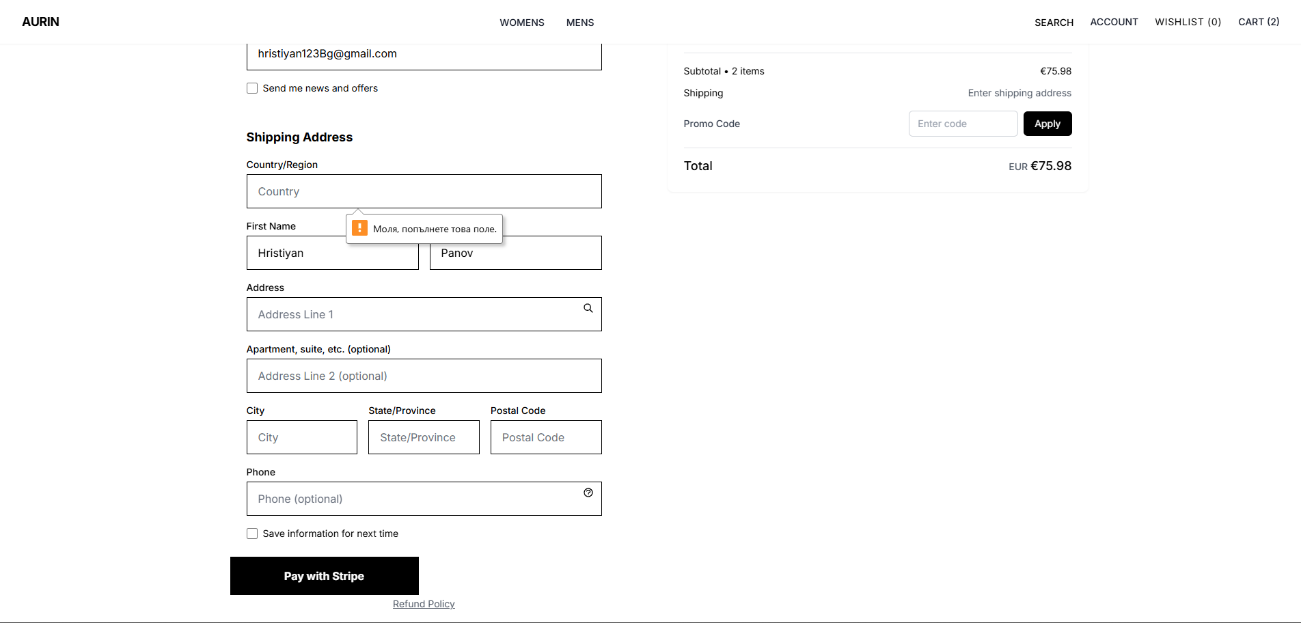
### 6.2.4. Модул „GET THE LOOK“

* 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.

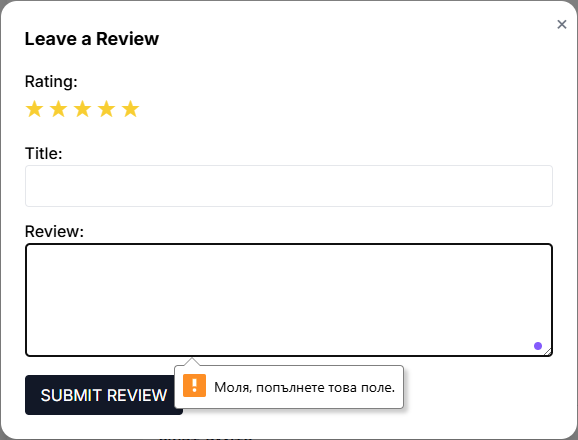
**Фигура 26: Модален прозорец на “Get the look” преди добавяне на артикули в количката**

### 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.

# 7. ЗАКЛЮЧЕНИЕ

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.

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.

# 8. ИЗПОЛЗВАНА ЛИТЕРАТУРА И ИЗТОЧНИЦИ

1. Django Documentation. The Web framework for perfectionists with deadlines.  
   URL: <https://docs.djangoproject.com/>
2. PostgreSQL Global Development Group. PostgreSQL Documentation.

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. Gmail SMTP. Sending emails with Gmail SMTP.  
   URL: <https://support.google.com/a/answer/176600?hl=en>
9. Zara – <https://www.zara.com/bg/>
10. H&M – <https://www2.hm.com/bg_bg/index.html>
11. 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)
12. ASOS – <https://www.asos.com/>

## 8.1. Приложение

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

<https://github.com/HristiyanPanov/ThesisWork>

В процеса на разработка на системата за онлайн магазин за дрехи и при съставянето на документацията се използват редица технически термини и съкращения, характерни за уеб програмирането, електронната търговия и софтуерната архитектура. За по-голяма яснота и удобство, в (Табл. 2) са обобщени най-често срещаните от тях, като към всяко е добавено кратко обяснение или превод.

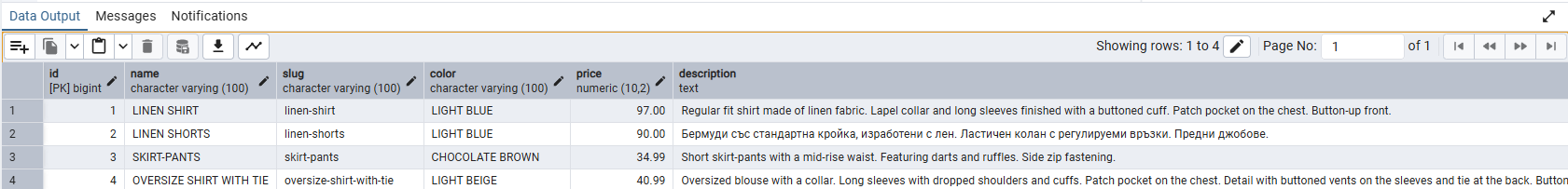
Включени са съкращения, обозначаващи технологии и архитектурни модели като **ORM (Object-Relational Mapping)**, **CRUD (Create, Read, Update, Delete)**, **API (Application Programming Interface)**, както и термини, свързани с потребителския интерфейс като **UI (User Interface)** и **UX (User Experience)**. Някои от съкращенията се отнасят до защитни механизми и протоколи, използвани в системата – например **CSRF (Cross-Site Request Forgery)**, **SMTP (Simple Mail Transfer Protocol)** и **PBKDF2**, алгоритъмът за хеширане на пароли, предоставян от Django.

В таблицата са включени и специфични за системата функционалности, като **Cart (Количка)**, **Wishlist (Списък с желания)** и модулът **GET THE LOOK**, който предлага готови облекла. По този начин таблицата съчетава както общоприети термини от сферата на уеб технологиите, така и конкретни понятия, свързани с реализираната платформа.

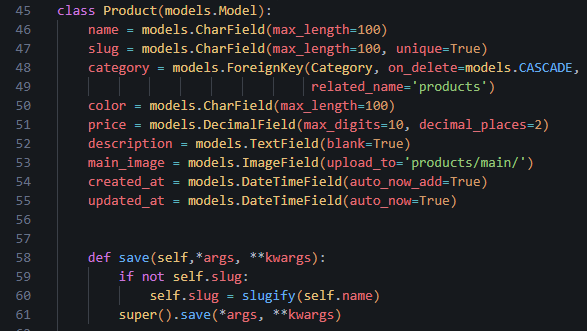
**Таблица 2: Използвани термини и съкращения**

|  |  |
| --- | --- |
| Съкращение / Термин | Обяснение |
| ORM | *Object-Relational Mapping* – механизъм за свързване на обекти с бази данни, използван в Django. |
| CRUD | *Create, Read, Update, Delete* – основни операции за работа с данни, реализирани чрез Django ORM и админ панела. |
| UI | *User Interface* – потребителски интерфейс, визуалната част на системата. |
| UX | *User Experience* – потребителско изживяване при работа със системата. |
| API | *Application Programming Interface* – интерфейс за взаимодействие между системи. |
| Съкращение / Термин | **Обяснение** |
| CSRF | *Cross-Site Request Forgery* – уеб атака, срещу която Django осигурява защита чрез токени. |
| HTML | *HyperText Markup Language* – стандартен език за изграждане на уеб страници. |
| CSS | *Cascading Style Sheets* – технология за визуално оформление на уеб страници. |
| JavaScript (JS) | Език за програмиране, който осигурява динамика и интерактивност в браузъра. |
| AJAX | *Asynchronous JavaScript and XML* – техника за асинхронна комуникация със сървъра, използвана за обновяване на съдържание без презареждане. |
| IDE | *Integrated Development Environment* – интегрирана среда за разработка, в случая Visual Studio Code. |
| PostgreSQL | Релационна система за управление на бази данни, използвана в проекта. |
| Stripe | Външен доставчик за онлайн плащания, интегриран в системата. |
| SMTP | *Simple Mail Transfer Protocol* – протокол за изпращане на имейли, използван в модула „Forgot password“. |
| PBKDF2 | *Password-Based Key Derivation Function 2* – алгоритъм за хеширане на пароли, използван по подразбиране в Django. |
| GET THE LOOK | Модул в онлайн магазина, който представя готови аутфити и позволява добавянето им в количката наведнъж. |
| Wishlist (Списък с желания) | Функционалност за запазване на избрани продукти за бъдеща покупка. |
| Cart (Количка) | Модул, който позволява добавяне, преглед и управление на продукти преди финализиране на поръчка. |

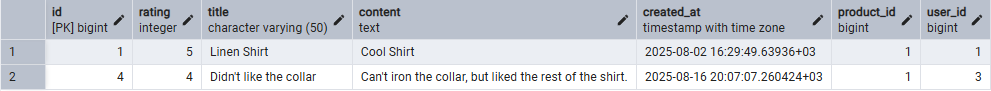
### 8.1.1. Приложение A: База данни – таблици и модели

**Продукти (product)**.Таблицата съхранява основната информация за артикулите: наименование, уникален slug, категория, цвят, цена, описание, основно изображение и времеви маркери (създаване/актуализация). В модела Product се дефинират тези полета и бизнес правилото за автоматично генериране на slug от името чрез slugify. Връзката към Category е N:1, което позволява организиране на каталога по категории и подкатегории.

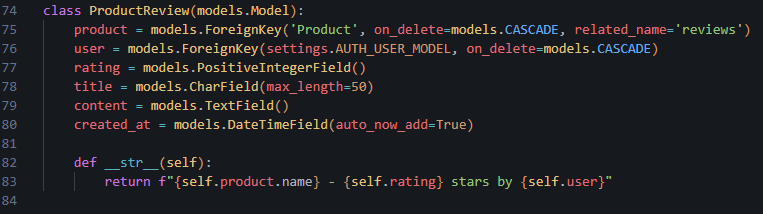
**Фигура 29: Регистрирани продукти в сайта**



**Фигура 30: Клас Product – структура и логика на запис на продукт от models.py**

**Ревюта на продукти (product\_review).** Таблицата съхранява обратна връзка от потребителите: рейтинг (цяло число), заглавие и текст на ревюто, както и връзка към продукта и автора. Полето created\_at позволява сортиране по най-нови отзиви. В модела има и удобен \_\_str\_\_, който показва „име на продукт – рейтинг – от потребител“.

**Фигура 31: Записване на оставените отзиви**

**Фигура 32: Клас ProductReview – структура и логика на запис на отзив от models.py**

### 8.1.2. Приложение Б: Извадки от сорс кода

Функция за абонамент към бюлетин**.** Във **views.py** е реализирана функцията subscribe\_newsletter, която обработва POST заявка от формата за абонамент.

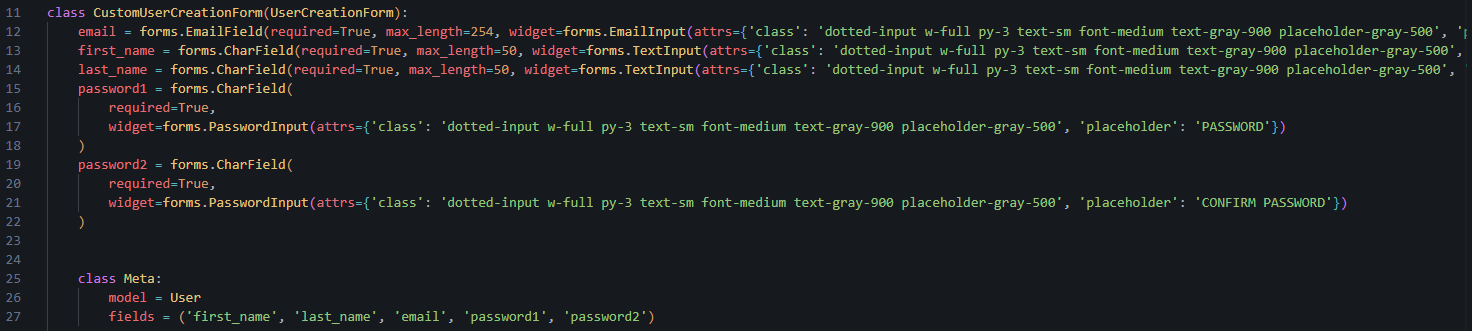
1. **Валидация** – проверява се въведен имейл чрез NewsletterForm.
2. **Проверка в базата** – имейлът трябва да съществува като регистриран **User**, иначе се връща грешка.
3. **Създаване на абонат** – ако е нов, записът се добавя в **NewsletterSubscriber** и се присвоява код за отстъпка WELCOME10.
4. **Изпращане на имейл** – новият абонат автоматично получава имейл с кода чрез send\_mail.
5. **JSON отговори** – функцията връща статус дали абонаментът е нов, вече съществува или има грешки.

Така се комбинират валидирани форми, работа с база данни и изпращане на имейли, като всичко е оптимизирано за динамична работа с AJAX.

 **Фигура 33: Функция subscribe\_newsletter – Абониране и изпращане на код за отстъпка**

Форма за регистрация на потребител. Във forms.py е дефинирана класa CustomUserCreationForm, разширяваща стандартната UserCreationForm на Django.

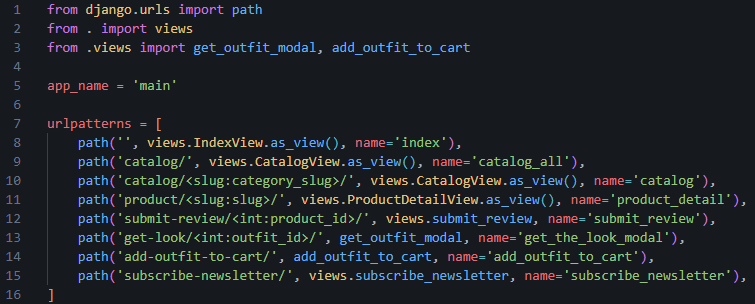
* Добавени са полета за email, first\_name и last\_name, всички задължителни и с ограничение за максимална дължина.
* Полетата password1 и password2 се използват за въвеждане и потвърждение на паролата.
* Чрез widget атрибутите са дефинирани HTML елементи с персонализирани CSS класове и placeholder текстове, за да се постигне унифициран визуален стил.
* Вътрешният клас Meta указва модела User и списък с използвани полета.

Формата осигурява валидация на въведените данни и предотвратява създаването на акаунти с липсваща или некоректна информация.

**Фигура 34: CustomUserCreationForm – регистрационна форма**

Във urls.py на приложението *main* са дефинирани основните пътища на системата. Всеки път е асоцииран с конкретен view, който обработва заявката и връща подходящ отговор:

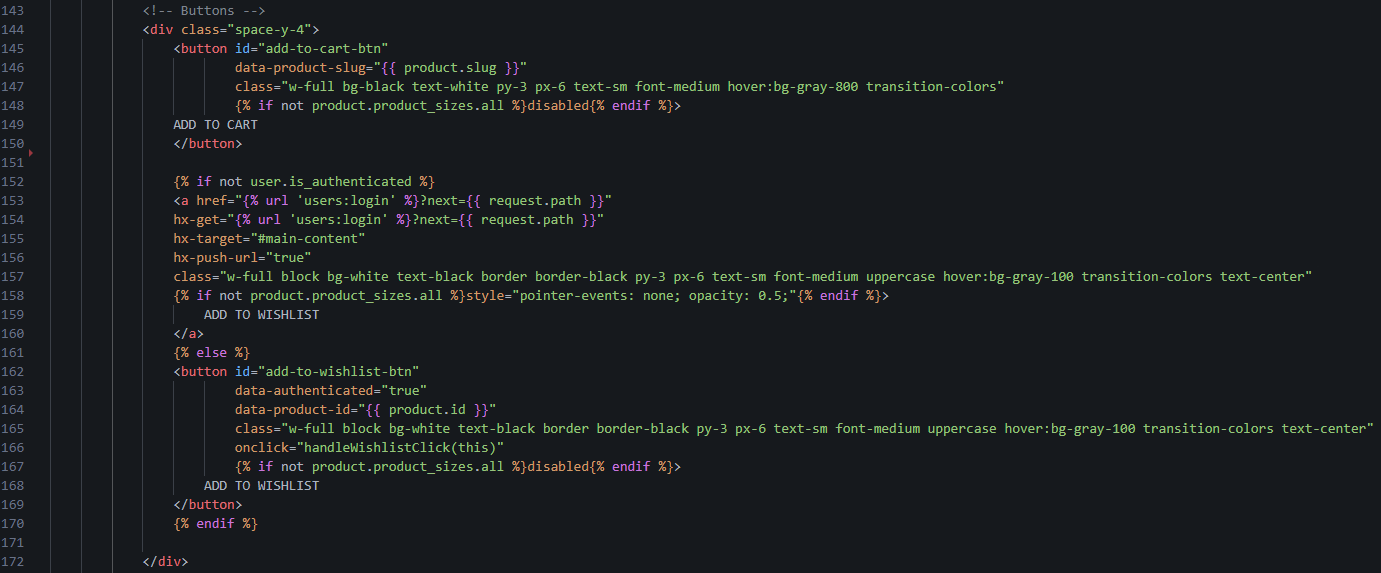
* **'' → IndexView** – начална страница с визуализация на продукти и основни категории.
* **catalog/** → CatalogView – показва пълния каталог с възможност за филтриране.
* **catalog/<slug:category\_slug>/ → CatalogView** – зарежда продукти само от избраната категория.
* **product/<slug:slug>/ → ProductDetailView** – детайлна страница на продукт.
* **submit-review/<int:product\_id>/ → submit\_review** – обработва изпращане на ревю за даден продукт.
* **get-look/<int:outfit\_id>/ → get\_outfit\_modal** – връща съдържанието на модал за „GET THE LOOK“.
* **add-outfit-to-cart/ → add\_outfit\_to\_cart** – добавя всички продукти от аутфит директно в количката.
* **subscribe-newsletter/ → subscribe\_newsletter** – реализира абонамента за бюлетин с промоционален код.

Тази конфигурация осигурява ясен път за навигация между различните модули на системата и улеснява разделянето на логиката по приложения.

**Фигура 35: URL маршрути в приложението main**

В product\_detail.html са реализирани бутоните за добавяне на продукт в количката и списъка с желания. Те демонстрират как фронт-енд логиката е обвързана със сървърната част чрез динамични атрибути и JavaScript.

* „Add to Cart“
  + Бутона използва атрибут data-product-slug за идентифициране на продукта.
  + Ако няма налични размери, се добавя disabled, което предотвратява добавянето.
  + При кликване се извиква AJAX заявка, която добавя продукта в количката без презареждане на страницата.
* „Add to Wishlist“
  + Ако потребителят не е логнат, бутонът води към страницата за вход (users:login) и е интегриран с HTMX за динамично зареждане на формата.
  + Ако потребителят е логнат, се показва истински бутон с data-product-id, който извиква JavaScript функцията handleWishlistClick.
  + При липса на размери бутонът също се деактивира.

Комбинацията от условна логика в шаблона, атрибути data-\* и JavaScript позволява динамично управление на действията спрямо статуса на потребителя и наличностите.

**Фигура 36: HTML шаблон на бутоните за добавяне в количка и списък с желания**

Другите елементи от системата – включително страницата за пазарска количка, процеса на създаване на поръчка, регистрацията и входа на потребители, потребителския профил с история на поръчките, както и модулите „GET THE LOOK“ и „Списък с желания“ – са реализирани по аналогичен начин чрез HTML шаблони, CSS стилове и JavaScript логика, съобразени с минималистичния и модерен дизайн на сайта.  
За динамичните взаимодействия (например добавяне на продукти в количката или списъка с желания) е използвано AJAX/Fetch API, комбинирано с **toast известия**, които информират потребителя за успешно действие или грешка, без да се презарежда страницата.

Всички представени примери от сорс кода и потребителския интерфейс са част от реално работещата система за онлайн магазин за дрехи, разработена в рамките на дипломната работа. Пълният код на проекта е включен в архива, който придружава документацията.