1 АНАЛИЗ И КОНЦЕПТУАЛЬНОЕ МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

В данной курсовой работе будет рассмотрена информационная система «Магазин оружия».

Оружие – объекты, которые используются для нанесения вреда живым существам, структурам, системам. Они используются для повышения эффективности и действенности таких актов, как охота, самозащита, преступление, боевые действия.

Магазин оружия – хозяйственная единица, или объединение таких единиц, деятельность которых сводима к обороту, купле, продаже оружия. Внутреннюю структуру такой единицы можно представить, как тесно связанную структурированную целокупность средств, направленных на поддержание деятельности самой единицы, к которым относятся персонал, предназначенные для продажи товары, деньги, программные средства, и т.д.

Под товаром далее понимается единица оружия, его составная часть, или то, что может быть использовано в некоторой связи с оружием.

Данная система предназначена для просмотра пользователями информации о предлагаемых товарах, покупки товаров, организации хранения товаров, управления персоналом интернет – магазина. Система позволяет как покупателям, так и менеджерам узнать текущий статус заказа (местоположение, дату доставки). Администраторы могут добавлять, удалять, изменять информацию о товарах, настраивать гибкую систему категоризации.

Предметная область «Интернет - магазин оружия» на сегодняшний день сталкивается с такими проблемами, как недостаточная универсальность и низкая настраиваемость. Необходима возможность удобного изменения состава корзины, отслеживания своего заказа, взаимодействия с сотрудниками магазина. Разработанная информационная система предназначена для автоматизирования приведенных выше проблем и значительно упростит покупку оружия.

Функциональная структура схематически представлена на рисунке 1.1. Данная структура описывает основные взаимодействия, которые происходят в процессе использования информационной системы. Покупатель должен иметь возможность выбрать нужные ему товары, добавить их в корзину, оформить заказ, зарегистрировавшись на сайте, при необходимости связаться с сотрудником магазина, сделать и оплатить заказ, посматривать информацию о статусе заказа и доставке. Сотрудник магазина должен иметь возможность отвечать на вопросы покупателя, перемещать товары между магазинами и складами, переназначать сотрудников (они могут совмещать роли), добавлять новые товары (с указанием фотографий, различных характеристик и свойств), управлять категориями товаров, задавать свойства, которыми обладают все товары, входящие в заданную категорию, и значения этих свойств для конкретных товаров. Информация обо всех изменениях местоположения товаров также обязана храниться в базе данных.

После путешествия по сайту с целью ознакомления с ассортиментом предлагаемых товаров и заполнения корзины товарами, пользователь либо регистрируется и заполняет заказ одновременно, либо просто оформляет заказ, если он уже зарегистрирован. Можно поговорить с сотрудниками магазина. Сотрудники магазина могут настраивать множество товаров и предикаты, с помощью которых это множество может быть разделено на подмножества пользователем. Доступ к более низкоуровневым сущностям имеют только администраторы и высокопоставленные сотрудники магазина, имеющие соответствующее утверждение.

Организация работы данной системы является комплексной областью деятельности. В таком случае функциональными объектами системы являются администраторы, пользователи, сотрудники и сама программа (см. рис. 1.1).

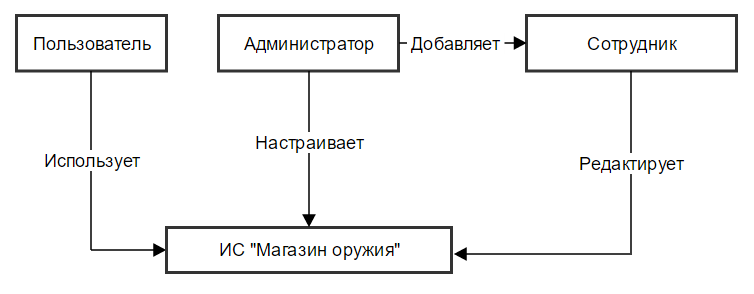


Рисунок 1.1 – Функциональная структура информационной системы «Магазин оружия»

Информационные потребности пользователей данной информационной системы следующие.

Для незарегистрированных покупателей:

1. регистрация на сайте;
2. вход на сайт;
3. просмотр информации о товарах с возможностью поиска, сортировки, фильтрации по части названия модели, части описания, категориям и конкретным значениям свойств;
4. просмотр информации о заказах;
5. настройка корзины;
6. связь с сотрудником магазина.

Для зарегистрированных покупателей:

1. все потребности незарегистрированных пользователей;
2. оформление заказа;
3. отслеживание заказа;
4. получать уведомления, когда отслеживаемый товар появится в наличии;
5. поиск товаров
6. просмотр подмножеств товаров,
7. замена множества категорий товаров.

Для сотрудников:

1. изменение товаров на сайте;
2. добавление товаров на сайт;
3. удаление товаров с сайта;
4. просмотр статистики о продажах.

Для администраторов:

1. все потребности сотрудников;
2. блокирование пользователей;
3. добавление новых сущностей (склады, магазины);
4. добавление новых сотрудников;
5. настройка полномочий сотрудников.

Документооборот предметной области «Магазин оружия» состоит из:

1. отчетов о продажах за определенный период (день, квартал, год);
2. заказов товаров (огнестрельное оружие, ножи, арбалеты, топоры, алебарды, перочинные ножи, патроны, картечь, противотанковые мины и пр.);
3. товарных накладных, выписываемых при внутренних перемещениях товаров, а также при поступлении новых;
4. инвентарных отчетов о товарах, находящихся на конкретном складе;
5. QR – наклеек товаров;
6. аналитических отчетов, которые предлагают закупить в следующем месяце больше тех товаров, которые будут популярны:
7. скидочных купонов.

Далее приведено описание объектов данной информационной системы и описание связей между ними.

Обобщенная схема взаимодействия объектов ПО изображается на рисунке 1.2. Рассмотрим ее подробнее.

Описание объектов ПО и их атрибутов:

1. покупатель: Id покупателя, ФИО, адрес, контактные телефоны, email адрес, хеш-сумма пароля;
2. товар: Id товара, модель, описание, размеры, вес, количество, цена, производитель, фотографии;
3. корзина: Id корзины, покупатель, список товаров с указанием их количества;
4. заказ: Id заказа, покупатель, список товаров с указанием их количества;
5. производитель: Id производителя, товары, название, страна;
6. категория: Id категории, имя, родительская категория, свойства категории;
7. свойство товара: Id свойства, тип свойства, допустимые значения, категория.

Описание ограничений целостности:

1. один производитель производит множество товаров;
2. к каждому типу товара могут прилагаться фотографии
3. каждый товар имеет свой тип
4. каждая категория включает типы товаров
5. один тип товара может относиться к разным категориям
6. каждая категория имеет ряд характеристик товаров;
7. каждому конкретному типу товара соответствует множество значений характеристик, которые определяются категориями, в которые этот товар входит.

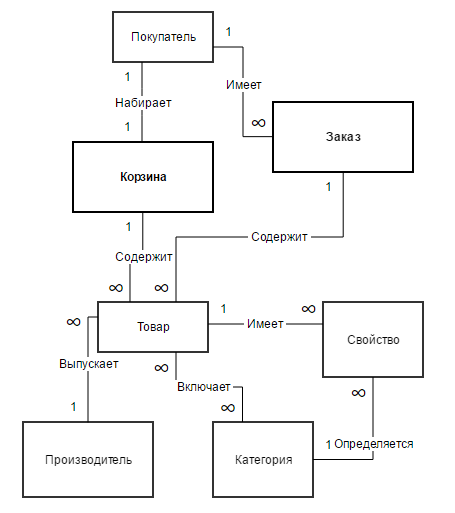


Рисунок 1.2 – Схема взаимосвязи объектов ПО

В данной предметной области присутствуют такие алгоритмические зависимости:

1. категория определяет набор свойств;
2. свойства могут иметь либо встроенный, либо пользовательский тип;
3. для пользовательских типов свойств задается множество допустимых значений;
4. свойства дочерней категории наследуются от родительской;
5. товар, входящий в дочернюю категорию, также входит в родительскую;
6. MD5-шифрование паролей пользователей системы;
7. оформление заказа представляет собой операцию по преобразованию текущей корзины в заказ, при этом она очищается
8. для каждого товара можно отследить его историю от изначального поступления до доставки конечному покупателю;
9. каждый товар соответствует одному реальному объекту;
10. анализ и предсказание популярности товаров;
11. генерация и считывание QR – символов;

Для выполнения поставленных задач и целей будет создана информационная система «Магазин оружия» средствами языка программирования C# в виде веб-приложения на основе технологии ASP.NET Core. Приложение будет разработано в среде Microsoft Visual Studio 2015 с использованием СУБД Microsoft SQL Server Express 2016.

2 ПОСТАНОВКА ЗАДАЧИ

Необходимо организовать процесс регистрации пользователя с внесением в базу данных информации о нём, а также с последующим входом в систему. Существует следующая информация о пользователе: фамилия, имя, отчество, адрес, email-адрес, номера телефонов, множество утверждений. Также пользователь после регистрации получает свой логин, пароль, и корзину.

Система должна выполнять следующие задачи:

1. изменение множества товаров путем добавления, удаления, изменения отдельных его элементов;
2. изменение множества товаров в корзине;
3. изменение множества категорий;
4. изменение подмножеств товаров, относящихся к данной категории;
5. изменение множества свойств товаров;
6. манипуляции с типами свойств товаров;
7. автоматическое создание бланка заказа;
8. автоматическое создание товарной накладной при перемещении товаров (список товаров, дата составления, источник и цель);
9. поиск товаров по части названия, принадлежности к категории, значению характеристики, комбинации вышеперечисленных признаков;
10. отображение множества товаров, получаемых посредством объединения категорий;
11. задача автоматизации загрузка данных;
12. отметка заказа подтвержденным;
13. изменение множества складов администратором;
14. изменение множества магазинов администратором;

3 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

3.1 Построение UML-диаграммы

В данной системе присутствуют 6 видов актеров, каждый из которых играет отведенную ему роль.

Актер «незарегистрированный покупатель» имеет возможность работать с корзиной. Незарегистрированный покупатель также может просматривать информацию о товарах и магазинах.

Актор «зарегистрированный покупатель» имеет возможность просматривать множества товаров, магазинов, своих заказов, корзину, сделать заказ.

Актор «сотрудник магазина» имеет возможность изменять множество товаров, изменять и добавлять категории и их содержимое, просматривать заказы.

Актор «сотрудник склада» может изменять множество товаров на складе.

Актор «администратор» имеет возможность изменять множества магазинов, складов, заказов, корзин, товаров, производителей, характеристик, пользователей сайта, категорий, выполнять произвольные запросы к базе данных.

В эту модель не включены разнообразные другие сотрудники магазина, так как благодаря системе утверждений, которая используется вместо традиционной системы ролей, их возможности могут формироваться самым различным образом, что невозможно отобразить на USE CASE диаграмме без ее чрезмерного переусложнения.

На основе этого составим USE CASE диаграмму предложенной системы (см. рис. 3.1).

Данная система будет иметь сервис-ориентированную архитектуру. Это позволит улучшить структуру системы и повысить ее безопасность.

Можно построить диаграмму развертывания (см. рис. 3.2).

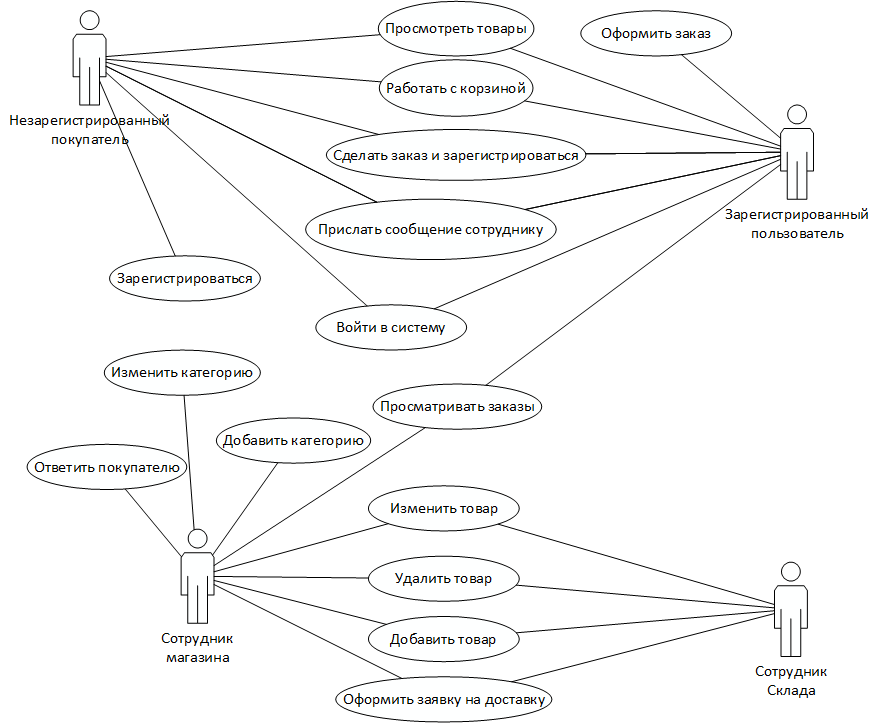


Рисунок 3.1 – USE CASE диаграмма «Магазин оружия»

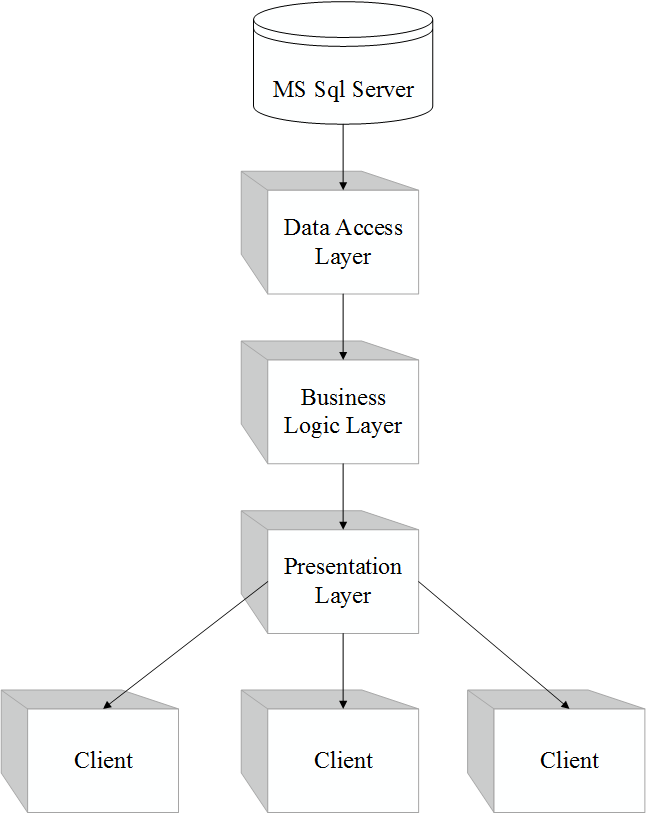


Рисунок 3.2 – Диаграмма развертывания «Магазин оружия»

3.2 Построение ER-диаграммы

На основании концептуальной модели предметной области, которая была получена в первом разделе можно получить такие сущности (см. рис. 3.3):

1. зависимая сущность «Тип Товара» с атрибутами модель, описание, вес, габариты, цена, первичным ключом Id, внешним ключом Id производителя;
2. зависимая сущность «Товар» с внешними ключами Id типа товара, Id заказа, Id корзины, Id склада, первичным ключом Id;
3. стержневая сущность «Производитель» с атрибутами название, страна, первичным ключом Id;
4. стержневая сущность «Покупатель» с атрибутами email, ФИО, телефоны, хеш пароля, первичным ключом Id;
5. зависимая сущность «Заказ» с атрибутом дата заказа, внешним ключом Id покупателя, первичным ключом Id;
6. ассоциативная сущность «КатегорияТовар» с первичным внешним ключом Id категории, первичным внешним ключом Id типа товара;
7. стержневая сущность «Категория» с первичным ключом Id, атрибутами название, внешним ключом Id родительской категории;
8. зависимая сущность «Характеристика» с атрибутами имя, тип, допустимые значения, первичным ключом Id, внешним ключом Id категории;
9. ассоциативная сущность «Значение Характеристики» с атрибутом значение, первичным внешним ключом Id характеристики, первичным внешним ключом Id типа товара;
10. стержневая сущность «Склад» с атрибутами адрес, название, координаты, первичным ключом Id;
11. зависимая сущность «Магазин» с атрибутами адрес, название, координаты, первичным ключом Id, внешним ключом Id склада;
12. ассоциативная сущность «Товар в Корзине» с первичным внешним ключом Id товара, первичным внешним ключом Id покупателя;
13. ассоциативная сущность «Товарная Накладная» с атрибутом дата составления, первичным ключом Id, внешними ключами Id первого склада, Id второго склада;
14. ассоциативная сущность «Товар в Накладной» с составным первичным внешним ключом из двух атрибутов Id товарной накладной, Id товара.

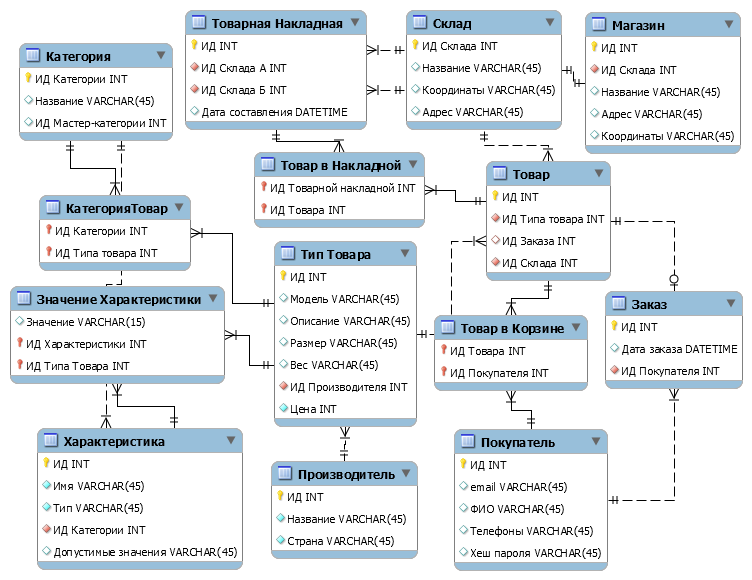


Рисунок 3.3 – ER диаграмма базы данных

3.3 Построение схемы реляционной базы данных в третьей нормальной форме

Покажем, что данная схема находится в 3НФ.

Все ассоциативные сущности, состоящие из 2-х ключевых атрибутов, не имеют неключевых атрибутов, и количества атрибутов недостаточно для образования транзитивной зависимости, то есть они уже находятся в 3НФ (КатегорияТовар, Товар в Корзине, Товар в Накладной).

Рассмотрим отношение «Категория», представленное на таблице 3.1. Единственный потенциальный ключ атомарен, ТФЗ отсутствуют.

Таблица 3.1 – Отношение «Категория»

|  |  |
| --- | --- |
| ИД\* | ─┐ |
| Название | ←┤ |
| Мастер-категория | ←┘ |

Рассмотрим отношение «Товарная накладная», представленное на таблице 3.2. Единственный потенциальный ключ атомарен, ТФЗ отсутствуют.

Таблица 3.2 – Отношение «Товарная накладная»

|  |  |
| --- | --- |
| ИД\* | ─┐ |
| Склад А | ←┤ |
| Склад Б | ←┤ |
| Дата | ←┘ |

Рассмотрим отношение «Склад», представленное на таблице 3.3. Единственный потенциальный ключ атомарен, ТФЗ отсутствуют.

Таблица 3.3 – Отношение «Склад»

|  |  |
| --- | --- |
| ИД\* | ─┐ |
| Название | ←┤ |
| Адрес | ←┤ |
| Координаты | ←┘ |

Рассмотрим отношение «Магазин», представленное на таблице 3.4. Единственный потенциальный ключ атомарен, ТФЗ отсутствуют.

Таблица 3.4 – Отношение «Магазин»

|  |  |
| --- | --- |
| ИД\* | ─┐ |
| Склад | ←┤ |
| Название | ←┤ |
| Адрес | ←┤ |
| Координаты | ←┘ |

Рассмотрим отношение «Товар», представленное на таблице 3.5. Единственный потенциальный ключ атомарен, ТФЗ отсутствуют.

Таблица 3.5 – Отношение «Товар»

|  |  |
| --- | --- |
| ИД\* | ─┐ |
| Тип | ←┤ |
| Склад | ←┤ |
| Заказ | ←┘ |

Рассмотрим отношение «Значение Характеристики», представленное на таблице 3.6. Единственный потенциальный ключ состоит из двух атрибутов, ТФЗ отсутствуют.

Таблица 3.6 – Отношение «Значение Характеристики»

|  |  |
| --- | --- |
| Характеристика\* | ─┐ |
| Тип Товара\* | ─┤ |
| Значение | ←┘ |

Рассмотрим отношение «Тип Товара», представленное на таблице 3.7. Может показаться, что Модель, Производитель – потенциальный ключ, однако это не так. Одинаковые модели могу иметь разное состояние, год выпуска, что отражено в описании. Таким образом, единственный потенциальный ключ атомарен, ТФЗ отсутствуют.

Таблица 3.7 – Отношение «Тип Товара»

|  |  |
| --- | --- |
| ИД\* | ─┐ |
| Модель | ←┤ |
| Описание | ←┤ |
| Размер | ←┤ |
| Вес | ←┤ |
| Цена | ←┤ |
| Производитель | ←┘ |

Рассмотрим отношение «Заказ», представленное на таблице 3.8. Единственный потенциальный ключ атомарен, ТФЗ отсутствуют.

Таблица 3.8 – Отношение «Заказ»

|  |  |
| --- | --- |
| ИД\* | ─┐ |
| Покупатель | ←┤ |
| Дата | ←┘ |

Рассмотрим отношение «Характеристика», представленное на таблице 3.9. Единственный потенциальный ключ атомарен, ТФЗ отсутствуют.

Таблица 3.9 – Отношение «Характеристика»

|  |  |
| --- | --- |
| ИД\* | ─┐ |
| Имя | ←┤ |
| Тип | ←┤ |
| Категория | ←┤ |
| Допустимые значения | ←┘ |

Рассмотрим отношение «Производитель», представленное на таблице 3.10. Единственный потенциальный ключ атомарен, ТФЗ отсутствуют.

Таблица 3.10 – Отношение «Производитель»

|  |  |
| --- | --- |
| ИД\* | ─┐ |
| Название | ←┤ |
| Страна | ←┘ |

Рассмотрим отношение «Покупатель», представленное на таблице 3.11. Единственный потенциальный ключ атомарен, ТФЗ отсутствуют.

Таблица 3.11 – Отношение «Покупатель»

|  |  |
| --- | --- |
| ИД\* | ─┐ |
| Почта | ←┤ |
| ФИО | ←┤ |
| Телефон | ←┤ |
| Пароль | ←┘ |

4 ОПИСАНИЕ ПРОГРАММЫ

4.1 Общие сведения

Данный продукт был реализован в виде веб-сайта, с серверной частью на языке C# с помощью фреймворка ASP.NET Core и клиентской частью на языке разметки HTML5 с использованием генератора шаблонов Razor, JavaScript и CSS3. Используется СУБД MS SQL Server 2016, доступ к которой осуществляется с помощью Entity Framework Core и технологии Code First. Также используются библиотеки Bootstrap, JQuery, AngularJS.

Итоговая программа занимает 50 мегабайт. Для запуска программы на персональном компьютере потребуется операционная система Windows, Linux, или Mac OS X, а также .NET Core SDK версии 1.0.0-rc2-3002702.

Для генерации отчетов понадобится пакет программ texlive.

Для установки программного продукта потребуется сервер Microsoft SQL Server.

4.2 Вызов и загрузка

Приложение поставляется в папке GunShop.

Внутри папки находится главный файл GunShop.dll, являющийся точкой входа. Для запуска необходимо выполнить команду «dotnet GunShop.dll».

Создание в локальной директории пользователя и подключение базы данных происходит автоматически при старте программы. При необходимости параметры подключения можно изменить в файле appsettings.json.

4.3 Назначение и логическая структура

При реализации программы «Магазин оружия» за основу была взята модель MVC (Model-View-Controller). В данной модели классы, связанные с логической обработкой данных, классы, связанные со взаимодействием с пользователем, и классы, отвечающие за визуальное представление данных, отделяются друг от друга.

Данная архитектурная модель является удобной и выгодной. Одно из основных ее достоинств состоит в том, что благодаря слабой связанности модификация некоторых частей не обязывает изменять другие.

Идея категорий такова. Все типы товаров не имеют свойств, то есть они совершенно безлики. Это однородное множество можно разделить на части с помощью предикатов, например предикат «является ножом» K(x) делит множество U на 2 класса: Ножи К и не-ножи NK. Однако возникает явная избыточность в случае с атомарными предикатами свойств, например «Является ножом с деревянной ручкой и металлическим лезвием». Подобный подход серьезно бы затруднил прикладное использование нашей системы. Поэтому «категорией» здесь называется либо составной предикат, либо предикат без характеристик. Входящие в состав предиката «свойства» или «характеристики» определяются категорией. При этом логично, что свойства категорий наследуются от старшей категории. Например, для категории А с характеристиками a1 и a1 и категории В с характеристиками b1 и b 1

∀x ((A(x)⇔a1(x)∧a2(x)) ∧ (B(x) ⇔ b1(x) ∧ b2(x)))

∀x (B(x) ⇒A(x))

Следовательно, ∀x (B(x) ⇔ a1(x) ∧ a2(x) ∧ b1(x) ∧ b2(x))

Все классы информационной системы на основании их смысла и назначения разбиты на следующие группы:

1. Связь базы данных с объектно-реляционным отображением представлен пространством имен Data и содержит классы ApplicationDbContext, а также миграции.
2. Модуль файлов представления представлен директорией wwwroot и содержит такие подкаталоги:
   1. Дополнительной логики представлений на языке javascript
   2. Стили представления сss
   3. Библиотеки javascript;
3. Вспомогательные утилиты в пространстве имен Utils, содержащие класс для создания и выполнения произвольных запросов к базе данных SqlMaker и класс генерации отчетов на языке разметки TeX и конвертации их в PDF
4. Файлы локализации .resx расположены в папке Resources
5. Классы, моделирующие предметную область расположены в пространстве имен Models. Для удобства для некоторых из них выделены интерфейсы в папке Interfaces
6. Сервисы, реализующие сложную логику программы и ее взаимодействие с базой данных, расположены в директории Services. Для каждого из них выделен интерфейс, используемый в инъекции зависимостей. Рассмотрим их:
   1. Сервис управления типами товаров и товарами CommoditiesService
   2. Сервис работы с корзиной ChartService
   3. Сервис категоризации и настройки категорий CategorizationService
   4. Сервис поиска SearchService
   5. Сервис управления локациями LocationsService
7. Контроллеры, выполняющие остальную серверную логику:
   1. AccountController, отвечающий за регистрацию и авторизацию пользователей
   2. CategoryController, выполняющий основные функции по категоризации и изменению категорий, их характеристик
   3. CommodityController, отвечающий за элементарную работу с товарами и их типами
   4. LocationController, отвечающий за управление складами и перемещение товаров между ними
   5. ReportsController, набирающий отчеты в формате TeX, конвертирующий их в PDF, удаляющий старые отчеты.
   6. SearchController, выполняющий поиск по категориям, характеристикам и названиям товаров
   7. StatisticsController, выполняющий статистику
   8. FileController, сохраняющий файлы
   9. HomeController, отвечает за общий обзор доступных функций, а также выполнение произвольных запросов
8. Директория Views, содержащая пользовательский интерфейс, соответстующий вышеперечисленным контроллерам
9. Директория ViewModels, содержащая сложные модели, составленные из данных нескольких таблиц (Business Objects), а также промежуточные классы для передачи данных между Views и Controllers

4.4 Описание физической модели базы данных

Схема физической модели базы данных приведена на рисунке 4.1. Ее описание содержится в таблице 4.1, таблице 4.2, таблице 4.3, таблице 4.4, таблице 4.5, таблице 4.6, таблице 4.7, таблице 4.8, таблице 4.9, таблице 4.10, таблице 4.11, таблице 4.12, таблице 4.13, таблице 4.14.

Таблица 4.1 – Поля таблицы «Categories»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Ограничения | Описание |
| Id | INT | PRIMARY KEY, IDENTITY (1, 1) NOT NULL | Идентификатор категории |
| MasterCategoryId | INT | FOREIGN KEY REFERENCES Category(Id),  NULL | Старшая категория (пуста только у корневой категории) |
| Name | NVARCHAR(MAX) | NOT NULL | Название категории |

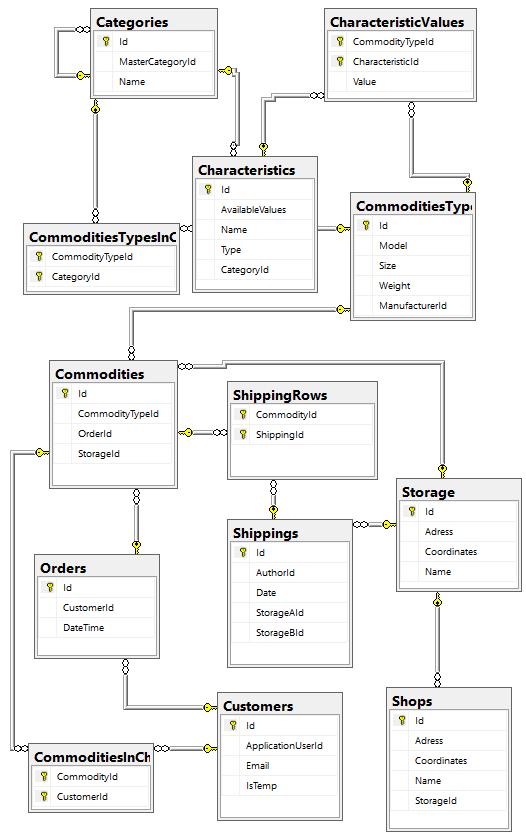


Рисунок 4.1 – Схема физической модели базы данных

Таблица 4.2 – Поля таблицы «CharacteristicValue»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Ограничения | Описание |
| CommodityTypeId | INT | PRIMARY KEY,  FOREIGN KEY REFERENCES CommodityType (Id),  NOT NULL | Идентификатор типа товара |
| CharacteristicId | INT | PRIMARY KEY,  FOREIGN KEY REFERENCES Characteristic (Id),  NOT NULL | Ссылка на Характеристику |
| Value | NVARCHAR(MAX) | NOT NULL | Значение |

Таблица 4.3 – Поля таблицы «Characteristics»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Ограничения | Описание |
| Id | INT | PRIMARY KEY, IDENTITY (1, 1) NOT NULL | Идентификатор характеристики |
| CategoryId | INT | FOREIGN KEY REFERENCES Category(Id),  NOT NULL | Ссылка на категорию |
| Type | NVARCHAR(MAX) | NOT NULL | Допустимый тип. Может быть C(Custom), I(Integer), D(Double), S(String). |
| AvailableValues | NVARCHAR(MAX) | NOT NULL | Допустимые строковые значения Custom типа-перечисления через «;» |
| Name | NVARCHAR(MAX) | NOT NULL | Имя характеристики |

Таблица 4.4 – Поля таблицы «CommoditiesTypesInCategory»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Ограничения | Описание |
| CommodityTypeId | INT | FOREIGN KEY REFERENCES CommodityType(Id),  NOT NULL | Идентификатор товара |
| CategoryId | INT | FOREIGN KEY REFERENCES Category(Id),  NOT NULL | Ссылка на категорию |

Таблица 4.5 – Поля таблицы «Manufacturers»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Ограничения | Описание |
| Id | INT | PRIMARY KEY, IDENTITY (1, 1) NOT NULL | Идентификатор |
| Country | NVARCHAR(MAX) | NOT NULL | Страна производителя |
| Name | NVARCHAR(MAX) | NOT NULL | Название компании |

Таблица 4.6 – Поля таблицы «CommoditiesTypes»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Ограничения | Описание |
| Id | INT | PRIMARY KEY, IDENTITY (1, 1) NOT NULL | Идентификатор товара |
| ManufacturerId | INT | FOREIGN KEY REFERENCES Manufacturer (Id),  NOT NULL | Идентификатор производителя |
| Weight | INT | NOT NULL | Вес товара в граммах |
| Size | NVARCHAR(MAX) | NOT NULL | Размеры(мм) в формате Д;Ш;В |
| Model | NVARCHAR(MAX) | NOT NULL | Название модели |

Таблица 4.7 – Поля таблицы «Commodities»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Ограничения | Описание |
| Id | INT | PRIMARY KEY, IDENTITY (1, 1) NOT NULL | Идентификатор товара |
| CommodityTypeId | INT | FOREIGN KEY REFERENCES CommodityType (Id),  NOT NULL | Ссылка на тип данного товара |
| OrderId | INT | FOREIGN KEY REFERENCES Order (Id),  NULL | Ссылка на заказ |
| StorageId | INT | FOREIGN KEY REFERENCES Storage (Id),  NOT NULL | Ссылка на склад(изначально все товары хранятся на складе ROOT) |

Таблица 4.8 – Поля таблицы «ShippingRows»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Ограничения | Описание |
| CommodityId | INT | PRIMARY KEY, FOREIGN KEY REFERENCES Commodity (Id),  NOT NULL | Идентификатор товара |
| ShippingId | INT | PRIMARY KEY, FOREIGN KEY REFERENCES Shipping (Id),  NOT NULL | Ссылка на накладную |

Таблица 4.9 – Поля таблицы «Orders»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Ограничения | Описание |
| Id | INT | PRIMARY KEY, IDENTITY (1, 1) NOT NULL | Идентификатор заказа |
| CustomerId | INT | FOREIGN KEY REFERENCES Customer (Id),  NOT NULL | Идентификатор покупателя |
| DateTime | DATETIME2(7) | NOT NULL | Дата составления заказа |

Таблица 4.10 – Поля таблицы «Shippings»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Ограничения | Описание |
| Id | INT | PRIMARY KEY, IDENTITY (1, 1) NOT NULL | Идентификатор накладной |
| AuthorId | INT | FOREIGN KEY REFERENCES AspNetUser (Id),  NOT NULL | Составитель накладной |
| StorageAId | INT | FOREIGN KEY REFERENCES Storage (Id),  NOT NULL | Начальный склад |
| StorageBId | INT | FOREIGN KEY REFERENCES Storage (Id),  NOT NULL | Конечный склад |
| Date | DATETIME2(7) | NOT NULL | Дата составления |

Таблица 4.11 – Поля таблицы «Storages»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Ограничения | Описание |
| Id | INT | PRIMARY KEY, IDENTITY (1, 1) NOT NULL | Идентификатор склада |
| Coordinates | NVARCHAR(MAX) | NOT NULL | Координаты склада в формате «xx.xx,yy.yy» |
| Address | NVARCHAR(MAX) | NOT NULL | Адрес склада |
| Name | NVARCHAR(MAX) | NOT NULL | Название склада |

Таблица 4.12 – Поля таблицы «CommoditiesInCharts»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Ограничения | Описание |
| CustomerId | INT | PRIMARY KEY, FOREIGN KEY REFERENCES Customer (Id),  NOT NULL | Идентификатор покупателя |
| CommodityId | INT | PRIMARY KEY, FOREIGN KEY REFERENCES Commodity (Id),  NOT NULL | Идентификатор товара |

Таблица 4.13 – Поля таблицы «Customers»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Ограничения | Описание |
| Id | INT | PRIMARY KEY, IDENTITY (1, 1) NOT NULL | Идентификатор |
| ApplicationUserId | NVARCHAR(MAX) | FOREIGN KEY REFERENCES AspNetUsers(Id),  NULL | Системный пользователь |
| IsTemp | BIT | NOT NULL | Является ли пользователь зарегистрированным |
| Email | NVARCHAR(MAX) | NOT NULL | Электронная почта пользователя |

Таблица 4.14 – Поля таблицы «Shops»

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Тип данных | Ограничения | Описание |
| Id | INT | PRIMARY KEY, IDENTITY (1, 1) NOT NULL | Идентификатор |
| Coordinates | NVARCHAR(MAX) | NOT NULL | Координаты склада в формате «xx.xx,yy.yy» |
| Name | NVARCHAR(MAX) | NOT NULL | Название магазина |
| StorageId | INT | FOREIGN KEY REFERENCES Storages(Id),  NOT NULL | Главный склад, привязанный к данному магазину |
| Address | NVARCHAR(MAX) | NOT NULL | Адрес |

4.5 Описание программной реализации

При первом запуске программы открывается обзор всех товаров (рисунок 4.2). Отсюда можно добавить товар в корзину, просматривать категории товаров, перейти к детальному обзору товара, сменить язык. Администратор может добавить товар к категории.

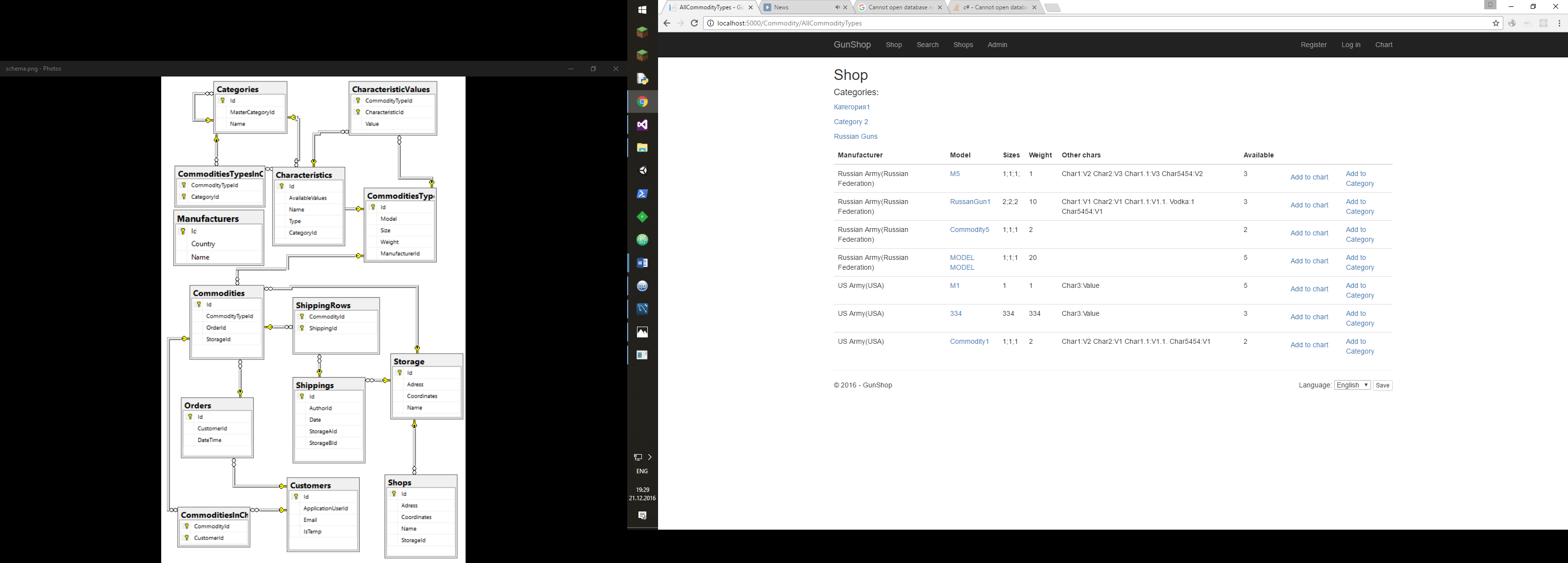


Рисунок 4.2 – Главная страница

Перейдем к странице администратора, нажав на кнопку «Admin» на верхней панели. На странице (рисунок 4.3) отображены ссылки на страницы обзора магазина, добавления производителя, добавления типа товара, очищения анонимных корзин, добавления категории, поиска, обзора магазинов и складов, обзора накладных, произвольного запроса, статистики.

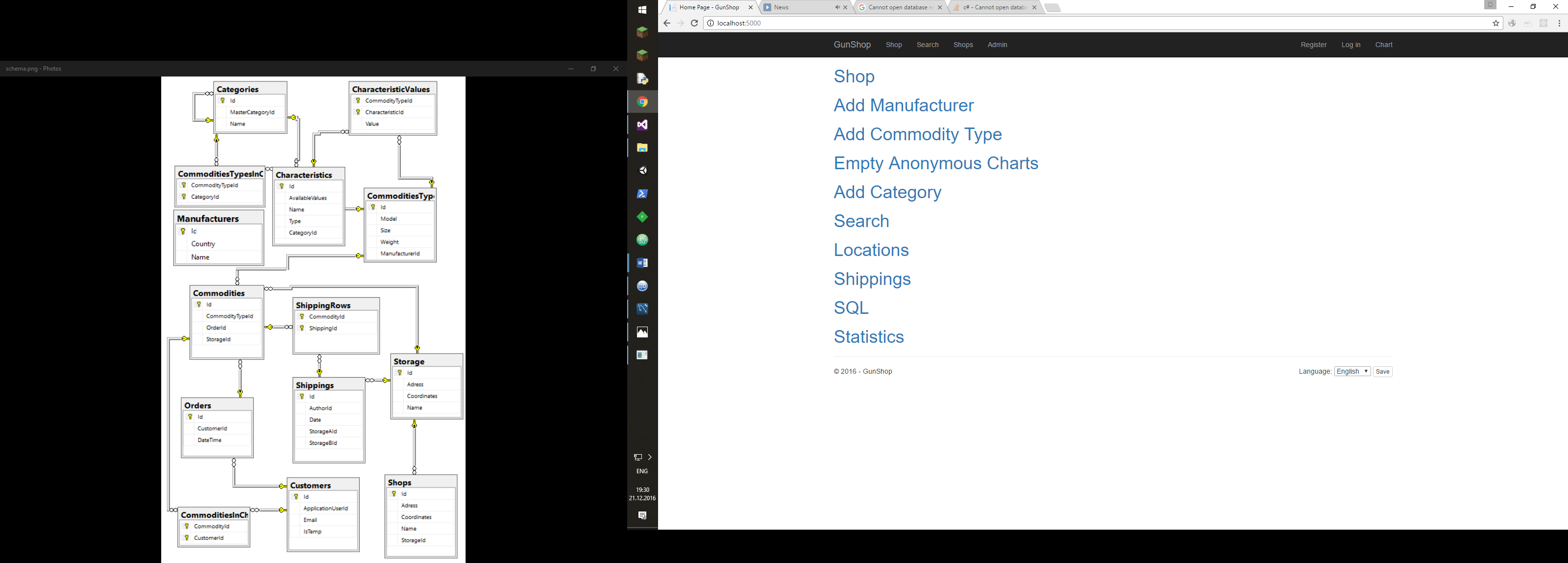


Рисунок 4.3 – Панель администратора

С первой мы уже ознакомились. Добавим производителя (рисунок 4.4). Введем название страны и завода в соответствующие поля, и нажмем кнопку «Create».

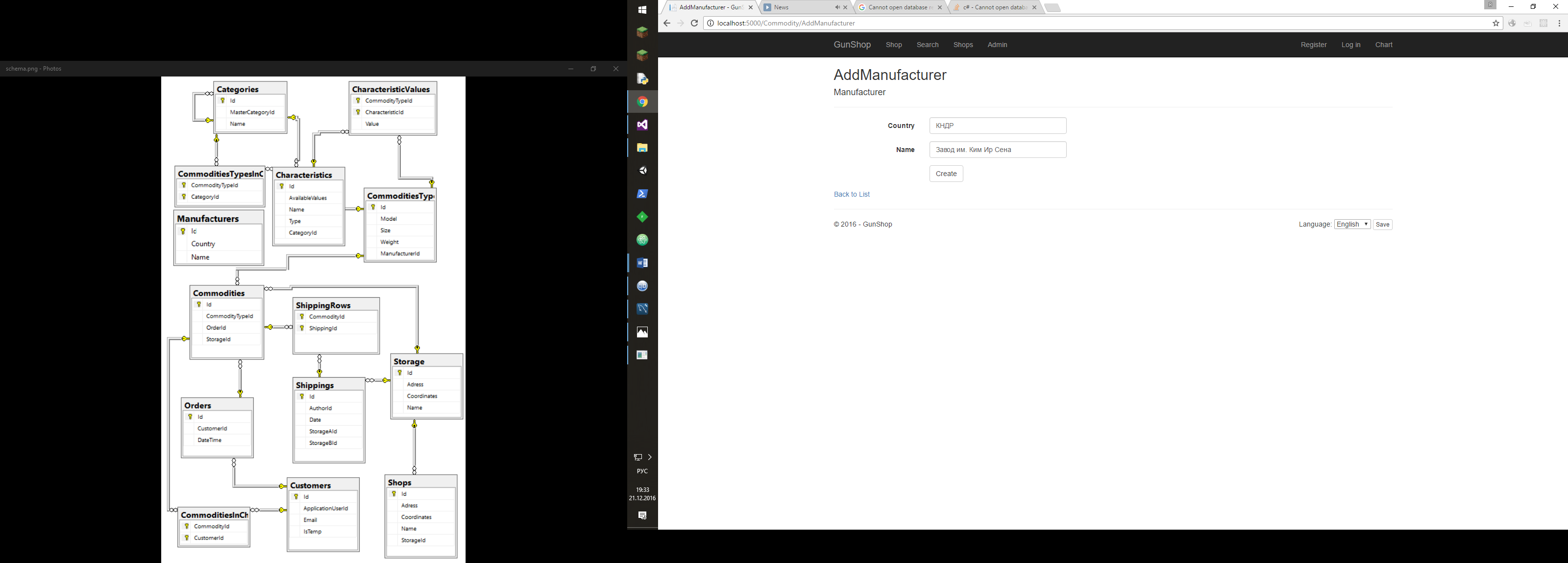


Рисунок 4.4 – Добавление производителя

Вернувшись на страницу администратора, добавим категорию (рисунок 4.5). Для добавления новой характеристики необходимо ввести ее имя в поле ввода, нажать клавишу перевода каретки, ввести в поле допустимых значений допустимые значения в формате «Значение1; Значение2; …». Мы будем посланы на страницу магазина, где уже добавлена наша категория. Добавим подкатегорию, повторив наши действия, но изменив значение полей характеристик и старшей категории.

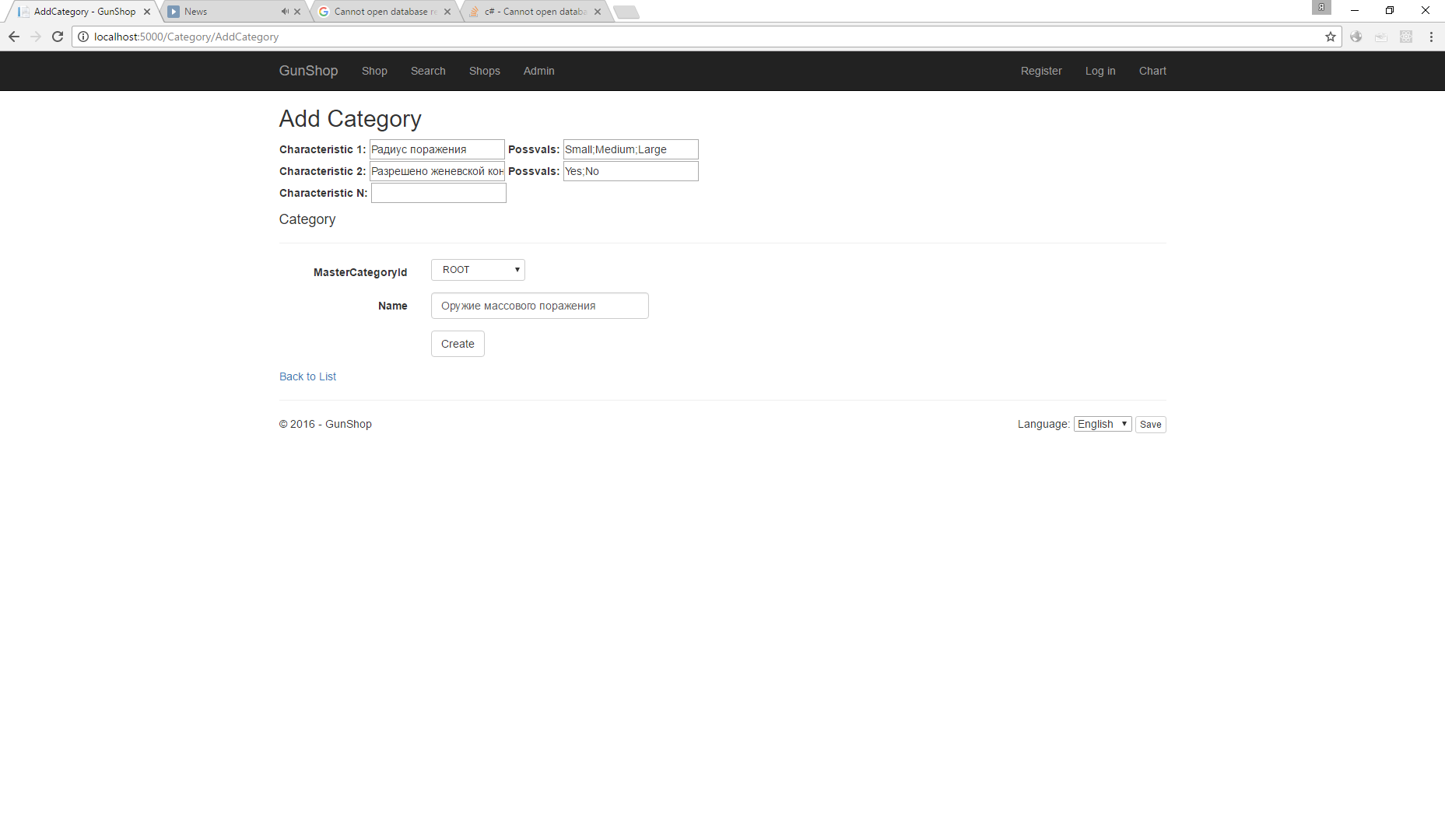


Рисунок 4.5 – Добавление категории

Теперь добавим новый магазин, заполнив форму в верхней части соответствующей страницы (рисунок 4.6) и нажав кнопку «Add». Теперь все готово для добавления нового типа товара.

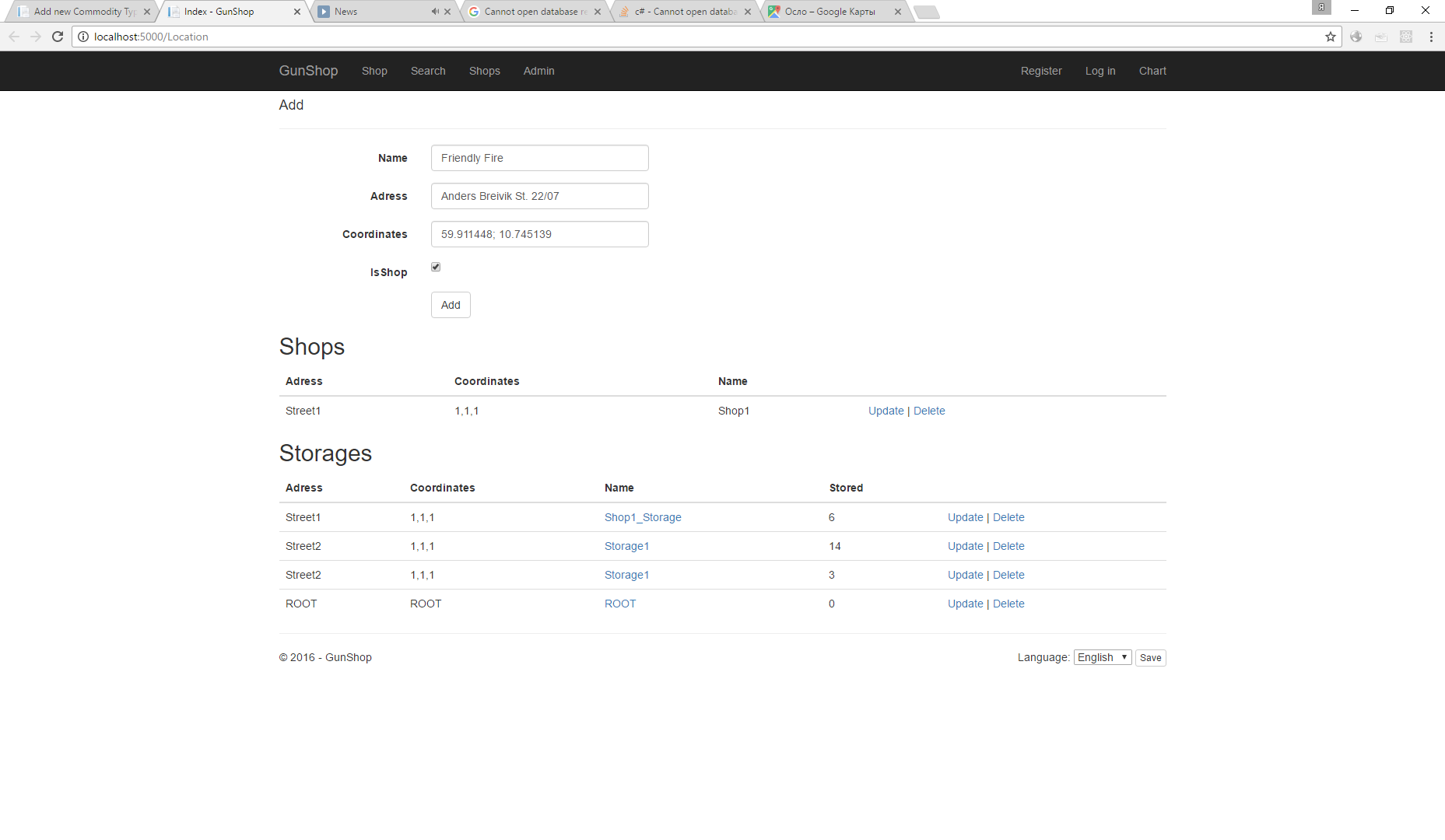


Рисунок 4.6 – Добавление магазина или склада

Чтобы добавить новый тип товара, заполним необходимые поля соответствующей формы (рисунок 4.7), указав в качестве склада, к примеру «ROOT».

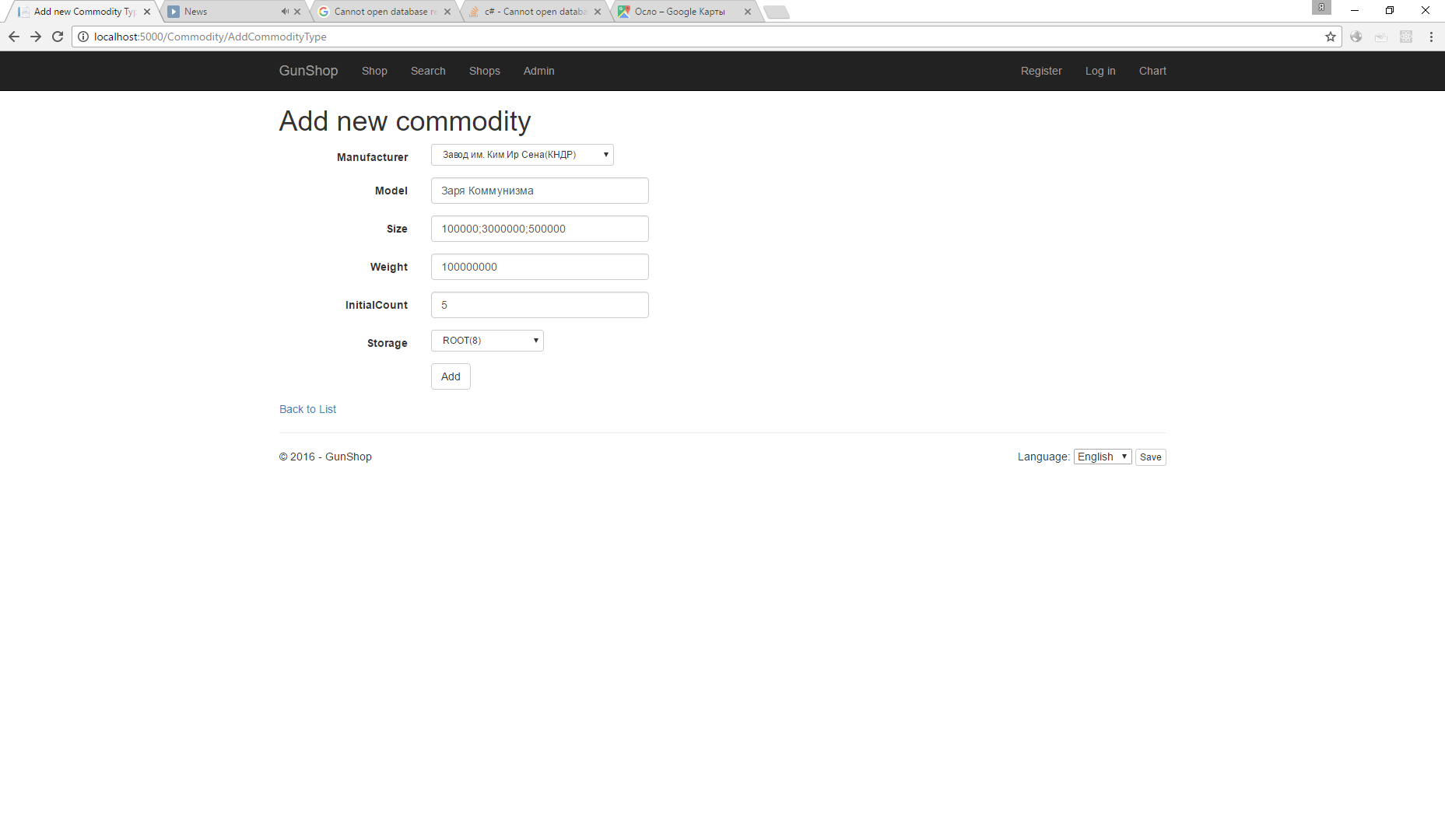


Рисунок 4.7 – Добавление товара

Теперь мы создали товар в главной категории и не приписанный к складу. Добавим его в категорию, созданную ранее. Для этого найдем его на главной странице и нажмем на кнопку «Add to Category» (рисунок 4.2). Перед нами откроется список доступных категорий, из которого нужно выбрать нужную нам. Выберем значения характеристик из выпадающих списков и нажмем кнопку «Create». В результате окажемся на странице обзора товаров в только что выбранной категории. Впоследствии можно добавить наш товар в те категории, в которые он еще не входит.

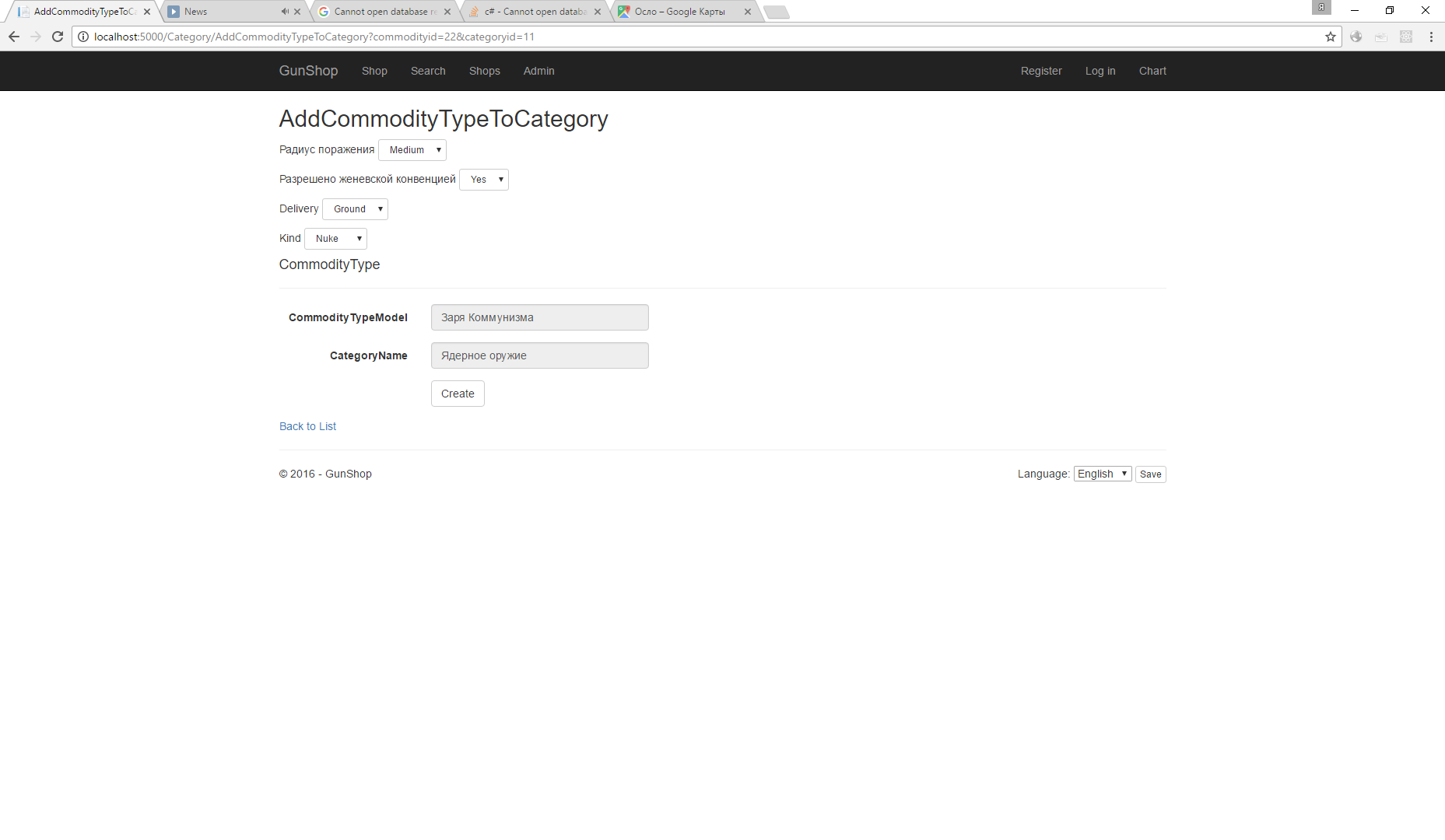


Рисунок 4.8 – Добавление товара в категорию

Теперь перейдем к скучным и бесполезным функциям, вроде поиска (рисунок 4.9). Для формирования поискового запроса необходимо ввести поле модели часть названия модели, отметить галочками необходимые категории и точечками необходимые значения характеристик и нажать на кнопку «Поиск». В результате на странице отобразится только соответствующие товары. Нажав на кнопку «Details», мы можем просмотреть подробную информацию о товаре (рисунок 4.10).

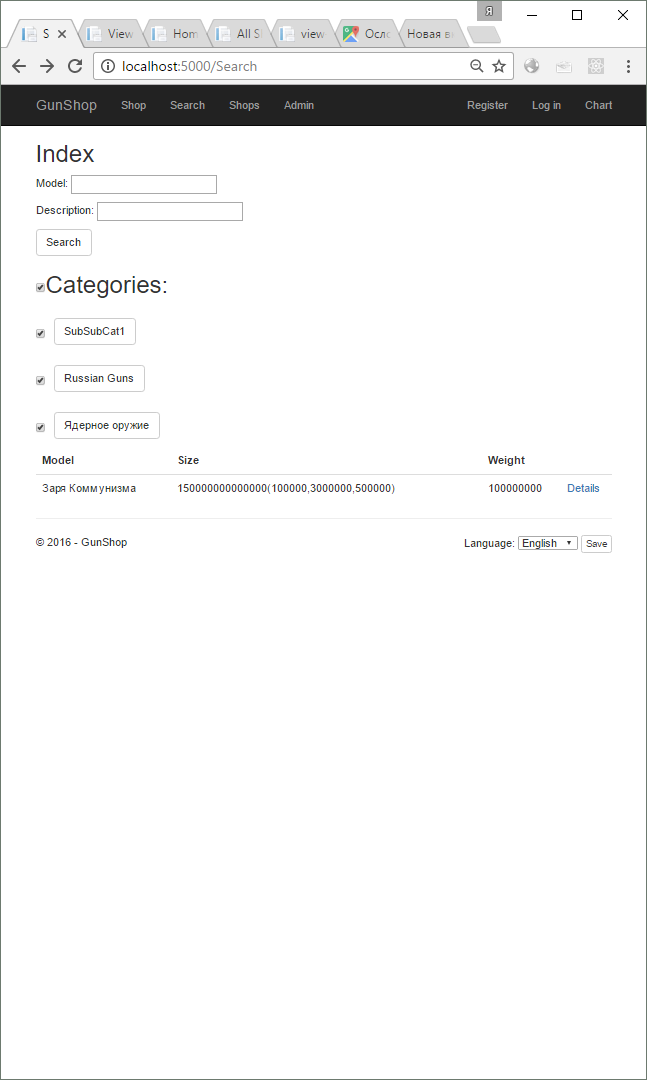


Рисунок 4.9 – Поиск

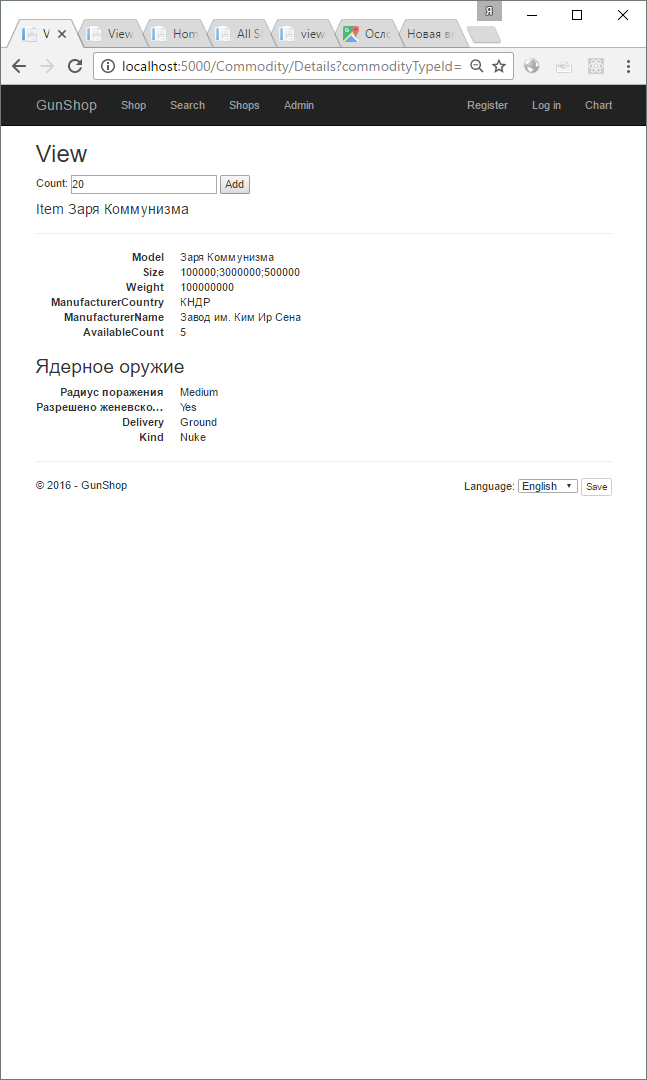


Рисунок 4.10 – Детальный обзор типа товара

Переместим товар на новый склад. На странице мест нажмем на имя склада, на котором хранится наш товар. Перед нами предстанет форма, отображенная на рисунке 4.11. В верхнем выпадающем списке выберем желаемый склад, справа от каждой единицы хранения отметим галочкой, что мы хотим ее переместить. После нажатия кнопки «Переместить», мы увидим наши товары уже в обзоре нового склада. Нажав на кнопку «Получить PDF отчет» для нас будет сверстан загружаемый отчет (рисунок 4.12).

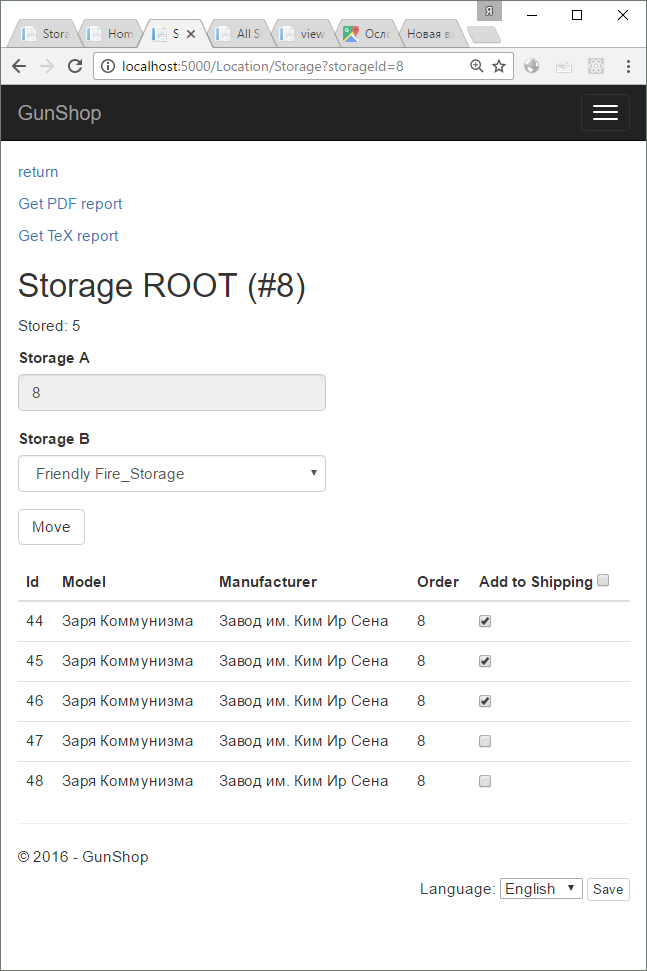


Рисунок 4.11 – Склад

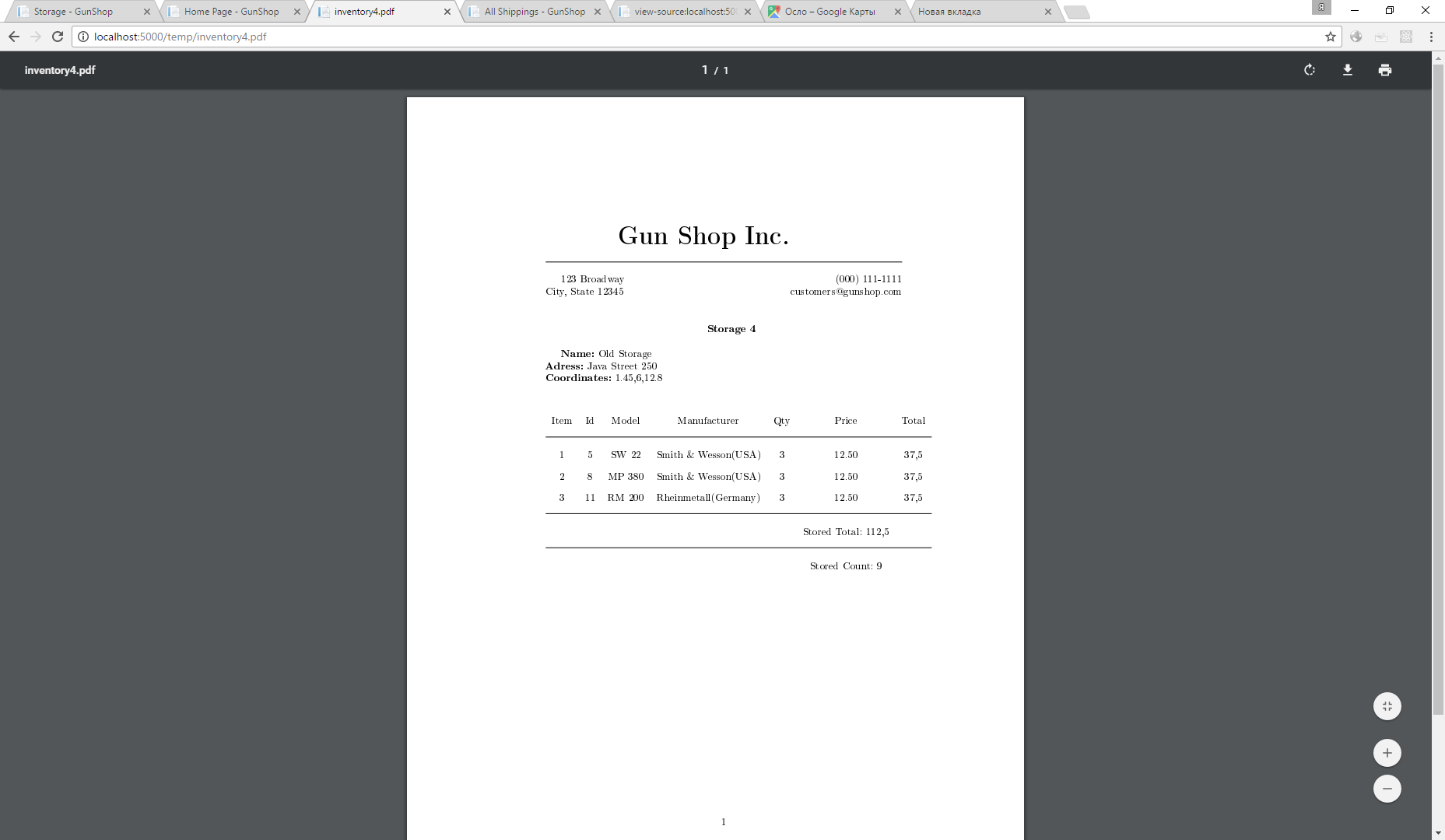


Рисунок 4.12 – Пример инвентарного отчета

Теперь рассмотрим накладные. Чтобы сделать это, перейдем на страницу «Накладные» (рисунок 4.13). Выберем накладную из списка и нажмем кнопку «Показать». Выпадет список товаров. Нажмем на кнопку «PDF» справа, если нам нужен этот формат для нашей накладной. Если нужен исходный код в формате TeX, нажмем на соседнюю соответствующую кнопку. Скорее всего документ (рисунок 4.14) откроется в браузере, оттуда же его можно скачать. Иногда бывает необходимо перезагрузить страницу.

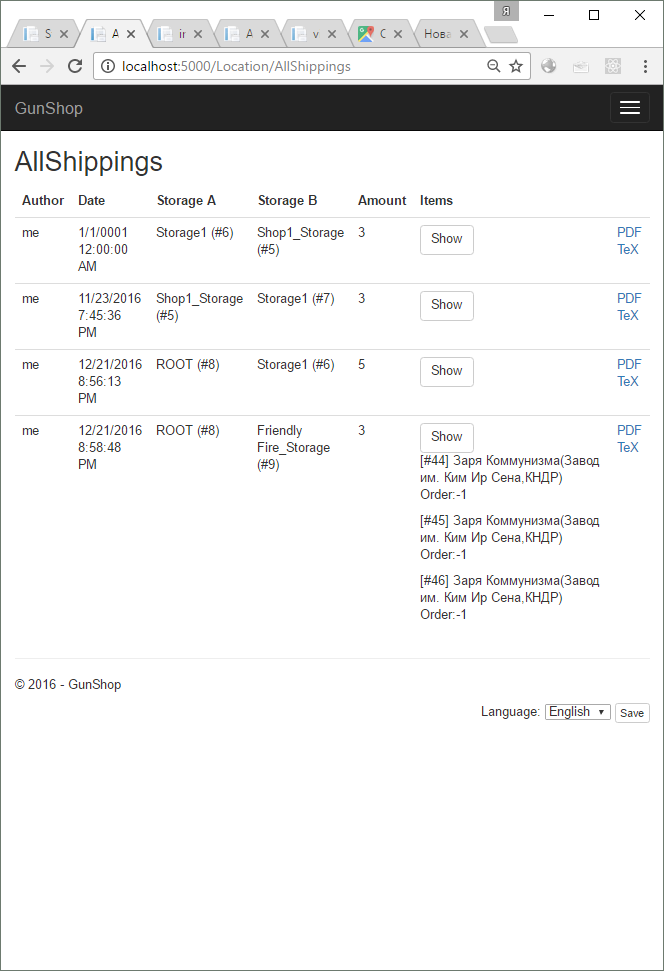


Рисунок 4.13 – Накладные

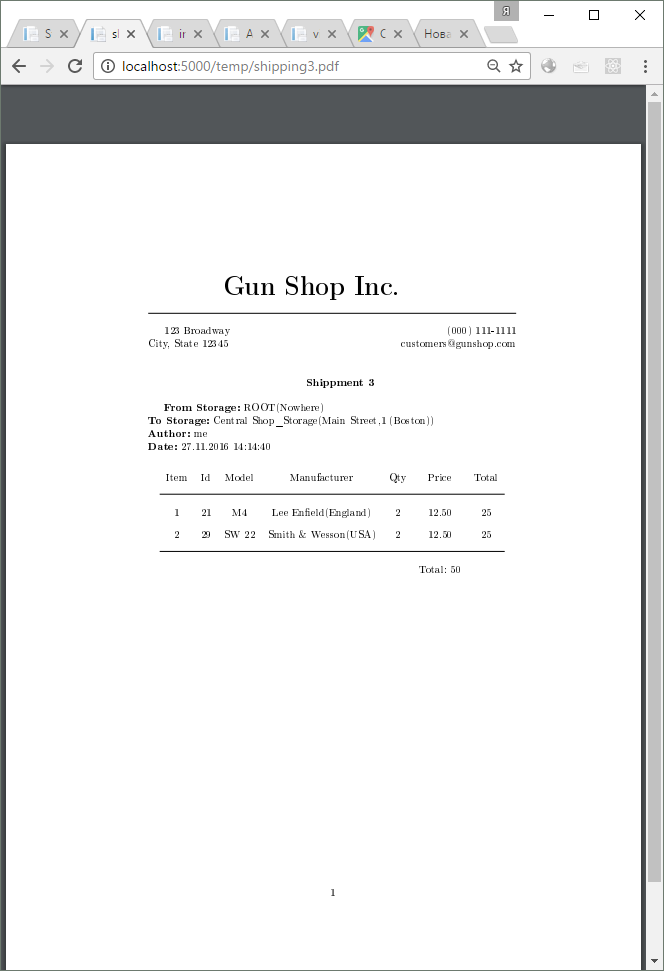


Рисунок 4.14 – Пример накладной

Для того, чтобы попасть на страницу произвольных запросов SQL, попадем на страницу произвольных запросов SQL. Введем в единственное поле ввода наш запрос и выполним его. Получим отформатированный вид содержимого таблицы (рисунок 4.15).

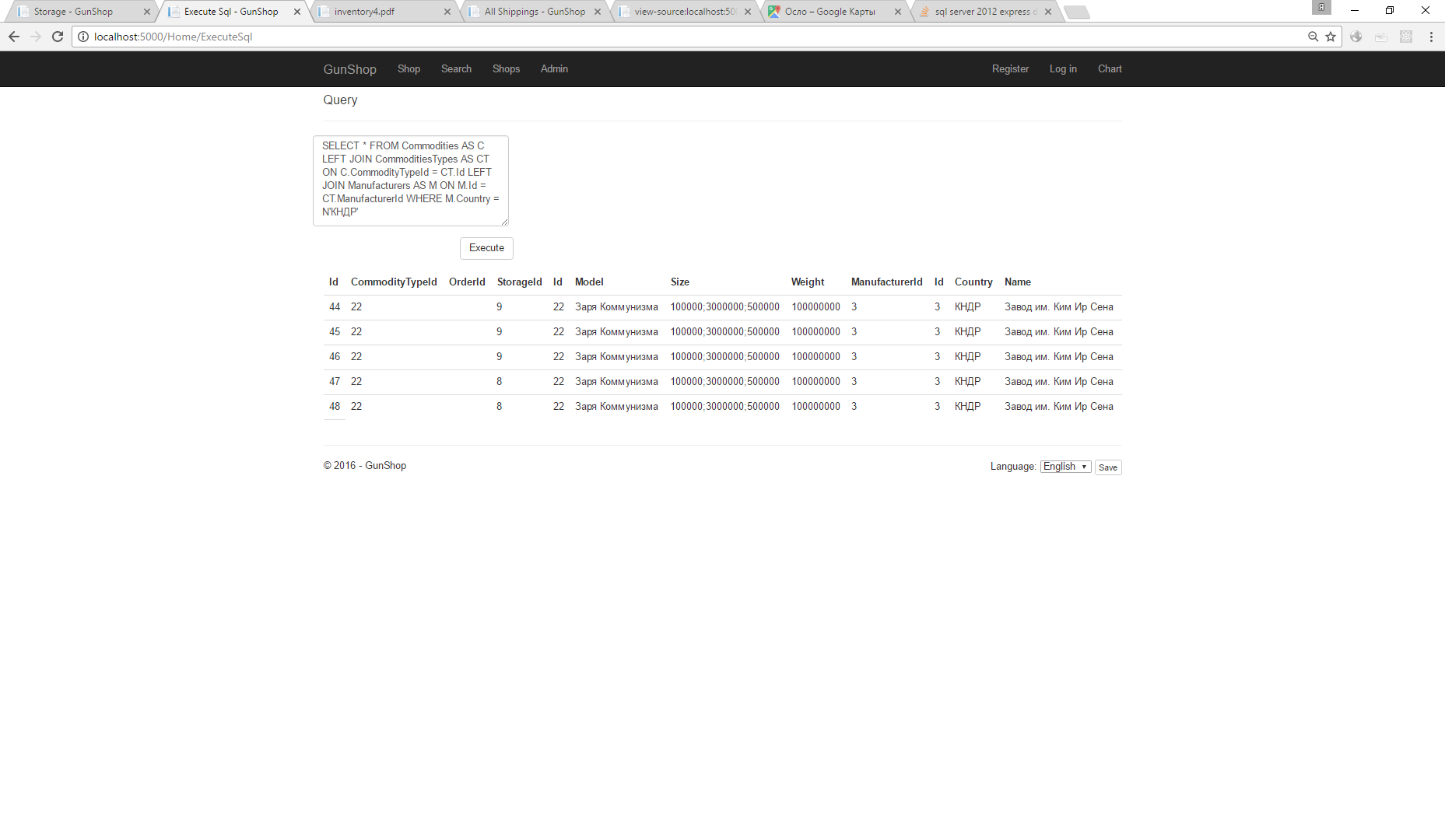


Рисунок 4.15 – Страница произвольных запросов

Наконец, рассмотрим страницу статистики (рисунок 4.16). На ней в виде интерактивных диаграмм изображено распределение товаров по производителям, складам, корзинам пользователей.

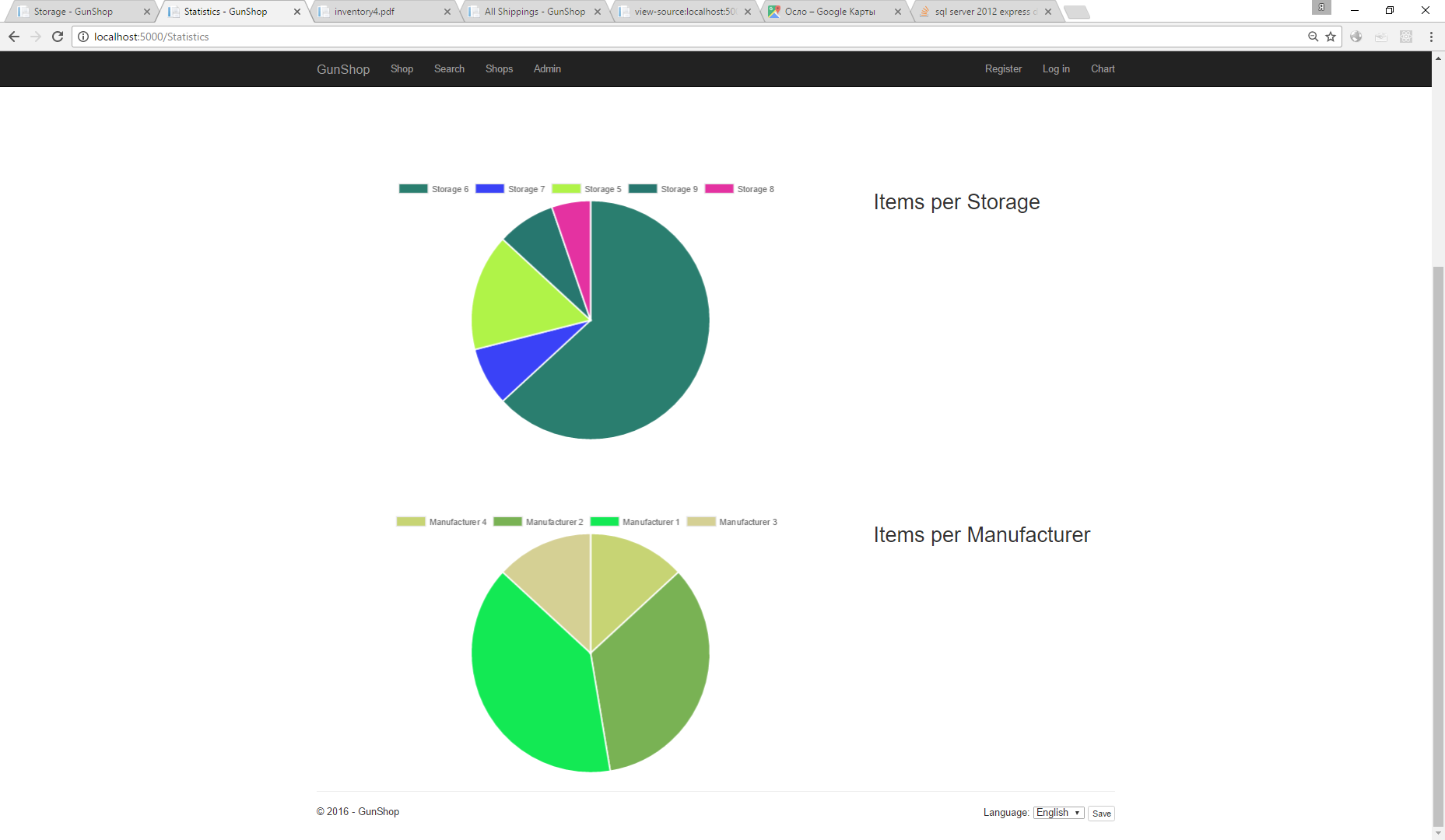


Рисунок 4.16 – Страница статистики

4.6 Описание задачи автоматизации

Задачей автоматизации является загрузка производителей из файла в формате xml. Пример файла приведен ниже:

<?xml version="1.0" encoding="utf-8"?>

<ArrayOfManufacturer xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<Manufacturer>

<Name>M1</Name>

<Country>C1</Country>

</Manufacturer>

<Manufacturer>

<Name>M2</Name>

<Country>C2</Country>

</Manufacturer>

</ArrayOfManufacturer>

Для выполнения этой операции необходимо нажать на кнопку «Выбрать файл» в нижней части панели администратора. Далее надо выбрать с помощью встроенного в браузер проводника (рисунок 4.17) файл XML. После того, как загрузка будет завершена, будет выведено окно, изображенное на рисунке 4.18. Далее приведен код серверной части.

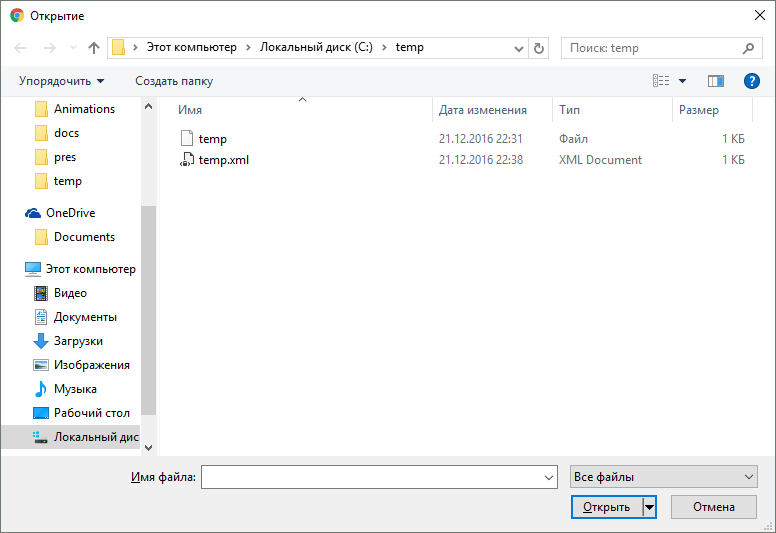


Рисунок 4.17 – Окно выбора файла

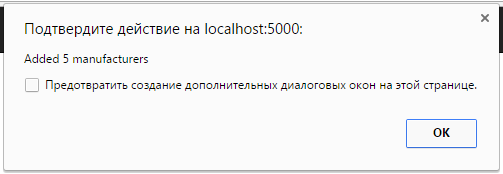


Рисунок 4.18 – Всплывающее окно

public class FileController : Controller

{

private IHostingEnvironment hostingEnvironment;

private Random random;

private ApplicationDbContext \_context;

public FileController(IHostingEnvironment \_hostingEnvironment,

ApplicationDbContext context)

{

hostingEnvironment = \_hostingEnvironment;

random = new Random();

\_context = context;

}

private async Task<string> addFile(IFormFile file)

{

string chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";

int len = 10;

var filename = file.FileName;

var extension = filename.Split('.').Last();

filename = filename.Replace(extension, "");

filename = new string(Enumerable.Repeat(chars, len)

.Select(s => s[random.Next(s.Length)]).ToArray());

filename = filename + "." + extension;

var uploads = Path.Combine(hostingEnvironment.WebRootPath, "temp");

var filepath = Path.Combine(uploads, filename);

using (var fileStream = new FileStream(filepath, FileMode.Create))

{

await file.CopyToAsync(fileStream);

}

return filepath;

}

public async Task<IActionResult> AddManufacturersXML(IFormFile file)

{

var res = LoadXML<Manufacturer>(await addFile(file));

\_context.Manufacturers.AddRange(res);

await \_context.SaveChangesAsync();

return Content(res.Count.ToString());

}

[HttpPost]

public async Task<IActionResult> AddFile(IFormFile file)

{

return Content(await addFile(file));

}

public static void SaveXML<T>(string filePath, List<T> elems)

{

XmlSerializer writer = new XmlSerializer(typeof(List<T>));

using (var file = new StreamWriter(new FileStream(filePath, FileMode.Open)))

{

writer.Serialize(file, elems);

}

}

public static List<T> LoadXML<T>(string filePath)

{

if (!System.IO.File.Exists(filePath))

{

throw new ArgumentException("File not found!");

}

XmlSerializer writer = new XmlSerializer(typeof(List<T>));

using (var file = new StreamReader(new FileStream(filePath, FileMode.Open)))

{

return (List<T>)writer.Deserialize(file);

}

}

}

Далее приведен соответствующий клиентский код:

<script>

function uploadManufacturers(){

var data = new FormData($("#uploadForm")[0]);

$.ajax({

url: "/File/AddManufacturersXML",

type: "POST",

data: data,

contentType: false,

dataType: false,

processData: false,

success: function (r) {

alert("Added " + r + " manufacturers");

}

});

}

</script>

<form id="uploadForm" enctype="multipart/form-data">

<p>Manufacturers:

<input type="file" name="file" onchange="uploadManufacturers()" /></p>

</form>

ВЫВОДЫ

В курсовом проекте были реализованы некоторые функции информационной системы магазина оружия. Были изучены основы работы с базами данных, получен опыт разработки и проектирования баз данных, а также разработки программных продуктов с использованием баз данных, а также осмотрены границы использования реляционной теории в ее связи с логикой.

Несмотря на то, что пользовательский интерфейс и различные вспомогательные функции реализованы не в полном объеме, единственная главная задача выполнена. Ответ на вопрос «Можно ли создать достаточно универсальную систему работы с множествами объектов реального мира средствами реляционных БД» положительный.

Почти все второстепенные запланированные в постановке задачи цели не выполнены. Тем не менее, это совершенно не важно, так как они были лишены всякого смысла. Стоит отметить, что некоторая возможная субъективная сложность в восприятии идеи данной системы не является виной её автора, так как более простого ответа на вышеприведенный вопрос нет. В ходе создания разнообразных бесполезных функций, однако, был извлечен некоторый побочный практический эффект, заключающийся в том, что система категоризации работает как следует, и может быть перенесена в другие приложения.

ПЕРЕЧЕНЬ ССЫЛОК

1. Дейт К.Дж. Введение в системы баз данных [Текст] : пер. с англ. / К.Дж. Дейт. – М. : Издательский дом «Вильямс», 2005. – 1328 с.
2. Ульман Дж. Основы реляционных баз данных [Текст] : пер. с англ. / Дж. Ульман. – К.: Лори, 2006. – 376 с.
3. Codd, E.F. (June 1970). A Relational Model of Data for Large Shared Data Banks. Communications of the ACM. 13 (6): 377–387.
4. Church, A. , Introduction to mathematical logic, Princeton: Princeton University Press. Classic textbook, 1956
5. Quine, W. V. O. , Philosophy of logic, second edition, Englewood Cliffs: Prentice-Hall, 1986

Приложение А

Фрагменты кода приложения

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using GunShop.Data;

using GunShop.Models;

using Microsoft.Extensions.Logging;

using GunShop.Models.Interfaces;

namespace GunShop.Services

{

public class CategorizationService

{

private readonly ApplicationDbContext \_context;

private readonly ILogger \_logger;

public CategorizationService(

ILoggerFactory loggerFactory,

ApplicationDbContext context)

{

\_context = context;

\_logger = loggerFactory.CreateLogger<ChartService>();

}

public void AddCommodityTypeToCategory(

CommodityType ct,

Category cat,

IEnumerable<CharacteristicValue> charvals)

{

var charsToFill = \_context

.Characteristics

.Where(c => c.CategoryId == cat.Id)

.ToArray();

foreach(var c in charsToFill)

{

var charval = charvals.FirstOrDefault(cv => cv.CharacteristicId == c.Id);

if (charval != null)

{

SetCharacteristicValue(c, ct, charval.Value);

}

else

{

throw new ArgumentException($"Characteristic {c.Name}({c.Id}) value not filled");

}

}

var newConnection = new CommodityTypeInCathegory

{

CategoryId = cat.Id,

CommodityTypeId = ct.Id

};

\_context.CommoditiesTypesInCathegories.Add(newConnection);

\_context.SaveChanges();

int? topCategoryId = cat.MasterCategoryId;

while(!(topCategoryId == null))

{

var upperCat = \_context.Categories.First(c => c.Id == topCategoryId);

\_context.CommoditiesTypesInCathegories.Add(new CommodityTypeInCathegory

{

CategoryId = upperCat.Id,

CommodityTypeId = ct.Id

});

topCategoryId = upperCat.MasterCategoryId;

}

\_context.SaveChanges();

}

public void RemoveCommodityTypeFromCategory(ICommodityType ct, ICategory cat)

{

var oldConnection =

\_context.CommoditiesTypesInCathegories.FirstOrDefault(

conn => conn.CommodityTypeId == ct.Id && conn.CategoryId == cat.Id);

if (oldConnection == null)

{

throw new ArgumentException($"Commodity {ct.Id} not in Category {cat.Id}");

}

var oldCharVals = \_context

.CharacteristicValues

.Where(cv =>

cv.CommodityTypeId == ct.Id

&& cv.CharacteristicId == cat.Id)

.ToArray();

\_context.CharacteristicValues.RemoveRange(oldCharVals);

\_context.CommoditiesTypesInCathegories.Remove(oldConnection);

\_context.SaveChanges();

}

public void AddCharacteristicToCategory(ICategory cat, Characteristic charact)

{

charact.CategoryId = cat.Id;

if(\_context.Characteristics.Count(ch=>ch.Id == charact.Id) > 0)

{

\_context.Characteristics.Update(charact);

}

else

{

\_context.Characteristics.Add(charact);

}

\_context.SaveChanges();

}

public void AddCategory(

ICategory cat,

IEnumerable<Characteristic> newCharacteristics)

{

var newcat = new Category

{

MasterCategoryId = cat.MasterCategoryId,

Name = cat.Name

};

\_context.Categories.Add(newcat);

\_context.SaveChanges();

var characteristics = newCharacteristics.ToList();

int? topCategoryId = cat.MasterCategoryId;

while (!(topCategoryId == null))

{

characteristics.AddRange(\_context.Characteristics

.Where(c => c.CategoryId == topCategoryId)

.ToList());

topCategoryId = \_context.Categories

.First(c => c.Id == topCategoryId)

.MasterCategoryId;

}

foreach (var c in characteristics)

{

AddCharacteristicToCategory(newcat, c);

}

}

public void AddCategory(Category cat)

{

AddCategory(cat, Enumerable.Empty<Characteristic>());

}

public void SetCharacteristicValue(

Characteristic charact,

CommodityType ct,

string value)

{

var oldVal = \_context.CharacteristicValues

.FirstOrDefault(cv =>

cv.CharacteristicId == charact.Id

&& cv.CommodityTypeId == ct.Id);

var possvals = PossibleValsFromString(charact.AvailableValues);

if (!possvals.Contains(value))

{

throw new ArgumentException($"Value {value} not allowed for " +

$"characteristic {charact.Name}({charact.Id})");

}

if (oldVal == null)

{

\_context.CharacteristicValues.Add(new CharacteristicValue

{

CharacteristicId = charact.Id,

CommodityTypeId = ct.Id,

Value = value

});

}

else

{

oldVal.Value = value;

\_context.CharacteristicValues.Update(oldVal);

}

\_context.SaveChanges();

}

public IEnumerable<Category> GetSubCategories(int categoryId)

{

return \_context

.Categories

.Where(x => x.MasterCategoryId == categoryId)

.ToArray();

}

public IEnumerable<int> CommoditiesTypesIdsInCategory(int categoryId)

{

return \_context

.CommoditiesTypesInCathegories

.Where(ctic => ctic.CategoryId == categoryId)

.Select(ctic => ctic.CommodityTypeId)

.ToArray();

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using GunShop.Data;

using GunShop.Services;

using GunShop.Services.Interfaces;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Logging;

namespace GunShop.Controllers

{

public class StatisticsController : Controller

{

ApplicationDbContext \_context;

ICommoditiesService \_commoditiesService;

ILogger \_logger;

public StatisticsController(

ApplicationDbContext context,

ILoggerFactory loggerFactory,

ICommoditiesService commoditiesService)

{

\_context = context;

\_commoditiesService = commoditiesService;

\_logger = loggerFactory.CreateLogger<StatisticsController>();

}

[HttpGet]

public IActionResult Index()

{

return View();

}

[HttpGet]

public IActionResult ChartSizes()

{

var result = \_context.CommoditiesInCharts

.ToArray()

.GroupBy(cic => cic.CustomerId)

.Select(gr => new ChartSize

{

CustomerId = gr.Key,

Count = gr.Count()

});

return Json(result);

}

[HttpGet]

public IActionResult StoragesFilled()

{

var result = \_context.Commodities

.ToArray()

.GroupBy(c => c.StorageId)

.Select(gr => new StorageFillment

{

StorageId = gr.Key,

Count = gr.Count()

});

return Json(result);

}

[HttpGet]

public IActionResult ManufacturersFilled()

{

var result = \_context.Commodities

.ToArray()

.Join(\_context.CommoditiesTypes,

comm => comm.CommodityTypeId,

ct => ct.Id,

(comm, ct) => ct.ManufacturerId

).GroupBy(mid => mid)

.Select(gr => new ManufacturerFillment

{

ManufacturerId = gr.Key,

Count = gr.Count()

});

return Json(result);

}

}

public struct ChartSize

{

public int CustomerId { get; set; }

public int Count { get; set; }

}

public struct StorageFillment

{

public int StorageId { get; set; }

public int Count { get; set; }

}

public struct ManufacturerFillment

{

public int ManufacturerId { get; set; }

public int Count { get; set; }

}

}