****

**Пловдивския университет „Паисий Хилендарски“**

**Факултет по математика и информатика**

**Дипломна работа**

**Електронен магазин за електроника със складови функционалности**

Изработил:............... Научен ръководител.................

Мустафа Ахмед ас. Йордан Тодоров

Съдържание

[Увод 2](#_Toc518407843)

[1. Анализ на проблемната област 3](#_Toc518407844)

[1.1 Какво прави уеб сайтовете необходими в съвременния живот? 4](#_Toc518407845)

[1.2 Изисквания към приложението 5](#_Toc518407846)

[2. Използвани технологии и средства за програмиране 7](#_Toc518407847)

[2.1. MySQL 7](#_Toc518407848)

[2.2. HTML (Hyper Text Markup Language) 7](#_Toc518407849)

[2.3. JavaScript 7](#_Toc518407850)

[2.3. AJAX 8](#_Toc518407851)

[2.4. PHP 8](#_Toc518407852)

[3. Архитектура на Уеб Приложение 9](#_Toc518407853)

[3.1. Избор на софтуер за приложението 9](#_Toc518407854)

[3.2. Избор на език за програмиране 9](#_Toc518407855)

[3.3. Избор на технология за реализиране на приложението 10](#_Toc518407856)

[3.4. Use Case диаграми 14](#_Toc518407857)

[3.4.1. Уеб Клиент актьор 14](#_Toc518407858)

[4. Разработка на приложението 16](#_Toc518407859)

[4.1. Проектиране на базата данни 16](#_Toc518407860)

[4.1.1. Използвани таблици 17](#_Toc518407861)

[4.2. Контролери 19](#_Toc518407862)

[4.3. Модели 26](#_Toc518407863)

[5. Ръководство на потребителя 29](#_Toc518407864)

[5.1. Управление на потребителски интерфейс 29](#_Toc518407865)

[5.1.1 Регистрация на потребител 29](#_Toc518407866)

[5.1.2. Вход на потребител в системата 30](#_Toc518407867)

[5.1.3. Обратна връзка с клиент 31](#_Toc518407868)

[5.1.4. Търсене на продукти по различни критерии 32](#_Toc518407869)

[5.1.5. Поръчване на даден продукт 33](#_Toc518407870)

[5.1.6. Промяна на потребителския профил 36](#_Toc518407871)

[5.2. Администраторски панел 37](#_Toc518407872)

[5.2.1 Основни настройки на магазина 39](#_Toc518407873)

[5.2.2. Управление на каталозите 42](#_Toc518407874)

[5.2.3.Управление на продукти 44](#_Toc518407875)

[5.2.4 Управление на хората 48](#_Toc518407876)

[5.2.5. Управление на Поръчки 50](#_Toc518407877)

[5.2.6. Качване на изображение 54](#_Toc518407878)

[5.2.7. Поща 56](#_Toc518407879)

[5.2.8. Купони 56](#_Toc518407880)

[ЗАКЛЮЧЕНИЕ 59](#_Toc518407881)

[ИЗПОЛЗВАНА ЛИТЕРАТУРА 59](#_Toc518407882)

# **Увод**

С навлизането на съвременните технологии в ежедневието ни, ние отделяме все по-малко време на пазаруването по щандовете и магазините. Напредъкът на компютърните технологии и Интернет предизвикват „революция“ в търговията, разкривайки нови възможности пред фирмите. С разрастването на така наречената „електронна търговия“ бавно, но сигурно се променят старите навици на пазаруване. Чрез навлизането и разрастването на електронната търговия постепенно ще се разреши проблема с разстоянието между продавача на дадена стока и нейния потребител. Глобалната мрежа ще премахне пространствените бариери и вече няма да е необходимо за една компания да присъства физически в даден географски район, за да осъществява бизнеса си там. Вместо това фирмата ще използва виртуалното си присъствие за транзакциите си с клиентите, като разчита на куриерски услуги за доставка на продукта. Понякога обаче конкурентните предимства на виртуалните магазини може да бъдат неутрализирани от конкретните нужди на клиентите. Такава ситуация има в случаи, когато потребителят желае незабавното придобиване на стоката, което е невъзможно през мрежата.

Целта на дипломната работа е създаване на динамичен уеб сайт за продажба на компютърни компоненти. В процеса на разработка ще преминем от етап планиране до същинската реализация на сайта, ще разгледаме кои критерии правят един сайт атрактивен и защо се налага използването на сайтове. Крайният резултат от дипломната ще бъде един напълно функционален сайт.

С цел придобиване на обща представа за приложението на подобни сайтове, ще бъдат разгледани вече съществуващи такива, а тяхната функционалност ще бъде използвана за да се определи какво място заема предложеното решение сред сайтове, които реализират реална печалба.

# **Анализ на проблемната област**

Основното предназначение на всеки сайт е функцията му да информира. В Интернет пространството се среща огромно разнообразие от сайтове и всеки от тях може да се причисли към дадена група, която да съответства на неговата цел, тематика и насоченост. Такива сайтове могат да бъдат:

* Образователни сайтове;
* Сайтове за сваляне на музика и филми;
* Онлайн магазини;
* Фирмени сайтове;
* Социални мрежи;
* Сайтове за забавление.

Всяко място в Интернет, което може да бъде посетено, където може да се получи или да се сподели информация се причислява към групата на сайтовете. Съществуват различни видове и всеки от тях се различава от останалите по информацията, която предоставя на посетителите си.

В Интернет човек може да развие бизнеса си, да използва свободния достъп до информация и да го обърне в печалба. Интернет не е „нова икономика“, той се подчинява на същите икономически закони, но разширява границите на класическите методи в маркетинга, търговията, обществените контакти и рекламата. Информацията, която се предоставя в сайтове се актуализира лесно и е достъпна непосредствено след актуализацията.

Уеб приложенията представляват един голям пазар, имащ за цел да улесни и автоматизира пазаруването от домовете, като по този начин да спести време и пари на клиентите си. Днес броят на такива приложения е огромен и продължава да расте, ето защо пред нас изникват следните въпроси:

* Какво прави нашия сайт по-различен от останалите?
* Кое го прави атрактивен?

За да се отговори на тези въпроси трябва да се следват основните стъпки при изграждането на един сайт. Още в процеса на планирането трябва да се даде отговор на следните въпроси.

* Защо е необходимо да се създаде сайт?
* За кого е предназначен?
* Какво съдържание ще има сайта?

Основната цел, която има моето приложение е да извършва продажба на компютърни компоненти. Целевата група, към която е предназначен е всеки посетител, който може да удовлетвори своя потребност от предлаганите от мен услуги.

Съдържанието на един сайт играе ключова роля в това дали той ще бъде успешен или неуспешен. Наложили са се следните практики при подготвянето на информацията:

* Предлагане на същностна информация;
* Предлагане на цялата нужна информация;
* Предлагане на допълнителна информация;

Основните критерии, които правят един сайт успешен са:

* **Дизайн** – дизайнът е първото нещо, с което се сблъсква посетителите. Той трябва да вдъхва увереност в тях, че работят със сериозна фирма и че ще получат качествена услуга;
* **Посещаемост** – тя дава шанс за самооценка до каква степен потребителите намират сайтът за полезен;
* **Ползваемост** – един сайт винаги трябва да поставя клиентите си на първо място, тоест трябва да е интуитивен и лесен за използване;
* **Конвертируемост** – без съмнение добрият сайт е този, който превръща посетителите си в клиенти;
* **Поддръжка** – несъмнено един сайт трябва да бъде лесен за поддръжка, което да позволи той да бъде актуален за по-дълго време, тоест да задържи своите клиенти.

## 1.1 Какво прави уеб сайтовете необходими в съвременния живот?

Използването на уеб сайтове е изключително мощно оръжие в маркетинга. Един сайт е полезен за една фирма, защото:

* **Работи 24 часа в денонощието** – сайтът работи без почивка и предоставя достоверна и пълна информация 24 часа в денонощието, 365 дни в годината;
* **Намалява разходите в офиса** – сайтът напълно автоматизира представянето на офертите, спецификациите на продуктите, условията за ползването и цените, което облекчава служители;
* **Повишава се доверието във фирмата** – фирмите, които разполагат със собствени сайтове се ползват с висок кредит на доверие сред клиентите;
* **Дава възможност за обратна връзка с клиента** –клиентът може да остави своето мнение, което би могло да спомогне за подобряване на качеството на предлаганата услуга;
* **Дава глобално покритие –** предлаганите от вас услуги и продукти не са ограничени за определен географски район.

## 1.2 Изисквания към приложението

Тъй като разработката трябва да е съобразена с конкретни фирмени нужди и цели, първата стъпка при разработването на едно приложение е определяне и анализиране на потребителските изисквания. Това е важно, за дефиниране на способностите на приложението. За всяко уеб приложение съществуват два типа потребители. Първият е привилегированият потребител, който има достъп до функции на приложението посредством своя браузър. Вторият е администратор, който управлява сайта и има достъп до всички функции.

За да запазя приложението достатъчно просто съм наложил следните изисквания:

* Изисквания към крайния клиент:
* Потребителите трябва да могат да използват приложението от произволен браузър, който поддържа HTML и *бисквитки1*;

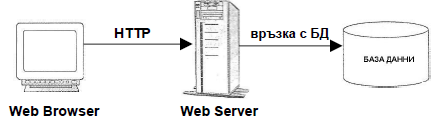
1 *бисквитки* са пакет информация, изпратен от [уеб сървър](http://bg.wikipedia.org/wiki/%D0%A3%D0%B5%D0%B1_%D1%81%D1%8A%D1%80%D0%B2%D1%8A%D1%80)а към [браузър](http://bg.wikipedia.org/wiki/%D0%91%D1%80%D0%B0%D1%83%D0%B7%D1%8A%D1%80)а, а след това връщан от браузъра всеки път, когато той получи достъп до този сървър. Обикновено те се използват за удостоверяване на самоличността на регистриран потребител на даден уеб сайт.

* Новите потребители трябва да могат да се регистрират без намесата на администратора, като всеки потребител ще се разпознава чрез уникален потребителски идентификатор;
* Транзакциите на потребителя трябва да са защитени, тоест в приложението трябва да се вгради някакъв базов механизъм за удостоверяване, който да не позволява на неупълномощени лица да извършват транзакции от името на потребителя;
* Потребителите трябва да могат да закупуват компютърни части посредством приложението;
* Потребителите трябва да могат да разглеждат пълния списък на наличните в сайта продукти;
* Потребителите трябва да могат да променят личната си информация в профила по всяко време;
* Потребителите трябва да могат да търсят продукти по определени критерии.
* Изисквания към администратора:
* Администраторът трябва да може да управлява приложението чрез своя браузър;
* Администраторът трябва да може да управлява поръчките;
* Администраторът трябва да може да следи списъка на регистрираните потребители във всеки един момент;
* Администраторът трябва да може да наблюдава броя на отворените сесии и тяхната продължителност;
* Администраторът трябва да може да освобождава „заключените“ потребители;
* Администраторът трябва да може да премахва коментари от *блога2*;
* Администраторът трябва да може да добавя събития в календара;
* Администраторът трябва да може да добавя снимки към галерията.

2 *блогът* е информационен сайт, публикуван на World Wide Web, и се състой от различни теми („posts“), които обикновено се показват в обратен хронологичен ред.

## 1.3. Избор на софтуер за приложението

Изборът на софтуер, който да се използва за създаване на приложението ще окаже ефект върху гъвкавостта, производителността и динамичността на приложението. Изискванията за приложението са то да е уеб-базирано с клиентска част (браузър). Приложението има нужда от база данни, в която да съхранява потребителските транзакции, информацията за всички потребители, коментари, снимки и т.н. Приложението ще има и среден слой (т.е. уеб-сървър, който ще изпълнява PHP скриптове) за обработката на заявките към приложението изпратени през браузър. Браузърът ще изпраща *HTTP*3 заявки към средния слой. Средният слой ще взима данни от база данните в последния слой, ще извършва някаква обработка на данните и ще изпраща отговор на клиента.



**Фигура 1-1 Комуникация между клиент и сървър в едно уеб приложение.**

## 1.4. Избор на език за програмиране

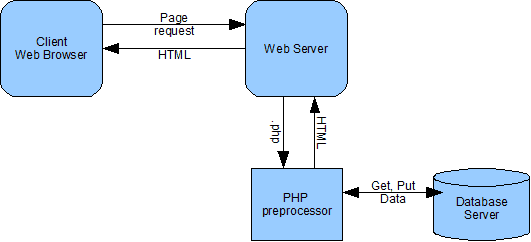
За създаване, но моето приложение избрах да използвам PHP. PHP е широко използван скриптов език с отворен код, предназначен за обща употреба, който е изключително удобен за уеб разработки и може да се вгражда в HTML 5.

**За какво се използва PHP ?**

PHP се използва най-често за обработка на информация при генерирането на динамични интернет страници, поради отличната поддръжка на редица[***бази данни***](http://www.parvisait.com/database.html) като MySQL, Oracle и PostgreSQL. PHP предлага над 800 вградени функции, групирани в отделни библиотеки.

**Защо използването му е удобно за нашето уеб приложение ?**

За разлика от клиентско-ориентирани езици като [JavaScript](http://www.parvisait.com/javascript.html), PHP командите се изпълняват от сървъра. Когато сървърът получи заявка за дадена страница от клиент (най-често браузър), той първо проверява разширението на файла-страница. Ако то е .php, сървърът разпознава файла като PHP, и изпълнява отделните php команди. Това може да са команди за зареждане на необходима информация от външен файл или база данни, обработка на данни и т.н. След като обработката приключи, резултатът се изпраща към клиента като [HTML](http://www.parvisait.com/html-introduction.html) код. Браузърът няма как да разбере коя част на една страница е била създадена с обикновен HTML код и коя е генерирана от PHP. Цялата страница изглежда като създадена само с HTML.



**Фигура 1-2 Как се генерира съдържанието на една динамична страница**

## 1.5. Избор на технология за реализиране на приложението

В процеса на планиране бях изправен пред избор как да напиша своето приложение, дали да бъде с процедурен код, дали да бъде с обектно ориентиран код или някакво средно решение. Всяка една от посочените опции предлага добри условия за различни класове приложения. Процедурният код отпадна още в самото начало, тъй като той има два много големи недостатъка:

* Проследяването на кода е сложно и отнема прекалено много време;
* Поддръжката на системата и нейното доразвиване са равносилни на изцяло нова разработка.

Технологията, която съм използвал за разработка на уеб сайта е **Framework Laravel**.

**Какво е Laravel?**

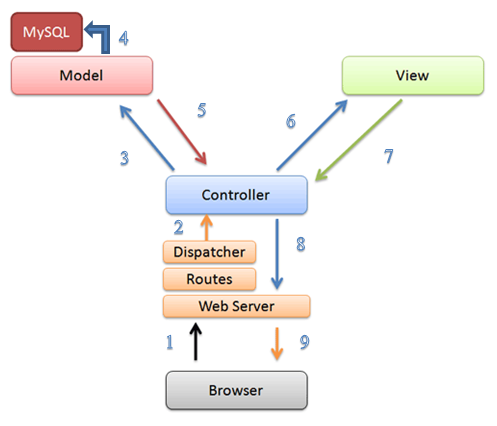
***Laravel*** е мощен и същевременно изключително лек PHP Framework. Той е създаден за тези PHP разработчици, които искат да създават бързо и лесно богати на функционалност уеб приложения. ***Laravel*** е MVC (Models-Views- Controllers) базиран, което ще рече, че всички обръщения към база от данните ще се съхраняват в така наречените Models, целият HTML код ще се съхранява във Views файловете и разбира се логиката на приложението в Controllers.

**Какво е MVC ?**

Преди да продължа с ***Laravel***, нека да кажа каква е основната концепция на MVC.Модел-Изглед-Контролер (MVC) е архитектурен [шаблон за дизайн](http://bg.wikipedia.org/wiki/%D0%A8%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD_%D0%B7%D0%B0_%D0%B4%D0%B8%D0%B7%D0%B0%D0%B9%D0%BD) в програмирането, основан на разделянето на бизнес логиката от графичния интерфейс и данните в дадено приложение.

MVC работи по следния алгоритъм:

* Изпращане на заявка от клиента (браузър);
* Уеб сървърът обработва заявката и я изпраща на рутера;
* Рутерът взима решение кой контролер да бъде извикат според подадените на входа данни;
* Dispatcher предава управлението на контролера;
* Контролерът от своя страна зарежда съответния модел и му предава всички POST и GET параметри, които са подадени през браузъра;
* Моделът извършва съответните функции и ако е необходимо се обръща към база данни, след което връща резултата отново на контролера;
* Той предава резултатите като масив на даден изглед (обикновено шаблон );
* Изгледът използва подадените му параметри за да подготви заявената от потребителя страница, след което връща резултата като html страница на контролера;
* Той от своя страна предава HTML файла на уеб сървъра;
* Уеб сървърът връща отговор на браузъра и му зарежда съответната страница.

****

**Фигура 1-3 Принцип на работа на MVC.**

**Как се Инсталира Laravel ?**

Laravel използва  [[Composer](https://getcomposer.org/), за](https://getcomposer.org/) да управлява зависимостите му. Така че, преди да използвате Laravel, уверете се, че сте инсталирали Composer на вашата машина.

Инсталатор Via Laravel

Първо, изтеглете инсталатора на Laravel с помощта на Composer:

composer global require "laravel/installer"

Уверете се, че сте поставили директорията $PATH на компилатора в системния склад, така че изпълним може да се намира от вашата система. Тази директория съществува на различни места въз основа на вашата операционна система; Някои често срещани места обаче са:

* MacOS: $HOME/.composer/vendor/bin
* Разпределения на GNU / Linux: $HOME/.config/composer/vendor/bin

След като бъде инсталирана, командата ще създаде нова инсталация на Laravel в указаната директория. Например, ще създадете директория с име, съдържащо свежа инсталация на Laravel с вече инсталираните зависимости на Laravel: laravel new laravel new blogblog

laravel new blog

#### Via Composer Създаване на проект

Друга възможност е да инсталирате Laravel като издавате командата Composer във вашия терминал:create-project

composer create-project --prefer-dist laravel/laravel <project name>

Сървър за локално развитие

Ако имате инсталиран PHP на местно ниво и бихте искали да използвате вградения сървър за разработка на PHP, за да обслужвате приложението си, можете да използвате serve командата Artisan. Тази команда ще стартира сървър за разработка на адрес : http://localhost:8000

php artisan serve

**Как се конфигурира Laravel ?**

След като *Laravel*бъде свален, има няколко настройки, които трябва да се направят, за да може да го използваме за даденото приложение. Основните промени ще бъдат върху следните конфигурации:

**Публична директория**

След като инсталирате Laravel, трябва да конфигурирате public директорията на документа / уеб адреса на уеб сървъра си като директория. В тази директория служи като предния контролер за всички HTTP заявки, въвеждащи вашето приложение.index.php

**Файлове за конфигурация**

Всички конфигурационни файлове за рамката Laravel се съхраняват в config директорията. Всяка опция е документирана, така че можете да разгледате файловете и да се запознаете с възможностите, с които разполагате.

**Разрешения за директории**

След като инсталирате Laravel, може да се наложи да конфигурирате някои разрешения. Директорите в storage и директорията трябва да могат да се запишат от уеб сървъра ви или Laravel няма да работи. Ако използвате виртуалната машина [Homestead](https://laravel.com/docs/5.6/homestead) , тези права вече трябва да бъдат зададени. bootstrap/cache

**Ключ за приложението**

Следващото нещо, което трябва да направите след инсталирането на Laravel, е да настроите клавиша за приложение на произволен низ. Ако сте инсталирали Laravel чрез Composer или инсталатора Laravel, този ключ вече е зададен за вас от командата. php artisan key:generate

Обикновено този низ трябва да е с дължина 32 знака. Ключът може да бъде зададен в .env файла за околната среда. Ако не сте преименували файла , трябва да го направите сега. **Ако ключът за приложението не е настроен, вашите потребителски сесии и други криптирани данни няма да бъдат защитени!** .env.example.env

**Допълнителна конфигурация**

**Ларавел няма нужда от почти никаква друга конфигурация извън кутията. Вие сте свободни да започнете да се развивате! Можете обаче да прегледате файла и документацията му. Той съдържа няколко опции и може би желаете да промените според вашата кандидатура.** config/app.phptimezonelocale

**Може също да искате да конфигурирате няколко допълнителни компонента на Laravel, като например:**

* [Cache](https://laravel.com/docs/5.6/cache#configuration)
* [Database](https://laravel.com/docs/5.6/database#configuration)
* [Session](https://laravel.com/docs/5.6/session#configuration)

# **2. Използвани технологии и средства за програмиране**

## 2.1. MySQL

MySQL е най-популярната система за управление на бази данни с отворен код. SQL е съкращение от Structured Query Language (структуриран език за заявки), предназначен за създаване, обработване и четене на релационни бази данни. MySQL поддържа интерфейси за програмиране под множество езици - C/C++, Java, Perl, Python, но най-често се използва в комбинация с езика PHP. Данните, съхранявани в MySQL могат да бъдат от всякакъв вид - от обикновени текстови списъци до сложно структурирана информация. MySQL организира данните в таблици и позволява бърз достъп до тях, поради което широко се използва при изграждането на уеб сайтове, особено такива с по-голям обем информация, изискващи бързо четене и обработка[3]. Той не изисква опитен администратор и лесно се мащабира, ето защо ще бъде използван за създаването на конкретното приложение.

## 2.2. HTML (Hyper Text Markup Language)

Език за форматиране на хипертекстови документи, специално разработен за създаване на Web страници. Той се е превърнал в основен език за тази цел, което се обяснява с неговата простота и сравнително малкото на брой команди, довело до лесното му усвояване и използване. Създаден е специален стандарт дефиниращ набора от команди в HTML, въведен и поддържан от организацията Word Wide Web Consortium или W3C. Превърнал се в стандарт, чрез HTML се създават структурирани статични документи, като се описват елементи като заглавия, параграфи, списъци, таблици, връзки и други. Чрез HTML обаче, не могат да се създават динамични страници, чието съдържание да се променя в зависимост от действията на потребителя. За целта се използват допълнителни технологии, чрез които това става възможно. Потребителският интерфейс на разглежданата система е изграден на основата на HTML, в комбинация с CSS, JavaScript, AJAX и JSON, което позволява всеки екран и информацията предоставена чрез него, да се променя без да е необходимо той да бъде непрекъснато презареждан. Това прави работата със системата удбона и бърза – такава каквато сме свикнали да получаваме от локалните приложения, изпълнявани на персонален компютър.

## 2.3. JavaScript

Интерпретиран език за програмиране, който се използва за добавяне на интерактивност в Web страниците. Скриптът, написан на JavaScript е програма, която се включва в HTML страницата. Тъй като скриптът се огражда с тагове <SCRIPT>, съдържанието му не се появява на екрана, а Web браузърът изпълнява (интерпретира) JavaScript програмата. Най-често тагът <SCRIPT> се разполага в рамките на секция <HEAD>. JavaScript е обектно-ориентиран език, който допуска създаване на активен интерфейс за потребителите, предоставяйки им обратна връзка при придвижване из страниците. JavaScript може да се използва за проверка валидността на въвежданата информация във форми, като по този начин се предпазваме от загуба на време и ресурси, за управление на визуални елементи в уеб-страниците, за допълнително удобство на потребителя, както и за динамична промяна на информацията представена на уеб-страниците.

## 2.3. AJAX

AJAX е съкращение на Asynchronous JavaScript And XML2. Това не е програмен език, а термин описващ начин на употреба на група технологии като JavaScript, dynamic HTML (DHTML), Extensible Markup Language (XML), Cascading Style Sheets (CSS), Document Object Model (DOM). Това е технология за създаване на по-добри, по-бързи и по-динамични уеб страници. Позволява да се разменя информация между сървъра и браузъра без да се презареждат отделните страници. AJAX е технология, която работи с браузъра.

## 2.4. PHP

PHP е сървърен скриптов език, проектиран специално за създаване на Интернет базирани приложения. PHP кодът може да се вгради в HTML страница и ще се изпълнява при всяко нейно зареждане. Кодът се интерпретира от уеб-сървъра и генерира изходни данни, които посетителят на сайта или потребителя на уеб-приложението вижда.

# **3.** **Архитектура на Уеб Приложение**

## 3.1. Въведение

През последните години се наблюдава бързо разрастване на Web базираните информационни системи (ИС), които предоставят възможност на потребителите за търсене на информация, за структурирането й по желание на клиента, както и попълване на данните в тези системи от отдалечени работни станции. Възможността да се въвежда и ползва информация директно през Web браузър поставя Web базираните ИС на първите места на услугите, предлагани в интернет. Актуални задачи на Web базираните ИС [1,2] са:

- Потребителите да могат да използват приложното програмно осигуряване (ППО) от произволен браузър, който поддържа HTML;

- Голям брой потребители да имат възможност да използват ППО едновременно;

- Разполагане на често използваните данни близо до потребителите, с цел намаляване на мрежовия трафик;

- Съхраняване на периодично актуализиращите се данни в отделно физическо пространство от данните предназначени само за селектиране на информация;

- Осигуряване на копия на данните в си-туации на технически аварии;

- Осигуряване на данни за ППО от раз-лични източници – бази от данни (БД), директорийни услуги, електронна поща и др.;

- Ефективността на ППО не трябва да намалява с увеличаване на количество-то данни.

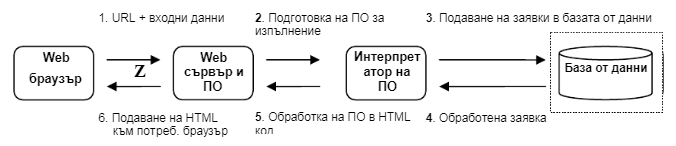
Задължителен елемент от една такава система е хранилището от данни и в частност базата от данни, в които се съсредоточава цялата информация и се осигурява достъп до исканата информация. Целта на настоящия доклад е да разгледа най-често използваните архитектури на връзката на Web приложения с база от данни и предложи разширена архитектура на връзката Web приложение – разпределено хранилище от данни.

2. АРХИТЕКТУРИ НА ВРЪЗКАТА WEB

ПРИЛОЖЕНИЕ – БАЗА ОТ ДАННИ

## 3.2. Архитектури на връзката Web приложение – База от данни

Архитектурата на връзката на Web приложение с база от данни се състои от потребителски Web браузър, Web сървър, интерпретатор на програмното осигуряване (ПО) и база от данни. Манипулирането с данните се осъществява с използване на скриптове, които се стартират на Web сървъра или на потребителския браузър. Популярни архитектури на връзката Web приложение – база от данни са архитектура, при която браузъра е посредник за приемане и предаване на заявки и резултати и архитектура, при която Web браузъра подготвя заявка към БД [3]. На фиг. 1 е показана блок схема на най-често използваната архитектура на връзката на Web приложение с база от данни, при която Web браузъра е посредник на приемане и предаване на заявки и резултати.



Фигура 1. Блок-схема на архитектура на връзка, при която браузъра е посредник

за приемане и предаване на заявки и резултати

**Фигура 3-1. Блок-схема на архитектура на връзка, при която браузъра е посредник за приемане и предаване на заявки и резултати**

Една транзакция към базата от данни тук се състои от следните стъпки:

- Потребителски Web браузър изпраща HTTP заявка за определена Web страница;

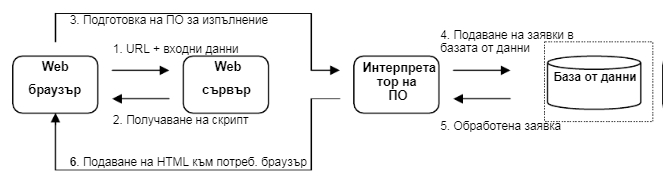
- Web сървъра получава заявката от клиента. Извлича точно този файл отговарящ на изискването и го подава на интерпретатора на ПО за обработка и изпълнение;

- Интерпретаторът обработва ПО. Ако в ПО се съдържат операции по осъщес-твяване на връзка с БД и търсене и обработка на данните в нея интерпретаторът осъществява връзка с БД и изпраща поотделно заявките;

- Сървърът с БД получава заявките, обработва ги и изпраща резултатите обратно на интерпретатора на ПО;

- Интерпретаторът на ПО завършва изпълнението на ПО, което обикновено включва форматиране на резултатите в код удобен за интерпретиране от потребителски Web браузър, след което връща получения код на Web сървъра;

- Сървърът изпраща кода към браузъра, при което потребителят получава Web страница.

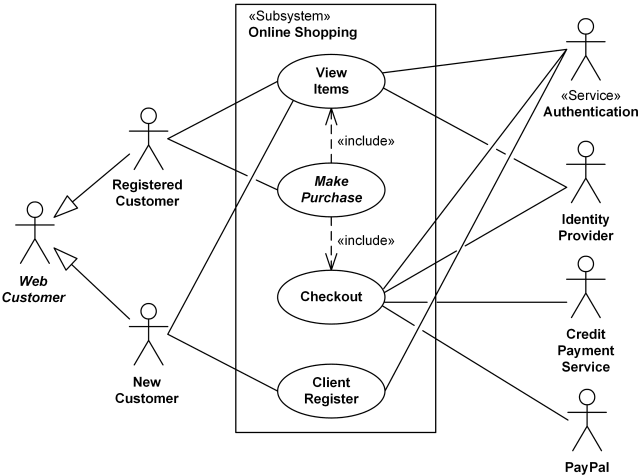


**Фигура 3-2. Блок-схема на архитектура на връзка, при която браузъра подготвя заявка към БД**

## 

## 3.3. Use Case диаграми

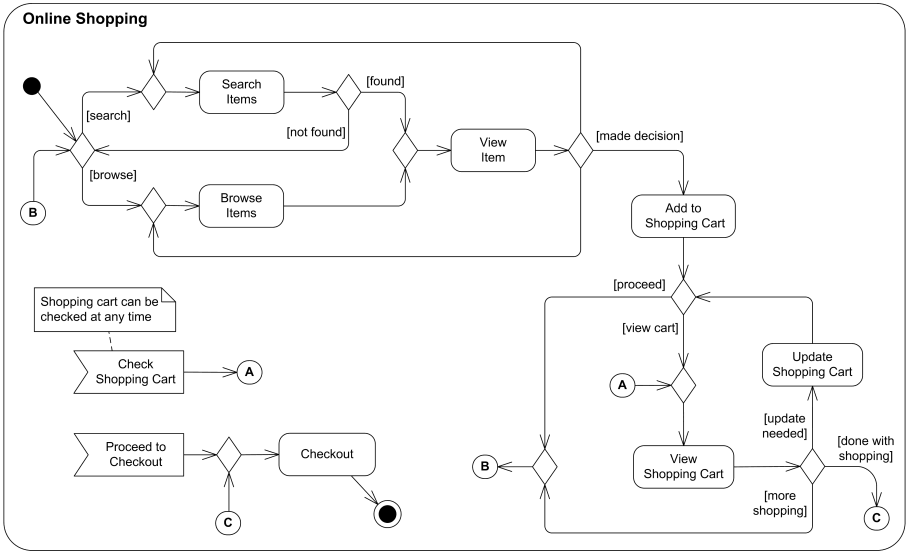
### 3.3.1. Уеб Клиент актьор

Използва някои уеб сайт, за да се правят покупки онлайн. Най- често използваните случаи са View Items , Make Purchase и Client Register . Прегледът на случаите на използване на елементи може да се използва от клиента като случай на най-високо ниво на използване, ако клиентът иска да намери и види някои продукти. Този случай на използване също може да бъде използван като част от случая за използване на покупка. Регистрационният случай на клиентски регистър позволява на клиента да се регистрира на уеб сайта, например да получи талони или да бъде поканен за частни продажби. Имайте предвид, че случаят за използване на Checkout е включен в случая на употреба, който не е налице сам - плащането е част от извършването на покупка. ****

**Фигура 3-3 Онлайн пазаруване Уеб Клиент актьор**

### 3.3.2. Схема на действие

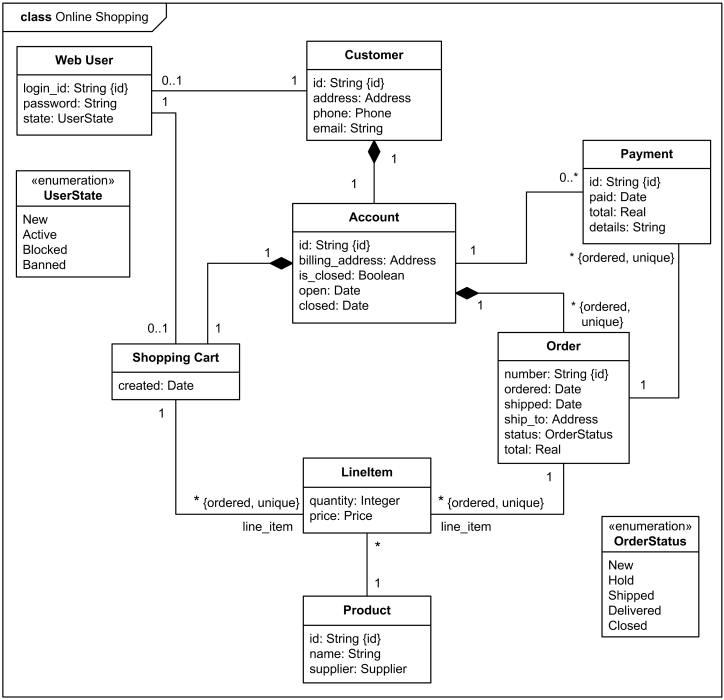
Пример за **схема на действие** за **онлайн пазаруване** . Онлайн клиентът може да разглежда или да търси елементи, да преглежда конкретен артикул, да го добавя в пазарската кошница, да преглежда и актуализира пазарската кошница, касов апарат. Потребителят може да разглежда кошницата по всяко време. Очаква се Checkout да включва регистрация и влизане в системата.



**Фигура 3-4 Онлайн пазаруване Схема на действие**

### 3.3.3. Диаграма на UML класа

Всеки клиент има уникален идентификационен номер и е свързан с точно един **профил** . Сметката притежава пазарска кошница и поръчки. Клиентът може да се регистрира като уеб потребител, за да може да купува артикули онлайн. От клиента не се изисква да бъде уеб-потребител, защото покупките могат да се правят и по телефона или чрез поръчка от каталози. Уеб-потребител има емаил, което също служи като уникален идентификатор.



**Фигура 3-4 Онлайн пазаруване Схема на класа**

## 3.4. Проектиране на базата данни

Преди да започнем да говорим за база данни е добре да знаем какво представлява тя? В най-простия вариант , база данните е колекция от логически свързани данни в конкретна предметна област, които са структурирани по определен начин. В първоначалния смисъл на понятието, използван в компютърната индустрия, база данните се състои от [записи](http://bg.wikipedia.org/w/index.php?title=%D0%97%D0%B0%D0%BF%D0%B8%D1%81_(%D0%B1%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D0%B8)&action=edit&redlink=1), подредени систематично, така че [компютърна програма](http://bg.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8E%D1%82%D1%8A%D1%80%D0%BD%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0) да може да извлича информация по зададени критерии. Поддръжката на база данните се осъществява от т.нар. [система за управление на бази от данни](http://bg.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%B7%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%BD%D0%B0_%D0%B1%D0%B0%D0%B7%D0%B8_%D0%BE%D1%82_%D0%B4%D0%B0%D0%BD%D0%BD%D0%B8) (DBMS).[1]

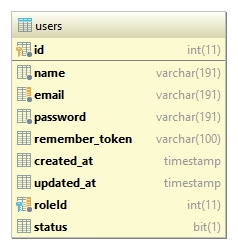
През 1970 от Едгар Код въвежда понятието релационни бази данни. Най-общо казано релационните бази от данни групират информацията по характерни признаци. В MySQL релациите (отношенията) на данните се описват с доста близки до човешкото разбиране структури – таблици. Всяка таблица се състои от редове и колони. Всяка колона има наименование, което определя данните въведени в нея по редове, както и тип на тези данни. Обикновено имената и типа на колоните се наричат атрибути. В релационните бази данни се избира първичен ключ (primary key), който е основният инструмент за осъществяването на връзка между таблиците. Той трябва да е уникален и може да се състой от повече от един атрибут. Данните в таблиците се записват с определен тип. Често не е възможно да се запише дадена информация. За целта се използва специална стойност на полето – NULL. Едно от основните ограничения в релационните бази от данни е, че **първичните ключове не могат да приемат стойност NULL**! За осъществяване на връзката се използва и външен ключ. Външен ключ се нарича атрибут на дадена таблица, който съответства на първичен ключ в друга. Може да се приеме, че външният ключ е указател към първичен ключ. Ограничението за външен ключ е, че той може да приема или стойност NULL, или стойност равна на стойността в съответстващия му първичен ключ. Всъщност осъществяването на връзки между външен и първичен ключ не е задължително. Можем въобще да не използваме външни ключове, а да ги разгледаме като обикновени колони на таблица. Така обаче ще се лишим от няколко удобства, като интегритет на данните, използване на индекси и т.н. [2] Има няколко вида връзки между таблиците, според това даден запис, към колко записа от втората таблица се отнася:

* 1: 1 – тази връзка представлява отношение на един запис от дадена таблица към един запис от друга таблица;
* 1: M – тази връзка представлява отношение на един запис от дадена таблица към един или няколко записа от друга таблица;
* N: M – тази връзка представлява отношение на няколко записа от една таблица към няколко записа от друга таблица, по същество се организира като две връзки 1: М.

Съвкупността от всичките използвани таблици, полетата им и връзките между тях се използват за създаване на модел на база данните.

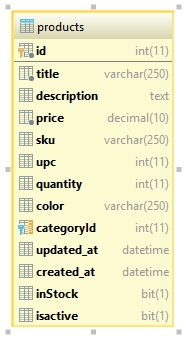
### 4.1.1. Използвани таблици

**Users -** Тази таблица съхранява информация за потребителя след успешната регистрация и допълнителна информация, която се добавя от потребителския профил.



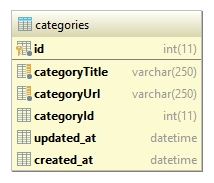
**Фигура 2-1 Таблица Users**

**Products -** Съдържа цялата налична информация за предлаганите продукти от приложението.



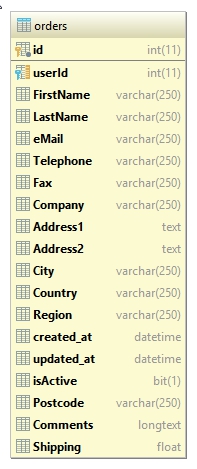
**Фигура 2-2 Таблица Products**

**Categories -** Съдържа всички възможни категории на продуктите, които се предлагат от приложението.



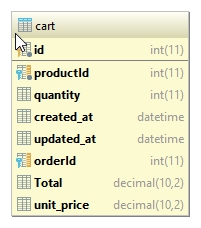
**Фигура 2-3 Таблица Categories**

**Оrders -** Съдържа информация за активните поръчки.

****

**Фигура 2-4 Таблица Orders**

**Cart -** Съдържа информация за клиентите, които са направили поръчка.

****

**Фигура 2-11 Таблица Cart**

Тук ще бъде обяснено как работят най-важните функции от контролерите и моделите, тъй като те съдържат основната част от логиката на сайта.

# **4. Разработка на приложението**

## 4.2. Контролери

**IndexController**

**class** IndexController **extends** Controller  
{  
 **public function** index ()  
 {  
 $products = Products::*all*();  
 $coupons = Coupons::*where*(**'status'**, **true**)->get();  
 $sale = Sales::*sales*($products,$coupons);  
  
 **return** view(**'index'**)->with(**'sliders'**, $sliders)  
 ->with(**'productsAll'**, $sale);  
 }  
  
}

**index** - Тази функция се използва за зареждане на съответното view (**'index'**) т.е начална страница и зарежда всички активни продукти проверява да ли има намаление ако има показва намалената стойност и старата стойност.

**CategoryController**

**class** CategoryController **extends** Controller  
{  
 **public function** categoryGet($title = **null**){  
 **if**($title != **null**) {  
 $category = Categories::*where*(**'categoryUrl'**, $title)->get();  
 $id = $category[0]->**id**;  
 $products = Products::*where*(**'categoryId'**,$id)->get();  
 $coupon = Coupons::*where*(**'status'**,**true**)->get();  
 $sales = Sales::*sales*($products,$coupon);  
 }  
 **else** {  
 $products = Products::*all*();  
 $coupon = Coupons::*where*(**'status'**,**true**)->get();  
 $sales = Sales::*sales*($products,$coupon);  
 }  
 **return** view(**'category.category'**)->with(**'products'**,$sales);  
 }  
}

**categoryGet** - Тази функция се използва за показване на продуктите според зададените категории, ако няма такива показва всички налични продукти. Функцията взима параметър от URL и ако има стойност прави заявка на базата с този параметър който идва от URL сортира всички продукти от тази категория и зарежда съответното view(**'category.category'**) и показва, ако няма стойност връща всички продукти.

**MiniCartControlle**

**class** MiniCartController **extends** Controller  
{  
 **public static function** miniCart ()  
 {  
 $cart = **new** CartController();  
 $session = Session::**get**(**'cart'**);  
 $mini\_cart = **null**;  
 **if** ($session != **null**) {  
 $products = Products::*whereIn*(**'id'**, $session)->get();  
 $coupon = Coupons::*where*(**'status'**,**true**)->get();  
 $sale = Sales::*sales*($products,$coupon);  
 **if**($sale !=**null**){  
 **foreach** ($sale **as** $item) {  
 $item->**price** = $item->**price** - $item->**sale**;  
 }  
 }  
 $mini\_cart = $cart->add($session, $sale);  
 }  
 $products\_all = **null**;  
 $total\_price = **null**;  
 **if** ($mini\_cart != **null**) {  
 **foreach** ($mini\_cart **as** $item) {  
 $products\_all += $item[**'quantity'**];  
 $total\_price += $item[**'item'**]->**price** \* $item[**'quantity'**];  
 }  
 }  
 $shipping = Shipping::*first*();  
 **return** [  
 **'mini\_cart'**=>$mini\_cart,  
 **'products\_all'**=> $products\_all,  
 **'total\_price'**=> $total\_price,  
 **'shipping'**=>$shipping  
 ];  
 }  
}

**miniCart** - Тази функция се използва за показване на продукти в малка количка, където се намира горе вдясно. Функцията работи със сесии (***Session***), клиента избира продукта който иска да поръчва и като се добави в количката продукт се запазва на сесия (***Session***) за да не прави всеки път заявка на базата клиента когато иска та прави поръчка. Тогава всички продукти които иска да поръчва се записва на базата с една заявка. Защо правим така, за да не претоварваме сървъра и така приложението работи по бързо.

**ProductController**

**class** ProductController **extends** Controller  
{  
 **public function** productByIdGet ($id)  
 {  
 $products = Products::*where*(**'id'**, $id)->get();  
 $coupon = Coupons::*where*(**'status'**,**true**)->get();  
 $sale = Sales::*sales*($products,$coupon);  
 **return** view(**'product.product'**)->with(**'products'**, $sale);  
 }  
}

**productByIdGet** - Тази функция се използва, когато бъде избрана даден продукт с цел преглеждане на цялото й съдържание.Функцията взима параметът ID от URL този параметът идва от избрана даден продукт по този начин правим заявка на базата и ни връща резултата който искаме да видим.

**CartController**

**class** CartController **extends** Controller  
{  
 **public function** cart ()  
 {  
 $session = Session::*get*(**'cart'**);  
 $shipping = Shipping::*first*();  
 **if** ($session != **null**) {  
 $products = Products::*whereIn*(**'id'**, $session)->get();  
 $coupon = Coupons::*where*(**'status'**,**true**)->get();  
 $sale = Sales::*sales*($products,$coupon);  
 $carts = $this->add($session, $sale);  
 }  
 **else** {  
 **return** view(**'cart.cart'**)->with(**'carts'**, **null**);  
 }  
 **return** view(**'cart.cart'**)->with(**'carts'**, $carts)->with(**'shipping'**,$shipping);  
 }

**Cart** - Тази функция се използва за зареждане на съответното view(**'cart.cart'**) и показва продукти които са добавени в количката. Функцията проверява сесията (Session) който запазва поръчките ако има продукти в сесията зарежда съответното view (**'cart.cart'**) и продукти който са в количката ако няма продукти в сесията тогава редиректва на началната страница.

**public function** cartRemoveGet ($id)  
 {  
 $session = Session::*get*(**'cart'**);  
 Session::*pull*(**'cart'**, $id);  
 **if** ($session != **null**) {  
 **foreach** ($session **as** $item) {  
 **if** ($item != $id) {  
 Session::*push*(**'cart'**, $item);  
 }  
 }  
 } **else** {  
 **echo 'Items Not Found'**;  
 }  
 }

**cartRemoveGet -** Тази функция се използва за премахване на даден продукт от количката за пазаруване. Функцията взима параметър ID и проверява сесията ако има продукт с този параметър премахва от сесията.

**public function** cartUpdateGet($id,$quantity)  
 {  
 $session = Session::**get**(**'cart'**);  
 Session::*pull*(**'cart'**,$id);  
 **if** ($session != **null**) {  
 **foreach** ($session **as** $item) {  
 **if** ($item != $id) {  
 Session::*push*(**'cart'**, $item);  
 }  
 }  
 **if**($quantity != **null**) {  
 **for** ($i = 0; $i < $quantity; $i++) {  
 Session::*push*(**'cart'**, $id);  
 }  
 }  
 } **else** {  
 **echo 'Items Not Found'**;  
 }  
 }

**cartUpdateGet** - Тази функция се използва за обновяване на количеството на даден продукт от количката за пазаруване. Функцията взима 2 параметъра ID на продукта и quantity количеството на този продукт първо проверява сесията намира продукта и премахва после с **for** вмъква количеството пити **id** на продукта и така се променя количеството на продукта.

**public function** cartAddGet ($id)  
{  
 Session::*push*(**'cart'**, $id);  
}

**cartAddGet** - Тази функция се използва за добавяне на даден продукт в количката за пазаруване. Функцията взима 1 параметър ID на продукта и добавя в сесията.

**CheckoutController**

**public function** checkoutPost (Request $request)  
{  
 $request->validate([  
 **'FirstName'** => **'required'**,  
 **'LastName'** => **'required'**,  
 **'Telephone'** => **'required'**,  
 **'Address1'** => **'required'**,  
 **'Email'** => **'required'**,  
 **'City'** => **'required'**,  
 **'Postcode'** => **'required'**,  
 ]);

$shipping = Shipping::*first*();  
$session = Session::**get**(**'cart'**);  
$products = Products::*whereIn*(**'id'**, $session)->get();  
$coupon = Coupons::*where*(**'status'**, **true**)->get();  
$sale = Sales::*sales*($products, $coupon);  
**foreach** ($sale **as** $product) {  
 $product->**price** = $product->**price** - $product->**sale**;  
}

$carts = $this->add($session, $sale);  
 $cart = $this->cartSet($carts);  
 $orders = **new** Orders();  
 $orders->**FirstName** = $request->get(**'FirstName'**);  
 $orders->**LastName** = $request->get(**'LastName'**);  
 $orders->**Telephone** = $request->get(**'Telephone'**);  
 $orders->**Address1** = $request->get(**'Address1'**);  
 $orders->**Address2** = $request->get(**'Address2'**);  
 $orders->**Email** = $request->get(**'Email'**);  
 $orders->**Postcode** = $request->get(**'Postcode'**);  
 $orders->**Fax** = $request->get(**'Fax'**);  
 $orders->**Company** = $request->get(**'Company'**);  
 $orders->**City** = $request->get(**'City'**);  
 $orders->**Country** = $request->get(**'Country'**);  
 $orders->**Region** = $request->get(**'Region'**);  
 $user\_id = $request->get(**'UserId'**);  
 **if**($user\_id != **"0"**)  
 $orders->**userId** = $request->get(**'UserId'**);  
 **else** $orders->**userId** = 999;  
 $orders->**isActive** = **true**;  
 $orders->**Comments** = $request->get(**'Comments'**);  
 $orders->**Shipping** = $shipping->**price**;  
 $orders->save();  
 $orders->cart()->saveMany($cart);  
 Session::*pull*(**'cart'**);  
 Mail::*to*($request->get(**'Email'**))  
 ->send(**new** OrderShippedMail($orders,$cart));  
}

**checkoutPost -** Тази функция се използва за създаване на записи в таблиците `**orders**` и `**cart**` при успешно направена поръчка. Функцията взима обект Request този обект се използва в Laravel за n брой параметри който се праща от метода post и с $request->validate Валидираме параметрите, примерно името на клиента задължително и телофона и т.н. ако клиента е пропуснал да си напише името си validate метода връща съобщение „Името задължително поле“ ако всичко е точно записва се информацията на клиента в таблица Order а продукта на клиента в таблица Cart, между тях има връзка едно към много, когато приключи всичко сесията се изтрива.След това Laravel има функция **Mail::to** за изпращане на писмо към електронната поща на клиента с направени поръчки, количество, единична цена, доставката и общата сума.

## 4.3. Модели

**Products**

**class** Products **extends** Model  
{  
 **protected $table** = **'Products'**;  
  
 **public function** categories(){  
  
 **return** $this->hasOne(**'App\Models\Categories'**,**'id'**,**'categoryId'**);  
 }  
  
 **public function** images(){  
  
 **return** $this->hasMany(**'App\Models\Images'**,**'productId'**,**'id'**);  
 }  
}

**protected $table** = **'Products'**; - Имена на таблици

Можете да зададете персонализирана таблица, като дефинирате **$table** собственост върху вашия модел. Ако не се дефинира **$table** Laravel автоматично ще вземе името на класа.

‚**categories**‘ - Тази функция се използва за релации между таблици.

**return** $this->hasOne(**'App\Models\Categories'**,**'id'**,**'categoryId'**); - Едно към едно

Отношението "един към един" е много основна връзка. Например **'Products'** може да бъде свързан с една **'Categories'**. За да дефинираме тази връзка, поставяме ***categories*** върху **Products** модела като метод . В ***categories*** метода трябва да се обадите на hasOne метода и да се върне своя резултат:

**return** $this->hasMany(**'App\Models\Images'**,**'productId'**,**'id'**); - Един до много

Отношението "едно към много" се използва за дефиниране на взаимоотношения, при които един модел притежава всякакви други модели. Например публикацията в **Products** може да има безкраен брой **Images**. Подобно на всички други отношенията "един към много" се дефинират чрез поставяне на функция върху вашия модел.

**Categories**

**class** Categories **extends** Model  
{  
 **protected $table** = **'Categories'**;  
  
 **public function** products(){  
  
 **return** $this->hasMany(**'app/models/product'**);  
 }  
  
 **public function** subCategory(){  
  
 **return** $this->hasMany(**'App\Models\Categories'**,**'categoryId'**);  
 }  
}

**return** $this->hasMany(**'app/models/product'**); - Един към много

Отношението "едно към много" се използва за дефиниране на взаимоотношения, при които един модел притежава всякакви други модели. Например в **Categories** може да има безкраен брой **Products**." Един към много" в ***Laravel*** се дефинират чрез поставяне на функция върху вашия модел.

**Cart**

**class** Cart **extends** Model  
{  
 **protected $table** = **'cart'**;  
   
 **public function** products(){  
   
 **return** $this->hasMany(**'App\Models\Products'**,**'id'**,**'productId'**);  
 }  
}

**return** $this->hasMany(**'App\Models\Products'**,**'id'**,**'productId'**); - Един към много

Отношението "едно към много" се използва за дефиниране на взаимоотношения, при които един модел притежава всякакви други модели. Например в **Cart** може да има безкраен брой **Products**." Един към много" в ***Laravel*** се дефинират чрез поставяне на функция върху вашия модел.

**Orders**

**class** Orders **extends** Model  
{  
 **protected $table** = **'orders'**;  
  
 **public function** cart(){  
  
 **return** $this->hasMany(**'App\Models\Cart'**,**'orderId'**,**'id'**);  
 }  
}

**return** $this->hasMany(**'App\Models\Cart'**,**'orderId'**,**'id'**); - Един към много

Отношението "едно към много" се използва за дефиниране на взаимоотношения, при които един модел притежава всякакви други модели. Например в **Orders** може да има безкраен брой **Cart**." Един към много" в ***Laravel*** се дефинират чрез поставяне на функция върху вашия модел.

**Coupons**

**class** Coupons **extends** Model  
{  
 **protected $table** = **'coupons'**;  
  
 **public function** category(){  
 **return** $this->belongsToMany(**'App\Models\Categories'**,**'coupon\_categories'**,  
 **'coupon\_id'**,**'category\_id'**);  
 }  
  
 **public function** product(){  
 **return** $this->belongsToMany(**'App\Models\Products'**,**'coupon\_products'**,  
 **'coupon\_id'**,**'product\_id'**);  
 }  
}

**return** $this->belongsToMany(**'App\Models\Products'**,**'coupon\_products'**,  
 **'coupon\_id'**,**'product\_id'**); - Много до много

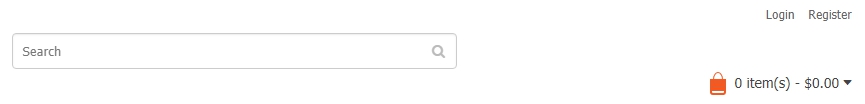
Много към много отношения са малко по-сложни, отколкото hasOne и hasMany взаимоотношения. Например, много **Products** могат да имат **Coupon** за "Намаление". За да определите тази връзка, са нужни три таблиците в базата данни: **Products**, **Coupons**, и **Products**\_ **Coupons**.

# **5. Ръководство на потребителя**

## 5.1. Управление на потребителски интерфейс

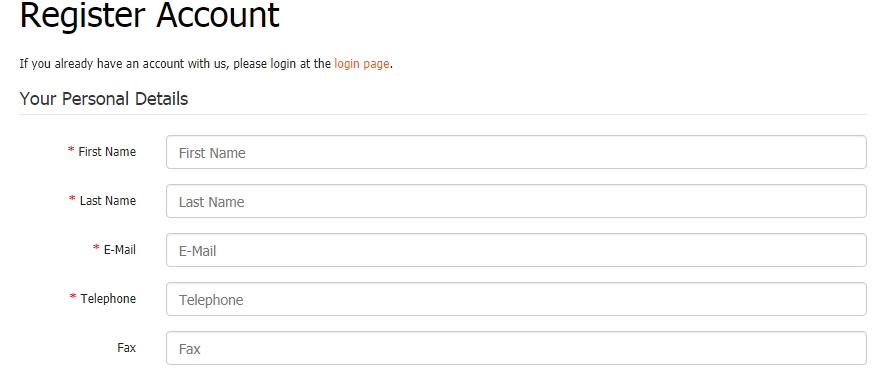
### 5.1.1 Регистрация на потребител

Регистрацията на нов потребител е изцяло без намеса на администратора. Над навигацията на сайта има опция за вход на потребител или регистрация на нов потребител.



**Фигура 5-1 Навигация** **за регистрация на нов потребител.**

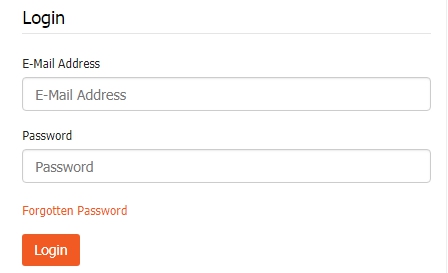
Ако изберем регистрация се зарежда формата, в която да бъде попълнена основната информация, необходима за регистрация на нов потребител, а именно:

* Собствено име
* Фамилно име
* Електронна поща
* Парола

**Фигура 5-2 Форма за регистрация на нов потребител.**

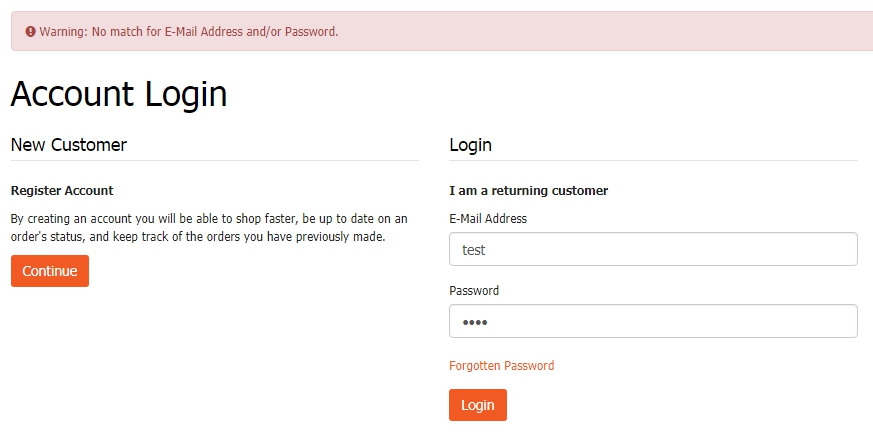
### 5.1.2. Вход на потребител в системата

След като успешно бъде регистриран потребителя, той може да влезне в системата, което му дава определени привилегии. В системата има два вида потребители: администратор и стандартен потребител. Всички те ползват една и съща форма за вход.



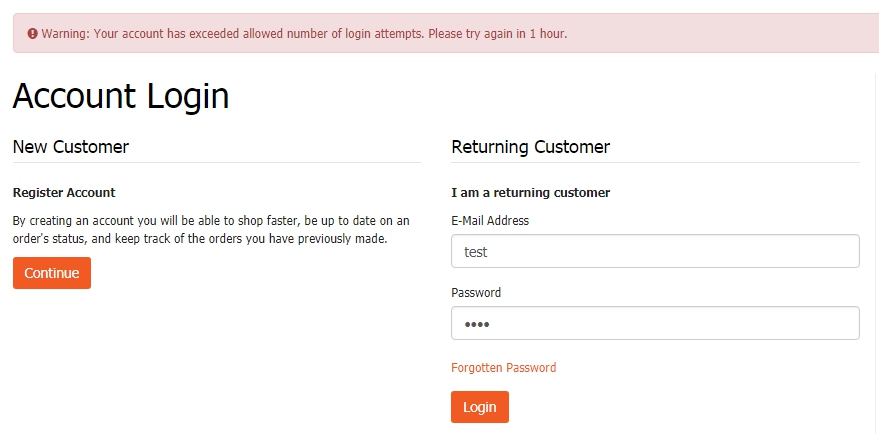
**Фигура 5-3 Формуляр за вход в системата.**

Ако данните въведени в двете полета не отговарят на нито един потребител от съдържанието на таблицата ***User*** ще бъде върнато съобщение към потребителя.



**Фигура 5-4 Невалидно потребителско име и/или парола.**

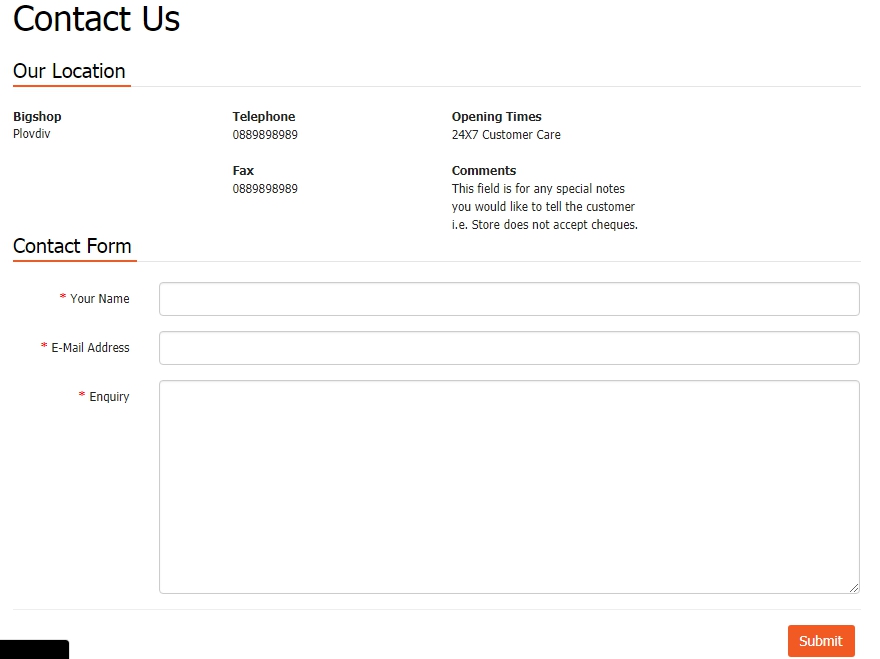
Ако се направят три последователни неуспешни опита за вход , потребителят се заключва. Заключеният потребител може да бъде възстановен след 60 минути автоматично.



**Фигура 5-5 Заключен потребител за 1 час.**

### 5.1.3. Обратна връзка с клиент

Всеки сайт трябва да взима мнението на потребителите си под внимание. За да може да бъде осъществена връзка между администратора и потребителя е създадена форма за контакти. На нея може да се оставят мнения, съвети и оплаквания, които биха помогнали за разрастването на сайта.



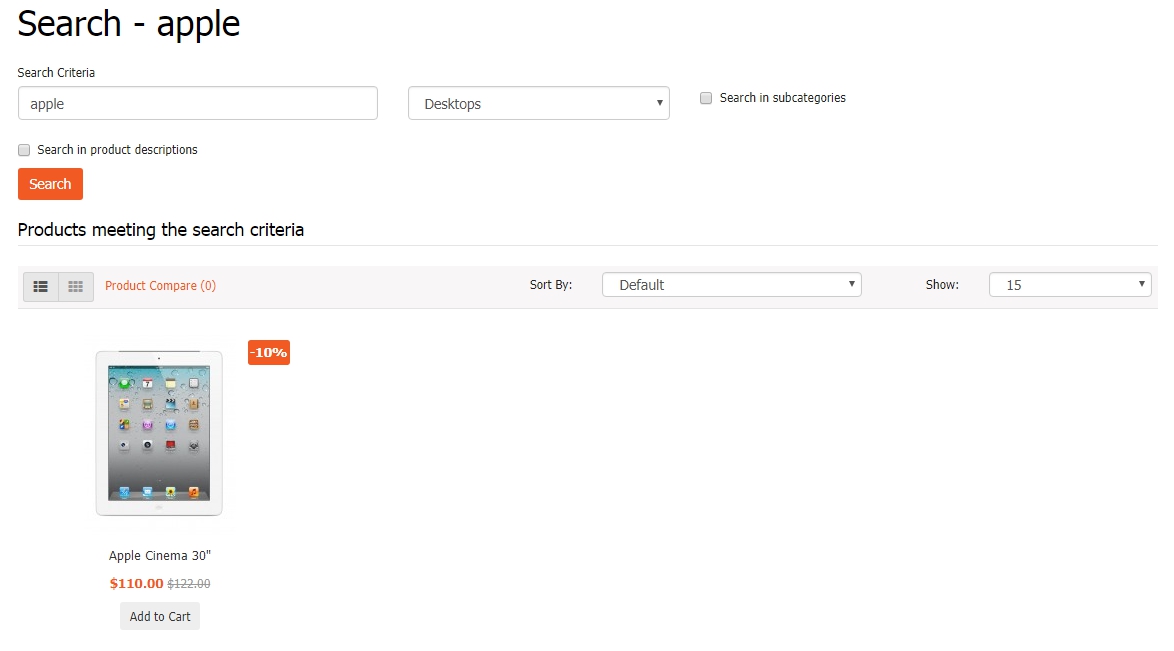
**Фигура 5-6 Изпращане на писмо през формата за контакт с администратора.**

### 5.1.4. Търсене на продукти по различни критерии

Търсене става чрез Search Bar



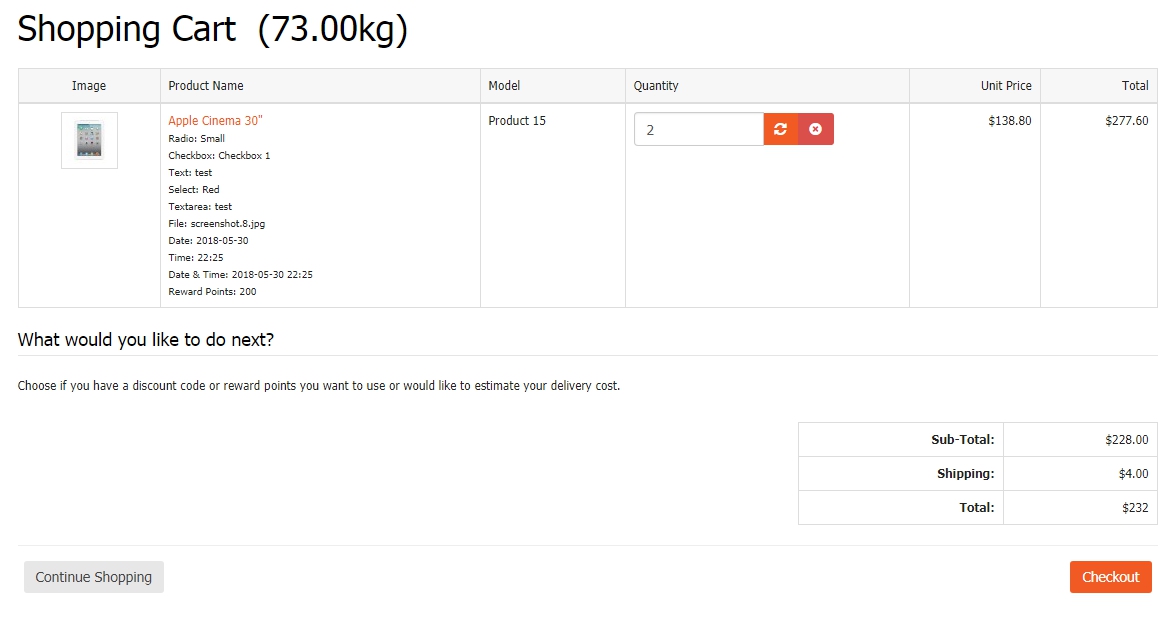
Ако бъде избрана опцията ***Search*** от навигацията ще може да се разгледат всички налични продукти с подробно описание за всеки един. Дава се шанс да се извърши търсене на продукти по различни критерии или да се направи сортиране на продуктите по име, номер на продукта, цена или категория.



**Фигура 5-7 Търсене и сортиране на продукти по определен критерии.**

### 5.1.5. Поръчване на даден продукт

В крайна сметка основната цел на този сайт е реализиране на продажби, за целта в опция ***Cart*** от навигацията се дава възможност да се добавят избраните продукти към количка за пазаруване и от там да се направи поръчка или да се съхранят продуктите в профила на потребителя, с цел списъкът с продукти да бъде преразгледан по-късно. Ако потребителят не е влезнал в системата опциите за поръчка и съхраняване на съдържанието няма да бъдат видими.



**Фигура 5-8 Добавяне на артикули към количката за пазаруване.**

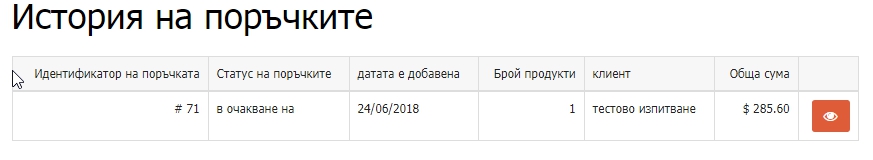
Ако бъде преценено да се направи поръчка ще се отвори форма за попълване на нужната потребителска информация. По подразбиране формата ще бъде попълнена с личните данни, които са посочени в профила на съответния потребител, но ще се даде право за промяна.



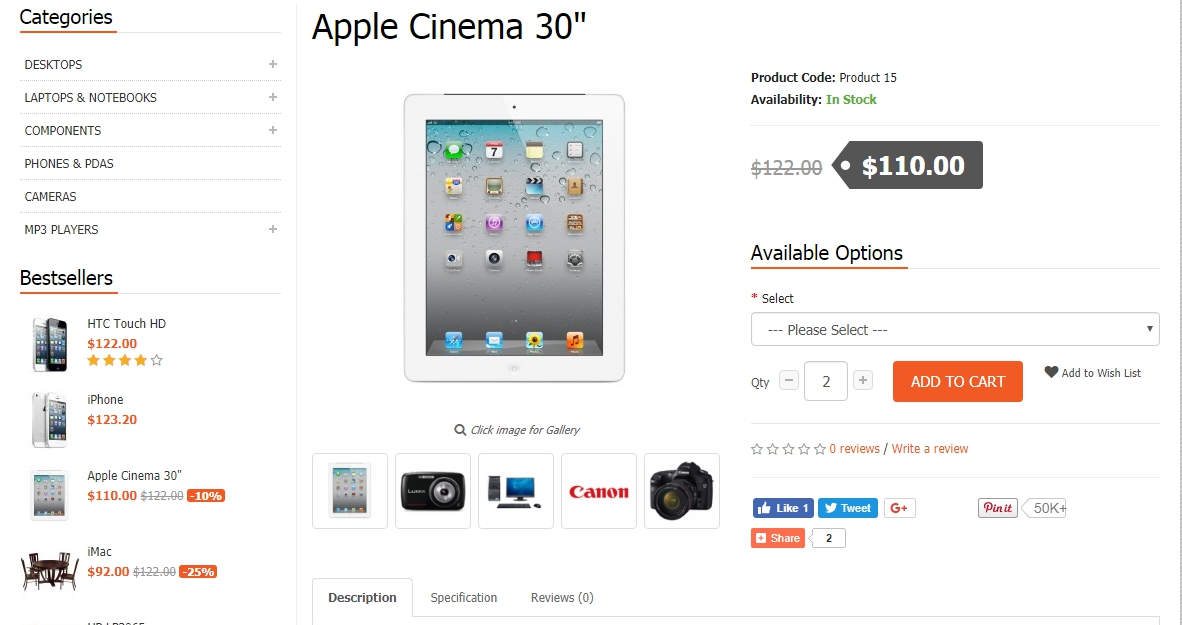
**Фигура 5-9 Форма за поръчка.**

След извършване на операцията съдържанието на количката ще бъде освободено и информацията ще се пази единствено в профила.

**Фигура 5-10 История на поръчките.**



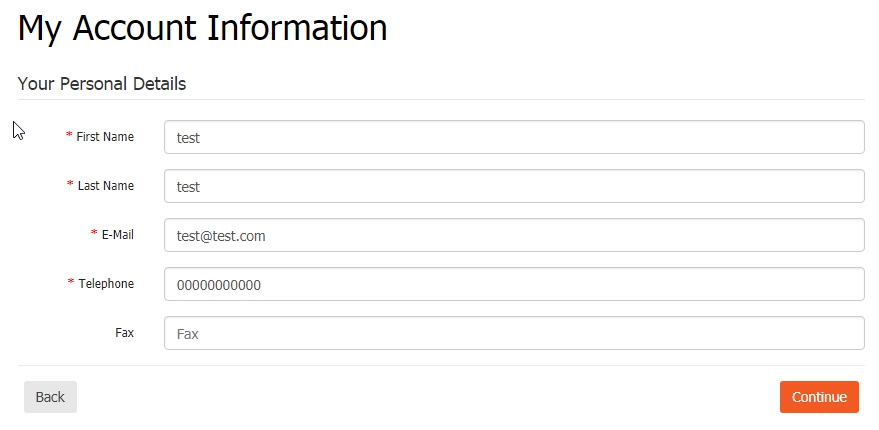
Ако потребителят иска да получи допълнителна информация за даден продукт, преди да го добави в количката си, той трябва да избере конкретният продукт и ще му бъде отворена страница с подробно описание на продукта.



**Фигура 5-11 Допълнителна информация за даден продукт от избраната категория.**

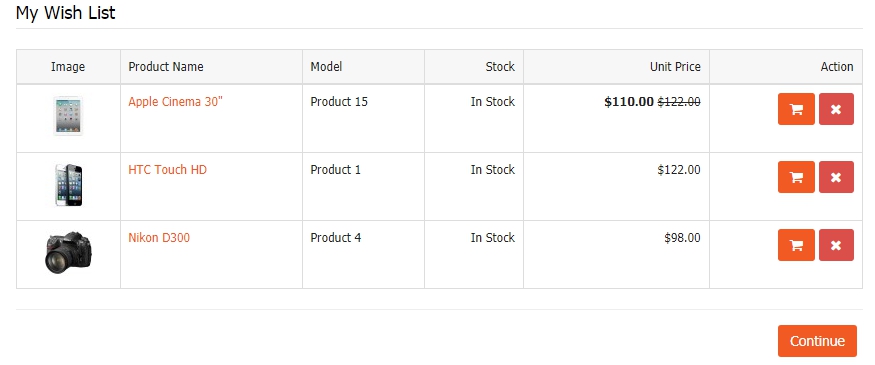
### 5.1.6. Промяна на потребителския профил

Всеки потребител има право да променя съдържанието на профила си по всяко време. Тъй като при регистрацията не се дава информация за телефон и адрес, именно тук е мястото за това, освен това потребителят може да зададе своята потребителска снимка. За да се стигне до съдържанието на профила трябва да се избере опцията „потребителски профил“, в горния десен ъгъл на екрана след като даденият потребител влезне в системата.



**Фигура 5-12 Потребителски профил преди първата настройка.**

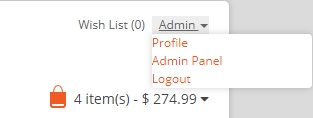
Може да се направи поръчка на продуктите, които са съхранени в профила.



**Фигура 5-13 Поръчка от профила на потребителя.**

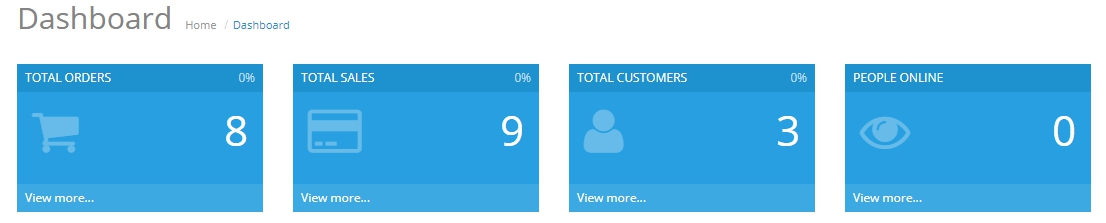
## 5.2. Администраторски панел

След като влезне в системата, за разлика от обикновения потребител, в горния десен ъгъл администраторът има две допълнителни опции в менюто.



**Фигура 5-14 Администратор.**

В опцията „Администраторски панел“, администраторът има възможността да проследи броя на регистрираните потребители, броя на активните сесии, тяхната продължителност.

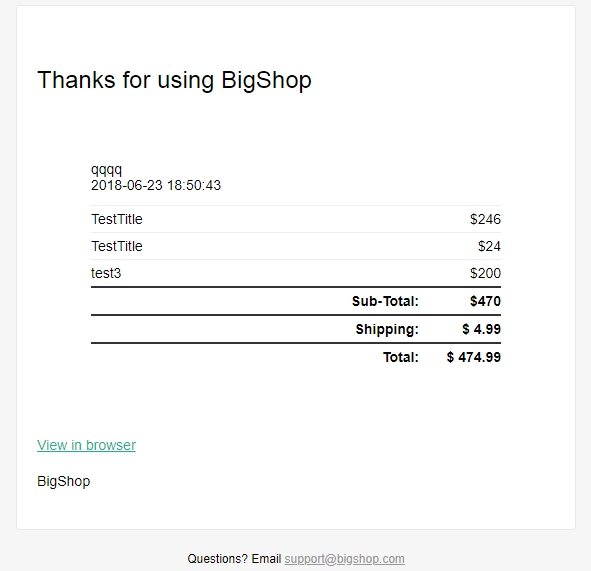


**Фигура 5-15 Списък на регистрирани потребители.**

Администраторът има още една контролна функция – да следи списъка с наличните поръчки и да отбелязва като завършени изпълнените. За тази цел избира втората опция от менюто – „Списък на поръчките“.



**Фигура 5-16 Списък на поръчките.**

След като поръчката бъде потвърдена като изпълнена се изпраща уведомително писмо на съответната електронна поща.

**Фигура 5-17. Уведомително писмо.**

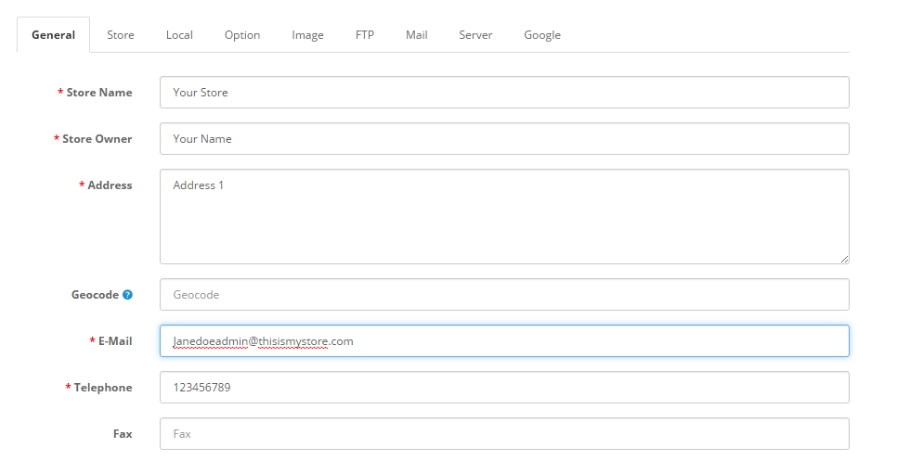
### 5.2.1 Основни настройки на магазина

Този раздел преглежда основните настройки, които трябва да конфигурирате за магазина, за да започнете.



#### 5.2.1.1. Общи настройки

Общият раздел в "Настройки" показва следните полета:

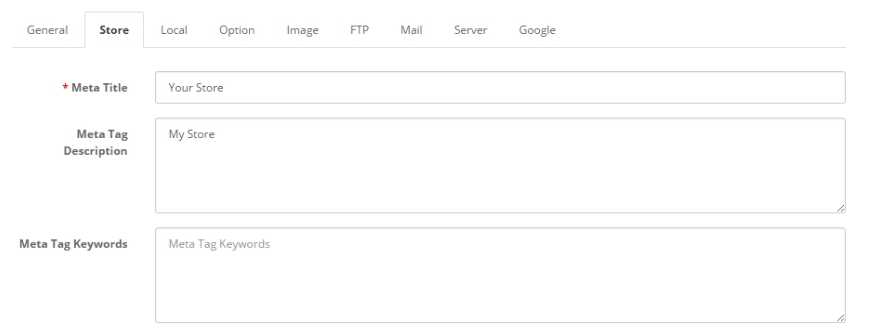
* **URL** адрес на магазина **:** URL адресът на магазина. Не забравяйте да включите "http:" в началото и "/" в края.
* **Име на магазина** : Името на магазина. Тази информация ще бъде показана на страницата "Свържете се с нас" в секцията "Обслужване на клиенти" на долния колонтитул.
* **Собственик на магазина** : Името, използвано в секцията "От:" на всеки имейл, изпратен от магазина.
* **Адрес** : Адресът на магазина. Тази информация ще бъде показана на страницата "Свържете се с нас" в секцията "Обслужване на клиенти" на долния колонтитул.
* **Електронна поща** : **Имейл адресът** на собственика на магазина или общият имейл адрес за магазина, който да се използва за имейл известия, изпратени от OpenCart.
* **Телефон** : телефонен номер за магазина. Тази информация ще бъде показана на страницата "Свържете се с нас" в секцията "Обслужване на клиенти" на долния колонтитул.
* **Факс** (по избор): Факс номер за магазина. Тази информация ще бъде показана на страницата "Свържете се с нас" в секцията "Обслужване на клиенти" на долния колонтитул.

**Фигура 5-18. Настройки на сайта.**

#### 5.2.1.2. Настройки на магазина

Оттук ще зададете основните настройки за магазина си, включително:

* **Заглавие** : Името на магазина да се показва в лентата с инструменти на заглавието на браузъра.
* **Мета маркер Описание** : Етикет, предоставящ описание на метаданните за магазина. Мета маркерите се виждат от търсещите машини.

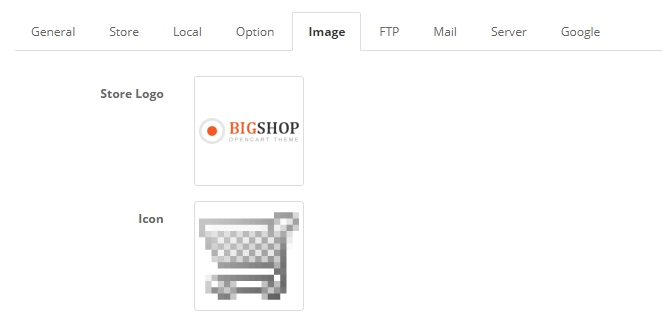


**Фигура 5-19 Настройки** **на магазина.**

#### 5.2.1.3. Раздел "Изображение"

Разделът "Изображение" в "Настройки" персонализира изображенията по подразбиране за магазина:

* **Лого на магазина:** Качете лого на магазин от магазина или компютъра на администратора чрез [Image Manager](http://docs.opencart.com/en-gb/administration/image-manager/) .
* **Икона:** Качете [favicon](http://en.wikipedia.org/wiki/Favicon) за магазина чрез [Image Manager](http://docs.opencart.com/en-gb/administration/image-manager/) .

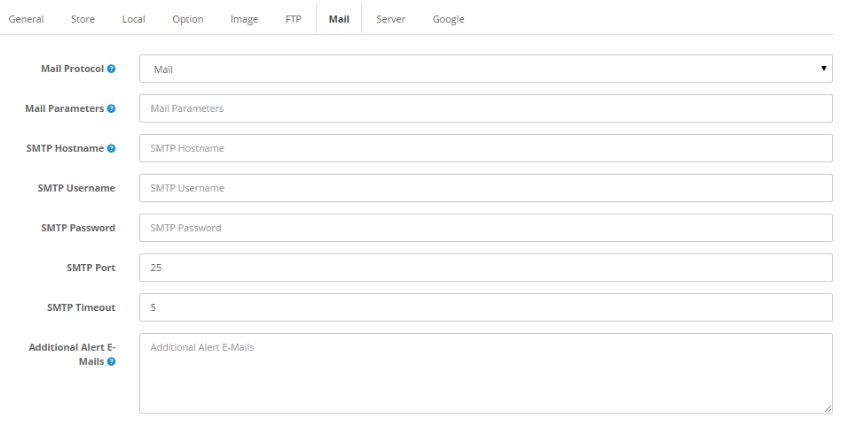


**Фигура 5-20 Настройки** **на Изображение.**

#### 5.2.1.4. Имейл настройки

Разделът за поща в "Настройки" показва следните полета:

* **Протокол за електронна поща:** Позволява ви да избирате между изпращане на поща до имейла на хоста или чрез SMTP. Трябва да се свържете с доставчика си на хостинг на магазина, за да видите дали да конфигурирате PHP пощата. Ако не, трябва да използвате SMTP, като например Gmail.
* **Mail Paramaters: Поиска** имейл адрес за магазина, ако е избрана "Mail" по Mail Protocol.
* **SMTP хост:** Адресът на хоста на smtp.
* **SMTP Потребителско име:** Пълният имейл адрес за хоста SMTP
* **SMTP парола**
* **SMTP порт:** номерът на порт, предоставен от SMTP хоста.
* **SMTP Timeout:** Номерът на изчакване, предоставен от SMTP хоста.
* **Нова поръчка Промяна на пощата:** Избирането на "Да" ще изпрати имейл с известие за всяка поръчка, подадена в магазина.
* **Нова поща за предупреждение в профила:** Избирането на "Да" ще изпрати имейл с известие за всеки профил, създаден в магазина.
* **Допълнителни електронни съобщения за предупреждения:** Избройте всички допълнителни имейли, които също трябва да получават уведомяващите имейли по-горе.



**Фигура 5-21. Имейл настройки.**

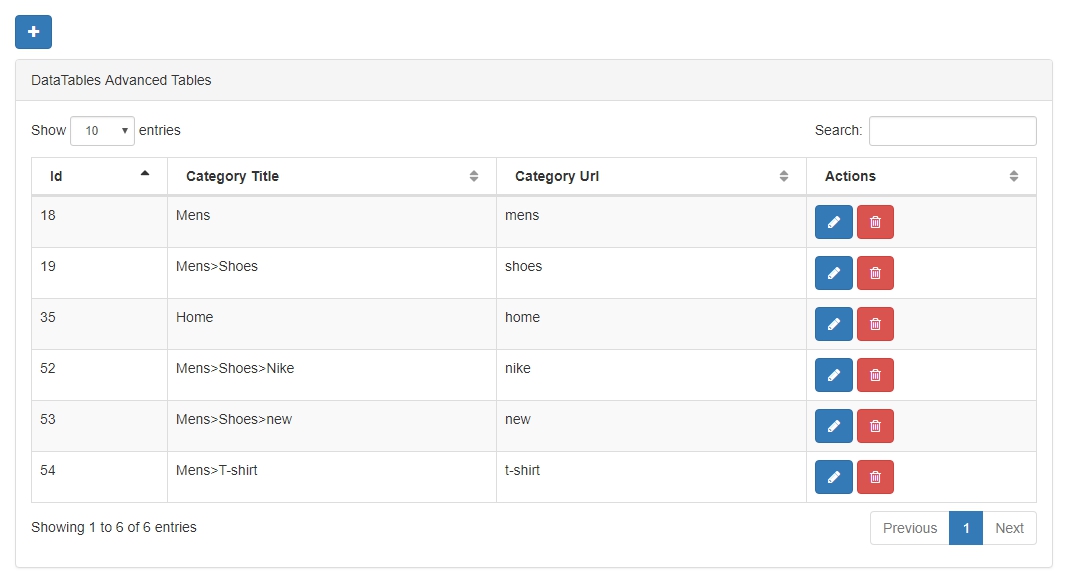
### 5.2.2. Управление на каталозите

Този раздел прави преглед на администрирането на каталога на онлайн магазина. Това включва категориите продукти, наричани страници и продукти.

#### 5.2.2.1. Каталог категории

Преди да научите как да добавяте продукти, ще трябва да разберете как да създадете продуктови категории. Всяка продуктова категория получава своето собствено място в магазина, за да покаже всички налични продукти за тази категория.

Организирането на продуктите в категории е полезно за навигация в инвентара на магазина както в магазина, така и в администрацията. В администрацията създаването на категории за продукти ще помогне на собственика на магазина да следи конкретни продукти в рамките на дадена категория. В предната част на магазина клиентите ще могат да разглеждат любимите си продукти по категории. Страниците за категории могат да бъдат достъпни в горното меню и от полето за категории на страницата с продукти.



**Фигура 5-22. Списък с категории.**

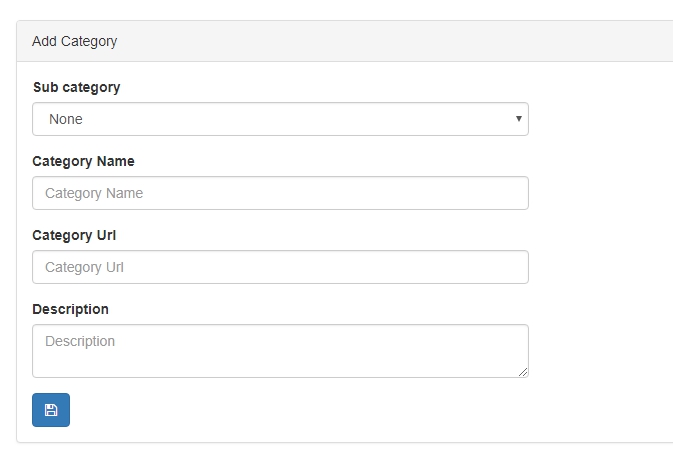
#### 5.2.2.2. Създайте категория

За да добавите продукт, ще кликнете върху бутона



**Разделът Създайте изисква следните данни:**

* **Име на категория \***
* **Описание** : Текст, който ще опише тази категория на страницата на категорията, която ще бъде видяна от клиентите в магазина пред магазина.
* **URL на категория**
* **Категория родител** : Категориите са подредени в йерархична структура; с основната категория винаги на върха. Можете да изберете да създадете категория под родителска категория, превръщайки я в подкатегория. В списъка с категории категорията ще се покаже като "Категория на родителите> Подкатегория". Ако е избрано "Няма", създадената категория ще бъде категория родител.



**Фигура 5-23.****Формa за** **Създаване на категория.**

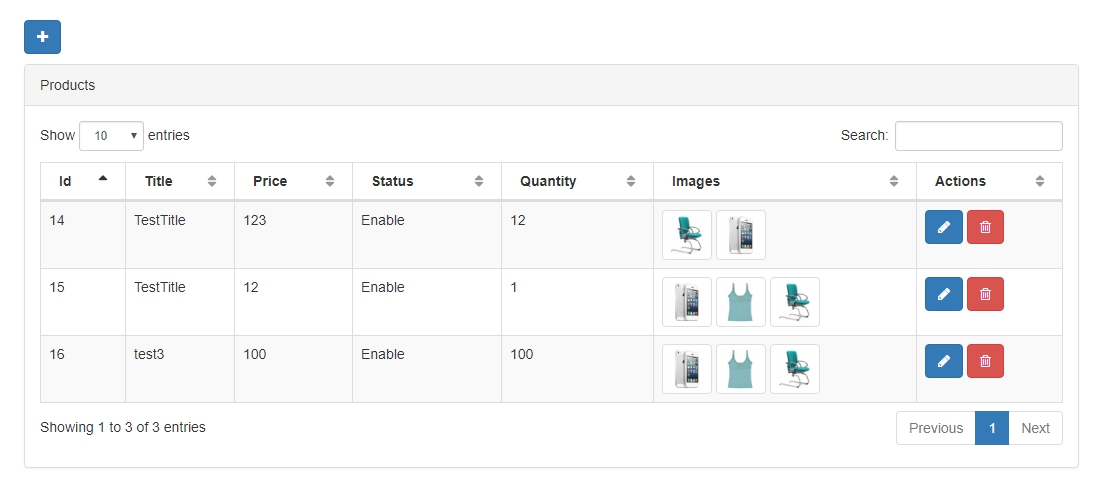
### 5.2.3.Управление на продукти

Продуктовата секция, достъпна под каталога, ще покаже всички налични в магазина продукти.

Всеки ред продукти ще съдържа следната информация:

* **Изображение**
* **Име на продукта**
* **Модел/ID**
* **SKU (склад за запаси)** : произволен код за продукта.
* **UPC (универсален код на продукта)** : уникалният баркод на продукта.
* **Цена**
* **Количество**
* **Състояние** : Избирането на "Активирано" позволява продуктът да бъде видим в магазина на магазина. Избирането на "Забранено" ще премахне продукта от изгледа му.

Проверката на отделно поле в първата колона ще избере информацията за продукта в същия ред. След като сте избрали, можете да изберете "Редактиране" или "Изтриване" на продукта. "Редактиране" ще редактира продукта и неговите информации и категории. "Изтриване" ще премахне напълно избора на продукт (включително и неговата информация).



**Фигура 5-24.****Списък с продукти.**

#### 5.2.3.1. Добавяне на продукти

**Добавяне на продукт**

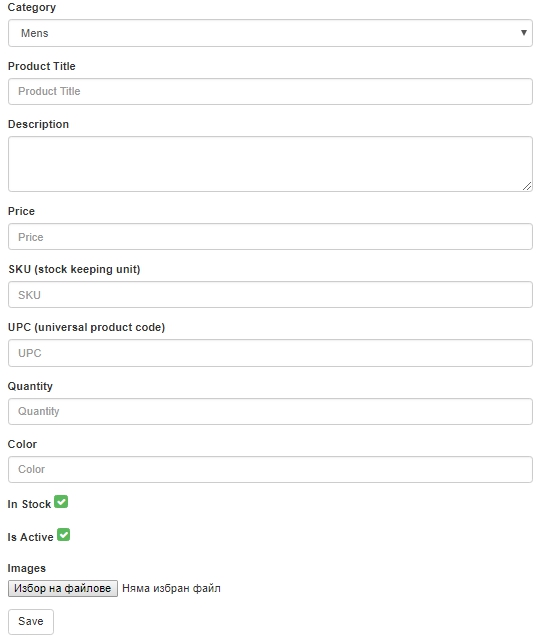
За да добавите продукт, ще кликнете върху бутона



Това води до страницата "Създаване на продукт", както е показано по-долу.

От страницата Създаване на продукт ще въведете следното:

* **Име на продукта**
* **Описание** : Текст, който ще опише този продукт на страницата на продукта, за да бъде видян от клиентите в магазина пред магазина.
* **Цена** : задайте цена за продукта.
* **Количество** : Общата налична сума в магазина за този конкретен продукт.
* **SKU (склад за запаси)** : произволен код за продукта.
* **UPC (универсален код на продукта)** : уникалният баркод на продукта.
* **Изображение** : кликването върху "Преглед на папките" ще доведе администратора до ***Image Manager*** да качи ново изображение.
* **Статус** : Разрешаването прави продукта публично достъпен в магазина. Деактивирането позволява продуктът да бъде редактиран в администрацията, но го скрива от предната част на магазина.
* **Категории** : Проверете категориите, към които ще бъде поставен продуктът. Продуктът ще бъде показан на страницата на категорията в магазина.
* **Наличност Състояние** : изберете "Няма наличност" или "Наличност" като съобщение, показано на страницата на продукта.



**Фигура 5-26. Формa за** **създаване на продукт.**

Отидете напред и кликнете върху бутона "Запазване", за да запазите новия си продукт.

#### 5.2.3.2. Редактиране на продукти

Екранът за редактиране на продукт също е достъпен от продуктовия списък, когато кликнете върху бутона "EDIT" до продукт, който изглежда така:

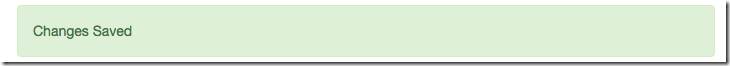


**Навигиране на екрана Редактиране на продукта**

Екранът за редактиране на продукти може да редактирате следното

* **Име на продукта** :Тук можете да управлявате името на продукта си
* **Описание** : Текст, който ще опише този продукт на страницата на продукта, за да бъде видян от клиентите в магазина пред магазина.
* **Цена** : задайте цена за продукта.
* **Количество** : Общата налична сума в магазина за този конкретен продукт.
* **Изображение** : кликването върху "Преглед на папките" ще доведе администратора до ***Image Manager*** да качи ново изображение.
* **Статус** : Разрешаването прави продукта публично достъпен в магазина. Деактивирането позволява продуктът да бъде редактиран в администрацията, но го скрива от предната част на магазина.
* **Категории** : Проверете категориите, към които ще бъде поставен продуктът. Продуктът ще бъде показан на страницата на категорията в магазина.
* **Наличност Състояние** : изберете "Няма наличност" или "Наличност" като съобщение, показано на страницата на продукта.

Накрая има бутон за запазване, за да се запазят всички промени. Промените няма да бъдат запазени, освен ако не кликнете върху бутона за запазване. Кликването върху бутона за запазване на екрана ще се опреснява и ще се покаже информационен изскачащ прозорец, както е показано по-долу, който потвърждава, че промените ви са запазени.



#### 5.2.3.4. Изтриване на продукти

Червеният бутон на страницата е бутонът за изтриване.



Кликването върху бутона за изтриване ще доведе до изтриването на продукта ви за постоянно.

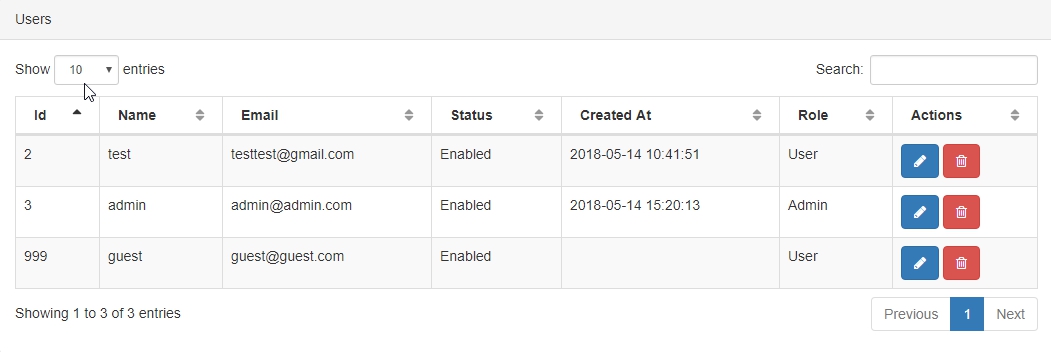
### 5.2.4 Управление на хората

#### 5.2.4.1. Управление на хората - общ преглед

#### Управлението на входа и изхода от администраторската страна на онлайн магазина може да бъде задачата само на един човек да се справи. Може да се окаже, че е необходимо да зададете разрешения за администрацията на конкретни хора от вашия екип. Потребителският раздел ви позволява да персонализирате кои потребители могат да имат достъп до администраторската страна на магазина и какви секции могат да бъдат достъпни или променени от тях.

**Потребители**

Ще имате достъп до потребителите, като кликнете върху "**Каталог**" и след това върху "**Потребители**", така че ще се появи екранът за управление на потребителите, както е показано по-долу.



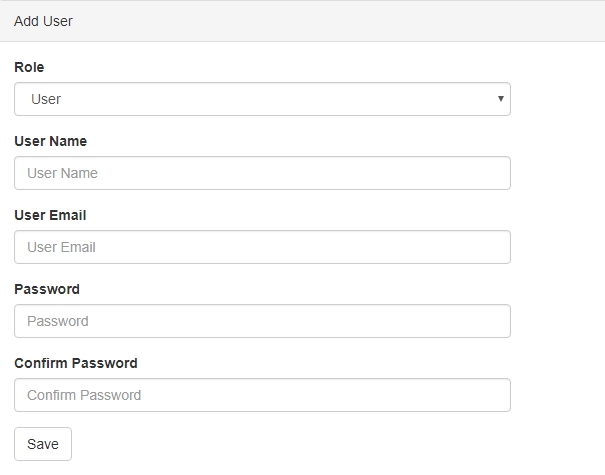
**Фигура 5-27. Списък с Потребители.**

От този екран можете да търсите в списъка си с потребители, да създавате нови потребители и да редактирате, управлявате или изтривате потребители според нуждите си.

#### 5.2.4.2. Създаване на нов потребител

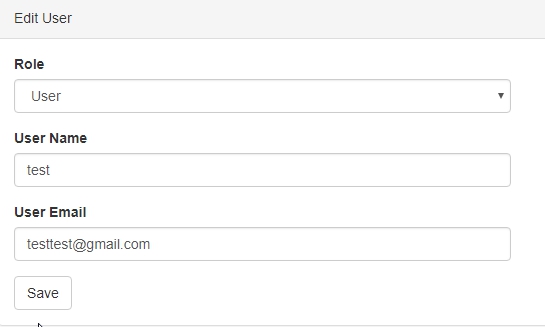
За да създадете нов потребител, кликнете върху бутона "Създаване на нов потребител" на екрана "Потребители" и екранът ще се актуализира, както е показано по-долу.





**Фигура 5-28. Формa за** **създаване на нов Потребител.**

#### 5.2.4.3. Редактиране на потребител

****За да редактирате потребител, кликнете върху полето за редактиране до потребителя, който искате да редактирате. Това ще покаже екрана "Редактиране на потребител", както е показано по-долу.

**Фигура 5-29. Формa за** **редактиране на Потребител.**

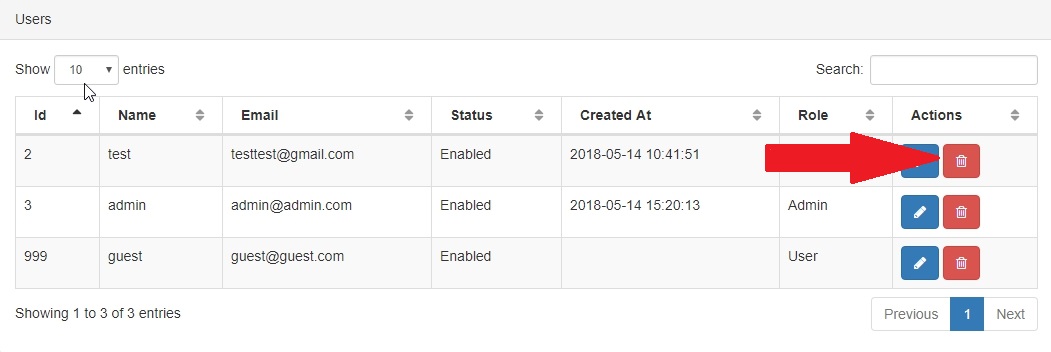
От този екран можете да редактирате следните полета за вашите потребители

1. Имейл - можете да промените имейл адреса, зададен на потребителя
2. Първо име - Можете да промените първото име на потребителя
3. Фамилия - Можете да промените фамилното име на потребителя
4. Роля - Тук можете да промените ролята на потребител във вашата система

След като направите промените, можете да кликнете върху бутона "Запазване", за да запазите редакциите си.

#### 5.2.4.4. Изтриване на потребител

За да изтриете потребител, кликнете върху връзката за изтриване до потребителя в списъка с потребители, както е посочено по-долу.

****

**Фигура 5-30. Изтриване на потребител.**

Изтритите потребители не могат да бъдат обръщани или деактивирани, така че използвайте тази функция предпазливо.

### 5.2.5. Управление на Поръчки

Когато даден клиент преминава през Checkout, информацията за поръчката му автоматично се прехвърля в секцията Поръчки, за да можете да проследявате. В администрацията можете да видите всички поръчки, направени на сайта им, ръчно да добавите поръчки или да редактирате данните за съществуващи поръчки.

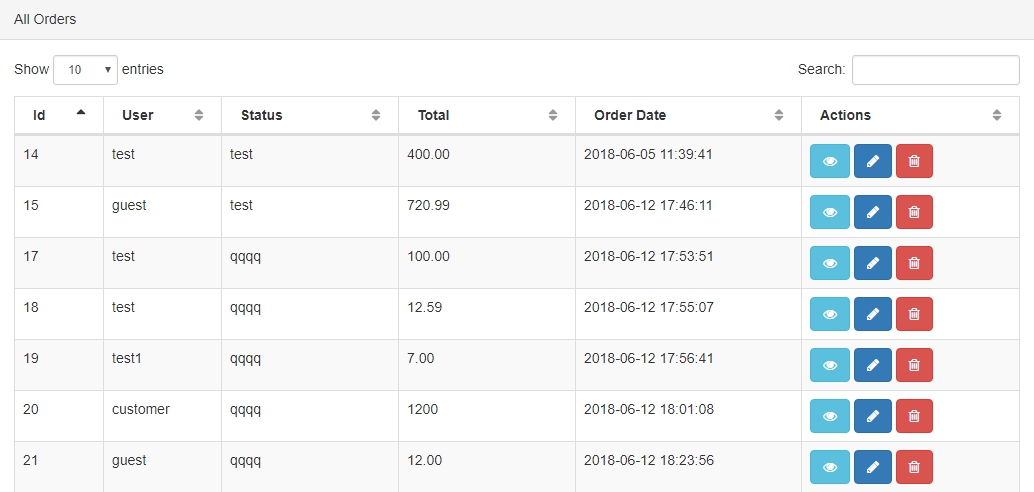
**Разглеждане на подробности за поръчката**

Секцията Поръчки се намира в раздел "**Каталог** **>** **Поръчки**". На тази страница всяка поръчка, направена някога от магазина, е описана подробно.

Следните подробности се показват за всяка поръчка в списъка:

* **Идент. № на поръчката**
* **Клиент** : името на клиента, извършил поръчката.
* **Състояние** : актуализиране на състоянието на поръчката. След първоначалното създаване на поръчката статусът ще бъде "Чакащ". Опциите по подразбиране са "Липсващи поръчки", "Анулирани", "Анулирани", "Обратно връщане", "Пълна", "Отказана", "Изтекла", "Неуспешна".
* **Общо** : общата цена на поръчката.
* **Датата е добавена**
* Преглед на бутона за поръчка

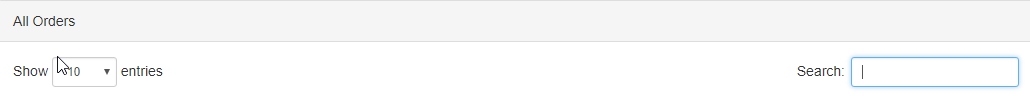
След като изберете, ще се покаже списък на вашите поръчки, както и текущото им състояние

****

**Фигура 5-31. Списък с поръчки.**

**Търсене на** **поръчки**

Ако искате бързо да търсите поръчки, можете да използвате полето за търсене отдясно на страницата с поръчки, както е показано по-долу:

****

**Фигура 5-32. Търсене на поръчки.**

С бързи опции за състояние, състояние на плащане и състояние на доставка.

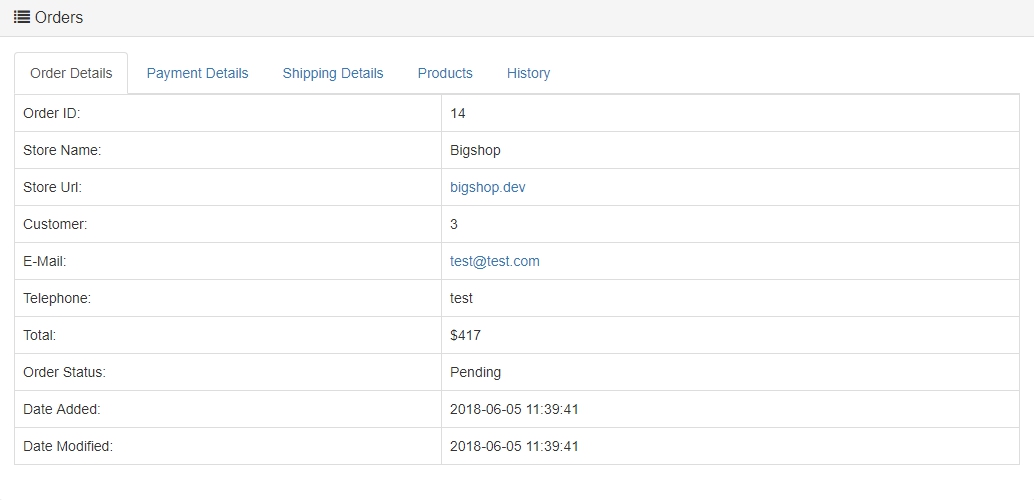
**Преглед на поръчка**

За да видите поръчка, кликнете върху бутона "Преглед" до реда в списъка с поръчки. Това ще доведе до страницата с подробности за поръчката, както е показано по-долу.



**Данни за клиента**

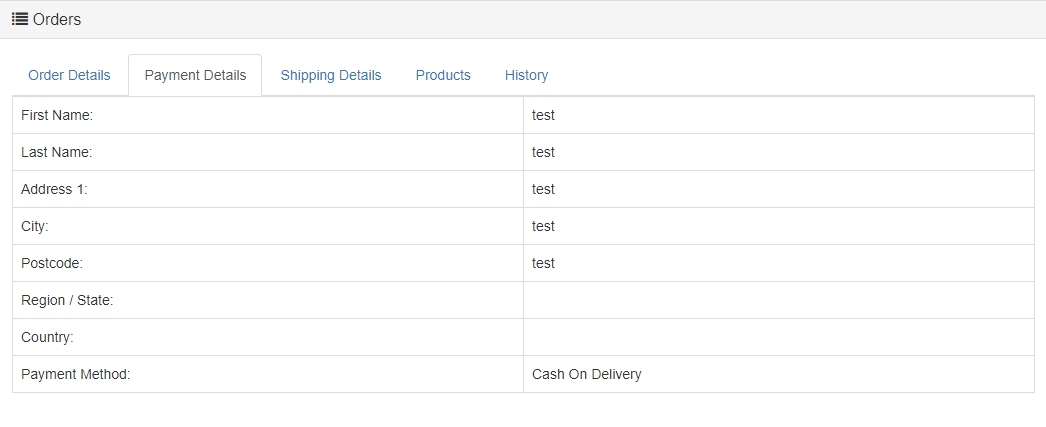
Първата секция за промяна на поръчка се нарича "Данни на клиента". Въвеждането на име на клиент, което вече е създало акаунт в магазина, ще попълни автоматично формуляра, като изведе информацията си, за да попълните празните места.



**Фигура 5-33. Данни за клиента.**

**Подробности на плащане**

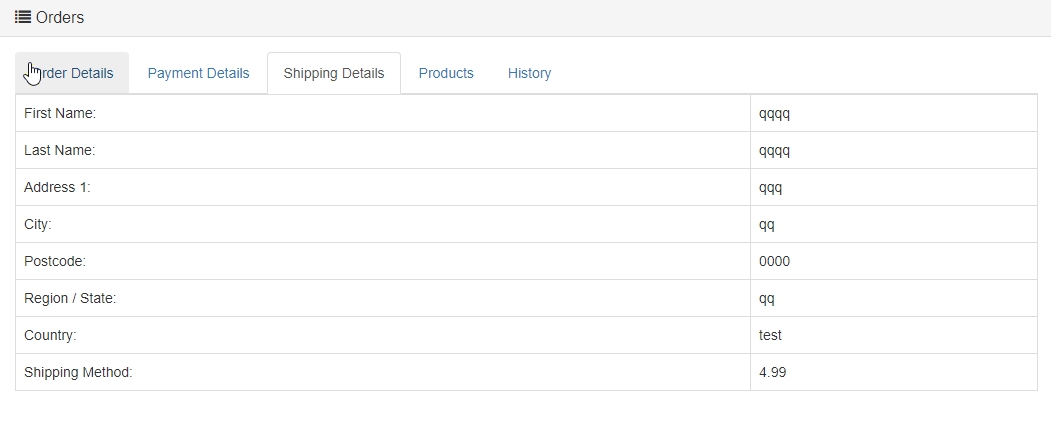
Ако към клиента е добавен адрес, изберете опцията за адрес в "Избор на адрес". В противен случай адресът на клиента ще трябва да бъде попълнен ръчно във формуляра.



**Фигура 5-34. Подробности на плащане.**

**Данни за доставката**

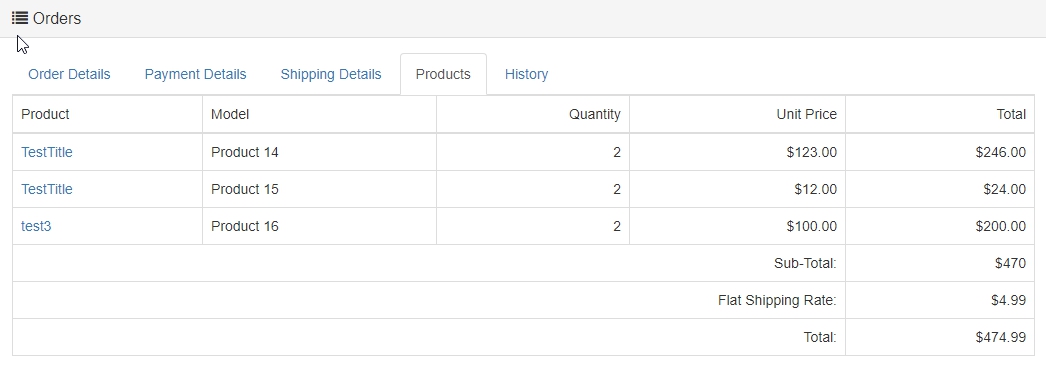
Този раздел проследява адреса за доставка за тази конкретна поръчка. Ако адресът за доставка и плащането са същите, копирайте и поставете информацията по-горе, когато я добавяте ръчно.



**Фигура 5-35. Данни за доставката.**

**Продукти**

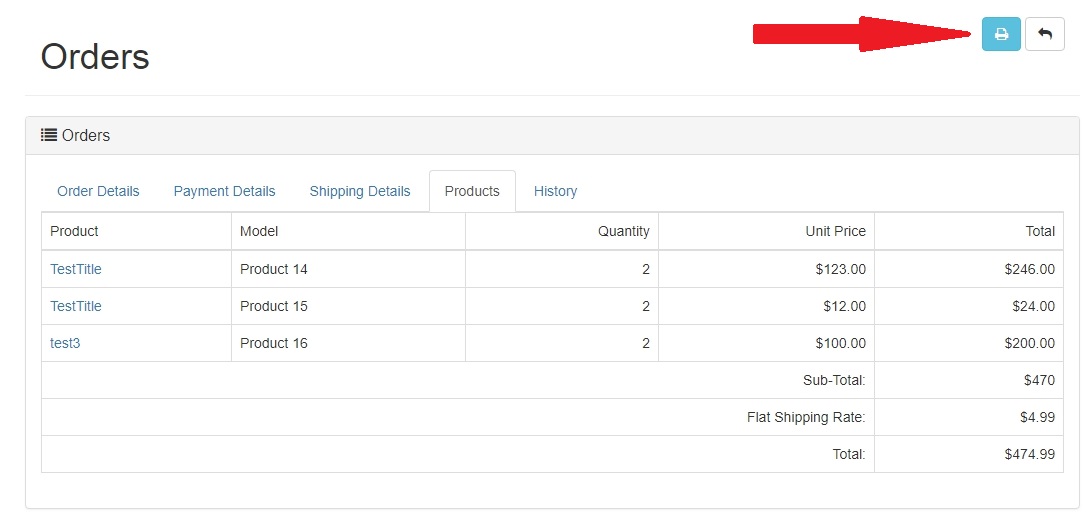
В "Избор на продукт" започнете да въвеждате името на продукта. Ще се покаже падащ списък с продукти, съдържащи това име. Избирането на продукт от този списък ще доведе до конкретни опции за продукта за тази поръчка.



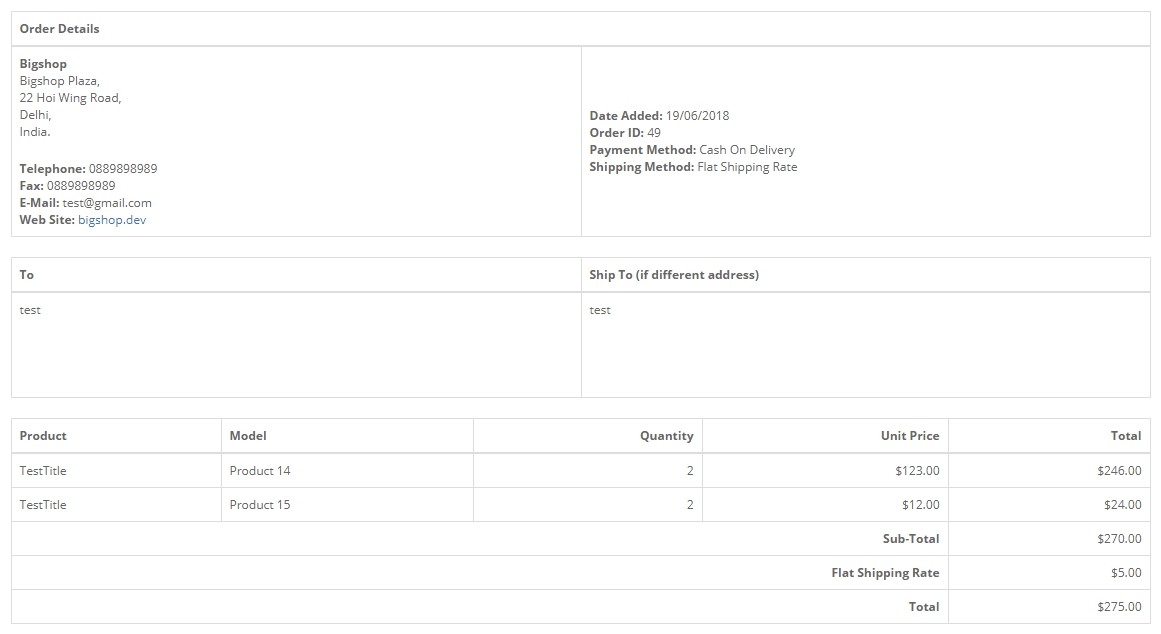
**Фигура 5-36. Продукти** **за тази поръчка.**

**Отпечатване на фактури**

За да отпечатате фактура, върнете се на страницата Поръчки в раздел Продажби> Поръчки. Над списъка с поръчки в горния десен ъгъл е бутонът "Печатна фактура". Изберете реда за отпечатване и кликнете върху "Отпечатване на фактура". Това действие ще ви насочи към страница за печат на фактурата. Фактурата може да бъде отпечатана и включена в доставката.



**Фигура 5-37. Бутон за Отпечатване на фактури.**

****

**Фигура 5-38. Фактура за отпечатване.**

### 5.2.6. Качване на изображение

**Свойства на продукта – изображения**

Настройките за изображения за продукти се намират, като кликнете върху опцията в менюто "Изображения" в лявото меню на страницата с продуктите. Тук можем да качим изображения за нашите продукти и след това можем да зададем и текста на надписите и "Alt Image Text", които са важни за използваемостта и достъпността с увреждания.

**Мениджър на изображения**

Image Manager е инструмент, използван в администрацията за качване на файлове с изображения. Банери, изображения на продукти, логото на магазина и производителите ще имат нужда от изображения, качени с помощта на Image Manager.

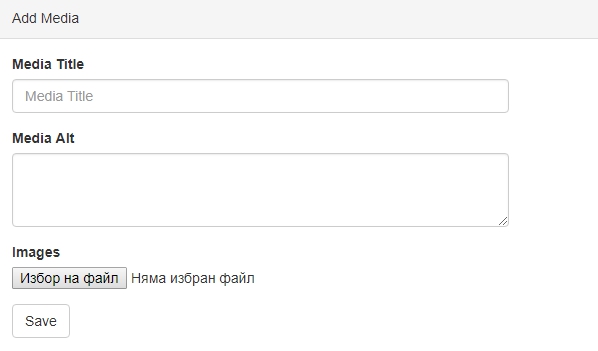


**Фигура 5-39. Списък с изображения.**

За да качите изображение за продукт, изберете опцията "Избор на файлове" на екрана, показан по-долу, изберете изображението от компютъра и кликнете върху "Качване на изображения", за да качите изображението в магазина.

**Редактиране на текст на изображение**

След качването на изображението кликнете върху бутона "Редактиране", както е показано на изображението по-горе, и това показва екрана "Редактиране на изображение на продукта", показан по-долу



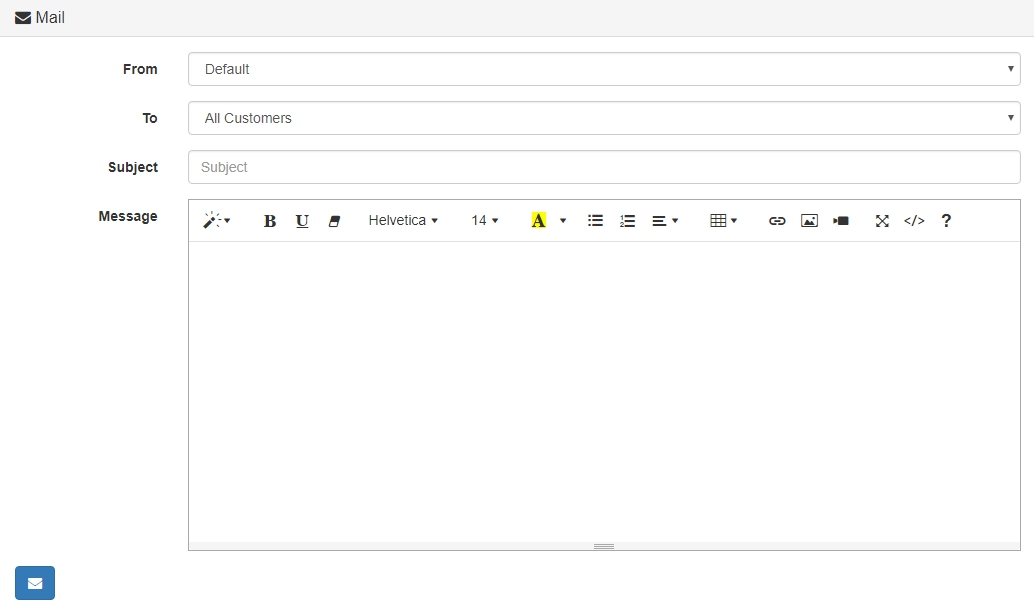
**Фигура 5-40. Редактиране на текст на изображение.**

Оттук ще въведете нов текст за надписи и Alt Text, които ще се показват за изображението на вашия продукт. След това кликнете върху бутона "Запазване", за да запазите тези промени. Екранът ще се опреснява обратно до екрана "Изображения" и промените ви автоматично се запазват.

### 5.2.7. Поща

**Изпращане по имейл на група клиенти**

Системата Mail ви позволява да изпращате имейли до определени клиенти или на всички клиенти. Можете да използвате тази функция, за да изпращате бюлетини, информация за специални оферти или да съобщавате за всякакъв вид информация за магазина. Страницата "Поща" може да бъде достъпна от "Продажби> Поща".



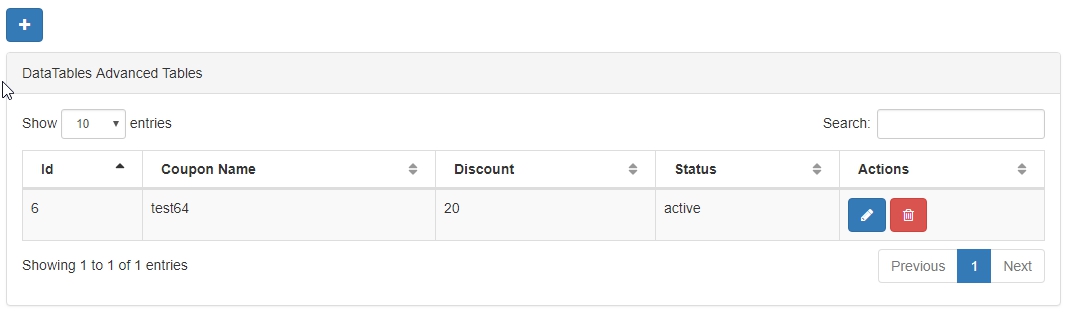
**Фигура 5-41. Изпращане по имейл.**

Изберете групата клиенти в полето "До:", създайте тема за имейла и съобщение и кликнете върху "Изпращане". Този имейл ще бъде изпратен на всички клиенти с тази конкретна селекция от клиенти.

### 5.2.8. Купони

Можете да създавате и обозначавате талони за конкретни продукти или продуктови категории в раздела Продажби> Купони.

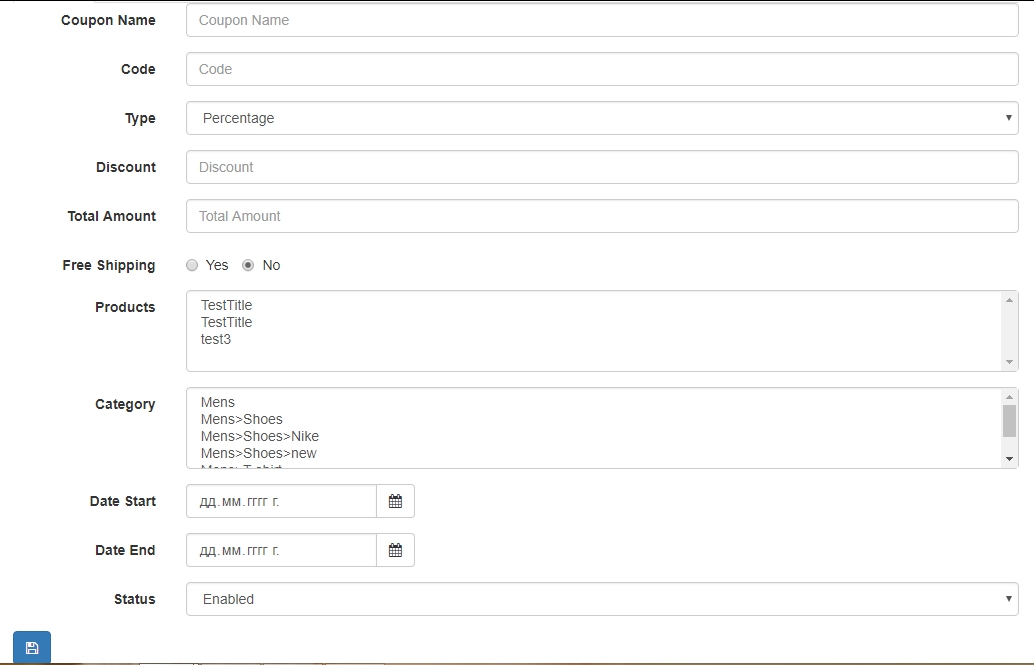
В страницата Купони ще се покаже списък с всички талони, създадени в администрацията.



**Фигура 5-42. Списък с Купони.**

Редактирането или вмъкването на талони ще доведе администратора до формуляр, съдържащ следните полета:

* **Име на купона** : Описателно име за талона.
* **Код:** Код, който клиентът трябва да въведе в пазарската кошница, за да използва купона в предната част на магазина.
* **Тип:** Отстъпката може да бъде определена сума или процент от размера на общата сума.
* **Отстъпка:** Броят, изваден от общата сума за покупка.
* **Обща сума:** Сумата, която клиентът трябва да достигне, преди да използва талона.
* **Вход за клиента:** Избирането на "да" изисква клиентът да бъде влязъл в профила си в магазина, за да използва купона. Избирането на "не" ще позволи на клиента да използва талона като гост.
* **Безплатна доставка:** Избирането на "да" дава на клиента безплатна доставка за използване на отстъпката код.
* **Категория:** Избирането на категория продукти ще прилага отстъпката за всички продукти в рамките на тази категория. Вижте „Категории“ за повече информация как да създадете нова категория продукти.
* **Продукти:** Индивидуалните продукти могат да бъдат избрани чрез въвеждане на името на продукта в предоставеното място. За да премахнете продукт от отстъпката, кликнете върху червения минус бутон в списъка с продукти под него.
* **Дата на стартиране:** Първият ден, в който може да се използва отстъпката.
* **Крайна дата:** В последния ден може да се използва отстъпката.
* **Употреби на купон:** Ограничава колко пъти купонът може да бъде използван от клиентите.
* **Статус:** Активира или деактивира талона за употреба в предната част на магазина.



**Фигура 5-43. Редактирането или вмъкването на талони.**

# **ЗАКЛЮЧЕНИЕ**

Ако взема под внимание темата на дипломната работа и представените изходни резултати смятам, че уеб приложението е разработено съгласно всички изисквания от страна на интерфейса (дизайна на уеб приложение) и логиката на програмната реализация. Приложението наподобява до известна степен конкурентни приложения, които са успешно доказани на пазара. Приликата е най-вече в това, че през целият етап от проектирането до реализирането съм се придържал към основните пазарни изисквания към такъв тип приложения. Системата е “user friendly”, тоест достатъчно интуитивна, за да даде възможност на потребители от различни възрастови групи и с различни познания в областта на информационните технологии, да могат да се ориентират безпроблемно при работа с него.

# **ИЗПОЛЗВАНА ЛИТЕРАТУРА**

1. “Database”: <https://www.amazon.com/All-One-Dummies-Allen-Taylor/dp/0470929960/ref=lp_3789_1_10?s=books&ie=UTF8&qid=1530779063&sr=8-10>

2. Релационна база от данни в “MySQL”: <https://www.w3schools.com/sql/default.asp>

3. “PHP”: <http://php.net/>

4. “Model-View-Controller”:<https://www.amazon.de/Professional-Apress-Experts-Voice-Source/dp/1430241640>

5. “Laravel”: <https://laravel.com/>