



**СОФИЙСКА ПРОФЕСИОНАЛНА ГИМНАЗИЯ ПО ЕЛЕКТРОНИКА „ДЖОН
АТАНАСОВ“**

**ДЪРЖАВЕН ИЗПИТ
ЗА ПРИДОБИВАНЕ НА ТРЕТА СТЕПЕН НА ПРОФЕСИОНАЛНА КВАЛИФИКАЦИЯ –
ЧАСТ ПО ТЕОРИЯ НА ПРОФЕСИЯТА**

ДИПЛОМЕН ПРОЕКТ

Тема: „Уеб-базирано приложение за модно облекло, обувки и аксесоари“

**професия: 481020 „Системен програмист“
специалност: 4810201 „Системно програмиране“**

Дипломант: Марк Огнянов Весков, 12А клас

Ръководител на дипломния проект: Димитрина Лукова Делийска

Подпис:

(дипломант)

Подпис:

(ръководител)

София, 2024 г.

Съдържание

1. Въведение.....	2
1.1. Задание за разработка.....	2
1.2. Анализ на заданието	2
2. Основна част.....	3
2.1. Избор на технологии.....	3
2.1.1. Избор на технологии за потребителската част	3
2.1.2. Избор на технология за сървърната част	4
2.1.3. Избор на технология за базата данни	5
2.2. Софтуерен дизайн	6
2.2.1. Софтуерна архитектура	6
2.2.2. Файлова структура на проекта	7
2.2.3. Описание на взаимодействието със сайта	8
2.2.4. Графичен дизайн на уебсайта.....	12
2.2.5. Дизайн на базата данни	13
2.2.6. Функционалност на сайта	15
2.2.7. Дизайн на сървъра.....	19
2.2.8. CRUD заявки	24
3. Заключение.....	30
3.1. Постигнати цели	30
3.2. Перспективи за развитие.....	31
4. Източници.....	32
5. Приложение.....	33

1. Въведение

1.1. Задание за разработка

В съвременния свят на електронната търговия онлайн магазините за мода се радват на голям успех и популярност. С растящия брой потребители, които предпочитат да пазаруват онлайн, разработването на функционален, атрактивен и лесен за използване онлайн магазин за модно облекло, обувки и аксесоари представлява значима възможност за предприемачески успех и привличане на нови клиенти.

Заданието е разработване на уеб-базирано приложение за модно облекло, обувки и аксесоари. Уебсайтът ще се нарича Casspie и ще предлага широка гама от продукти за различни стилове. Приложението трябва да предлага на потребителите лесна навигация, като им предоставя всички важни функции за пазаруване в един електронен магазин. Освен интуитивния начин на пазаруване, приложението трябва да има и възможност за създаване на профили. Включен е и администраторски профил, който може да следи статистики, навигира поръчки, добавя, актуализира и премахва продукти. Това улеснява работата на администраторите, като намалява евентуалните разходи за собственика на уебсайта.

1.2. Анализ на заданието

Уеб приложението Casspie представлява електронен магазин, който се фокусира върху предоставянето на отлично потребителско преживяване. В днешно време има сходни на тази тема уебсайтове, затова е важно приложението да не изостава нито с графичния интерфейс, нито с необходимите функционалности на един онлайн магазин, а именно да ги надгражда.

За да може да се постигнат изискванията потребителите трябва да могат да навигират лесно и удобно чрез разделението на продуктите на категории. Интуитивният графичен интерфейс им позволява да филтрират и сортират продуктите според техните предпочитания. Освен това, приложението предлага функционалност за добавяне на любими артикули, което улеснява потребителите в търсенето и достъпа до техните предпочитани продукти. За всеки артикул се предоставят детайлни описания, снимки и таблица с размери, което помага на потребителите да направят информиран избор. След като потребителите изберат продуктите, които искат да закупят, те могат лесно да ги добавят в количката и да извършат онлайн поръчка. При плащането, потребителите имат възможност да избират между три опции: наложен платеж, плащане с карта чрез виртуален POS терминал или

МАРК ВЕСКОВ

банков трансфер. Също така, има два начина за доставка: до офис на куриер или до адрес. След успешната поръчка, потребителите получават персонализиран имейл с информация за поръчката си. Те могат да проследят статуса на поръчката си както чрез имейла, така и чрез потребителския си профил в приложението. Потребителският профил включва лични данни, история на поръчките и възможност за абониране за бюлетина на магазина. Освен потребителски профили, приложението включва и администраторски профил, който предоставя достъп до поръчките, управлява продуктовата база данни като добавя, актуализира и изтрива артикули, както и следи статистики за продажбите. В случай на забравена парола, потребителите имат възможност да я сменят чрез сигурна процедура за възстановяване на паролата. Сигурността на данните е от основно значение за Casspie, като се използва криптиране на данни.

2. Основна част

2.1. Избор на технологии

2.1.1. Избор на технологии за потребителската част

За уеб-базираното приложение за модно облекло, обувки и аксесоари, изборът на подходящи технологии за потребителския интерфейс е от решаващо значение за осигуряване на гладко и ефективно изживяване за потребителите. В този контекст, след сравняване на различни технологии, оптималните за представеното приложение включват HTML5, EJS, CSS, Bootstrap и JavaScript.

HTML5 предоставя семантични елементи, които подобряват структурата на уеб страниците и улесняват тяхното индексване от търсачки, което е от съществено значение за електронни магазини. Тези елементи могат да бъдат използвани за ясно представяне на различните категории и продукти, които се предлагат в приложението.

EJS е избран като шаблонен език, който позволява динамичното вграждане на данни в HTML шаблоните. Това е от изключителна важност за уеб магазина, тъй като позволява лесното обновяване на продуктовата информация и динамичното генериране на уеб страниците в реално време.

CSS и Bootstrap са ключови за стилизирането на интерфейса и създаването на елегантен и функционален дизайн. Чрез тях може да се осигури еднакво привлекателен и

МАРК ВЕСКОВ

консистентен външен вид на уеб приложението, като се използват готови компоненти и стилове за по-бърза и мащабируема разработка. Също така Bootstrap помага за по-лесното осъществяване на responsive design на приложението.

JavaScript е мощен език за програмиране, който се използва за добавяне на интерактивност и динамичност към уеб приложенията. Той допринася за интерактивността на уебсайтовете, позволявайки добавянето на различни функционалности като валидация на форми, анимации, AJAX заявки, манипулиране на DOM елементи и други, което не само ги прави по-привлекателни и функционални за потребителите, но и подобрява изживяването им чрез филтриране на продуктите, динамично обновяване на цените и кошницата за артикули.

Съчетавайки HTML5, EJS, CSS, Bootstrap и JavaScript, уеб-базираното приложение за модно облекло, обувки и аксесоари може да осигури удобна и функционална платформа за потребителите, която отговаря на техните нужди и предпочитания.

2.1.2. Избор на технология за сървърната част

Изборът на подходяща технология за сървърната част на уеб приложението представлява важна роля за осигуряване на стабилност, сигурност и ефективност на приложението. За целта, Express.js се явява един от най-популярните, мощни и гъвкави фреймуърци за създаване на сървърни приложения с Node.js.

Express.js се използва за създаване на бързи и мащабируеми сървърни приложения и API-та. Той предоставя гъвкава и лесна за използване архитектура, която улеснява създаването на маршрути, обработка на HTTP заявки и управление на сесии. Също така осигурява възможност за интеграция с различни модули и библиотеки, което прави едно отлично решение за уеб приложения, които изискват бързодействие и гъвкавост.

В сравнение с други технологии за сървърна част, като например Flask за Python или Ruby on Rails за Ruby, Express.js се отличава със своята широка популярност и активна общност. Въпреки че всички тези фреймуърци предоставят сходни възможности за създаване на сървърни приложения, Express.js е често предпочитан заради своята лесна за използване архитектура и гъвкавост.

С употребата на пакети като bcryptjs, body-parser, cookie-parser, jsonwebtoken, mysql2 и nodemailer, проектът получава допълнителни предимства.

bcryptjs осигурява мощно криптиране на пароли, което подобрява сигурността на потребителските данни.

МАРК ВЕСКОВ

body-parser улеснява обработката на данни, пристигащи в заявките към сървъра, като JSON, URL-encoded форми и др.

cookie-parser позволява удобната обработка на HTTP бисквитки, което е от съществено значение за управлението на сесии и автентикацията на потребители.

jsonwebtoken е полезен за създаване на идентификационни токени за автентикация и изпълномощаване на потребителите.

mysql2 е бърз и ефективен драйвер за взаимодействие с MySQL, който осигурява надеждно свързване с базата данни и изпълнение на заявки. Предлага удобен начин за извличане, обработка и манипулиране на данни от базата данни, като осигурява оптимална производителност.

nodemailer предоставя възможност за изпращане на имейли от сървъра, което е важно за имплементацията на функционалности като забравена парола, форма за контакти и изпращане на имейл след осъществена поръчка.

Използването на Express.js в комбинация с тези пакети осигурява мощна и сигурна сървърна част за уеб приложението като същевременно се гарантира висока производителност и лесна поддръжка.

Причината да избира тази технология са именно тези фактори и също така заради своята отлична документация и широка общност от разработчици, което прави изучаването на този фреймуърк по-лесно.

2.1.3. Избор на технология за базата данни

Базата данни служи за съхраняването, организацията и управлението на данните. В контекста на проекта е избрана MySQL като технология за база данни поради няколко ключови предимства и причини.

MySQL е една от най-популярните релационни бази данни, известна със своята стабилност, сигурност и гъвкавост. Тази платформа е широко използвана от множество уеб приложения, включително и тези с високи нива на трафик и данни.

Едно от основните предимства на MySQL е неговата отлична производителност и оптимизация. Тя е проектирана да обработва големи обеми от данни ефективно и бързо, като предлага множество индекси и оптимизации за заявки.

MySQL осигурява и високо ниво на сигурност на данните, като предлага различни механизми за управление на достъпа и шифроване на информацията. Това е от изключително

МАРК ВЕСКОВ

значение, особено при работа с чувствителна информация като лични данни на потребителите.

Също така, MySQL е известен със своята лесна употреба и гъвкавост. Той предлага много инструменти за администриране и мониторинг, както и възможност за интеграция с различни програмни езици и приложения.

В сравнение с други бази данни, като например PostgreSQL или Microsoft SQL Server, MySQL се отличава със своята широка популярност и гъвкавост. Въпреки че всички тези системи предлагат стабилност, сигурност и оптимизация, MySQL често се предпочита заради своята лесна употреба и гъвкавост. Докато някои алтернативи могат да предложат специфични функции или по-напреднали инструменти за управление на данните, MySQL се отличава с балансиран подход, който го прави подходящ за широк кръг от приложения, включително и за уеб приложения с високо ниво на трафик и данни.

Изборът на MySQL за този проект е обусловен от тези ключови предимства. Неговата стабилност, производителност и сигурност го правят идеално решение за съхраняване и управление на данни в уеб приложението ми. В допълнение, опитът и познанията ми в работата с MySQL улесняват разработката и поддръжката на базата данни, което е също важен фактор при вземането на решение.

2.2. Софтуерен дизайн

2.2.1. Софтуерна архитектура

MVC (Model-View-Controller) е избраната архитектура в софтуерния дизайн на приложението, използвана за внедряване на потребителски интерфейси, данни и контролна логика. Тя разделя бизнес логиката от дисплея, осигурявайки по-добро разделение на труда и поддръжката. Приложението е разделено на следните три основни компонента:

Моделът (model) е отговорен за съхранение и управление на данните на приложението. Той е независим от гледката (view) и контролера (controller). В приложението MySQL е използван като система за управление на бази данни (СУБД) за съхранение на информация за продукти, потребители, категории, клиенти, поръчки и други данни.

Изгледът (View) е отговорен за визуализацията на данните към потребителя. Той е представен чрез HTML5, EJS, CSS и Bootstrap, JavaScript.

Контролерът (Controller) е посредник между модела и view. Той обработва потребителски вход на данни, взаимодейства с модела за извличане на данни и изпраща

МАРК ВЕСКОВ

данни към view за визуализация. Express.js е използван като фреймуърк за Node.js, който улеснява създаването на API и маршрутизация. JavaScript е използван за добавяне на интерактивност към приложението.

Избраната архитектура за уеб апликацията е MVC, тъй като тя е добре позната, лесна за разбиране и поддръжка. MVC е подходящ за уеб приложения, тъй като разделя приложението на ясни слоеве, което улеснява разработването и тестването.

Освен това, MVC е гъвкава архитектура, която може да се адаптира към различни нужди. Използвани са различни технологии за всеки слой, като MySQL за модела, HTML5, EJS, CSS, Bootstrap и JavaScript за view и Express.js и JavaScript за контролера.

В заключение MVC е подходяща архитектура за разработка на уеб-базирани приложения. Тя предлага разделение на отговорностите, повторно използваемост, яснота и лесно тестване.

2.2.2. Файлова структура на проекта

Структурата на приложението, както е дефинирана във Фиг. 2.2.2.1., играе ключова роля за организацията и разграничаването на различните части от кода:

Главната директория на уеб апликацията, Main directory, служи като централно място за координиране на всички компоненти на приложението. Тук се намират основните файлове и директории, които определят основната структура на приложението.

Папката config е важна за конфигуриране на приложението. Тя съдържа файлове, които дефинират как приложението се свързва с базата данни и с nodemailer (back-end услугата за изпращане на имейли).

Controllers папката е мястото, където се намират всички контролери, отговорни за обработката на функционалностите от сървърната част на приложението. Тук се дефинира и реализира логиката за различните операции, като например вход, регистрация, автентикация и авторизация на потребители.

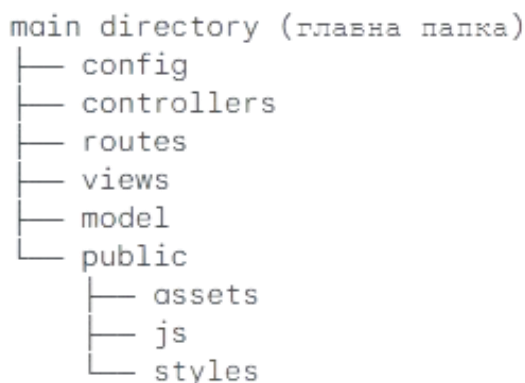
В public се съдържат статичните файлове, разделени на три подпапки за аудио, видео и изображения, CSS и JavaScript, които се изпращат директно на потребителя. Тези файлове са от потребителската част.

Routes папката е отговорна за дефинирането на маршрутите в приложението, които съпоставят URL адресите с контролерите и действията им. Това улеснява навигацията в приложението и осигурява ясен и лесен за разбиране начин за комуникация между различните компоненти.

МАРК ВЕСКОВ

Папката views в MVC приложението е отговорна за визуализацията на данните към потребителя. Тя съдържа шаблони, които се използват за генериране на HTML.

В папката models се съдържат моделите, които дефинират структурата и логиката на данните, използвани от приложението. Тези модели обикновено представляват таблиците в базата данни или други структури от данни, необходими за функционирането на системата.



Фиг. 2.2.2.1. Файлова структура

2.2.3. Описание на взаимодействието със сайта

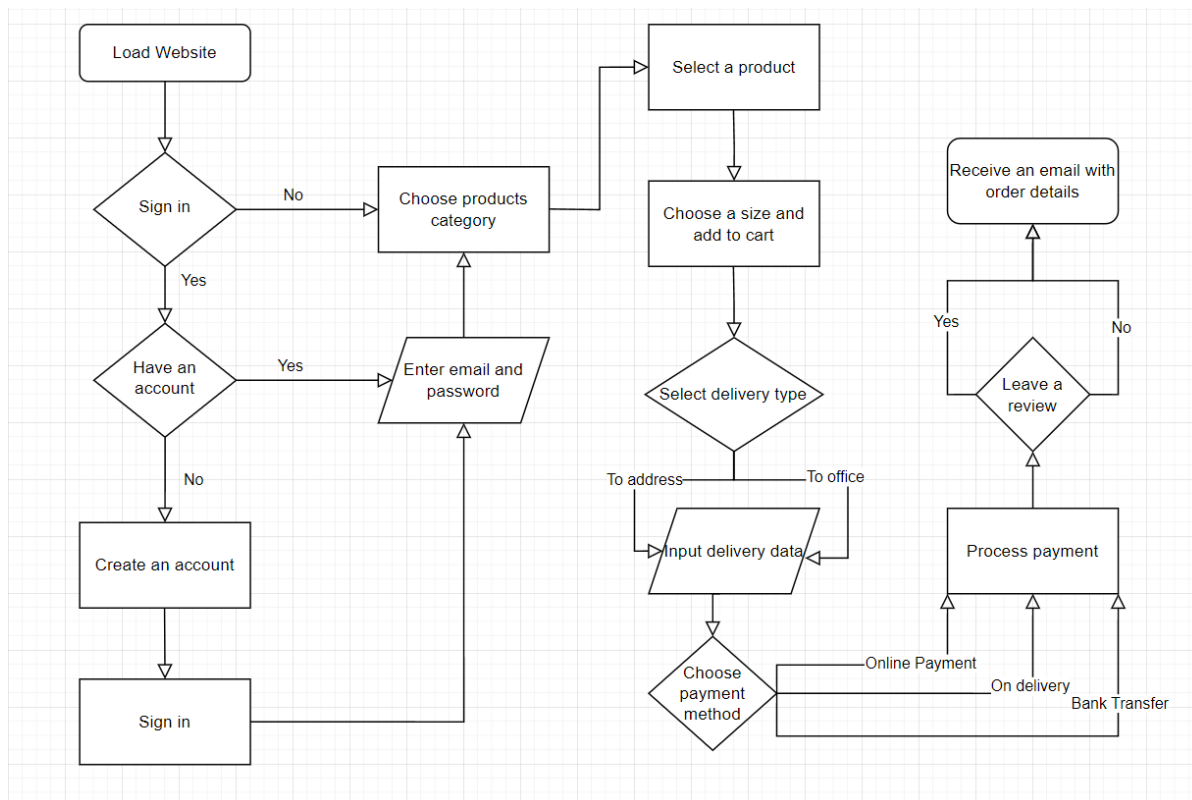
Да, разбирането на потребителския поток и как данните се движат през системата е от съществено значение за създаването на ефективно и интуитивно уеб приложение. Data-flow диаграмата е мощно средство за визуализиране на този процес, като предоставя ясна представа за последователността на действията и потребителските преживявания.

Чрез тази диаграма може да се изследва как потребителите взаимодействат с различните части на системата и как данните се прехвърлят между тях. Това не само помага за идентифициране на възможни проблеми или затруднения в процеса на използване на приложението, но също така предоставя основа за подобрения в дизайна и функционалността на системата.

Като част от разработката на уеб приложение, data-flow диаграмата служи като инструмент за комуникация между разработчиците, дизайнерите и другите заинтересовани страни. Тя улеснява обсъждането и разбирането на потребителските нужди и изисквания и помага за създаването на уеб приложение, което е лесно за използване и отговаря на очакванията на потребителите.

МАРК ВЕСКОВ

Работата на потребителите с графичния интерфейс е представена схематично с data-flow диаграма (виж Фиг. 2.2.3.1.).



Фиг. 2.2.3.1. Data-flow диаграма на уебсайта

След зареждане на приложението, първата страница, която потребителят вижда е заглавната. Той не е изпратен до страницата за регистрация или влизане в уебсайта, защото електронният магазин предлага възможност за пазаруване без създаване на акаунт (тази функционалност е добавена с цел улеснение на купувачите).

Въпреки това ако потребителят желае да влезе в акаунта си, той може да навигира към страницата за влизане в профил чрез бутона в навигационната лента. В тази форма потребителят трябва да въведе e-mail и парола като се правят проверки към базата данни (съществуват ли въведените парола и e-mail; ако информацията от поне едно поле съществува – прави се проверка на конкретното друго). При успешно влизане в акаунта, се генерира JWT токен, който се изпраща до потребителя и се използва за автентикация на последващите заявки. При грешка при влизане в акаунта се генерира съобщение, което индикира на потребителя какво трябва да направи.

МАРК ВЕСКОВ

В случай че потребителят желае да смени неговата парола, апликацията предоставя две опции за нейната смяна. Първата опция е за смяна на парола при налична стара такава (въвеждат се стара парола и нова парола). Втората опция е смяна на парола, в случай че старата е забравена (въвежда се e-mail, на който се изпраща уникален линк, препращат потребителят към страница, в която може да въведе нова парола).

Ако потребителят няма акаунт и иска да си направи регистрация, приложението трябва да му предостави форма за създаване на акаунт. В тази форма потребителят трябва да въведе свой e-mail и желана парола, като се правят проверки за уникалност на e-mail адреса и сила на паролата. След попълване на формата и придържане към изискванията, акаунтът се създава и потребителят може да влезе в него със създадените данни.

При успешно влизане в акаунта, потребителят може да продължи с навигацията в приложението. Първата стъпка е избирането на категория продукти от менюто. След това потребителят може да преглежда различните продукти в избраната категория, като има възможност за филтриране и сортиране по различни критерии, като например цена, пол, тип продукт, най-много продажби и др.

След като потребителят избере конкретен продукт, се отваря страница за този продукт, където се предоставят детайли като снимки, таблица с размери, описание и друга релевантна информация. Потребителят може да избере желанния размер на продукта и след това да го добави към количката.

При добавяне на продукт в количката, се извършват проверки за налично количество от продукта. Информацията за добавените артикули се съхранява в local storage, за да бъде запазена дори при презареждане на страницата или затваряне на браузъра.

След като потребителят е добавил всички желани продукти в количката, може да продължи към страницата за завършване на поръчката. Там той има възможност да избере начин на доставка - до адрес или до офис на куриерска компания. След въвеждане на информация за доставка, се показва резюме с информация за поръчката и опциите за плащане.

Потребителят може да избере между различните методи за плащане, като наложен платеж, плащане чрез карта във виртуален POS терминал или банков превод.

След като поръчката е обработена, потребителят получава персонализиран имейл с цялата информация за нея, включително избраните продукти, адрес за доставка и избрания метод на плащане.

МАРК ВЕСКОВ

Накрая, след успешното завършване на поръчката, потребителят е пренасочен към страница, където може да оцени преживяването си в сайта, предоставяйки обратна връзка и оценка за услугата. Това завършва цикъла на поръчката.

Друга много важна част от един електронен магазин е да може да предлага профилни страници както за потребители, така и за администратори, като информацията в тях е различна в зависимост от ролята на потребителя.

- За потребителите, профилната страница включва следните секции:

Информация за потребителя: тук се показват данни като име, e-mail адрес, парола, които потребителят е предоставил при създаването на профила си и информация за смяна на парола.

Секция за информацията за поръчките: в тази секция потребителят може да види историята на своите поръчки, включително текущи и предишни поръчки, статус на доставка и други подробности.

Секция за абониране за бюлетина: потребителят има възможност да се абонира за електронния бюлетин на магазина, за да получава актуална информация за нови продукти, промоции и специални оферти.

Секция за добавени продукти към любими: приложението предоставя функционалност за добавяне на продукти към списъка с любими, където потребителят може да съхранява продукти, които му харесват и иска да следи.

Секция за често задавани въпроси (FAQ): тук потребителят може да намери отговори на често задавани въпроси относно ползването на магазина, поръчки, доставка и други съответни теми.

- За администраторите, профилната страница включва допълнителни секции и функционалности, които са свързани с управлението и администрацията на магазина:

Dashboard: тази секция съдържа статистики за поръчки, продукти и продажби, които позволяват на администратора да следи и анализира активността на магазина.

Секция за всички регистрирани поръчки: администраторът може да види списък с всички регистрирани поръчки в приложението, като има възможност да ги сортира и да променя статуса им, за да може потребителят да следи процеса на доставка. След промяна на статуса на поръчка (в процес на изпълнение, обработка, изпратено, доставено, анулирано, възстановена сума, очакващо плащане) потребителят извършил съответната поръчка получава e-mail с обновена информация. Освен това в тази секция администраторът може да

МАРК ВЕСКОВ

генерира PDF файл с цялата информация за поръчките, което улеснява архивирането и споделянето на информацията с други участници.

Управление на продукти: администраторът има възможност да добавя, актуализира и изтрива продукти чрез графичен интерфейс, което значително улеснява процеса на управление на асортимента от продукти в магазина. Този графичен интерфейс позволява на администратора да променя цени, добавя описания и снимки, управлява наличността и други атрибути на продуктите, което допринася за ефективното функциониране на магазина.

2.2.4. Графичен дизайн на уебсайта

В днешния свят на дигиталната естетика, графичният дизайн на уебсайтовете играе ключова роля в привличането и задържането на потребителско внимание. В този контекст, разработката на уебсайта е особено насочена към използването на черно, бяло и нюанси на сивото, с цел създаване на силно визуално впечатление (виж точка 5).

Изборът на тези цветове не е случаен. Черното и бялото създават красив контраст, който подчертава важните детайли и аспекти на продуктите, като същевременно им придава модерен вид. Нюансите на сивото допълват този контраст, придавайки на уебсайта дълбочина и изтънченост.

Стилът на дизайна е съобразен с модната индустрия, като внимателно подбрани шрифтове, качествени изображения и изчистени елементи осигуряват на потребителите незабравимо визуално изживяване.

Разработеният responsive design е от ключово значение за удобството на потребителите. Благодарение на него, уебсайтът се адаптира автоматично към различните устройства и размери на екрана, осигурявайки оптимално изживяване както на десктоп, така на мобилни устройства и други.

CSS и Bootstrap са използвани за създаването на функционалност и естетика, които не само привличат вниманието, но и улесняват потребителите в тяхното онлайн пазаруване. Анимации и ефекти са интегрирани в дизайна, за да добавят интерактивност и да привлекат още повече внимание към продуктите.

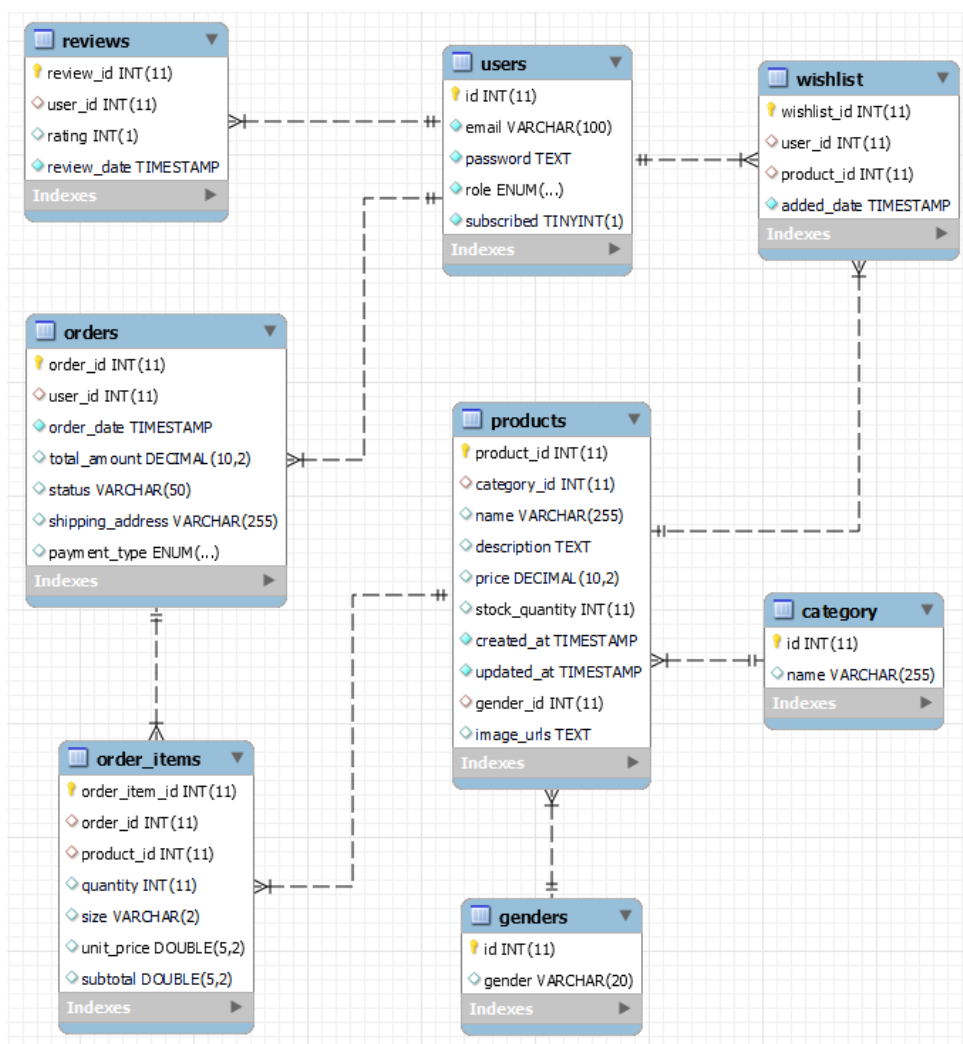
В днешно време, когато конкуренцията в онлайн магазините е ожесточена, графичният дизайн е от критично значение за установяването на уникалност и привличането на вниманието на потребителите. Представеният електронен магазин не само отговаря на тези

МАРК ВЕСКОВ

изисквания, но и подчертава стилът и класата на бранда, като същевременно предоставя удобство и леснота в пазаруването на клиентите.

2.2.5. Дизайн на базата данни

Дизайнът на базата данни е от съществено значение за функционалността и ефективността на всяко приложение. Използвайки предоставената ER диаграма (виж Фиг.2.2.5.1.) и информацията за състава на таблиците, могат да се опишат релациите между тях и техните характеристики.



Фиг. 2.2.5.1. Entity Relationship (ER) диаграма на базата данни

Дизайнът на базата данни съдържа общо 8 таблици, които представляват различни аспекти от функционалността на приложението. Тези таблици се свързват чрез външни

МАРК ВЕСКОВ

ключове и установяват връзки между данните, за да осигурят ефективно управление и достъп до информацията.

Таблица "reviews":

- Включва информация за отзивите на потребителите.
- Съдържа уникален идентификатор review_id, свързан с потребителя чрез външния ключ user_id.
- Включва оценка (rating) и дата на отзива (review_date).

Таблица "users":

- Съхранява данни за потребителите на приложението.
- Включва уникален идентификатор user_id.
- Съдържа информация като имейл (email), парола (password), роля (role) и информация за абониране (subscribed).

Таблица "wishlist":

- Представя списъка с желания на потребителите.
- Свързва потребителя с желания продукт чрез външните ключове user_id и product_id.

Таблица "orders":

- Съхранява информация за поръчките на потребителите.
- Включва уникален идентификатор order_id, свързан с потребителя чрез външния ключ user_id.
- Съдържа данни като дата на поръчката (order_date), общо количество (total_amount), статус (status), адрес за доставка (shipping_address) и тип на плащане (payment_type).

Таблица "order_items":

- Представя детайлите на продуктите във всяка поръчка.
- Включва уникален идентификатор order_item_id, свързан с поръчката чрез външния ключ order_id и с продукта чрез product_id.
- Съдържа данни като количество (quantity), размер (size), единична цена (unit_price) и обща сума (subtotal).

Таблица "products":

- Съхранява информация за продуктите в магазина.
- Включва уникален идентификатор product_id, категория (category_id), име (name), описание (description), цена (price), наличност (stock_quantity), дата на създаване

МАРК ВЕСКОВ

(created_at), дата на обновяване (updated_at), идентификатор на пола (gender_id) и URL адреси на изображения (image_urls).

Таблица "category":

- Съдържа различните категории на продуктите.
- Включва уникален идентификатор id и име на категорията (name).

Таблица "genders":

- Представа различните полове за продуктите, ако е приложимо.
- Включва уникален идентификатор id и име на пола (gender).
- Релациите между таблиците са следните:

Връзката между "reviews" и "users" е едно към много, тъй като един потребител може да напише много отзиви.

Таблицата "wishlist" е свързана с "users" и "products" чрез външните ключове, като всяка поръчка може да съдържа множество продукти.

"orders" е свързана с "users" чрез външния ключ "user_id", като всяка поръчка може да има много елементи в "order_items".

"order_items" е свързана както с "orders", така и с "products", като всяка поръчка може да съдържа много продукти и всеки продукт може да бъде част от множество поръчки.

"products" има връзки с "category" и "genders" чрез външни ключове "category_id" и "gender_id", съответно, което позволява категоризация на продуктите и указване на полове, ако е необходимо.

Тези релации осигуряват структуриран и ефективен начин за организиране и достъп до данните в базата данни.

2.2.6. Функционалност на сайта

Представеният код е част от клиентската страна на уеб приложението и има за цел да управлява интерактивността на потребителския интерфейс и да обработва данните, които потребителят избира. Използва новите функционалности на JavaScript, като стрелкови функции, за да подобри четливостта и ефективността на кода. Функциите са разделени в отделни модули, което прави кода по-структуриран и лесен за разбиране.

Променливата `currentSortingCriteria` се използва за съхранение на текущите критерии за сортиране на продуктите в приложението. Тя се използва в други функции за обработка на събития, където се нуждаем от информация за текущия режим на сортиране.

МАРК ВЕСКОВ

Функцията `getSelectedGenders()` се използва за извличане на списък с избраните полове от потребителя. Тя обхожда всички чекбоксове и добавя тези, които са отменати, към списъка с избрани полове.

Функцията `getSelectedPriceRange()` се използва за извличане на избран диапазон от цени от потребителя. Тя връща текстовото съдържание на елемента, който показва избрания от потребителя диапазон от цени.

Функцията `getSelectedCategories()` се използва за извличане на списък с избраните категории от потребителя. Тя обхожда всички чекбоксове за категории и добавя тези, които са отменати, към списъка с избрани категории.

Предназначението на този JavaScript код е да улесни потребителя при навигацията и филтрирането на продуктите в уеб приложението за модно облекло, обувки и аксесоари. Чрез тези функции потребителят може да избира желаните критерии за търсене и да филтрира резултатите, които се показват в интерфейса. Това допринася за подобряване на потребителското изживяване и прави приложението по-лесно за използване и навигация.

```
1 let currentSortingCriteria = "default";
2
3 function getSelectedGenders() {
4   const selectedGenders = [];
5   genderCheckboxes.forEach((checkbox) => {
6     if (checkbox.checked) {
7       selectedGenders.push(checkbox.value);
8     }
9   });
10  return selectedGenders;
11 }
12
13 function getSelectedPriceRange() {
14   return selectedPriceRange.textContent;
15 }
16
17 function getSelectedCategories() {
18   const selectedCategories = [];
19   categoryCheckboxes.forEach((checkbox) => {
20     if (checkbox.checked) {
21       selectedCategories.push(checkbox.value);
22     }
23   });
24   return selectedCategories;
25 }
```

МАРК ВЕСКОВ

Следващата част от JavaScript са две функции, които се използват за управление на сортирането на продуктите в уеб приложението за модно облекло, обувки и аксесоари.

Тази функция се извиква при натискане на бутона за прилагане на сортиране или при запазване на промените във филтрите. Тя първоначално извлича избраните полове (гендър), категории и ценови диапазони от потребителя, като използва предишните функции `getSelectedGenders()`, `getSelectedCategories()` и `getSelectedPriceRange()`. След това извиква функцията `fetchAndRenderSortedProducts()`, която е отговорна за извличането и рендерирането на сортираните продукти в интерфейса, като предоставя на нея текущите критерии за сортиране и филтриране.

Тази функция се извиква при натискане на бутоните за сортиране на мобилни и таблетни устройства. Тя приема параметър `order`, който представлява критерият за сортиране (например, "цена възходящо" или "цена низходящо"). Функцията променя глобалната променлива `currentSortingCriteria`, която пази текущите критерии за сортиране, и след това извиква `applySorting()`, за да приложи новите критерии за сортиране върху продуктите.

Тези функции се прилагат при натискане на бутоните за запазване на промените във филтрите и при бутоните за сортиране на мобилни и таблетни устройства. Те предоставят удобен начин за потребителите да управляват начина, по който са представени продуктите в уеб приложението, като им дават възможност да сортират и филтрират резултатите в съответствие с техните предпочитания.

```
1  function applySorting() {
2    const selectedGenders = getSelectedGenders();
3    const selectedCategories = getSelectedCategories();
4    const selectedPriceRange = getSelectedPriceRange();
5    fetchAndRenderSortedProducts(
6      currentSortingCriteria,
7      selectedGenders,
8      selectedPriceRange,
9      selectedCategories
10   );
11 }
12 function handleSorting(order) {
13   currentSortingCriteria = order;
14   applySorting();
15 }
```

МАРК ВЕСКОВ

Последната част от клиентската част отговорна за филтрирането и сортирането на продукти според различни критерии представлява функция, която се използва за извличане и рендериране на сортирани продукти.

`fetchAndRenderSortedProducts()` функцията приема няколко параметъра:

- ``order``: Критерий за сортиране на продуктите (например, "по цена" или "по популярност").
- ``selectedGenders``: Масив с избрани полове (гендър), които ще бъдат включени в заявката за сортиране на продуктите.
- ``selectedPriceRange``: Избран ценови диапазон за продуктите.
- ``selectedCategories``: Масив с избрани категории за продуктите.

Функцията генерира заявка към сървъра за сортиране на продуктите, като включва параметрите ``order``, ``selectedGenders``, ``selectedPriceRange`` и ``selectedCategories`` в URL-а на заявката. Тя създава URL с параметри, които се предават към сървъра чрез заявката за сортиране на продукти.

След това функцията изпраща заявката към сървъра чрез ``fetch()``, която връща отговор с HTML код за рендиране на сортираните продукти. Този HTML код се връща като текстов отговор на заявката.

След като получи отговор от сървъра, функцията рендира сортираните продукти в HTML блока с идентификатор ``productsBlock``, като замества съдържанието му с HTML кода за сортираните продукти. Също така функцията обновява текстовия елемент с клас ``num-products``, който показва общия брой на продуктите, които са изобразени в момента.

В случай на възникване на грешка при извличане и рендиране на продуктите, функцията хваща грешката и я записва в конзолата.

Последният ред в кода представлява извикване на функцията ``fetchAndRenderSortedProducts()`` с критерия за сортиране "default". Това означава, че при първоначалното зареждане на уеб страницата се извличат и рендират продуктите със стандартен начин на сортиране (по последователност на primary key).

МАРК ВЕСКОВ

```
1 function fetchAndRenderSortedProducts(  
2   order,  
3   selectedGenders = [],  
4   selectedPriceRange = "",  
5   selectedCategories = []  
6 ) {  
7   const queryParams = `?order=${order}${  
8     selectedGenders.length ? `&genders=${selectedGenders.join(",")}` : ""  
9   }${selectedPriceRange ? `&priceRange=${selectedPriceRange}` : ""}${  
10     selectedCategories.length  
11     ? `&categories=${selectedCategories.join(",")}`  
12     : ""  
13   }`;  
14  
15   fetch(`/products/sort${queryParams}`)  
16     .then((response) => response.text())  
17     .then((sortedProductsHTML) => {  
18       productsBlock.innerHTML = sortedProductsHTML;  
19       document.querySelector(  
20         ".num-products"  
21       ).textContent = `${productsBlock.childElementCount} total`;  
22     })  
23     .catch((error) => {  
24       console.error("Error sorting products:", error);  
25     });  
26   }  
27  
28   fetchAndRenderSortedProducts("default");
```

2.2.7. Дизайн на сървъра

Представеният код представлява контролер функция, която се извиква при опит за влизане в системата (логин) на потребител в уеб приложението.

Първоначално функцията извлича данните от заявката (`req`), по-специално имейл адреса и паролата, въведени от потребителя във формата за вход.

След това кодът проверява дали са въведени както имейл адресът, така и паролата. Ако някое от тях липсва, се връща JSON обект със статус "error" и съобщение за грешка, което посочва на потребителя да попълни и двете полета.

След това функцията прави заявка към базата данни, за да потърси потребител със зададения имейл адрес. Тя използва SQL заявка, която търси потребител в таблицата с потребители (`users`), където имейлът съвпада с въведения от потребителя.

След получаване на резултат от заявката, кодът проверява дали има резултат от заявката и дали паролата, въведена от потребителя, съвпада със записаната парола в базата

МАРК ВЕСКОВ

данни. Това се извършва чрез сравнение на хешираната парола от базата данни с паролата, която е въведена от потребителя, след като е хеширана.

След като потребителят е успешно идентифициран и паролата е вярна, функцията генерира JWT (JSON Web Token) токен, който представлява енциодирана информация за идентификацията на потребителя. Този токен съдържа уникален идентификатор на потребителя ('id') и ролята му ('role') в системата. Генерираният токен е подписан със зададения секретен ключ, осигурявайки неговата цялостност и неподправяемост.

След генерирането на токена, той се включва като съдържание на HTTP отговора, като се използва механизма на бисквитките. Това позволява на клиента (потребителя) да запази този токен и да го използва за идентификация при последващи заявки към сървъра. Бисквитката съдържа токена и обикновено е конфигурирана да изтича след определен период от време, което увеличава сигурността на приложението.

Това удостоверява потребителя и му предоставя достъп до защитени ресурси и функционалности на приложението. След успешно влизане в системата, клиентът получава отговор със статус "success" и подходящо съобщение, което потвърждава успешната идентификация и влизане в системата.

Разделянето на функционалностите в отделни модули, като например модулите за обработка на заявки и за взаимодействие с базата данни, помага за поддръжката на кода и позволява лесното добавяне на нови функционалности в бъдеще.

Тази част от кода е отговорна за обработката на потребителския вход, като осигурява сигурно и ефективно управление на идентификацията на потребителите. Използването на JWT токени предоставя сигурен механизъм за автентикация и авторизация на потребителите и позволява лесното управление на техния достъп до защитени ресурси.

Все пак, важно е да се обърне внимание на валидацията на данните, преди да бъдат изпратени към базата данни, за да се предотвратят възможни атаки, като например SQL инжекции или XSS атаки.

В цялост, този код представлява добра практика за обработка на потребителския вход и осигуряване на достъп до защитени ресурси в уеб приложение.

МАРК ВЕСКОВ

```
1  const login = async (req, res) => {
2    const { email, password } = req.body;
3    if (!email || !password)
4      return res.json({
5        status: "error", error: "Please enter your email and password",
6      });
7    else {
8      db.query(
9        "SELECT * FROM users WHERE email = ?",
10       [email],
11       async (Err, result) => {
12         if (Err) throw Err;
13         if (!result.length || !(await bcrypt.compare(password, result[0].password))
14         )
15           return res.json({
16             status: "error", error: "Incorrect email or password",
17           });
18         else {
19           const token = jwt.sign(
20             { id: result[0].id, role: result[0].role },
21             process.env.JWT_SECRET,
22             {
23               expiresIn: process.env.JWT_EXPIRES,
24             }
25           );
26           const cookieOptions = {
27             expiresIn: new Date(
28               Date.now() + process.env.COOKIE_EXPIRES * 24 * 60 * 60 * 1000
29             ),
30             httpOnly: true,
31           };
32           res.cookie("userRegistered", token, cookieOptions);
33           return res.json({
34             status: "success", success: "User has been logged in",
35           });
36         }
37       });
38     }
39   };
```

Тази middleware функция в Express.js е важен елемент от дизайна на сървъра, тъй като осигурява защита на определени ресурси на уеб приложението, като гарантира, че само автентикирани потребители имат достъп до тях. Като част от контролера, тази функция играе ключова роля за управлението на сесиите и автентикацията в приложението. Освен това помага за актуализиране на навигационната лента на уеб приложението в съответствие със статуса на потребителя, като му предоставя достъп до съответните ресурси и функционалности в зависимост от това дали е влязъл в системата или не.

Първата стъпка на функцията е да провери дали в бисквитките на текущия потребител има съхранена информация за регистрация (userRegistered). Това може да се използва за определяне дали потребителят е влязъл в системата или не.

МАРК ВЕСКОВ

Ако бисквитката `userRegistered` е налична, следващата стъпка е да се декодира JWT токенът, който е съхранен в тази бисквитка. JWT токенът съдържа информация за идентификацията на потребителя след успешно влизане в системата.

След като токенът е декодиран успешно, `middleware` функцията използва идентификатора на потребителя, получен от JWT токена, за да извлече допълнителна информация за потребителя от базата данни. Тази информация включва данни като потребителско име, електронна поща и други.

След като информацията за потребителя е успешно извлечена от базата данни, тя се добавя към обекта `req`, така че да може да бъде използвана в следващите `middleware` функции или контролери. След това се извиква функцията `next()`, която позволява на заявката да продължи към следващия `middleware` в стека или контролер.

В случай на грешка при декодиране на JWT токена или при извличане на информация за потребителя от базата данни, `middleware` функцията обработва грешката и преминава към следващия `middleware` в стека.

```
1  const loggedIn = (req, res, next) => {
2    if (!req.cookies.userRegistered) return next();
3    try {
4      const decoded = jwt.verify(
5        req.cookies.userRegistered,
6        process.env.JWT_SECRET
7      );
8      db.query(
9        "SELECT * FROM users WHERE id = ?",
10       [decoded.id],
11       (err, result) => {
12         if (err) return next();
13         req.user = result[0];
14         return next();
15       }
16     );
17   } catch (err) {
18     if (err) return next();
19   }
20 };
```

МАРК ВЕСКОВ

Следващият код представлява контролер функция, която се извиква при заявка за нулиране на паролата (forgotten password) на потребител в уеб приложението.

Първоначално функцията извлича имейл адреса от заявката (`req.body``), който е предоставен от потребителя.

След това кодът прави заявка към базата данни, за да потърси потребител със зададения имейл адрес. Той използва SQL заявка, която търси потребител в таблицата с потребители (``users``), където имейлът съвпада с въведения от потребителя (това е важен момент за проверка на идентичността).

След получаване на резултат от заявката, кодът проверява дали има потребител със съответния имейл адрес. Ако няма такъв потребител, се връща JSON обект със статус "error" и съобщение, че няма регистриран потребител с този имейл адрес (това е важно за предотвратяване на потенциални атаки, свързани със сигурността на потребителските данни.).

Ако има потребител със зададения имейл адрес, кодът генерира временен токен, който се използва за нулиране на паролата. Този токен е валиден за определено време (15 минути) и се създава чрез функцията ``jwt.sign()`` със секретен ключ, който включва и хеша на паролата на потребителя. Създава се и уникален линк, който съдържа този токен и идентификатора на потребителя.

След това кодът създава HTML имейл съобщение, което съдържа линк към страницата за нулиране на паролата. Този линк съдържа идентификационния номер на потребителя и генерирания токен. След това се изпраща имейл до потребителя с помощта на ``transporter.sendMail()``, който е конфигуриран предварително за изпращане на имейли.

Накрая, в зависимост от успеха или неуспеха при изпращането на имейла с линк за нулиране на паролата, се връща JSON обект със съответния статус и съобщение, което информира потребителя за резултата от операцията.

В заключение контролер функцията, която се използва за нулиране на паролата (forgotten password), представлява важна част от сигурността и функционалността на уеб приложението. Тя осигурява възможност за възстановяване на паролата за потребители, които са забравили своята парола или са изгубили достъп до нея.

МАРК ВЕСКОВ

```
1 router.post("/forgot-password", async (req, res, next) => {
2   const JWT_SECRET = process.env.JWT_SECRET;
3   const { email } = req.body;
4   db.query(
5     "SELECT * FROM users WHERE email = ?", [email],
6     async (err, result) => {
7       if (err) {
8         return res.json({status: "error", error: "Internal server error",});
9       }
10      if (!result.length) {
11        return res.json({status: "error", error: "There is no user with this e-mail",});
12      }
13
14      const user = result[0];
15      const secret = JWT_SECRET + user.password;
16      const payload = {email: user.email, id: user.id,};
17
18      const token = jwt.sign(payload, secret, { expiresIn: "15m" });
19      const link = `http://localhost:3000/reset-password/${user.id}/${token}`;
20      const mailOptions = {
21        from: process.env.EMAIL_EMAIL,
22        to: email,
23        subject: "Password Reset | Casspie",
24        html: `
25          <h3>Your password reset link (valid for 15 minutes)</h3>
26          <p><span>Reset password: <a href="${link}">Click here</a></span></p>`,
27      };
28
29      transporter.sendMail(mailOptions, (error, info) => {
30        if (error) {
31          return res.json({
32            status: "error", error: "Error sending reset password email",
33          });
34        } else {
35          return res.json({
36            status: "success", success: "Reset password email sent successfully",
37          });
38        }
39      });
40    });
41  });
```

2.2.8. CRUD заявки

В процеса на създаване на това приложение са използвани множество CRUD заявки, които са от съществено значение за манипулиране на данните в базата данни. CRUD заявките са отразени на множество места в приложението, като играят ключова роля в осигуряването на функционалност за създаване (Create), четене (Read), обновяване (Update) и изтриване (Delete) на данни. Тези заявки не само позволяват на потребителите да взаимодействат с данните, но и осигуряват стабилност и сигурност на системата, като гарантират правилното управление на информацията. Освен това, използването на CRUD заявки допринася за подобряване на ефективността на разработката и поддръжката на приложението, като предоставя ясен и структуриран начин за манипулиране на данните. Те

МАРК ВЕСКОВ

също така играят важна роля в обезпечаването на сигурността на данните чрез прилагане на различни нива на разрешения и проверки за валидност, като се гарантира, че достъпът до информацията е под контрол и е в съответствие със зададените правила и изисквания.

За пример може да се вземе следната заявка от тип POST, която има за цел добавянето на нов продукт към базата данни и предоставя удобство за администраторите чрез предоставяне на форма за въвеждане на данни. Това значително улеснява процеса на добавяне на нови артикули към продуктовия каталог, като предоставя алтернатива на ръчното добавяне чрез писане на код и внасяне на промени в страниците на уебсайта. Такъв подход осигурява динамично обогатяване на съдържанието на сайта, като позволява лесно добавяне на нова информация към базата данни и нейното незабавно отразяване върху уебсайта.

Първата част на кода извлича данните от тялото на заявката (req.body), които са необходими за създаване на новия продукт. Тези данни включват име на продукта, URL адреси на изображения, идентификатор на категория, идентификатор на пол, описание, цена и наличност на склад.

След това се конструира SQL заявка за добавяне на нов запис в таблицата за продукти. Тази заявка включва колоните, които трябва да бъдат попълнени със съответните данни, предоставени от потребителя.

Следващата част от кода използва db.query() за изпълнение на SQL заявката за добавяне на новия продукт в базата данни. След успешно добавяне, потребителят се пренасочва към страницата за профил (/profile).

Тази заявка осигурява възможност за добавяне на нови продукти към продуктовия каталог. Чрез POST метода, приложението може да комуникира със сървъра и да добавя нови записи в базата данни, което е от съществено значение за управлението на асортимента от артикули. Тази операция отговаря на функцията за създаване (Create) в модела CRUD.

МАРК ВЕСКОВ

```
1 router.post("/add-product", (req, res) => {
2   const {
3     name,
4     image_urls,
5     category_id,
6     gender_id,
7     description,
8     price,
9     stock_quantity,
10  } = req.body;
11
12  const insertSql = `
13    INSERT INTO products (name, image_urls, category_id, gender_id,
14    description, price, stock_quantity)
15    VALUES (?, ?, ?, ?, ?, ?, ?)
16  `;
17  db.query(
18    insertSql,
19    [
20      name,
21      image_urls,
22      category_id,
23      gender_id,
24      description,
25      price,
26      stock_quantity,
27    ],
28    (err, result) => {
29      if (err) throw err;
30
31      res.redirect("/profile");
32    }
33  );
34 });
```

Заявката, която се разглежда, е пример за CRUD операция от тип SELECT, която е основна операция върху бази от данни и представлява извличане на данни.

Тази заявка има за цел да предостави началната страница на уеб-приложението, като показва последно добавените продукти. Приложението изисква информация за продуктите и техните категории, за да може потребителите да разглеждат и пазаруват от широката гама артикули.

Преди да бъде показана началната страница, маршрутът проверява дали потребителят е влязъл в системата. Това се осъществява чрез middleware функцията `loggedIn`, която обезпечава защита на определени ресурси, достъпни само за влезли потребители. Това е

МАРК ВЕСКОВ

важна мярка за сигурност, която гарантира, че само упълномощени потребители имат достъп до данните на приложението.

Заявката използва структуриран език за заявки (SQL), за да извлече информация за последно добавените продукти от базата данни. Това включва данни като име на продукта, цена, описание и категория.

След успешно извличане на информацията от базата данни, заявката рендерира HTML шаблона на началната страница. В зависимост от състоянието на влизане на потребителя, се визуализира съответната информация за потребителя (ако е влязъл в системата) и списък с последно добавените продукти (виж Фиг.).

Тази заявка използва Express за създаване на маршрут, който обработва GET заявки към път `"/index"`. След като изпълни необходимата логика, тя рендерира HTML шаблона за началната страница на приложението.

Тази SELECT заявка е от съществено значение за функционирането на уеб-приложението, тъй като осигурява необходимите данни за показване на началната страница на потребителите. Тя отговаря на операцията за четене (Read) в модела CRUD и е ключов елемент от приложението за управление на данни.

Частта от заявката на ред 3 определя колоните, които ще бъдат извлечени от базата данни. `products.*` означава, че се избират всички колони от таблицата `"products"`, а `category.name AS category_name` добавя името на категорията към резултата и я кръщава като `"category_name"`. `FROM products:` Тук се посочва таблицата, от която ще бъдат извлечени данните, в случая `"products"`. Ред 5 представлява операция за свързване на данните от две таблици. Тук се извличат данни от таблицата `"category"`, като се използва връзка между `"category_id"` в таблицата `"products"` и `"id"` в таблицата `"category"`. Чрез ред 6 резултатите се сортират по време на създаване на продуктите в низходящ ред, за да се покажат последно добавените продукти първи. `LIMIT 4:` Ограничава броя на резултатите до първите 4 записа, което е полезно, когато искаме да покажем само няколко от последно добавените продукти.

МАРК ВЕСКОВ

```
1 router.get("/index", loggedIn, (req, res) => {
2   const query = `
3     SELECT products.*, category.name AS category_name
4     FROM products
5     INNER JOIN category ON products.category_id = category.id
6     ORDER BY products.created_at DESC
7     LIMIT 4
8   `;
9   if (req.user) {
10    db.query(query, (err, results) => {
11      if (err) throw err;
12      res.render("index", {
13        status: "loggedIn",
14        user: req.user,
15        products: results,
16      });
17    });
18  } else {
19    db.query(query, (err, results) => {
20      if (err) throw err;
21      res.render("index", { status: "no", user: "nothing", products: results
22    });
23  }
24 });
```

Тази заявка е от тип POST и има за цел да актуализира статуса на поръчка в базата данни. Приложението използва тази заявка, за да промени статуса на поръчка спрямо новото състояние, което е предоставено от администратора.

Вторият и третият ред на кода извличат параметрите от заявката, които са необходими за актуализацията на статуса на поръчката. Тези параметри са идентификаторът на поръчката (orderId) и новият статус (newStatus).

След това заявката използва SQL операцията UPDATE, за да промени статуса на поръчката в базата данни. Тя използва подадените параметри orderId и newStatus за да намери и актуализира съответната поръчка.

В случай на възникване на грешка при изпълнението на заявката, кодът извежда съобщение за грешка в конзолата и връща отговор със статус 500 (вътрешна грешка). Това осигурява удобен начин за откриване и отстраняване на проблемите с актуализацията на статуса на поръчката.

МАРК ВЕСКОВ

Ако актуализацията на статуса на поръчката премине успешно, се извежда съобщение за успешно актуализиране на статуса и се връща отговор със статус 200 (OK).

Тази заявка представлява важна част от функционалността на уеб-приложението, тъй като осигурява възможност за актуализиране на статуса на поръчка. Чрез POST метода, приложението може да комуникира със сървъра и да променя данните в базата данни, което е от съществено значение за управлението на поръчките. Тази операция отговаря на функцията за актуализиране (Update) в модела CRUD.

```
1 router.post("/updateStatus", (req, res) => {
2   const orderId = req.query.orderId;
3   const newStatus = req.query.newStatus;
4
5   const sql = "UPDATE orders SET status = ? WHERE order_id = ?";
6   db.query(sql, [newStatus, orderId], (err, result) => {
7     if (err) {
8       console.error("Error updating status:", err);
9       res.status(500).send("Error updating status");
10      return;
11    }
12    console.log("Status updated successfully");
13    res.sendStatus(200);
14  });
15 });
16
```

Тази заявка е от тип POST и има за цел да изтрие продукт от базата данни. Приложението използва тази заявка, за да позволи на администраторите да изтриват продукти от продуктовия каталог.

Първата част на кода извлича данните от тялото на заявката (req.body), което съдържа идентификатора на продукта (product_id), който трябва да бъде изтрит.

След това се конструира SQL заявка за изтриване на продукта от таблицата за продукти. Тази заявка се основава на предоставения идентификатор на продукта и е предназначена за изтриване на съответния запис от базата данни.

Следващата част от кода използва db.query() за изпълнение на SQL заявката за изтриване на продукта от базата данни. След успешно изтриване, потребителят се пренасочва към страницата за профил (/profile).

МАРК ВЕСКОВ

Тази заявка осигурява възможност за изтриване на продукти от продуктовия каталог. Чрез POST метода, приложението може да комуникира със сървъра и да изтрива записи от базата данни. Тази операция отговаря на функцията за изтриване (Delete) в модела CRUD и е ключов елемент от приложението за управление на данни.

```
1 router.post("/delete-product", loggedIn, (req, res) =>
  {
2   const { product_id } = req.body;
3
4   const deleteSql = `
5   DELETE FROM products WHERE product_id = ?;
6   `;
7
8   db.query(deleteSql, [product_id], (err, result) => {
9     if (err) throw err;
10
11     res.redirect("/profile");
12   });
13 });
```

3. Заключение

3.1. Постигнати цели

В хода на разработката на приложението бяха постигнати няколко ключови цели, които значително подобриха употребата и функционалността на системата както за потребителите, така и за администраторите. Всички зададени цели и функционалности бяха изпълнени успешно.

Първо и най-важно, беше създаден интуитивен графичен интерфейс, който прави навигацията лесна и приятна за потребителите на приложението. Графичният дизайн е разработен със специално внимание към потребителското изживяване, като стремежът е насочен към това да се предостави удобство и ефективност при използване на системата. Също така, интерфейсът за администраторите е оптимизиран за управление на данните и функциите на приложението, като осигурява лесен достъп до необходимата информация и инструменти за администрация.

Освен това, беше реализирана функционалност, при която не е необходимо намеса от страна на разработчиците при добавяне на нови продукти в системата. Този аспект е от

МАРК ВЕСКОВ

ключово значение за непрекъснатото функциониране на уебсайта и улеснява администрирането на приложението. Благодарение на автоматизирания процес за добавяне на продукти, администраторите могат бързо и лесно да обновяват асортимента си, без да се налага да прекъсват работата на уебсайта или да изискват помощ от разработчиците.

В крайна сметка, постигнатите цели в приложението не само подобриха изживяването на потребителите, но и улесниха администрирането и поддръжката на системата.

3.2. Перспективи за развитие

В развитието на приложението се откриват множество перспективи за бъдещо разширение и подобрение, които биха допринесли за по-голямото удовлетворение на потребителите и повишение на ефективността на системата.

Една от ключовите перспективи за развитие е създаването на мобилно приложение. Тази идея ще допринесе за увеличаване на достъпността на приложението, като потребителите ще могат да се възползват от него по всяко време и от всяко място, дори когато не са пред компютър или лаптоп. Мобилното приложение ще осигури по-голяма гъвкавост и удобство за потребителите, които са в движение или предпочитат да използват устройства като смартфони и таблети.

Освен това, може да се разгледа възможността за добавяне на нови функционалности и подобрения в съществуващата система. Това включва разработване на допълнителни модули за управление на данни, разширяване на функционалността за анализ и статистика, както и внедряване на нови методи за комуникация и взаимодействие с потребителите.

Въвеждането на персонализирани промоции и купони, подобряване на системата за препоръки и разработване на допълнителни алгоритми за филтриране на продуктите са още някои от идеите за развитие, които биха подкрепили растежа на приложението и увеличили ангажираността на потребителите.

Допълнително, може да се разгледа възможността за интеграция с външни системи и платформи, като например социални медии или други доставчици на услуги за доставка. Това ще разшири функционалността на приложението и ще го направи още по-полезно за потребителите.

В крайна сметка, непрекъснатото развитие и иновации в приложението са от съществено значение за запазване на конкурентното му предимство и задържане на лоялността на потребителите. Възползвайки се от възможностите за разширение и подобрение, би могло да

МАРК ВЕСКОВ

бъде създаден по-добър и по-изпълнителен продукт, който отговаря на нуждите и очакванията на аудиторията.

4. Източници

<https://advanceacademy.bg/blog/programirane-s-javascript-4-neshta-koito-mozhem-da-napravim-s-javascript>

<https://dev.mysql.com/doc/refman/8.0/en/tutorial.html>

<https://ejs.co/#docs>

https://en.wikipedia.org/wiki/Database_design

<https://en.wikipedia.org/wiki/XAMPP>

<https://expressjs.com/>

<https://getbootstrap.com/>

<https://www.geeksforgeeks.org/sql-server-crud-operations>

<https://www.lucidchart.com/pages/data-flow-diagram>

<https://www.programiz.com/javascript/ES6>

<https://softuni.bg/blog/what-is-model-view-controller-architecture-pattern>

<https://softuni.bg/blog/what-to-know-about-mysql>

https://www.tutorialspoint.com/nodejs/nodejs_express_framework.htm

<https://www.tutorialsteacher.com/mvc>

<https://windpoly.run/posts/mvc-and-mvp/>

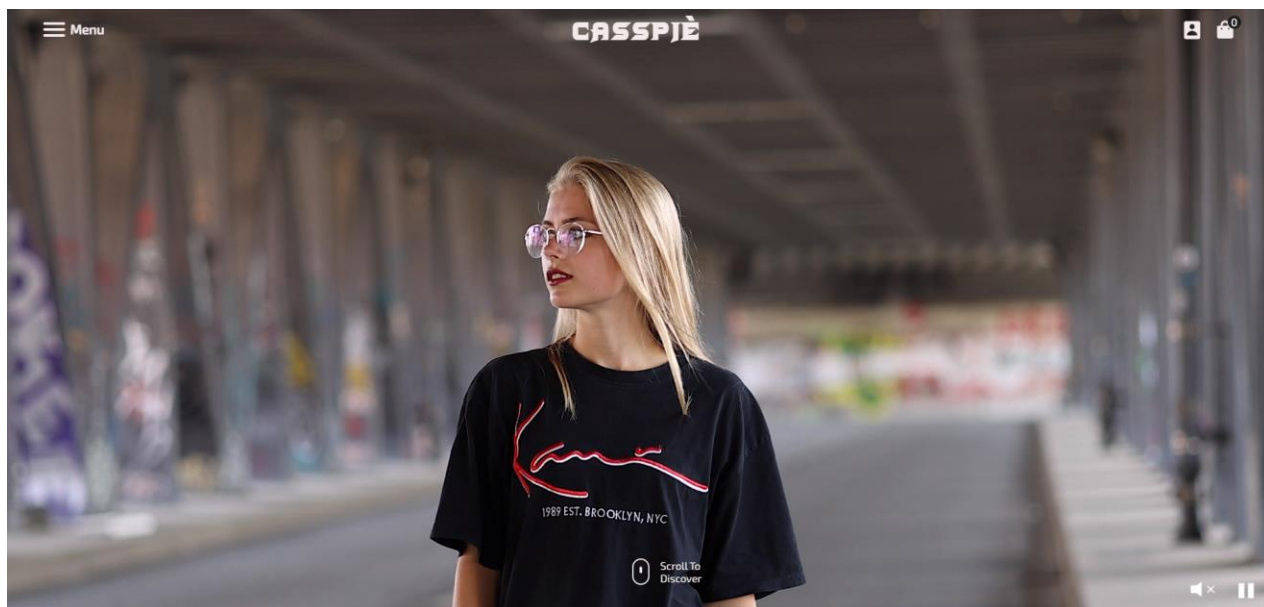
<https://www.w3schools.com/sql/>

https://www.youtube.com/watch?v=7S_tz1z_5bA

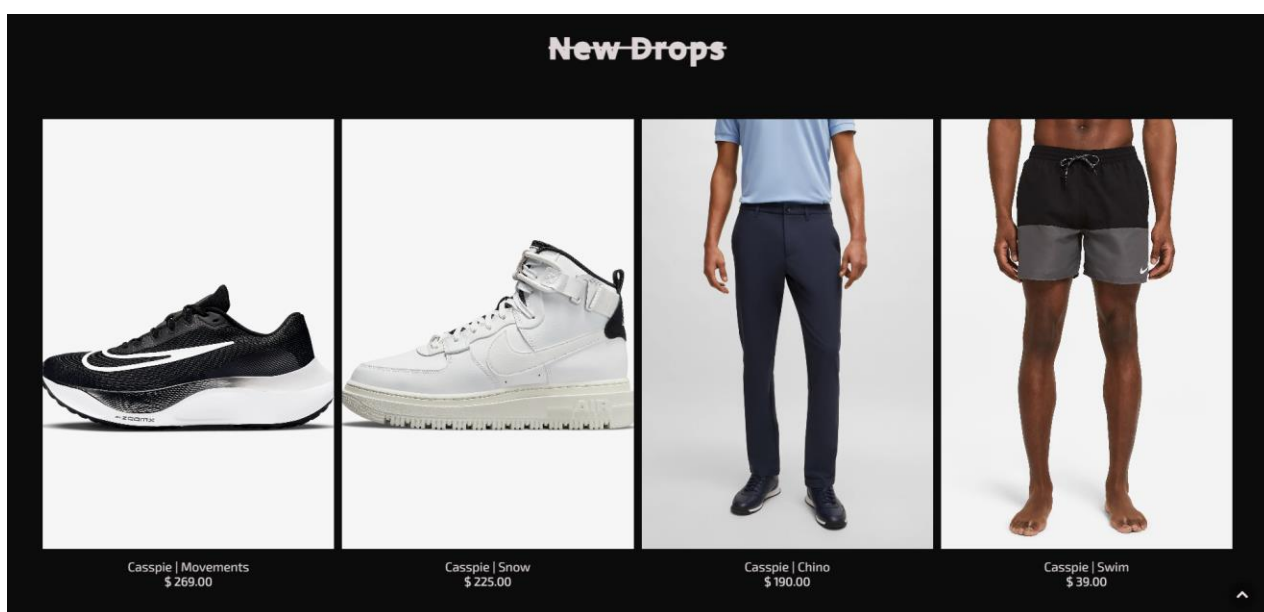
<https://www.youtube.com/watch?v=-MTSQjw5DrM&t=253s>

<https://www.youtube.com/watch?v=nH9E25nkk3I&t=11s>

5. Приложение

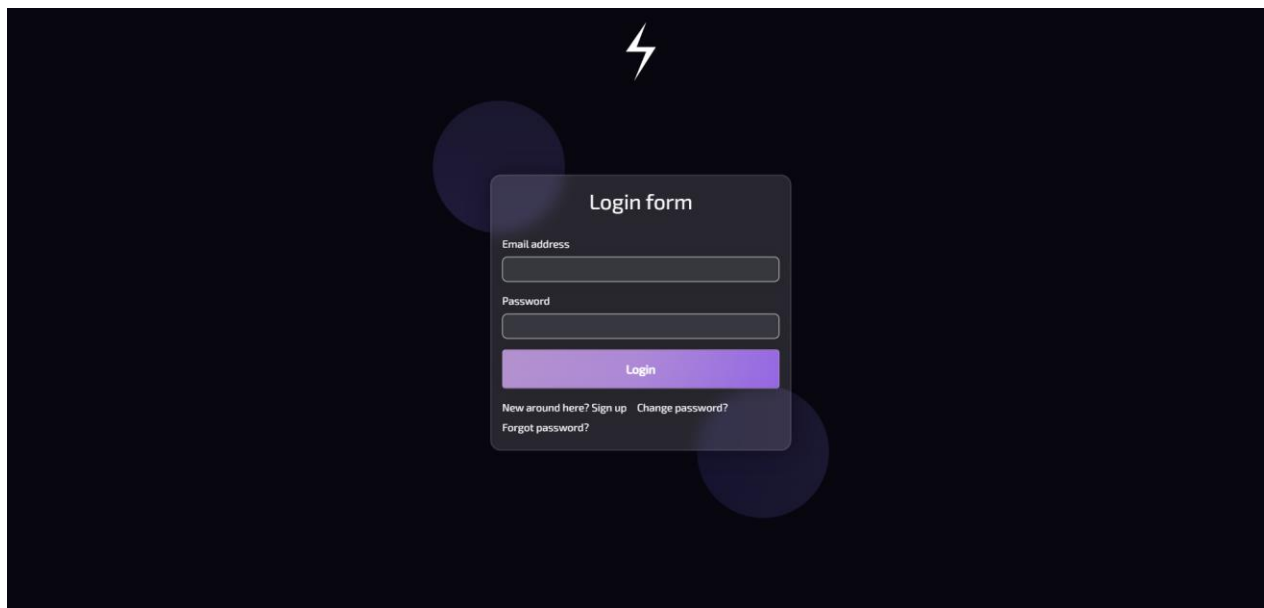


5.1. Начална страница в уебсайта

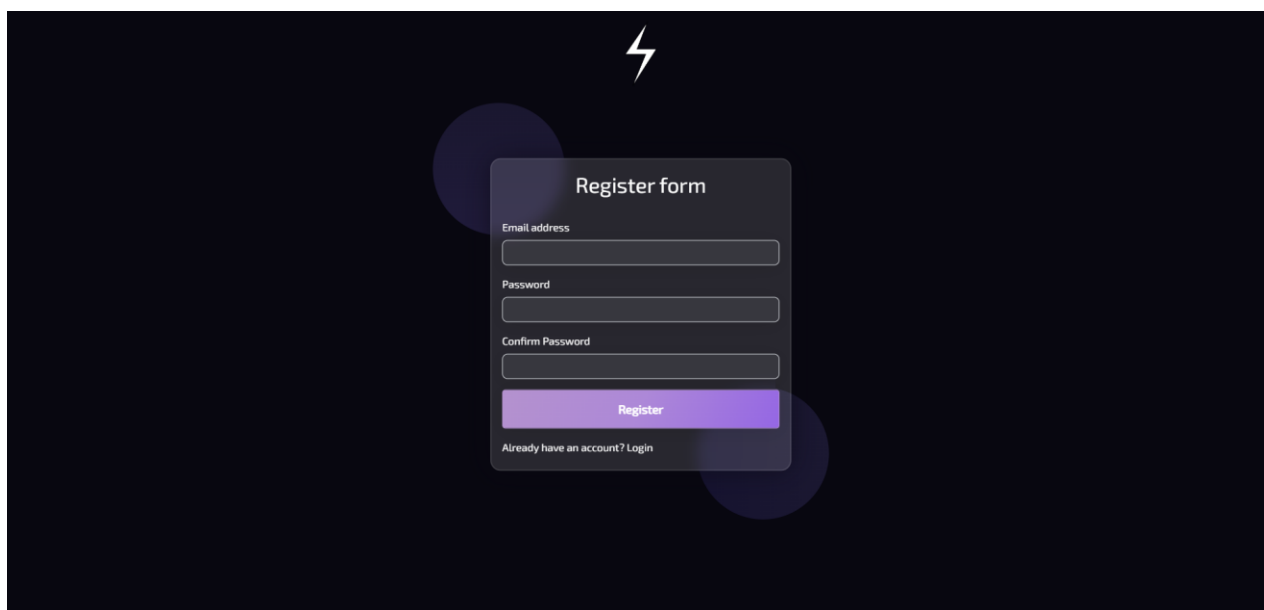


5.2. Секция от начална страница - последно добавени продукти

МАРК ВЕСКОВ

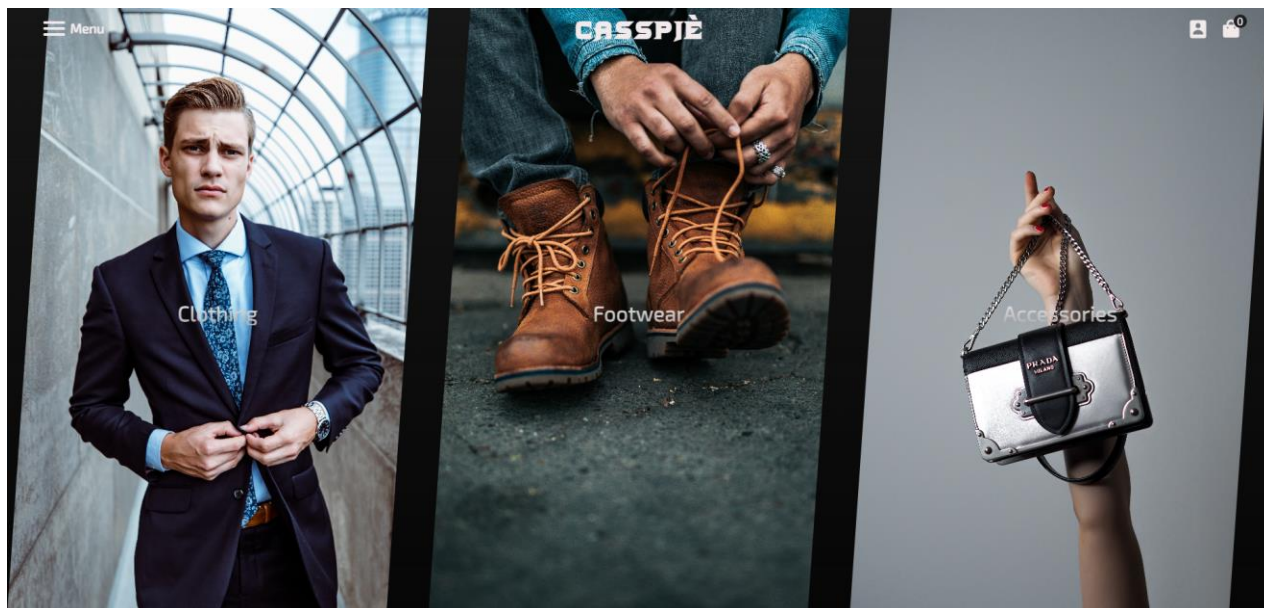


5.3. Страница за вход в уебсайта

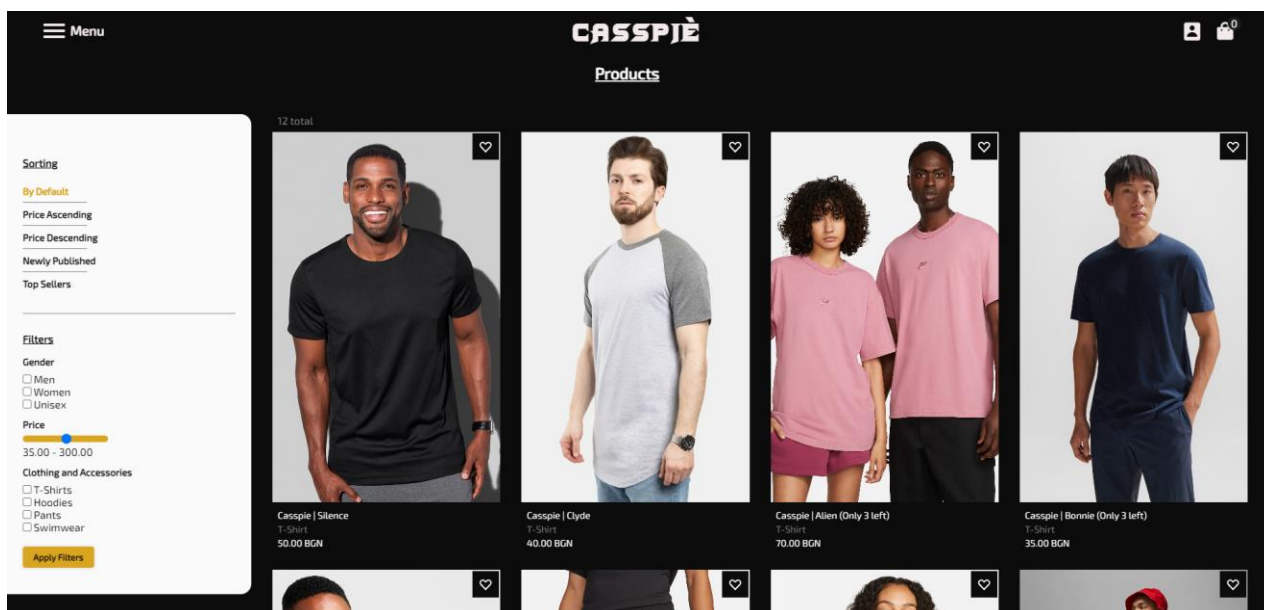


5.4. Страница за регистрация в уебсайта

МАРК ВЕСКОВ

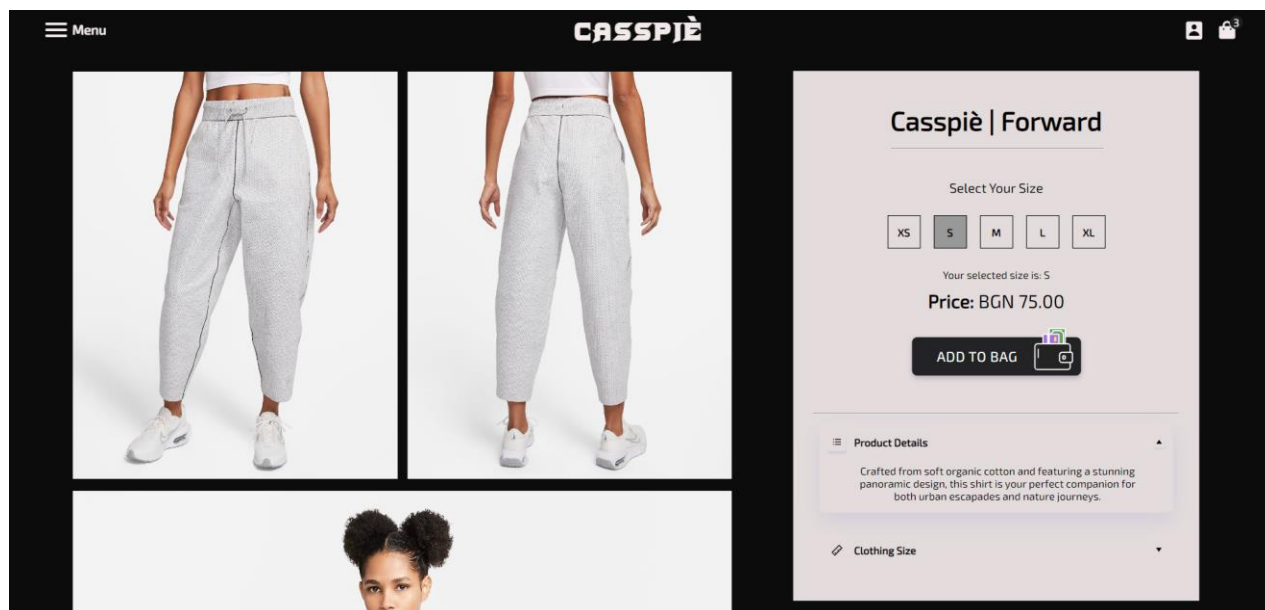


5.5. Страница Products - страница с артикулите, разделени по категории

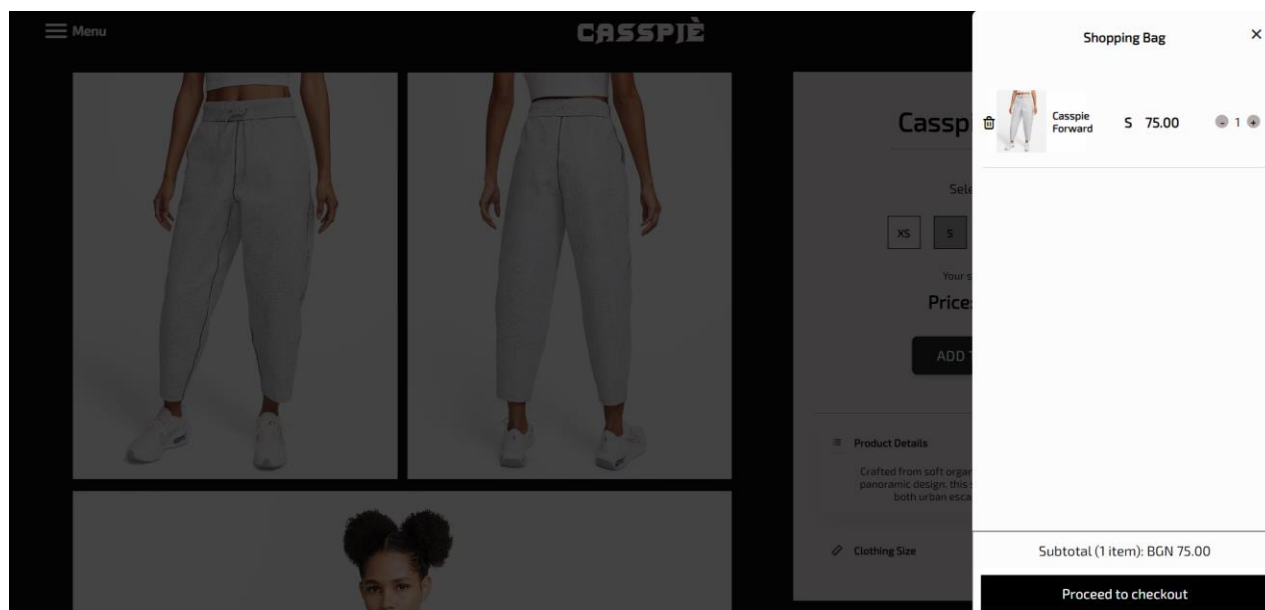


5.6. Страница Clothing - страница с продуктите от категория "clothing"

МАРК ВЕСКОВ



5.7. Страница на конкретен продукт



5.8. Изглед на количка след добавяне на продукт

МАРК ВЕСКОВ

CASSPIE

Order Form

Order to Office

First name
Mark

Last name
Veskov

Email
example@email.com

Phone
088 888 8888

Additional Information

Моля, добавете писмо с надпис - "Честит рожден ден, Пепи!"

Office:

Моля, изберете офис на куриер

Айтос Орела (Айтос ул. Васил Левски №44)

ЕКОНТ

Айдемир (Айдемир кв. Деленките ул. София №7)

ЕКОНТ

Айтос Автогара (Айтос ул. Хаджи Димитър №31)

ЕКОНТ


Айтос Орела (Айтос ул. Васил Левски №44)

ЕКОНТ

Айтос Орела (Айтос ул. Васил Левски №44)

Order to Address

Proceed to Checkout




Casspie Forward

Size: S

Price: 75.00

Units: 1

Remove



Casspie Silence

Size: XS

Price: 50.00

Units: 2

Remove

Total sum: BGN 175.00

5.9. Страница Checkout - страница за поръчка (при стъпка 1 – въвеждане на адрес)

CASSPIE

← Shipping Details

Delivery type: Office

First name: Mark | Last name: Veskov

Email: example@email.com | Phone: 088 888 8888

Office: Айтос Орела (Айтос ул. Васил Левски №44)

Additional Information: Моля, добавете писмо с надпис - "Честит рожден ден, Пепи!"

Payment on Delivery


Pay with Card

Payment via Bank

Банкова Сметка: BG0657SA9888888888888

BIC: STSABG5F Банка ДСК ЕАД

Finalize Purchase




Casspie Forward

Size: S

Price: 75.00

Units: 1

Remove



Casspie Silence

Size: XS

Price: 50.00

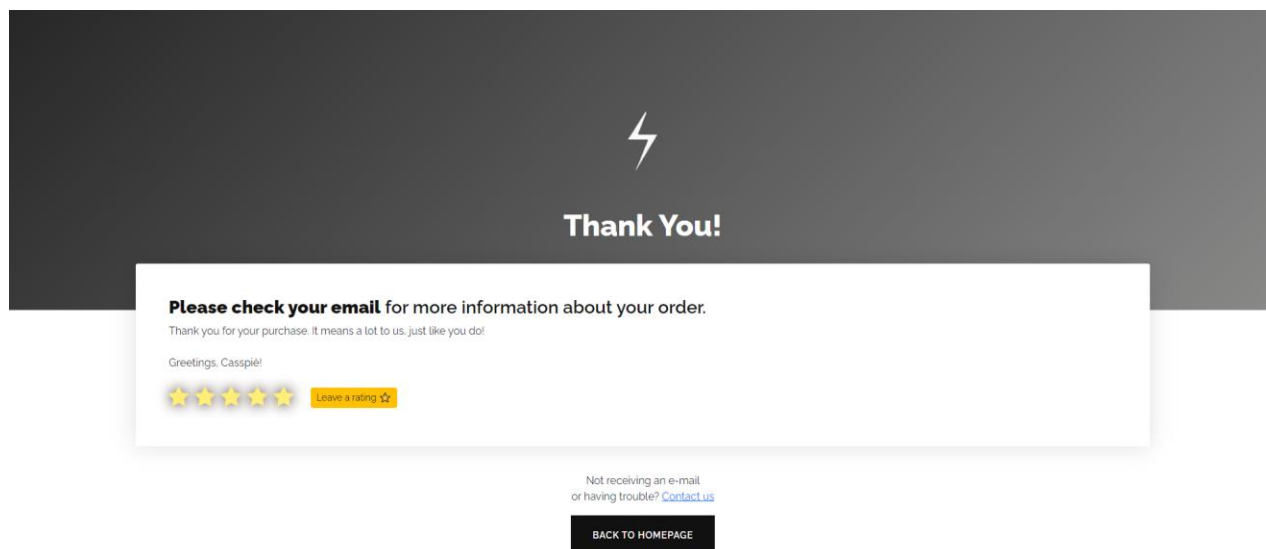
Units: 2

Remove

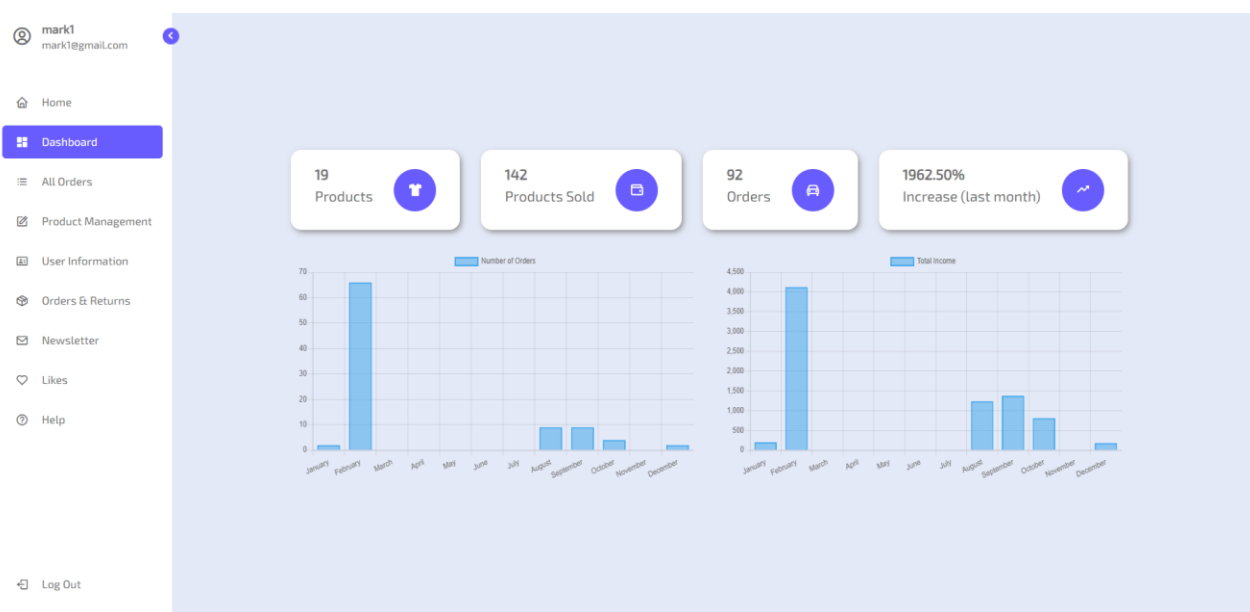
Total sum: BGN 175.00

5.10. Страница Checkout - страница за поръчка (при стъпка 2 – избиране на разплащателен метод)

МАРК БЕСКОВ



5.11. Страница след завършване на поръчка



5.12. Профилна страница - администраторско меню за следене на статистики

МАРК ВЕСКОВ

Export Data in PDF

Order ID	User ID	Order Date GMT+0300	Total Amount	Status	Shipping Address	Payment Type
4	3	Thu Aug 31 2023 11:30:51	BGN 80.00	delivered	Айтос Автогара (Айтос ул. Хаджи Димитър №31)	on_delivery
5	3	Thu Aug 31 2023 11:32:33	BGN 195.00	processing	Айтос Автогара (Айтос ул. Хаджи Димитър №31)	via_bank
6	3	Thu Aug 31 2023 12:22:29	BGN 240.00	processing	Айтос Автогара (Айтос ул. Хаджи Димитър №31)	via_card
7	3	Thu Aug 31 2023 12:24:20	BGN 55.00	processing	Аксаково (Аксаково ж.к. Надежда ул. Капитан Петко Войвода №1)	via_card
20	3	Thu Aug 31 2023 12:47:31	BGN 55.00	processing	Айтос Орела (Айтос ул. Васил Левски №44)	via_card
21	3	Thu Aug 31 2023 12:48:52	BGN 55.00	processing	Айтос Автогара (Айтос ул. Хаджи Димитър №31)	via_card
22	3	Thu Aug 31 2023 12:49:57	BGN 110.00	processing	Айтос Автогара (Айтос ул. Хаджи Димитър №31)	via_card
23	3	Thu Aug 31 2023 12:52:51	BGN 90.00	processing	Айдемир (Айдемир кв. Деленките ул. София №7)	via_card
24	3	Thu Aug 31 2023 16:36:14	BGN 355.00	processing	1592, Sofia, ulitza, 5049	on_delivery
25	3	Fri Sep 01 2023 16:11:04	BGN 80.00	processing	Айтос Орела (Айтос ул. Васил Левски №44)	on_delivery
34	4	Mon Sep 04 2023 00:58:05	BGN 110.00	processing	Бургас 24/7 Еконтмат- МОЛ Бургас Плаза (Бургас бул. Транспортна №31)	on_delivery
35	0	Mon Sep 04 2023 17:28:38	BGN 320.00	processing	София (София жк Хаджи Димитър ул. Резбарска №11 След магазин Кауфланд до гаран Малашевци)	via_card

5.13. Профилна страница - администраторска таблица с всички извършени поръчки

Name

Image Uri1

Image Uri2

Image Uri3

Category

T-Shirt

Gender

men

Description

Price

Stock Quantity

Add Product

Product ID

Delete Product

Product ID

Name

Image Uri1

Image Uri2

Image Uri3

Category

T-Shirt

Gender

men

Description

Price

Stock Quantity

Edit Product

5.14. Профилна страница - администраторски полета за добавяне, актуализиране и изтриване на артикули

МАРК ВЕСКОВ

My Profile

Username

mark1

E-mail

mark1@gmail.com

E-mail is used for username.

Password

To change your password, navigate to login page and click on change password button.

5.15. Профилна страница - информация за потребителят

Date of order placement: Thu Aug 31 2023 11:30:51 GMT+0300 (Eastern European Summer Time) | Number of products: 1
Status: delivered

Casspie | Joy

Quantity: 1
BGN 80.00
women Hoodie
Size M

Delivery	Айтос Автогара (Айтос ул. Хаджи Димитър №31)
Payment	on_delivery
Summary	
Subtotal	80.00
Delivery	BGN 0
Total	80.00

Date of order placement: Thu Aug 31 2023 11:32:33 GMT+0300 (Eastern European Summer Time) | Number of products: 4
Status: processing

Casspie | Ice

Quantity: 1
BGN 60.00
men Pants
Size M

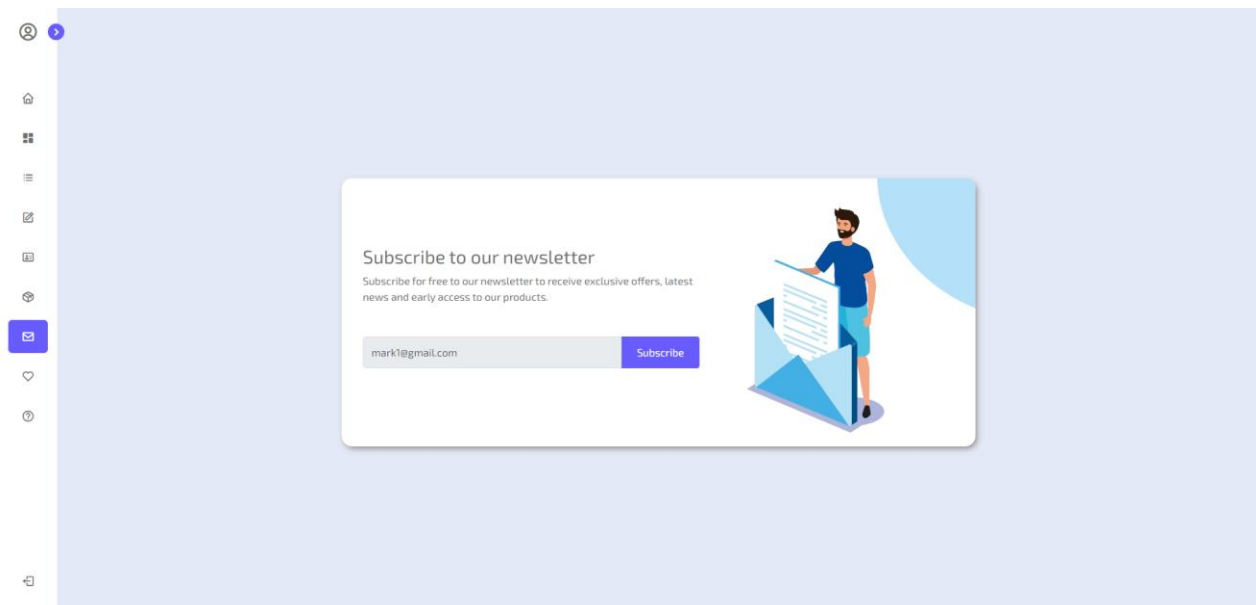
Casspie | Summer

Quantity: 2
BGN 45.00
women Sandals

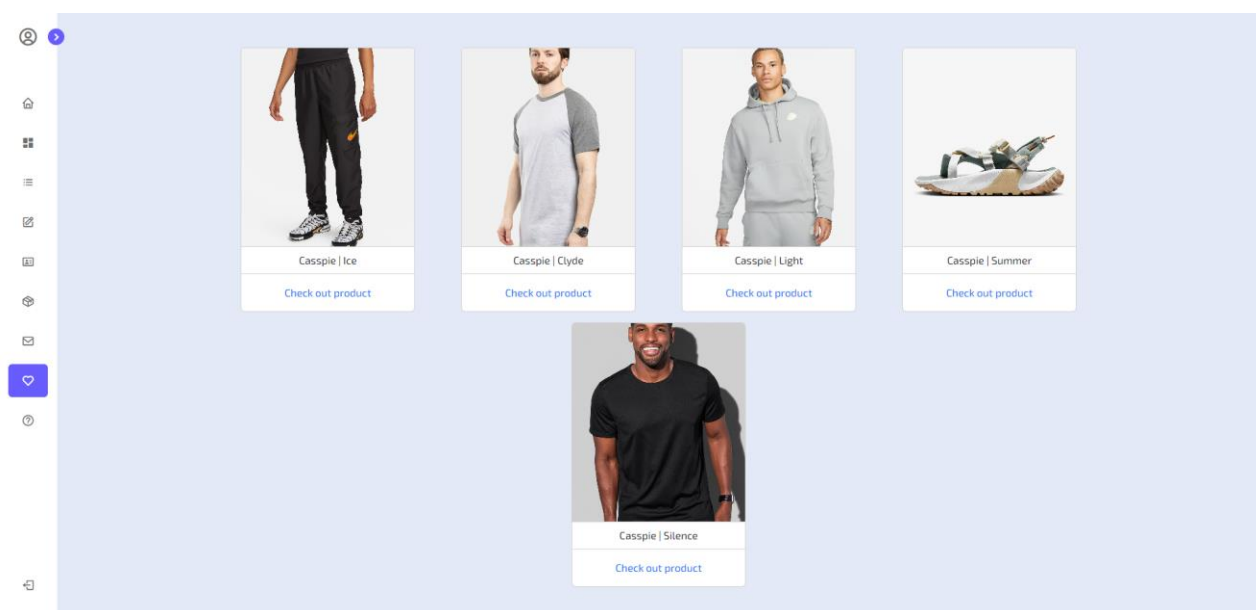
Delivery	Айтос Автогара (Айтос ул. Хаджи Димитър №31)
Payment	via_bank
Summary	
Subtotal	195.00
Delivery	BGN 0
Total	195.00

5.16. Профилна страница – информация за извършените поръчки

МАРК ВЕСКОВ



5.17. Профилна страница – записване на e-mail бюлетин



5.18. Профилна страница - изглед на любимите продукти