МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«Гомельский государственный университет

имени Франциска Скорины»

Факультет математики и технологий программирования

Кафедра математических проблем управления и информатики

|  |  |
| --- | --- |
|  | Допущена к защите  Зав. кафедрой\_\_\_\_\_\_\_\_\_\_\_\_\_В.С. Смородин  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2023 г. |

**Разработка интернет-магазина с использованием фреймворков Spring Boot и Angular**

Дипломная работа

Исполнитель

студентка группы ИТП-41 \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Я. Ясиновская

Научный руководитель

доцент кафедры МПУ и

информатики

к.т.н., доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.В. Клименко

Рецензент

доцент кафедры ВМ и

программирования,

к.ф.-м.н., доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е. А. Ружицкая

Гомель 2023

РЕФЕРАТ

Дипломная работа 69 страниц, 37 рисунков, 10 таблиц, 9 источников, 2 приложения

Ключевые слова: Java, Spring Framework, MySQL, Angular Framework, Liquibase, веб-приложение, Docker, Stripe, интернет-магазин

Объект исследования: интернет-магазин.

Цель дипломной работы: разработка интернет-магазина с использова-нием Angular Framework и Spring Framework.

Задачами дипломной работы являются: углубленное теоретическое и практическое изучение Framework Angular; улучшение знаний практическом применении Spring Framework; разработка в среде Intellij IDEA и Visual Studio Code.

Область применения: коммерческие сайты для продажи товаров.

Выводы: получен опыт разработки для интернет-магазина для использования его коммерческих целей с использованием Angular Framework и Spring Framework. Разработанный интернет-магазин имеет практическое применение, интернет-магазин является коммерческим сайтом, который будет способствовать развитию бизнеса.

СОДЕРЖАНИЕ

[Введение 4](#_Toc104542102)

[1 Назначение разработанных программных средств 5](#_Toc104542103)

[2 Средства разработки приложения 6](#_Toc104542104)

[2.1 Angular Framework 6](#_Toc104542105)

[2.2 Язык разметки HTML и CSS 7](#_Toc104542106)

[2.3 Язык программирования Java 8](#_Toc104542107)

[2.4 Spring Framework 9](#_Toc104542108)

[2.5 Docker 1](#_Toc104542108)1

[2.6 Liquibase 11](#_Toc104542108)

[2.7 MySQL 11](#_Toc104542108)

[2.8 Okta 1](#_Toc104542108)2

[2.9 Stripe 1](#_Toc104542108)3

[3 Структура базы данных интернет-магазина 14](#_Toc104542109)

[3.1 Состав таблиц базы данных](#_Toc104542110) 15

[3.2 Структура таблиц базы данных 1](#_Toc104542111)7

[4 Процесс использования интернет -магазина 1](#_Toc104542112)9

[4.1 Процесс авторизации пользователя 19](#_Toc104542113)

[4.2 Отображение интернет-магазина 2](#_Toc104542114)3

[4.3 Коризна товаров 2](#_Toc104542115)8

[4.4 Заказы 3](#_Toc104542116)0

[Заключение 3](#_Toc104542121)6

[Список использованных источников 37](#_Toc104542122)

[Приложение А Фрагмент программного код приложения 38](#_Toc104542123)

[Приложение Б Заключение о результатах плагиат-проверки текста](#_Toc104542123)

[дипломной работы 68](#_Toc104542123)

# ВВЕДЕНИЕ

Интернет-магазин – сайт, торгующий товарами посредством сети Интернет. Позволяет пользователям онлайн, в своём браузере или через мобильное приложение, сформировать заказ на покупку, выбрать способ оплаты и доставки заказа, оплатить заказ. При этом продажа товаров осуществляется дистанционным способом и она накладывает ограничения на продаваемые товары.

Актуальность темы исследования заключается в том, что разработка интернет-магазина позволит удовлетворить потребности пользователя, предоставив им уникальную возможность заказывать понравившиеся. Практическая значимость данного дипломного проекта заключается в создании полноценного интернет-магазина, которое будет способствовать развитию бизнеса и обеспечивать пользователям безопасность персональных данных.

Типичный интернет-магазин позволяет клиенту просматривать ассортимент продуктов и услуг фирмы, просматривать фотографии или изображения продуктов, а также информацию о технических характеристиках продукта и ценах.

С точки зрения рядового пользователя, вам больше не нужно будет стоять по долгу в магазине, выбирая понравившиеся товары или же стоять в очередях для их покупки.

В ходе выполнения дипломной работы решались следующие задачи:

- изучение ORM Spring Framework;

- разработка структуры базы данных через классы;

- углубление знаний Angular Framework для построения визуальной части интернет -магазина;

- изучение языка разметки HTML и CSS;

- использование дополнительного функционала, таких как Okta и Stripe.

# 1 Назначение разработанных программных средств

Идея интернет-магазина: дать возможность покупателям использовать функции поиска, чтобы найти конкретные модели, бренды или предметы, заказать их и получить через промежуток времени.

Пользователь можете войти в систему, и начать полноценно пользоваться такими вещами как покупка товаров, где он сможет выбирать понравившиеся ему товары и заказывать их онлайн.

В случае утери пароля, пользователь может его сбросить посредством своей электронной почты.

Если пользователь даже не зарегистрировался на сайте, он может добавить товар в корзину, но заказать могут только зарегистрированные пользователи.

Сайт имеет удобный интерфейс для взаимодействия с клиентом, таких функции как пагинация или поиск нужного продукта есть в интернет- магазине, а ещё все товары отсортированы по категориям, это сделано для того, чтоб пользователю было удобно ориентироваться на сайте.

Сайт защищён SLL-сертификатом, это значит, что его взломать тяжелее, чем сайты, не защищённые сертификатами. Сайт использует в своём url адресе https, который является новым стандартным протоколом безопасности.

Так же сайт использует для заказов Stripe.

Stripe – международная система для защищённых интернет-платежей с помощью пластиковых карт.

Говоря проще, Stripe выступает посредником между покупателем и продавцом, который проводит безопасный с точки зрения защиты информации платёж. Stripe решает все проблемы платежей картами, включая хранение данных карт, периодические платежи и выводы на банковский счёт.

Систему Stripe отличает высокая скорость обработки платежей и безопасность, а с точки зрения бизнеса – очень удобные средства интеграции с сайтами (особенно для программистов).

Таким образом, разработанный интернет-магазин направлен на предоставление различного рода товаров, где пользователь может покупать товар, не беспокоясь о потере своих персональных данных.

**2 Средства разработки приложения**

**2.1 Framework Angular**

Angular – это бесплатная платформа веб-приложений с открытым исходным кодом на основе TypeScript, возглавляемая командой Angular в Google и сообществом отдельных лиц и корпораций. Angular полностью переписан той же командой, которая создала AngularJS. Angular – это одностраничный фреймворк для создания быстрых веб-приложений. Он использует концепцию одностраничного приложения, в котором пользовательский интерфейс доставляется в начале запроса приложения, а затем запрашиваются только данные, что делает одностраничные приложения быстрыми.

Отличительные достоинства Angular:

- TypeSrcipt: Angular написан на TypeSrcipt, что обеспечивает более строгую типизацию и более безопусный код;

- Директивы: Angular имеет множество встроенных директив, которые позволяют управлять DOM-элементами и создавать собственные директивы;

- Модульность: Angular имеет модульную структуру, которая позволяет разбивать приложение на небольшие части и повторно использовать их в других проектах;

- Dependency Injection: Angular имеет мощную систему внедрения зависимостей, которая позволяет легко управлять зависимостям компонентов;

- RxJS: Angular использует RxJS для работы с асинхронными операциями и потоками данных;

- Angular CLI: Angular имеет свой собственный инструмент командной строки (CLI), который упрощает создание новых проектов, добавление компонентов и модулей, тестирование и развёртывание приложений;

- Angular Universal: Angular имеет возможность создания универсальных приложений, которые могут работать как на сервере, так и на клиенте;

- Поддержка: Angular имеет активное сообщество разработчиков, которые создают плагины, расширения и библиотеки для улучшения функциональности фреймворка. Для каждой вводимой функции или при обнаружении неверного синтаксиса во время разработки интерфейса, будет выведена соответствующая информация.

Непосредственно Angular:

- Сложность: Angular имеет довольно высокий порог входа для новичков, из-за большого количества концепций и технологий, которые нужно изучить;

- Размер: Angular имеет довольно большой размер, что может привести к дополнительным затратам на загрузку и выполнение кода;

- Скорость: Из-за большого количества абстракций и сложной структуры, Angular может работать медленнее, чем другие фреймворки;

- Сложность тестирования: Из-за сложной структуры и зависимостей, тестирование приложений на Angular может быть сложным и требовательным к ресурсам;

- Обновления: Обновление Angular до новых версий может потребовать значительных изменений в коде приложения, что может быть сложно и затратно в ресурсном плане.Ускорение, достигается аппаратным ускорением графики, вне зависимости от того, работаем мы с 2D или 3D, графикой или текстом, все компоненты транслируются в объекты понятные Direct3D и затем визуализируются с помощью процессора на видеокарте, что повышает производительность и делает графику плавной [1].

**2.2 Язык разметки HTML и CSS**

HTML (язык гипертекстовой разметки) и CSS (каскадные таблицы стилей) – два ключевых процесса для создания веб-страниц. HTML гарантирует текстуру страницы, CSS – модель (визуальный, а также акустический) для разных приборов. Наравне с графикой и сценариями HTML и CSS являются базовой для создания веб-страниц и веб-приложений.

Особенности HTML:

- Простота: HTML является довольно простым языком, который легко понимать и использовать;

- Кроссплатформенность: HTML может быть использован на любой операционной системе и в любом браузере;

- Семантика: HTML позволяет создавать структурированный контент, который легко понимать поисковым системам и другим приложениям;

- Доступность: HTML имеет инструменты для создания доступных веб-страниц для пользователей с ограниченными возможностями.

Недостатки HTML:

- Ограниченность: HTML не может обеспечить динамический контент или интерактивность без использования других технологий, таких как JavaScript;

- Сложность разработки: Создание сложных веб-страниц может быть сложным и требовательным к ресурсам;

- Необходимость обновления: HTML постоянно развивается и обновляется, что может потребовать изменения кода веб-страниц.

Особенности CSS:

- Разделение дизайна и содержимого: CSS позволяет разделить стиль и дизайн от содержимого веб-страницы, что облегчает ее разработку и поддержку;

- Гибкость: CSS позволяет создавать разнообразные стили и дизайны для веб-страниц, что делает их более привлекательными для пользователей;

- Масштабируемость: CSS позволяет легко изменять стиль и дизайн веб-страницы без изменения ее содержимого;

- Поддержка мультимедиа: CSS позволяет создавать анимации, трансформации и другие эффекты для мультимедийного контента;

Недостатки CSS:

- Сложность: CSS может быть сложным для новичков из-за большого количества свойств и возможностей;

- Кроссбраузерность: Различные браузеры могут по-разному интерпретировать CSS, что может привести к несовместимости;

- Производительность: Использование большого количества CSS может снизить производительность веб-страницы [2].

**2.3 Язык программирования Java**

Java – это объектно-ориентированный язык программирования, который был разработан в 1995 году компанией Sun Microsystems (сейчас часть компании Oracle). Он используется для создания приложений на различных платформах, включая компьютеры, мобильные устройства и веб-серверы.

Java была создана компанией Sun Microsystems в 1995 году. Sun Microsystems была американской компанией, занимавшейся разработкой и производством компьютерного оборудования и программного обеспечения. Она была основана в 1982 году и стала одной из крупнейших компаний в сфере вычислительной техники. В 2010 году компания была приобретена компанией Oracle. Сейчас Java продолжает развиваться под управлением Oracle.

Преимущества Java:

- Переносимость: Java-приложения могут работать на любой платформе, так как они компилируются в байт-код, который может быть запущен на любой машине, где установлена Java Virtual Machine (JVM);

- Безопасность: Java имеет встроенные механизмы безопасности, которые защищают от вирусов и других угроз;

- Объектно-ориентированность: Java поддерживает объектно-ориентированное программирование, что делает код более модульным и легко читаемым;

- Большая библиотека: Java имеет обширную библиотеку классов, которая содержит множество готовых решений для различных задач;

Недостатки Java:

- Низкая производительность: Использование JVM может привести к снижению производительности приложения;

- Сложность: Java может быть сложным для начинающих программистов из–за большого количества концепций и возможностей;

- Медленная загрузка: Приложения на Java могут загружаться медленнее, чем на других языках программирования;

- Необходимость обновления: Java постоянно обновляется, и разработчики должны следить за новыми версиями и обновлять свой код [3].

**2.4 Spring Framework**

Spring framework – это один из самых популярных фреймворков для разработки приложений на языке Java. Он был создан в 2003 году Родом Джонсоном и стал первым в своём роде фреймворком для управления объектами.

Spring предоставляет множество функций, таких как управление жизненным циклом объектов, управление транзакциями, управление конфигурацией, а также интеграцию с другими фреймворками и библиотеками.

Spring framework состоит из нескольких модулей, каждый из которых предоставляет определённый функционал. Например, модуль Spring MVC используется для разработки веб-приложений, модуль Spring Security – для обеспечения безопасности приложения и т.д.

Spring также поддерживает инверсию управления (IoC) и внедрение зависимостей (DI), что делает разработку приложений более гибкой и удобной.

Благодаря своей популярности, Spring framework имеет большое сообщество разработчиков, которые создают дополнительные модули и расширения для фреймворка.

Spring Data – это модуль Spring framework, который предоставляет унифицированный подход к работе с различными хранилищами данных, такими как реляционные базы данных, NoSQL базы данных, key-value хранилища и т.д.

Spring Data предоставляет абстракции для работы с данными, такие как репозитории и шаблоны доступа к данным, которые позволяют разработчикам более эффективно работать с данными в их приложениях.

Spring Data также предоставляет множество модулей, каждый из которых предназначен для работы с определенным типом хранилища данных. Например, модуль Spring Data JPA предназначен для работы с реляционными базами данных через Java Persistence API (JPA), а модуль Spring Data MongoDB предназначен для работы с базами данных MongoDB.

Благодаря Spring Data, разработчики могут более эффективно работать с различными хранилищами данных, что позволяет им создавать более мощные и гибкие приложения.

Spring JPA (Java Persistence API) – это модуль Spring framework, который предоставляет унифицированный подход к работе с реляционными базами данных через Java Persistence API (JPA).

Spring JPA позволяет разработчикам более эффективно работать с базами данных, предоставляя абстракции для работы с данными, такие как репозитории и шаблоны доступа к данным.

Spring JPA также предоставляет множество функций, таких как автоматическое создание таблиц и запросов, кэширование данных и т.д., что позволяет разработчикам более быстро и эффективно работать с базами данных.

Благодаря Spring JPA, разработчики могут создавать более мощные и гибкие приложения, которые легко масштабировать и поддерживать.

Spring Security – это модуль Spring framework, который предоставляет механизмы аутентификации, авторизации и защиты приложений от атак.

Spring Security позволяет разработчикам легко интегрировать механизмы безопасности в свои приложения, предоставляя готовые решения для настройки аутентификации и авторизации, контроля доступа к ресурсам, защиты от CSRF-атак и других угроз.

Spring Security также поддерживает различные механизмы аутентификации, такие как базовая аутентификация, форма входа, OAuth и другие.

Благодаря Spring Security, разработчики могут создавать более безопасные приложения, которые защищены от угроз и соответствуют современным стандартам безопасности.

Spring Boot – это фреймворк для быстрого создания приложений на основе Spring framework. Он предоставляет набор инструментов и функций, которые позволяют разработчикам создавать приложения с минимальной конфигурацией и усилиями.

Spring Boot позволяет автоматически настраивать и подключать необходимые библиотеки и инструменты, что упрощает процесс разработки и ускоряет время выхода приложения на рынок. Он также обеспечивает удобную работу с базами данных, веб-серверами, сервисами облачных платформ и другими компонентами.

Spring Boot является отличным выбором для создания микросервисных архитектур, так как он позволяет создавать отдельные модули приложения, которые могут быть легко масштабированы и управляемы [4].

**2.5 Docker**

Docker – это платформа для разработки, доставки и запуска приложений в контейнерах. Контейнеры Docker позволяют упаковывать приложения со всеми их зависимостями в единый пакет, который может быть запущен на любой платформе, где установлен Docker.

Docker позволяет разработчикам и операторам создавать, развёртывать и масштабировать приложения с помощью контейнеров, что упрощает процесс разработки и ускоряет время выхода приложения на рынок. Он также обеспечивает удобную работу с базами данных, веб–серверами, сервисами облачных платформ и другими компонентами.

Docker является отличным выбором для создания микросервисных архитектур, так как он позволяет создавать отдельные модули приложения, которые могут быть легко масштабированы и управляемы [5].

**2.6 Liquibase**

Liquibase – это инструмент для управления версиями баз данных. Он позволяет разработчикам и администраторам баз данных контролировать изменения в структуре баз данных и автоматически применять эти изменения к различным средам (например, отладочной, тестовой или производствен-ной).

Liquibase использует XML–файлы для описания изменений, которые нужно внести в базу данных. Эти файлы могут быть хранены в системе контроля версий, что обеспечивает прозрачность и контроль над изменениями в базе данных.

Liquibase поддерживает множество СУБД, таких как Oracle, MySQL, PostgreSQL, Microsoft SQL Server и другие. Он также предоставляет API для интеграции с другими инструментами и системами.

Использование Liquibase позволяет сократить время на развёртывание и обновление баз данных, повысить надёжность и безопасность баз данных, а также облегчить процесс совместной работы разработчиков и администраторов баз данных [6].

**2.7 MySQL**

MySQL – это одна из самых популярных реляционных баз данных, которая используется в различных приложениях и веб-сайтах. MySQL обладает широкими возможностями для хранения, организации и манипулирования данными, а также обеспечивает высокую производительность и надёжность. MySQL поддерживает множество языков программирования и платформ, что делает его универсальным инструментом для разработки приложений.

MySQL был создан в 1995 году шведским разработчиком Майклом Виднерссоном и его компанией MySQL AB. В 2008 году компания была приобретена корпорацией Sun Microsystems, а затем перешла во владение Oracle Corporation.

MySQL поддерживает стандарт SQL и имеет множество функций, включая:

- Поддержку транзакций для обеспечения целостности данных;

- Возможность создания индексов для ускорения поиска и сортировки данных;

- Поддержку хранимых процедур и функций для обработки данных на стороне сервера;

- Поддержку репликации для создания копий базы данных на нескольких серверах;

- Поддержку различных типов данных, включая текстовые, числовые, даты и времена, изображения и другие;

- Возможность создания и управления пользователями и правами доступа к данным;

MySQL также имеет множество инструментов для управления базой данных, включая командную строку, графические интерфейсы и API для различных языков программирования [7].

**2.8 Okta**

Okta – это облачный сервис управления идентификацией и доступом, который позволяет организациям управлять доступом пользователей к приложениям и сервисам в облаке. С помощью Okta пользователи могут авторизоваться в различных приложениях с помощью единого логина и пароля, а также получить доступ к приложениям на основе своих ролей и прав доступа.

Основные функции Okta включают:

- Поддержку транзакций для обеспечения целостности данных;

- Единый вход для пользователей в различные приложения;

- Управление идентификацией и авторизацией пользователей;

- Управление ролями и правами доступа пользователей к приложениям;

- Многофакторную аутентификацию для повышения безопасности;

- Мониторинг и анализ доступа пользователей к приложениям;

- Интеграцию с различными облачными и локальными приложениями.

Okta используется множеством организаций, включая крупные корпорации, государственные учреждения и малые бизнесы, для управления доступом пользователей к различным приложениям в облаке.

Существует два типа токенов в Okta:

- Access Token – это токен, который выдаётся после успешной аутентификации пользователя. Он используется для получения доступа к защищённым ресурсам, таким как API и веб–приложения.

- Refresh Token – это токен, который выдаётся вместе с Access Token и используется для обновления Access Token без необходимости повторной аутентификации пользователя.

Токены в Okta используются для обеспечения безопасности и защиты конфиденциальности пользователей при доступе к приложениям и сервисам. Они также позволяют компаниям управлять доступом пользователей к различным ресурсам и контролировать их активности [8].

**2.9 Stripe**

Stripe – это платёжная система, которая позволяет компаниям и частным лицам принимать онлайн–платежи по всему миру. Stripe предоставляет набор инструментов для обработки платежей, включая API для интеграции с сайтами и приложениями, а также инструменты для управления подписками и повторными платежами.

Основные функции Stripe включают:

- Обработку кредитных и дебетовых карт;

- Поддержку различных валют и способов оплаты;

- Возможность создания и управления подписками и повторными платежами;

- Инструменты для борьбы с мошенничеством и обеспечения безопасности платежей;

- Интеграцию с различными CMS, платформами электронной коммерции и другими сервисами.

Stripe используется множеством компаний по всему миру, включая Airbnb, Lyft, Shopify и другие. Stripe также предоставляет инструменты для разработчиков, которые позволяют создавать собственные приложения и сервисы для обработки платежей [9].

**3 Структура базы данных интернет-магазина**

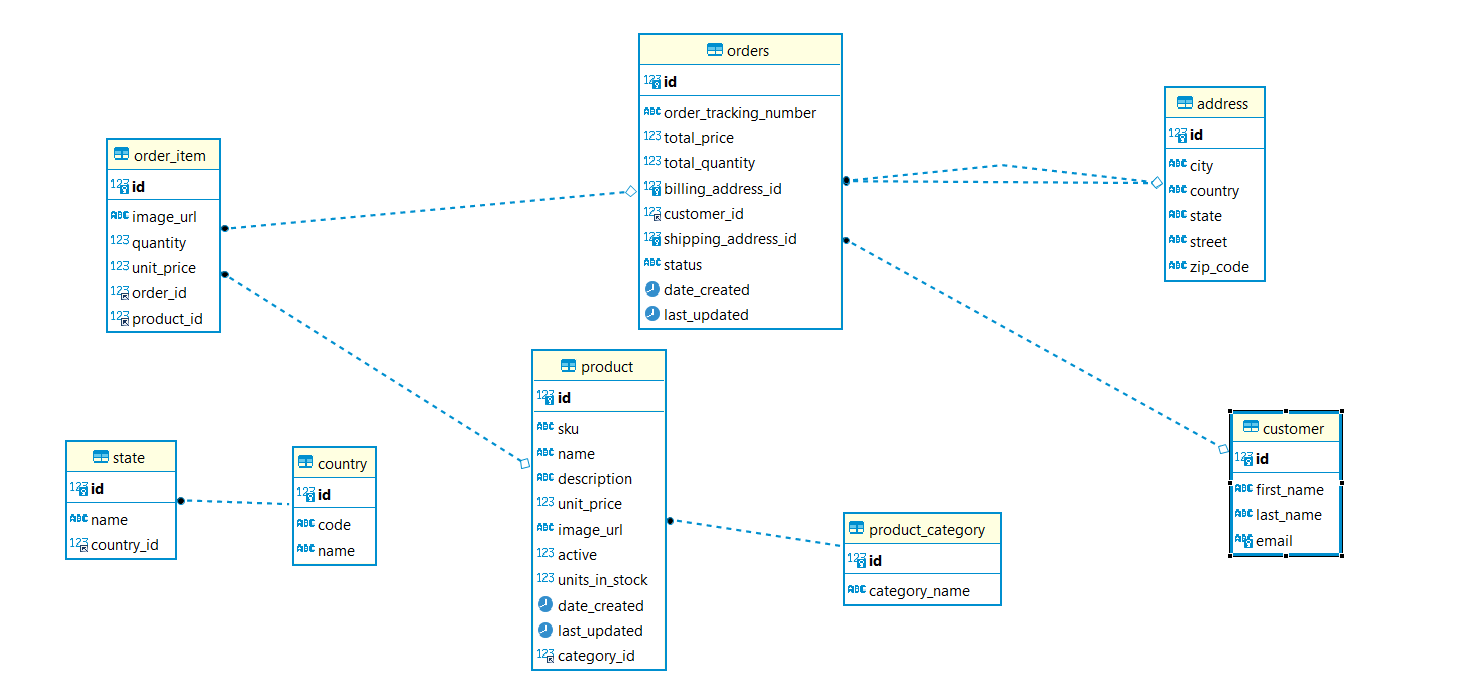
**3.1 Состав таблиц базы данных**

База данных и таблицы были созданы с помощью Spring Boot. Состав используемых в приложении таблиц базы данных приведён в таблице 3.1.

**Таблица 3.1 − Состав таблиц базы данных разработанного приложения**

|  |  |
| --- | --- |
| Наименование | Общие сведения о содержимом |
| 1 address | Адрес доставки товаров покупателю |
| 2 country | Страна доставки товаров покупателю |
| 3 customer | Список покупателей |
| 4 order\_item | Список товаров в заказе |
| 5 orders | Заказ со списком товаров, который сделал покупатель |
| 6 product | Список товаров |
| 7 product\_category | Список категорий товаров |
| 8 state | Штат доставки товаров покупателю |

Взаимосвязь между таблицами показана на рисунке 3.1.



**Рисунок 3.1 – Схема базы данных приложения**

**3.2 Структура таблиц базы данных**

Структуры таблиц базы данных приведены в таблицах 3.2 – 3.9

**Таблица 3.2 – Структура таблицы address**

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Формат поля в модели | Формат поля в базе данных | Содержимое поля |
| id | Bigint | bigint(20) | Ключевое поле для таблица address |
| city | Varchar | varchar(255) | Название города, в котором проживает покупатель |
| country | Varchar | varchar(255) | Название страны, в которой проживает покупатель |
| state | Varchar | varchar(255) | Название штата, в котором проживает покупатель |
| street | Varchar | varchar(255) | Название улицы, в котором проживает покупатель |
| zip\_code | Varchar | varchar(255) | Почтовый индекс, который должен быть у покупателя |

**Таблица 3.3 – Структура таблицы country**

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Формат поля в модели | Формат поля в базе данных | Содержимое поля |
| id | Smallint | smallint(5) unsigned | Ключевое поле для таблица country |
| code | Varchar | varchar(2) | Код страны |
| name | Varchar | varchar(255) | Название страны |

**Таблица 3.4 – Структура таблицы customer**

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Формат поля в модели | Формат поля в базе данных | Содержимое поля |
| 1 | 2 | 3 | 4 |

**Окончание таблицы 3.4**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| id | Bigint | bigint(20) | Ключевое поле для таблица customer |
| first\_name | Varchar | varchar(255) | Имя покупателя |
| last\_name | Varchar | varchar(255) | Фамилия покупателя |
| email | Varchar | varchar(255) | Электронная почта покупателя |

**Таблица 3.5 – Структура таблицы order\_item**

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Формат поля в модели | Формат поля в базе данных | Содержимое поля |
| id | Bigint | bigint(20) | Ключевое поле для таблица order\_item |
| image\_url | Varchar | varchar(255) | Путь к картинке товара |
| quantity | Int | Int(11) | количество товара в корзине |
| unit\_price | Decimal | decimal(19,2) | Сумма товаров в корзине |
| order\_id | Bigint | bigint(20) | Вторичное поле, которая связывает таблицы order\_item и orders |
| product\_id | Bigint | bigint(20) | Вторичное поле, которая связывает таблицы order\_item и product |

**Таблица 3.6 – Структура таблицы orders**

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Формат поля в модели | Формат поля в базе данных | Содержимое поля |
| 1 | 2 | 3 | 4 |
| id | Bigint | bigint(20) | Ключевое поле для таблица orders |
| order\_tracking\_number | Varchar | varchar(255) | Номер заказа покупателя |

**Окончание таблицы 3.6**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| total\_price | Decimal | decimal(19,2) | Общая сумма заказа |
| total\_quantity | Int | int(11) | Общее количество заказа |
| billing\_address\_id | Bigint | bigint(20) | Вторичное поле, которая связывает таблицы orders и address |
| customer\_id | Bigint | bigint(20) | Вторичное поле, которая связывает таблицы orders и customer |
| shipping\_address\_id | Bigint | bigint(20) | Вторичное поле, которая связывает таблицы orders и shipping |
| status | Varchar | varchar(128) | Статус товара |
| date\_created | Datetime | Datetime(6) | Дата создания заказа |
| last\_updated | Datetime | Datetime(6) | Дата последнего изменения заказа |

**Таблица 3.7 – Структура таблицы product**

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Формат поля в модели | Формат поля в базе данных | Содержимое поля |
| 1 | 2 | 3 | 4 |
| id | Bigint | bigint(20) | Ключевое поле для таблица product |
| sku | Varchar | varchar(255) | Название картинки для продуктов |
| name | Varchar | varchar(255) | Название продукта |
| description | Varchar | varchar(255) | Описание продукта |
| unit\_price | Decimal | decimal(19,2) | Цена продукта |

**Окончание таблицы 3.7**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 4 |
| image\_url | Varchar | varchar(255) | Путь к картинке товара |
| active | Bit | bit(20) | Показывает, находиться ли товар в козине или нет |
| unit\_in\_stock | Int | int(11) | Сколько товар на складе |
| date\_created | Datetime | Datetime(6) | Дата создания заказа |
| last\_updated | Datetime | Datetime(6) | Дата последнего изменения заказа |
| category\_id | Bigint | bigint(20) | Вторичное поле, которая связывает таблицы product и category |

**Таблица 3.8 – Структура таблицы product\_category**

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Формат поля в модели | Формат поля в базе данных | Содержимое поля |
| id | Bigint | bigint(20) | Ключевое поле для таблица product\_category |
| category\_name | Varchar | varchar(255) | Название категории |

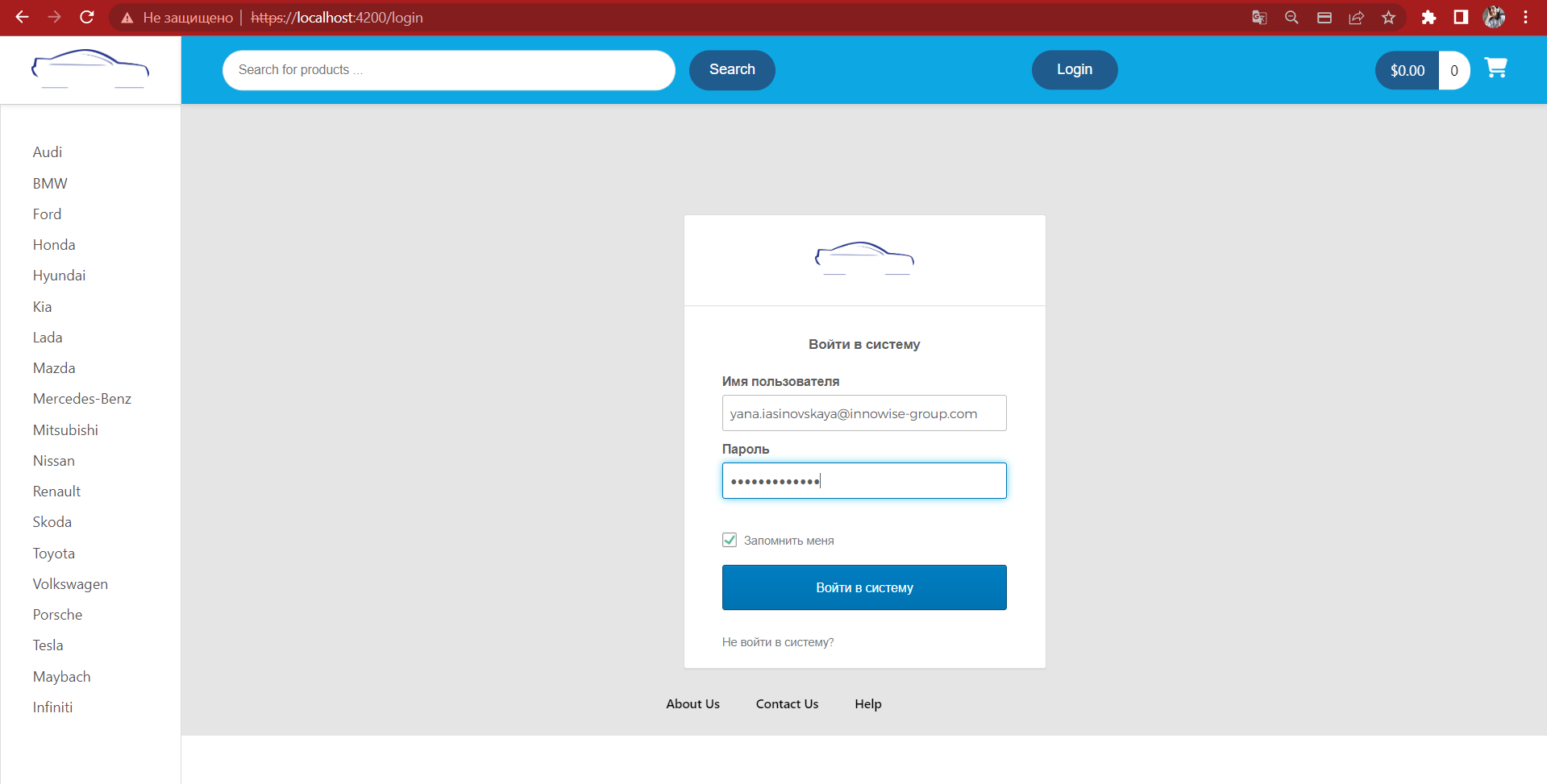
**Таблица 3.9 – Структура таблицы state**

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Формат поля в модели | Формат поля в базе данных | Содержимое поля |
| id | Smallint | smallint(5) unsigned | Ключевое поле для таблица state |
| name | Varchar | varchar(255) | Название страны, для доставки |
| country\_id | Smallint | smallint(5) unsigned | Вторичное поле, которая связывает таблицы state и country |

**4 Процесс использования интернет-магазина**

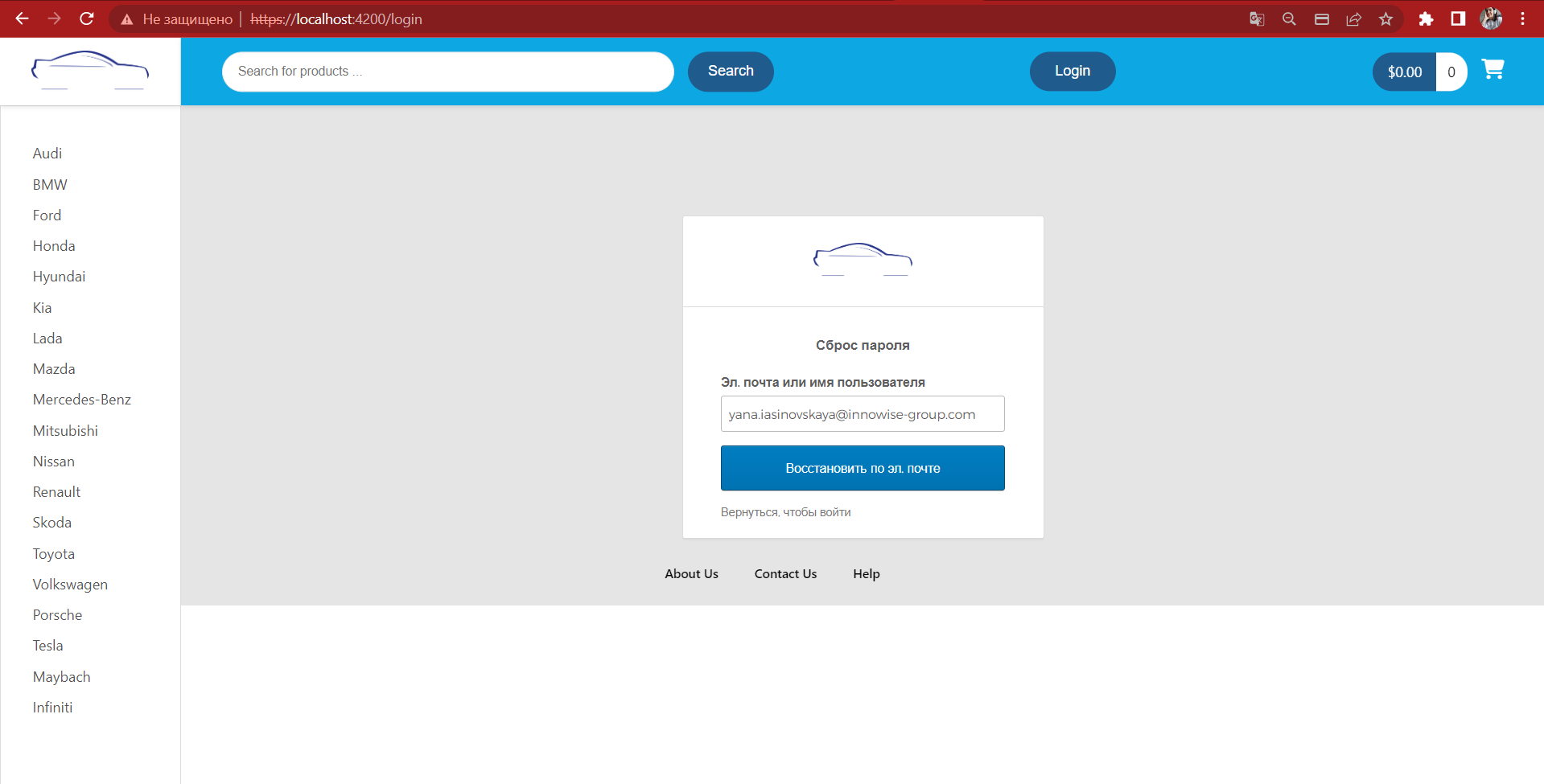
**4.1 Процесс авторизации пользователя**

Здесь рассмотрим наглядный пример как происходит авторизация пользователя при открытии приложения. На рисунке 4.1 показано окно авторизации с заполненными данными для авторизации. Авторизация проходит через систему Okta. Для авторизации пользователей в Okta используется протокол OAuth2.

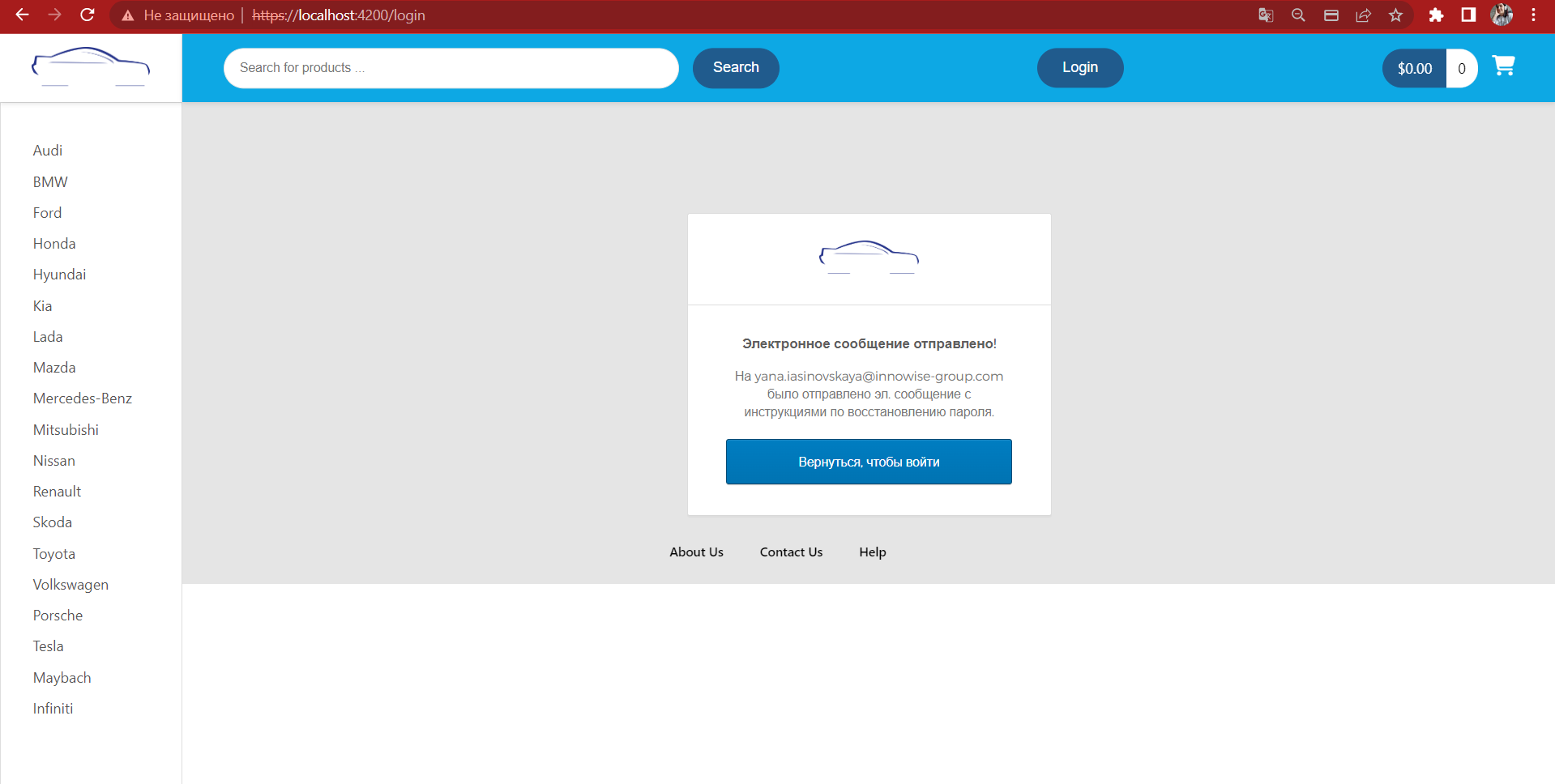


**Рисунок 4.1 – Авторизация**

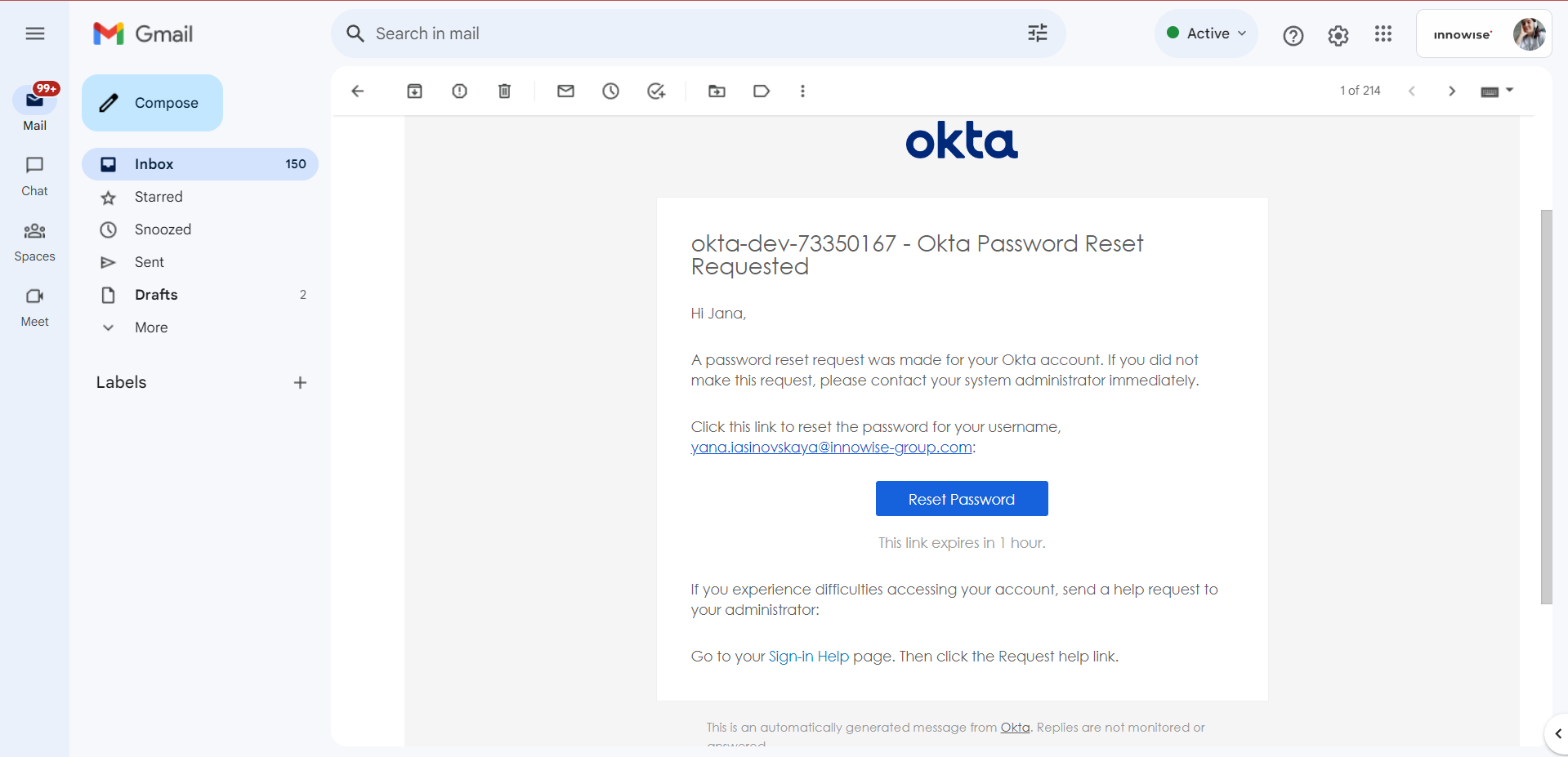
Так же, если пользователь забыл пароль, его можно восстановить с помощью адреса электронной почты, как показано на рисунке 4.2– 4.5.



**Рисунок 4.2 – Восстановление пароля**

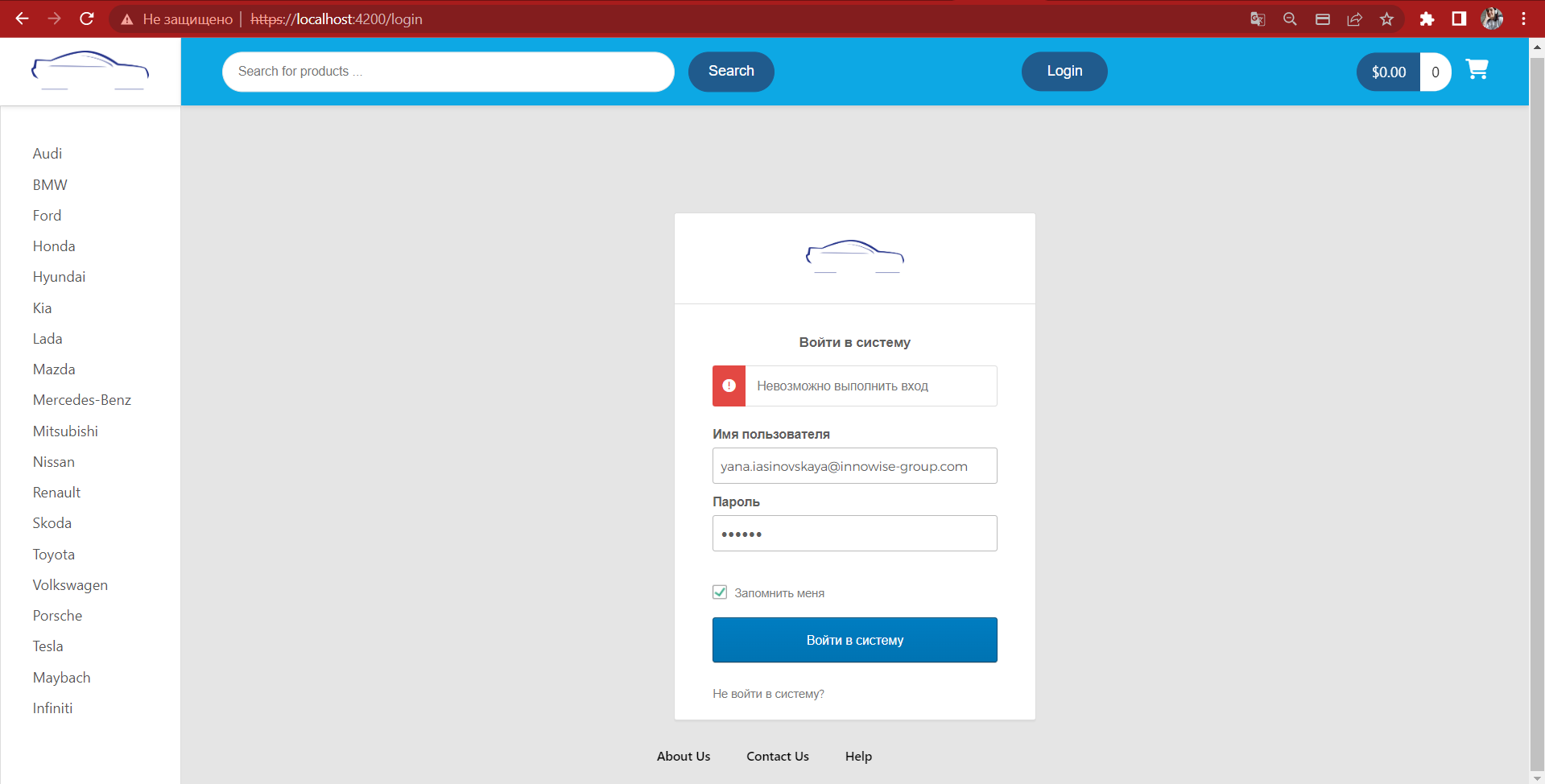


**Рисунок 4.3 – Отправка пароль на электронную почту**



**Рисунок 4.4 – Получение сообщения о сбрасывания пароля на электронную почту**

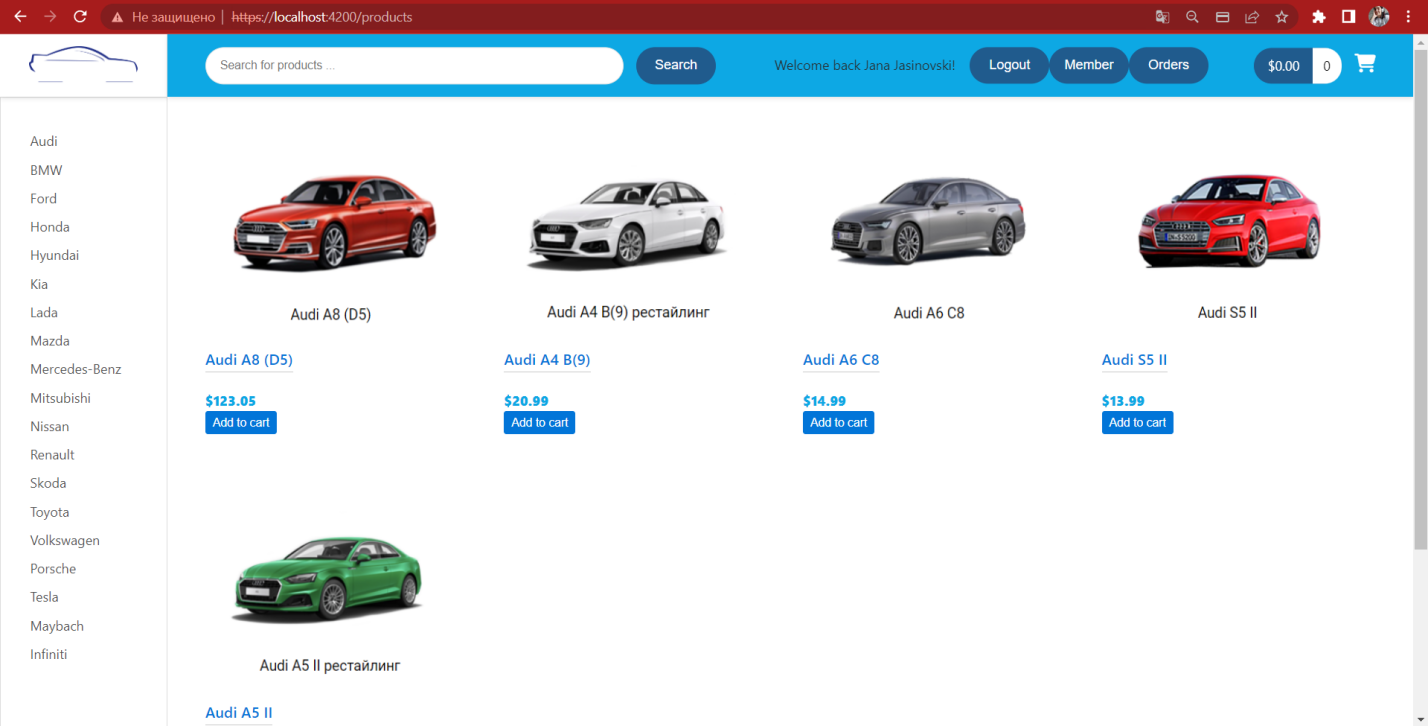
Если пользователь введёт некорректные данные, система покажет ошибку, её можно увидеть на рисунке 4.5.



**Рисунок 4.5 – Ошибка авторизации**

Так же стоит заметить, что url адрес не http, а https. Это значит, что приложение использует защищённый порт, который работает на транспортном уровне. Https порт подписан SSL–[сертификатом](https://www.google.com/search?rlz=1C1CHZN_ruBY1033BY1033&sxsrf=APwXEddg4PXwlZokVZbnmVN-NApcXukOQA:1685907322357&q=%D1%81%D0%B5%D1%80%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%82%D0%BE%D0%BC&spell=1&sa=X&ved=2ahUKEwjA4MqYrqr_AhWEDRAIHb0gA4sQBSgAegQIBxAB), который даёт приложению защиту от несанкционированного доступа при авторизации.

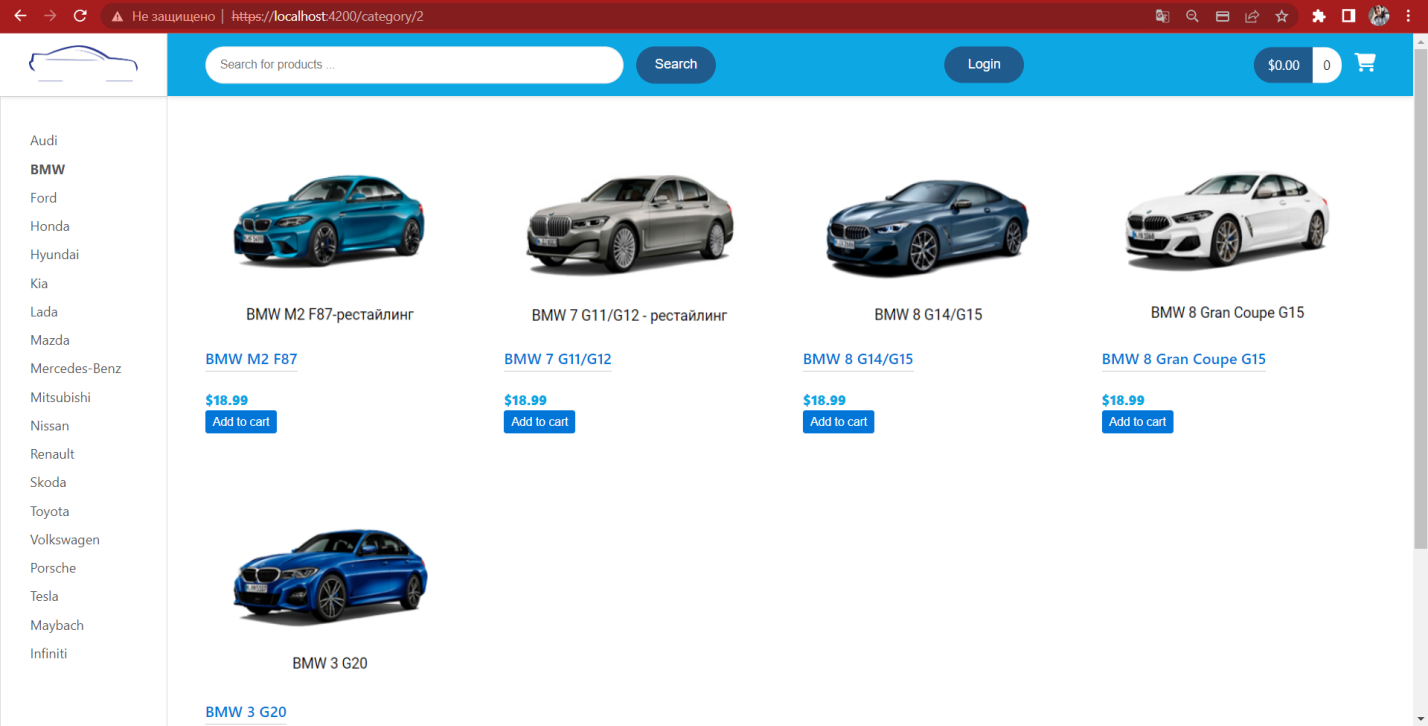
После успешной авторизации в заголовке приложения можно увидеть приветственное письмо, которое изображено на рисунке 4.6.



**Рисунок 4.6 – Успешная авторизация**

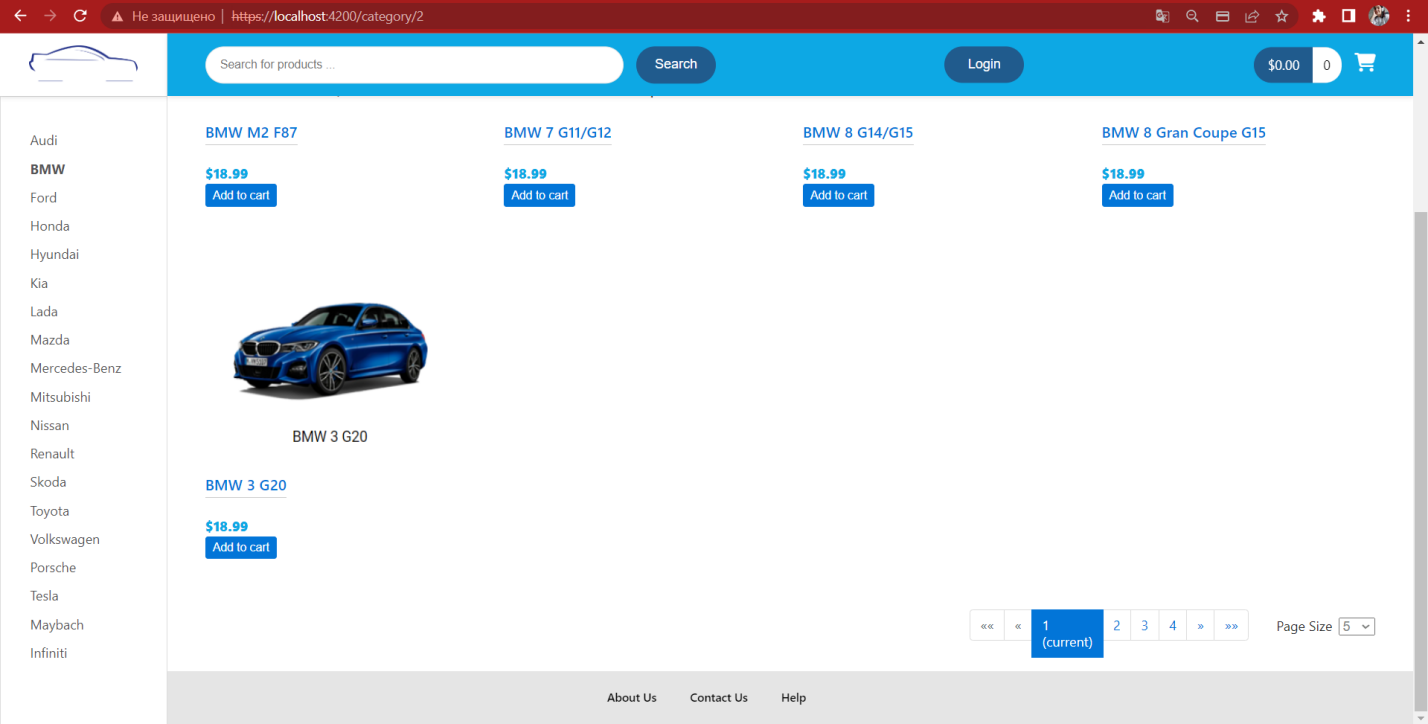
**4.2 Отображение магазина автомобилей**

На рисунке 4,6 показана страница, которую пользователь может видеть. Пользователь может находить товар, который ему необходим по категориям, как на рисунке 4.7 (категория: BMW)

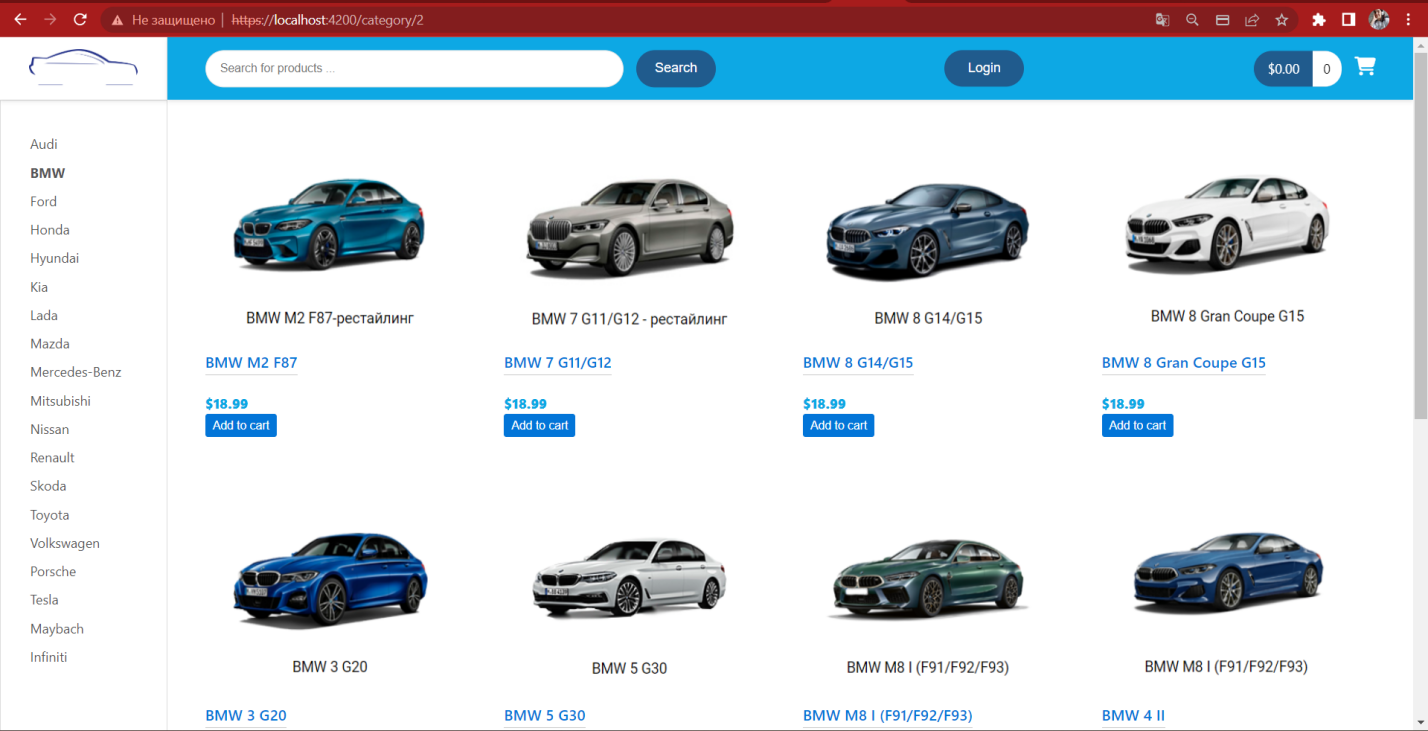


**Рисунок 4.7 – Категория BMW**

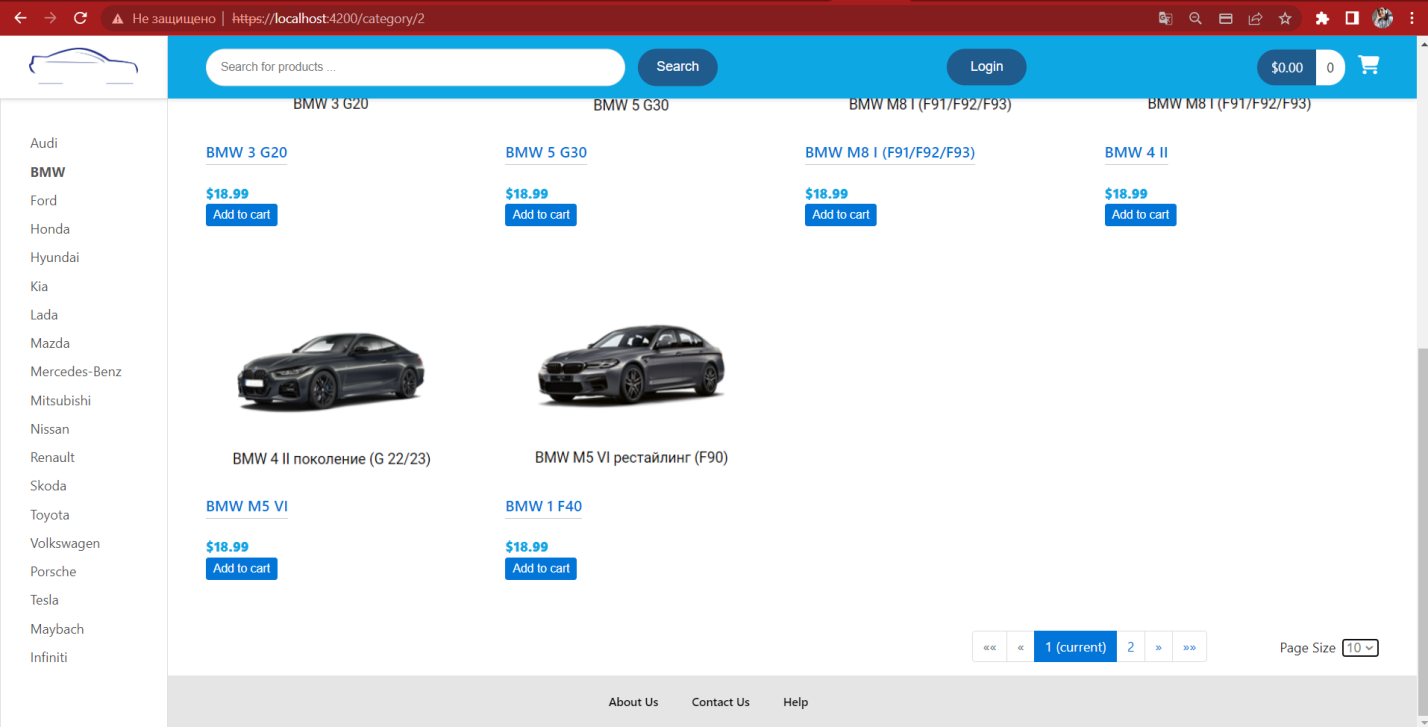
Пользователь может отсортировать товары с помощью пагинации и выбора количества товаров, которая располагается внизу каждой страницы товара, рисунок 4.8 – 4.10. Пагинация зависит от количества товара, чем больше пользователь просит показать товаров, тем меньше количество страниц пользователь можем выбирать.



**Рисунок 4.8 – Пагинация**

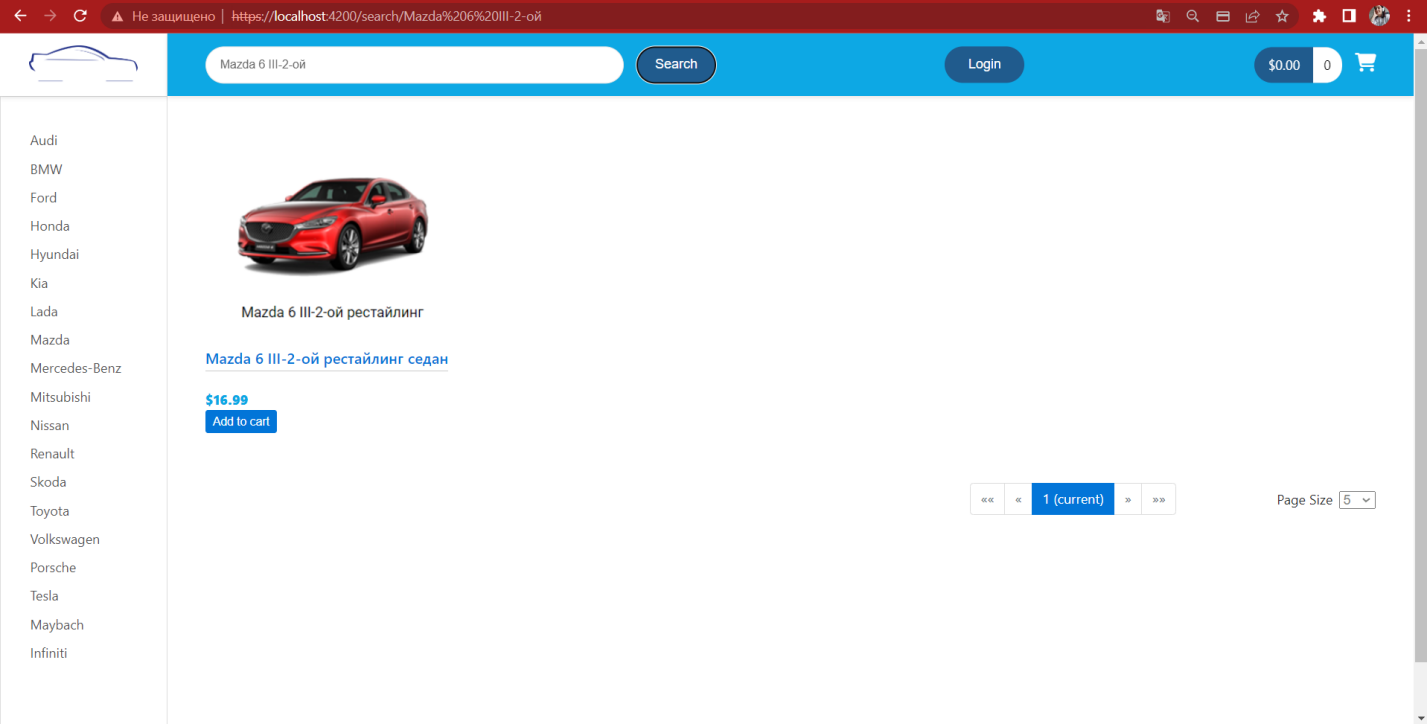


**Рисунок 4.9 – Выбор количества товаров**



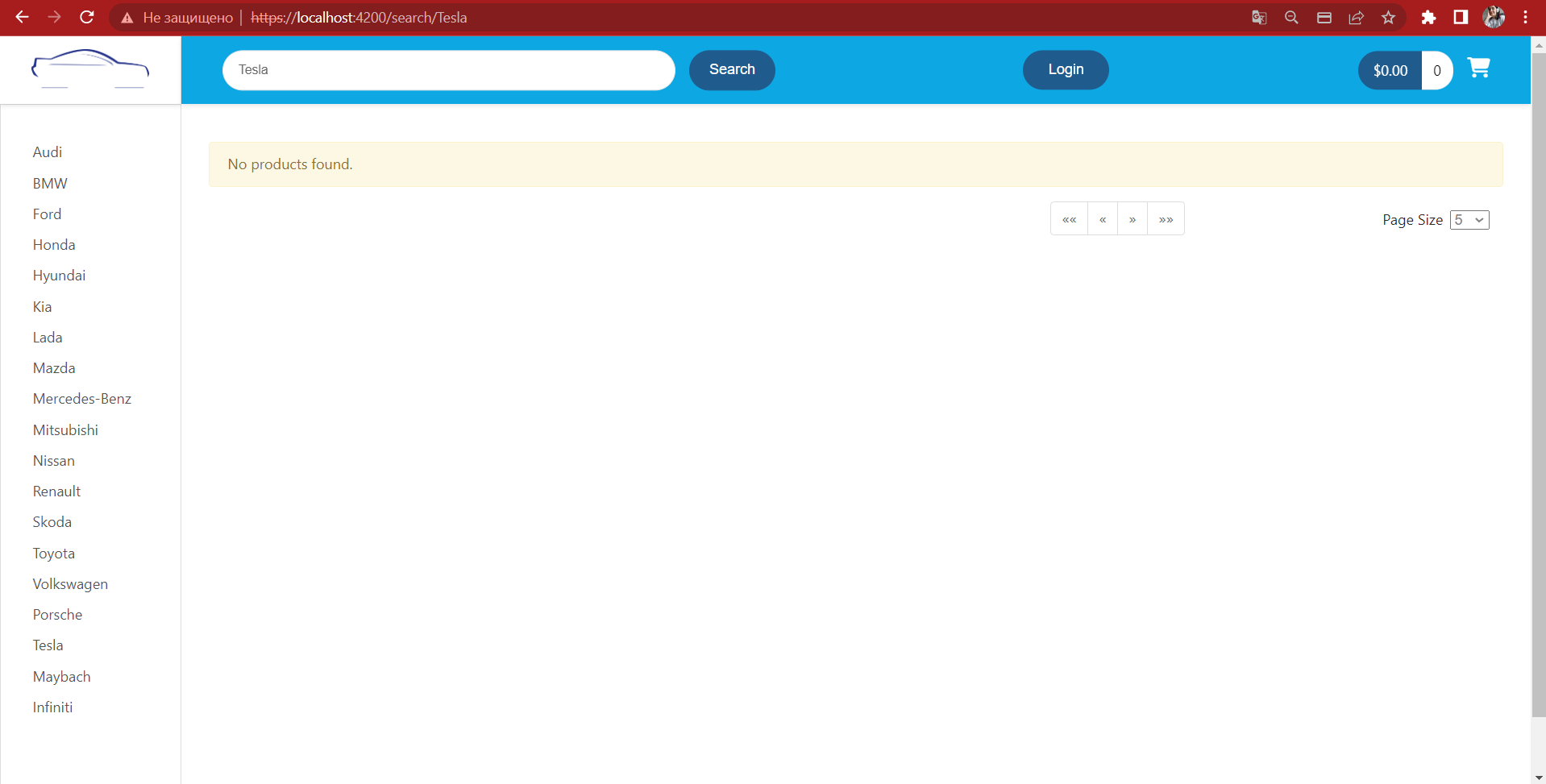
**Рисунок 4.10 – Уменьшение цифр в пагинации**

Так же в приложении имеется строка поиска товара, пользователь может написать как полное название товара или частичное, так и категорию, пример на рисунке 4.11.



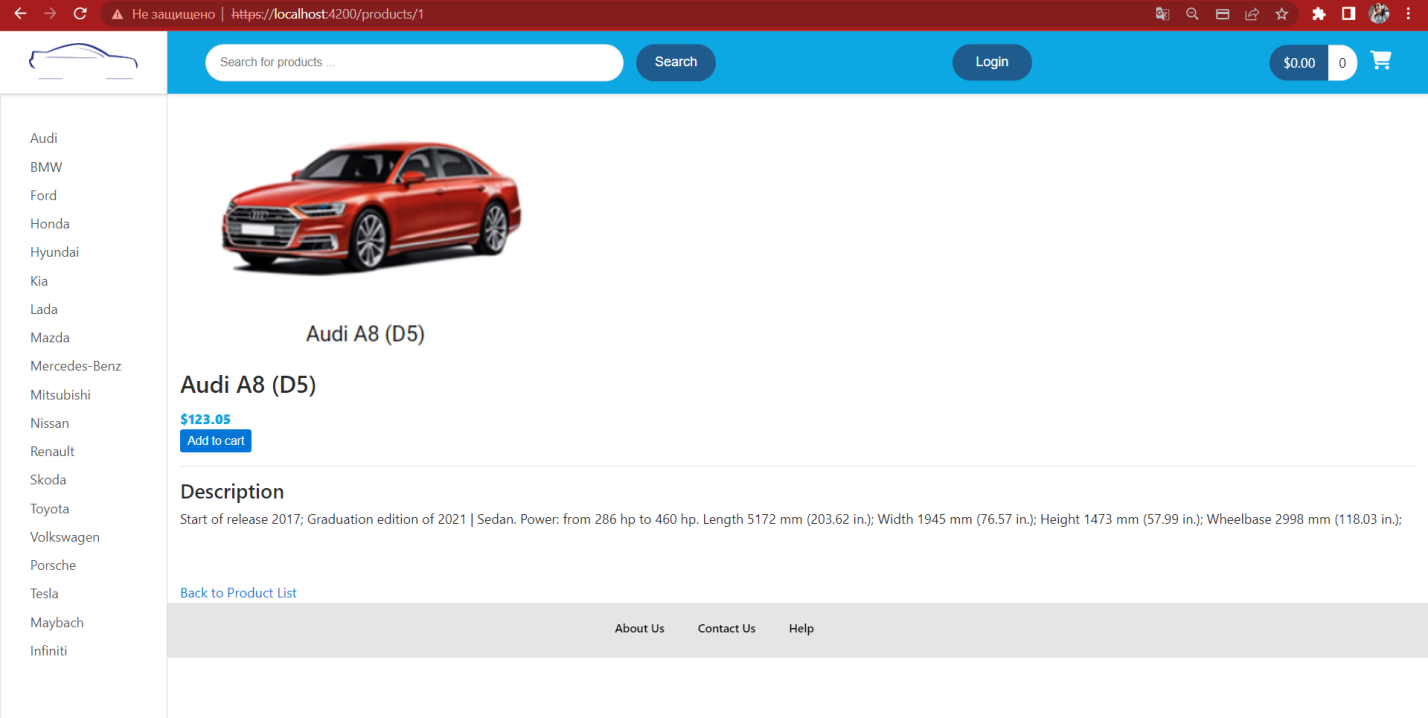
**Рисунок 4.11 – Строка поиска**

Если такого товара пока нет в магазине, то будет высвечена строка «No products found.», как показано на рисунке 4.12.



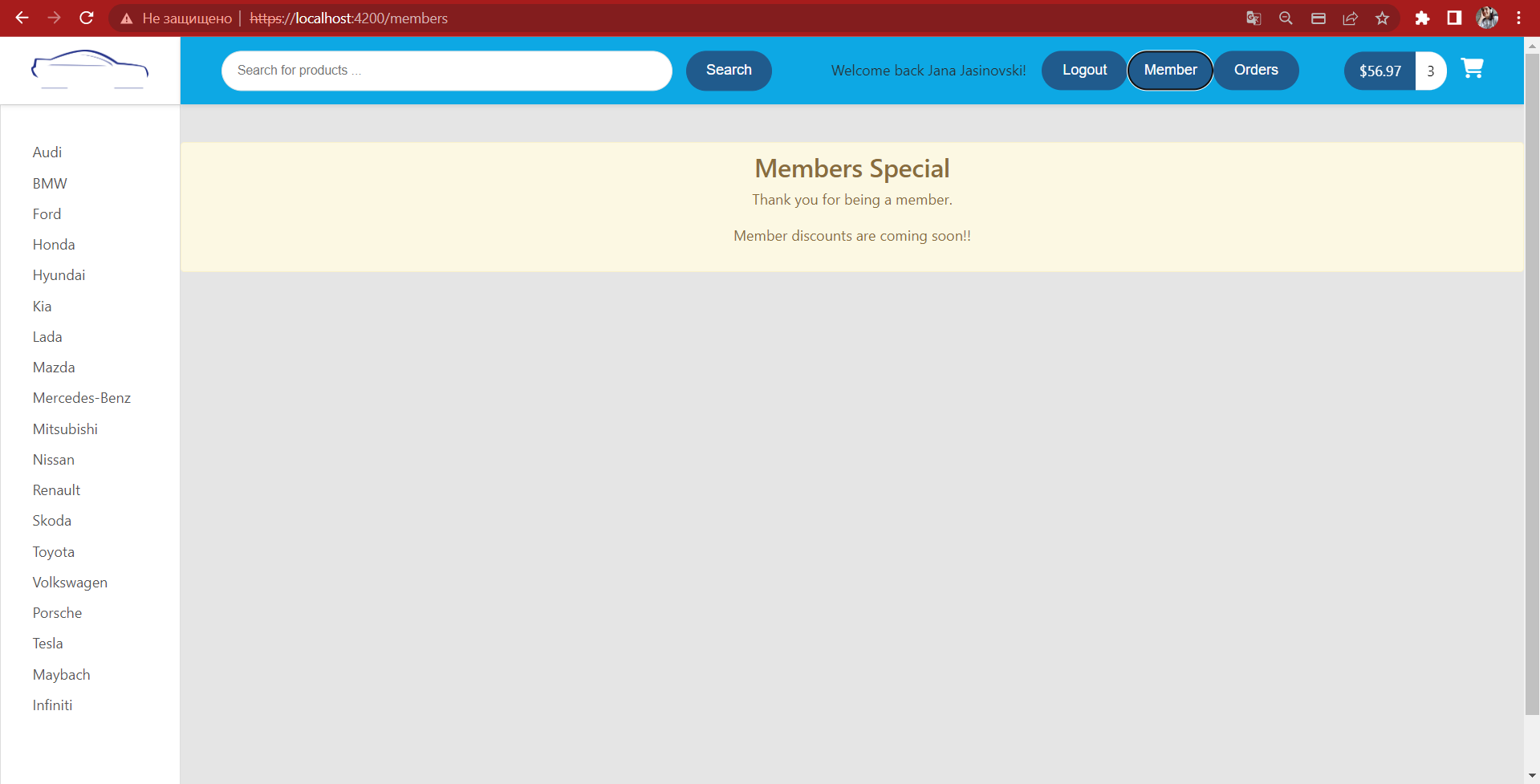
**Рисунок 4.12 – No products found**

Так же при нажатии на любой товар, пользователь может посмотреть описание товара, рисунок 4.13.



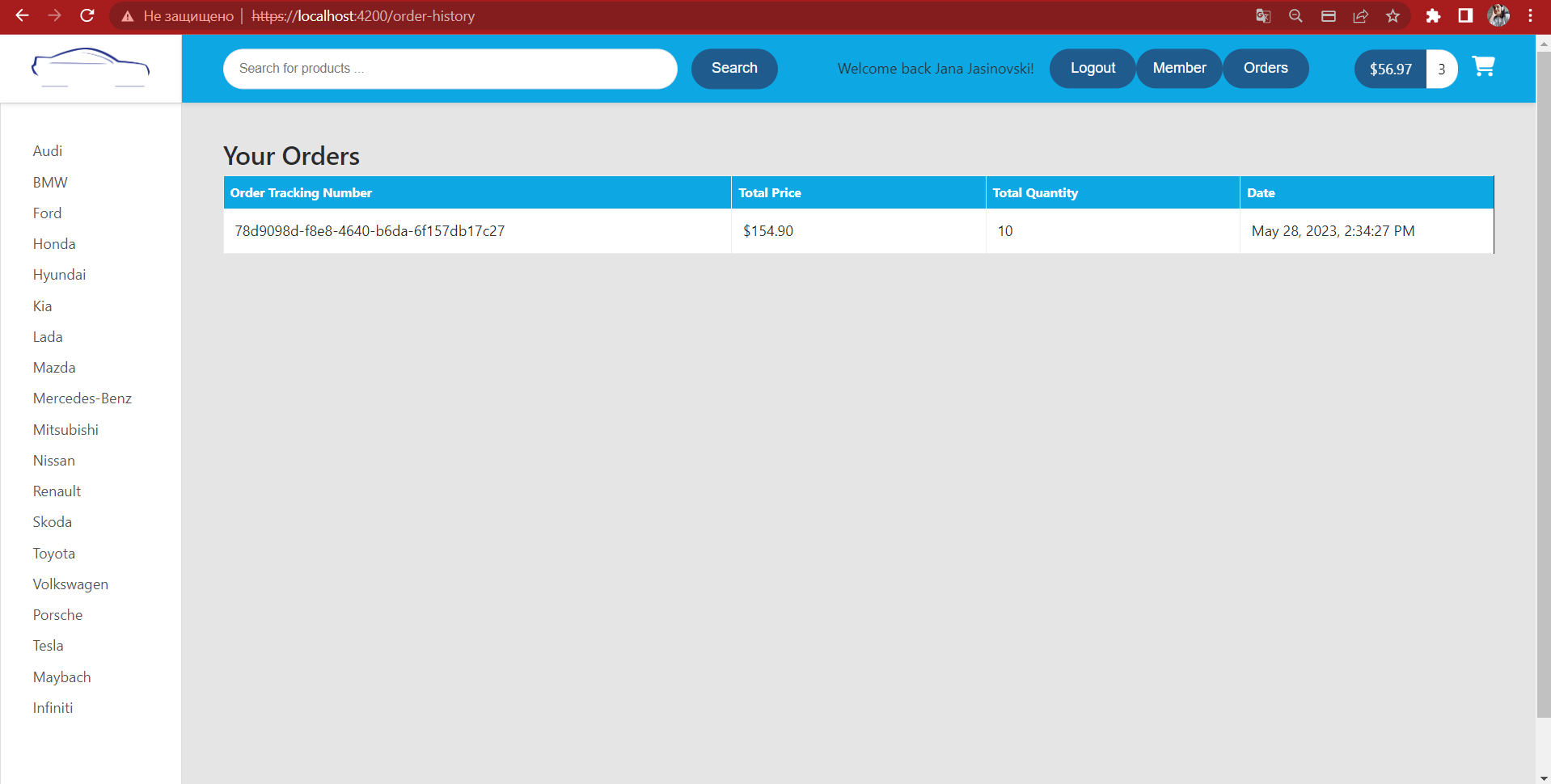
**Рисунок 4.13 – Описание товара**

У зарегистрированных пользователей так же есть возможность просматривать страницы будущих скидок и сделанных заказов. На рисунке 4.14 изображена страница будущих скидок.



**Рисунок 4.14 – Страница скидок**

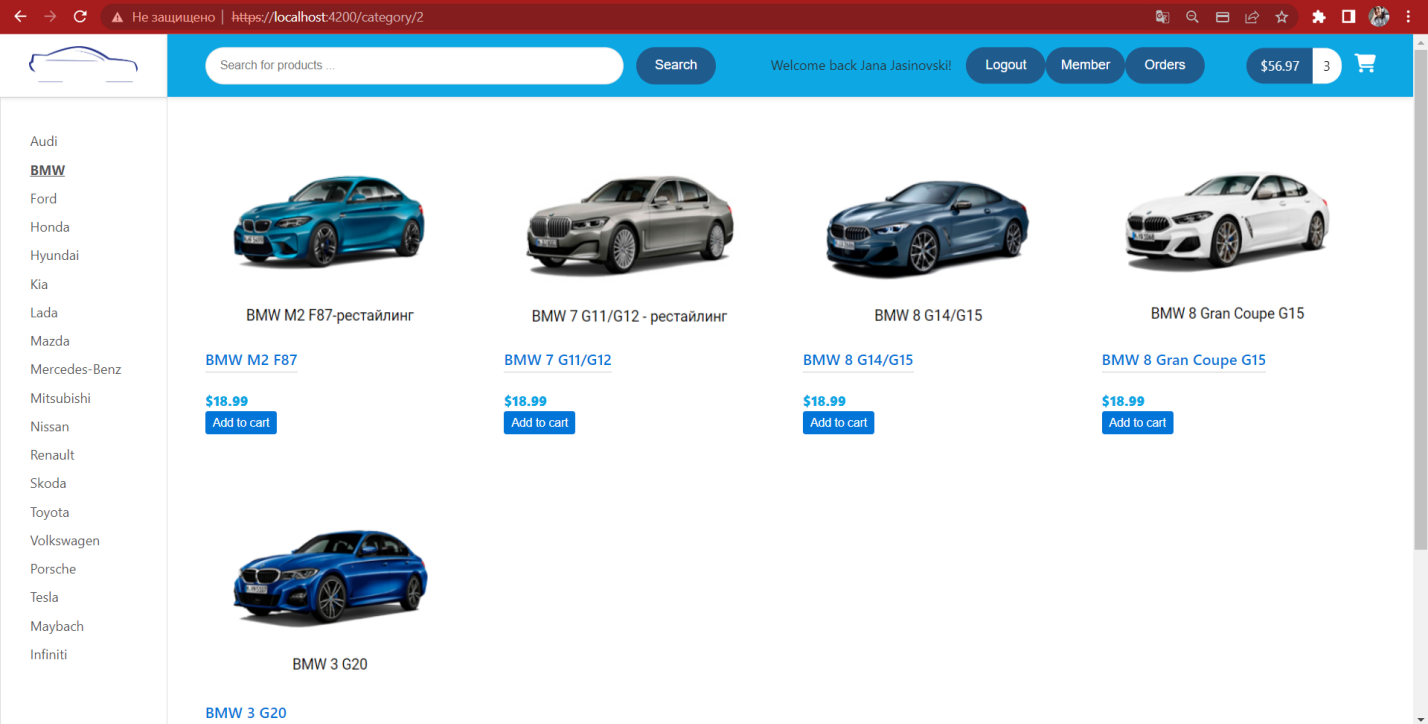
На рисунке 4.15 изображена страница заказов.



**Рисунок 4.15 – Страница заказов**

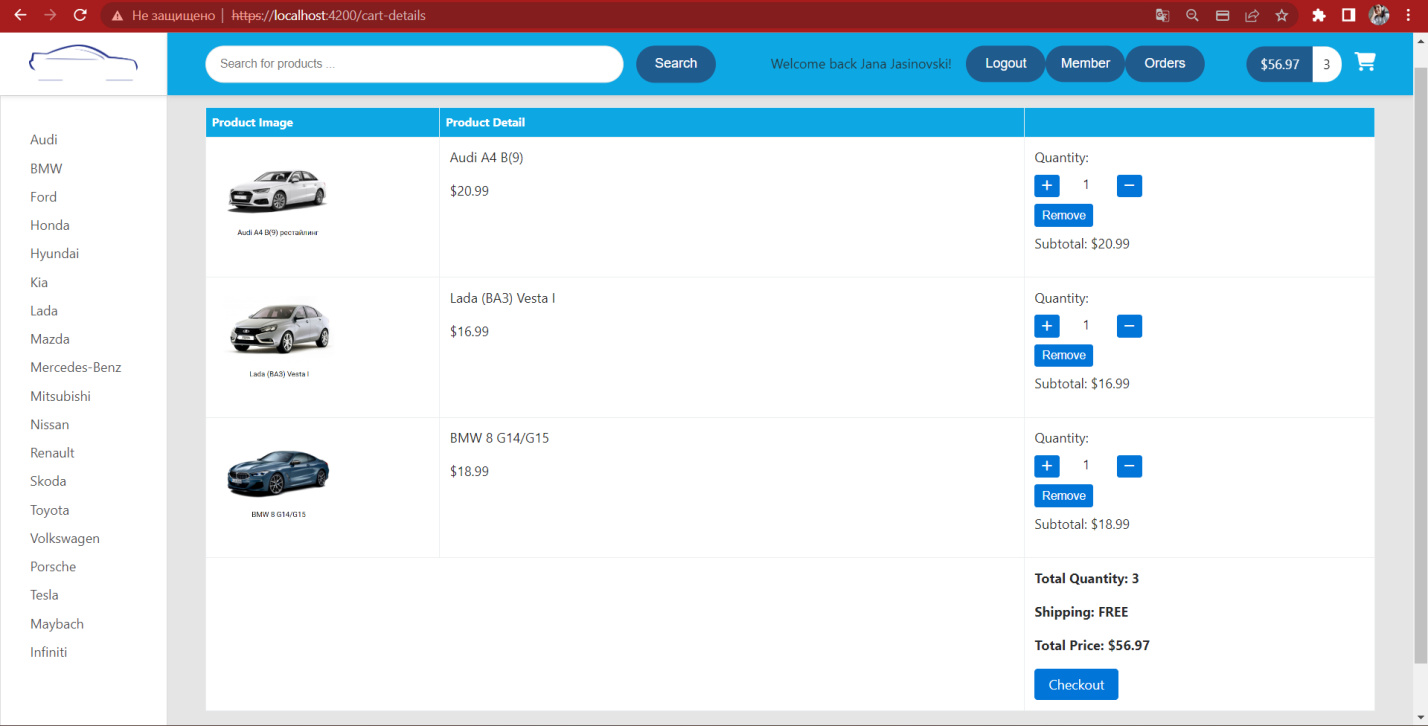
**4.3 Корзина товаров**

Пользователь может добавить понравившиеся товары в корзину, просто нажав на кнопку «Add to cart», тогда в правом верхнем углу у значка корзины появиться количество товаров в корзине и общая стоимость корзины, рисунок 4.16.



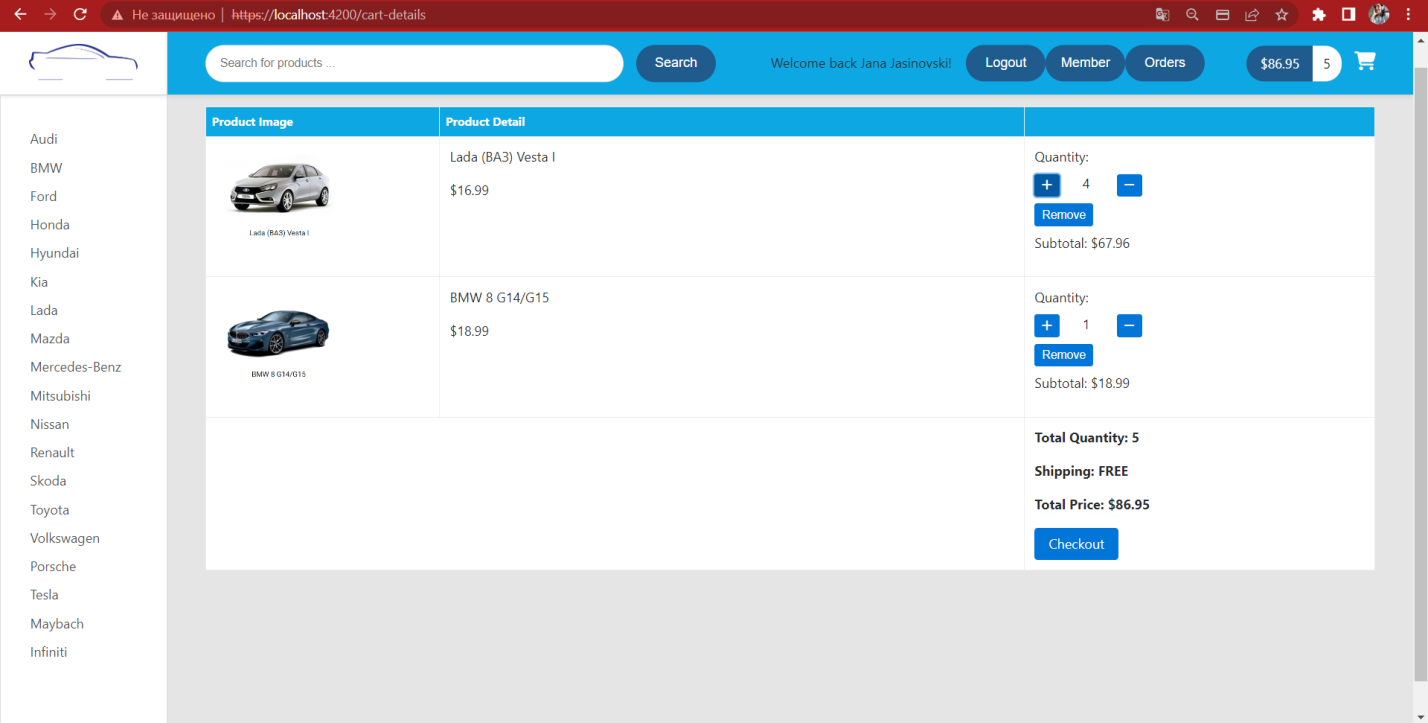
**Рисунок 4.16 – Добавление в корзину**

На рисунке 4.17 можно увидеть страницу корзины.



**Рисунок 4.17 – Страница корзины**

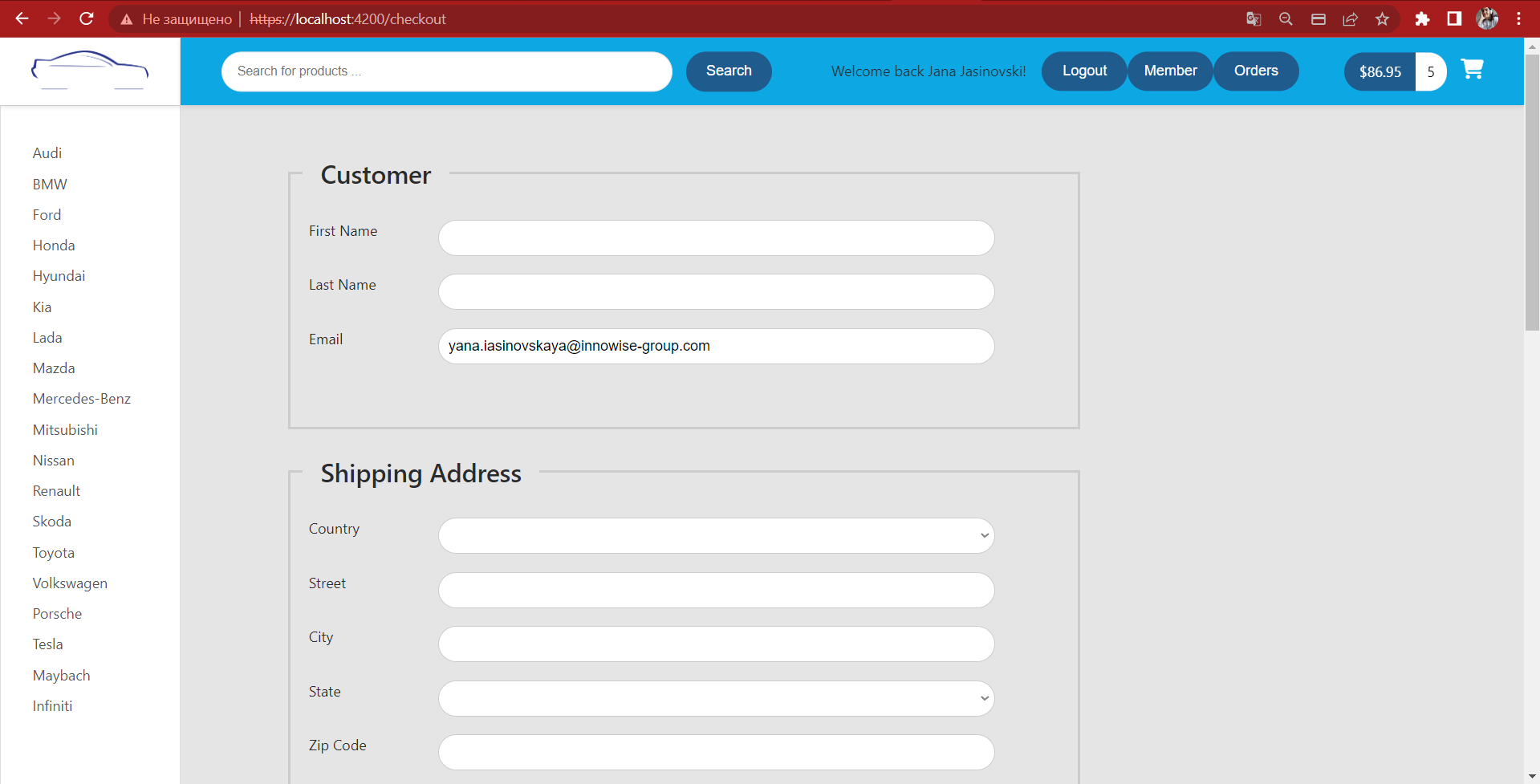
Нажав на выпадающий кнопки «+» и «–» можно инкрементировать и декрементировать количество данного товара в корзине. Нажав на кнопку «Remove» можно удалить товар из корзины. Причём Total Quantity и Total Price тоже изменяться, это можно наблюдать на рисунке 4.18.



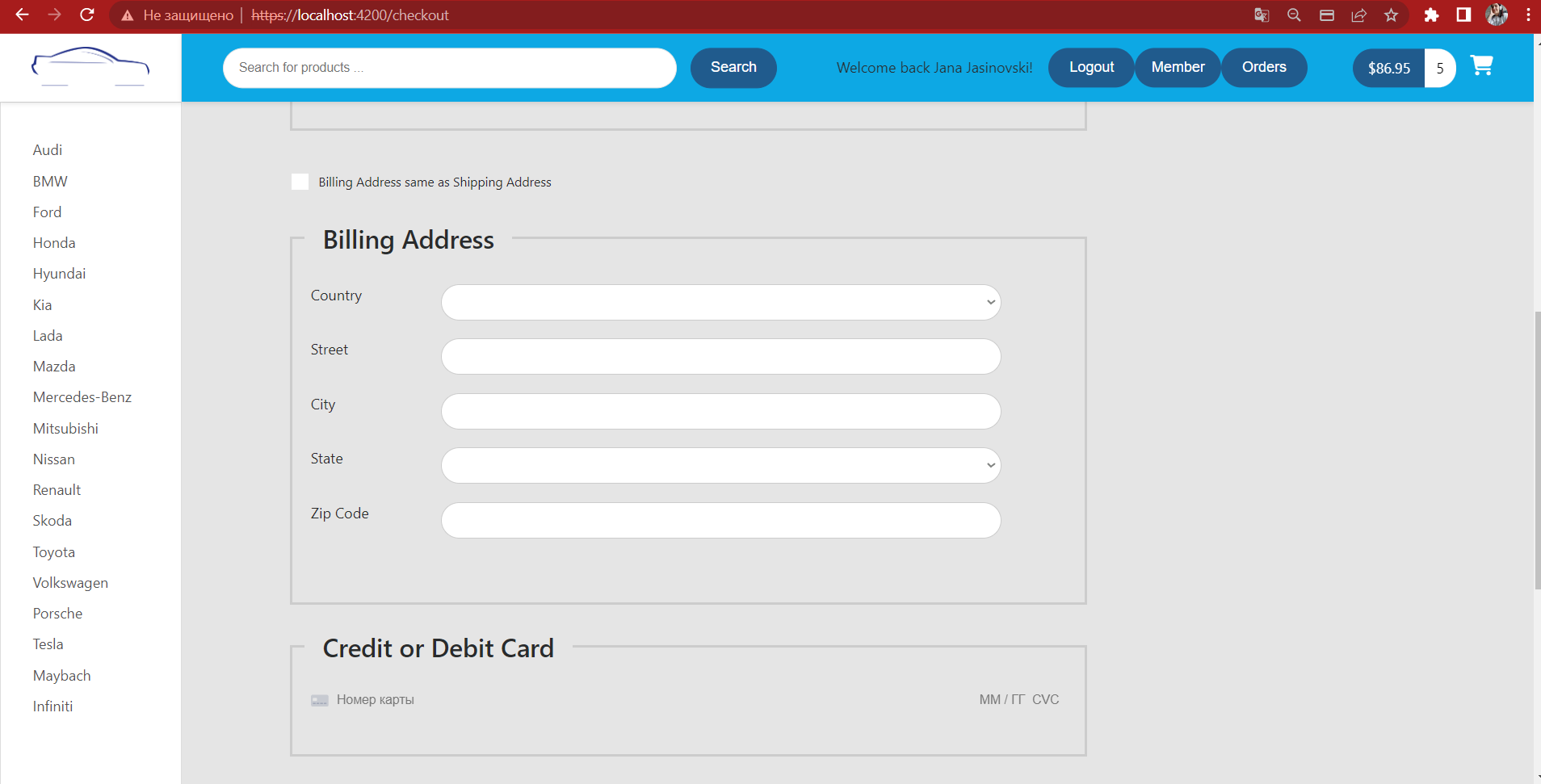
**Рисунок 4.18 – Функции в корзине**

**4.4 Заказы**

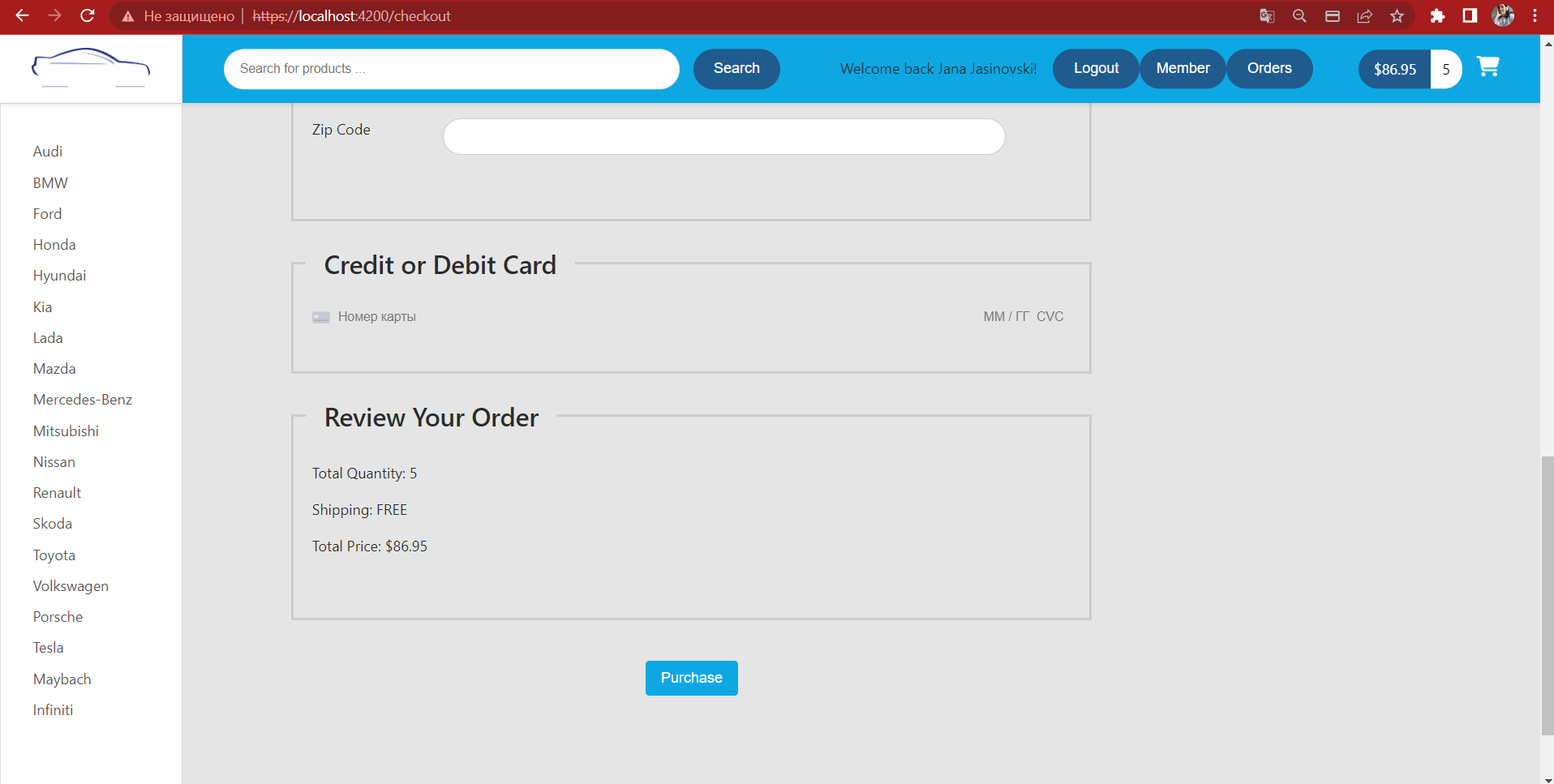
При нажатии на кнопку «Checkout» пользователь переходит на страницу с бланками для заказа товаров из корзины, рисунок 4.19 – 4.21.



**Рисунок 4.19 – Бланки для заказа**



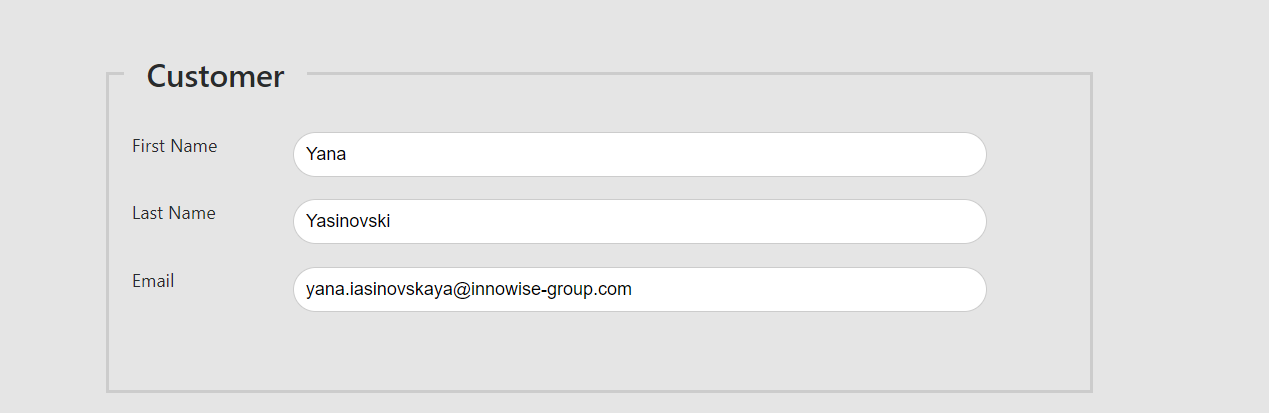
**Рисунок 4.20 – Бланки для заказа**



**Рисунок 4.21– Бланки для заказа**

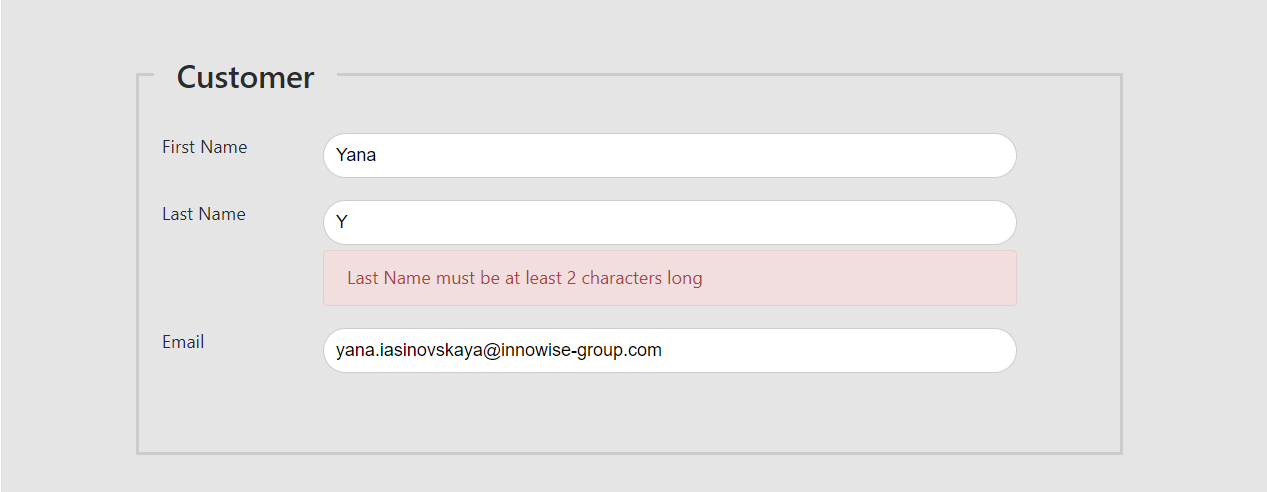
Всего у пользователя 4 бланка «Customer», «Shipping Address», «Billing Address» и «Credit or Debit card».

В бланке «Customer» пользователь заполняет свои имя и фамилия, а так же адрес электронной почты, рисунок 4.22.



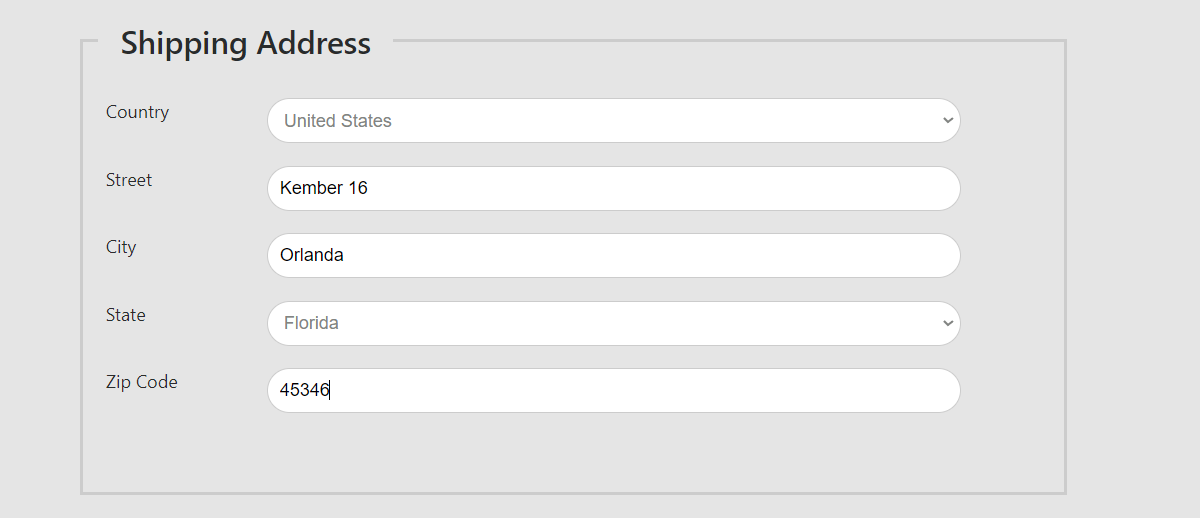
**Рисунок 4.22 – Заполнение бланка «Customer»**

Если пользователь неправильно введёт имя, фамилия или адрес электронной почты, то получит сообщение об некорректности, как например с фамилией, как на рисунке 4.23.



**Рисунок 4.23 – Некорректный ввод фамилии**

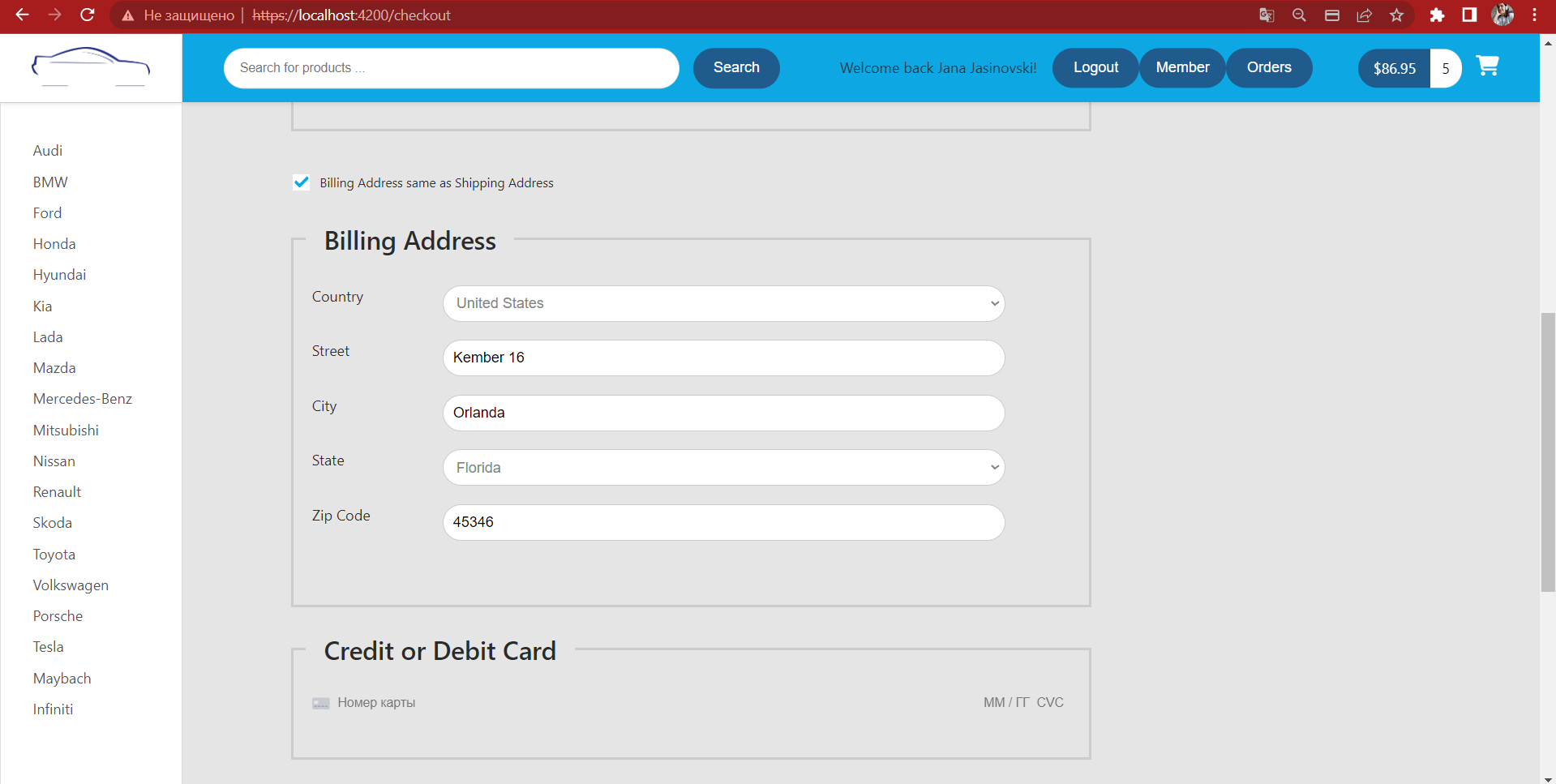
В бланке «Shipping Address» пользователь пользователь должен ввести данные, куда должна прийти доставка, рисунок 4.24.



**Рисунок 4.24 – Заполнение бланка «Shipping Address»**

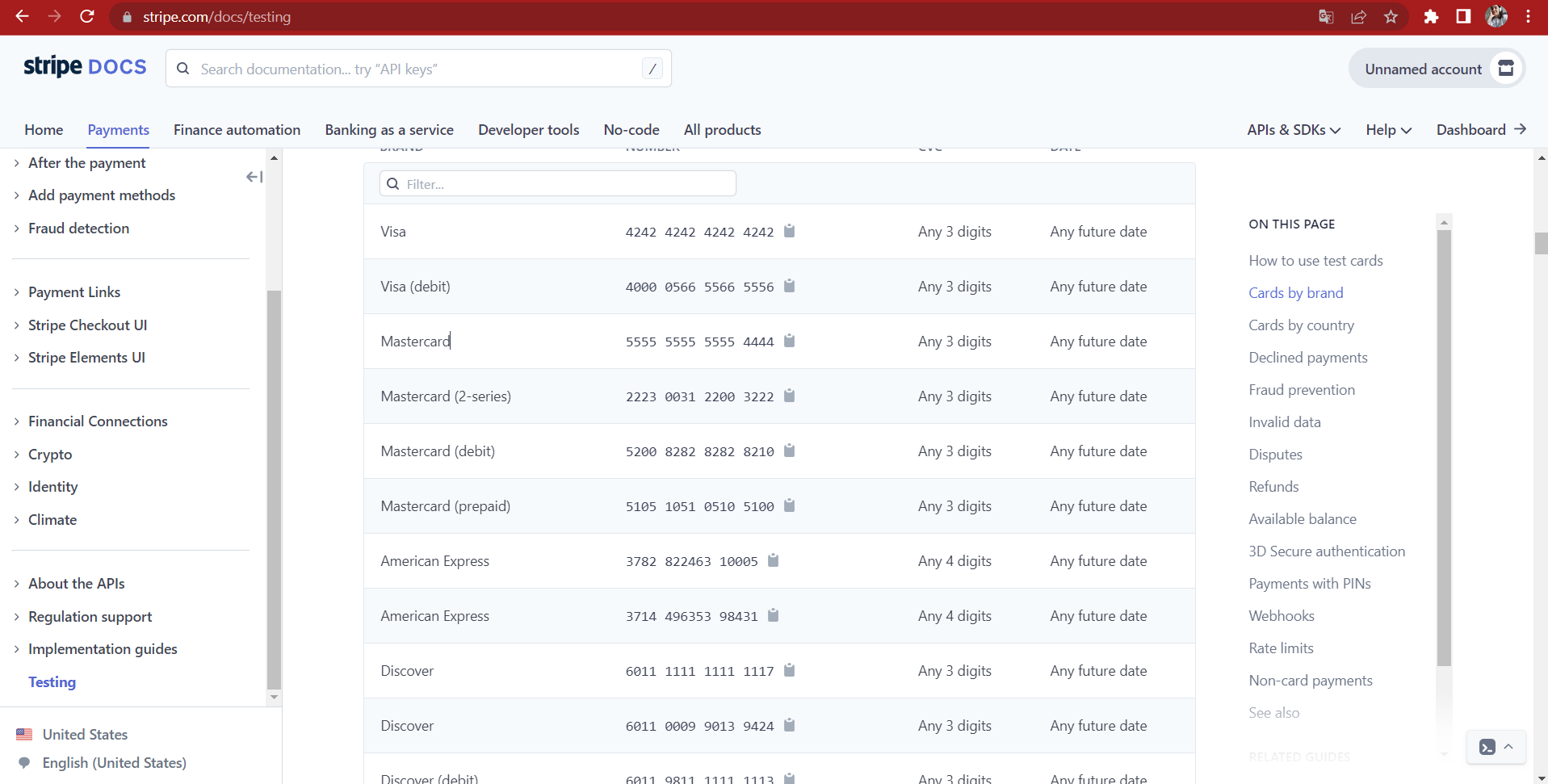
Здесь так же при некорректном вводе данных будет сообщаться об ошибке, как и в бланке «Customer».

В бланке «Billing Address» нужно ввести данные для выставления счёта. Если данные такие же как и в «Shipping Address», то можно поставить галочку в «Billing Address same as Shipping Address» и данные заполняться такими же , как и в «Shipping Address», что пользователь и сделал на рисунке 4.25.



**Рисунок 4.25 – Заполнение бланка «Billing Address»**

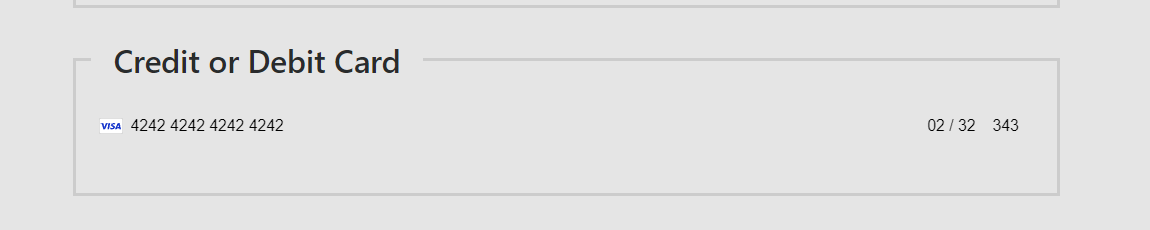
В бланке «Credit or Debit card» пользователь должен ввести свой номер банковской карты, а так же срок годности. Банковская карта сделана при помощи сайта Stripe, где можно увидеть все номера карточек, на рисунке 4.26.



**Рисунок 4.26 – Часть банковских карт, который нам предоставляет платформа Stripe**

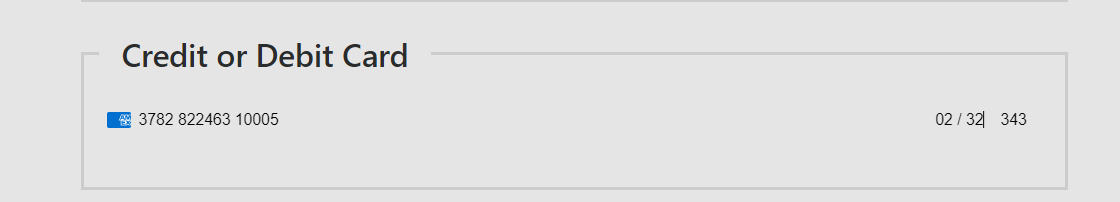
Возьмёт к примеру банковские карты «Visa» и «American Express», чтоб показать функционал нашего бланка «Credit or Debit card».

Банковская карта «Visa» показана на рисунке 4.27.



**Рисунок 4.27 – Банковская карта «Visa»**

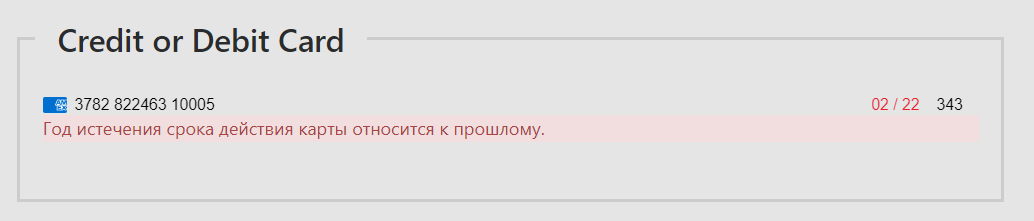
Банковкая карта «American Express» показана на рисунке 4.28.



**Рисунок 4.28 – Банковская карта «American Express»**

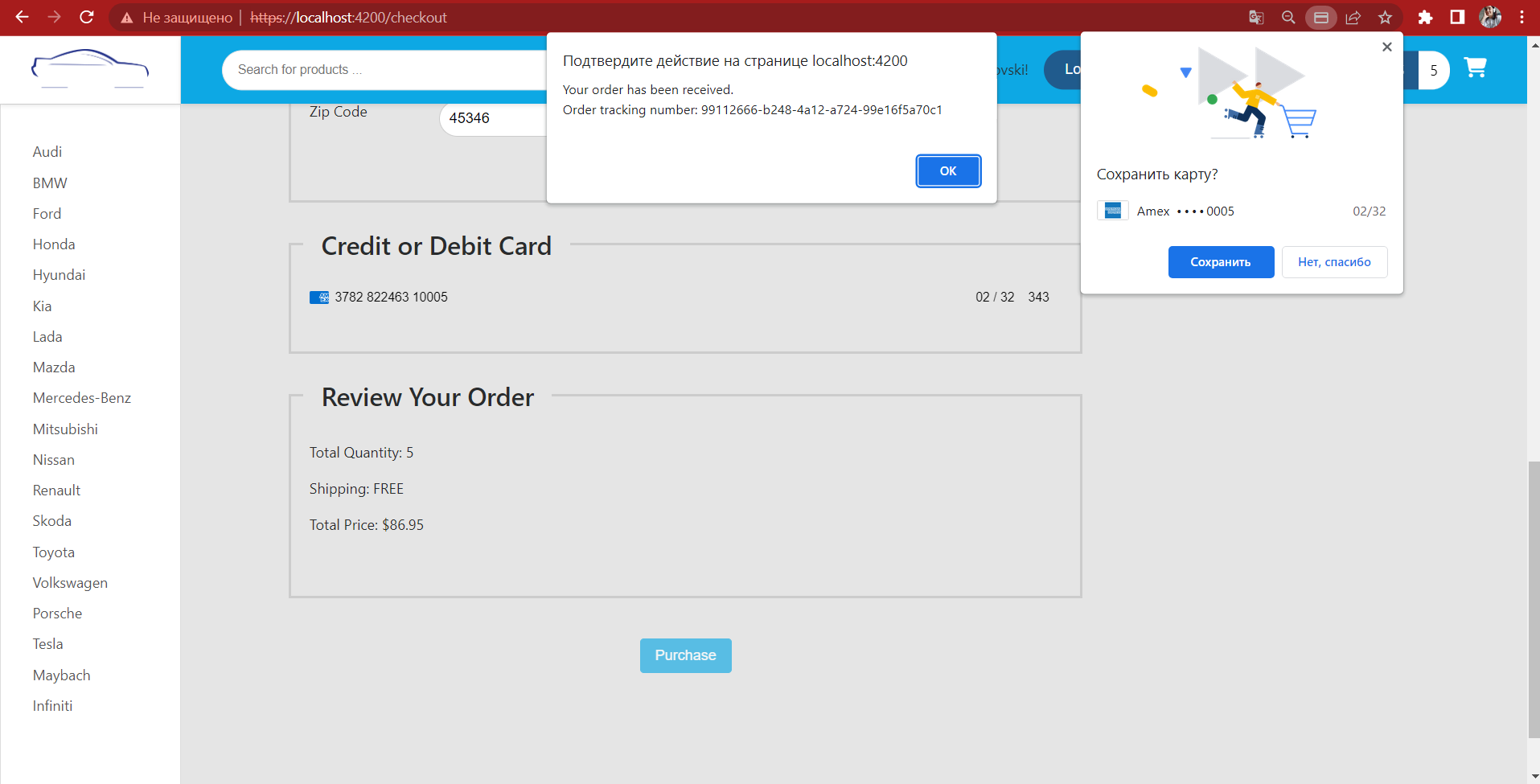
На рисунках 4.27 – 4.28 показано, как изменяется наша картинка банковских карт, от номера банковской карты.

Так же есть и валидация банковской карты, к примеру, если год карты истёк, а пользователь пытается банковскую карту ввести. Показано на рисунке 4.29.



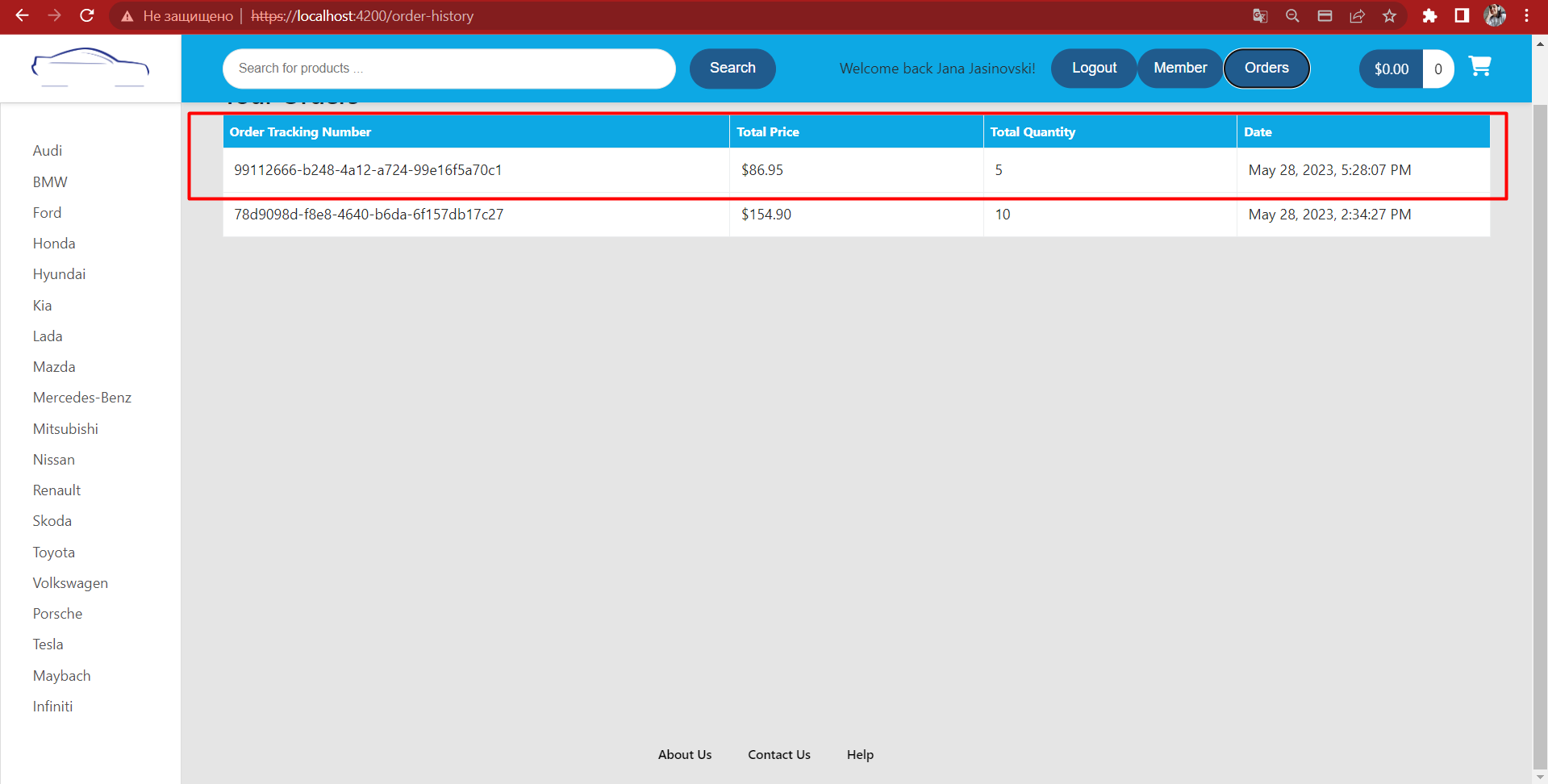
**Рисунок 4.29 – Валидация банковской карты**

После того, как пользователь успешно внёс все данных, он может нажать на кнопку «Purchase» и тем самым заказать выбранные товары. Высветится окошко номера заказа, как показано на рисунке 4.30.



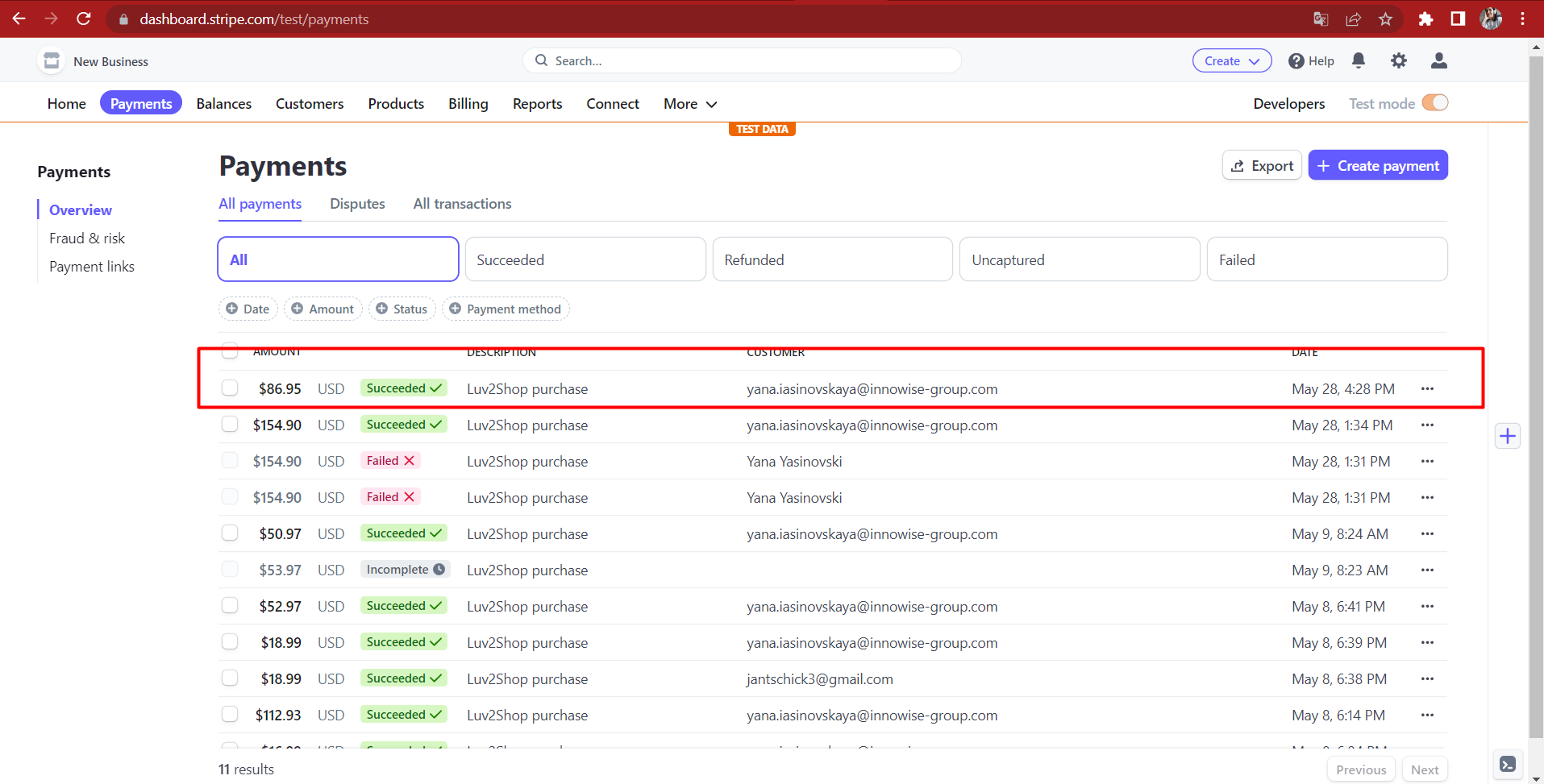
**Рисунок 4.30 – Номер заказа**

После заказа, корзина пользователя обнуляется и можно посмотреть номер заказа на страница «Orders» рисунок 4.31.



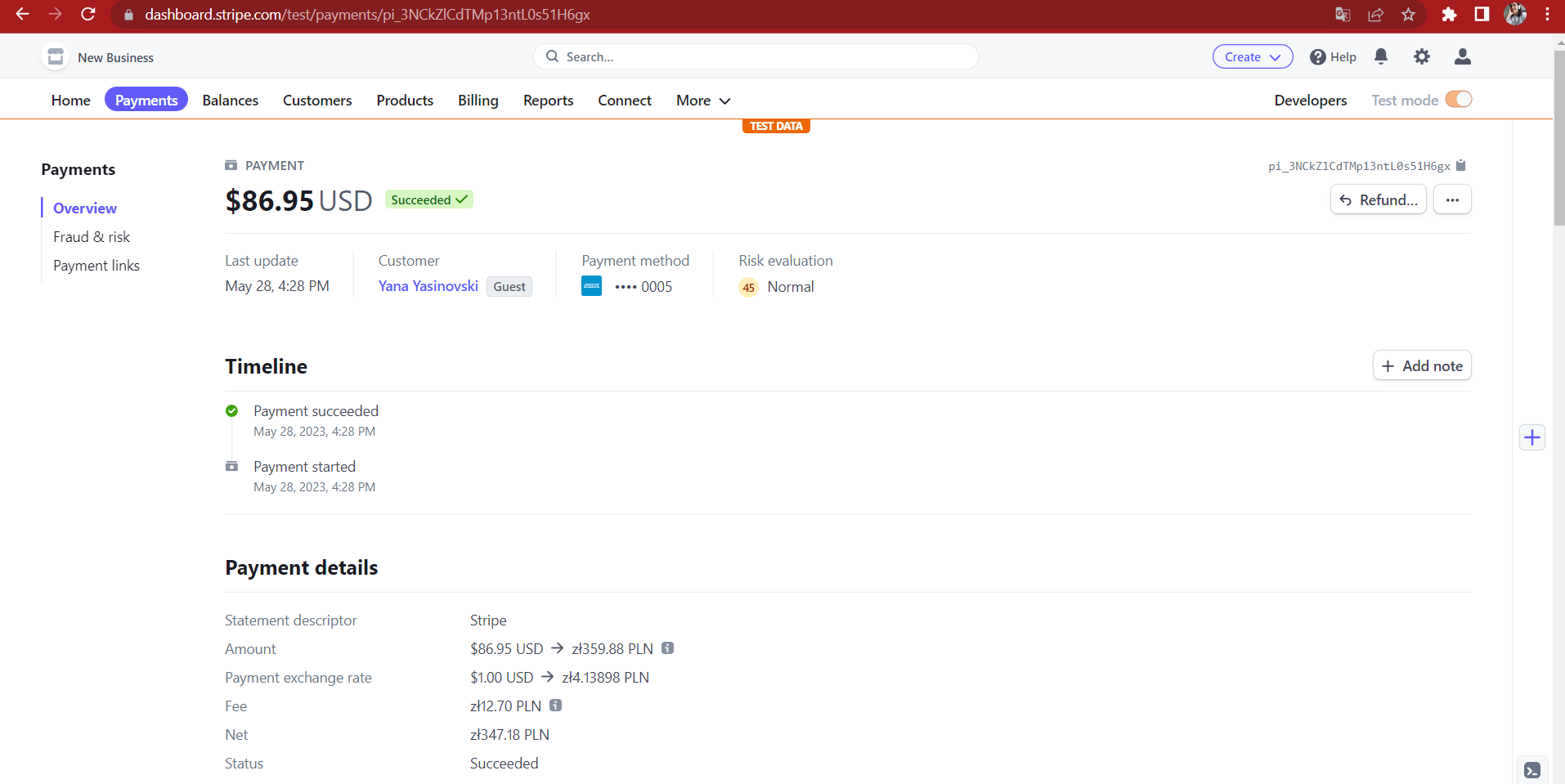
**Рисунок 4.31 – Новый заказ на сумму 86.95$**

Так же его заказ появиться на странице международного сайта заказов, показано на странице 4.32.

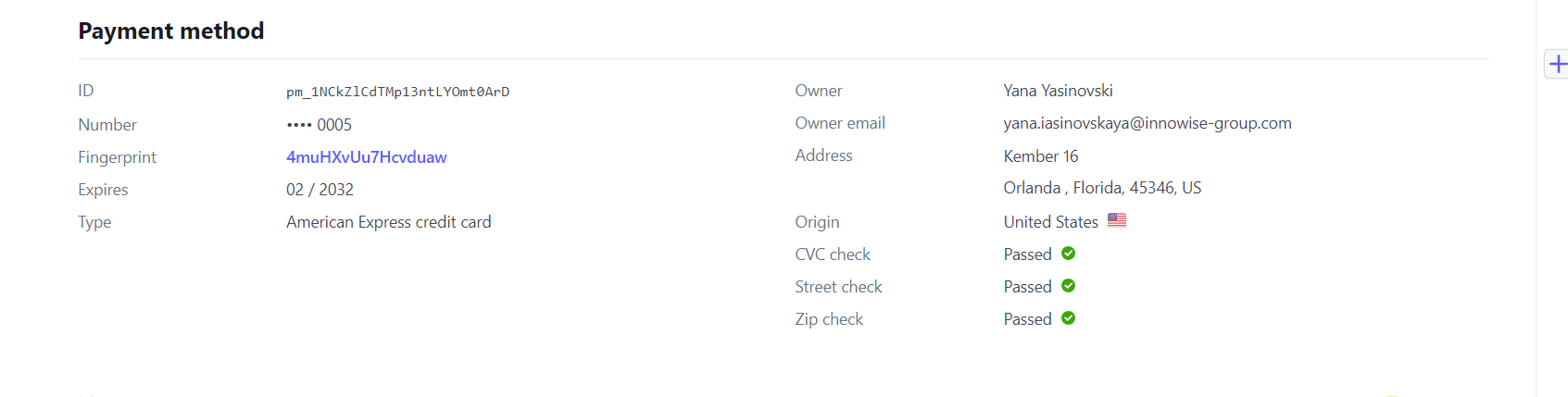


**Рисунок 4.32 – Международная страница заказов**

На страницах 4.33 – 4.34 можно увидеть кто заказал товар, куда заказал, и что за карту пользователь при этой использовал на страницах.

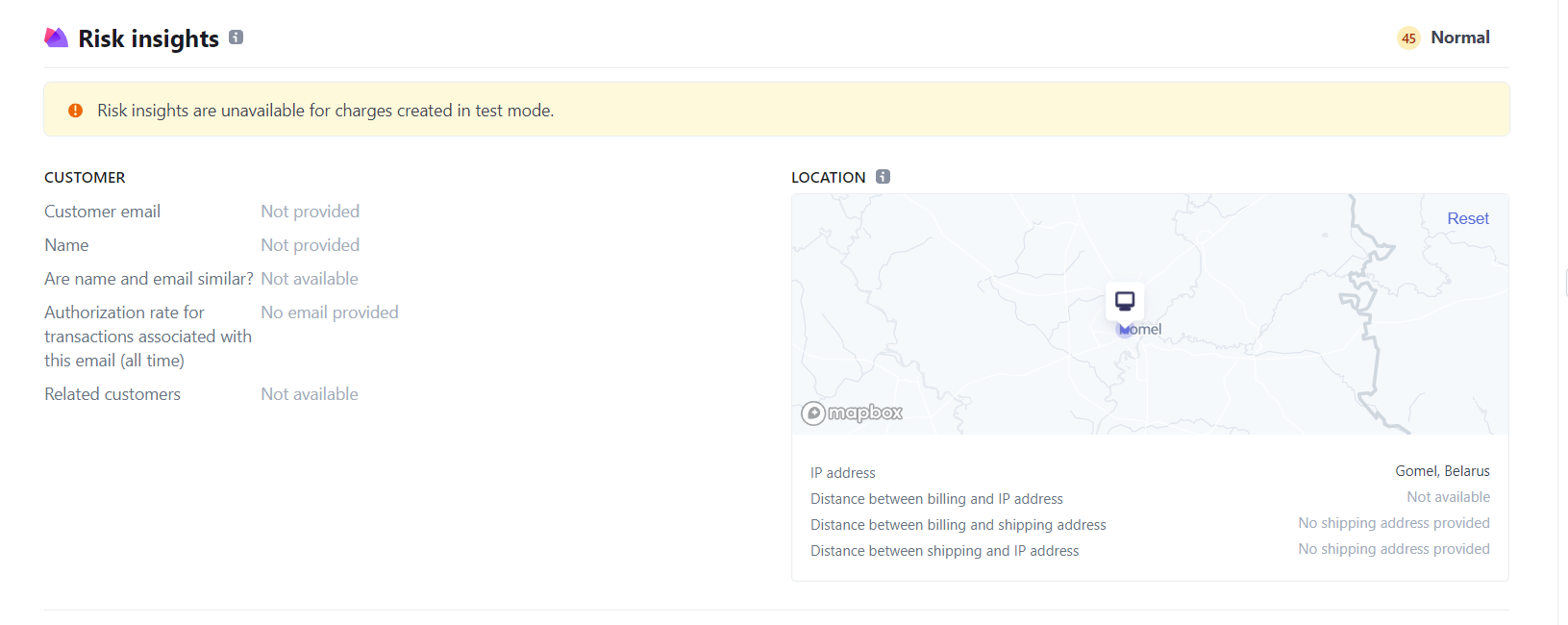


**Рисунок 4.33 – Информация о заказе**



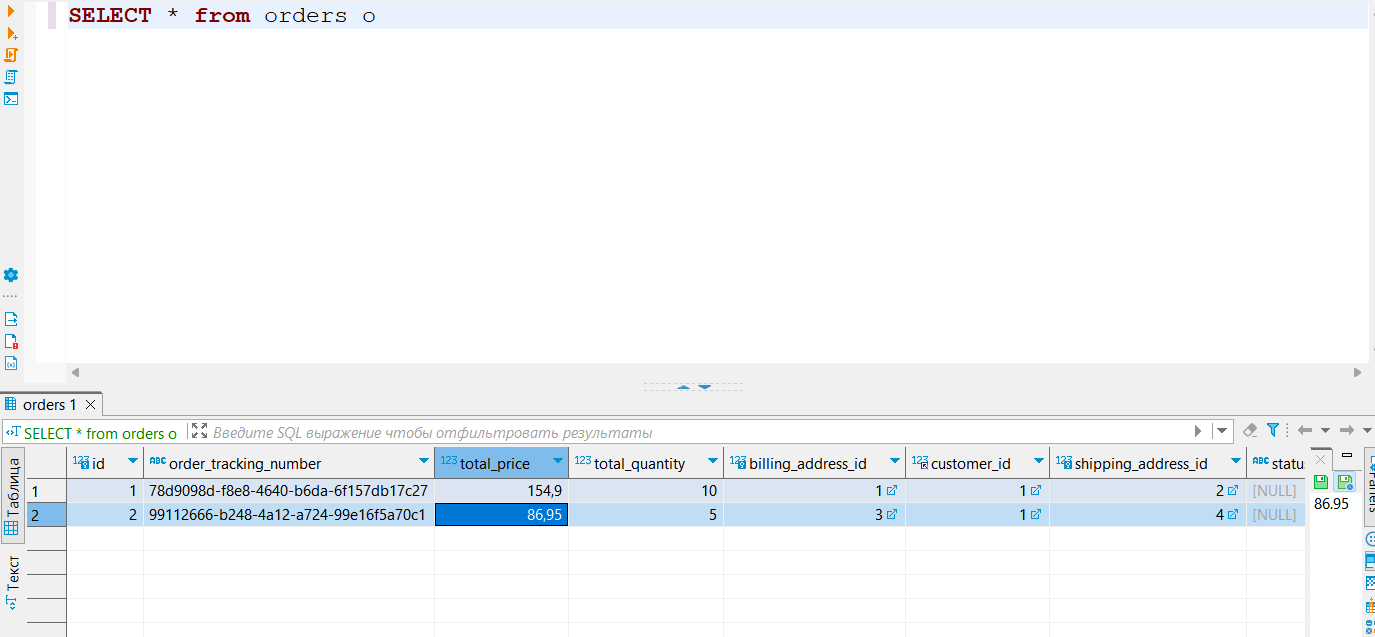
**Рисунок 4.34 – Информация о карте и доставке**

Так же можно посмотреть от куда был сделан заказ на рисунке 4.35.



**Рисунок 4.35 – Информация от куда был сделан заказ**

В базу данных так же был добавлен этот заказ, рисунок 4.36.



**Рисунок 4.36 – Добавления заказа**

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломной работы было создан интернет-магазин с использованием современных Angular Framework и Spring Framework, позволяющее настроить backend и frontend части при помощи языка разметки HTML, CSS, и MySQL, к которому приложение подключается при помощи Spring Framework. Весь интерфейс написан на языке HTML, CSS, взаимодействие с базой данных осуществляется через Angular Framework. Для отображения информации было реализована привязка данных на соответствующих страницах приложения: Продукты, Категории, Авторизация, Корзина и Заказы. На основе данных интернет-магазина можно построить несколько микросервисов с разными базами данных и языками программирования. В самой программе реализованы все необходимые модели и классы.

В ходе выполнения дипломной работы были решены следующие задачи:

- изучен ORM фреймворка Spring Framework;

- разработана структура базы данных через классы;

- углубление знаний Angular Framework для построения визуальной части интернет -магазина;

- изучены языки разметки HTML и CSS;

- использовался дополнительные функционалы, таких как Okta и Stripe.

В результате было успешно создан интернет-мазазин, который удовлетворяет требованиям и ожиданиям пользователей. Интернет-магазин обладает привлекательным пользовательским интерфейсом, позволяет заказывать необходимый товар, предоставляет актуальную информацию о товаре, а также гарантирует безопасность пользовательских данных.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. TypeScript: JavaScript With Syntax For Types. [Электронный ресурс]. – Режим доступа: https://www.typescriptlang.org/ – Дата доступа 20.05.2023.
2. Фрейн. HTML5 и CSS. Разработка сайтов для любых браузеров и устройств / Б. Фрейн. – Питер: Apress, 2014. – 304с.
3. Руководство по языку программирования Java [Электронный ресурс]. – Режим доступа: <https://metanit.com/java/tutorial/> – Дата доступа: 27.03.2023.
4. Craig Walls, Spring in Action (5th Edition) / Craig Walls. – Лондон: Apress, 2022. – 1253 с.
5. Docker: Accelerated, Containerized Application Development [Электронный ресурс]. – Режим доступа: https://www.docker.com/ – Дата доступа: 01.05.2023.
6. The Liquibase Community | The Database DevOps Community [Электронный ресурс]. – Режим доступа: <https://www.liquibase.org/> – Дата доступа: 01.05.2023.
7. Groff, Weinberg, Oppel. SQL The Complete Reference / J. Groff, P.Weinberg, A. Oppel. – Москва, 2015. – 959с.
8. Identity | Okta [Электронный ресурс]. – Режим доступа: https://www.okta.com/ – Дата доступа: 27.03.2023.
9. Stripe | Payment Processing Platform for the Internet [Электронный ресурс]. – Режим доступа: https://stripe.com/ – Дата доступа: 20.04.2023.

ПРИЛОЖЕНИЕ А

**Фрагмент программного код приложения**

//CheckoutServiceImpl.java

package com.luv2code.ecommerce.service;

import com.luv2code.ecommerce.dao.CustomerRepository;

import com.luv2code.ecommerce.dto.PaymentInfo;

import com.luv2code.ecommerce.dto.Purchase;

import com.luv2code.ecommerce.dto.PurchaseResponse;

import com.luv2code.ecommerce.entity.Customer;

import com.luv2code.ecommerce.entity.Order;

import com.luv2code.ecommerce.entity.OrderItem;

import com.stripe.Stripe;

import com.stripe.exception.StripeException;

import com.stripe.model.PaymentIntent;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.stereotype.Service;

import javax.transaction.Transactional;

import java.util.\*;

@Service

public class CheckoutServiceImpl implements CheckoutService {

private CustomerRepository customerRepository;

public CheckoutServiceImpl(CustomerRepository customerRepository,

@Value("${stripe.key.secret}") String secretKey) {

this.customerRepository = customerRepository;

Stripe.apiKey = secretKey;

}

@Override

@Transactional

public PurchaseResponse placeOrder(Purchase purchase) {

Order order = purchase.getOrder();

String orderTrackingNumber = generateOrderTrackingNumber();

order.setOrderTrackingNumber(orderTrackingNumber);

Set<OrderItem> orderItems = purchase.getOrderItems();

orderItems.forEach(item -> order.add(item));

order.setBillingAddress(purchase.getBillingAddress());

order.setShippingAddress(purchase.getShippingAddress());

Customer customer = purchase.getCustomer();

String theEmail = customer.getEmail();

Customer customerFromDB = customerRepository.findByEmail(theEmail);

if (customerFromDB != null) {

customer = customerFromDB;

}

customer.add(order);

customerRepository.save(customer);

return new PurchaseResponse(orderTrackingNumber);

}

@Override

public PaymentIntent createPaymentIntent(PaymentInfo paymentInfo) throws StripeException {

List<String> paymentMethodTypes = new ArrayList<>();

paymentMethodTypes.add("card");

Map<String, Object> params = new HashMap<>();

params.put("amount", paymentInfo.getAmount());

params.put("currency", paymentInfo.getCurrency());

params.put("payment\_method\_types", paymentMethodTypes);

params.put("description", "Luv2Shop purchase");

params.put("receipt\_email", paymentInfo.getReceiptEmail());

return PaymentIntent.create(params);

}

private String generateOrderTrackingNumber() {

return UUID.randomUUID().toString();

}

}

//MyDateRestConfig.java

package com.luv2code.ecommerce.config;

import com.luv2code.ecommerce.entity.Country;

import com.luv2code.ecommerce.entity.Order;

import com.luv2code.ecommerce.entity.Product;

import com.luv2code.ecommerce.entity.ProductCategory;

import com.luv2code.ecommerce.entity.State;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.context.annotation.Configuration;

import org.springframework.data.rest.core.config.RepositoryRestConfiguration;

import org.springframework.data.rest.webmvc.config.RepositoryRestConfigurer;

import org.springframework.http.HttpMethod;

import org.springframework.web.servlet.config.annotation.CorsRegistry;

import javax.persistence.EntityManager;

import javax.persistence.metamodel.EntityType;

import java.util.ArrayList;

import java.util.List;

import java.util.Set;

@Configuration

public class MyDataRestConfig implements RepositoryRestConfigurer {

@Value("${allowed.origins}")

private String[] theAllowedOrigins;

private EntityManager entityManager;

@Autowired

public MyDataRestConfig(EntityManager theEntityManager) {

entityManager = theEntityManager;

}

@Override

public void configureRepositoryRestConfiguration(RepositoryRestConfiguration config, CorsRegistry cors) {

HttpMethod[] theUnsupportedActions = {HttpMethod.PUT, HttpMethod.POST, HttpMethod.DELETE, HttpMethod.PATCH};

disableHttpMethods(Product.class, config, theUnsupportedActions);

disableHttpMethods(ProductCategory.class, config, theUnsupportedActions);

disableHttpMethods(Country.class, config, theUnsupportedActions);

disableHttpMethods(State.class, config, theUnsupportedActions);

disableHttpMethods(Order.class, config, theUnsupportedActions);

exposeIds(config);

cors.addMapping(config.getBasePath() + "/\*\*").allowedOrigins(theAllowedOrigins);

}

private void disableHttpMethods(Class theClass, RepositoryRestConfiguration config, HttpMethod[] theUnsupportedActions) {

config.getExposureConfiguration()

.forDomainType(theClass)

.withItemExposure((metdata, httpMethods) -> httpMethods.disable(theUnsupportedActions))

.withCollectionExposure((metdata, httpMethods) -> httpMethods.disable(theUnsupportedActions));

}

private void exposeIds(RepositoryRestConfiguration config) {

Set<EntityType<?>> entities = entityManager.getMetamodel().getEntities();

List<Class> entityClasses = new ArrayList<>();

for (EntityType tempEntityType : entities) {

entityClasses.add(tempEntityType.getJavaType());

}

Class[] domainTypes = entityClasses.toArray(new Class[0]);

config.exposeIdsFor(domainTypes);

}

}

// Product.java

package com.luv2code.ecommerce.entity;

import lombok.Data;

import org.hibernate.annotations.CreationTimestamp;

import org.hibernate.annotations.UpdateTimestamp;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import javax.persistence.Table;

import java.math.BigDecimal;

import java.util.Date;

@Entity

@Table(name="product")

@Data

public class Product {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "id")

private Long id;

@ManyToOne

@JoinColumn(name = "category\_id", nullable = false)

private ProductCategory category;

@Column(name = "sku")

private String sku;

@Column(name = "name")

private String name;

@Column(name = "description")

private String description;

@Column(name = "unit\_price")

private BigDecimal unitPrice;

@Column(name = "image\_url")

private String imageUrl;

@Column(name = "active")

private boolean active;

@Column(name = "units\_in\_stock")

private int unitsInStock;

@Column(name = "date\_created")

@CreationTimestamp

private Date dateCreated;

@Column(name = "last\_updated")

@UpdateTimestamp

private Date lastUpdated;

}

//product-list-grid.component.html

<div class="main-content" style="background-color: white;">

<div class="section-content section-content-p30">

<div class="container-fluid">

<div class="row">

<div \*ngFor="let tempProduct of products" class="col-md-3">

<div class="product-box">

<a routerLink="/products/{{ tempProduct.id }}">

<img src="{{ tempProduct.imageUrl }}" class="img-responsive" style="width: 300px; height: 250px;">

</a>

<div><a routerLink="/products/{{ tempProduct.id }}">

<h1>{{ tempProduct.name }}</h1>

</a></div>

<div class="price">{{ tempProduct.unitPrice | currency:'USD' }}</div>

<button (click)="addToCart(tempProduct)" class="btn btn-primary btn-sm">Add to cart</button>

</div>

</div>

<div \*ngIf="products?.length == 0" class="alert alert-warning col-md-12" role="alert">

No products found.

</div>

</div>

<div class="footer-pagination">

<div class="row">

<div class="col-md-6"></div>

<div class="col-md-6">

<div class="row">

<div class="col-md-9" style="padding-left: 30%">

<ngb-pagination [(page)]="thePageNumber"

[pageSize]="thePageSize"

[collectionSize]="theTotalElements"

[maxSize]="5"

[boundaryLinks]="true"

(pageChange)="listProducts()">

</ngb-pagination>

</div>

<div class="col-md-3 mt-2" style="text-align: right;">

<span class="mr-2">Page Size</span>

<select #myPageSelect (change)="updatePageSize(myPageSelect.value)">

<option>2</option>

<option selected="true">5</option>

<option>10</option>

<option>20</option>

<option>50</option>

</select>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

</div>

//checkout.compoent.html

<div class="main-content page-m">

<div class="section-content section-content-p30">

<div class="container-fluid">

<form [formGroup]="checkoutFormGroup" (ngSubmit)="onSubmit()">

<!-- customer form group -->

<div formGroupName="customer" class="form-area">

<h3>Customer</h3>

<div class="row">

<div class="col-md-2"> <label>First Name</label></div>

<div class="col-md-9">

<div class="input-space">

<input formControlName="firstName" type="text">

<div \*ngIf="firstName?.invalid && (firstName?.dirty || firstName?.touched)" class="alert alert-danger mt-1">

<div \*ngIf="firstName?.errors?.['required'] || firstName?.errors?.['notOnlyWhitespace']">

First Name is required

</div>

<div \*ngIf="firstName?.errors?.['minlength']">

First Name must be at least 2 characters long

</div>

</div>

</div>

</div>

</div>

<div class="row">

<div class="col-md-2"> <label>Last Name</label></div>

<div class="col-md-9">

<div class="input-space">

<input formControlName="lastName" type="text">

<div \*ngIf="lastName?.invalid && (lastName?.dirty || lastName?.touched)" class="alert alert-danger mt-1">

<div \*ngIf="lastName?.errors?.['required'] || lastName?.errors?.['notOnlyWhitespace']">

Last Name is required

</div>

<div \*ngIf="lastName?.errors?.['minlength']">

Last Name must be at least 2 characters long

</div>

</div>

</div>

</div>

</div>

<div class="row">

<div class="col-md-2"> <label>Email</label></div>

<div class="col-md-9">

<div class="input-space">

<input formControlName="email" type="text">

<div \*ngIf="email?.invalid && (email?.dirty || email?.touched)" class="alert alert-danger mt-1">

<div \*ngIf="email?.errors?.['required']">

Email is required

</div>

<div \*ngIf="email?.errors?.['pattern']">

Email must be a valid email address format

</div>

</div>

</div>

</div>

</div>

</div>

<!-- shipping Address -->

<div formGroupName="shippingAddress" class="form-area">

<h3>Shipping Address</h3>

<div class="row">

<div class="col-md-2"> <label>Country</label></div>

<div class="col-md-9">

<div class="input-space">

<select formControlName="country" (change)="getStates('shippingAddress')">

<option \*ngFor="let country of countries" [ngValue]="country">

{{ country.name }}

</option>

</select>

<div \*ngIf="shippingAddressCountry?.invalid && (shippingAddressCountry?.dirty || shippingAddressCountry?.touched)" class="alert alert-danger mt-1">

<div \*ngIf="shippingAddressCountry?.errors?.['required']">

Country is required

</div>

</div>

</div>

</div>

</div>

<div class="row">

<div class="col-md-2"> <label>Street</label></div>

<div class="col-md-9">

<div class="input-space">

<input formControlName="street" type="text">

<div \*ngIf="shippingAddressStreet?.invalid && (shippingAddressStreet?.dirty || shippingAddressStreet?.touched)" class="alert alert-danger mt-1">

<div \*ngIf="shippingAddressStreet?.errors?.['required'] || shippingAddressStreet?.errors?.['notOnlyWhitespace']">

Street is required

</div>

<div \*ngIf="shippingAddressStreet?.errors?.['minlength']">

Street must be at least 2 characters long

</div>

</div>

</div>

</div>

</div>

<div class="row">

<div class="col-md-2"> <label>City</label></div>

<div class="col-md-9">

<div class="input-space">

<input formControlName="city" type="text">

<div \*ngIf="shippingAddressCity?.invalid && (shippingAddressCity?.dirty || shippingAddressCity?.touched)" class="alert alert-danger mt-1">

<div \*ngIf="shippingAddressCity?.errors?.['required'] || shippingAddressCity?.errors?.['notOnlyWhitespace']">

City is required

</div>

<div \*ngIf="shippingAddressCity?.errors?.['minlength']">

City must be at least 2 characters long

</div>

</div>

</div>

</div>

</div>

<div class="row">

<div class="col-md-2"> <label>State</label></div>

<div class="col-md-9">

<div class="input-space">

<select formControlName="state">

<option \*ngFor="let state of shippingAddressStates" [ngValue]="state">

{{ state.name }}

</option>

</select>

<div \*ngIf="shippingAddressState?.invalid && (shippingAddressState?.dirty || shippingAddressState?.touched)" class="alert alert-danger mt-1">

<div \*ngIf="shippingAddressState?.errors?.['required']">

State is required

</div>

</div>

</div>

</div>

</div>

<div class="row">

<div class="col-md-2"> <label>Zip Code</label></div>

<div class="col-md-9">

<div class="input-space">

<input formControlName="zipCode" type="text">

<div \*ngIf="shippingAddressZipCode?.invalid && (shippingAddressZipCode?.dirty || shippingAddressZipCode?.touched)" class="alert alert-danger mt-1">

<div \*ngIf="shippingAddressZipCode?.errors?.['required'] || shippingAddressZipCode?.errors?.['notOnlyWhitespace']">

Zip code is required

</div>

<div \*ngIf="shippingAddressZipCode?.errors?.['minlength']">

Zip code must be at least 2 characters long

</div>

</div>

</div>

</div>

</div>

</div>

<div class="input-space">

<label class="au-checkbox">

<input type="checkbox" (change)="copyShippingAddressToBillingAddress($any($event).target.value)">

<span class="au-checkmark"></span>Billing Address same as Shipping Address

</label>

</div>

<div formGroupName="billingAddress" class="form-area">

<h3>Billing Address</h3>

<div class="row">

<div class="col-md-2"> <label>Country</label></div>

<div class="col-md-9">

<div class="input-space">

<select formControlName="country" (change)="getStates('billingAddress')">

<option \*ngFor="let country of countries" [ngValue]="country">

{{ country.name }}

</option>

</select>

<div \*ngIf="billingAddressCountry?.invalid && (billingAddressCountry?.dirty || billingAddressCountry?.touched)" class="alert alert-danger mt-1">

<div \*ngIf="billingAddressCountry?.errors?.['required']">

Country is required

</div>

</div>

</div>

</div>

</div>

<div class="row">

<div class="col-md-2"> <label>Street</label></div>

<div class="col-md-9">

<div class="input-space">

<input formControlName="street" type="text">

<div \*ngIf="billingAddressStreet?.invalid && (billingAddressStreet?.dirty || billingAddressStreet?.touched)" class="alert alert-danger mt-1">

<div \*ngIf="billingAddressStreet?.errors?.['required'] || billingAddressStreet?.errors?.['notOnlyWhitespace']">

Street is required

</div>

<div \*ngIf="billingAddressStreet?.errors?.['minlength']">

Street must be at least 2 characters long

</div>

</div>

</div>

</div>

</div>

<div class="row">

<div class="col-md-2"> <label>City</label></div>

<div class="col-md-9">

<div class="input-space">

<input formControlName="city" type="text">

<div \*ngIf="billingAddressCity?.invalid && (billingAddressCity?.dirty || billingAddressCity?.touched)" class="alert alert-danger mt-1">

<div \*ngIf="billingAddressCity?.errors?.['required'] || billingAddressCity?.errors?.['notOnlyWhitespace']">

City is required

</div>

<div \*ngIf="billingAddressCity?.errors?.['minlength']">

City must be at least 2 characters long

</div>

</div>

</div>

</div>

</div>

<div class="row">

<div class="col-md-2"> <label>State</label></div>

<div class="col-md-9">

<div class="input-space">

<select formControlName="state">

<option \*ngFor="let state of billingAddressStates" [ngValue]="state">

{{ state.name }}

</option>

</select>

<div \*ngIf="billingAddressState?.invalid && (billingAddressState?.dirty || billingAddressState?.touched)" class="alert alert-danger mt-1">

<div \*ngIf="billingAddressState?.errors?.['required']">

State is required

</div>

</div>

</div>

</div>

</div>

<div class="row">

<div class="col-md-2"> <label>Zip Code</label></div>

<div class="col-md-9">

<div class="input-space">

<input formControlName="zipCode" type="text">

<div \*ngIf="billingAddressZipCode?.invalid && (billingAddressZipCode?.dirty || billingAddressZipCode?.touched)" class="alert alert-danger mt-1">

<div \*ngIf="billingAddressZipCode?.errors?.['required'] || billingAddressZipCode?.errors?.['notOnlyWhitespace']">

Zip code is required

</div>

<div \*ngIf="billingAddressZipCode?.errors?.['minlength']">

Zip code must be at least 2 characters long

</div>

</div>

</div>

</div>

</div>

</div>

<!-- Credit Card -->

<!-- Stripe Elements Credit Card Section -->

<div formGroupName="creditCard" class="form-area">

<h3>Credit or Debit Card</h3>

<div id="card-element">

<!-- a Stripe element will be inserted here -->

</div>

<!-- Used to display form errors -->

<div id="card-errors" class="displayError.textContent!=='': 'alert alert-danger mt-1'"></div>

</div>

<!-- Order details -->

<div class="form-area">

<h3>Review Your Order</h3>

<p>Total Quantity: {{ totalQuantity }}</p>

<p>Shipping: FREE</p>

<p>Total Price: {{ totalPrice | currency: 'USD' }}</p>

</div>

<!-- submit button-->

<div class="text-center">

<button type="submit" class="btn btn-info" [disabled]="isDisabled">Purchase</button>

</div>

</form>

</div>

</div>

</div>

//cart.details.component.html

<div class="main-content">

<div class="section-content section-content-p30">

<div class="container-fluid">

<div \*ngIf="cartItems.length > 0">

<table class="table table-bordered">

<tr>

<th width="20%">Product Image</th>

<th width="50%">Product Detail</th>

<th width="30%"></th>

</tr>

<tr \*ngFor="let tempCartItem of cartItems">

<td>

<img src="{{ tempCartItem.imageUrl }}" class="img-responsive" width="150px" />

</td>

<td>

<p>{{ tempCartItem.name }}</p>

<p>{{ tempCartItem.unitPrice | currency: 'USD' }}</p>

</td>

<td>

<div class="items">

<label>Quantity:</label>

<div class="row no-gutters">

<div class="col">

<button (click)="incrementQuantity(tempCartItem)" class="btn btn-primary btn-sm">

<i class="fas fa-plus"></i>

</button>

</div>

<div class="col ml-4 mr-2">

{{ tempCartItem.quantity }}

</div>

<div class="col">

<button (click)="decrementQuantity(tempCartItem)" class="btn btn-primary btn-sm">

<i class="fas fa-minus"></i>

</button>

</div>

<div class="col-8"></div>

</div>

</div>

<div>

<button (click)="remove(tempCartItem)" class="btn btn-primary btn-sm mt-2">Remove</button>

</div>

<p class="mt-2">Subtotal: {{ tempCartItem.quantity! \* tempCartItem?.unitPrice! | currency: 'USD' }}</p>

</td>

</tr>

<tr>

<td colspan="2"></td>

<td style="font-weight: bold">

<p>Total Quantity: {{ totalQuantity }}</p>

<p>Shipping: FREE</p>

<p>Total Price: {{ totalPrice | currency: 'USD' }}</p>

<div>

<a routerLink="/checkout" class="btn btn-primary">Checkout</a>

</div>

</td>

</tr>

</table>

</div>

<div \*ngIf="cartItems.length == 0" class="alert alert-warning col-md-12" role="alert">

Your shopping cart is empty.

</div>

</div>

</div>

</div>

//app.module.ts

import { BrowserModule } from '@angular/platform-browser';

import { Injector, NgModule } from '@angular/core';

import { HttpClientModule, HTTP\_INTERCEPTORS } from '@angular/common/http';

import { AppComponent } from './app.component';

import { ProductListComponent } from './components/product-list/product-list.component';

import { ProductService } from './services/product.service';

import { Routes, RouterModule, Router} from '@angular/router';

import { ProductCategoryMenuComponent } from './components/product-category-menu/product-category-menu.component';

import { SearchComponent } from './components/search/search.component';

import { ProductDetailsComponent } from './components/product-details/product-details.component';

import { NgbModule } from '@ng-bootstrap/ng-bootstrap';

import { CartStatusComponent } from './components/cart-status/cart-status.component';

import { CartDetailsComponent } from './components/cart-details/cart-details.component';

import { CheckoutComponent } from './components/checkout/checkout.component';

import { ReactiveFormsModule } from '@angular/forms';

import { BrowserAnimationsModule } from '@angular/platform-browser/animations';

import { LoginComponent } from './components/login/login.component';

import { LoginStatusComponent } from './components/login-status/login-status.component';

import { AuthInterceptorService } from './services/auth-interceptor.service';

import {

OktaAuthGuard,

OktaAuthModule,

OktaCallbackComponent,

OKTA\_CONFIG

} from '@okta/okta-angular';

import { OktaAuth } from '@okta/okta-auth-js';

import myAppConfig from './config/my-app-config';

import { MembersPageComponent } from './components/members-page/members-page.component';

import { OrderHistory } from './common/order-history';

import { OrderHistoryComponent } from './components/order-history/order-history.component';

const oktaConfig = myAppConfig.oidc;

const oktaAuth = new OktaAuth(oktaConfig);

function sendToLoginPage(oktaAuth: OktaAuth, injector: Injector) {

const router = injector.get(Router);

router.navigate(['/login']);

}

const routes: Routes = [

{path: 'order-history', component: OrderHistoryComponent, canActivate: [OktaAuthGuard],

data: {onAuthRequired: sendToLoginPage} },

{path: 'members', component: MembersPageComponent, canActivate: [OktaAuthGuard],

data: {onAuthRequired: sendToLoginPage} },

{path: 'login/callback', component: OktaCallbackComponent},

{path: 'login', component: LoginComponent},

{path: 'checkout', component: CheckoutComponent},

{path: 'cart-details', component: CartDetailsComponent},

{path: 'products/:id', component: ProductDetailsComponent},

{path: 'search/:keyword', component: ProductListComponent},

{path: 'category/:id', component: ProductListComponent},

{path: 'category', component: ProductListComponent},

{path: 'products', component: ProductListComponent},

{path: '', redirectTo: '/products', pathMatch: 'full'},

{path: '\*\*', redirectTo: '/products', pathMatch: 'full'}

];

@NgModule({

declarations: [

AppComponent,

ProductListComponent,

ProductCategoryMenuComponent,

SearchComponent,

ProductDetailsComponent,

CartStatusComponent,

CartDetailsComponent,

CheckoutComponent,

LoginComponent,

LoginStatusComponent,

MembersPageComponent,

OrderHistoryComponent,

],

imports: [

RouterModule,

RouterModule.forRoot(routes),

BrowserModule,

HttpClientModule,

NgbModule,

ReactiveFormsModule,

BrowserAnimationsModule,

OktaAuthModule

],

providers: [ProductService, { provide: OKTA\_CONFIG, useValue: { oktaAuth }},

{provide: HTTP\_INTERCEPTORS, useClass: AuthInterceptorService, multi: true}],

bootstrap: [AppComponent]

})

export class AppModule { }

//auth-interceptor.service.ts

import { HttpEvent, HttpHandler, HttpInterceptor, HttpRequest } from '@angular/common/http';

import { Inject, Injectable } from '@angular/core';

import { OKTA\_AUTH } from '@okta/okta-angular';

import { OktaAuth } from '@okta/okta-auth-js';

import { from, lastValueFrom, Observable } from 'rxjs';

import { environment } from 'src/environments/environment';

@Injectable({

providedIn: 'root'

})

export class AuthInterceptorService implements HttpInterceptor {

constructor(@Inject(OKTA\_AUTH) private oktaAuth: OktaAuth) { }

intercept(request: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {

return from(this.handleAccess(request, next));

}

private async handleAccess(request: HttpRequest<any>, next: HttpHandler): Promise<HttpEvent<any>> {

const theEndpoint = environment.luv2shopApiUrl + '/orders';

const securedEndpoints = [theEndpoint];

if (securedEndpoints.some(url => request.urlWithParams.includes(url))) {

const accessToken = this.oktaAuth.getAccessToken();

request = request.clone({

setHeaders: {

Authorization: 'Bearer ' + accessToken

}

});

}

return await lastValueFrom(next.handle(request));

}

}

//cart.details.component.html

<div class="main-content">

<div class="section-content section-content-p30">

<div class="container-fluid">

<div \*ngIf="cartItems.length > 0">

<table class="table table-bordered">

<tr>

<th width="20%">Product Image</th>

<th width="50%">Product Detail</th>

<th width="30%"></th>

</tr>

<tr \*ngFor="let tempCartItem of cartItems">

<td>

<img src="{{ tempCartItem.imageUrl }}" class="img-responsive" width="150px" />

</td>

<td>

<p>{{ tempCartItem.name }}</p>

<p>{{ tempCartItem.unitPrice | currency: 'USD' }}</p>

</td>

<td>

<div class="items">

<label>Quantity:</label>

<div class="row no-gutters">

<div class="col">

<button (click)="incrementQuantity(tempCartItem)" class="btn btn-primary btn-sm">

<i class="fas fa-plus"></i>

</button>

</div>

<div class="col ml-4 mr-2">

{{ tempCartItem.quantity }}

</div>

<div class="col">

<button (click)="decrementQuantity(tempCartItem)" class="btn btn-primary btn-sm">

<i class="fas fa-minus"></i>

</button>

</div>

<div class="col-8"></div>

</div>

</div>

<div>

<button (click)="remove(tempCartItem)" class="btn btn-primary btn-sm mt-2">Remove</button>

</div>

<p class="mt-2">Subtotal: {{ tempCartItem.quantity! \* tempCartItem?.unitPrice! | currency: 'USD' }}</p>

</td>

</tr>

<tr>

<td colspan="2"></td>

<td style="font-weight: bold">

<p>Total Quantity: {{ totalQuantity }}</p>

<p>Shipping: FREE</p>

<p>Total Price: {{ totalPrice | currency: 'USD' }}</p>

<div>

<a routerLink="/checkout" class="btn btn-primary">Checkout</a>

</div>

</td>

</tr>

</table>

</div>

<div \*ngIf="cartItems.length == 0" class="alert alert-warning col-md-12" role="alert">

Your shopping cart is empty.

</div>

</div>

</div>

</div>

// login.component.html

import { Component, Inject, OnInit } from '@angular/core';

import { OKTA\_AUTH } from '@okta/okta-angular';

import { OktaAuth } from '@okta/okta-auth-js';

import OktaSignIn from '@okta/okta-signin-widget';

import myAppConfig from '../../config/my-app-config';

@Component({

selector: 'app-login',

templateUrl: './login.component.html',

styleUrls: ['./login.component.css']

})

export class LoginComponent implements OnInit {

oktaSignin: any;

constructor(@Inject(OKTA\_AUTH) private oktaAuth: OktaAuth) {

this.oktaSignin = new OktaSignIn({

logo: 'assets/images/logo.png',

baseUrl: myAppConfig.oidc.issuer.split('/oauth2')[0],

clientId: myAppConfig.oidc.clientId,

redirectUri: myAppConfig.oidc.redirectUri,

authParams: {

pkce: true,

issuer: myAppConfig.oidc.issuer,

scopes: myAppConfig.oidc.scopes

}

});

}

ngOnInit(): void {

this.oktaSignin.remove();

this.oktaSignin.renderEl({

el: '#okta-sign-in-widget'},

(response: any) => {

if (response.status === 'SUCCESS') {

this.oktaAuth.signInWithRedirect();

}

},

(error: any) => {

throw error;

}

);

}

}

//styles.css

/\*

\* ----------------------------

\* Common CSS

\* ----------------------------

\*/

\*{

margin:0;

padding:0;

-webkit-box-sizing:border-box;

-moz-box-sizing:border-box;

box-sizing:border-box

}

body{

font-family:Poppins,sans-serif;

font-weight:400;

font-size:16px;

line-height:1.625;

color:#666;

-webkit-font-smoothing:antialiased;

-moz-osx-font-smoothing:grayscale;

}

ul{

margin:0

}

button,input[type=button]{

cursor:pointer

}

button:focus,input:focus{

outline:0

}

input{

border:none

}

button{

border:none;

background:0 0

}

img{

max-width:100%;

height:auto

}

p{

margin:0

}

h1,h2,h3,h4,h5,h6{

color:#333;

font-weight:700;

margin:0;

line-height:1.2

}

h1{

font-size:36px

}

h2{

font-size:30px

}

h3{

font-size:24px

}

h4{

font-size:18px

}

h5{

font-size:15px

}

h6{

font-size:13px

}

a{

display:inline-block

}

a:active,a:focus,a:hover{

text-decoration:none;

outline:0

}

a,a:hover{

-webkit-transition:all .3s ease;

-o-transition:all .3s ease;

-moz-transition:all .3s ease;

transition:all .3s ease

}

/\*Layout Css\*/

.section-content-p35{

padding:0 35px

}

.page-wrapper{

overflow:hidden;

background:#e5e5e5;

padding-bottom:8vh

}

.page-container{

background:#e5e5e5;

padding-left:200px

}

.main-content{

padding-top:116px;

min-height:100vh

}

aside.menu-sidebar.d-none.d-lg-block.clr{

background:0 0;

z-index:99999999;

height:75px;

box-shadow:-2px 0 4px #0000003b

}

/\*

\* ----------------------------

\* Header Desktop Section

\* ----------------------------

\*/

.header-desktop{

background:#0da8e4;

-webkit-box-shadow:0 2px 5px 0 rgba(0,0,0,.1);

-moz-box-shadow:0 2px 5px 0 rgba(0,0,0,.1);

box-shadow:0 2px 5px 0 rgba(0,0,0,.1);

position:fixed;

top:0;

right:0;

left:200px;

height:75px;

z-index:3

}

.header-desktop .section-content{

overflow:visible;

position:absolute;

top:50%;

left:0;

right:0;

-webkit-transform:translateY(-50%);

-moz-transform:translateY(-50%);

-ms-transform:translateY(-50%);

-o-transform:translateY(-50%);

transform:translateY(-50%)

}

/\*

\* ----------------------------

\* Footer section

\* ----------------------------

\*/

.header-mobile .header-mobile-bar{

padding:10px 0;

background:#fff

}

.header-wrap .account-item>.content{

display:none

}

}

@media (max-width:991px){

.section-content-p30{

padding:0

}

.page-container{

position:relative;

top:88px;

padding-left:0

}

.page-wrapper{

background:#e5e5e5;

padding-bottom:12vh

}

.main-content{

padding-top:50px;

padding-bottom:100px

}

.cart-area.d-n{

display:none

}

.form-header{

-webkit-box-pack:center;

-webkit-justify-content:center;

-moz-box-pack:center;

-ms-flex-pack:center;

justify-content:center

}

.au-input-xl{

min-width:350px

}

.logo{

text-align:center

}

.header-desktop{

position:relative;

top:0;

left:0;

height:80px

}

.header-wrap{

-webkit-box-orient:vertical;

-webkit-box-direction:normal;

-webkit-flex-direction:column;

-moz-box-orient:vertical;

-moz-box-direction:normal;

-ms-flex-direction:column;

flex-direction:column

}

}

//luv2-shop-form.service.ts

import { Injectable } from '@angular/core';

import { Observable, of } from 'rxjs';

import { HttpClient } from '@angular/common/http';

import { Country } from '../common/country';

import { map } from 'rxjs/operators';

import { State } from '../common/state';

import { environment } from 'src/environments/environment';

@Injectable({

providedIn: 'root'

})

export class Luv2ShopFormService {

private countriesUrl = environment.luv2shopApiUrl + '/countries';

private statesUrl = environment.luv2shopApiUrl + '/states';

constructor(private httpClient: HttpClient) { }

getCountries(): Observable<Country[]> {

return this.httpClient.get<GetResponseCountries>(this.countriesUrl).pipe(

map(response => response.\_embedded.countries)

);

}

getStates(theCountryCode: string): Observable<State[]> {

const searchStatesUrl = `${this.statesUrl}/search/findByCountryCode?code=${theCountryCode}`;

return this.httpClient.get<GetResponseStates>(searchStatesUrl).pipe(

map(response => response.\_embedded.states)

);

}

getCreditCardMonths(startMonth: number): Observable<number[]> {

let data: number[] = [];

for (let theMonth = startMonth; theMonth <= 12; theMonth++) {

data.push(theMonth);

}

return of(data);

}

getCreditCardYears(): Observable<number[]> {

let data: number[] = [];

const startYear: number = new Date().getFullYear();

const endYear: number = startYear + 10;

for (let theYear = startYear; theYear <= endYear; theYear++) {

data.push(theYear);

}

return of(data);

}

}

interface GetResponseCountries {

\_embedded: {

countries: Country[];

}

}

interface GetResponseStates {

\_embedded: {

states: State[];

}

}

//OrderItem.java

package com.luv2code.ecommerce.entity;

import lombok.Getter;

import lombok.Setter;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import javax.persistence.Table;

import java.math.BigDecimal;

@Entity

@Table(name="order\_item")

@Getter

@Setter

public class OrderItem {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name="id")

private Long id;

@Column(name="image\_url")

private String imageUrl;

@Column(name="unit\_price")

private BigDecimal unitPrice;

@Column(name="quantity")

private int quantity;

@Column(name="product\_id")

private Long productId;

@ManyToOne

@JoinColumn(name = "order\_id")

private Order order;

}

//db.changelog-1.0.sql

--liquibase formatted sql

--changeset jjasinovski:1

CREATE TABLE IF NOT EXISTS `fullStackEcommerce`.`product\_category` (

`id` BIGINT(20) NOT NULL AUTO\_INCREMENT,

`category\_name` VARCHAR(255) NULL DEFAULT NULL,

PRIMARY KEY (`id`))

ENGINE=InnoDB

AUTO\_INCREMENT = 1;

--rollback DROP TABLE product\_category;

--changeset jjasinovski:2

CREATE TABLE IF NOT EXISTS `fullStackEcommerce`.`product` (

`id` BIGINT(20) NOT NULL AUTO\_INCREMENT,

`sku` VARCHAR(255) DEFAULT NULL,

`name` VARCHAR(255) DEFAULT NULL,

`description` VARCHAR(255) DEFAULT NULL,

`unit\_price` DECIMAL(13,2) DEFAULT NULL,

`image\_url` VARCHAR(255) DEFAULT NULL,

`active` BIT DEFAULT 1,

`units\_in\_stock` INT(11) DEFAULT NULL,

`date\_created` DATETIME(6) DEFAULT NULL,

`last\_updated` DATETIME(6) DEFAULT NULL,

`category\_id` BIGINT(20) NOT NULL,

PRIMARY KEY (`id`),

CONSTRAINT `fk\_category` FOREIGN KEY (`category\_id`) REFERENCES `product\_category` (`id`)

)

ENGINE=InnoDB

AUTO\_INCREMENT = 1;

--changeset jjasinovski:3

CREATE TABLE `country` (

`id` smallint unsigned NOT NULL,

`code` varchar(2) DEFAULT NULL,

`name` varchar(255) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=1;

--changeset jjasinovski:4

CREATE TABLE `state` (

`id` smallint unsigned NOT NULL AUTO\_INCREMENT,

`name` varchar(255) DEFAULT NULL,

`country\_id` smallint unsigned NOT NULL,

PRIMARY KEY (`id`),

CONSTRAINT `fk\_country` FOREIGN KEY (`country\_id`) REFERENCES `country` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=1;

--changeset jjasinovski:5

CREATE TABLE `address` (

`id` bigint NOT NULL AUTO\_INCREMENT,

`city` varchar(255) DEFAULT NULL,

`country` varchar(255) DEFAULT NULL,

`state` varchar(255) DEFAULT NULL,

`street` varchar(255) DEFAULT NULL,

`zip\_code` varchar(255) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=1;

--changeset jjasinovski:7

CREATE TABLE `customer` (

`id` bigint NOT NULL AUTO\_INCREMENT,

`first\_name` varchar(255) DEFAULT NULL,

`last\_name` varchar(255) DEFAULT NULL,

`email` varchar(255) DEFAULT NULL unique,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=1;

--changeset jjasinovski:8

CREATE TABLE `orders` (

`id` bigint NOT NULL AUTO\_INCREMENT,

`order\_tracking\_number` varchar(255) DEFAULT NULL,

`total\_price` decimal(19,2) DEFAULT NULL,

`total\_quantity` int DEFAULT NULL,

`billing\_address\_id` bigint DEFAULT NULL,

`customer\_id` bigint DEFAULT NULL,

`shipping\_address\_id` bigint DEFAULT NULL,

`status` varchar(128) DEFAULT NULL,

`date\_created` datetime(6) DEFAULT NULL,

`last\_updated` datetime(6) DEFAULT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `UK\_billing\_address\_id` (`billing\_address\_id`),

UNIQUE KEY `UK\_shipping\_address\_id` (`shipping\_address\_id`),

CONSTRAINT `FK\_customer\_id` FOREIGN KEY (`customer\_id`) REFERENCES `customer` (`id`),

CONSTRAINT `FK\_billing\_address\_id` FOREIGN KEY (`billing\_address\_id`) REFERENCES `address` (`id`),

CONSTRAINT `FK\_shipping\_address\_id` FOREIGN KEY (`shipping\_address\_id`) REFERENCES `address` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=1;

--changeset jjasinovski:9

CREATE TABLE `order\_item` (

`id` bigint NOT NULL AUTO\_INCREMENT,

`image\_url` varchar(255) DEFAULT NULL,

`quantity` int DEFAULT NULL,

`unit\_price` decimal(19,2) DEFAULT NULL,

`order\_id` bigint DEFAULT NULL,

`product\_id` bigint DEFAULT NULL,

PRIMARY KEY (`id`),

CONSTRAINT `FK\_order\_id` FOREIGN KEY (`order\_id`) REFERENCES `orders` (`id`),

CONSTRAINT `FK\_product\_id` FOREIGN KEY (`product\_id`) REFERENCES `product` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=1;

//db.changelog-2.0.sql

--liquibase formatted sql

--changeset jjasinovski:1

INSERT INTO product\_category(category\_name) VALUES ('Audi');

INSERT INTO product\_category(category\_name) VALUES ('BMW');

INSERT INTO product\_category(category\_name) VALUES ('Ford');

INSERT INTO product\_category(category\_name) VALUES ('Honda');

INSERT INTO product\_category(category\_name) VALUES ('Hyundai');

INSERT INTO product\_category(category\_name) VALUES ('Kia');

INSERT INTO product\_category(category\_name) VALUES ('Lada');

INSERT INTO product\_category(category\_name) VALUES ('Mazda');

INSERT INTO product\_category(category\_name) VALUES ('Mercedes-Benz');

INSERT INTO product\_category(category\_name) VALUES ('Mitsubishi');

INSERT INTO product\_category(category\_name) VALUES ('Nissan');

INSERT INTO product\_category(category\_name) VALUES ('Renault');

INSERT INTO product\_category(category\_name) VALUES ('Skoda');

INSERT INTO product\_category(category\_name) VALUES ('Toyota');

INSERT INTO product\_category(category\_name) VALUES ('Volkswagen');

INSERT INTO product\_category(category\_name) VALUES ('Porsche');

INSERT INTO product\_category(category\_name) VALUES ('Tesla');

INSERT INTO product\_category(category\_name) VALUES ('Maybach');

INSERT INTO product\_category(category\_name) VALUES ('Infiniti');

--changeset jjasinovski:2

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('AUDI-TECH-1000', 'Audi A8 (D5)', 'Start of release 2017; Graduation edition of 2021 | Sedan. Power: from 286 hp to 460 hp. Length 5172 mm (203.62 in.); Width 1945 mm (76.57 in.); Height 1473 mm (57.99 in.); Wheelbase 2998 mm (118.03 in.);', 'assets/images/products/audi/1000.png', 1, 100, 123.045, 1, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('AUDI-TECH-1001', 'Audi A4 B(9)', '', 'assets/images/products/audi/1001.png', 1, 100, 20.99, 1, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('AUDI-TECH-1002', 'Audi A6 C8', '', 'assets/images/products/audi/1002.png', 1, 100, 14.99, 1, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('AUDI-TECH-1003', 'Audi S5 II', '', 'assets/images/products/audi/1003.png', 1, 100, 13.99, 1, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('AUDI-TECH-1004', 'Audi A5 II', '', 'assets/images/products/audi/1004.png', 1, 100, 18.99, 1, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('AUDI-TECH-1005', 'Audi A3 IV', '', 'assets/images/products/audi/1005.png', 1, 100, 23.99, 1, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('AUDI-TECH-1006', 'Audi RS 6 (C8) Avant', '', 'assets/images/products/audi/1006.png', 1, 100, 14.99, 1, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('AUDI-TECH-1007', 'Audi A4 B(9)', '', 'assets/images/products/audi/1007.png', 1, 100, 16.99, 1, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('AUDI-TECH-1008', 'Audi А5 Sportback I', '', 'assets/images/products/audi/1008.png', 1, 100, 16.99, 1, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('AUDI-TECH-1009', 'Audi A7 Sportback II', '', 'assets/images/products/audi/1009.png', 1, 100, 16.99, 1, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('AUDI-TECH-1010', 'Audi S5 II', '', 'assets/images/products/audi/1010.png', 1, 100, 16.99, 1, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('AUDI-TECH-1011', 'Audi А3 Sportback IV', '', 'assets/images/products/audi/1011.png', 1, 100, 16.99, 1, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('AUDI-TECH-1012', 'Audi A6 allroad quattro С8 IV', '', 'assets/images/products/audi/1012.png', 1, 100, 16.99, 1, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('AUDI-TECH-1013', 'Audi E-tron I', '', 'assets/images/products/audi/1013.png', 1, 100, 16.99, 1, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('AUDI-TECH-1014', 'Audi Q8 I', '', 'assets/images/products/audi/1014.png', 1, 100, 16.99, 1, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('AUDI-TECH-1015', 'Audi SQ8 I', '', 'assets/images/products/audi/1015.png', 1, 100, 16.99, 1, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('AUDI-TECH-1016', 'Audi Q3 II', '', 'assets/images/products/audi/1016.png', 1, 100, 27.99, 1, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('AUDI-TECH-1017', 'Audi Q3 Sportback II', '', 'assets/images/products/audi/1017.png', 1, 100, 27.99, 1, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('AUDI-TECH-1018', 'Audi Q5 II', '', 'assets/images/products/audi/1018.png', 1, 100, 27.99, 1, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('AUDI-TECH-1019', 'Audi SQ5 II', '', 'assets/images/products/audi/1019.png', 1, 100, 27.99, 1, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('AUDI-TECH-1020', 'Audi Q7 II', '', 'assets/images/products/audi/1020.png', 1, 100, 27.99, 1, NOW());

--changeset jjasinovski:3

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('BMW-1000', 'BMW M2 F87', '', 'assets/images/products/bmw/1000.png', 1, 100, 18.99, 2, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('BMW-1001', 'BMW 7 G11/G12', '', 'assets/images/products/bmw/1001.png', 1, 100, 18.99, 2, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('BMW-1002', 'BMW 8 G14/G15', '', 'assets/images/products/bmw/1002.png', 1, 100, 18.99, 2, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('BMW-1003', 'BMW 8 Gran Coupe G15', '', 'assets/images/products/bmw/1003.png', 1, 100, 18.99, 2, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('BMW-1004', 'BMW 3 G20', '', 'assets/images/products/bmw/1004.png', 1, 100, 18.99, 2, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('BMW-1005', 'BMW 5 G30', '', 'assets/images/products/bmw/1005.png', 1, 100, 18.99, 2, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('BMW-1006', 'BMW M8 I (F91/F92/F93)', '', 'assets/images/products/bmw/1006.png', 1, 100, 18.99, 2, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('BMW-1007', 'BMW 4 II', '', 'assets/images/products/bmw/1007.png', 1, 100, 18.99, 2, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('BMW-1008', 'BMW M5 VI', '', 'assets/images/products/bmw/1008.png', 1, 100, 18.99, 2, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('BMW-1009', 'BMW 1 F40', '', 'assets/images/products/bmw/1009.png', 1, 100, 18.99, 2, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('BMW-1010', 'BMW 6 Gran Turismo G32', '', 'assets/images/products/bmw/1010.png', 1, 100, 18.99, 2, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('BMW-1011', 'BMW X2 F39', '', 'assets/images/products/bmw/1011.png', 1, 100, 18.99, 2, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('BMW-1012', 'BMW X1 F48', '', 'assets/images/products/bmw/1012.png', 1, 100, 18.99, 2, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('BMW-1013', 'BMW X4 G02', '', 'assets/images/products/bmw/1013.png', 1, 100, 18.99, 2, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('BMW-1014', 'BMW X5 G05', '', 'assets/images/products/bmw/1014.png', 1, 100, 18.99, 2, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('BMW-1015', 'BMW X6 G06', '', 'assets/images/products/bmw/1015.png', 1, 100, 18.99, 2, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('BMW-1016', 'BMW X7 I', '', 'assets/images/products/bmw/1016.png', 1, 100, 18.99, 2, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('BMW-1017', 'BMW 8 G14/G15', '', 'assets/images/products/bmw/1017.png', 1, 100, 18.99, 2, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('BMW-1018', 'BMW M8 I (F91/F92/F93)', '', 'assets/images/products/bmw/1018.png', 1, 100, 18.99, 2, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('BMW-1019', 'BMW 4 II', '', 'assets/images/products/bmw/1019.png', 1, 100, 18.99, 2, NOW());

--changeset jjasinovski:4

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('FORD-1000', 'Ford Transit Custom I', '', 'assets/images/products/ford/1000.png', 1, 100, 17.99, 3, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('FORD-1001', 'Ford Tourneo Custom II', '', 'assets/images/products/ford/1001.png', 1, 100, 17.99, 3, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('FORD-1002', 'Ford Transit VIII (Mark 8)', '', 'assets/images/products/ford/1002.png', 1, 100, 17.99, 3, NOW());

--changeset jjasinovski:5

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('HONDA-1000', 'Honda Pilot III', '', 'assets/images/products/honda/1000.png', 1, 100, 16.99, 4, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('HONDA-1001', 'Honda CR-V V', '', 'assets/images/products/honda/1001.png', 1, 100, 16.99, 4, NOW());

--changeset jjasinovski:6

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('HYUNDAI-1000', 'Hyundai Solaris II', '', 'assets/images/products/hyundai/1000.png', 1, 100, 16.99, 5, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('HYUNDAI-1001', 'Hyundai Elantra VI', '', 'assets/images/products/hyundai/1001.png', 1, 100, 16.99, 5, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('HYUNDAI-1002', 'Hyundai Elantra VII', '', 'assets/images/products/hyundai/1002.png', 1, 100, 16.99, 5, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('HYUNDAI-1003', 'Hyundai Sonata VIII', '', 'assets/images/products/hyundai/1003.png', 1, 100, 16.99, 5, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('HYUNDAI-1004', 'Hyundai i30 III', '', 'assets/images/products/hyundai/1004.png', 1, 100, 16.99, 5, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('HYUNDAI-1005', 'Hyundai Palisade I', '', 'assets/images/products/hyundai/1005.png', 1, 100, 16.99, 5, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('HYUNDAI-1006', 'Hyundai Creta I', '', 'assets/images/products/hyundai/1006.png', 1, 100, 16.99, 5, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('HYUNDAI-1007', 'Hyundai Tucson III', '', 'assets/images/products/hyundai/1007.png', 1, 100, 16.99, 5, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('HYUNDAI-1008', 'Hyundai Tucson IV', '', 'assets/images/products/hyundai/1008.png', 1, 100, 16.99, 5, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('HYUNDAI-1009', 'Hyundai Santa Fe Premium IV', '', 'assets/images/products/hyundai/1009.png', 1, 100, 16.99, 5, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('HYUNDAI-1010', 'Hyundai H1 II', '', 'assets/images/products/hyundai/1010.png', 1, 100, 16.99, 5, NOW());

--changeset jjasinovski:7

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('KIA-1000', 'Kia K900 I', '', 'assets/images/products/kia/1000.png', 1, 100, 16.99, 6, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('KIA-1001', 'Kia Cerato IV', '', 'assets/images/products/kia/1001.png', 1, 100, 16.99, 6, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('KIA-1002', 'Kia Rio IV', '', 'assets/images/products/kia/1002.png', 1, 100, 16.99, 6, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('KIA-1003', 'Kia К5 V', '', 'assets/images/products/kia/1003.png', 1, 100, 16.99, 6, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('KIA-1004', 'Kia Stinger I', '', 'assets/images/products/kia/1004.png', 1, 100, 16.99, 6, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('KIA-1005', 'Kia Rio I поколение X (X-Line)', '', 'assets/images/products/kia/1005.png', 1, 100, 16.99, 6, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('KIA-1006', 'Kia Rio I поколение рестайлинг X', '', 'assets/images/products/kia/1006.png', 1, 100, 16.99, 6, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('KIA-1007', 'Kia Stinger I', '', 'assets/images/products/kia/1007.png', 1, 100, 16.99, 6, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('KIA-1008', 'Kia cee''d III хэтчбек', '', 'assets/images/products/kia/1008.png', 1, 100, 16.99, 6, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('KIA-1009', 'Kia cee''d III универсал', '', 'assets/images/products/kia/1009.png', 1, 100, 16.99, 6, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('KIA-1010', 'Kia cee''d III pro\_ceed универсал', '', 'assets/images/products/kia/1010.png', 1, 100, 16.99, 6, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('KIA-1011', 'Kia Picanto III', '', 'assets/images/products/kia/1011.png', 1, 100, 16.99, 6, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('KIA-1012', 'Kia Seltos I', '', 'assets/images/products/kia/1012.png', 1, 100, 16.99, 6, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('KIA-1013', 'Kia Mohave II', '', 'assets/images/products/kia/1013.png', 1, 100, 16.99, 6, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('KIA-1014', 'Kia Soul III кроссовер', '', 'assets/images/products/kia/1014.png', 1, 100, 16.99, 6, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('KIA-1015', 'Kia Sorento Prime III', '', 'assets/images/products/kia/1015.png', 1, 100, 16.99, 4, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('KIA-1016', 'Kia Sorento IV кроссовер', '', 'assets/images/products/kia/1016.png', 1, 100, 16.99, 6, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('KIA-1017', 'Kia Sportage IV-рестайлинг кроссовер', '', 'assets/images/products/kia/1017.png', 1, 100, 16.99, 6, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('KIA-1018', 'Kia cee''d XCeed кроссовер', '', 'assets/images/products/kia/1018.png', 1, 100, 16.99, 6, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('KIA-1019', ' Kia Carnival IV', '', 'assets/images/products/kia/1019.png', 1, 100, 16.99, 6, NOW());

--changeset jjasinovski:8

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('LADA-1000', 'Lada (ВАЗ) Vesta I', '', 'assets/images/products/lada/1000.png', 1, 100, 16.99, 7, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('LADA-1001', 'Lada (ВАЗ) Vesta Cross I', '', 'assets/images/products/lada/1001.png', 1, 100, 16.99, 7, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('LADA-1002', 'Lada (ВАЗ) Vesta Sport I', '', 'assets/images/products/lada/1002.png', 1, 100, 16.99, 7, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('LADA-1003', 'Lada (ВАЗ) Granta I', '', 'assets/images/products/lada/1003.png', 1, 100, 16.99, 7, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('LADA-1004', 'Lada (ВАЗ) Granta Cross I', '', 'assets/images/products/lada/1004.png', 1, 100, 16.99, 7, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('LADA-1005', 'Lada (ВАЗ) Vesta SW I', '', 'assets/images/products/lada/1005.png', 1, 100, 16.99, 7, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('LADA-1006', 'Lada (ВАЗ) Vesta SW Cross I', '', 'assets/images/products/lada/1006.png', 1, 100, 16.99, 7, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('LADA-1007', 'Lada (ВАЗ) Largus I', '', 'assets/images/products/lada/1007.png', 1, 100, 16.99, 7, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('LADA-1008', 'Lada (ВАЗ) Largus Cross I', '', 'assets/images/products/lada/1008.png', 1, 100, 16.99, 7, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('LADA-1009', 'Lada (ВАЗ) Granta I-рестайлинг универсал', '', 'assets/images/products/lada/1009.png', 1, 100, 16.99, 7, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('LADA-1010', 'Lada (ВАЗ) Granta I-рестайлинг хэтчбек', '', 'assets/images/products/lada/1010.png', 1, 100, 16.99, 7, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('LADA-1011', 'Lada (ВАЗ) Granta I-рестайлинг лифтбэк', '', 'assets/images/products/lada/1011.png', 1, 100, 16.99, 7, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('LADA-1012', 'Lada (ВАЗ) Niva I внедорожник', '', 'assets/images/products/lada/1012.png', 1, 100, 16.99, 7, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('LADA-1013', 'Lada (ВАЗ) XRAY I кроссовер', '', 'assets/images/products/lada/1013.png', 1, 100, 16.99, 7, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('LADA-1014', 'Lada (ВАЗ) XRAY Cross I', '', 'assets/images/products/lada/1014.png', 1, 100, 16.99, 7, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('LADA-1015', 'Lada (ВАЗ) 2121 (4x4) Niva Legend', '', 'assets/images/products/lada/1015.png', 1, 100, 16.99, 7, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('LADA-1016', 'Lada (ВАЗ) 2131 (4x4) Niva Legend', '', 'assets/images/products/lada/1016.png', 1, 100, 16.99, 7, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('LADA-1017', 'Lada (ВАЗ) Largus I рестайлинг фургон', '', 'assets/images/products/lada/1017.png', 1, 100, 16.99, 7, NOW());

--changeset jjasinovski:9

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MAZDA-1000', 'Mazda 6 III-2-ой рестайлинг седан', '', 'assets/images/products/mazda/1000.png', 1, 100, 16.99, 8, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MAZDA-1001', 'Mazda CX-30 I кроссовер', '', 'assets/images/products/mazda/1001.png', 1, 100, 16.99, 8, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MAZDA-1002', 'Mazda CX-5 II кроссовер', '', 'assets/images/products/mazda/1002.png', 1, 100, 16.99, 8, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MAZDA-1003', 'Mazda CX-9 II кроссовер', '', 'assets/images/products/mazda/1003.png', 1, 100, 16.99, 8, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MAZDA-1004', 'Mazda CX-9 II рестайлинг кроссовер', '', 'assets/images/products/mazda/1004.png', 1, 100, 16.99, 8, NOW());

--changeset jjasinovski:10

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1000', 'Mercedes-Benz CLA AMG C118 AMG седан', '', 'assets/images/products/mercedes\_benz/1000.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1001', 'Mercedes-Benz CLS C257 седан', '', 'assets/images/products/mercedes\_benz/1001.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1002', 'Mercedes-Benz S VII поколение (W223) седан', '', 'assets/images/products/mercedes\_benz/1002.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1003', 'Mercedes-Benz Maybach S VII поколение (Z 223) седан', '', 'assets/images/products/mercedes\_benz/1003.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1004', 'Mercedes-Benz A W177/V177 седан', '', 'assets/images/products/mercedes\_benz/1004.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1005', 'Mercedes-Benz C AMG W205/S205/C205/A205 - рестайлинг купе', '', 'assets/images/products/mercedes\_benz/1005.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1006', 'Mercedes-Benz C AMG купе', '', 'assets/images/products/mercedes\_benz/1006.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1007', 'Mercedes-Benz C AMG седан', '', 'assets/images/products/mercedes\_benz/1007.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1008', 'Mercedes-Benz C седан', '', 'assets/images/products/mercedes\_benz/1008.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1009', 'Mercedes-Benz C купе', '', 'assets/images/products/mercedes\_benz/1009.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1010', 'Mercedes-Benz E AMG седан', '', 'assets/images/products/mercedes\_benz/1010.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1011', 'Mercedes-Benz E купе', '', 'assets/images/products/mercedes\_benz/1011.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1012', 'Mercedes-Benz E седан', '', 'assets/images/products/mercedes\_benz/1012.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1013', 'Mercedes-Benz AMG GT купе', '', 'assets/images/products/mercedes\_benz/1013.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1014', 'Модификации и комплектации Mercedes-Benz A W177/V177 хэтчбек', '', 'assets/images/products/mercedes\_benz/1014.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1015', 'Mercedes-Benz E универсал', '', 'assets/images/products/mercedes\_benz/1015.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1016', 'Mercedes-Benz AMG GT хэтчбек', '', 'assets/images/products/mercedes\_benz/1016.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1017', 'Mercedes-Benz GLE Coupe кроссовер', '', 'assets/images/products/mercedes\_benz/1017.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1018', 'Mercedes-Benz GLA кроссовер', '', 'assets/images/products/mercedes\_benz/1018.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1019', 'Mercedes-Benz GLB внедорожник', '', 'assets/images/products/mercedes\_benz/1019.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1020', 'Mercedes-Benz GLE Coupe AMG кроссовер', '', 'assets/images/products/mercedes\_benz/1020.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1021', 'Mercedes-Benz GLS внедорожник', '', 'assets/images/products/mercedes\_benz/1021.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1022', 'Mercedes-Benz EQC кроссовер', '', 'assets/images/products/mercedes\_benz/1022.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1023', 'Mercedes-Benz GLE кроссовер', '', 'assets/images/products/mercedes\_benz/1023.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1024', 'Mercedes-Benz G внедорожник', '', 'assets/images/products/mercedes\_benz/1024.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1025', 'Mercedes-Benz G AMG внедорожник', '', 'assets/images/products/mercedes\_benz/1025.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1026', 'Mercedes-Benz GLC кроссовер', '', 'assets/images/products/mercedes\_benz/1026.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1027', 'Mercedes-Benz GLC AMG кроссовер', '', 'assets/images/products/mercedes\_benz/1027.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1028', 'Mercedes-Benz GLC Coupe кроссовер', '', 'assets/images/products/mercedes\_benz/1028.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1029', 'Mercedes-Benz GLC Coupe AMG кроссовер', '', 'assets/images/products/mercedes\_benz/1029.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1030', 'Mercedes-Benz Sprinter III микроавтобус', '', 'assets/images/products/mercedes\_benz/1030.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1031', 'Mercedes-Benz Sprinter III фургон', '', 'assets/images/products/mercedes\_benz/1031.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1032', 'Mercedes-Benz Vito W447 микроавтобус', '', 'assets/images/products/mercedes\_benz/1032.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1033', 'Mercedes-Benz Vito W447 фургон', '', 'assets/images/products/mercedes\_benz/1033.png', 1, 100, 16.99, 9, NOW());

INSERT INTO product (sku, name, description, image\_url, active, units\_in\_stock, unit\_price, category\_id,date\_created) VALUES ('MERCEDES\_BENZ-1034', 'Mercedes-Benz V W447 (III поколение) микроавтобус', '', 'assets/images/products/mercedes\_benz/1034.png', 1, 100, 16.99, 9, NOW());

--changeset jjasinovski:11

INSERT INTO `country` VALUES

(1,'BR','China'),

(2,'CA','Russia'),

(3,'DE','Germany'),

(4,'IN','Czech'),

(5,'TR','South Korea'),

(6,'US','United States');

--changeset jjasinovski:12

INSERT INTO `state` VALUES

(1,'Acre',1),

(2,'Alagoas',1),

(3,'Amapá',1),

(4,'Amazonas',1),

(5,'Bahia',1),

(6,'Ceará',1),

(7,'Distrito Federal',1),

(8,'Espírito Santo',1),

(9,'Goiás',1),

(10,'Maranhão',1),

(11,'Mato Grosso do Sul',1),

(12,'Mato Grosso',1),

(13,'Minas Gerais',1),

(14,'Paraná',1),

(15,'Paraíba',1),

(16,'Pará',1),

(17,'Pernambuco',1),

(18,'Piaui',1),

(19,'Rio de Janeiro',1),

(20,'Rio Grande do Norte',1),

(21,'Rio Grande do Sul',1),

(22,'Rondônia',1),

(23,'Roraima',1),

(24,'Santa Catarina',1),

(25,'Sergipe',1),

(26,'São Paulo',1),

(27,'Tocantins',1),

(28,'Alberta',2),

(29,'British Columbia',2),

(30,'Manitoba',2),

(31,'New Brunswick',2),

(32,'Newfoundland and Labrador',2),

(33,'Northwest Territories',2),

(34,'Nova Scotia',2),

(35,'Nunavut',2),

(36,'Ontario',2),

(37,'Prince Edward Island',2),

(38,'Quebec',2),

(39,'Saskatchewan',2),

(40,'Yukon',2),

(41,'Baden-Württemberg',3),

(42,'Bavaria',3),

(43,'Berlin',3),

(44,'Brandenburg',3),

(45,'Bremen',3),

(46,'Hamburg',3),

(47,'Hesse',3),

(48,'Lower Saxony',3),

(49,'Mecklenburg-Vorpommern',3),

(50,'North Rhine-Westphalia',3),

(51,'Rhineland-Palatinate',3),

(52,'Saarland',3),

(53,'Saxony',3),

(54,'Saxony-Anhalt',3),

(55,'Schleswig-Holstein',3),

(56,'Thuringia',3),

(57,'Andhra Pradesh',4),

(58,'Arunachal Pradesh',4),

(59,'Assam',4),

(60,'Bihar',4),

(61,'Chhattisgarh',4),

(62,'Goa',4),

(63,'Gujarat',4),

(64,'Haryana',4),

(65,'Himachal Pradesh',4),

(66,'Jammu & Kashmir',4),

(67,'Jharkhand',4),

(68,'Karnataka',4),

(69,'Kerala',4),

(70,'Madhya Pradesh',4),

(71,'Maharashtra',4),

(72,'Manipur',4),

(73,'Meghalaya',4),

(74,'Mizoram',4),

(75,'Nagaland',4),

(76,'Odisha',4),

(77,'Punjab',4),

(78,'Rajasthan',4),

(79,'Sikkim',4),

(80,'Tamil Nadu',4),

(81,'Telangana',4),

(82,'Tripura',4),

(83,'Uttar Pradesh',4),

(84,'Uttarakhand',4),

(85,'West Bengal',4),

(86,'Andaman and Nicobar Islands',4),

(87,'Chandigarh',4),

(88,'Dadra and Nagar Haveli',4),

(89,'Daman & Diu',4),

(90,'Lakshadweep',4),

(91,'Puducherry',4),

(92,'The Government of NCT of Delhi',4),

(93,'Alabama',6),

(94,'Alaska',6),

(95,'Arizona',6),

(96,'Arkansas',6),

(97,'California',6),

(98,'Colorado',6),

(99,'Connecticut',6),

(100,'Delaware',6),

(101,'District Of Columbia',6),

(102,'Florida',6),

(103,'Georgia',6),

(104,'Hawaii',6),

(105,'Idaho',6),

(106,'Illinois',6),

(107,'Indiana',6),

(108,'Iowa',6),

(109,'Kansas',6),

(110,'Kentucky',6),

(111,'Louisiana',6),

(112,'Maine',6),

(113,'Maryland',6),

(114,'Massachusetts',6),

(115,'Michigan',6),

(116,'Minnesota',6),

(117,'Mississippi',6),

(118,'Missouri',6),

(119,'Montana',6),

(120,'Nebraska',6),

(121,'Nevada',6),

(122,'New Hampshire',6),

(123,'New Jersey',6),

(124,'New Mexico',6),

(125,'New York',6),

(126,'North Carolina',6),

(127,'North Dakota',6),

(128,'Ohio',6),

(129,'Oklahoma',6),

(130,'Oregon',6),

(131,'Pennsylvania',6),

(132,'Rhode Island',6),

(133,'South Carolina',6),

(134,'South Dakota',6),

(135,'Tennessee',6),

(136,'Texas',6),

(137,'Utah',6),

(138,'Vermont',6),

(139,'Virginia',6),

(140,'Washington',6),

(141,'West Virginia',6),

(142,'Wisconsin',6),

(143,'Wyoming',6),

(144,'Adıyaman',5),

(145,'Afyonkarahisar',5),

(146,'Ağrı',5),

(147,'Aksaray',5),

(148,'Amasya',5),

(149,'Ankara',5),

(150,'Antalya',5),

(151,'Ardahan',5),

(152,'Artvin',5),

ПРИЛОЖЕНИЕ Б

**Заключение о результатах плагиат-проверки текста дипломной работы**

