МИНИСТЕРСТВО НА ОБРАЗОВАНИЕТО И НАУКАТА

ПРОФЕСИОНАЛНА ГИМНАЗИЯ ПО ЕЛЕКТРОТЕХНИКА И ЕЛЕКТРОНИКА
бул. Пещерско шосе № 26
4002 гр. Пловдив, България
тел. 032 / 643-657
info-1690174@edu.mon.bg

VOCATIONAL SCHOOL OF ELECTRICAL ENGINEERING AND ELECTRONICS
26 Peshtersko chaussee blvd.,
4002 Plovdiv, Bulgaria
Phone: 032 / 643-657
info-1690174@edu.mon.bg

професия код 481030 „Приложен програмист“

специалност код 4810301 „Приложно програмиране“

ДИПЛОМЕН ПРОЕКТ

за придобиване трета степен професионална квалификация

ТЕМА: ОНЛАЙН МАГАЗИН ЗА МОДА

Дипломант: …… /Стилиян Димитров/

Ръководител-консултант: …… /Катя Семкова/

Клас: 12б

e-mail: sdimitrov482@gmail.com

Пловдив

2024 година

СЪДЪРЖАНИЕ

УВОД

С разрастващия се свят на онлайн търговията, изграждането на ефективен и привлекателен уебсайт за модни дрехи става не просто важен, а основен елемент от стратегията за успех. "The Fashion World" не е просто едно място за пазаруване, а по-скоро виртуална платформа, която обединява света на модата с едно кликване.

Целта на този проект е създаването на уебсайт и електронен магазин за модни дрехи, обувки и аксесоари с наименование "The Fashion World". Проектът се стреми да предложи на своите клиенти не само широка гама от продукти, но и удобен достъп до тях чрез продуктови категории, за да се улесни търсенето на желаните артикули.

За да постигне своята цел, проектът разработва основните функционалности на уеб базиран електронен магазин, като същевременно осигурява достъп посредством администраторски и потребителски профил. Това включва работа с асортимента, като добавяне, редактиране и премахване на продукти, както и управление на поръчките и обслужване на клиенти. Проектът предлага удобен потребителски интерфейс за работа с артикулите – категоризация, система за търсене и филтриране.

Настоящата теоретична разработка проучва и анализира технологиите, необходими в процеса на имплементиране на основните функционалности на "The Fashion World", прави разбор за тяхното приложение в практиката. Това включва избор на подходящи езици за програмиране, инструменти за уеб разработка и бази данни

В първа глава на разработката се разглеждат теоретичните аспекти, на които тя е базирана. Осъществен е преглед на основните технологии, необходими за изграждането на уебсайта . Проектът изисква използването на съвременни езици за програмиране като HTML, CSS и JavaScript за разработка на клиентската част на уебсайта, както и използването на езици и технологии като PHP за създаване на сървърната част и обработка на заявките на потребителите. За управление на базата данни се използва PHP MyAdmin. Тези технологии и инструменти са проучени и са анализирани техните предимства и недостатъци. Разгледано е приложението им при изграждане на функционалностите на уебсайта "The Fashion World".

Във втора глава на разработката е разкрита основната структура на приложението. В тази глава се разглежда организацията на клиентската и сървърната части на приложението, както и взаимодействието между тях. Клиентската част включва потребителски интерфейс, който предоставя на потребителите удобен и интуитивен начин за навигация и избор на модни артикули. Тази част е разработена с HTML, CSS и JavaScript.

Сървърната част на приложението обработва заявките на потребителите, управлява връзката с базата данни. Тя е бъде разработена с помощта на сървърния език PHP, като за управление на базата данни се използва PHP MyAdmin. Сървърната част осигурява възможност за регистрация и вход на потребители, обработка на поръчки и управление на продуктовия каталог. Разгледани са възможности за интеграция с разплащателни системи , за да се осигури удобен и сигурен начин за плащане и доставка на поръчките.

В трета глава на разработката, наречена „Реализация“, е описана имплементацията на уеб приложението. Разгледани са създаване на потребителски и администраторски профили, управление на правата на потребителите. Показана е реализацията на избор на продукти, добавяне в количка и ще избор на метод за плащане.

В четвърта глава на разработката наречена „Ръководство на потребителя“, са описани дейностите на администратора, манипулации с продукти и поръчки. Включени са указания за работата му при управление на клиентските акаунти.

ОСНОВНА ЧАСТ

ГЛАВА 1 ИЗПОЛЗВАНИ ТЕХНОЛОГИИ

1.1. Използвани технологии и езици при разработка на уеб базирано приложение

Разработването на уеб приложение за управление на магазин за дрехи изисква използване на различни технологии както във frontend (клиентската страна), така и в backend (сървърната страна).

1.1.1. HTML

HTML, или HyperText Markup Language, е стандартен език за маркиране на уеб страници. Той се използва за създаване на структура и съдържание на уеб страници, като определя начина, по който информацията се представя в браузъра. Елементите в HTML се използват за да определят различни части от уеб страницата, като заглавия, параграфи, списъци, линкове, изображения и други[1].

HTML използва "тагове" или маркери, за да определи структурата на уеб страницата. Таговете са обикновени имена, обградени във въглови скоби (<>), които указват браузъра какво да прави със съдържанието между тях. Например, тагът за заглавие е <h1>, а тагът за параграф е <p>.

Всяка HTML страница започва със заглавен таг <html>, който определя началото на документа, и завършва със затварящ таг </html>, който означава края на документа. Между тези тагове се намира целият HTML код.

HTML не се грижи за външния вид на уеб страницата, той се фокусира върху структурата и семантиката. За стилизиране и оформление на уеб страници се използва CSS (Cascading Style Sheets), а за интерактивност - JavaScript.

[1] Google, 2024, html documentation, достъпена към 05.03.2024 от https://developer.mozilla.org/en-US/docs/Web/HTML

Фигура 1. Лого на HTML

1.1.2. CSS

CSS, или Cascading Style Sheets, е език за стилизиране на уеб страници. Той се използва за определяне на външния вид и оформлението на елементите на уеб страницата, като цветове, шрифтове, разположение, размери и други аспекти. CSS позволява разделянето на съдържанието на уеб страницата от нейния визуален вид, което прави кода по-лесен за поддръжка и управление. Той се състои от правила, които се прилагат към елементи на уеб страницата чрез селектори и декларации, които определят конкретните стилове. CSS е важна част от уеб дизайна и разработката на съвременни уеб приложения[2].

Фигура 2. Лого на CSS

1.1.3. JavaScript

JavaScript е високо ниво, интерпретируем, програмен език, който се използва главно за добавяне на интерактивност и динамично поведение към уеб

[2] Google,2024,What Css means?, достъпена към 01.03.2024 от https://www.atinternet.com/en/glossary/css/

страници. Той е основен елемент от модерното уеб разработване и се използва широко както от страна на клиента (в браузъра на потребителя), така и от страна на сървъра (с помощта на платформи като Node.js).

JavaScript позволява на разработчиците да манипулират HTML елементи, да реагират на събития като кликване или въвеждане на данни от потребителя, да извършват асинхронни заявки към сървъра и да създават динамично генерирани съдържания. Той също така поддържа обектно-ориентиран подход към програмирането, като предоставя вградени обекти и методи за манипулиране на данни и функционалности.

JavaScript е език с много разнообразни приложения, включително разработване на уеб приложения, игри, мобилни приложения и дори настолни приложения. Той е част от технологичния стек, който допълва HTML и CSS, за да създаде пълнофункционални и интерактивни уеб страници и приложения. Благодарение на постоянното развитие и подобрение на езика и браузърните двигатели, JavaScript остава един от най-важните елементи в сферата на уеб разработването[3].

Фигура 3. Лого на JavaScript

1.1.4 PHP

PHP е скриптов език за уеб програмиране, който се използва главно за създаване на динамични уеб страници и уеб приложения. Името PHP е акроним от

[3] Google, 2024, javascript documentation, достъпена към 03.03.2024 от https://devdocs.io/javascript/

"Hypertext Preprocessor". Той е много популярен и широко използван заради своята гъвкавост и силни възможности за обработка на данни на страната на сървъра[4].

PHP се изпълнява от сървъра, което означава, че кодът се изпълнява преди да се изпрати на потребителя. Това му позволява да генерира динамично съдържание в зависимост от различни условия и данни, получени от потребителя или от други източници.

Езикът предоставя голям брой вградени функции и библиотеки за обработка на формуляри, работа с бази данни, манипулиране на файлове, генериране на изображения и други. Също така, PHP е лесен за изучаване и използване, което го прави популярен сред уеб разработчиците на всички нива на умения.

PHP е често използван за създаване на системи за управление на съдържанието (CMS) като WordPress, Joomla и Drupal, както и за създаване на персонализирани уеб приложения и сървърни скриптове. Той е интегриран с почти всички основни бази данни, уеб сървъри и операционни системи, което го прави много удобен за разработване на различни видове уеб приложения.

Фигура 4. Лого на PHP

1.1.5. MySQL

MySQL е релационна база данни, която се използва широко за съхранение, управление и манипулиране на данни в уеб приложения и други софтуерни системи. Тя е една от най-популярните системи за управление на бази данни (СУБД) в света и е често използвана от уеб разработчиците поради своята

[4] Google,2024, PHP, достъпена към 17.03.2024 от https://www.php.net/docs.php

надежност, гъвкавост и лесна интеграция с различни програмни езици и платформи[5].

MySQL използва SQL (Structured Query Language) за манипулиране на данни. SQL е стандартен език за извличане, добавяне, промяна и изтриване на данни от релационни бази данни. С MySQL потребителите могат да създават, управляват и оптимизират бази данни с различни таблици, връзки и индекси.

MySQL е отворен код и е налице безплатна версия, наречена MySQL Community Edition, както и платени версии с допълнителни функционалности и поддръжка. Тя е изключително мащабируема и може да бъде използвана както за малки уеб сайтове, така и за големи уеб приложения с милиони записи в базата данни.

Поради своите качества, MySQL е предпочитан избор за разработчиците, които разработват уеб базирани приложения и имат нужда от мощна и надеждна система за управление на данни.

Фигура 5. Лого на My SQL

1.1.6. AJAX

AJAX, или Asynchronous JavaScript and XML, е технология за разработка на уеб приложения, която позволява на страниците да обменят данни със сървъра и да обновяват части от страницата без да се налага презареждане на цялата страница. Това се постига чрез изпращане на заявки към сървъра и получаване на отговорите от него във формат JSON, XML, HTML или друг. AJAX използва

[5] Google,2024, MySQL documentation, достъпена към 18.03.2024 от https://cloud.google.com/sql/docs/mysql

JavaScript за асинхронно изпълнение на заявките, като това позволява на потребителите да взаимодействат с уеб приложението без прекъсване на потока на работа[6].

Основните принципи на AJAX включват използването на обекти XMLHttpRequest за изпращане на заявки към сървъра и обработката на отговорите от него. След получаване на отговора, JavaScript може да обнови само част от страницата, без да се налага презареждане на цялата страница. Това позволява на уеб приложенията да се чувстват по-отзивчиви и да предоставят по-бързо и по-приятно потребителско изживяване.

Примери за използване на AJAX включват динамичното зареждане на съдържание, автоматичното допълване на формуляри, валидация на данни на страната на клиента и много други. AJAX е важен елемент в съвременното уеб разработване и е особено полезен за създаването на интерактивни и динамични уеб приложения.

Фигура 6. Лого на AJAX

1.1.7. PHP MyAdmin

PHP MyAdmin е безплатен уеб базиран административен интерфейс за управление на MySQL бази данни. Той предоставя графичен потребителски интерфейс (GUI), който позволява на администраторите и разработчиците да

[6] Google,2020, AJAX документация, достъпена към 16.03.2024 от https://doc.nette.org/bg/application/ajax

управляват бази данни и техните обекти (таблиците, записите, индексите и др.) безпроблемно и лесно през уеб браузър[7].

PHP MyAdmin предоставя широк набор от функции, включително създаване и изтриване на бази данни и таблици, добавяне, редактиране и изтриване на данни, изпълняване на SQL заявки, управление на потребители и техните привилегии, добавяне и извличане на данни, генериране на резервни копия и много други.

С помощта на PHP MyAdmin потребителите могат да използват MySQL базите данни без необходимостта да се занимават със сложни команди в командния ред. Той е особено полезен за начинаещи потребители, които искат да започнат да използват MySQL бази данни, както и за опитни администратори, които търсят бърз и лесен начин за управление на базите данни.

PHP MyAdmin е широко използван инструмент и е наличен за почти всяка уеб хостинг платформа. Той е с отворен код и постоянно се развива и подобрява от голямо общество от разработчици.

Фигура 7. Лого на PHP MyAdmin

1.1.8. XAMPP

XAMPP е популярен и безплатен софтуерен пакет, който предоставя среда за разработка на уеб приложения, базирани на PHP, Apache и MySQL. Името XAMPP идва от първите букви на основните компоненти: X - кръстосани (Cross-platform), A - Apache HTTP Server, M - MySQL, P - PHP и P - Perl[8].

[7] Google,2024, PHP myAdmin , достъпена към 10.03.2024 от https://www.phpmyadmin.net/

[8] Google,2023, XAMP, достъпена към 16.03.2024 от https://bg.wikipedia.org/wiki/XAMPP

Основната цел на XAMPP е да предостави интегрирана среда, която позволява на разработчиците да създават и тестват уеб приложения локално на техните компютри, преди да ги публикуват на живи уеб сървъри. Това е особено полезно за начинаещи програмисти и уеб дизайнери, които искат да се запознаят с уеб разработката и да тестват своите проекти без необходимостта да се свързват с реален уеб сървър.

XAMPP включва следните компоненти:

- Apache HTTP Server: Уеб сървърът, който обслужва уеб страници и приложения, написани на PHP, Perl и други езици за уеб програмиране.
- MySQL: Релационна база данни, която се използва за съхранение и управление на данни за уеб приложенията.
- PHP: Скриптов език за програмиране, който се използва за генериране на динамично съдържание на уеб страниците.
- Perl: Един от най-популярните скриптови езици за програмиране, използван за автоматизиране на различни задачи в уеб разработката.

С XAMPP потребителите могат бързо и лесно да инсталират цялата необходима инфраструктура за уеб разработка на своите компютри, като получават готова за употреба платформа за създаване на уеб приложения.

Фигура 8. Лого на XAMPP

1.2. Сравнителен анализ с други технологии и езици

Проучването на различни технологии е необходимо, за да се определи най-ефективната комбинация от езици и разработени инструменти, която би предложила сигурност, удобство и възможност за лесна поддръжка.

1.2.1. Предимства на използваните технологии и езици

HTML (HyperText Markup Language):

- Лесен за изучаване и разбиране.
- Стандартен език за маркиране на уеб страници.
- Позволява структуриране на съдържанието на уеб страници.
- Съвместим с различни браузъри и устройства.

CSS (Cascading Style Sheets):

- Позволява стилизиране и оформление на уеб страници.
- Разделя отделно съдържанието от външния вид на уеб страниците.
- Гъвкав и мощен инструмент за дизайнерите.
- Позволява създаването на атрактивни и професионални уеб дизайни.

JavaScript:

- Добавя интерактивност и динамичност към уеб страници.
- Мощен език за програмиране, който се изпълнява от страната на клиента.
- Позволява манипулиране на HTML елементи, обработка на събития и изпращане на заявки към сървъра без презареждане на страницата.
- Използва се широко за създаване на уеб приложения и игри.

PHP (Hypertext Preprocessor):

- Мощен и гъвкав език за уеб програмиране.
- Лесно интегрируем с MySQL бази данни и други технологии.
- Използва се за създаване на динамични уеб страници и уеб приложения.
- Предоставя голям набор от вградени функции и библиотеки за работа със сървърните данни и файлове.

MySQL:

- Мощна релационна база данни.
- Лесна за управление и мащабируема.
- Поддържа стандартни SQL заявки и операции.
- Бърза и надеждна, подходяща за различни видове уеб приложения и услуги.

PHP MyAdmin:

- Удобен и лесен за използване административен интерфейс за управление на MySQL бази данни.
- Предоставя графичен потребителски интерфейс (GUI) за управление на бази данни и техните обекти.
- Интегрира се лесно с почти всички хостинг платформи и предлага множество функции за управление на данните.

AJAX (Asynchronous JavaScript and XML):

- Позволява асинхронен обмен на данни между браузъра и сървъра, без да се налага презареждане на цялата уеб страница.
- Увеличава отзивчивостта на уеб приложенията, като позволява на потребителите да извършват действия без прекъсване на работата си.
- Намалява натовареността на сървъра и ускорява зареждането на уеб страници, като се изпращат само необходимите данни.
- Позволява динамично обновяване на части от уеб страницата, което подобрява потребителското изживяване.
- Използва се широко във всякакви уеб приложения, където е необходимо динамично зареждане на съдържание или взаимодействие с потребителя.

XAMPP

- Лесна инсталация и конфигурация: XAMPP е лесен за инсталиране и конфигуриране, което го прави идеален за начинаещи. С една инсталация можеш да получиш цялата инфраструктура, необходима за уеб разработка - уеб сървър, база данни и програмиращи езици.
- Портативност: XAMPP е портативен и може да бъде инсталиран на USB устройство или преносим твърд диск. Това позволява на потребителите да носят своята разработъчна среда навсякъде с тях и да работят на различни компютри без да се налага да инсталират отново софтуера.
- Интегрирана среда за разработка: XAMPP включва всичко необходимо за уеб разработка - уеб сървър (Apache), база данни (MySQL), програмиращи езици (PHP, Perl) и други инструменти. Това позволява на разработчиците да се концентрират върху създаването на приложения, без да се налага да търсят и инсталират различни компоненти отделно.

- Безплатен и отворен код: XAMPP е безплатен за изтегляне и ползване, като също така е разпространяван под лицензa GNU General Public License (GPL). Това означава, че всеки може да го използва, да го модифицира и да го разпространява свободно.
- Поддръжка на различни операционни системи: XAMPP е съвместим с различни операционни системи като Windows, Linux и macOS, което го прави подходящ за широк кръг от потребители.

1.2.2. Недостатъци на използваните технологии и езици

HTML (HyperText Markup Language):

- Не предоставя динамичност или интерактивност без използването на други технологии като CSS и JavaScript.
- Понякога може да бъде предизвикателство да се постигне желаният дизайн или оформление.

CSS (Cascading Style Sheets):

- Може да се окаже сложно да се поддържат по-големи проекти, особено когато стиловете стават сложни и дълги.
- При използването на стили за крос-браузърна съвместимост може да възникнат разлики в рендерирането на уеб страници.

JavaScript:

- Възможни са проблеми със съвместимостта с различни браузъри
- Злоупотребата с JavaScript може да доведе до забавяне на зареждането на уеб страници и увеличение на използваната от тях памет.

PHP (Hypertext Preprocessor):

- Известен софтуер, прибегващ към по-модерни алтернативи като Python или Node.js, е малко използван, поради факта, че съществуват по-съвременни опции.
- Възможно е да се сблъскате с проблеми със сигурността, ако не се прилагат правилни практики за програмиране.

MySQL:

- Може да възникнат проблеми с мащабируемостта при работа с големи обеми данни.
- Имплементирането на сложни структури от данни и заявки може да изисква по-високо ниво на опит от разработчика.

PHP MyAdmin:

- Възможни са проблеми със сигурността, особено ако не се актуализира редовно до последната версия.
- Може да се окаже неудобен за използване при работа с големи бази данни поради ограничения в производителността.

AJAX (Asynchronous JavaScript and XML):

- Приложенията, които използват AJAX, може да станат по-трудни за поддръжка и разбиране поради асинхронното им поведение и увеличава сложността на поддръжката.
- Може да се сблъска с проблеми със сигурността, а неправилната употреба може да доведе до злоупотреби и натоварване на сървъра.
- Разработването и отстраняването на грешки в AJAX приложения може да изисква повече усилия в сравнение с традиционните уеб приложения.
- Не всички браузъри поддържат AJAX или изискват специална конфигурация, което може да доведе до проблеми със съвместимостта.
- Неправилното използване на AJAX може да доведе до намалена достъпност и SEO на уеб сайта, тъй като поисковите роботи не винаги могат да проучат динамичното съдържание, заредено през AJAX.

XAMPP

- Сигурност: Понякога конфигурацията по подразбиране на XAMPP може да бъде настроена с недостатъчна сигурност, което може да представлява риск за разработваните приложения, особено ако са свързани с интернет. Разработчиците трябва да са внимателни и да внимават да настроят правилно сигурността на своите проекти.
- Обновления и поддръжка: XAMPP не предоставя автоматични механизми за обновяване на компонентите му, което може да доведе до забавяне в

интегрирането на нови версии на софтуера и поправки на сигурността. Потребителите трябва редовно да следят за обновления и да ги инсталират ръчно.

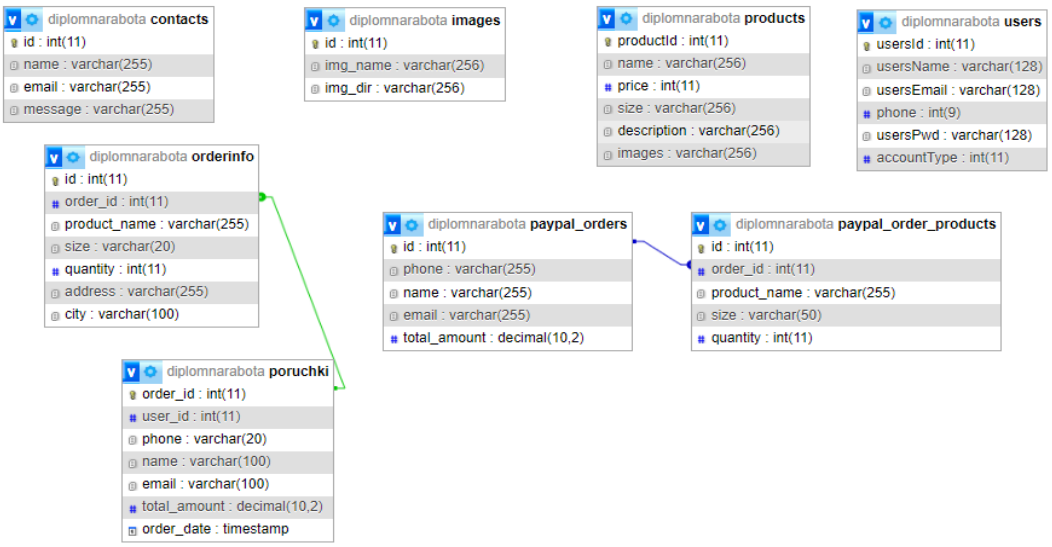
- Производителност и мащабируемост: Въпреки че XAMPP е подходящ за разработка и тестване на малки и средни уеб приложения, той може да се окаже недостатъчен за по-големи и мащабни проекти. Използването на XAMPP за големи уеб приложения може да доведе до намаляване на производителността и затруднения в управлението на системата.
- Липса на някои технологии и инструменти: Въпреки че XAMPP включва основните компоненти за уеб разработка, той може да липсва някои от по-специализираните технологии и инструменти, които се използват в съвременната уеб разработка. Разработчиците може да се наложи да инсталират и конфигурират допълнителни софтуерни пакети за тяхните нужди.
- Излишен софтуерен набор: За някои проекти XAMPP може да представлява излишен софтуерен набор, като включването на ненужни компоненти може да доведе до повишена сложност и конфигурационни проблеми.
- Заетост на ресурси: Понякога XAMPP може да изисква значителни ресурси от системата, което може да забави работата на компютъра, особено на по-стари или по-слаби машини.

Избраните технологии и езици HTML, CSS, JavaScript, PHP, MySQL и Composer, са използвани за изработката на уеб сайта, защото те предоставят необходимите инструменти и технологии за разработка на цялостно функциониращо уеб приложение. HTML, CSS и JavaScript са основните елементи за създаване на интерактивен и атрактивен потребителски интерфейс, докато PHP и MySQL са подходящи за изграждане на сървърната страна на уеб сайта и управление на база данни. Composer се използва за управление на зависимостите и пакетите, които са необходими за разработката и поддръжката на проекта. Използването на тези технологии дава възможност за създаване на функционално и ефективно уеб приложение, което може да отговори на нуждите на потребителите и да бъде лесно поддръжаемо в бъдеще.

ГЛАВА 2 ПРОЕКТИРАНЕ И СТРУКТУРА – МОДУЛИ НА ПРИЛОЖЕНИЕТО

2.1. Проектиране на базата от данни

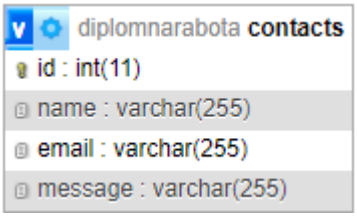
Фигура 9 представя структурната схема на базата от данни на приложението, която включва осем таблици с информация за потребителите и продуктите с техните описания, поръчки и изображения за продуктите . Някои таблици са свързани външен ключ и е показано кое поле от таблицата играе ролята на foreign key.



Фигура 9. Структура на таблиците в базата данни

2.2. Таблици в базата от данни

Таблицата contacts е създадена за съхранение на информацията за отзивите, подадени от потребителите.

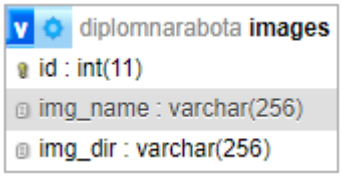


Фигура 10. Структура на таблицата contacts в базата данни

- **id** - колоната id е първичен ключ за релация за contacts, той е уникален идентификатор за записите в таблицата.
- **name** - колоната name съхранява информация за името на потребителя под формата на низ от символи.

- **email** – колоната съхранява информация за електронната поща на потребителят под формата на низ от символи.
- **message** - поле съдържащо информация относно съобщението, подадено от потребителя

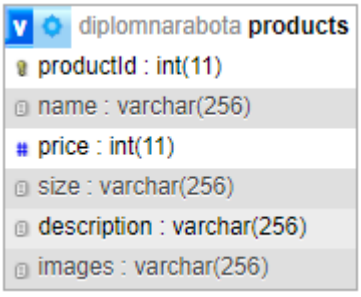
Таблица images е създадена за съхранение на изображенията на продуктите.



Фигура 11.Структура на таблицата images в базата данни

- **id** - колоната id е първичен ключ за релация за images, той е уникален идентификатор за записите в таблицата.
- **Img_name** - колоната img_name съхранява информация за името на изображението на продуктите и е от тип низ от символи.
- **Img_dir–** колоната съхранява информация за директорията на изображението.

Таблица Products е създадена за съхранение на характеристиките на продуктите.

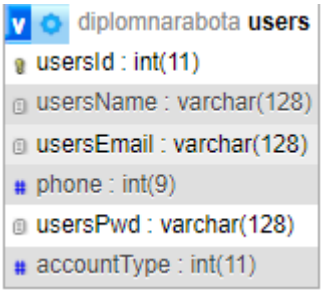


Фигура 12.Структура на таблицата products в базата данни

- **productid** - колоната productid е първичен ключ за таблица images, той е уникален идентификатор за записите в таблицата.
- **name** - колоната name съхранява информация за името на продуктите.
- **price** – колоната съхранява информация за цената на продукта.

- **size** – колоната съхранява информация за размера на съответния продукт.
- **description** - колоната съхранява информацияза описанието на съответния продукт.
- **images** - колоната съхранява информация за изображенията на съответния продукт

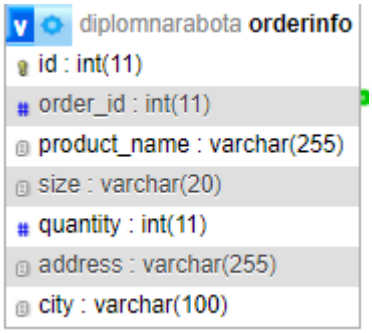
Таблицата users е създадена за съхранение на информацията за всеки един потребител.



Фигура 13. Структура на таблица users s в базата данни

- **usertid** - колоната userid е първичен ключ за таблица за images, той е уникален идентификатор за записите в таблицата.
- **username** - колоната username съхранява информация за потребителското име .
- **userEmail** – колоната съхранява информация за електронната поща на потребителя.
- **phone** – колоната съхранява информация за телефонния номер на потребителя.
- **usersPwd** - колоната съхранява информация за паролата на потребителя.
- **accountType** - колоната съхранява информация за вида на потребителя (администратор или потребител) като той се определя от администратора. Ако е 1 е администратор, а ако е 0 е обикновен потребител.

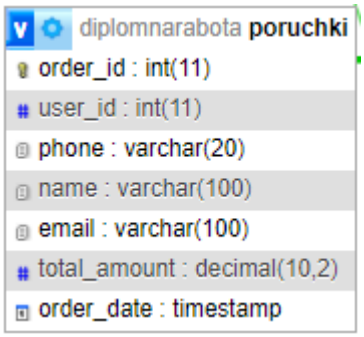
Таблицата orderinfo е създадена за съхранение на информацията за всяка една поръчка от потребителите.



Фигура 14.Структура на таблицата orderinfo в базата данни

- **id** - колоната userid е първичен ключ за таблица orderinfo, той е уникален идентификатор за записите в таблицата.
- **order_id** - колоната order_id съхранява информация за id-то на поръчката . Тази колона има роля на външен ключ с таблица porychki.
- **product_name** – колоната съхранява информация за името на поръчания продукт.
- **size**– колоната съхранява информация за размера на поръчания продукт.
- **quantity**- колоната съхранява количеството на поръчания вече продукт от потребителя.
- **adress** - колоната съхранява информация адреса на потребителя, до който трябва да бъде доставена поръчката.
- **city** - колоната съхранява информация града на потребителя, до който трябва да бъде доставена поръчката.

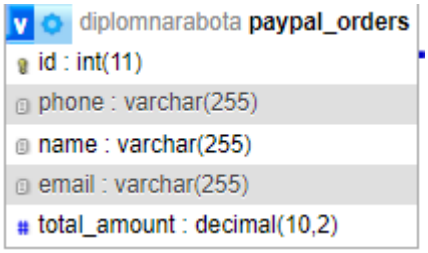
Таблицата poruchki е създадена за съхранение на информацията за всяка една поръчка от потребителите като в нея, обаче са въведени данни на потребителя ,който я е поръчал. Тази таблица записва данните, когато вече е направена поръчката.



Фигура 15.Структура на таблицата poruchki в базата данни

- **order_id** - колоната orderid е първичен ключ за таблица poruchki, той е уникален идентификатор за записите в таблицата. Тази колона има роля на външен ключ с таблица orderinfo.
- **user_id** - колоната order_id съхранява информация за id-то на потребителя. Тази колона има роля на външен ключ с таблица porychki.
- **phone–** колоната съхранява информация за телефонния номер на потребителя.
- **name–** колоната съхранява информация за името на потребителя.
- **email**- колоната съхранява електронната поща на потребителя.
- **total_amount**- колоната съхранява информация цялата сума на поръчката с включено ДДС.
- **order_date**- колоната съхранява информация за датата на поръчката. Тази информация не се попълва от потребителя.

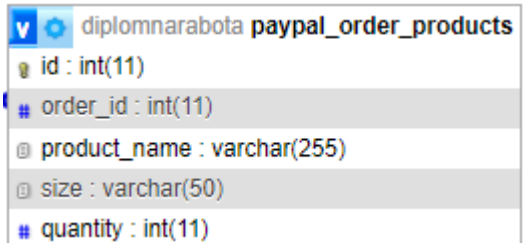
Таблицата paypal_orders е създадена за съхранение на информацията за всяка една поръчка от потребителите при плащане със съответния метод на плащане. Като в нея, обаче са въведени данни на потребителя, когато е направена вече поръчката и се преминава към плащането.



Фигура 16.Структура на таблицата paypal_orders в базата данни

- **id** - колоната id е първичен ключ за таблица paypal_orders, той е уникален идентификатор за записите в таблицата. Тази колона има роля на външен ключ с таблица paypal_order_products.
- **phone**- колоната order_id съхранява информация за телефонния номер на потребителя
- **name–** колоната съхранява информация за името на потребителя.
- **email**- колоната съхранява електронната поща на потребителя.
- **total_amount**- колоната съхранява информация цялата сума на поръчката с включено ДДС.

Таблицата paypal_order_products е създадена за съхранение на информацията за продуктите при направено вече плащане.

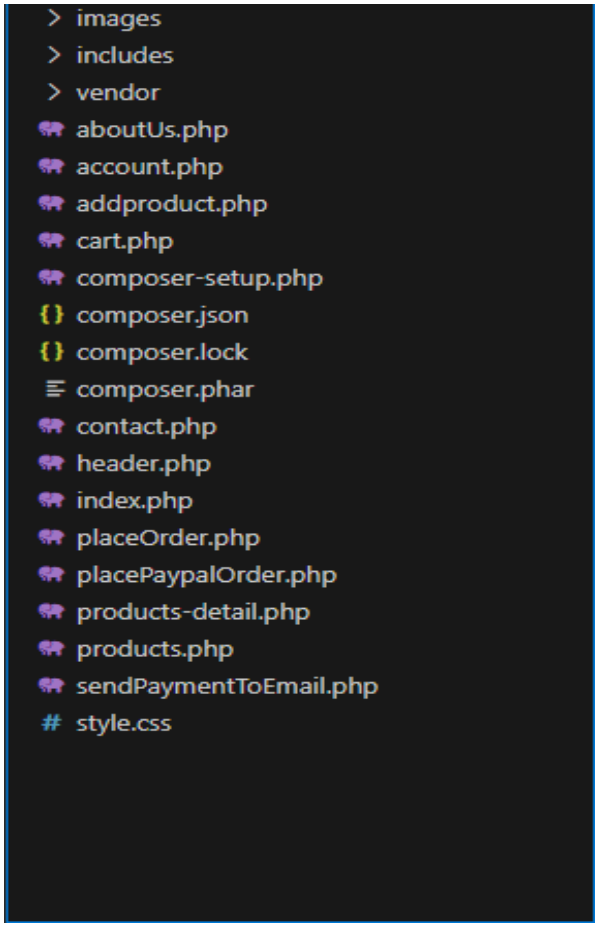


Фигура 17. Структура на таблица paypal_orders_products в базата данни

- **id** - колоната id е първичен ключ за таблица paypal_order_products, той е уникален идентификатор за записите в таблицата. Тази колона има роля на външен ключ с таблица orderinfo.
- **order_id** - колоната order_id съхранява информация за id-то на поръчката. Тази колона има роля на външен ключ с таблица paypal_orders.
- **Product_name**– колоната съхранява информация за името на поръчания продукт.
- **size**– колоната съхранява информация за размера на продукта.
- **quantity** – колоната съхранява информация за количеството на всеки един продукт..
- **order_date**- колоната съхранява информация за датата на поръчката. Тази информация не се попълва от потребителя.

2.3. Файлова структура на приложението

На фигура 15 са представени всички файлове използвани при разработката на уеб базираното приложение. Някои файлове са подредени в папки, а страниците, както и някои от функционалностите са като отделни документи.



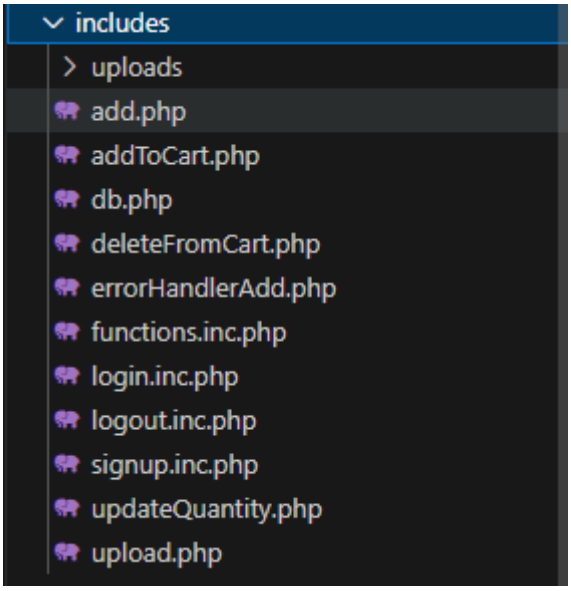
Фигура 18. Цялостна структура на приложението

Папка images съдържа всички изображенията в приложението като лого, снимки на продуктите, изображения за работния персонал и за количката.

Папка includes се съдържат файловете за почти всички функционалности в приложението.

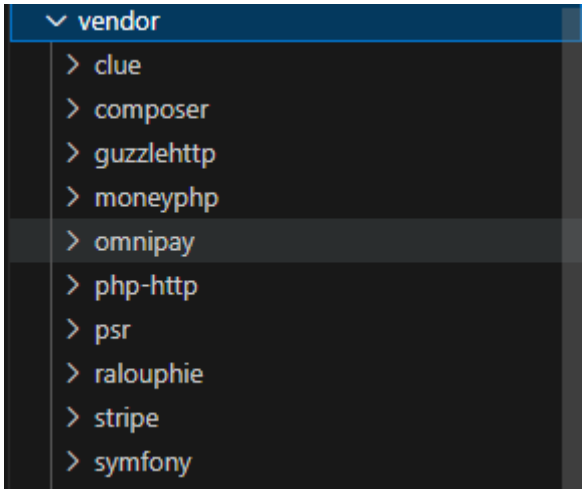
- add.php – тази функционалност добавя и записва продуктите в базата данни.
- addToCart.php – този файл добавя продуктите във виртуалната количка
- db.php – тук е свързана базата данни с localhost
- deleteFromCart – изтрива продуктите от количката
- errorHandlerAdd – показва и проверява за грешки при добавянето на продукт от страна на администратора.
- function.inc.php – чрез този файл са създадени функционалностите на полетата, които може да има един потребител.. Валидацията, добавена в този файл, проверява дали всички полета са правилни и показва, ако има грешки.

- login.inc.php – чрез този файл влизаш в приложението
- logout.inc.php – чрез този файл излизаш от приложението
- signup.inc.php - чрез този файл си създаваш профил
- updateQuantity.php – този файл обновява количеството на продукта в количката
- upload.php – в този файл се съдържа валидация при добавянето на продукта и грешките, които излизат при неверни данни.



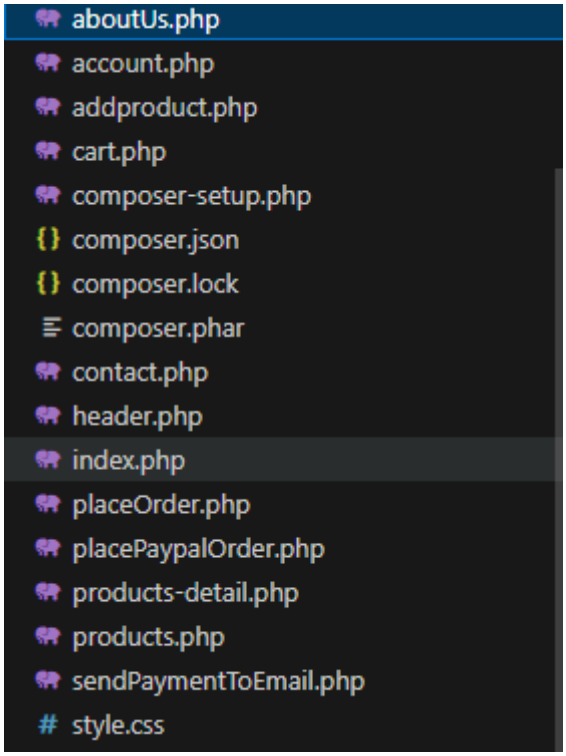
Фигура 19. Структура на папка uploads

Папка vendor съдържа всички файлове при избрания метод на плащане.



Фигура 20. Структура на папка vendor

В останалите файлове се съдържа frontend- а на страниците и някои други функционалности.



Фигура 21. Структура на основните страници

- aboutUs.php – страница, в която е написана кратка обобщаваща информация за нашия екип.
- account.php – страница, в която се регистрираш и влизаш в приложението.
- addproduct.php – страница за добавяне на продукти, от страна на администратора.
- cart.php – страницата за виртуалната количката
- composer-setup.php - Този файл е използван за инсталиране на Composer, който е инструмент за управление на зависимости в PHP проекти. Обикновено той се използва за създаване на файл composer.phar, който е изпълнимият файл на Composer.
- composer.json - В composer-setup.php се включва PHP код, който сваля composer.phar от интернет и го инсталира в текущата директория.
- composer.lock – Файлът съдържа информация за конкретните версии на зависимостите, използвани в проекта. Това включва и версии на зависимостите на зависимостите.
- composer.phar – Файлът composer.phar е изпълнимият файл на Composer, който се използва за управление на зависимостите на PHP проекта.

- contact.php – страница за Контакти
- header.php – в него се съдържа логото и навигационния бар.
- index.php – заглавната страница на приложението.
- placeOrder – файл, в който се вмъкват данните за продуктите в таблицата "orderinfo"
- placePaypalOrder.php –файл, в която потребителят въвежда информация за мястото, до където трябва да бъде доставена поръчката.
- product-detail.php – страница, в която се показват характеристиките на продуктите.
- products.php – страницата, която визуализира всички продукти.
- sendPaymentToEmail.php – изпраща информация за направена поръчка, като ги праща на моята електронна поща
- style.css – съдържа всички стилове на всеки един елемент от Html документа приложението.

ГЛАВА 3 ПРОГРАМНА РЕАЛИЗАЦИЯ

3.1. Програмна реализация за управление на потребителски профили

```
<form id="LoginForm"  action="includes/login.inc.php"
method="POST">

      <input type="text" name="username"
placeholder="Потребителско име:">

      <input type="password" name="password"
placeholder="Парола:">

      <button class="btn" type="submit" name="submit"
value="submit">Влез</button>

      <a href="">Забравена парола ?</a>

</form>
```

Представената HTML конструкция е форма, предназначена за вход на потребители в уеб приложение.

<form> елементът определя началото на формата.

id="LoginForm" задава уникален идентификатор на формата, който може да бъде използван за целите на стилизиране или в JavaScript скриптове.

action="includes/login.inc.php" указва пътя към PHP файл, който ще обработва данните от формата след натискане на бутона за изпращане.

method="POST" определя HTTP метода, използван за изпращане на данните към сървъра.

<input> елементът представлява текстово поле за въвеждане на потребителско име.

type="text" определя типа на полето като текстово.

name="username" задава името на полето, което ще бъде използвано за идентифициране на данните в PHP скрипта.

placeholder="Потребителско име:" предоставя подсказка (placeholder) в полето за въвеждане на данни.

<input> елементът представлява поле за въвеждане на парола.

type="password" указва, че паролата ще бъде скрита при въвеждане.

name="password" определя името на полето, което ще бъде използвано за паролата в PHP скрипта.

placeholder="Парола:" предоставя подсказка (placeholder) в полето за въвеждане на парола.

<button> елементът представлява бутон за изпращане на формата.

class="btn" определя класа на бутона, който може да бъде използван за стилизиране.

type="submit" указва, че бутона ще бъде използван за изпращане на формата.

name="submit" задава името на бутона, което ще бъде използвано за идентифициране на действието в PHP скрипта.

value="submit" задава стойността на бутона, която ще бъде изпратена към сървъра при изпращане на формата.

<a> елементът създава хипервръзка към друга страница или ресурс.

href="" определя адреса, към който връзката води. В този случай, атрибутът е празен, което означава, че връзката няма да води към никъде. Това често се използва за връзки, които ще бъдат обработени чрез JavaScript.

```
<form id="RegForm" action="includes/signup.inc.php"
method="POST">

      <input type="text" name="username"
placeholder="Потребителско име:">
```

```
        <input type="email" name="email"
placeholder="Имейл:">

      <input type="phonenumber" name="phone"
placeholder="Телефонен номер:">

      <input type="password" name="password"
placeholder="Парола:">

      <input type="password" name="confirmPassword"
placeholder="Потвърди парола:">

      <button class="btn" type="submit" name="submit"
value="submit">Регистрирай се</button>

</form>
```

Представеният код е за регистрационната форма на потребителите в уеб приложението.

id="RegForm" задава уникален идентификатор на формата, който може да бъде използван за стилизиране или в JavaScript скриптове.

action="includes/signup.inc.php" указва пътя към PHP файл, който ще обработва данните от формата след натискане на бутона за изпращане.

method="POST" определя HTTP метода, използван за изпращане на данните към сървъра.

<input> елементът представлява текстово поле за въвеждане на потребителско име.

type="text" определя типа на полето като текстово.

name="username" задава името на полето, което ще бъде използвано за идентифициране на данните в PHP скрипта.

placeholder="Потребителско име:" предоставя подсказка (placeholder) в полето за въвеждане на данни.

<input> елементът представлява поле за въвеждане на имейл адрес.

type="email" указва, че полето ще приема валиден имейл адрес.

name="email" задава името на полето, което ще бъде използвано за идентифициране на данните в PHP скрипта.

placeholder="Имейл:" предоставя подсказка (placeholder) в полето за въвеждане на имейл адрес.

<input> елементът представлява поле за въвеждане на телефонен номер.

<type="phonenumber" указва, че полето ще приема телефонен номер.

name="phone" задава името на полето, което ще бъде използвано за идентифициране на данните в PHP скрипта.

placeholder="Телефонен номер:" предоставя подсказка (placeholder) в полето за въвеждане на телефонен номер.

<input> елементът представлява поле за въвеждане на парола.

type="password" указва, че паролата ще бъде скрита при въвеждане.

name="password" задава името на полето, което ще бъде използвано за идентифициране на паролата в PHP скрипта.

placeholder="Парола:" предоставя подсказка (placeholder) в полето за въвеждане на парола.

<input> елементът представлява поле за въвеждане на потвърждение за паролата.

type="password" указва, че паролата ще бъде скрита при въвеждане.

name="confirmPassword" задава името на полето, което ще бъде използвано за идентифициране на потвърждението за паролата в PHP скрипта.

placeholder="Потвърди парола:" предоставя подсказка (placeholder) в полето за въвеждане на потвърждение за паролата.

<button> елементът представлява бутон за изпращане на формата.

class="btn" определя класа на бутона, който може да бъде използван за стилизиране.

type="submit" указва, че бутона ще бъде използван за изпращане на формата.

name="submit" задава името на бутона, което ще бъде използвано за идентифициране на действието в PHP скрипта.

value="submit" задава стойността на бутона, която ще бъде изпратена към сървъра при изпращане на формата.

Пълното съдържание на кода може да се види в **Приложение 1**.

3.2. Програмна реализация за добавянето на продукт от администраторски профил

```
<div class="uploadImg rounded lead text-center mt-3
center">
  <div class="title lead">Качи файл</div>
  <div class="dropzone mx-auto my-5">
  <div class="content p-5">
  <img src="images/upload.svg" class="upload" id="img">
 <span class="filename"></span>
  <input type="file" name="image[]" class="input" multiple>
 </div>
 </div>
</div>
```

Кодът представлява елемент за качване на файлове с помощта на dropzone (зона за пускане) от администраторския профил.

: Това е основният контейнер за елемента за качване на файлове. Включва различни класове за стилизация, като rounded за закръглени ръбове и text-center за центриран текст.

Качи файл : Заглавието на елемента, което се показва над зоната за качване на файлове. Класът lead се използва за увеличаване на размера на шрифта.

: Контейнерът за зоната за пускане (dropzone), където потребителят може да пусне файлове. mx-auto и my-5 са класове за центриране на dropzone и добавяне на вътрешни отстъпи.

: Контейнерът за съдържанието на dropzone, който включва изображението за качване на файл (upload.svg), както и полето за въвеждане на файл (<input type="file" name="image[]" class="input" multiple>). Класът p-5 се използва за добавяне на външни отстъпи.

<img src="images/upload.svg" class="upload" id="img">: Изображението за качване на файл, което се показва във вътрешността на dropzone. Класът upload се използва за стилизация и JavaScript манипулации.

: Празен спан, който може да се използва за показване на името на качения файл.

<input type="file" name="image[]" class="input" multiple>: Полето за въвеждане на файл, което позволява на потребителя да избере един или повече файлове за качване. Атрибутът multiple позволява избор на множество файлове.

Пълното съдържание на кода може да се види в **Приложение 2**.

3.3. Програмна реализация за осъществяването на поръчки и плащания

```
<div class="total-price">

   <table>

    <tr>

    <td>Междинна сума:</td>

    <td id="intermediateTotal"><?php echo
isset($intermediateTotal) ?
number_format($intermediateTotal, 2) : '0.00'; ?> лв.</td>

    </tr>

    <tr>

    <td>ДДС:</td>
```

```
    <td id="vat"><?php echo isset($intermediateTotal) ?
number_format($intermediateTotal * 0.2, 2) : '0.00'; ?>
лв.</td>

    </tr>

     <tr>

     <td>Обща сума:</td>

     <td id="total"><?php echo isset($intermediateTotal) ?
number_format(($intermediateTotal + $intermediateTotal *
0.2), 2) : '0.00'; ?> лв.</td>

      </tr>

      <tr>

      <td colspan="2">

      <button onclick="showPaymentForm()">Плащане</button>

      </td>

      </tr>

     </table>

    </div>
```

Този код представлява част от уеб страницата, която показва общата сума на поръчката и предлага бутон за преминаване към процеса на плащане.

: Това е контейнерът, който съдържа таблицата с общата сума на поръчката и бутона за плащане.

<table>: Таблицата, в която са подредени различните редове с информация за цената.

<tr>: Ред в таблицата.

<td>Междинна сума:</td>: Първата клетка от реда, която показва заглавие за междинната сума.

<td id="intermediateTotal"><?php echo isset($intermediateTotal) ? number_format($intermediateTotal, 2) : '0.00'; ?> лв.</td>: Втората клетка от реда, която показва междинната сума. Ако променливата $intermediateTotal е дефинирана, тя се изписва с точност до два десетични знака и със символа за валута "лв.". Ако променливата не е дефинирана, се изписва стойността "0.00".

Аналогично се постъпва и за редовете с ДДС и общата сума.

<button onclick="showPaymentForm()">Плащане</button>: Бутонът, който, при кликване на него, стартира функцията showPaymentForm(), която показва формата за плащане.

Този код се използва за показване на ценовата информация на поръчката на потребителя. Той включва три реда, които показват междинната сума, ДДС и общата сума на поръчката. Под тези редове има бутон "Плащане", който потребителят може да кликне, за да премине към следващия етап от процеса на плащане.

```
<div id="paymentForm" style="display: none;">
  <h2>Информация за плащане</h2>
   <form id="paymentFormSubmit" 
onsubmit="submitPaymentForm(event)">
    <label for="phone">Телефонен номер:</label><br>
    <input type="text" id="phone" name="phone" 
required><br>
     <label for="name">Име:</label><br>
     <input type="text" id="name" name="name" required><br>
      <label for="email">Имейл:</label><br>
      <input type="email" id="email" name="email" 
required><br>
      <label for="address">Адрес:</label><br>
```

```
      <input type="text" id="address" name="address"
required><br>

      <label for="city">Град:</label><br>

      <input type="text" id="city" name="city"
required><br><br>

      <input type="submit" value="Потвърди"
id="submitPaymentButton">

<div id="paypal-button-container"></div>

      </form>

  </div>
```

Този код представлява форма за въвеждане на информация за плащане, която първоначално е скрита от потребителя. Когато потребителят е готов да плати, формата може да бъде показана.

: Това е контейнерът за цялата форма за плащане. Стилът display: none; го прави невидим, докато не бъде активиран.

<h2>Информация за плащане</h2>: Заглавието на формата, което пояснява предназначението й.

<form id="paymentFormSubmit" onsubmit="submitPaymentForm(event)">: Това е самата форма за въвеждане на информация. Когато потребителят я подаде, тя изпълнява функцията submitPaymentForm(event).

<label> елементите се използват за означаване на полетата за въвеждане на информация.

<input> елементите представляват текстови полета, където потребителят може да въведе своя телефонен номер, име, имейл, адрес и град. Полетата са от тип text или email, като някои от тях са задължителни за попълване (атрибутът required).

<input type="submit" value="Потвърди" id="submitPaymentButton">: Това е бутонът за потвърждение на информацията. При натискане, формата се подава за обработка.

: Това е контейнерът за PayPal бутона, който се използва за плащане чрез PayPal. Той се визуализира във формата, ако се използва този метод на плащане.

Този код представлява форма за въвеждане на информация за плащане. Когато потребителят попълни формата и я подаде, се изпълнява JavaScript функцията submitPaymentForm(event).

JavaScript функцията има за цел да обработи информацията от формата и да я изпрати за обработка на сървъра. Тази функция използва AJAX заявка, за да прехвърли данните към определен скрипт на сървъра, който се грижи за обработката на плащането.

Когато потребителят натисне бутона "Потвърди" във формата, тя се предава на тази JavaScript функция, която взема данните от текстовите полета във формата (телефонен номер, име, имейл, адрес, град) и ги подготвя за изпращане към сървъра. В случай че се използва плащане чрез PayPal, тя също така може да включва инициализирането на PayPal бутона, който се визуализира в контейнера за PayPal бутона (paypal-button-container).

След като се изпрати заявката към сървъра и се обработи успешно, потребителят може да бъде пренасочен към друга страница, където да получи потвърждение за успешно извършеното плащане.

Пълното съдържание на кода може да се види в **Приложение 3**.

3.4. Хостинг

Хостването на уебсайт е от съществено значение поради редица причини:

- Позволява на потребителите да видят уебсайта в интернет.
- Служи като визитка за бизнеса или организацията онлайн.
- Помага за достигането до потенциални клиенти.
- Предоставя платформа за взаимодействие с клиентите и общността.
- Позволява на онлайн бизнесите да предлагат продукти и услуги.
- Осигурява съхранение на информацията и защита на данните на потребителите.

Infinityfree.app предоставя безплатен хостинг

Ако някой няма акаунт в InfinityFree, посещава уебсайта им и създава нов акаунт. За целите на регистрацията е необходим валиден имейл адрес.

След като влиза в акаунта си в InfinityFree, потребителят добавя домейн, който иска да хостне. DNS записите на домейна се променят, за да сочат към DNS сървърите на InfinityFree.

Използва се FTP (File Transfer Protocol) клиент, за да се качат файловете на уебсайта на сървъра на InfinityFree. При влизане в акаунта, се намират подробни инструкции за FTP достъп и как да се качат файловете.

Ако се използва база данни за уеб приложения (например MySQL), InfinityFree предоставя инструменти за управление на бази данни през контролния панел. Потребителят може да създаде и управлява бази данни от там.

Потребителят преглежда настройките на хостинг акаунта си, за да настрои сигурността, кеша, резервните копия и други опции, предоставени от InfinityFree.

След като се качат файловете и се настроят всички необходими настройки, потребителят посещава своя сайт, за да се увери, че всичко работи правилно.

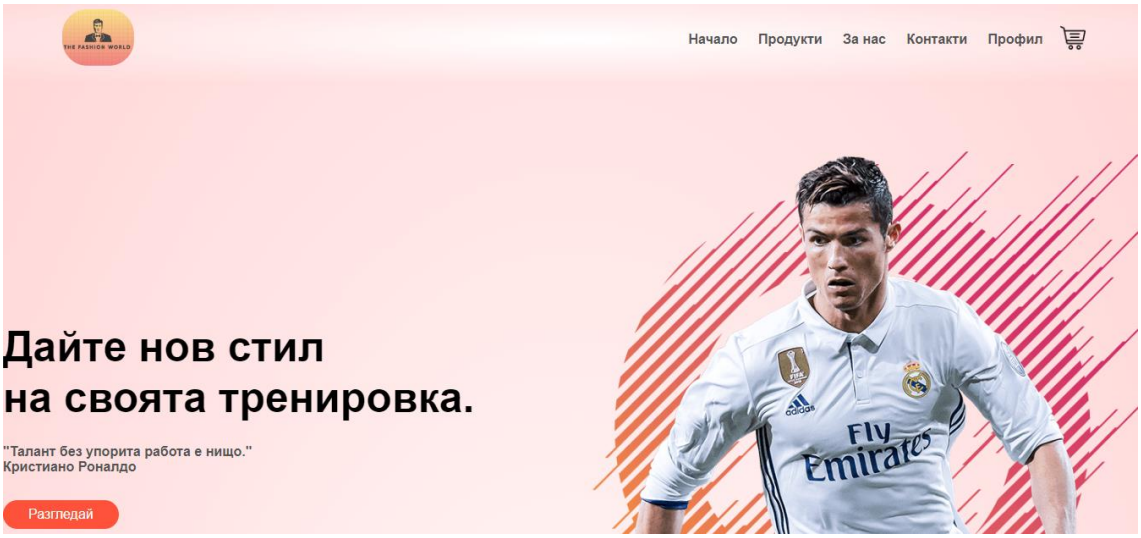
След изпълнение на тези стъпки, уеб приложението трябва да бъде успешно хостнато в InfinityFreeApp.

ГЛАВА 4 - РЪКОВОДСТВО НА ПОТРЕБИТЕЛЯ

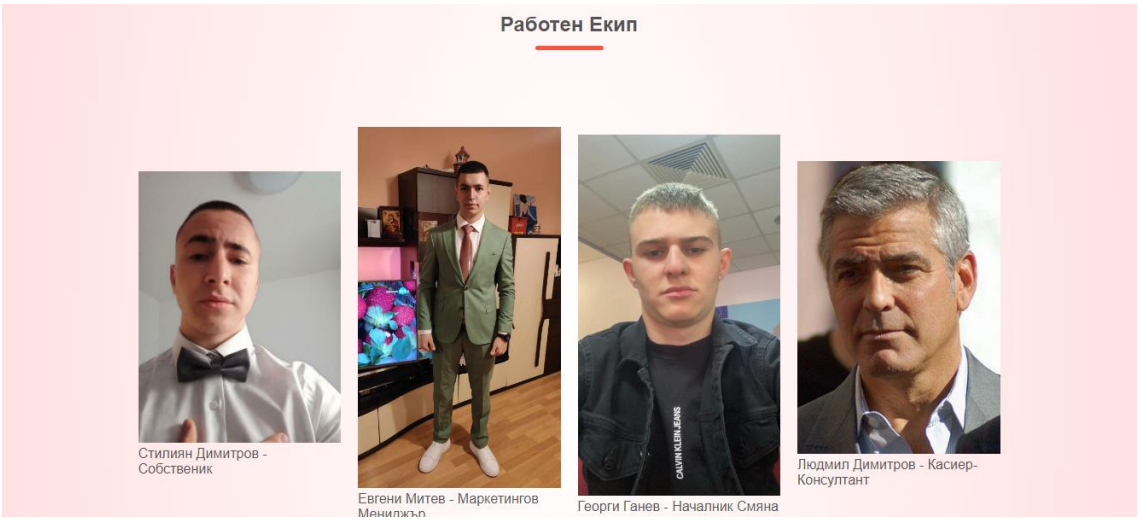
4.1. Представяне на началната страница на приложението:

При отваряне на линка http://thefashionworld.infinityfreeapp.com/ се отваря началната страница на разработения уебсайт, която се вижда на фигура 22. За тест на приложението могат да се използват следните данни: Потребителско име : test1 Парола: test123456@

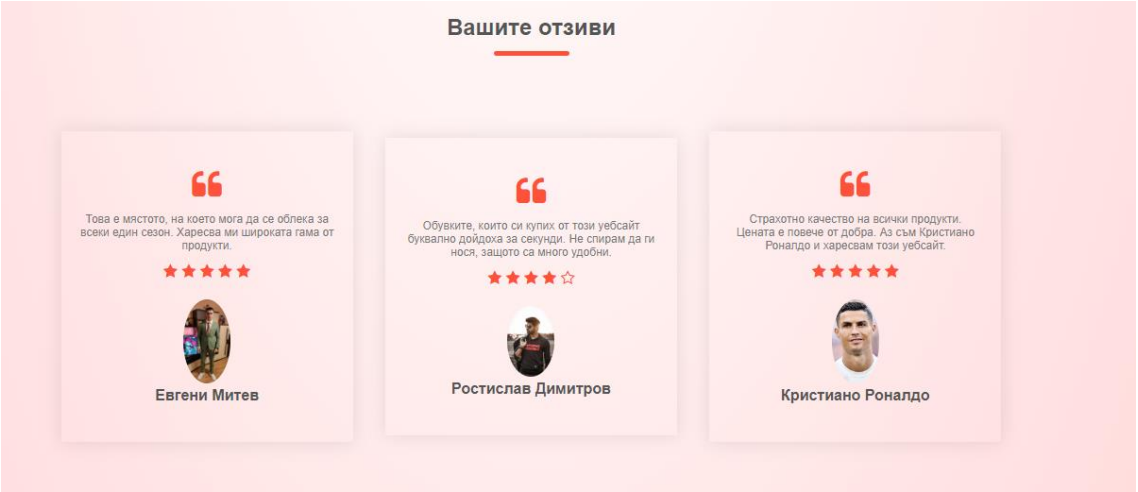
В началната страница е показан работния екип, отзиви на клиентите, , бутон с надпис (Разгледай),който ни отвежда до страницата с продуктите, логото на уеб приложението и навигационното поле, което ни отвежда до другите страници.



Фигура 22.Представяне на начална страница с навигационно меню и лого



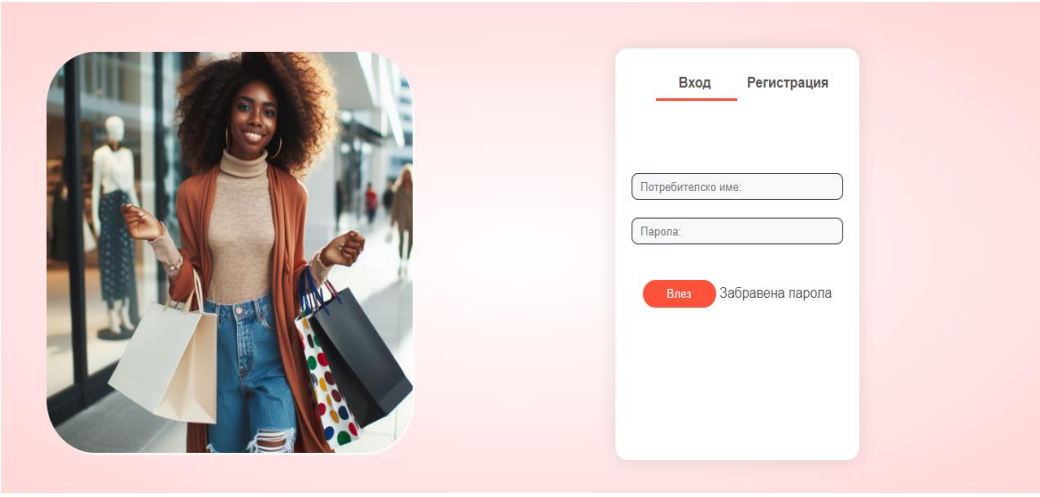
Фигура 23. Представяне на начална страница и информация за работния персонал



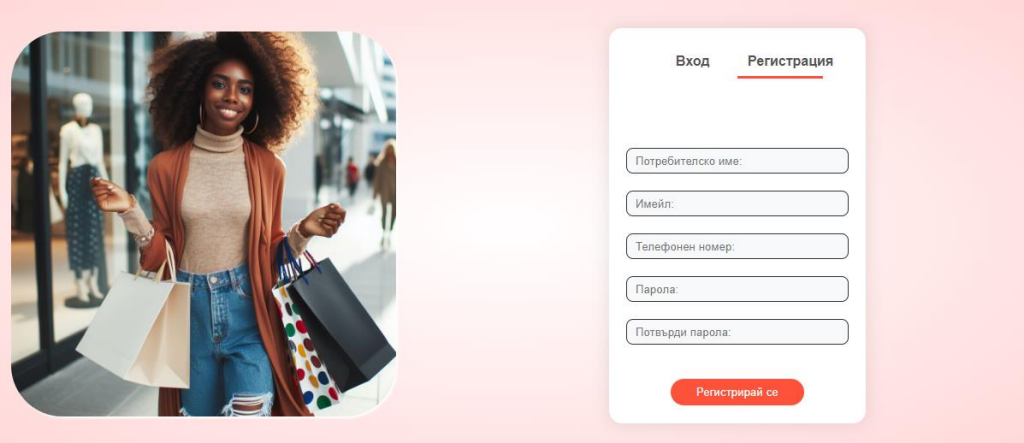
Фигура 24.Представяне на начална страница за отзиви от наши клиенти

4.2. Формуляри за вход и регистрация:

Във формуляра за вход на съществуващ потребител има две полета, в които потребителя трябва да въведе своите потребителско име и парола, а във формуляра за регистрация има полета за потребителско име, имейл, телефонен номер, парола и потвърди паролата.



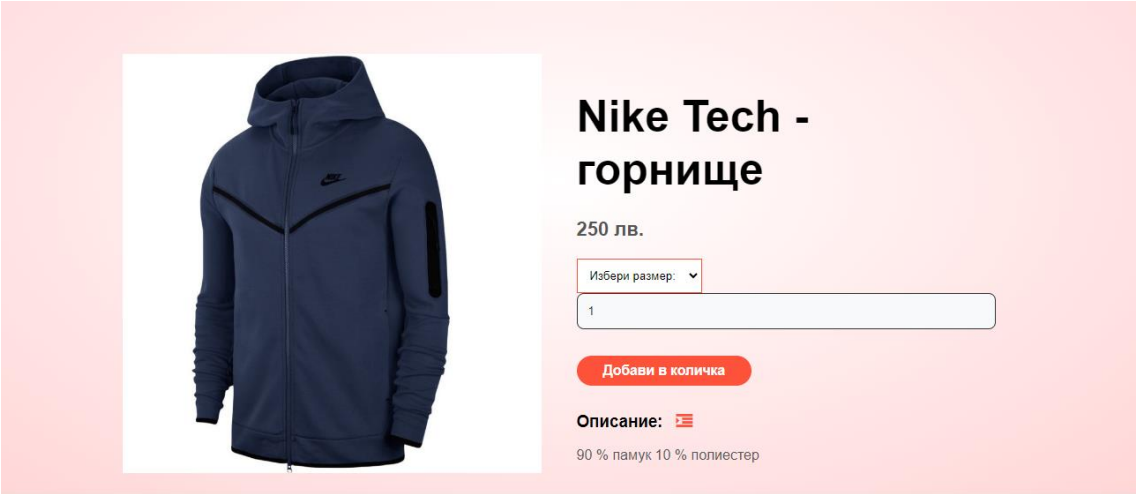
Фигура 25. Представяне на формуляр за вход на вече съществуващи потребители.



Фигура 26. Представяне на формуляр за регистрация на нов потребител.

4.3. Страница на индивидуален продукт:

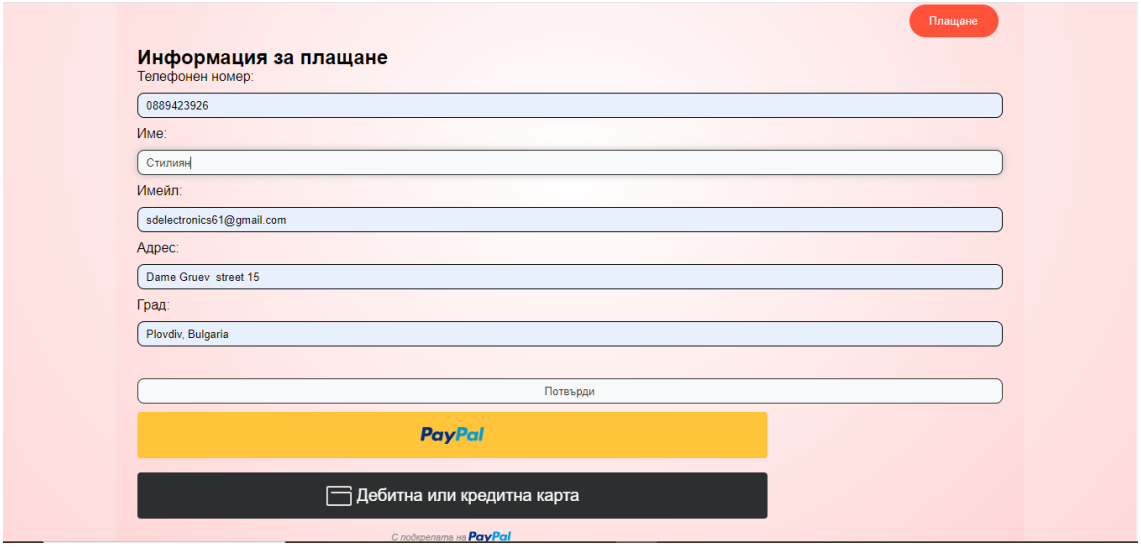
В тази страница има информация за избран от потребителя продукт, като има бутон за добавяне на продукт в количката, опция за избиране на размера на продукта, кратко описание и цена на продукта. Достига се до тази страница, след като потребителят натисне изображението на продукта в страницата за продуктите.



Фигура 27. Представяне на страницата за индивидуалния продукт

4.4. Формуляр за разплащане:

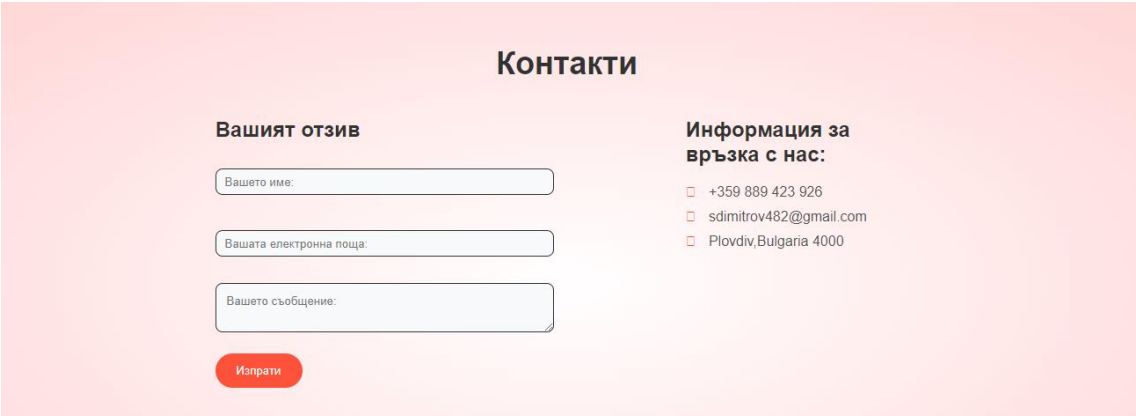
До тази част от страница се достига след като потребителят натисне бутона (Плащане).А след като стигне до нея потребителят попълва формуляра с информацията за плащане и след това избира метода на плащане като той може да избира между две опции PayPal и Дебитна или кредитна карта.



Фигура 28. Представяне на формуляр за избиране на метод за плащане.

4.5. Страница за подаване на съобщение:

Страницата с име Контакти има за цел да предостави на потребителите право да дадат отзив за уеб приложението. Те попълват формуляр с три полета за име, електронна поща и съобщение. След като потребителят е въвел всички полета, натиска бутона Изпрати. До формулярът се намира информация за нас като нашият телефонен номер, имейл и града, в който може да ни намерите.



Фигура 29. Представяне на формуляр за подаване на отзив

4.6. Страница с информация за нас

В тази страница на кратко е описана историята на нашия магазин и има изображение, което го показва.

Фигура 30. Представяне на страницата с информация за нас

4.7. Страница за добавяне на продукт от страна на администратора:

До тази страница има достъп само администратора и само той може да попълва полетата. След като информацията за всяко едно поле е вярна, администраторът избира файл за продукта. Като след това продукта с всички характеристики се показва в страницата Продукти.

Фигура 31. Представяне на формуляра в страница Добави продукт

ЗАКЛЮЧЕНИЕ

Уеб сайтът "The Fashion World" представлява не просто електронен магазин, а цялостно преживяване в модния свят. С широка гама от модни артикули, включващи дрехи от водещи световни марки, този сайт предлага на потребителите вълнуващо пазаруване, където те могат да изразят своя стил и индивидуалност. Възможността за лесно навигиране и избор на продуктите, правят пазаруването в "The Fashion World" удоволствие за всеки любител на модата.

От страна на потребителите, уеб сайтът предлага редица функционалности, включително:

- Преглед на каталога с продукти: Потребителите могат да разглеждат разнообразието от модни артикули
- Добавяне в количката: Потребителите могат да изберат желаните продукти и да ги добавят във виртуалната си количка, преди да продължат към плащане.
- Плащане и доставка: Удобни опции за плащане и доставка, които осигуряват безпроблемно завършване на поръчката.

От страна на администраторите, уеб сайтът предоставя инструменти за управление и контрол на платформата, включително:

- Управление на асортимента: Администраторите могат да добавят, редактират или премахват продукти от каталога, както и да управляват размера и цените им.
- Управление на поръчките: Възможност за преглед и обработка на поръчките, включително потвърждаване на плащания, подготовка за доставка и отчитане на изпълнението им.
- Управление на потребителите: Контрол над потребителските профили, включително създаване на нови профили, надзор на активността и реакция на запитвания и проблеми от страна на клиентите.

Със своите богати функционалности и лесен за управление административен панел, уеб сайтът "The Fashion World" предлага пълноценно модерно онлайн преживяване както за потребителите, така и за администраторите.

ИЗТОЧНИЦИ

1. Google, 2024, html documentation, достъпена към 05.03.2024 от https://developer.mozilla.org/en-US/docs/Web/HTML

2. Google,2024,What Css means?, достъпена към 01.03.2024 от https://www.atinternet.com/en/glossary/css/

3. Google, 2024, javascript documentation, достъпена към 03.03.2024 от https://devdocs.io/javascript/

4. Google,2024, PHP, достъпена към 17.03.2024 от https://www.php.net/docs.php
5. Google,2024, MySQL documentation, достъпена към 18.03.2024 от https://cloud.google.com/sql/docs/mysql

6. Google,2020, AJAX документация, достъпена към 16.03.2024 от https://doc.nette.org/bg/application/ajax

7. Google,2024, PHP myAdmin , достъпена към 10.03.2024 от https://www.phpmyadmin.net/

8. Google,2023, XAMP, достъпена към 16.03.2024 от https://bg.wikipedia.org/wiki/XAMPP

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ 1

```
<?php
    include_once "header.php";


?>


    </div>
  <div class="acount-page">
      <div class="container">
        <div class="row">
            <div class="snimkazalogina">
                <img src="images/loginphoto.jpg" width="100%">
            </div>


            <div class="col-2">
               <div class="form-container">
                  <div class="form-btn">
                     <span onclick="login()">Вход</span>
                     <span
onclick="register()">Регистрация</span>
                     <hr id="Indicator">
                  </div>
                  <form id="LoginForm"
action="includes/login.inc.php" method="POST">
                     <input type="text" name="username"
placeholder="Потребителско име:">
                     <input type="password" name="password"
placeholder="Парола:">
                     <button class="btn" type="submit"
name="submit" value="submit">Влез</button>
                     <a href="">Забравена парола ?</a>
                  </form>
```

```
                <form id="RegForm"
action="includes/signup.inc.php" method="POST">

                    <input type="text" name="username"
placeholder="Потребителско име:">

                    <input type="email" name="email"
placeholder="Имейл:">

                    <input type="phonenumber" name="phone"
placeholder="Телефонен номер:">

                    <input type="password" name="password"
placeholder="Парола:">

                    <input type="password"
name="confirmPassword" placeholder="Потвърди парола:">

                    <button class="btn" type="submit"
name="submit" value="submit">Регистрирай се</button>

                </form>

              </div>

            </div>

        </div>

      </div>

  </div>




        <div class="footer">

          <div class="container">

              <div class="row">

                  <div class="footer-col-1">

                      <h3>Кратко описание:</h3>

                      <p>"The Fashion World" е модерен магазин
за дамски, мъжки и детски дрехи, предлагащ широка гама от
актуални модни тенденции за всеки вкус и повод. </p>


                  </div>


                  <div class="footer-col-3">

                      <h3>Работно време: </h3>
```

```
                    <p> <span>Понеделник –Петък:</span> 8:00 -
19:00ч. </p>
                    <p> <span>Събота:</span> 10:00 - 17:00ч.
</p>
                    <p> <span>Неделя:</span> Затворено</p>
                    <p> <span>Официални  празници:</span>
Затворено</p>
                    <p> <span>Национален празник:</span>
Затворено</p>


                </div>
                <div class="footer-col-4">
                    <h3>Последвайте ни:</h3>
                    <ul>
                        <li><a class="fa fa-facebook" >
Стилиян Димитров</a></li>
                        <li><a class="fa fa-instagram" >
sdimitrov_18</a></li>
                        <li><a class="link"> <i class="fa fa-
mobile-phone"></i> +359-889-423-926 </a></li>
                        <li><a  class="link"> <i class="fa
fa-envelope"></i> sdimitrov482@gmail.com </a></li>
                    </ul>
                </div>
            </div>
            <hr>
            <p class="copyright"> &copy Създадено от Стилиян
Димитров | Всички права запазени!</p>
          </div>
        </div>



        <script>
            var MenuItems =
document.getElementById("MenuItems");
```

```
        MenuItems.style.maxHeight = "0px";
        function menutoggle(){
          if( MenuItems.style.maxHeight == "0px")
          {
              MenuItems.style.maxHeight = "200px";
          }
          else
          {
              MenuItems.style.maxHeight = "0px";
          }
        }
    </script>

    <script>
      var LoginForm =document.getElementById("LoginForm");
      var RegForm =document.getElementById("RegForm");
      var Indicator =document.getElementById("Indicator");

      function register( ){
          RegForm.style.transform ="translateX(0px)";
          LoginForm.style.transform ="translateX(0px)";
          Indicator.style.transform ="translateX(100px)";
      }
      function login( ){
          RegForm.style.transform ="translateX(300px)";
          LoginForm.style.transform ="translateX(300px)";
          Indicator.style.transform ="translateX(0px)";
      }

    </script>
  </body>
  </html>
```

ПРИЛОЖЕНИЕ 2

```
<?php
    include_once "header.php";

    if(!isset($_SESSION["accountType"]) ||
$_SESSION["accountType"] !== 1) {
        header("Location: index.php");
    }
?>

<div class="p-5">
    <div class="container">
        <div class="d-lg-flex justify-content-
between">
            <div class="postForm w-100 pe-lg-5">
                <form action="includes/add.php"
method="POST" enctype="multipart/form-data" class="text-
center">
                    <div class="justify-content-
between pb-2">
                        <?php

if(isset($_GET["productName"]) &&
!empty($_GET["productName"])) {
                            echo '<input
type="text" name="productName" class="form-control
```

```
postInput" placeholder="Име на продукт"
value="'.$_GET["productName"].'">';
                                } else
if(isset($_GET["productName"])) {
                                    echo '<input
type="text" name="productName" class="form-control
postInput error" placeholder="Име на продукт">';
                                } else {
                                    echo '<input
type="text" name="productName" class="form-control
postInput" placeholder="Име на продукт">';
                                }
                            ?>
                        </div>

                        <div class="uploadImg rounded lead
text-center mt-3 center">
                            <div class="title lead">Качи
файл</div>
                            <div class="dropzone mx-auto
my-5">
                                <div class="content p-5">
                                    <img
src="images/upload.svg" class="upload" id="img">
                                    <span
class="filename"></span>
                                    <input type="file"
name="image[]" class="input" multiple>
```

```
                    </div>
                </div>
            </div>

            <div class="justify-content-
between my-4">
                <?php

if(isset($_GET["description"]) &&
!empty($_GET["description"])) {
                    echo '<textarea
name="description" class="form-control postInput py-5"
placeholder="Описание">'.$_GET["description"].'</textarea>'
;
                } else
if(isset($_GET["description"])) {
                    echo '<textarea
name="description" class="form-control postInput py-5
error" placeholder="Описание"></textarea>';
                } else {
                    echo '<textarea
name="description" class="form-control postInput py-5"
placeholder="Описание"></textarea>';
                }
                ?>

            </div>
```

```
<?php
    if(isset($_GET["price"]) &&
!empty($_GET["price"])) {
        echo '<div class="justify-
content-between my-4">
        <input type="number"
name="price" class="form-control postInput"
placeholder="Цена" min="0" max="1500"
value="'.$_GET["price"].'">
    </div>';
    } else
if(isset($_GET["price"])) {
        echo '<div class="justify-
content-between my-4">
        <input type="number"
name="price" class="form-control postInput error"
placeholder="Цена" min="0" max="1500">
    </div>';
    } else {
        echo '<div class="justify-
content-between my-4">
        <input type="number"
name="price" class="form-control postInput"
placeholder="Цена" min="0" max="1500">
    </div>';
    }
?>
```

```
<?php
    if(isset($_GET["size"]) &&
!empty($_GET["size"])) {
        echo '<div class="justify-
content-between my-4">
        <input type="text"
name="size" class="form-control postInput"
placeholder="Размер" value="'.$_GET["size"].'">
    </div>';
    } else
if(isset($_GET["area"])) {
        echo '<div class="justify-
content-between my-4">
        <input type="text"
name="size" class="form-control postInput error"
placeholder="Размер">
    </div>';
    } else {
        echo '<div class="justify-
content-between my-4">
        <input type="text"
name="size" class="form-control postInput"
placeholder="Размер">
    </div>';
    }
?>
```

```
                        <button class="btn btn-lg
submitBtn" type="submit" name="submit">Публикувай</button>
                    </form>
                </div>
            </div>
        </div>
    </div>
```

ПРИЛОЖЕНИЕ 3

```
<?php
include_once "header.php";


// Проверяваме дали е изпратена поръчка
if ($_SERVER["REQUEST_METHOD"] == "POST" &&
isset($_POST["phone"]) && isset($_POST["name"]) &&
isset($_POST["email"]) && isset($_POST["address"]) &&
isset($_POST["city"]) && isset($_POST["totalAmount"]) &&
isset($_POST["products"])) {
    // Проверяваме дали сесията е стартирана
    if (session_status() == PHP_SESSION_NONE) {
        session_start();
    }


    // Изчистваме количката, ако съществува
    if (isset($_SESSION["cart"])) {
        unset($_SESSION["cart"]);
    }


    header("Location: index.php?success=true"); //
Пренасочваме към index.php със съобщение за успешна поръчка
    exit();
}
?>


</div>
<!-- Cart-Items details -->
<div class="small-container cart-page">
    <?php if (empty($_SESSION['cart'])) : ?>
        <p>Количката е празна.</p>
    <?php else : ?>
```

```
            <table>
                <tr>
                    <th>Продукт</th>
                    <th>Количество</th>
                    <th>Междинна сума</th>
                    <th>Изтрий</th> <!-- Добавен ред за
бутона за изтриване -->
                </tr>
                <?php
                // Инициализираме променливата за
междинна сума
                $intermediateTotal = 0;
                foreach ($_SESSION["cart"] as $index =>
$cartItem) {
                    $sql = "SELECT * FROM products WHERE
productId = " . $cartItem['id'];
                    $result = mysqli_query($conn, $sql);
                    $row = mysqli_fetch_assoc($result);

                    $sql2 = "";
                    if (!empty($row["images"])) {
                        $imagesArray = explode(" ",
$row["images"]);
                        if (is_array($imagesArray) &&
count($imagesArray) > 0) {
                            $sql2 = "SELECT * FROM images
WHERE id = " . $imagesArray[0];
                        }
                    }
                    $result2 = mysqli_query($conn,
$sql2);
                    $row2 = mysqli_fetch_assoc($result2);

                    if ($row && $row2) {
```

```
                            // Изчисляваме базовата цена на
продукта
                            $price = str_replace('.', ',',
$row["price"]);

                            foreach ($cartItem as $item) {
                                if (isset($item["size"]) &&
isset($item["count"])) {
                                    $size = $item["size"];
                                    $count = $item["count"];

                                    // Изчисляваме междинната
сума за текущия продукт
                                    $productIntermediateTotal
= $count * $price;
                                    $intermediateTotal +=
$productIntermediateTotal;

                                    ?>
                                    <tr>
                                        <td>
                                            <div class="cart-
info">
                                                <img
src="includes/<?php echo $row2['img_dir'] ?>">
                                                <div>
                                                    <p><?php
echo $row["name"] ?> - <?php echo $size ?></p>
<small>Цена: <?php echo $price ?> лв.</small>
                                                </div>
                                            </div>
                                        </td>
                                        <td class="quantity-
cell">
```

```
                                        <form
onsubmit="event.preventDefault(); updateQuantity(<?php echo
$cartItem['id']; ?>, <?php echo $index; ?>);">

                                            <input
type="number" name="quantity" value="<?php echo $count; ?>"
min="1">

                                            <button
type="submit">Обнови</button>

                                        </form>

                                    </td>

                                    <td
class="subtotal"><?php echo $productIntermediateTotal ?>
лв.</td>

                                    <td><button
onclick="deleteFromCart(<?php echo $cartItem['id']; ?>,
<?php echo $index; ?>)">Изтрий</button></td> <!-- Бутон за
изтриване -->

                                </tr>

                                <?php

                            }

                        }

                    }

                    ?>

                </table>

            <?php endif; ?>


            <div class="total-price">

                <table>

                    <tr>

                        <td>Междинна сума:</td>

                        <td id="intermediateTotal"><?php echo
isset($intermediateTotal) ?
number_format($intermediateTotal, 2) : '0.00'; ?> лв.</td>

                    </tr>
```

```
                <tr>
                    <td>ДДС:</td>
                    <td id="vat"><?php echo
isset($intermediateTotal) ?
number_format($intermediateTotal * 0.2, 2) : '0.00'; ?>
лв.</td>
                </tr>
                <tr>
                    <td>Обща сума:</td>
                    <td id="total"><?php echo
isset($intermediateTotal) ?
number_format(($intermediateTotal + $intermediateTotal *
0.2), 2) : '0.00'; ?> лв.</td>
                </tr>
                <tr>
                    <td colspan="2">
                        <button
onclick="showPaymentForm()">Плащане</button>
                    </td>
                </tr>
            </table>
        </div>
        <script
src="https://www.paypal.com/sdk/js?client-
id=ASIMF9IP0pjVrfVuduP3s_SWy6YH07GjPxfuCuv1nQ1A_XAMp3_8-
OdgZP6-dZ739mA-moVIdS6ag1dB&currency=USD"></script>
        <!-- Payment Form -->
        <div id="paymentForm" style="display: none;">
            <h2>Информация за плащане</h2>
            <form id="paymentFormSubmit"
onsubmit="submitPaymentForm(event)">
                <label for="phone">Телефонен
номер:</label><br>
                <input type="text" id="phone"
name="phone" required><br>
```

```
                    <label for="name">Име:</label><br>
                    <input type="text" id="name" name="name"
required><br>
                    <label for="email">Имейл:</label><br>
                    <input type="email" id="email"
name="email" required><br>
                    <label for="address">Адрес:</label><br>
                    <input type="text" id="address"
name="address" required><br>
                    <label for="city">Град:</label><br>
                    <input type="text" id="city" name="city"
required><br><br>
                    <input type="submit" value="Потвърди"
id="submitPaymentButton">
      <div id="paypal-button-container"></div>


                </form>
            </div>
      </div>


      <div class="footer">
          <div class="container">
              <div class="row">
                  <div class="footer-col-1">
                      <h3>Кратко описание:</h3>
                      <p>"The Fashion World" е модерен
магазин за дамски, мъжки и детски дрехи, предлагащ широка
гама от актуални модни тенденции за всеки вкус и повод.
</p>


                  </div>


                  <div class="footer-col-3">
                      <h3>Работно време: </h3>
```

```
                    <p> <span>Понеделник -Петък:</span>
8:00 - 19:00ч. </p>
                    <p> <span>Събота:</span> 10:00 -
17:00ч. </p>
                    <p> <span>Неделя:</span> Затворено</p>
                    <p> <span>Официални  празници:</span>
Затворено</p>
                    <p> <span>Национален празник:</span>
Затворено</p>


                </div>
                <div class="footer-col-4">
                    <h3>Последвайте ни:</h3>
                    <ul>
                        <li><a class="fa fa-facebook" >
Стилиян Димитров</a></li>
                        <li><a class="fa fa-instagram" >
sdimitrov_18</a></li>
                        <li><a class="link"> <i class="fa
fa-mobile-phone"></i> +359-889-423-926 </a></li>
                        <li><a  class="link"> <i
class="fa fa-envelope"></i> sdimitrov482@gmail.com
</a></li>
                    </ul>
                </div>
            </div>
            <hr>
            <p class="copyright"> &copy Създадено от
Стилиян Димитров | Всички права запазени!</p>
          </div>
        </div>


      <script src="https://code.jquery.com/jquery-
3.7.1.min.js"
```

```
            integrity="sha256-
/JqT3SQfawRcv/BIHPThkBvs0OEvtFFmqPF/lYI/Cxo="
            crossorigin="anonymous"></script>
      <script>
          function updateButtonState() {
          var phone =
document.getElementById("phone").value;
          var name = document.getElementById("name").value;
          var email =
document.getElementById("email").value;
          var address =
document.getElementById("address").value;
          var city = document.getElementById("city").value;
          var submitPaymentButton =
document.getElementById("submitPaymentButton");
          var paypalButton =
document.getElementById("paypal-button-container");

          if (phone && name && email && address && city) {

submitPaymentButton.removeAttribute("disabled");

submitPaymentButton.classList.remove("disabled");
              paypalButton.style.display = "block"; //
Показваме бутона на PayPal
          } else {
              submitPaymentButton.setAttribute("disabled",
"disabled");

submitPaymentButton.classList.add("disabled");
              paypalButton.style.display = "none"; //
Скриваме бутона на PayPal
          }
      }
```

```
      // Добавяме слушатели за промяна на съдържанието на
полетата за плащане
      document.getElementById("phone").addEventListener("in
put", updateButtonState);
      document.getElementById("name").addEventListener("inp
ut", updateButtonState);
      document.getElementById("email").addEventListener("in
put", updateButtonState);
      document.getElementById("address").addEventListener("
input", updateButtonState);
      document.getElementById("city").addEventListener("inp
ut", updateButtonState);


      // Извикваме функцията updateButtonState() при
зареждане на страницата
      document.addEventListener("DOMContentLoaded",
updateButtonState);


      // Функция за показване на формата за плащане
      function showPaymentForm() {

document.getElementById("paymentForm").style.display =
"block";
      }


      // Функция за изпращане на формата за плащане
      function submitPaymentForm(event) {
          event.preventDefault();
          // Останалите части от кода остават непроменени
          // ...
      }


      // PayPal Button
      paypal.Buttons({
```

```
            createOrder: function(data, actions) {
                // This function sets up the details of
the transaction, including the amount and line item
details.
                return actions.order.create({
                    purchase_units: [{
                        amount: {
                            value: '<?php echo
isset($intermediateTotal) ? $intermediateTotal : '0.00';
?>', // Тук посочете общата сума за плащане
                        }
                    }]
                });
            },
            onApprove: function(data, actions) {
          // This function captures the funds from the
transaction.
          return
actions.order.capture().then(function(details) {
            // This function shows a transaction success
message to your buyer.
            alert('Транзакцията е успешна!');

            // Събиране на данните от количката
            var cart = <?php echo
json_encode($_SESSION['cart']); ?>;

            // Пренасочване към placePaypalOrder.php с
добавени данни от количката
            window.location.href =
'placePaypalOrder.php?cart=' +
encodeURIComponent(JSON.stringify(cart));
          });
            }
          }).render('#paypal-button-container');
```

```
        // Проверка за запазено количество в Local
Storage и актуализация на формите за количеството
        document.addEventListener("DOMContentLoaded",
function() {
            var cartItems = <?php echo
json_encode($_SESSION["cart"]); ?>;
            cartItems.forEach(function(cartItem, index) {
                var storedQuantity =
localStorage.getItem("cart_" + cartItem.id);
                if (storedQuantity !== null) {
                    var inputElement =
document.querySelectorAll("table
tr")[index].querySelector(".quantity-cell
input[name='quantity']");
                    if (inputElement) {
                        inputElement.value =
storedQuantity;
                    }
                }
            });
            updateTotalPrice(); // Актуализация на
междинната и общата сума
        });

        // Функция за обновяване на количеството
        function updateQuantity(productId, index) {
            var inputElement =
document.querySelectorAll("table
tr")[index].querySelector(".quantity-cell
input[name='quantity']");
            var newQuantity = inputElement ?
inputElement.value : 0;
            $.ajax({
                url: "/includes/updateQuantity.php",
                type: "POST",
```

```
                    data: { productId: productId, quantity:
newQuantity, index: index },
                    success: function(data) {
                        // Ако актуализацията е успешна,
пресметнете новата междинна сума и обща сума
                        updateTotalPrice();


                        // Запазване на текущото количество в
Local Storage
                        localStorage.setItem("cart_" +
productId, newQuantity);


                        // Реагиране на отговора от сървъра
                        var response = JSON.parse(data);
                        if (response.success) {
                            // Успешно актуализиране на
количеството в сесията
                            // Актуализация на DOM елементите
на страницата с новите данни
                            var intermediateTotalCell =
document.getElementById("intermediateTotal");
                            intermediateTotalCell.innerText =
response.newIntermediateTotal.toFixed(2) + ' лв.';


                            var vatCell =
document.getElementById("vat");
                            vatCell.innerText =
(response.newIntermediateTotal * 0.2).toFixed(2) + ' лв.';


                            var totalCell =
document.getElementById("total");
                            totalCell.innerText =
(response.newIntermediateTotal * 1.2).toFixed(2) + ' лв.';
                        } else {
```

```
                        // Показване на съобщение за
грешка
                        console.log("Грешка при
актуализиране на количеството в сесията.");
                    }
                },
                error: function(xhr, status, error) {
                    console.log("Error:", error);
                    alert("Грешка при обновяване на
количеството.");
                }
            });
        }

        // Функция за изтриване на продукт от количката
        function deleteFromCart(productId, index) {
            $.ajax({
                url: "/includes/deleteFromCart.php",
                type: "POST",
                data: { productId: productId, index:
index },
                success: function(data) {
                    // Ако изтриването е успешно,
презареждаме страницата
                    location.reload();
                },
                error: function(xhr, status, error) {
                    console.log("Error:", error);
                    alert("Грешка при изтриване на
продукта от количката.");
                }
            });
        }
```

```
        // Функция за актуализация на междинната сума и
общата сума
        function updateTotalPrice() {
            var intermediateTotal = 0;
            var rows = document.querySelectorAll("table
tr");
            for (var i = 1; i < rows.length; i++) {
                var inputElement =
rows[i].querySelector(".quantity-cell
input[name='quantity']");
                if (inputElement) {
                    var quantity =
parseInt(inputElement.value);
                    var priceElement =
rows[i].querySelector(".cart-info
small").innerText.split(": ")[1];
                    var price = parseFloat(priceElement);
                    intermediateTotal += quantity *
price;
                    var subtotalCell =
rows[i].querySelector(".subtotal");
                    subtotalCell.innerText = (quantity *
price).toFixed(2) + ' лв.';
                }
            }
            var vat = intermediateTotal * 0.2;
            var total = intermediateTotal + vat;

document.getElementById("intermediateTotal").innerText =
intermediateTotal.toFixed(2) + ' лв.';
            document.getElementById("vat").innerText =
vat.toFixed(2) + ' лв.';
            document.getElementById("total").innerText =
total.toFixed(2) + ' лв.';
        }
```

```
        // Функция за показване на формата за плащане
        function showPaymentForm() {

document.getElementById("paymentForm").style.display =
"block";
        }


        // Функция за изпращане на формата за плащане
        function submitPaymentForm(event) {
        event.preventDefault();
        var phone =
document.getElementById("phone").value;
        var name = document.getElementById("name").value;
        var email =
document.getElementById("email").value;
        var address =
document.getElementById("address").value;
        var city = document.getElementById("city").value;
        var intermediateTotal =
parseFloat(document.getElementById("intermediateTotal").inn
erText.replace(",", ".")); // Междинна сума
      var vat =
parseFloat(document.getElementById("vat").innerText.replace
(",", ".")); // ДДС
      var totalAmount = (intermediateTotal +
vat).toFixed(2).replace(".", ","); // Обща сума


        var products = [];


        var rows = document.querySelectorAll("table tr");
        rows.forEach(function(row) {
            var productNameElement =
row.querySelector(".cart-info p");
```

```
            var quantityInput =
row.querySelector(".quantity-cell input[name='quantity']");
            var priceCell =
row.querySelector(".subtotal");


            if (productNameElement && quantityInput &&
priceCell) {
                var productName =
productNameElement.innerText.split(" - ")[0];
                var size =
productNameElement.innerText.split(" - ")[1];
                var quantity = quantityInput.value;
                var price =
parseFloat(priceCell.innerText.split(" ")[0]);
                var productData = {
                    name: productName,
                    size: size,
                    quantity: quantity
                };
                products.push(productData);
            }
        });

        $.ajax({
            url: "placeOrder.php",
            type: "POST",
            data: {
                phone: phone,
                name: name,
                email: email,
                address: address,
                city: city,
```

```
                totalAmount: totalAmount, // Изпращаме
общата сума
                products: JSON.stringify(products)
            },
            success: function(data) {
                alert("Поръчката е приета успешно!");
                location.href = "index.php?success=true";
// Пренасочваме към index.php със съобщение за успешна
поръчка
            },
            error: function() {
                alert("Възникна грешка при плащането.");
            }
        });
    }
    </script>

    </body>
    </html>
```