Міністерство освіти і науки України

Харківський радіотехнічний технікум

**ЗВІТ**

**З НАВЧАЛЬНОЇ КОМП’ЮТЕРНОЇ ПРАКТИКИ**

спеціальності 5.05010301

«Розробка програмного забезпечення»

Виконав (виконала):

студент (ка) групи ПР-425

ФІО

Перевірив:

Викладач

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оцінка\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Харків 2019

Оглавление

[Введение 3](#_Toc5453486)

[Раздел I Постановка задачи 4](#_Toc5453487)

[Раздел II Используемые технологии 5](#_Toc5453488)

[Раздел III Программная реализация 9](#_Toc5453489)

[Раздел IV Инструкция пользователя 27](#_Toc5453490)

[Вывод 37](#_Toc5453491)

# Введение

Интернет-магазин, отвечающий всем современным требованиям, является торговым каналом, с помощью которого предприятие имеет возможность продавать свои товары и услуги многомиллионной аудитории, пользующейся услугами интернет сети. В современных условиях создание интернет-магазина приобретает все большую актуальность. Это позволяет компании укреплять свои позиции не только на традиционных рынках, но и выходить далеко за их пределы.

Интернет-магазин имеет множество преимуществ, с его помощью посетители сайта не только имеют возможность получить полную информацию о товарах и услугах, но и сделать заказ. При этом практически не важно, в какой части мира находится покупатель, его удаленность от продавца не имеет существенного значения, ведь информация, представленная максимально полно, может быть найдена в течение минуты, где бы вы не находились. Оформив заказ, покупатель в течение некоторого времени ожидает его доставки, при этом совершенно не важно, где находятся офисные или складские помещения продавца и где продавец приобретает товар. Это удобно для обеих сторон и является еще одним неоспоримым преимуществом интернет-магазина перед традиционными торговыми точками.

Наблюдающийся рост числа граждан, совершающих покупки через Интернет, связан, в том числе, и с удобством, поскольку для интернет торговли не обязательно находиться в конкретном магазине или даже городе, все, что необходимо — устройство с выходом в Интернет. Появляется возможность сравнить цены на товар сразу в нескольких интернет-магазинах, при этом никуда не надо ходить. Для жителей малых городов и деревень с небольшим количеством магазинов и скудным выбором товара интернет-магазин позволяет заказать именно тот товар, который нужен, а не который есть, в пару действий и несколько дней ожидания доставки. Поэтому у интернет-магазина, предлагающего качественные товары и услуги, всегда найдутся покупатели, а со временем их число только возрастет.

В настоящее время все более востребованным является создание интернет-магазина под ключ. В этом случае исполнитель создает многофункциональный, самодостаточный и полностью завершенный проект в полном соответствии с пожеланиями заказчика. В результате технически сложной, комплексной работы заказчик получает готовую торговую площадку для своего бизнеса, а сочетание понятного и удобного интерфейса с привлекательным дизайнерским решением позволит использовать возможности интернета наиболее эффективно.

# Раздел I Постановка задачи

Постановка задачи. Подробно про разрабатываемый функционал. 4 страницы примерно.

В ходе практической работы необходимо создать функционал интернет-магазина спортивной обуви, а именно: подготовка серверной части приложения (то есть, какие обработчики с какой страницы будут вызываться, при каких сценариях), внедрение базы данных, использование интерфеса PDO для работы скриптов серверной части базы данных и технологии обращения к серверу без перезагрузки страницы (AJAX - аббревиатура от «Asynchronous Javascript And Xml»). Также необходимо с помощью Selenium WebDriver реализовать автоматизированное тестирование (со своим приложением-ботом), составить и использовать тестовые сценарии, соответствующие типовым операциям пользователя.

Реализован следующий функционал:

1. Регистрация пользователя – для этого необходимо заполнять поля «Имя», «Фамилия», «E-mail», «Пароль», «Повтор пароля». Все поля обязательны. Также предусмотрена проверка на совпадение паролей, на наличие такого пользователя
2. Вход в аккаунт – заполнение полей «E-mail» и «Пароль»
3. Страница со всеми товарами – содержит несколько карточек товара с названием и ценой
4. Просмотр карточки товара – содержит несколько фотографий товара, название, описание, цену, размеры, которые есть на складе, характеристика.
5. Добавление товара в корзину – необходимо выбрать размер, количество.
6. Корзина – список с предварительно добавленными товарами, возможно изменять количество или удалять товар.

# Раздел II Используемые технологии

Термин HTML (аббревиатура от англ. Hyper Text Markup Language) — в дословном переводе, «язык гипертекстовой разметки». Является основным и единым (но не единственным) стандартом для обмена данными в связке «сервер — клиент». В веб-дизайне служит основой всех технологий

HTML обеспечивает средства для практически полномасштабного формирования веб-страниц — то есть именно того, что мы с вами наблюдаем на экране собственного дисплея, когда посещаем какой-либо из сетевых ресурсов. Всё оформление страницы (тексты, изображения, заголовки и комментарии, таблицы и так далее), цвет и звук на сайте, а также важнейший элемент технологий веб-дизайна— являются результатом форматирования именно способами и средствами HTML.

CSS используется для стилизации и верстки веб-страниц. С помощью него можно менять шрифты, цвета, расстояние между блоками, разделять контент на колонки, добавлять анимацию и прочие декоративные элементы.

JavaScript — это кроссплатформенный, объектно-ориентированный скриптовый язык, являющийся небольшим и легковесным. Внутри среды исполнения JavaScript может быть связан с объектами данной среды и предоставлять программный контроль над ними, даёт возможность реализовывать сложное поведение веб-страницы - позволяет создать динамически обновляемый контент, управляет мультимедиа, анимирует изображения.

Современный JavaScript – это «безопасный» язык программирования. Он не предоставляет низкоуровневых средств работы с памятью, процессором, так как изначально ориентирован на браузеры, в которых это не требуется.

В браузере JavaScript может делать всё, что относится к манипуляции со страницей, взаимодействию с посетителем и, в какой-то мере, с сервером: создавать новые HTML-теги, удалять существующие, менять стили элементов, прятать, показывать элементы, реагировать на действия посетителя, обрабатывать клики мыши, перемещения курсора, нажатия на клавиатуру; посылать запросы на сервер и загружать данные без перезагрузки страницы (эта технология называется "AJAX"); получать и устанавливать cookie, запрашивать данные, выводить сообщения и т.д.

Особенности JavaScript:

* Полная интеграция с HTML/CSS.
* Простые вещи делаются просто.
* Поддерживается всеми распространёнными браузерами и включён по умолчанию.

Этих трёх вещей одновременно нет больше ни в одной браузерной технологии.Поэтому JavaScript и является самым распространённым средством создания браузерных интерфейсов.

Bootstrap это фреймворк для создания современных, кросс-браузерных и стандартизованных интерфейсов. Продуманная структура кода HTML, JavaScript и CSS дает возможность создавать множество самых разнообразных элементов интерфейса и сетку сайта.

Свидетельством того, что Bootstrap это очень удачная разработка, является его непрерывно растущая популярность. Нынешний уровень развития этого фреймворка уже позволяет полностью создать любой web-интерфейс.

Основным преимуществом использования Bootstrap является Less – динамический язык стилей, существенно расширяющий возможности CSS. С его помощью разработчики могут создавать переменные, вложенные колонки, управлять цветами и т.д.

Основные инструменты Bootstrap:

* Сетки — заранее заданные размеры колонок, которые можно сразу же использовать, например ширина колонки 90px относится к классу .span2, который мы можем использовать в CSS описании документа;
* Шаблоны — фиксированный или резиновый шаблон документа;
* Типографика — описания шрифтов, определение некоторых классов для шрифтов таких как код, цитаты и т.п.;
* Медиа — представляет возможности управления изображениями и видео;
* Таблицы — средства оформления таблиц, вплоть до добавления функциональности для обеспечения возможности сортировки;
* Формы — классы для оформления не только форм, но и некоторых событий происходящих с ними;
* Навигация — классы оформления для вкладок, страниц, меню и панелей инструментов;
* Алерты — оформление диалоговых окон, подсказок и всплывающих окон.

Язык PHP предназначен для написания скриптов, которые выполняются на стороне сервера, а в браузер клиента отправляется не сам скрипт, а только результаты его работы. Это означает, что пользователь, загрузивший страницу вашего сайта, никогда не увидит исходный код скрипта и может даже не догадываться, что страница создана динамически.

В свою очередь, на языке JavaScript создаются исключительно клиентские скрипты, которые исполняются в браузере клиента. Для сервера скрипты JavaScript – это обычные текстовые данные, ничем не отличающиеся от прочего содержимого страницы.

Различаются языки и по уровню безопасности. При выполнении PHP-скрипта вы можете обращаться к файловой системе сервера, базам данных и другим внешним объектам, на которые вам предоставили права администраторы. А вот JavaScript ограничен только встроенными объектами браузера и с их помощью вы не сможете читать или писать непосредственно на диск. Сделано это в первую очередь для защиты от злоумышленников, т.к. если разрешить доступ скриптам к системной информации и файлам – обязательно найдётся умелец, который напишет код, скрытно отсылающий ваши файлы кошельков, логины, пароли и другую закрытую информацию на адрес умельца.

Помимо прочего, протокол HTTP, по которому передаются все веб-страницы, не поддерживает постоянных соединений, т.е. браузер подключается к серверу, запрашивает страницу, получает данные и немедленно отключается. Это означает, что все скрипты PHP завершаются ещё ДО того, как начнутся выполняться скрипты JavaScript. По этому, прямое взаимодействие скриптов на PHP и JavaScript невозможно.

Косвенное взаимодействие возможна. Для передачи данных между клиентской и серверной стороной можно использовать один из трёх вариантов:

* Внесение изменений в текст страницы (в скрипт);
* Использование Cookies;
* Использование Ajax;

MySQL - это популярная система управления базами данных (СУБД), очень часто применяемая в сочетании с PHP.

База данных представляет собой структурированную совокупность данных. Эти данные могут быть любыми - от простого списка предстоящих покупок до перечня экспонатов картинной галереи или огромного количества информации в корпоративной сети. Для записи, выборки и обработки данных, хранящихся в компьютерной базе данных, необходима система управления базой данных, каковой и является ПО MySQL. Поскольку компьютеры замечательно справляются с обработкой больших объемов данных, управление базами данных играет центральную роль в вычислениях. Реализовано такое управление может быть по-разному - как в виде отдельных утилит, так и в виде кода, входящего в состав других приложений.

MySQL - это система управления реляционными базами данных. В реляционной базе данных данные хранятся не все скопом, а в отдельных таблицах, благодаря чему достигается выигрыш в скорости и гибкости. Таблицы связываются между собой при помощи отношений, благодаря чему обеспечивается возможность объединять при выполнении запроса данные из нескольких таблиц. SQL как часть системы MySQL можно охарактеризовать как язык структурированных запросов плюс наиболее распространенный стандартный язык, используемый для доступа к базам данных.

MySQL состоит из двух частей: серверной и клиентской.

Сервер MySQL постоянно работает на компьютере. Клиентские программы (например, скрипты PHP) посылают серверу MySQL SQL-запросы через механизм сокетов (то есть при помощи сетевых средств), сервер их обрабатывает и запоминает результат. То есть скрипт (клиент) указывает, какую информацио он хочет получить от сервера баз данных. Затем сервер баз данных посылает ответ (результат) клиенту (скрипту)

Cтруктура MySQL трехуровневая: базы данных — таблицы — записи. Базы данных и таблицы MySQL физически представляются файлами с расширениями frm, MYD, MYI. Логически - таблица представляет собой совокупность записей. А записи - это совокупность полей разного типа. Имя базы данных MySQL уникально в пределах системы, а таблицы - в пределах базы данных, поля - в пределах таблицы. Один сервер MySQL может поддерживать сразу несколько баз данных, доступ к которым может разграничиваться логином и паролем. Зная эти логин и пароль, можно работать с конкретной базой данных. Например, можно создать или удалить в ней таблицу, добавить записи и т. д. Обычно имя-идентификатор и пароль назначаются хостинг провайдерами, которые и обеспечивают поддержку MySQL для своих пользователей.

Расширение PHP Data Objects (PDO) определяет легкий, согласованный интерфейс для доступа к базам данных на PHP. Каждый драйвер базы данных, реализующий интерфейс PDO, может выставлять функции, специфичные для базы данных, как регулярные функции расширения. Обратите внимание, что вы не можете выполнять какие-либо функции базы данных, используя само расширение PDO; вы должны использовать драйвер PDO для конкретной базы данных для доступа к серверу базы данных.

Selenium - это инструмент с открытым исходным кодом и переносным автоматическим программным обеспечением для тестирования веб-приложений. Он имеет возможности работать в разных браузерах и операционных системах. Selenium - это не просто инструмент, а набор инструментов, который помогает тестировщикам более эффективно автоматизировать веб-приложения.

В наборе Selenium доступны для использования:

Selenium Integrated Development Environment (IDE) - это плагин Firefox, который позволяет тестировщикам записывать свои действия, следуя технологическому процессу, который им нужно протестировать.

Selenium Remote Control (RC) была система тестирования флагманской, что позволило более чем простые действия браузера и линейное исполнение. Он использует всю мощь языков программирования, таких как Java, C #, PHP, Python, Ruby и PERL для создания более сложных тестов.

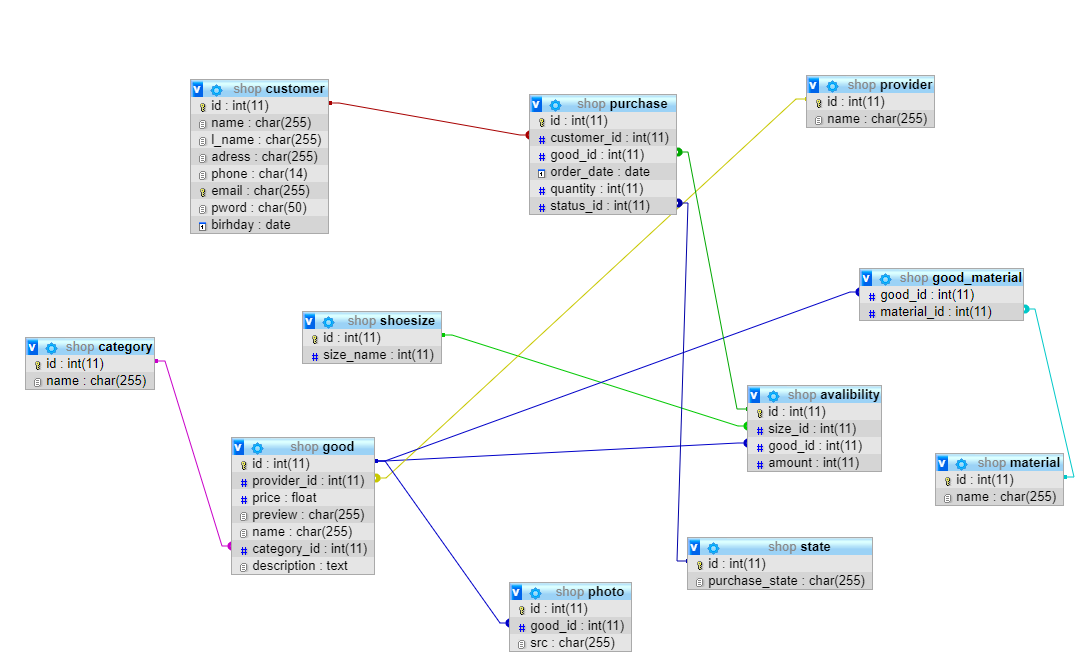
Selenium WebDriver - является преемником Selenium RC, который отправляет команды непосредственно в браузер и извлекает результаты.

Selenium Grid - это инструмент, используемый для параллельного тестирования на разных компьютерах и разных браузерах одновременно, что приводит к минимизации времени выполнения.

В рамках проекта Selenium разрабатываются драйверы для браузеров Firefox, Internet Explorer и Safari, а также драйверы для мобильных браузеров Android и iOS. Драйвер для браузера Google Chrome разрабатывается в рамках проекта Chromium, а драйвер для браузера Opera (включая мобильные версии) разрабатывается компанией Opera Software. Поэтому они формально не являются частью проекта Selenium, распространяются и поддерживаются независимо.

Аналогичная ситуация и с клиентскими библиотеками – в рамках проекта Selenium разрабатываются библиотеки для языков Java, .Net (C#), Python, Ruby, JavaScript. Все остальные реализации не имеют отношения к проекту Selenium

# Раздел III Программная реализация



**База данных «shop»** – используется для хранения данных о продуктах, покупателях и заказов.

Создание:

CREATE DATABASE IF NOT EXISTS `shop` DEFAULT CHARACTER SET utf8 COLLATE utf8\_general\_ci;

**Таблица «avalibility»** – связующая таблица между «size» и «good», хранит в себе данные о количестве товара.

*Структура:*

`id` int(11) – идентификатор, первичный ключ, целое число, максимальное количество символов– 11, автоинкремент,

`size\_id` int(11) – вторичный ключ ссылается на таблицу «shoesize», поле id, число, максимальное количество символов – 11,

`good\_id` int(11) – вторичный ключ ссылается на таблицу «good», поле id, число, максимальное количество символов – 11,

`amount` int(11) – количество, целое число, максимальное количество символов– 11.

*Создание таблицы:*

CREATE TABLE `avalibility` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`size\_id` int(11) DEFAULT NULL,

`good\_id` int(11) DEFAULT NULL,

`amount` int(11) NOT NULL,

FOREIGN KEY(size\_id) REFERENCES shoesize(id) ON UPDATE CASCADE ON DELETE SET NULL,

FOREIGN KEY(good\_id) REFERENCES good(id) ON UPDATE CASCADE ON DELETE SET NULL) ENGINE=InnoDB DEFAULT CHARSET=utf8;

*Добавление записей:*

INSERT INTO `avalibility` (`id`, `size\_id`, `good\_id`, `amount`) VALUES (1, 4, 1, 100), (2, 1, 1, 80),

(3, 2, 1, 50);

**Таблица «category»** – хранит в себе данные о поле покупателя.

*Структура:*

`id` int(11)– идентификатор, первичный ключ, целое число, максимальное количество символов– 11, автоинкремент,

`name` char(255) – название, символьный тип, максимальное количество символов – 255.

*Создание таблицы:*

CREATE TABLE `category` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`name` char(255) NOT NULL) ENGINE=InnoDB DEFAULT CHARSET=utf8;

*Добавление записей:*

INSERT INTO `category` (`id`, `name`) VALUES (1, 'Мужская'), (2, 'Женская');

**Таблица «customer»** – хранит в себе данные о пользователе.

*Структура:*

`id` int(11)– идентификатор, первичный ключ, целое число, максимальное количество символов– 11, автоинкремент,

`name` char(255) – имя, символьный тип, максимальное количество символов – 255,

`l\_name` char(255) – фамилия, символьный тип, максимальное количество символов – 255,

`adress` char(255) – адрес, символьный тип, максимальное количество символов – 255,

`phone` char(14) – телефон, символьный тип, максимальное количество символов – 14,

`email` char(255) – электронная почта, символьный тип, максимальное количество символов – 255,

`pword` char(50) – пароль, символьный тип, максимальное количество символов – 50,

`birhday` date – день рождения, тип данных дата.

*Создание таблицы:*

CREATE TABLE `customer` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`name` char(255) NOT NULL,

`l\_name` char(255) NOT NULL,

`adress` char(255) DEFAULT NULL,

`phone` char(14) DEFAULT NULL,

`email` char(255) NOT NULL,

`pword` char(50) NOT NULL,

`birhday` date DEFAULT NULL) ENGINE=InnoDB DEFAULT CHARSET=utf8;

*Добавление записей:*

INSERT INTO `customer` (`id`, `name`, `l\_name`, `adress`, `phone`, `email`, `pword`, `birhday`) VALUES

(1, 'Анастасия', 'Извекова', 'Секретный адрес', '746-50-42', 'adiala@mail.ru', '123123', '2000-05-19'),

(6, 'Алина', 'Климентьева', NULL, NULL, 'klimentjeva.a@mail.ru', '123123', NULL);

**Таблица «good»** – хранит в себе данные о товаре.

*Структура:*

`id` int(11)– идентификатор, первичный ключ, целое число, максимальное количество символов– 11, автоинкремент,

`provider\_id` int(11) – вторичный ключ ссылается на таблицу «provider», поле id, число, максимальное количество символов – 11,

`price` float – цена, вещественное число,

`preview` char(255) – ссылка на изображение, используется для предпросмотра товара на странице со всеми товарами,

`name` char(255) – имя, символьный тип, максимальное количество символов – 255,

`category\_id` int(11) – вторичный ключ ссылается на таблицу «category», поле id, число, максимальное количество символов – 11,

`description` text – описание, тип текст.

*Создание таблицы:*

CREATE TABLE `good` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`provider\_id` int(11) DEFAULT NULL,

`price` float NOT NULL,

`preview` char(255) NOT NULL,

`name` char(255) NOT NULL,

`category\_id` int(11) DEFAULT NULL,

`description` text DEFAULT NULL,

FOREIGN KEY(provider\_id) REFERENCES provider(id) ON UPDATE CASCADE ON DELETE SET NULL,

FOREIGN KEY(category\_id) REFERENCES category(id) ON UPDATE CASCADE ON DELETE SET NULL) ENGINE=InnoDB DEFAULT CHARSET=utf8;

*Добавление записей:*

INSERT INTO `good` (`id`, `provider\_id`, `price`, `preview`, `name`, `category\_id`, `description`) VALUES (1, 1, 1230, 'her.jpg', 'Falcon DB2714', 2, 'Удобны при ношении. Прорезиненный низ подошвы для лучшего сохранения'), (3, 2, 935, 'NewBalance501AthleticShoe4.jpg', 'Athletic Shoe', 2, ' Внутри такой обуви стопы равномерно дышат, чему способствуют продуманные вставки для отвода тепла и влаги.'), (6, 3, 890, 'PUMAxBTSBasketPatentSneakers.jpg', 'Basket Patent', 1, 'Отличный выбор для тех, кто любит качество и удобство'), (7, 1, 1530, 'SharksхNikanorYarmin2.jpg', 'Sharks Nikanor Yarmin', 1, 'Выбор номер 1 среди молодёжи!'), (8, 5, 999, 'Vans\_Classic\_Old\_Skool\_Custom\_Rose\_2.jpg', 'Old Skool Custom Rose', 2, 'Удобная и качественная кастомная модель кед');

**Таблица «good\_material»** – связующая таблица между «good» и «material», используется для привязки нескольких материалов к одному товару.

*Структура:*

`good\_id` int(11) – вторичный ключ ссылается на таблицу « good », поле id, число, максимальное количество символов – 11,

`material\_id` int(11) – вторичный ключ ссылается на таблицу «material», поле id, число, максимальное количество символов – 11.

*Создание таблицы:*

CREATE TABLE `good\_material` (

`good\_id` int(11) DEFAULT NULL,

`material\_id` int(11) DEFAULT NULL,

FOREIGN KEY(good\_id) REFERENCES good(id) ON UPDATE CASCADE ON DELETE SET NULL,

FOREIGN KEY(material\_id) REFERENCES material(id) ON UPDATE CASCADE ON DELETE SET NULL) ENGINE=InnoDB DEFAULT CHARSET=utf8;

*Добавление записей:*

INSERT INTO `good\_material` (`good\_id`, `material\_id`) VALUES (1, 1), (1, 2), (3, 4), (3, 5), (6, 2),

(6, 4), (3, 2), (8, 4), (8, 2);

**Таблица «material»** – хранит в себе данные о материале товара

*Структура:*

`id` int(11)– идентификатор, первичный ключ, целое число, максимальное количество символов– 11, автоинкремент,

`name` char(255) – имя, символьный тип, максимальное количество символов – 255.

*Создание таблицы:*

CREATE TABLE `material` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`name` char(255) NOT NULL) ENGINE=InnoDB DEFAULT CHARSET=utf8;

*Добавление записей:*

INSERT INTO `material` (`id`, `name`) VALUES(1, 'Текстиль'),(2, 'Синтетика'),(3, 'Кожа'),

(4, 'Резина'),(5, 'Кожзам');

**Таблица «photo»** – хранит в себе данные о фотографиях товара.

*Структура:*

`id` int(11) – идентификатор, первичный ключ, целое число, максимальное количество символов– 11, автоинкремент,

`good\_id` int(11)– вторичный ключ ссылается на таблицу good, поле id, число, максимальное количество символов – 11,

`src` char(255) – ссылка на фотографию, символьный тип, максимальное количество символов – 255.

*Создание таблицы:*

CREATE TABLE `photo` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`good\_id` int(11) DEFAULT NULL,

`src` char(255) NOT NULL,

FOREIGN KEY(good\_id) REFERENCES good(id) ON UPDATE CASCADE ON DELETE SET NULL) ENGINE=InnoDB DEFAULT CHARSET=utf8;

*Добавление записей:*

INSERT INTO `photo` (`id`, `good\_id`, `src`) VALUES(1, 1, 'img/OriginalsFalcon1.jpg'),

(2, 1, 'img/OriginalsFalcon2.jpg'), (10, 3, 'img/NewBalance501AthleticShoe1.jpg'),

(11, 3, 'img/NewBalance501AthleticShoe2.jpg'), (14, 6, 'img/PUMAxBTSBasketPatentSneakers1.jpg'),

(15, 6, 'img/PUMAxBTSBasketPatentSneakers2.jpg'), (18, 7, 'img/SharksхNikanorYarmin1.jpg'),

(19, 7, 'img/SharksхNikanorYarmin2.jpg'),

(22, 8, 'img/Vans\_Classic\_Old\_Skool\_Custom\_Rose\_1.jpg'),

(23, 8, 'img/Vans\_Classic\_Old\_Skool\_Custom\_Rose.jpg');

**Таблица «provider»** – хранит в себе данные о фирме товара.

*Структура:*

`id` int(11) – идентификатор, первичный ключ, целое число, максимальное количество символов– 11, автоинкремент,

`name` char(255) – имя, символьный тип, максимальное количество символов – 255.

*Создание таблицы:*

CREATE TABLE `provider` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`name` char(255) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

*Добавление записей:*

INSERT INTO `provider` (`id`, `name`) VALUES (1, 'Adidas'), (2, 'New Balance'), (3, 'Puma'), (5, 'Vans');

**Таблица «purchase»** – хранит в себе данные о заказе.

*Структура:*

`id` int(11) – идентификатор, первичный ключ, целое число, максимальное количество символов– 11, автоинкремент,

`customer\_id` int(11) – вторичный ключ ссылается на таблицу customer, поле id, число, максимальное количество символов – 11,

`good\_id` int(11) – вторичный ключ ссылается на таблицу good, поле id, число, максимальное количество символов – 11,

`order\_date` date – дата заказа, тип данных дата,

`quantity` int(11) – количество заказанных товаров, целое число, максимальное количество символов– 11,

`status\_id` int(11) – вторичный ключ ссылается на таблицу state, поле id, число, максимальное количество символов – 11.

*Создание таблицы:*

CREATE TABLE `purchase` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`customer\_id` int(11) DEFAULT NULL,

`good\_id` int(11) DEFAULT NULL,

`order\_date` date NOT NULL,

`quantity` int(11) NOT NULL,

`status\_id` int(11) DEFAULT NULL,

FOREIGN KEY(customer\_id) REFERENCES customer(id) ON UPDATE CASCADE ON DELETE SET NULL,

FOREIGN KEY(good\_id) REFERENCES avalibility(id) ON UPDATE CASCADE ON DELETE SET NULL) ENGINE=InnoDB DEFAULT CHARSET=utf8;

*Добавление записей:*

INSERT INTO `purchase` (`id`, `customer\_id`, `good\_id`, `order\_date`, `quantity`, `status\_id`) VALUES (1, 1, 1, '2019-03-23', 1, 1);

**Таблица «shoesize»** – хранит в себе данные о размерах товара.

*Структура:*

`id` int(11) – идентификатор, первичный ключ, целое число, максимальное количество символов– 11, автоинкремент,

`size\_name` int(11) – размер, целое число, максимальное количество символов – 11,

*Создание таблицы:*

CREATE TABLE `shoesize` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`size\_name` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

*Добавление записей:*

INSERT INTO `shoesize` (`id`, `size\_name`) VALUES (1, 35), (2, 36), (3, 37), (4, 38), (5, 39), (6, 40), (7, 41), (8, 42), (9, 43);

**Таблица «state»** – хранит в себе данные о состоянии заказа.

*Структура:*

`id` int(11) – идентификатор, первичный ключ, целое число, максимальное количество символов– 11, автоинкремент,

`purchase\_state` char(255) – состояние товара, символьный тип, максимальное количество символов – 255.

*Создание таблицы:*

CREATE TABLE `state` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`purchase\_state` char(255) DEFAULT NULL) ENGINE=InnoDB DEFAULT CHARSET=utf8;

*Добавление записей:*

INSERT INTO `state` (`id`, `purchase\_state`) VALUES (1, 'В корзине'), (2, 'Сделан заказ'), (3, 'Оплачено'), (4, 'Закрыт');

Функция подключения к базе данных. Создает экземпляр PDO для представления подключения к запрашиваемой базе данных с указанием хоста, названием базы данных, именем пользователя и пароля.

function connect (){  
$dsn = "mysql:host=localhost;dbname=shop";  
 return new PDO($dsn, 'root' , '');}

Проверка на наличие coolie с именем 'login'. Если такой существует, то в меню вывести значения из coolie с именем 'name' и 'l\_name' (имя и фамилию), иначе – вывеси меню гостя (пункты меню «Вход», «Регистрация»).

if ($\_COOKIE['login']){  
 echo "<a class=\"nav-link\" href=\"account.php\">";  
 echo $\_COOKIE['name'] . ' ' . $\_COOKIE['l\_name'] . "a>  
 <ul class=\"sub-menu-acc sub-menu\">  
 <li><a href=\"#\">Мои заказы</a></li>  
 <li><a href=\"\">Избранное</a></li>  
 <li><a href=\"#\">Мои данные</a></li>  
 <li><a href=\"#\">Выход</a></li>  
 </ul></li>";}  
 else{  
 echo "<a class=\"nav-link\" href=\"#\">Вход <span class=\"sr-only\">(current)</span></a>  
 <ul class=\" sub-menu\">  
 <form action=\"action.php\" method=\"post\">  
 <li>Здравствуйте!</li>  
 <li><input type=\"email\" placeholder=\"E-mail\" name=\"login\" onmouseover=\"mouselog(event)\" onmouseout=\"mouselog(event)\" required></li>  
 <li><input type=\"password\" placeholder=\"Пароль\" name=\"pass\" onmouseover=\"mouselog(event)\" required></li>  
 <li><input type=\"submit\" value=\"Войти\" class=\"btn fbtn\" onmouseover=\"mouselog(event)\"></li>  
 </form>  
 </ul>  
</li>  
<li class=\"nav-item\"><a class=\"nav-link\" href=\"signin.php\">Регистрация</a></li>";}

Функция проверки совпадения паролей при регистрации, срабатывает при нажатии на кнопку с id = ‘button’, занесение значений input с именами 'username', 'user\_l\_name', 'password', 'password2', 'email' в переменные username, l\_name, p1, p2, email соответственно. Приведение значений p1 и p2 к типу String для проверки пароля. Если пароли не совпадают, то вывод сообщения «Пароли не совпадают» и добавление CSS-свойства классу err значение "display", "block".

$("#button").on('click', function () {  
 var username = $("input[name='username']").val();  
 var l\_name = $("input[name='user\_l\_name']").val();  
 var p1 = $("input[name='password']").val();  
 var p2 = $("input[name='password2']").val();  
 var email = $("input[name='email']").val();  
 if (p1.toString() !== p2.toString()) {  
 $('.err').text("Пароли не совпадают");  
 $('.err').css("display", "block");}

Выполнение AJAX запроса для проверки на существование введённого e-mail при регистрации, для этого передаем значение "email" с именем "email" и название вызываемой функции "funct" с именем "email". Тип отправки – POST, адрес запроса – user.php. В случае успешного выполнения запроса, если ответ сервера равен 1, то вывод сообщения «Пользователь с таким e-mail уже существует» и добавление CSS-свойства классу err значение "display", "block".

$.ajax({

type: "POST",

url: "user.php",

data: {

"funct": "email",

"email": email },

success: function (data) {

if(data == 1){

$('.err').text("Пользователь с таким e-mail уже существует");

$('.err').css("display", "block"); }});

Работа с данными, проверка на наличие пользователя. Подключение файла 'connect.php' для вызова функции connect(), которая выполняет подключение к базе данных. Если передаваемое значение 'funct' равен 'email', то выполнение запроса для подсчёта количества записей в базе данных с помощью PDOStatement:: rowCount(). Если значение не равно нулю, то передаем 1, иначе – 0.

require\_once "connect.php";

switch ($\_POST['funct']){

case 'email': {

$link = connect();

$col = $link->query("Select id from customer where email = '".$\_POST['email']."'");

$i = $col->rowCount();

if ($i != 0){echo 1;}

else {echo 0;}

break;}}

Выполнение AJAX запроса для занесения данных в базу данных, для этого передаем значение "email" с именем "email", username с именем "username", l\_name с именем "l\_name", p1 с именем "password" и название вызываемой функции "reg" с именем "funct". Тип отправки – POST, адрес запроса – user.php. В случае успешного выполнения запроса, то вывод сообщения «Регистрация прошла успешно!» и добавление CSS-свойства классу access значение "display", "block".

$.ajax({

type: "POST",

url: "user.php",

data: {

"funct": "reg",

"email": email,

"username": username,

"l\_name": l\_name,

"password": p1 },

success: function () {

$('.access').text("Регистрация прошла успешно!");

$('.access').css("display", "block");}});

Работа с данными, для занесения значений в базу данных. Подключение файла 'connect.php' для вызова функции connect(), которая выполняет подключение к базе данных. Если передаваемое значение 'funct' равен 'reg', то принимаем переменные 'username', 'l\_name', 'email', 'password' из запроса AJAX и заносим в новые переменные username, user\_l\_name, email, password соответственно. Занесение в переменную sql запроса для добавления в таблицу «customer» (подготовленный запрос) с поименованными псевдопеременными, которые заменяются реальными во время запуска на выполнение. PDO поддерживает подготовленные запросы, которые полезны для защиты приложения от SQL-инъекций: метод prepare выполняет необходимые экранирования. В метод execute данные передаются в виде ассоциативного массива, в котором каждый ключ соответствует имени псевдопеременной, а значение массива — значению, которое требуется подставить в запрос.

switch ($\_POST['funct']){

case 'reg' : {  
 $link = connect();  
 $username = $\_POST['username'];  
 $user\_l\_name = $\_POST['l\_name'];  
 $email = $\_POST['email'];  
 $password = $\_POST['password'];  
 $sql = "INSERT INTO customer (name, l\_name, pword, email) VALUES (:name, :l\_name, :pword, :email) ";  
 $stmt = $link->prepare($sql); //метод prepare выполняет необходимые экранирования.  
 $stmt->execute(array(':name' => $username, ':l\_name' => $user\_l\_name, ':pword' => $password, ':email' => $email));  
 echo 1;  
 break;}}

Работа с данными, вход в аккаунт. Подключение к базе данных. Если существует 'login' и 'pass', которые передаются с помощью AJAX, то заносим эти значения в переменные login и pass. В переменную sql заносим запрос, который выбирает все имена и фамилии зарегистрированных пользователей. Если количество строк запроса равна нулю (если такого пользователя нет), то перенаправляем на страницу signin.php, иначе – запоминаем значения переменной login, с именем "login" на 300 секунд, pass с именем " password" на 300 секунд, name с именем " 'name'" на 300 секунд, l\_name с именем " l\_name " на 300 секунд и перенаправляем на страницу account.php.

$link = mysqli\_connect("localhost", "root", "", "shop");  
if($\_POST['login'] && $\_POST['pass']){  
 $login = $\_POST['login'];  
 $pass = $\_POST['pass'];  
 $sql = "SELECT name, l\_name FROM customer WHERE email = '$login' AND pword = '$pass'";  
 $result = mysqli\_query($link, $sql);  
 if(mysqli\_num\_rows($result) == 0) {  
 header("Location: signin.php");}   
 else {  
 setcookie("login", $login, time()+300);  
 setcookie("password", $pass, time()+300);  
 $res = mysqli\_fetch\_array($result);  
 $name = $res['name'];  
 $l\_name = $res['l\_name'];  
 setcookie('name', $name, time()+300);  
 setcookie('l\_name', $l\_name, time()+300);  
 header("Location: account.php");}}

Работа с данными, вывод товаров. Подключение файла 'connect.php' для вызова функции connect(), которая выполняет подключение к базе данных. Создаем запрос, который выбирает данные (id, ссылку на изображение, название, цену) из таблицы «good». Выбираем режим выборки (PDO::*FETCH\_ASSOC –* возвращает массив, проиндексированный по имени столбца в наборе результатов), вывод карточки, где указываем адрес ссылки (<img src=\"img/" .$row['preview'] ."\"), название товара (<p class=\"card-text ttle\">" .$row['name'] ."</p>), цену (<p class=\"card-text money\">" .$row['price'] ." грн.</p>), ссылку на страницу с карточкой этого товара (<a href='information.php?id=".$row['id']."'>).

require\_once('connect.php');  
$link = connect();  
# создаем запрос  
$STH = $link->query('Select id, preview, name, price from good');  
# выбираем режим выборки  
$STH->setFetchMode(PDO::*FETCH\_ASSOC*);  
# выводим результат  
while($row = $STH->fetch()) {

echo "<div class=\"col-md-4 col-lg-4 col-xl-4 mx-auto mt-3\">

<div class=\"card\">

<img src=\"img/" .$row['preview'] ."\" class=\"card-img-top\" alt=\"...\">

<div class=\"card-body\">

<p class=\"card-text ttle\">" .$row['name'] ."</p>

<p class=\"card-text money\">" .$row['price'] ." грн.</p>

<div class=\"buttons\">

<a href='information.php?id=".$row['id']."'><input type=\"button\" class=\"btn sbtn\" value=\"Подробнее\" </a>

<div class=\"input-image\">

<input type=\"image\" src=\"img/heart.png\" alt=\"\" onclick='checkRegistration()'></div>

</div>

</div>

</div>

</div>";}

Работа с данными, вывод нескольких изображений на карточке товара. Подключение файла 'connect.php' для вызова функции connect(), которая выполняет подключение к базе данных. Выполняем запрос, результатом которого будут ссылки на изображения товара. Выбираем режим выборки (PDO::*FETCH\_ASSOC)*, заносим результат в значение src картинок с классом pics (<img src=\"".$res['src']."\" class='pics' >). Запоминаем последнее значение результирующего массива со ссылками и устанавливаем его значение в src изображения, которое находится в классе main\_image – главная картинка (<img src=\"".$photo\_src."\" >).

require\_once('connect.php');  
$link = connect();  
$res\_photo = $link->query("SELECT DISTINCT src from good, photo where photo.good\_id = good.id and good.id=".$\_GET['id']);  
$res\_photo->setFetchMode(PDO::*FETCH\_ASSOC*);  
while ($res = $res\_photo->fetch()){  
 echo "<div class=\"one\_pic\">  
 <img data-auto-id=\"image\" src=\"".$res['src']."\" alt=\"\" class='pics' onclick='changeBG()' >  
 </div>";  
 $photo\_src = $res['src'];}  
echo "</div> <div class=\"main\_image\">  
 <img data-auto-id=\"image\" src=\"".$photo\_src."\" alt=\"\" id='mImg'>  
 </div>";}

Работа с данными, отображение названия, цены и описания товара. Подключение файла 'connect.php' для вызова функции connect(), которая выполняет подключение к базе данных. Выполняем запрос с выборкой необходимых полей определённого товара и заносим в переменную row. Заносим имя (<h3>".$row['name']."</h3>), описание товара (<p>".$row['description']."</p>), цену (<span id=\"price\">".$row['price']."</span>).

require\_once('connect.php');  
$link = connect();  
$result = $link->query("SELECT name, price, description FROM good where id=".$\_GET['id']);  
$result->setFetchMode(PDO::*FETCH\_ASSOC*);  
$row = $result->fetch();

echo"<div class=\"about-boots\">  
 <h3>".$row['name']."</h3>  
 <p>".$row['description']."</p>  
 <b><span id=\"price\">".$row['price']."</span> гривен</b></div>";

Работа с данными, отображение названия, цены и описания товара. Подключение файла 'connect.php' для вызова функции connect(), которая выполняет подключение к базе данных. Выполняем запрос, который выбирает все размеры определённого товара, которые есть на складе, результат заносим в тег <label> (<label for=\"radio".$i."\"id=\"size\">".$res['size\_name']."</label>)

require\_once('connect.php');  
$link = connect();

$result = $link->query("SELECT size\_name from good, avalibility, shoesize where avalibility.good\_id = good.id and size\_id=shoesize.id and amount>0 and good.id=".$\_GET['id']." ORDER BY shoesize.id ");  
$result->setFetchMode(PDO::*FETCH\_ASSOC*);  
$i=1;

while ($res=$result->fetch()){  
 echo "<div>  
 <input type=\"radio\" name=\"option\" id=\"radio".$i."\" class=\"radio\" value=\"".$res['size\_name']."\"/>  
 <label for=\"radio".$i."\" id =\"size\">".$res['size\_name']."</label>  
</div>";$i++;}

Работа с данными, отображение характеристик товара (артикул, материалы, фирма). Подключение файла 'connect.php' для вызова функции connect(), которая выполняет подключение к базе данных. Выполняем запрос, который отбирает названия материалов и имя фирмы. Занесение значения артикула, всех материалов и имени фирмы.

require\_once('connect.php');  
$link = connect();

$res\_name = $link->query("SELECT provider.name from provider, good where good.provider\_id=provider.id and good.id=".$\_GET['id']);  
$res\_name->setFetchMode(PDO::*FETCH\_ASSOC*);  
$name = $res\_name->fetch();  
$materials = $link->query("SELECT material.name from good\_material, good, material where good.id=good\_material.good\_id AND material.id=good\_material.material\_id and good.id=".$\_GET['id']);  
$materials->setFetchMode(PDO::*FETCH\_ASSOC*);  
echo "<div class=\"more-info text-center\">  
 <h3>Характеристики товара</h3>  
 <p>Артикул: ". $\_GET['id']."</p>  
 <p>Материалы:";  
 while ($result\_materials = $materials->fetch()){  
 echo" ".$result\_materials['name']." ";}  
 echo "</p>  
 <p>Бренд: ".$name['name']."</p></div></div>"

Функция изменения главного изображения при нажатии на изображение с помощью jQuery.

function changeBG(){  
 $(".pics").on('click', function () {  
 $("#mImg").attr("src", $(this).attr("src"));})}

Функция изменения количества для добавления товара в корзину при некорректном вводе числа.

function changeNumber() {  
 $('#product').on('change', function () {  
 if ($('#product').val() < 1){  
 $(this).val(1);}})}

Функция добавления товара в корзину. Занесение значения атрибута 'data-id' в переменную idGood, определение значения выбранного размера – в переменную size, количество – в count\_product, создание идентификатора – idGood+'+'+ size, так как у одного товара может быть несколько размеров. Проверка, выбран ли какой-то размер, если нет, то вывод сообщения "Размер не выбран!", иначе – выполнение AJAX запроса для проверки количества товара выбранного размера на складе. При успешном результате получаем данные корзины или создаем новый объект, берём значения name, idGood, size, count\_product. Проверка на наличие такого товара в корзине, если есть, то проверяем, будет ли количество на складе больше либо равно сумме товара (его количество в корзине и то количество, которое ввёл), если результат True, то заносим в корзину новое количество товаров и выводим сообщение "Товар добавлен", если результат False, то выводим сообщение "Товар уже есть, cлишком большое количество!". Если товара в корзине нет, то добавляем в объект и выводим сообщение "Товар добавлен", если введённое количество меньше либо равно количеству на складе, иначе – сообщение с указанием максимально допустимого количества. Обновляем данные в LocalStorage.

function addToBasket() {  
 var idGood = $(this).attr('data-id'); var size = $(".radio:checked").val();  
 var count\_product = Number($('#product').val()); var name = idGood+'+'+ size;  
 if(size!== undefined) {$.ajax({  
 type: "POST",url: "core.php",  
 data : {"action": "addToBasket","good\_id": idGood,"count\_product": count\_product,"size": size,},dataType: "JSON",  
 success: function(data){  
 var cartData = getCartData() || {}, itemName = name; itemId = idGood;  
 itemSize = size; itemCount = count\_product;

if (cartData.hasOwnProperty(itemName)) {   
 a = cartData[itemName][2] + Number(itemCount);  
 if (data.amount>= a){  
 alert("Товар добавлен");  
 cartData[itemName][2] += Number(itemCount);}  
 else {  
 alert("Товар уже есть, cлишком большое количество!");}}  
 else { if (data.amount>= Number(itemCount)){  
 cartData[itemName] = [itemId, itemSize, itemCount];alert("Товар добавлен");}  
 else {alert("Слишком большое количество! Максимально - " + data.amount);}}  
 setCartData(cartData);} });} else{ alert("Размер не выбран!");}}

Функция, которая записывает данные в LocalStorage

function setCartData(o){  
 localStorage.setItem('cart', JSON.stringify(o));  
 return false;}

Функция, которая берёт данные из LocalStorage

function getCartData() {

return JSON.parse(localStorage.getItem('cart'));}

Работа с данными, добавление товара в корзину. Выполнение запроса, который выбирает количество товара на складе и название определённого товара и передает результат обратно клиентской части.

function addToBasket(){  
 $link = connect();  
 $size = $\_POST['size'];  
 $good\_id = $\_POST['good\_id'];  
 $result = $link->query("SELECT DISTINCT amount, good.name from avalibility, good, shoesize where avalibility.good\_id = good.id and shoesize.id = avalibility.size\_id and shoesize.size\_name='$size' and good.id = '$good\_id' ");  
 $result->setFetchMode(PDO::*FETCH\_ASSOC*);  
 $row = $result->fetch();  
 echo json\_encode($row);}

Работа с данными, отображение корзины. Выборка полей good\_id, name, price, src from good, photo для отображения товара в корзине, занесение результата в массив объектов, возвращаем его обратно клиентской части.

function showBasket(){

$link = connect();

$result = $link->query("SELECT good\_id, name, price, src from good, photo where good.id = photo.good\_id GROUP BY good.name");

$result->setFetchMode(PDO::FETCH\_ASSOC);

$res = array();

$i = 1;

while($row = $result->fetch()){

$res[$i]['id'] = $row['good\_id'];

$res[$i]['name'] = $row['name'];

$res[$i]['price'] = $row['price'];

$res[$i]['src'] = $row['src'];

$i++;}echo json\_encode($res);}

Функция проверки на пустоту корзины. Если существует объект, то возвращает True, иначе – False

function isEmpty(cart){

for (var key in cart)

if (cart.hasOwnProperty(key)) return true;

return false;}

Функция удаления из корзины по идентификатору id, после этого вызываем функцию, которая записывает данные в LocalStorage, вызов функции отображения корзины.

function deleteItem(id){

delete cart[id];

setCartData();

showBasket();}

Функция добавления количества товара на один при нажатии на кнопку «+» в корзине (по определённому идентификатору). Вызов функции для записи обновлённых данных. Вывод корзины.

function plusGoods(good) {

id = $(good).data('id');

cart[id][2]++;

setCartData();

showBasket();}

Функция определения количества товаров в корзине. Если товар был добавлен ранее, то расшифровываем и записываем в переменную cart. Вызов функции отображения корзины.

function loadCart() {

if (localStorage.getItem('cart')){

cart = getCartData();}

showBasket();}

Функция уменьшения количества товара на один при нажатии на кнопку «–» в корзине (по определённому идентификатору). Если количество товара была равна 1, то вызываем функцию удаления из корзины, иначе – уменьшаем на 1. Вызов функции для записи обновлённых данных. Вывод корзины.

function minusGoods(good) {

id = $(good).data('id');

if (cart[id][2] == 1){

deleteItem(id);}

else{

cart[id][2]--; }

setCartData();

showBasket();}

ОТОБРАЖЕНИЕ КОРЗИНЫ

function showBasket() {  
 if (!isEmpty(cart)) {  
 varcssValues = {"font-size": "30px", "padding-top": "22%"};  
 $('.all-cards').html('Корзина пуста!').height(55 + 'vh').css(cssValues);  
 $('#num-of-goods').text(" 0");  
 $('#oplata').hide();  
 $('.allCost').hide();  
 } else {  
 $.ajax({  
 type: "POST",  
 url: "core.php",  
 data: {  
 "action": "showBasket"  
 },  
 dataType: "json",  
 success: function (data) { // id, name, price, src  
 console.log(data);  
 var idCard = JSON.parse(localStorage.getItem('cart'));  
  
 var num = 0;  
 var out = '';  
 var allCount = 0;  
 $.each(data, function (index, arr) { // id  
 for(var k in cart) {  
 var idProduct = k.split('+')[0];  
 var size = k.split('+')[1];  
 $.each (arr, function (idResultQuery, result) { //arr  
  
 if (result == parseInt(idProduct) && idResultQuery == "id" ){  
 out += `<div class="row" style="padding: 3%; ">`;  
 out += `<div class="col-md-3 col-lg-3 col-xl-3 mx-auto mt-3">`;  
 out += `<img src="${data[index].src}" alt="boots" class="preview" style="width: 200px">`;  
 out += `</div>`;  
 out += `<div class="col-md-4 col-lg-4 col-xl-4 mx-auto mt-3">`;  
 out += `<div class="descr">`;  
 out += `<h4 class="title">${data[index].name}</h4>`;  
 out += `<p class="inf">${idCard[k][1] + " размер"}</p>`;  
 out += `<a href="" onclick='deleteItem(this.id)' id = "${k}">Удалить</a>`;  
 out += `</div>`;  
 out += `</div>`;  
 out += `<div class="col-md-3 col-lg-3 col-xl-3 mx-auto mt-3">`;  
 // <!--ввод кол-ва-->  
 out += `<div class="quantity-block">`;  
 out += `<button class="quantity-arrow-minus" data-id = "${k}" onclick="minusGoods(this);"> - </button>`;  
 out += `<input class="quantity-num" type="number" value="${idCard[k][2]}" readonly='readonly'/>`;  
 out += `<button class="quantity-arrow-plus" data-id = "${k}" onclick="plusGoods(this);"> + </button>`;  
 out += `</div>`;  
 out += `</div>`;  
 out += `<div class="col-md-2 col-lg-2 col-xl-2 mx-auto mt-3">`;  
 out += `<p class="cost"><span class="cost-value">${data[index].price}</span> грн.</p>`;  
 out += `</div>`;  
 out += `</div>`;  
 allCount += parseInt(`${data[index].price}`) \* parseInt(`${idCard[k][2]}`);  
 num++;  
 }  
 });  
 }  
 });  
 $('#num-of-goods').text(num);  
 $('.all-cards').html(out);  
 $('#allCost').text(allCount);});}}

Тест – регистрация нового пользователя, успешная попытка. Проверка Title страницы с названием "Регистрация", если не совпадает – остановка теста из-за ошибки, иначе – вводим данные в поля input (email – нет в базе данных, пароли совпадают), после чего нажимаем на кнопку с идентификатором "button", находим элемент, используя xpath, заносим его значение и сравниваем с ожидаемым, если значение не совпадает с ожидаемым – остановка теста из-за ошибки.

Assert.assertEquals(driver.getTitle(), "Регистрация");

driver.findElement(By.name("username")).sendKeys("Lia");

driver.findElement(By.name("user\_l\_name")).sendKeys("Kamper");

driver.findElement(By.name("email")).sendKeys("lia@mail.ru");

driver.findElement(By.name("password")).sendKeys("qwe");

driver.findElement(By.name("password2")).sendKeys("qwe");

driver.findElement(By.id("button")).click();

String text = driver.findElement(By.xpath("//div[@class='access']")).getText();

Assert.assertEquals(text, "Регистрация прошла успешно!");

Тест – регистрация нового пользователя, неудачная попытка – пароли не совпадают. Вводим данные в поля input (email – нет в базе данных, пароли не совпадают), после чего нажимаем на кнопку с идентификатором "button", находим элемент, используя xpath, заносим его значение и сравниваем с ожидаемым, если значение не совпадает с ожидаемым – остановка теста из-за ошибки.

driver.findElement(By.name("username")).sendKeys("Ирина");

driver.findElement(By.name("user\_l\_name")).sendKeys("Романова");

driver.findElement(By.name("email")).sendKeys("us@mail.ru");

driver.findElement(By.name("password")).sendKeys("aas");

driver.findElement(By.name("password2")).sendKeys("as");

driver.findElement(By.id("button")).click();

String text = driver.findElement(By.xpath("//div[@class='err']")).getText();

Assert.assertEquals(text, "Пароли не совпадают");

Тест – регистрация нового пользователя, неудачная попытка – пользователь уже существует. Вводим данные в поля input (email – есть в базе данных, пароли совпадают), после чего нажимаем на кнопку с идентификатором "button", находим элемент, используя xpath, заносим его значение и сравниваем с ожидаемым, если значение не совпадает с ожидаемым – остановка теста из-за ошибки.

driver.findElement(By.name("username")).sendKeys("Ирина");

driver.findElement(By.name("user\_l\_name")).sendKeys("Романова");

driver.findElement(By.name("email")).sendKeys("adiala@mail.ru");

driver.findElement(By.name("password")).sendKeys("aas");

driver.findElement(By.name("password2")).sendKeys("aas");

driver.findElement(By.id("button")).click();

String text = driver.findElement(By.xpath("//div[@class='err']")).getText();

Assert.assertEquals(text, "Пользователь с таким email уже существует");

Тест – успешный вход. Переходим на главную страницу, наводим на элемент с идентификатором "login", находим элементы по имени "login" и "pass", вводим данные зарегистрированного пользователя (email и пароль), подтверждаем операцию нажатием на кнопку с классом "signin". Сравниваем Title страницы с ожидаемым (Аккаунт), если не совпадает, то остановка теста из-за ошибки.

driver.get("http://softshop.ua/index.php");

Actions actions = new Actions(driver);

actions.moveToElement(driver.findElement(By.id("login"))).build().perform();

driver.findElementByName("login").sendKeys("adiala@mail.ru");

driver.findElementByName("pass").sendKeys("123123");

driver.findElementByClassName("signin").submit();

Assert.assertEquals("Аккаунт", driver.getTitle());

Тест – не успешный вход. Переходим на главную страницу, наводим на элемент с идентификатором "login", находим элементы по имени "login" и "pass", вводим данные (email и неверный пароль), подтверждаем операцию нажатием на кнопку с классом "signin". Сравниваем Title страницы с ожидаемым (Регистрация), если не совпадает, то остановка теста из-за ошибки.

driver.get("http://softshop.ua/index.php");

Actions actions = new Actions(driver);

actions.moveToElement(driver.findElement(By.id("login"))).build().perform();

driver.findElementByName("login").sendKeys("adiala@mail.ru");

driver.findElementByName("pass").sendKeys("1223");

driver.findElementByClassName("signin").submit();

Assert.assertEquals("Регистрация", driver.getTitle());

Тест – выбор и добавление товара в корзину. Находим второй элемент из списка карточек товара, запоминаем название. Находим ссылку для перехода страницу с карточкой этого товара, проверяем заранее сохранённое значение с действительным, если не совпадает, то остановка теста из-за ошибки, иначе – находим элементы с классом "pics" и заносим в список, чтобы прокликать на каждое изображение. Выбор размера, заполнение желаемого количества, подтверждение добавления в корзину – клик по кнопке, которая содержит в названии «Добавить», возврат на страницу назад и переход в корзину.

WebElement card = driver.findElement(By.xpath("(//div[contains(@class, 'card-body')])[2]"));

actions.moveToElement(card).build().perform();

String name = card.findElement(By.className("ttle")).getText();

card.findElement(By.cssSelector("a[href\*=\"information\"] ")).click();

String ttle = driver.findElement(By.xpath("//div[contains(@class, 'about-boots')]/h3")).getText();

Assert.assertEquals(name, ttle);

List <WebElement> images = driver.findElements(By.className("pics"));

for (int i = 0; i<images.size(); i++) {images.get(i).click();}

driver.findElement(By.tagName("label")).click();

driver.findElementByCssSelector("input[type='number']").clear();

driver.findElementByCssSelector("input[type='number']").sendKeys("2");

driver.findElementByCssSelector("input[value^='Добавить']").click();

driver.navigate().back();

driver.findElement(By.cssSelector("a[href=\"basket.php\"]")).click();

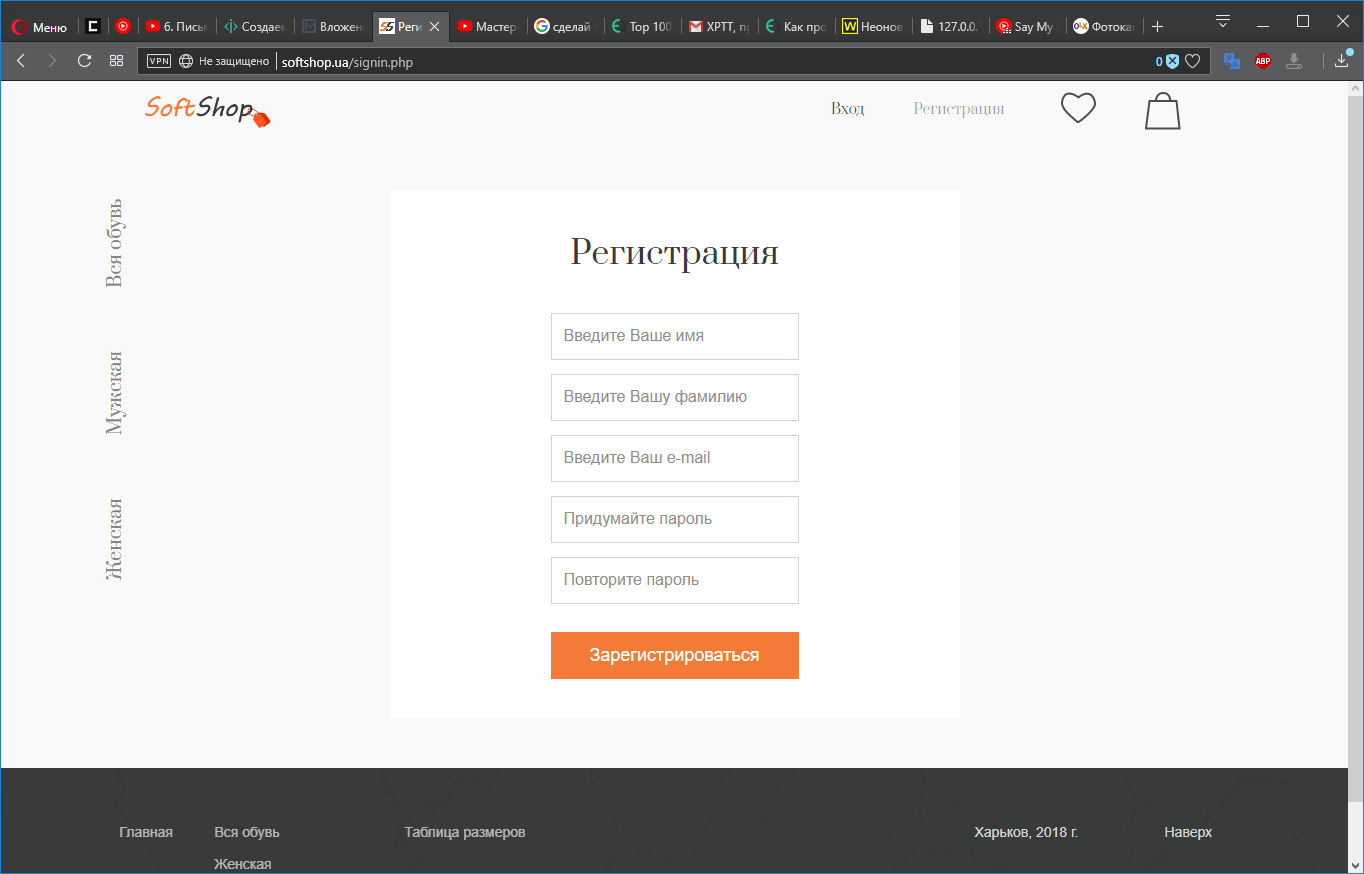
# Раздел IV Инструкция пользователя

Регистрация

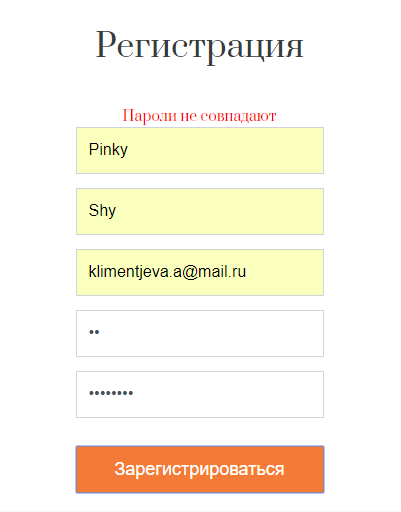
Для перехода к странице с регистрацией нового пользователя необходимо в меню (хедере) нажать на пункт «Регистрация»



После перехода, появляется страница с формой регистрации



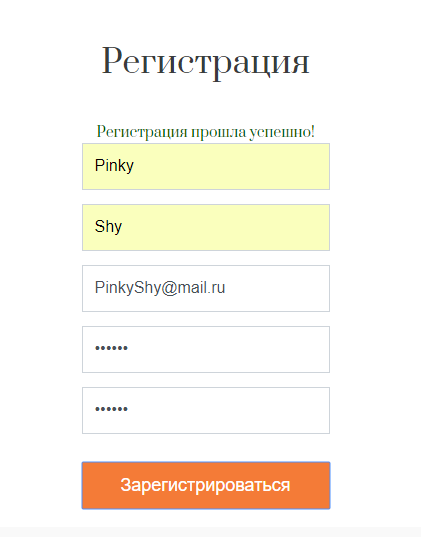
Заполняем поля, вводим имя, фамилию, е-mail, пароль, повтор пароля. Также предусмотрено, если пароли совпадать не будут, то появляется текстовое поле с надписью: «Пароли не совпадают».



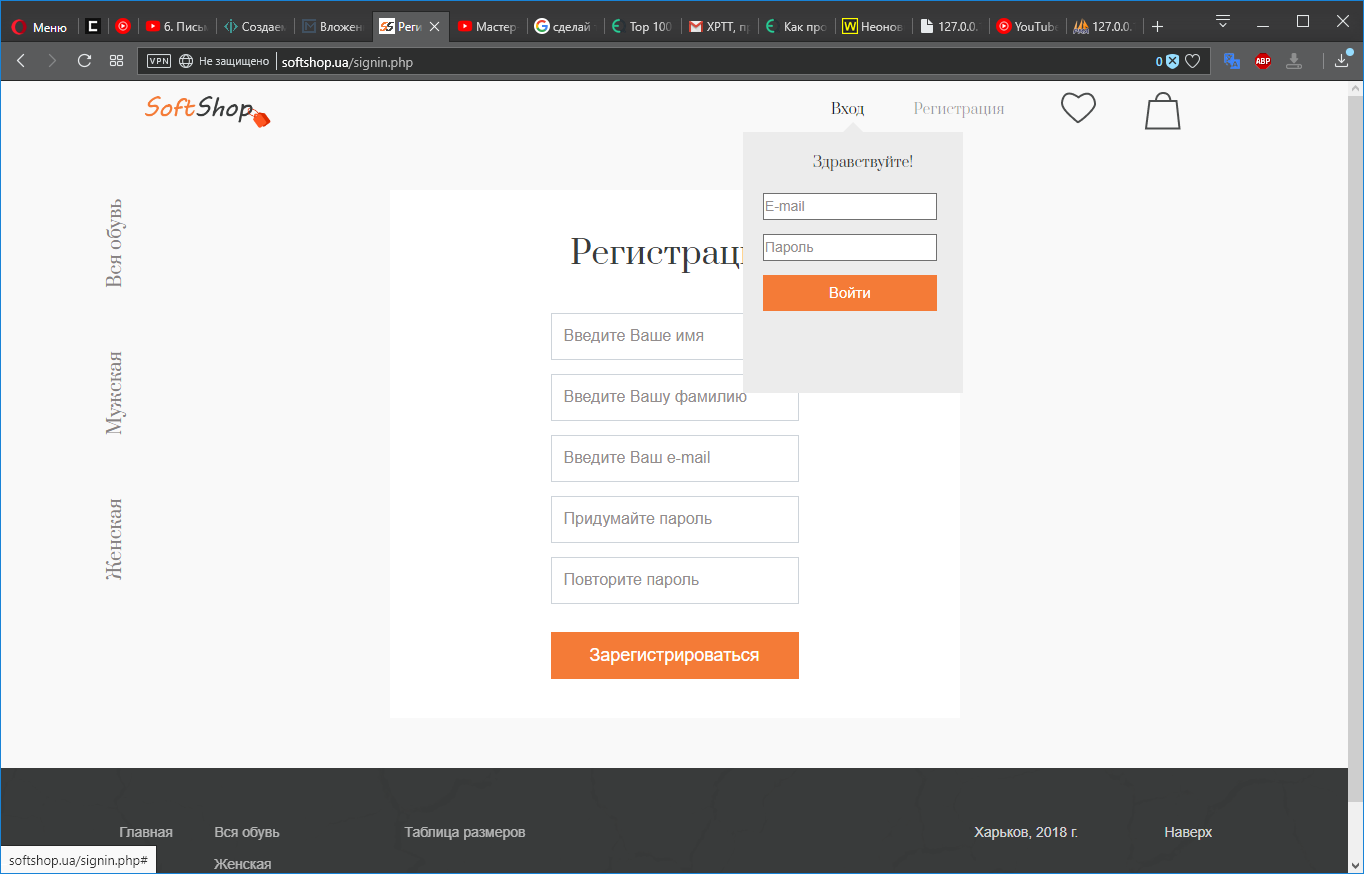
Проверка на наличие такого пользователя, если существует, то выводится текстовое поле с надписью: «Пользователь с таким e-mal уже существует».



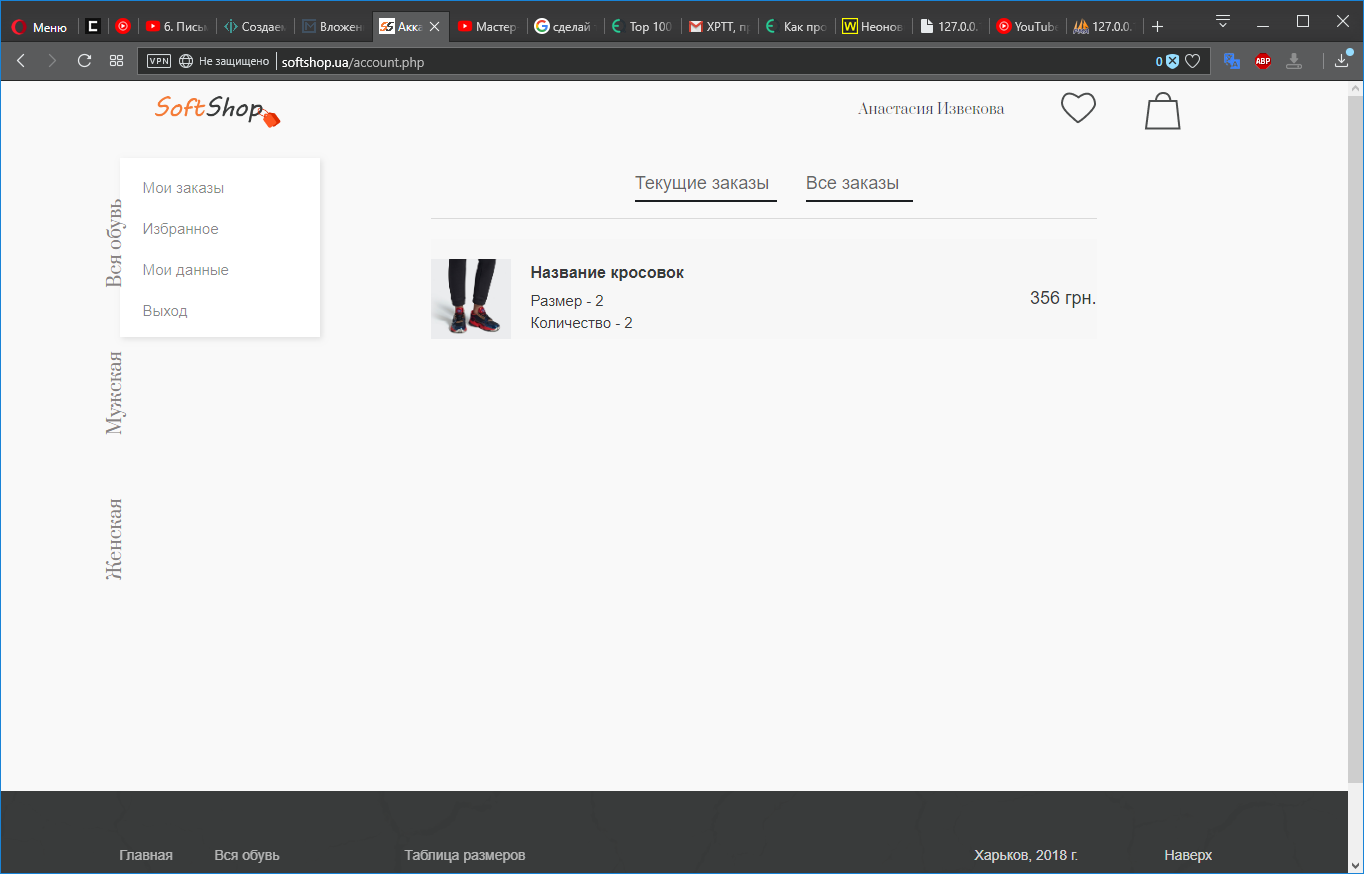
Проверка на наличие такого пользователя, если такого нет, то выводится текстовое поле с надписью: «Регистрация прошла успешно!»



Для входа в аккаунт навести на надпись «Вход», появляется форма для заполнения полей «E-mail» и «Пароль»



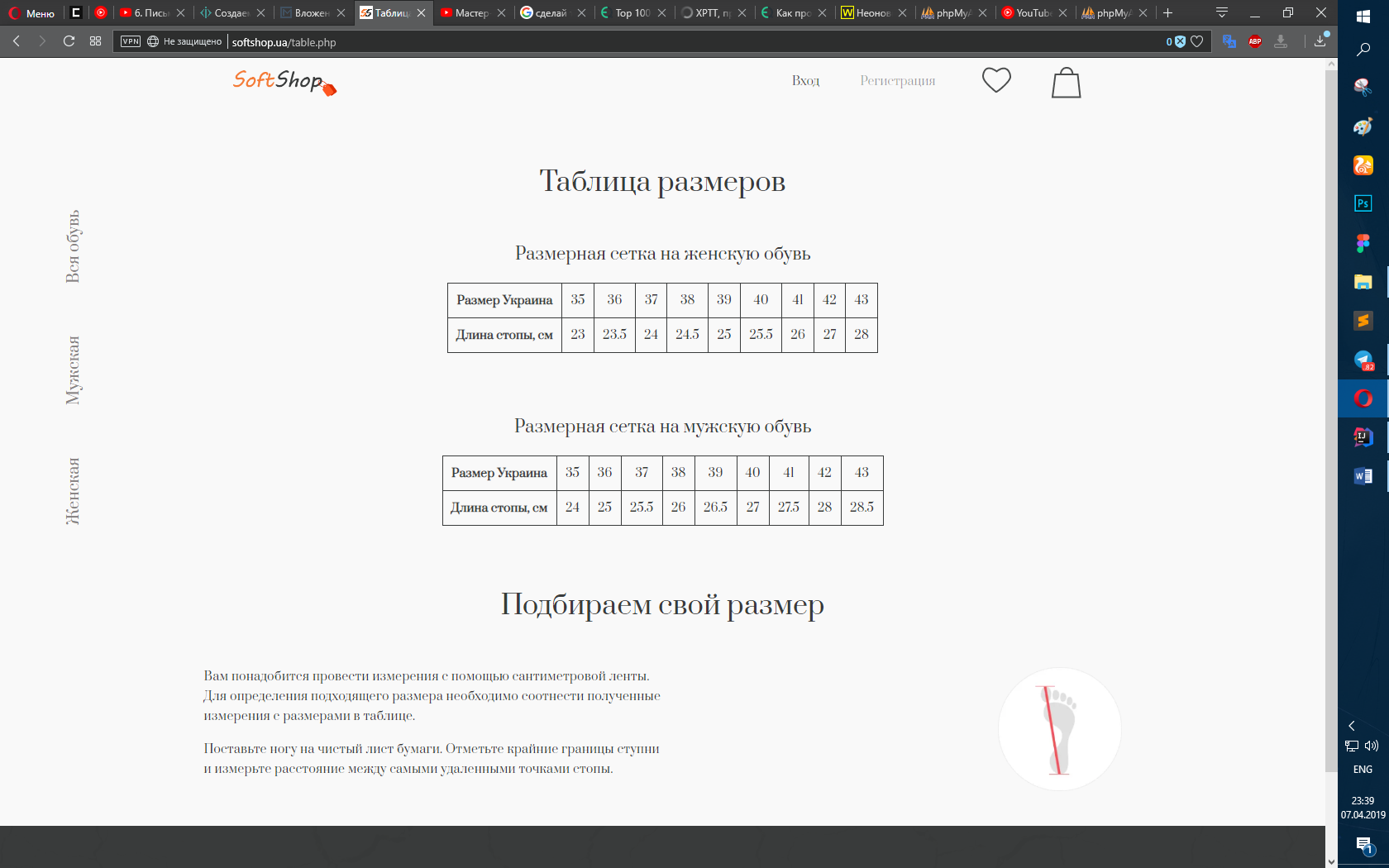
При неправильном заполнении (несуществующий пользователь), после нажатия кнопки с надписью: «Войти», открывается страница регистрации, а при правильном – входим в аккаунт, в меню (хедере) отображается имя и фамилия пользователя.



Переход к странице «Таблица размеров»

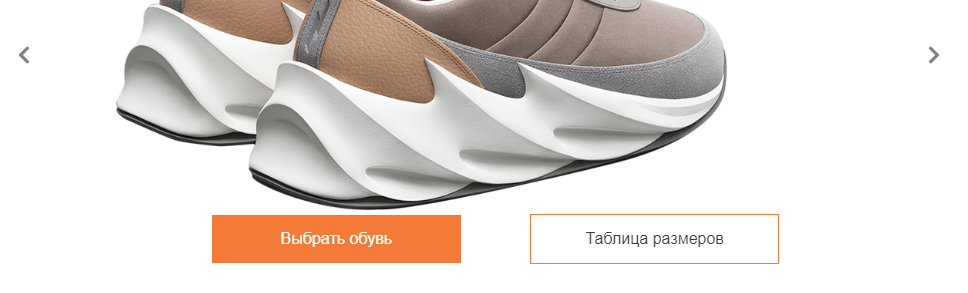
При нажатии на кнопку в главном меню с названием «Таблица размеров», в футере – нажатие ссылки «Таблица размеров», либо на странице с товаром нажать на ссылку «Таблица размеров», откроется нужная страница (во втором случае в новом окне)

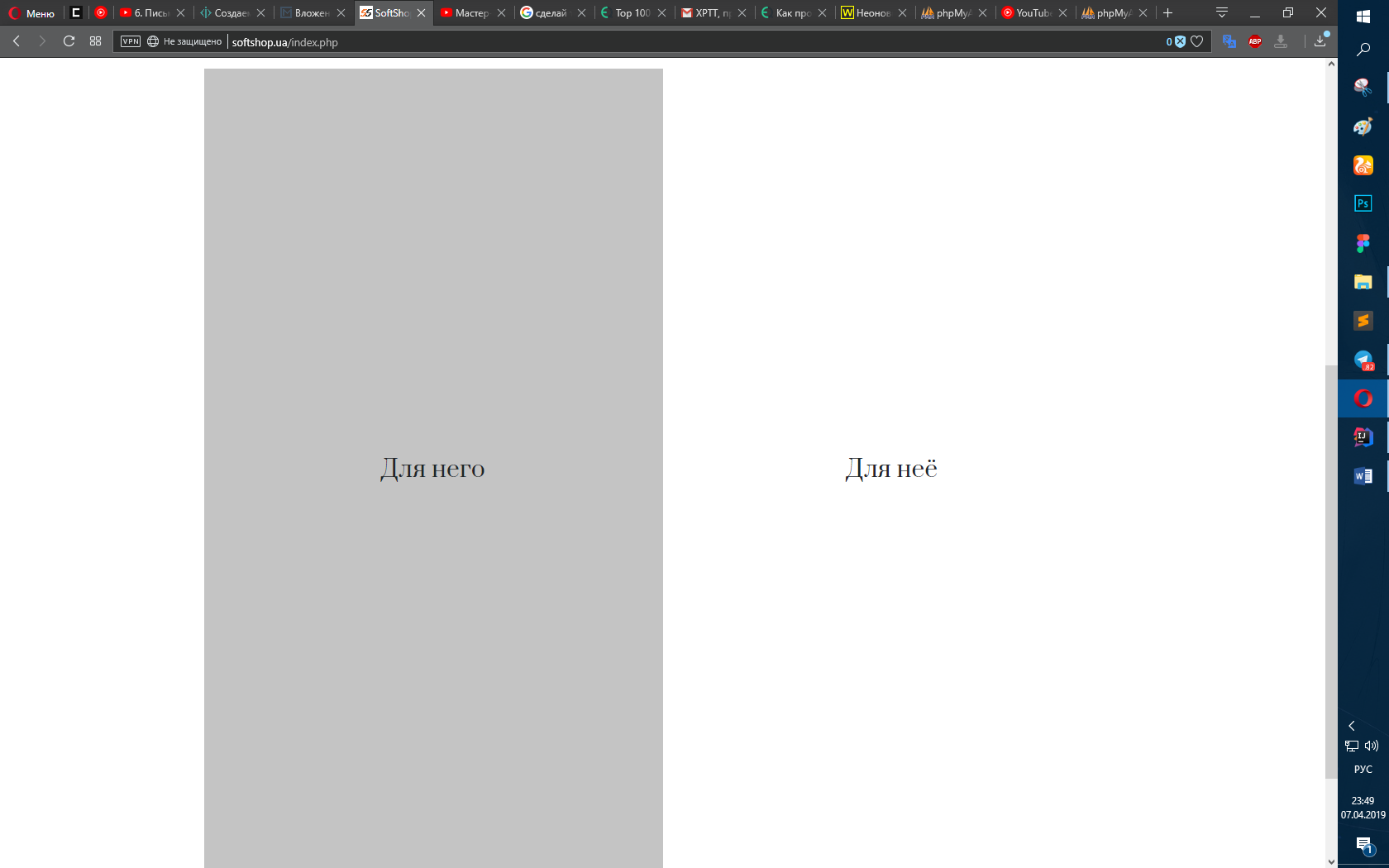
, 



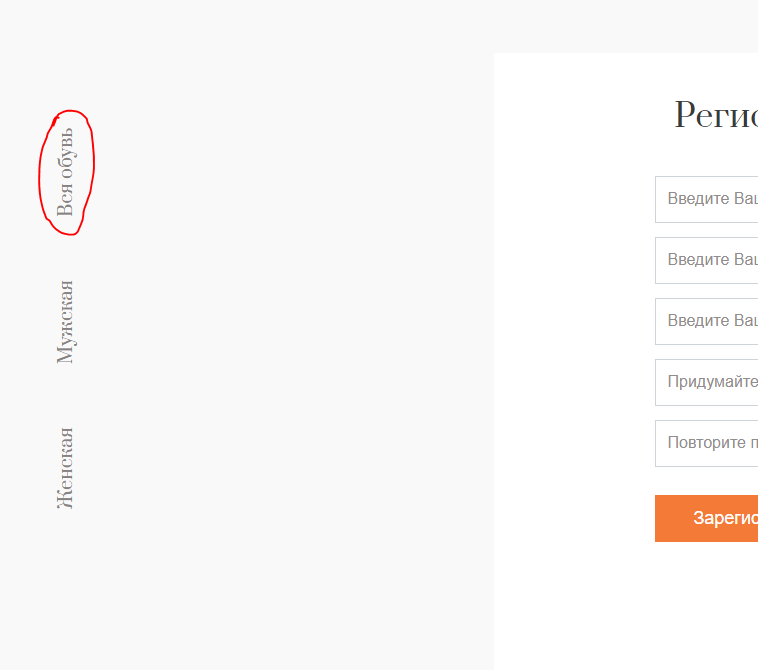
Переход на страницу со всеми товарами

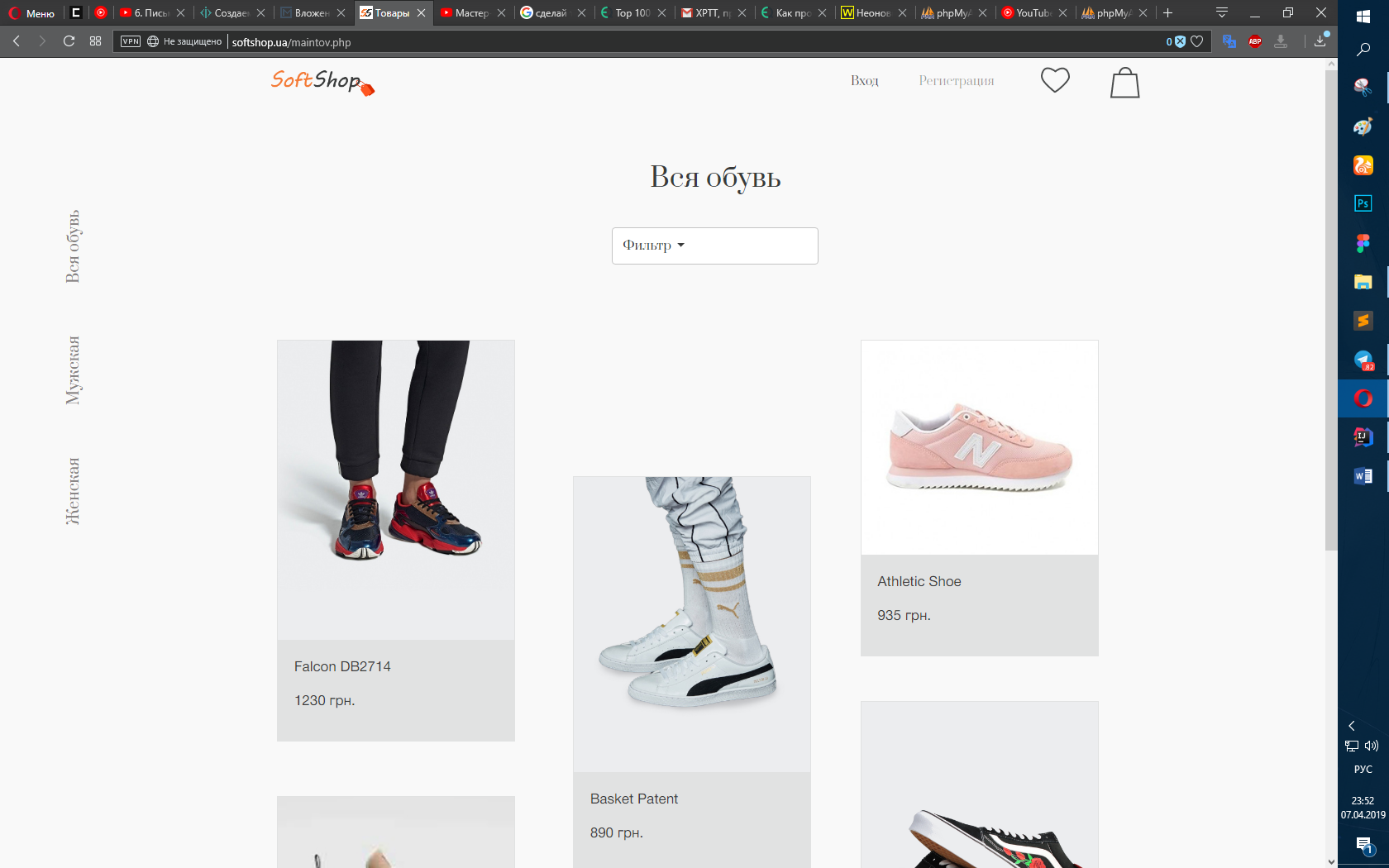
При нажатии на кнопку в главном меню с названием «Выбрать обувь», нажатие на блок с надписью: «Для него», нажатие на блок с надписью: «Для неё», в футере – при нажатии на ссылку с названием «Вся обувь», либо на любой другой странице (кроме главной) при нажатии на боковое меню с текстом «Вся обувь» открывается страница со всеми товарами.





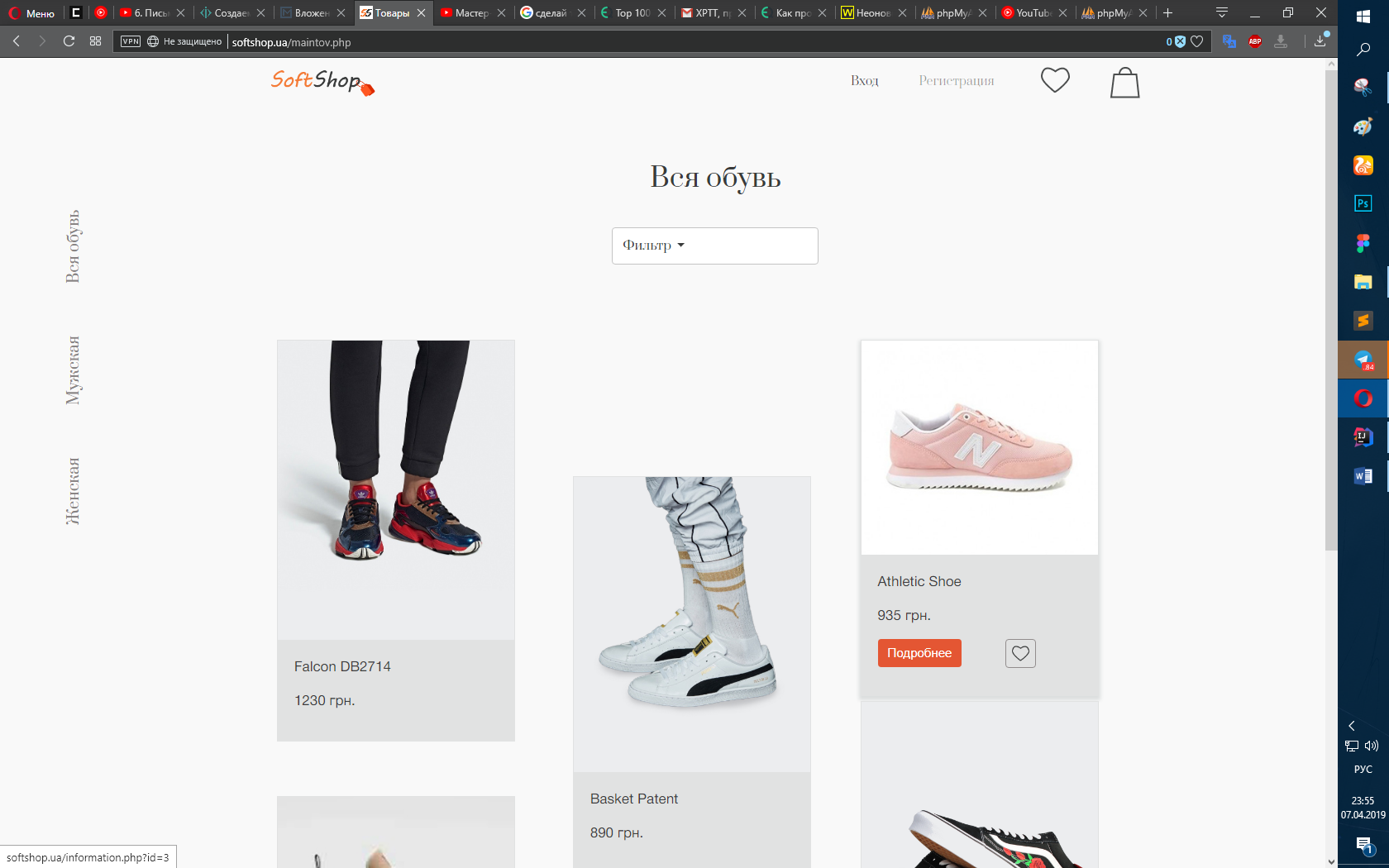




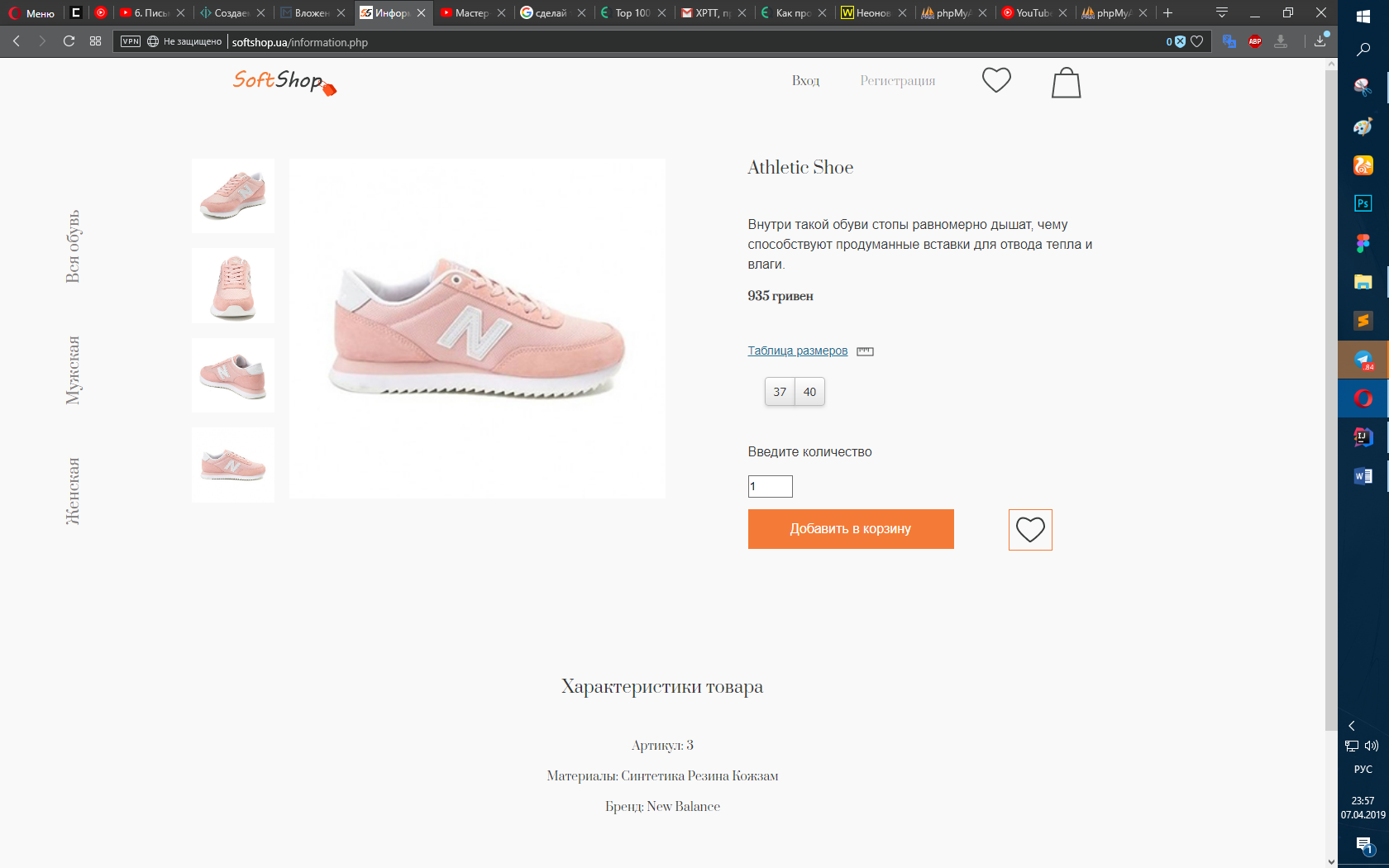


Выбор товара

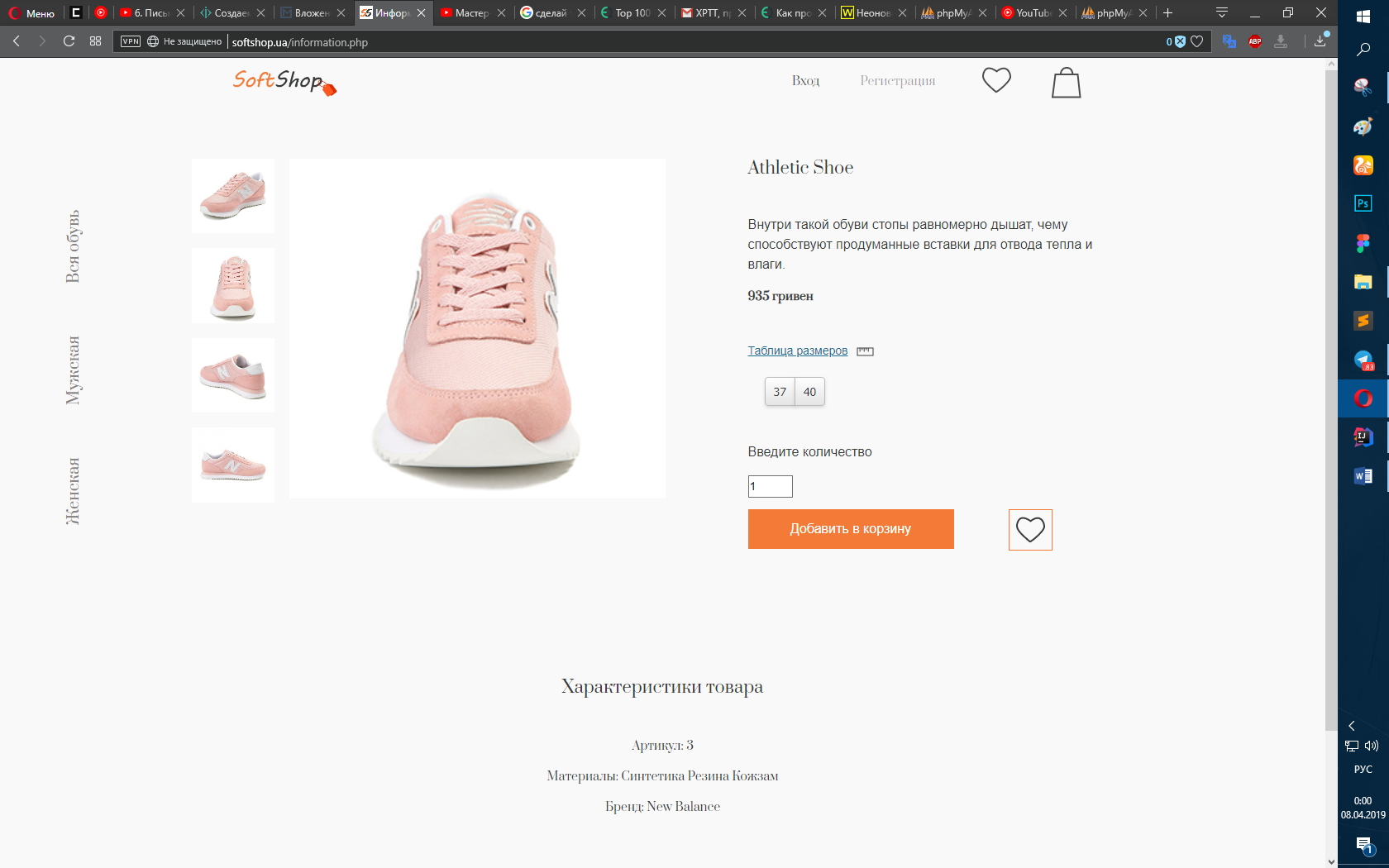
Из страницы со всеми товарами при наведении на карточку товара отображается кнопка с названием «Подробнее»



При нажатии на кнопку с надписью: «Подробнее» отображается страница с информацией об этом товаре (название, описание, цена, возможные размеры, поле для ввода количества и характеристики товара, а именно – артикул, материалы и бренд).

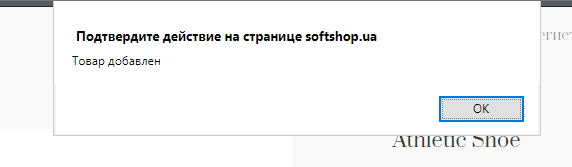


При нажатии на изображения, которые находятся слева от основного, изменяется главная картинка.

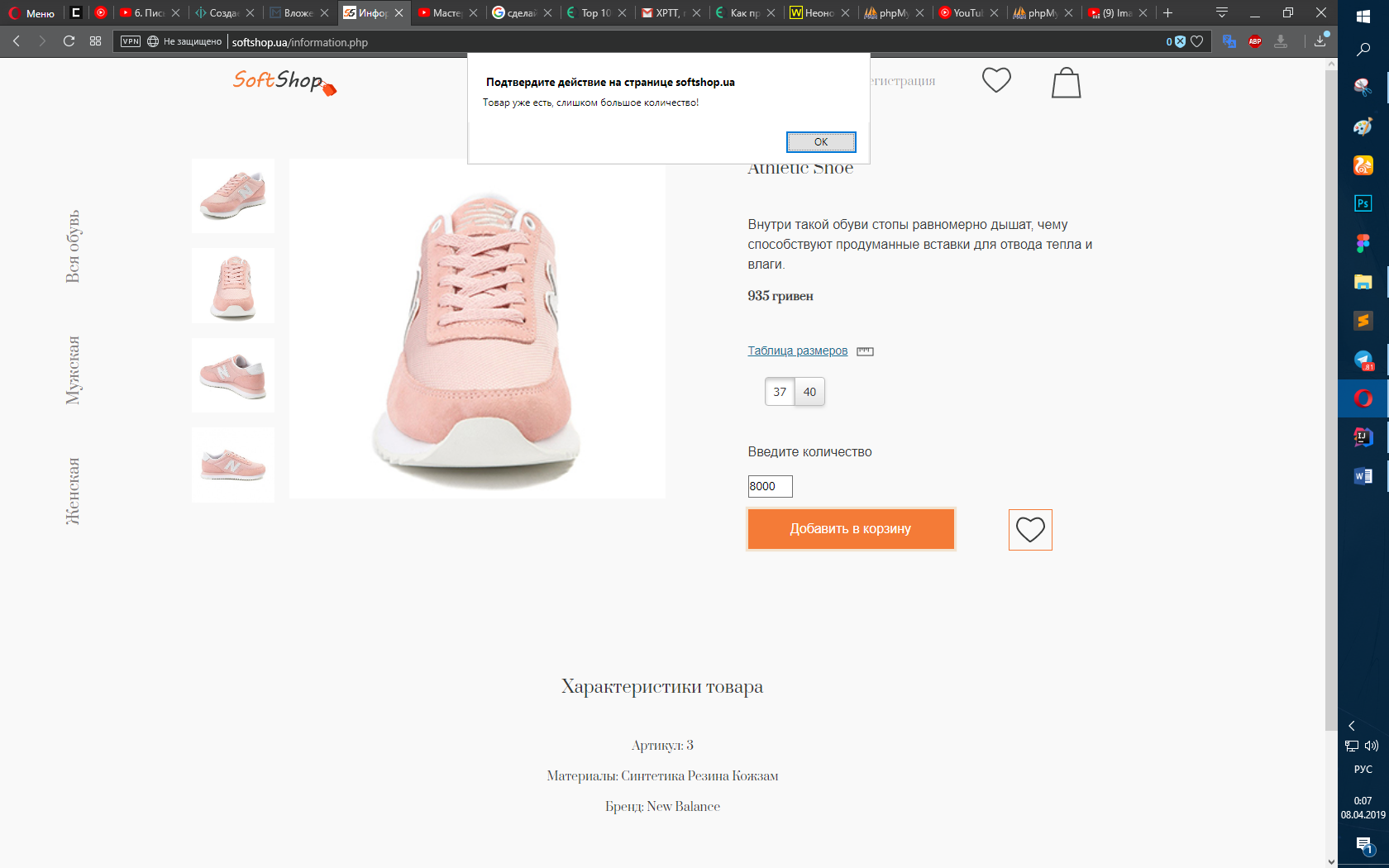


Добавление в корзину

Для добавления товара в корзину необходимо выбрать размер и ввести (или оставить равным 1) количество, при успешном добавлении выдается сообщение о том, что товар добавлен.



Если вести количество больше, чем есть на складе, и этот товар уже есть в корзине, то выдается сообщение «Товар уже есть, слишком большое количество!»

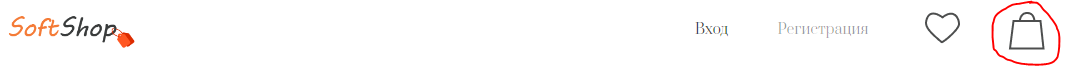


Если вести количество больше, чем есть на складе, и этого товара нет в корзине, то выдается сообщение с указанием максимально возможного количества «Слишком большое количество! Максимально – (максимальное количество)».

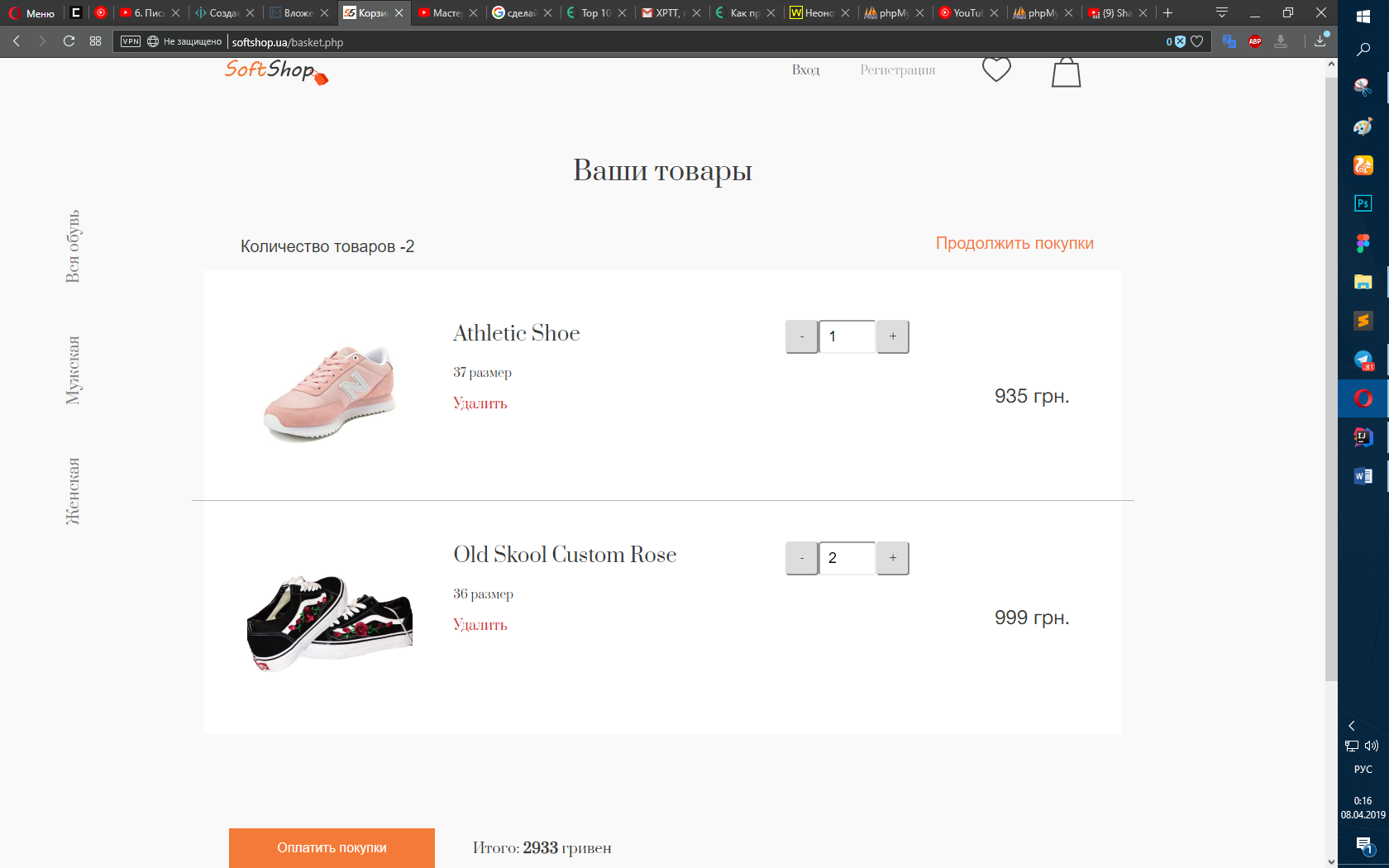


Переход в корзину

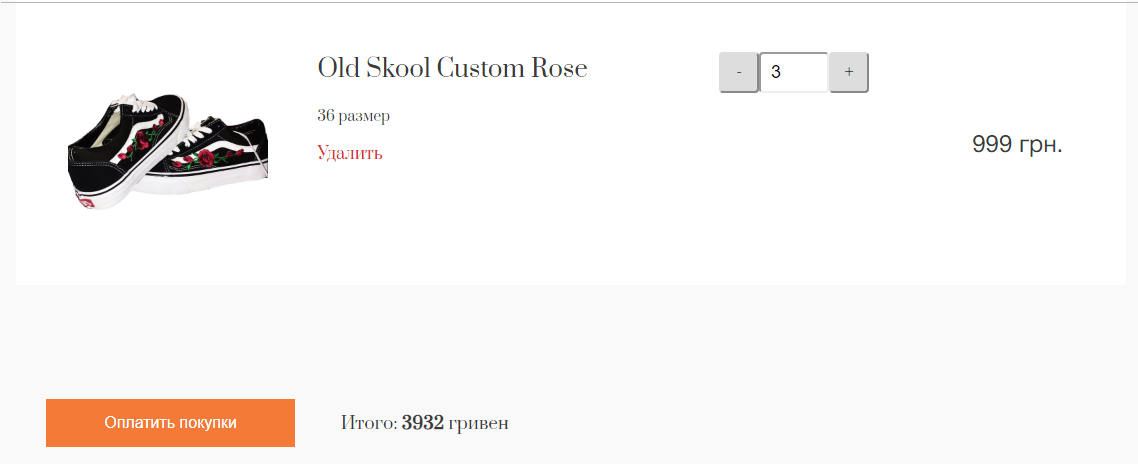
Чтобы посмотреть список товаров, которые находятся в корзине, необходимо в меню (хедере) нажать на иконку с изображением корзины.



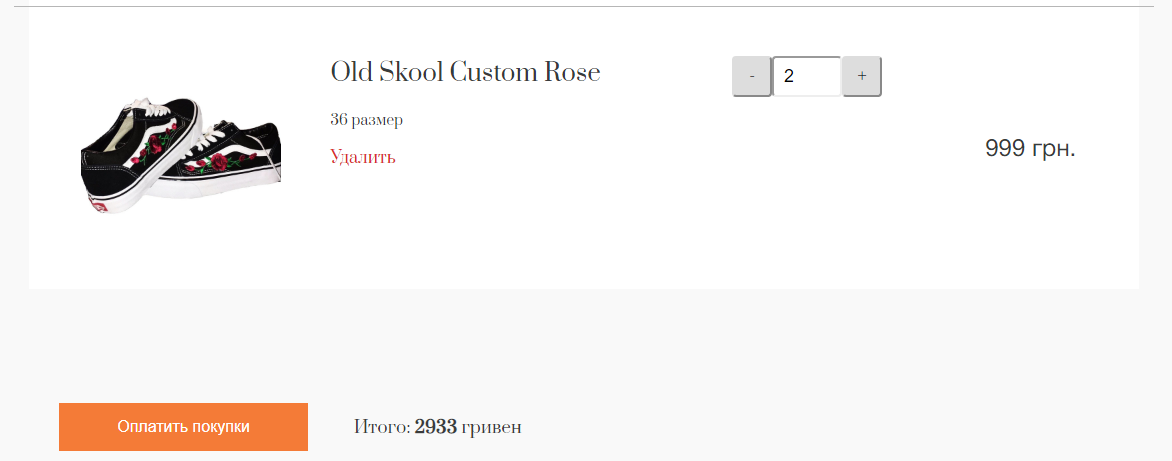
После перехода открывается страница со списком товаров в корзине и информацией о них (название, размер, количество, цена). Также отображается информация об общем количестве товаров в корзине, ссылка «Продолжить покупки», при нажатии на которую можно вернуться на страницу со всеми товарами, итоговый счёт за все товары возле кнопки «Оплатить покупки».



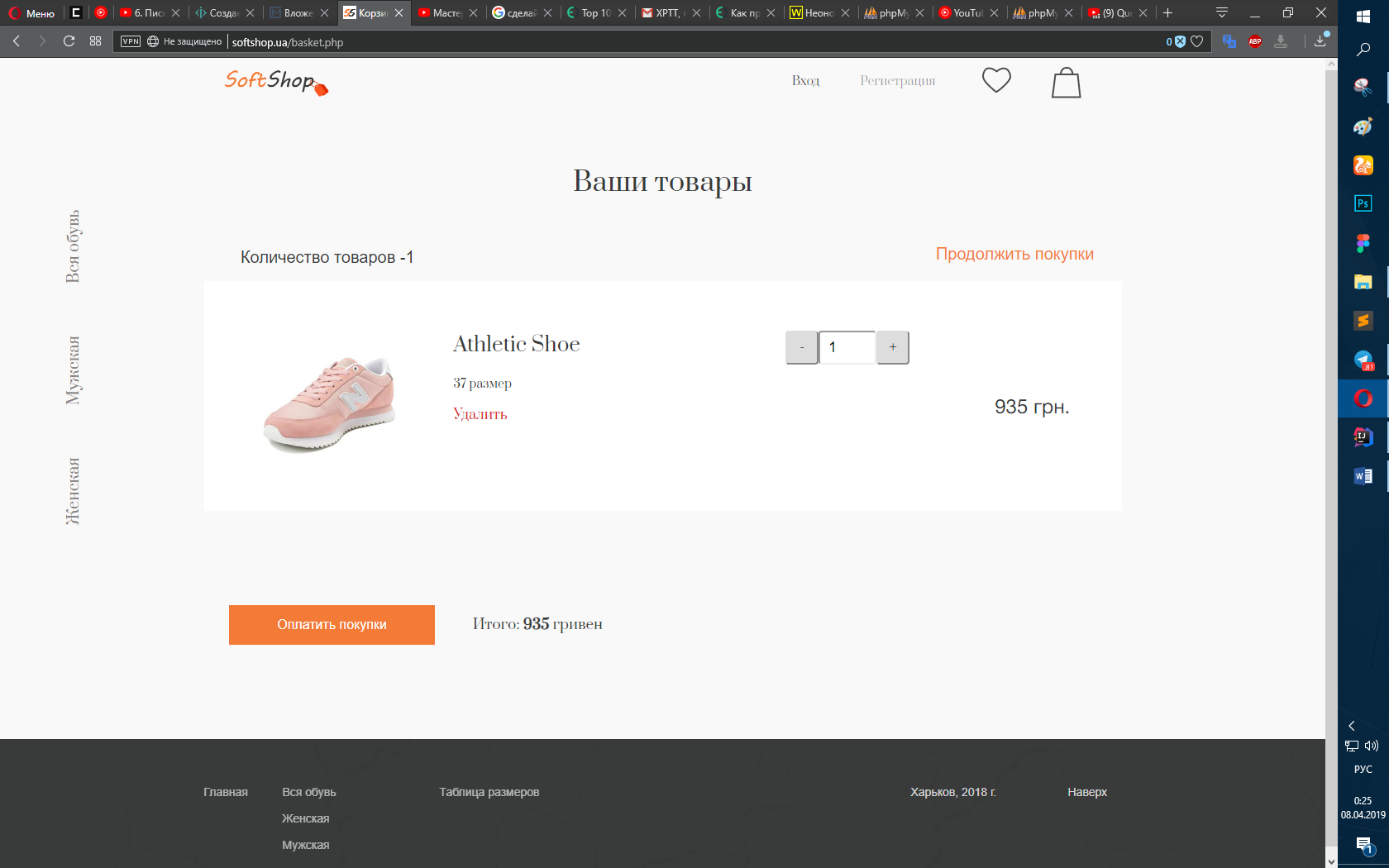
Возможно регулировать количество товаров, при нажатии на кнопку «+», увеличивается количество товара на 1



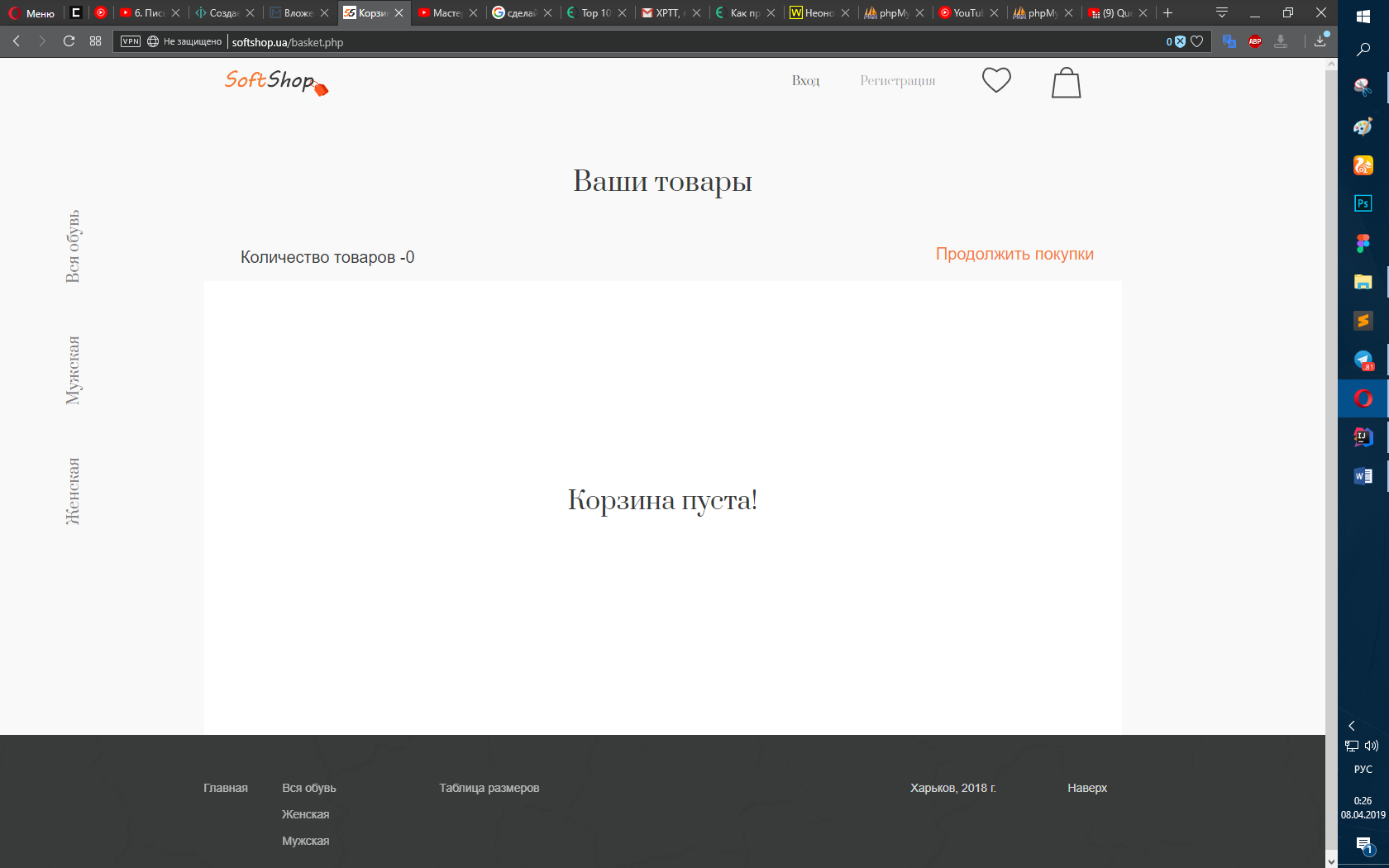
При нажатии на кнопку «–» уменьшается значение количества товаров на 1 (если до нажатия значение было равно 1, то товар удаляется из корзины).



При нажатии на кнопку «Удалить», товар удаляется из корзины.

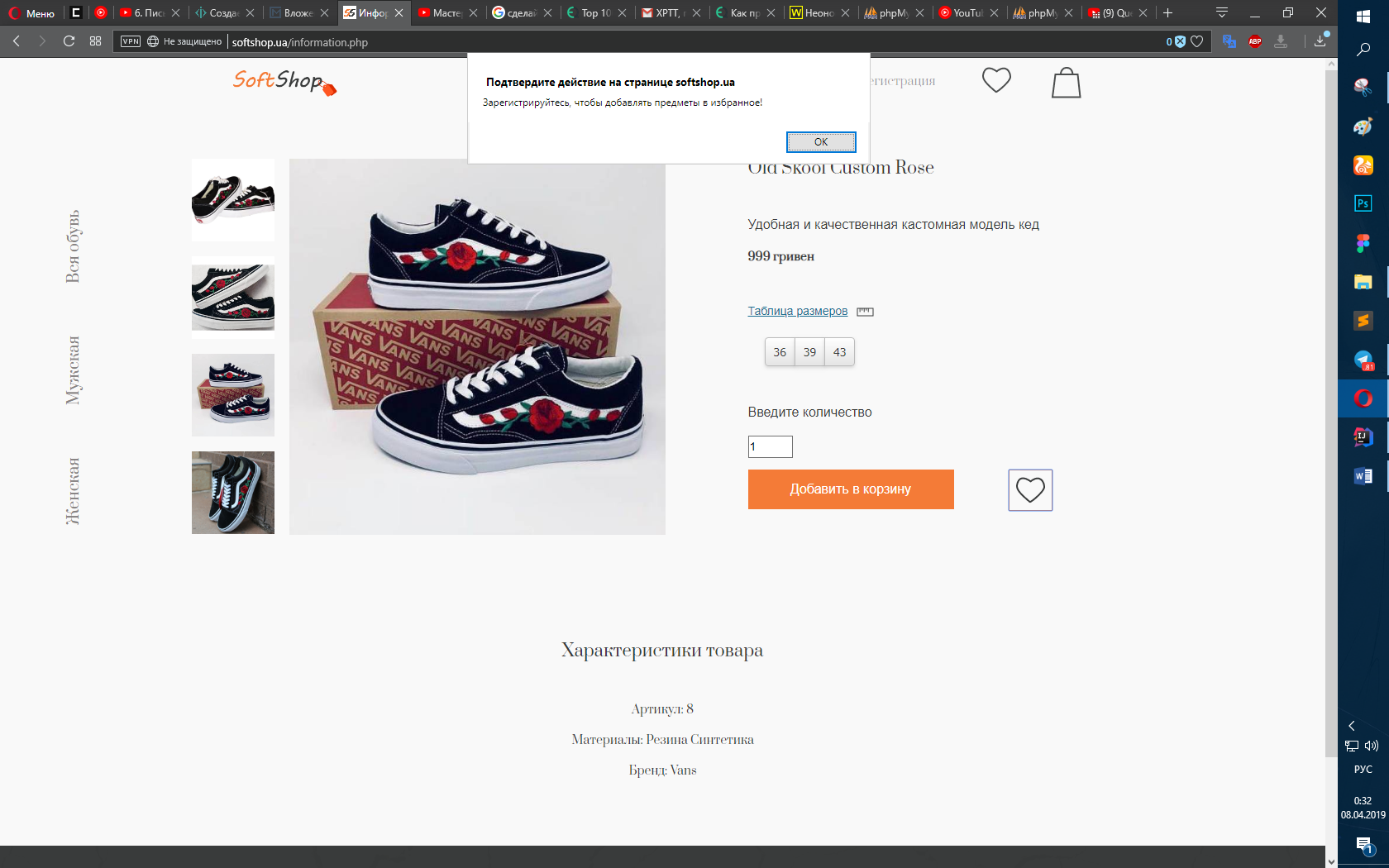


Если корзина пуста, то отображается надпись «Корзина пуста!»



Избранное

Могут просматривать только зарегистрированные пользователи, если гость нажмёт на иконку с изображением сердца в меню (хедере) либо на странице с товаром, то выдается сообщение «Зарегистрируйтесь, чтобы добавлять предметы в избранное!»



# Вывод

вода про все хорошее и перспективы дальнейшего развития

В ходе работы была реализована некоторая серверная часть:

* Возможность перехода между страницами
* Отображение страницы со всеми актуальными (которые есть на складе) товарами с указанием названия, цены.
* Переход на страницу с конкретным товаром с отображением его описания (название, описание, цена, возможные размеры, поле для ввода количества и характеристики товара, а именно – артикул, материалы и бренд) и всех изображений.
* Предусмотрены несколько возможных вариантов добавления в корзину (успешный и нет)
* Регистрация нового пользователя и уведомления о вводе некорректных данных (email существующего пользователя, неверном пароле)
* Вход в аккаунт
* Просмотр корзины, изменение количества товаров, а также их удаление из корзины

Перспективы дальнейшего развития – изменения внешнего вида некоторых элементов на сайте, доработка оформления заказа и страницы с зарегистрированным пользователем, где будет возможность просмотра текущих и всех заказов, добавление товаров в избранное. Также возможно добавление панели администратора и рассылки сообщений о сделанном заказе для пользователя и управляющего.