МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Высшая школа экономики и управления
Кафедра «Информационные технологии в экономике»

ДОПУСТИТЬ К ЗАЩИТЕ
Зав. кафедрой, д.т.н., с.н.с.
/ Б.М. Суховилов /
«_____»_____20____ г.

Создание системы для автоматизации учёта запчастей на авторазборе

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 09.03.03.2021.023.ВКР

Руководитель, ст. преподаватель
_____/С.Ю. Нестеренко/
«_____» _____2021 г.

Автор,
студент группы ЭУ-402
_____/У.А. Алексеева/
«_____» _____2021 г.

Нормоконтролер, доцент
_____/Е.А. Конова/
«_____» _____2021 г.

Челябинск 2021

АННОТАЦИЯ

Алексеева У.А. Создание системы для автоматизации учета запчастей на авторазборе – Челябинск: ЮУрГУ, ЭУ-402, 2021. – 88 с., 39 рис., 30 табл., библиогр. список – 12 наим., 9 прил.

Выпускная квалификационная работа посвящена созданию системы для автоматизации учета запчастей на авторазборе.

Проведен анализ существующих наиболее похожих на разрабатываемую систему учета приложений.

Описан процесс разработки системы учета. Приведены требования, выдвигаемые к ней. Описана структура, методы и технологии, использованные при создании системы учета.

Приведено экономическое обоснование разработки системы учета. Представлены расчёты затрат на разработку.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

При создании малого бизнеса учет материальных ценностей обычно ведется в таблицах Excel, либо на бумажном носителе. Пока фирма небольшая, такой учет оправдан, так как позволяет минимизировать затраты на программные продукты. Но с появлением у фирмы все большего количества наименований товаров, данный вид учета становится неудобен и требует огромного количества времени на поиск необходимой информации о товаре. Также при ведении такого учета неизбежно возникают различные ошибки, зависящие от человеческого фактора.

Автоматизация системы учета позволяет детализировать учет складских запасов, определять оборот продукции по различным критериям, проводить анализы продаж, а также, что не менее важно – позволяет избегать множества ошибок, связанных с ручным ведением учета и сокращать количество потраченного на учет времени. Еще одна причина для ведения автоматизированного учета – это обеспечение мобильности при переездах и налаживании работы на новом месте.

Целью выпускной квалифиационной работы является создание системы для автоматизации учета на авторазборе.

1 ПОСТАНОВКА ЗАДАЧИ

1.1 Технические требования к системе

Авторазбор – это процесс разбора машины на б/у комплетующие, которые в дальнейшем будут использованы для ремонта другой машины.

Авторазборка – фирма, осуществляющая авторазбор.

Требуется разработать автоматизированную систему учета автозапчастей для фирмы, осуществляющей авторазбор. Система должна позволять добавление, изменение, удаление данных, их сортировку и фильтрацию.

Основные требования к программе:

- добавление, изменение и удаление данных о запчастях;
- хранение информации о запчастях в таблицах базы данных;
- веб-приложение для работы в локальной и глобальной сетях;
- приложение должно иметь удобный и интуитивно-понятный графический интерфейс;
- возможность сортировки и фильтрации данных, имеющихся в базе;
- отслеживание изменений цен автозапчастей;
- возможность добавления пользователей и групп пользователей;
- разграничение прав доступа пользователей;
- отслеживание продаж автозапчастей.

Дополнительные требования к программе:

- разработка телеграм-бота для добавления запчастей.

1.1.1 Требования к подходу разработки

Для разработки веб-приложения выбран подход MPA или Multi Page Application – это многостраничные приложения, которые работают по традиционной схеме. То есть при любом изменении данных или получении новой информации, страница обновляется. Вес приложений такого типа больше, чем вес одностраничных приложения, но, так как система учета должна отображать большое количество данных, использование такого подхода целесообразно. Также для разработки необходимо использовать AJAX, который позволяет обновлять

лишь часть страницы, а не всю страницу целиком, что улучшает производительность веб-приложения.

Преимущества данного подхода:

- простая SEO оптимизация – можно оптимизировать каждую страницу веб-приложения под нужные ключевые запросы;
- привычность для пользователей – достигается за счет простого интерфейса и классической навигации;
- улучшенная производительность за счет использования технологии AJAX.

1.1.2 Требования к интерфейсу

Пользовательский интерфейс должен быть максимально прост для пользователя. Дизайн должен быть минималистичным и не отвлекать внимание отдельными элементами, это необходимо для улучшения читабельности и восприятия текстовой информации. Для того, чтобы не возникали проблемы со здоровьем (головная боль, «усталость» глаз) после длительной работы в приложении, стоит использовать неяркие, сочетающиеся между собой, цвета.

1.1.3 Требования к хранению данных

Данные приложения необходимо хранить в реляционной базе данных. База данных должна быть нормализована, то есть находиться как минимум в третьей номальной форме (3NF), во избежание избыточности информации и появления аномалий. Управление данными должно быть удобным и гибким.

1.1.4 Требования к программному обеспечению

Требования к программному обеспечению клиентской части:

- браузеры: Microsoft Internet Explorer 7.0 и выше, Microsoft Edge 1.0 и выше, Mozilla Firefox 3.5 и выше, Opera 9.5 и выше, Apple Safari 3.2.1 и выше, Google Chrome 2.0 и выше, Яндекс.Браузер 1.0 и выше;
- поддержка JavaScript;
- веб-сервер Apache;

Дополнительные требования к программному обеспечению клиентской части (для использования бота):

- установленное приложение Telegram.

1.1.5 Требования к аппаратному обеспечению

Требования к аппаратному обеспечению серверной части:

- Не менее 500 МБ свободного места на диске.

Со стороны клиента требуется поддержка программного обеспечения из пункта 1.5.

1.2 Анализ существующих автоматизированных систем учета запчастей

Складские запасы – это запасы товаров, которые имеются в наличии у компании для дальнейшей реализации. Как правило, такие запасы – это одновременно и вложенный капитал, и своеобразный риск.

Необходимость управления складскими запасами становится ясной при рассмотрении оборотного капитала компании, объема продаж и складских запасов.

В зависимости от объема продаж можно применить четыре основных вида складского учета.

1. Ручной учет товаров: подходит для предпринимателей и организаций с узким ассортиментом и малым количеством сделок за единицу времени.

2. Учет товаров в текстовых редакторах или таблицах Excel: в такие программы заносится больше информации, но, как и в первом виде учета, вручную, также Excel способен выполнять некоторые математические расчеты в автоматическом режиме.

3. Автоматизированный учет товаров на складе: этот вариант становится возможным при использовании специальных программ – товароучетных систем, в этом варианте все происходит в автоматическом режиме – нужно только создать первичную базу и делать отметки о движении.

4. Комплексная автоматизация учета: под автоматизацию попадают не только склады, но и торговые площади, а также витрины магазинов, закупки,

продажи и другие процессы организации, такой вид учета тоже реализуется с помощью приложений для автоматизации учета.

Ручной учет и контроль складов в Excel-таблицах постепенно отходят в прошлое даже для организаций с небольшим ассортиментом, так как эти виды учета имеют множество существенных недостатков. Перечислим основные из них:

- влияние человеческого фактора: работник забыл сделать отметку о продаже, неправильно принял товар – учет ведется с ошибками;
- абсолютная непрозрачность склада: полной информацией о складских остатках владеют только кладовщики, все остальные работники, включая аппарат управления, имеют лишь приблизительное представление о том, что происходит на складе;
- отсутствие автоматизации: любое действие с товаром необходимо записывать в тетрадь или таблицу, что времязатратно, неудобно и тоже приводит к ошибкам учета;
- большие временные затраты: то, что система учета считает за доли секунды, приходится делать человеку;
- необходимость в дополнительной рабочей силе: наступает как следствие всех предыдущих пунктов, так как недостает работников, которые должны все записывать, считать и контролировать;
- появление незаменимых сотрудников: происходят такие ситуации, что работник, занимающийся учетом, уходит в отпуск – и никто не знает, где расположен нужный товар.

Автоматизация учета – это программное обеспечение, а также процесс его внедрения в деятельность компании.

Автоматизация работы склада проводится организациями для упрощения контроля за движением товаров. За счет автоматизации можно быстро определить количество каждого товара, вычислить остатки, составить план заказа продукции. Автоматизация особенно нужна, если увеличивается ассортимент товаров и объемы поставок [1].

Далее рассмотрены некоторые программы для автоматизации учета запчастей.

1.2.1 Автодонор

Программа «Автодонор» является конфигурацией универсальной учетной системы с облачным хранением данных OUTOFBOX.RU. Предназначена для автоматизации учёта в магазинах запчастей и на авторазборках. Подходит как для малого бизнеса, так и для компаний с филиалами и множеством складов. Конфигурация настраивается под конкретные требования автобизнеса, например, индивидуальный учёт деталей с маркировкой, связка запчастей с автомобилем-донором, многовариантный поиск с учетом применимости, отдельный реестр автодоноров с расчетом прибыли с каждой машины. Поддерживает 54-ФЗ и имеет возможность подключения внешнего оборудования. Пример работы программы приведен на рисунке 1.

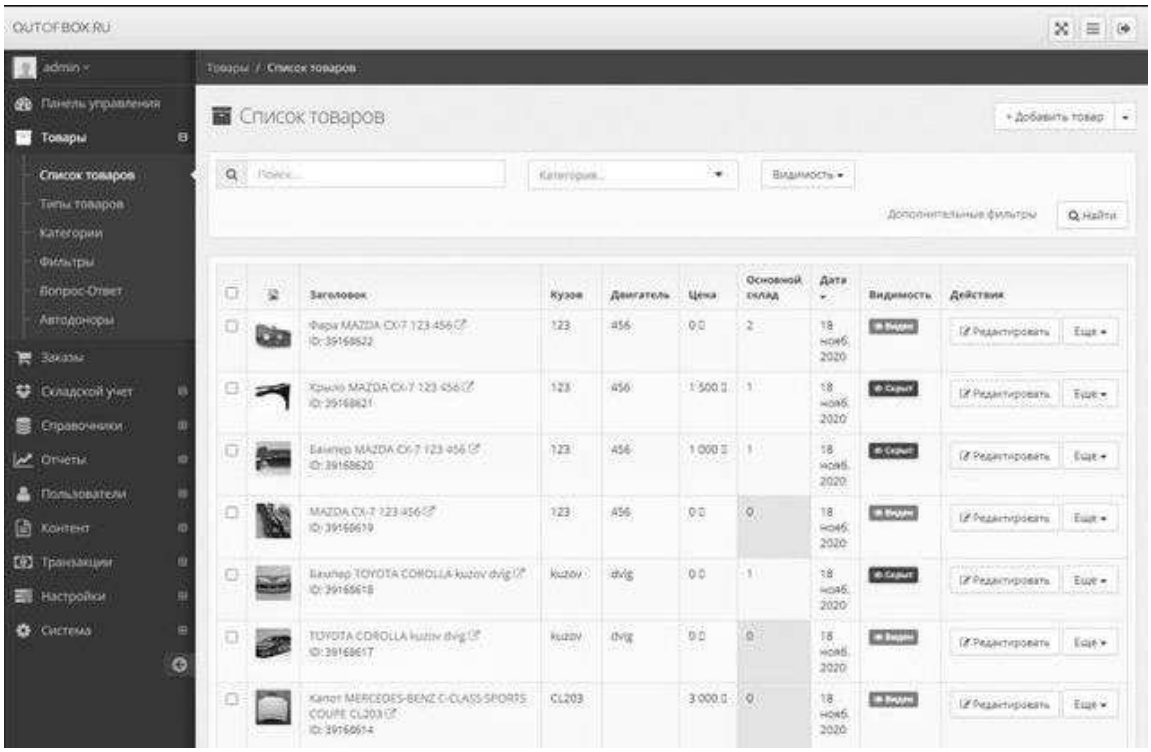

Рисунок 1 – Интерфейс программы «Автодонор»

Достоинством этой программы является понятный интерфейс, а недостатком – множество лишних функций.

1.2.2 Bazon

Bazon – специализированная учетная система для продавцов б/у автозапчастей. Подходит небольшим авторазборам и крупным компаниям с филиалами. Разработчики предлагают следующие функции: автоматические выгрузки на рекламные-площадки, занесение товара с помощью мобильного приложения, учет поставок, продаж и остатков, определение места хранения на складе и различную аналитику. Чем больше автозапчастей хранится в базе данных, тем выше стоимость тарифа, также стоимость зависит от выбранных функций программы. Интерфейс интернет-магазина на данной платформе приведен на рисунке 2.

Рисунок 2 – Интерфейс интернет-магазина на платформе «Bazon»

У этой программы широкий функционал, часть которого избыточна, и высокая стоимость, что не подходит заказчику.

1.2.3 Учет автозапчастей

Программа «Учет автозапчастей» предназначена для ведения учета автозапчастей в автомагазине, автосервисе, СТО или на авторазборке. Эта программа предлагает следующие функции: возможность работы на одном ПК, в

локальной сети или удаленно через интернет; печать прайсов и ценников, работа со сканером штрихкодов и принтером кассовых чеков; поддержка ККТ «Штрих-М».

Интерфейс программы приведен на рисунке 3.

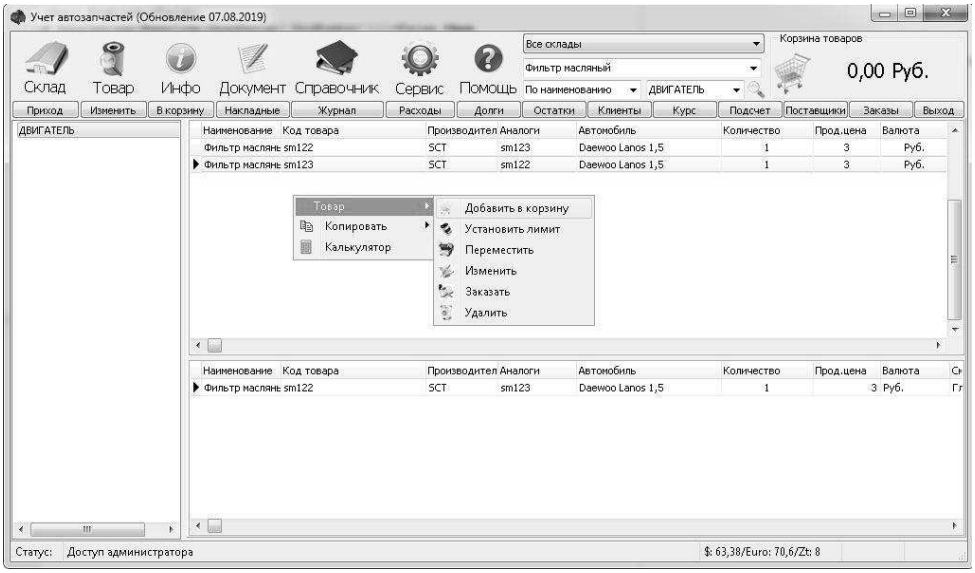

Рисунок 3 – Интерфейс программы «Учет автозапчастей»

Недостатком программы является сложный перегруженный интерфейс.

1.2.4 Склад Автозапчастей

Система «Склад Автозапчастей» как программа складского учета предоставляет ряд возможностей, направленных на облегчение процесса формирования базы данных поступающих на хранение товаров. Поддерживаемая сетевая интеграция в совокупности с возможностью экспорта данных делают программу «Склад Автозапчастей» гибким и понятным инструментом, позволяющим оптимизировать работу, и в разы уменьшить затраты времени на решение организационных вопросов.

Интерфейс программы показан на рисунке 4.

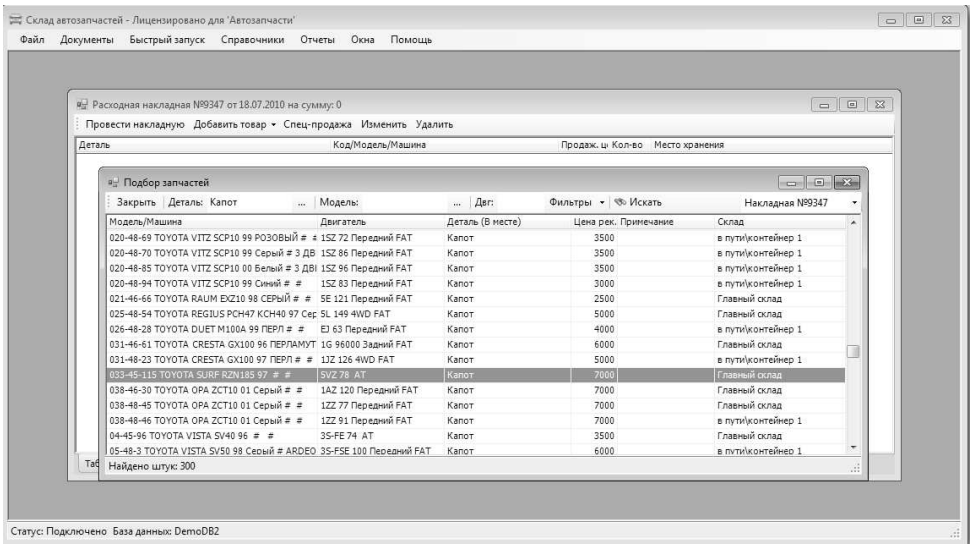

Рисунок 4 – Интерфейс программы «Склад Автозапчастей»

1.2.5 Итоговое сравнение

Сравнение ведется по 7 основным требованиям, предъявленным заказчиком.

1. Веб-приложение лишь для внутреннего использования, это не должен быть интернет-магазин.
2. Простой ненагруженный интерфейс, следовательно, простота в использовании.
3. Возможность разраничения прав пользователей.
4. Отсутствие количественных ограничений в добавлении запчастей в базу данных.
5. Хранение истории цен и истории продаж.
6. Веб-приложение, не десктопное, так как планируется удаленный доступ.
7. Низкая стоимость приложения.

Итоговое сравнение показано в таблице 1.

Таблица 1 – Итоговое сравнение существующих систем учета запчастей

Требование/Приложение	Автодонор	Bazon	Учет автозапчастей	Склад Автозапчастей
веб-приложение для внутреннего использования, не интернет-магазин	+	+	+	+
простой ненагруженный интерфейс	+	+	-	-
разраничение прав пользователей	+	+	+	-
отсутствие количественных ограничений в добавлении запчастей в базу данных	+	+	-	+
хранение истории цен и истории продаж	+	-	-	-
веб-приложение	+	+	+	-
низкая стоимость	-	-	-	+

Ни одна из вышеперечисленных программ не соответствует всем требованиям заказчика, следовательно, необходимо разработать новую систему учета.

Вывод по первому разделу

Поставлены требования к приложению, а также к аппаратному и программному обеспечению. Проведено сравнение существующих систем учета автозапчастей. Принято решение разработки, потому что рассмотренные системы не удовлетворяют требованиям заказчика.

2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА

2.1 Построение DFD-диаграммы приложения

DFD – общепринятое сокращение от англ. data flow diagrams – диаграммы потоков данных. Диаграмма DFD наглядно отображает течение информации в пределах процесса или системы. Для изображения входных и выходных данных, точек хранения информации и путей ее передвижения между источниками и пунктами доставки в таких диаграммах применяются стандартные фигуры, такие как прямоугольники и круги, а также стрелки и краткие текстовые метки. Диаграммы DFD применяются для анализа существующих и моделирования новых систем [2].

2.1.1 Контекстная диаграмма

Контекстная диаграмма веб-приложения автоматизированной системы учета на предприятии «Авангард» представлена на рисунке 5.

Рисунок 5 – Контекстная диаграмма

В таблице 2 представлены потоки данных модели, для каждой из которой есть соответствующие нотации и описание.

Таблица 2 – Потоки данных DFD-модели

Название	Описание	Нотация
Данные о запчастях	Передаются данные о запчастях	Запчасть: код запчасти: счетчик; наименование: строка; цена: число; автомобиль донор: число; год выпуска: строка; производитель: строка; номер производителя: строка; маркировка: строка; состояние: число; расположение: число; цвет: строка; дата: дата; склад: число; комментарий: строка; фото: изображение. Шина: код шины: счетчик; наименование: строка; цена: число; посадочный диаметр: число; сезон: число; производитель: число; модель: строка; ширина: число; высота: число; износ: строка; центральное отверстие: число; дата: дата; склад: число; комментарий: строка; Количество: число; фото: изображение. Диск: код диска: счетчик; наименование: строка; цена: число; посадочный диаметр: число; тип: число; ширина: число; сверловка: число; вылет: строка; центральное отверстие: число; дата: дата; склад: число; комментарий: строка; количество: число; фото: изображение. Комплект: код комплекта: число; номер: строка; название: строка; цена: число; посадочный диаметр: число; тип диска: число; ширина диска: число; сверловка: число; вылет: строка; центральное отверстие: число; сезон: число; производитель шины: число; модель шины: строка; ширина шины: число; высота шины: число; износ: строка; дата: дата; склад: число; комментарий: строка; количество: число; фото: изображение.

Продолжение таблицы 2

Название	Описание	Нотация
Данные учетных записей	Передаются данные учетных записей	Логин: строка; пароль: строка; электронная почта: строка.
Данные о запчастях	Передаются данные о запчастях	История цен запчасти: код истории цен запчасти: счетчик; запчасть: число; цена: число; дата: дата. История цен шины: код истории цен шины: счетчик; шина: число; цена: число; дата: дата. История цен диска: код истории цен диска: счетчик; диск: число; цена: число; дата: дата. История цен комплекта: код истории цен комплекта: счетчик; комплект: число; цена: число; дата: дата. История продаж запчасти: код истории продаж запчасти: счетчик; запчасть: число; цена: число; количество: число; дата: дата. История продаж шины: код истории продаж шины: счетчик; шина: число; цена: число; количество: число; дата: дата. История продаж диска: код истории продаж диска: счетчик; диск: число; цена: число; количество: число; дата: дата. История продаж комплекта: код истории продаж комплекта: счетчик; комплект: число; цена: число; количество: число; дата: дата.

В контекстной диаграмме имеется только один функциональный блок – «Система учета на предприятии «Авангард»». Он содержит информацию о процессах в данной информационной системе:

- изменение настроек администратора;
- изменение настроек пользователя;
- работа с запчастями;
- работа с запчастями через телеграм-бота.

В контекстной диаграмме приведены следующие внешние сущности:

- пользователь;
- администратор;
- телеграм-бот.

2.1.2 Диаграмма декомпозиции

Диаграмма декомпозиции веб-приложения автоматизированной системы учета на предприятии «Авангард» показана на рисунке 6.

Рисунок 6 – Диаграмма декомпозиции

В таблице 3 приведены описания и нотации функциональных блоков диаграммы декомпозиции.

Таблица 3 – Функциональные блоки диаграммы декомпозиции DFD-модели

Название	Описание	Нотация
Форма отображения и редактирования данных о запчастях	Получение и вывод на экран данных о запчастях администратору приложения.	См. нотацию данных о запчастях приложения А
Форма отображения и редактирования данных учетных записей	Получение и вывод на экран данных учетных записей администратору приложения.	См. нотацию данных учетных записей приложения А.
Форма отображения данных о запчастях	Получение и вывод на экран данных о запчастях пользователю приложения.	См. нотацию данных о запчастях приложения А.

Хранилище данных – это абстрактное устройство для хранения информации, которую можно в любой момент поместить в накопитель и через некоторое время извлечь.

2.2 Выбор способа разработки

2.2.1 Проектирование веб-приложений с помощью конструктора

Конструкторы сайтов – это системы, позволяющие из готового типового набора модулей и компонентов сформировать приложение и разместить его в сети Интернет. Одни из наиболее популярных конструкторов веб-приложений – это системы ucoz.ru, narod.ru, sites.google.com [3].

Преимущества:

- не требуют знаний в сфере разработки приложений, большинство шаблонов страниц имеют функции и элементы инферфейса, которые упрощают настройку веб-приложения;
- большинство таких конструкторов бесплатны или имеют низкую стоимость.

Недостатки:

- несмотря на то, что такие конструкторы просты в использовании и довольно недороги, у них есть свои ограничения: самое важное ограничение состоит в том, что создатель приложения не имеет доступа к коду этого приложения, то есть функции ограничиваются теми, что есть в конструкторе;
- ограничения в выборе доменного имени, количестве страниц и хранимых данных, для снятия каких-либо ограничений необходимо платить.

2.2.2 Использование системы управления контентом

CMS (англ. Content Management System) – это система управления контентом сайта. По данным агентства W3Techs на CMS разработано больше половины сайтов в интернете.

На практике CMS – это веб-приложение, в котором создают и обслуживают сайты. Основное преимущество CMS состоит в том, что для создания сайта и дальнейшей работы с ним не требуется знание языков программирования.

Что касается дизайна, есть возможность установить на сайт готовый шаблон и настроить отдельные его элементы прямо в панели управления CMS: шрифты, цвета, изображения, логотип, кнопки, ссылки. Возможности по настройке отличаются в зависимости от шаблона.

Существуют CMS с открытым кодом. Открытый код означает, что систему может изменять любой желающий. Благодаря этому в подобных CMS регулярно появляются новые шаблоны и дополнения, а также работа с уязвимостями системы происходит быстрее. Популярные CMS с открытым кодом: WordPress, OpenCart, Joomla!, Drupal, Magento, PrestaShop.

Преимущества:

- простота установки, настройки, создания сайта и шаблонов;
- наличие готовых модулей гостевой, почтовой формы, формы поиска и так далее;
- отсутствие затрат на квалифицированный персонал;
- некоторый выигрыш по времени для обновления материалов.

Недостатки:

- увеличение времени загрузки сайта;
- CMS создает код сайта из модулей, на создание кода уходит время, и, чем сложнее сайт, тем больше временных затрат требуется.
- поисковые системы плохо видят код, который сгенериловала CMS;
- сложности при изменении структуры сайта.
- уязвимость, найденная в любом модуле CMS на одном сайте, вызывает всплеск атак на сайты, разработанные в той же системе. А, учитывая, что у бесплатных CMS код открыт для изучения, назвать такой сайт хорошо защищенным нельзя.

2.2.3 Создание веб-приложения с нуля

Последний вариант создания веб-приложения – это разработка сайта с помощью написания собственного кода. Этот вариант позволяет создать уникальное приложение с любыми необходимыми функциями и дизайном. Для такого подхода

необходимо знать различные языки программирования, понимать процессы, происходящие в приложении, а также этапы разработки.

Преимущества:

- такой подход безопаснее: так как сайт уникален, код не является достоянием общественности;
- на выходе получается новый уникальный продукт;
- разработчик не ограничен шаблонами и готовыми функциями;
- самостоятельно написанное веб-приложение проще продвигать, так как поисковые системы хорошо распознают код.

Недостатки:

- требует знаний в разработке программных продуктов;
- если создавать самостоятельно, то процесс требует много времени и трудозатрат на изучение языков и основ создания приложений;
- если нанимать разработчика, то в разработку необходимо вложить немалую сумму денег, так как труд разработчиков оценивается дорого.

Согласно требованиям заказчика, необходимо разработать новый продукт, следовательно, первые два варианта разработки веб-приложений в данном случае не подходят. Сделан выбор в пользу самостоятельной разработки веб-приложения.

2.3 Выбор системы управления базами данных

Исходя из потребностей заказчика, а также для удобства и простоты разработки, для автоматизированной системы учета автозапчастей выбрана СУБД MySQL.

MySQL, считается одной из самых распространенных СУБД, это реляционная система управления базами данных, имеющая открытый исходный код, и главными достоинствами которой являются ее скорость и гибкость, достигаемая за счет поддержки большого количества различных типов таблиц.

Более того, эта система управления базами данных предоставляется бесплатно, она надежна, а также имеет простой интерфейс и средства синхронизации

с другими базами данных. Все эти факторы вместе позволяют использовать MySQL как крупным корпорациям, так и маленьким фирмам.

2.4 Выбор фреймворка для веб-приложения

С точки зрения бизнеса разработка приложения с использованием фреймворков практически всегда экономически выгоднее и качественнее, чем написание приложения без использования платформ. Разработка приложения без использования платформ может являться верным решением исключительно в двух случаях – если проект совсем простой и не требует дальнейшего развития, или он очень нагруженный и требует низкоуровневой оптимизации. В остальных случаях разработка с использованием программной платформы качественнее и быстрее.

Одним из главных плюсов в использовании фреймворков является то, что фреймворк определяет стандартизированную структуру для построенных на его базе приложений. Поэтому приложения на фреймворках гораздо проще сопровождать и дорабатывать, так как структура организации компонентов ясна любым разработчикам, использующих эту платформу, и не требуется долго разбираться в архитектуре, чтобы понять принцип работы приложения или найти место реализации того или иного функционала. Большинство фреймворков для разработки веб-приложений использует парадигму MVC (модель-представление-контроллер) – то есть многие фреймворки используют идентичный подход к организации компонентов приложения, и это делает понимание архитектуры приложения еще проще.

Проектирование архитектуры программного обеспечения при разработке с использованием фреймворка сильно упрощается – в методологиях фреймворков обычно заложены лучшие практики программной инженерии, и следуя этим правилам можно избежать многих проблем и ошибок в проектировании. По сути, фреймворк – это множество конкретных и абстрактных классов, связанных между собой и упорядоченных согласно методологии фреймворка. Конкретные классы обычно реализуют взаимные отношения между классами, а абстрактные классы представляют собой точки расширения, в которых заложенный во фреймворк базовый функционал может быть использован «как есть» или адаптирован под задачи

конкретного приложения. Для обеспечения расширения возможностей в большинстве фреймворков используются техники объектно-ориентированного программирования: например, части приложения могут наследоваться от базовых классов фреймворка или отдельные модули могут быть подключены как примеси.

Экосистемы веб-фреймворков также богаты на готовые реализации многих функциональных возможностей. Разработчикам при работе над типовыми задачами не надо придумывать что-то новое, так как они могут воспользоваться уже созданной сообществом реализацией, что не только сокращает затраты времени и денег, но и позволяет добиться более высокой стабильности решения – компонент, который используется и дорабатывается тысячами других разработчиков обычно более качественно реализован и лучше протестирован на всевозможных сценариях, чем решение, которое может в адекватные сроки разработать один разработчик или даже небольшая команда [4].

Для разработки приложения выбран фреймворк Django.

Django – это высокоуровневый Python веб-фреймворк, позволяющий в кратчайшие сроки создавать безопасные и поддерживаемые веб-сайты. Написанный опытными разработчиками, Django решает большую часть проблем, возникающих во время веб-разработки. Он распространяется бесплатно и имеет открытый исходный код, а также растущее и активное сообщество, подробную документацию и множество вариантов как бесплатной, так и платной поддержки.

Преимущества:

- Django предоставляет почти всё, что разработчики могут захотеть реализовать; поскольку всё, что нужно, является частью единого программного продукта, всё это совместимо, соответствует последовательным принципам проектирования и имеет обширную и актуальную документацию;
- Django используется при создания практически любого типа веб-сайтов – от систем управления контентом и wiki до социальных сетей и новостных сайтов; этот фреймворк может работать с любой клиентской средой и способен доставлять

контент практически в любом формате (включая HTML, RSS-каналы, JSON, XML и т. д.);

- внутренне Django при необходимости может быть расширен сторонними компонентами;
- Django позволяет разработчикам избежать многих распространённых ошибок безопасности; одним из способов защиты информации является безопасный способ управления учётными записями пользователей и их паролями. он заключается в том, что файлы cookie содержат только ключ, а фактические данные хранятся в базе данных; также в базе данных хранится лишь хэш пароля, а не сам пароль; Django обеспечивает защиту от многих уязвимостей, таких как SQL-инъекции, межсайтовый скриптинг, подделка межсайтовых запросов и кликджекинг;
- Django написан на языке программирования Python, который поддерживается на многих платформах, следовательно, можно запускать приложения на разных версиях Linux, Windows и Mac OS, более того, Django поддерживается многими веб-хостингами, которые часто предоставляют определённую инфраструктуру и документацию для размещения сайтов Django [5-7].

2.5 Выбор языка программирования для разработки

Поскольку Django создан для разработки приложений на языке программирования Python, то его и стоит выбрать для разработки. Для разработки бота также выбран именно этот язык программирования.

2.5.1 Python

Python – это активно развивающийся скриптовый язык, который используют для решения большого объема самых разноплановых проблем и задач. Python применяется при разработке компьютерных и мобильных приложений, его используют при работе с большим объемом данных, при разработке веб-приложений и других проектов, также его используют в машинном обучении. Этот язык программирования используют известные корпорации, такие как Spotify и Амазон

(например, для анализа данных и создания алгоритма рекомендаций), YouTube, Инстаграм и Walt Disney [8].

Преимущества:

- благодаря логичному синтаксису код легко читается и воспринимается;
- большое интернет-сообщество и подробная актуальная документация;
- гибкость и масштабируемость: Python позволяет разработчикам адаптировать высокоуровневую логику приложения, что позволяет при необходимости легко расширять сложные приложения;
- Python является интерпретируемым языком программирования. Это значит, что до запуска он представляет собой обычный текстовый файл., соответственно, программировать можно почти на всех платформах.

Недостатки:

- Python не самый быстрый среди языков программирования, скорость выполнения программ может быть чуть ниже;
- не самый удобный язык для мобильных разработок;
- из-за гибкости типов данных потребление памяти Python не минимальное.

2.5.2 XHTML

XHTML – это основанный на XML язык разметки гипертекста, максимально приближенный к текущим стандартам HTML. XHTML отличается от HTML строгостью написания кода. Строгие требования к оформлению XHTML-кода позволяют избежать многих ошибок ещё на стадии написания и отладки [9].

HTML (HyperText Markup Language) переводится, как язык разметки гипертекста. HTML используется для верстки веб-страниц. То есть при переходе на веб-приложение браузер скачивает с сервера HTML-документ, в котором содержится код с командами (тегами), которые браузер выполняет, и пользователи видят привычные для них таблицы, ссылки, абзацы, списки и другое [10].

2.5.3 CSS

Cascading Style Sheets (CSS) – это язык иерархических правил (каскадных таблиц стилей), его применяют для представления внешнего вида документа, написанного на HTML или XML (включая различные языки XML, такие как SVG и XHTML). CSS описывает, каким образом элемент должен отображаться на экране.

CSS является одним из основных языков свободной веб-разработки, который стандартизован спецификацией W3C. Стандарт CSS делится на уровни: CSS1 в настоящее время устарел, CSS2.1 – рекомендован для применения, а CSS3, разбитый на более мелкие модули, развивается на пути стандартизации [11].

2.5.4 JavaScript

JavaScript – это полноценный динамический язык программирования, который применяется к HTML документу, и может обеспечить динамическую интерактивность на веб-сайтах [12].

2.5.5 AJAX

AJAX Asynchronous JavaScript & XML – специальная технология взаимодействия с сервером, которая не требует выполнения перезагрузки. Она позволяет повысить скорость загрузки страниц веб-ресурса, поскольку нет необходимости обновлять их каждый раз. Наличие этой опции помогает сделать пользование сайтом для пользователя максимально комфортным [13-14].

Технология работы AJAX включает четыре этапа.

1. Пользователь обращается к AJAX, чаще всего это происходит с помощью нажатия кнопки;

2. Сервис пересылает запрос на сервер вместе с сопутствующими данными, например, может понадобиться загрузка файла или определенных сведений из базы данных.

3. Получив ответ из базы данных, сервер направляет его в браузер.

4. JavaScript получает ответ, расшифровывает и показывает пользователю.

Вывод по второму разделу

Составлены контекстная диаграмма и диаграмма декомпозиции DFD-модели, выполнено сравнение различных средств разработки и языков программирования, обоснован выбор средств разработки и языков программирования.

3 РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА

3.1 Проектирование базы данных

Схема данных разделена на четыре части, так как она слишком объемная, и поля таблиц иначе не видно.

Первая часть схемы данных показана на рисунке 7.

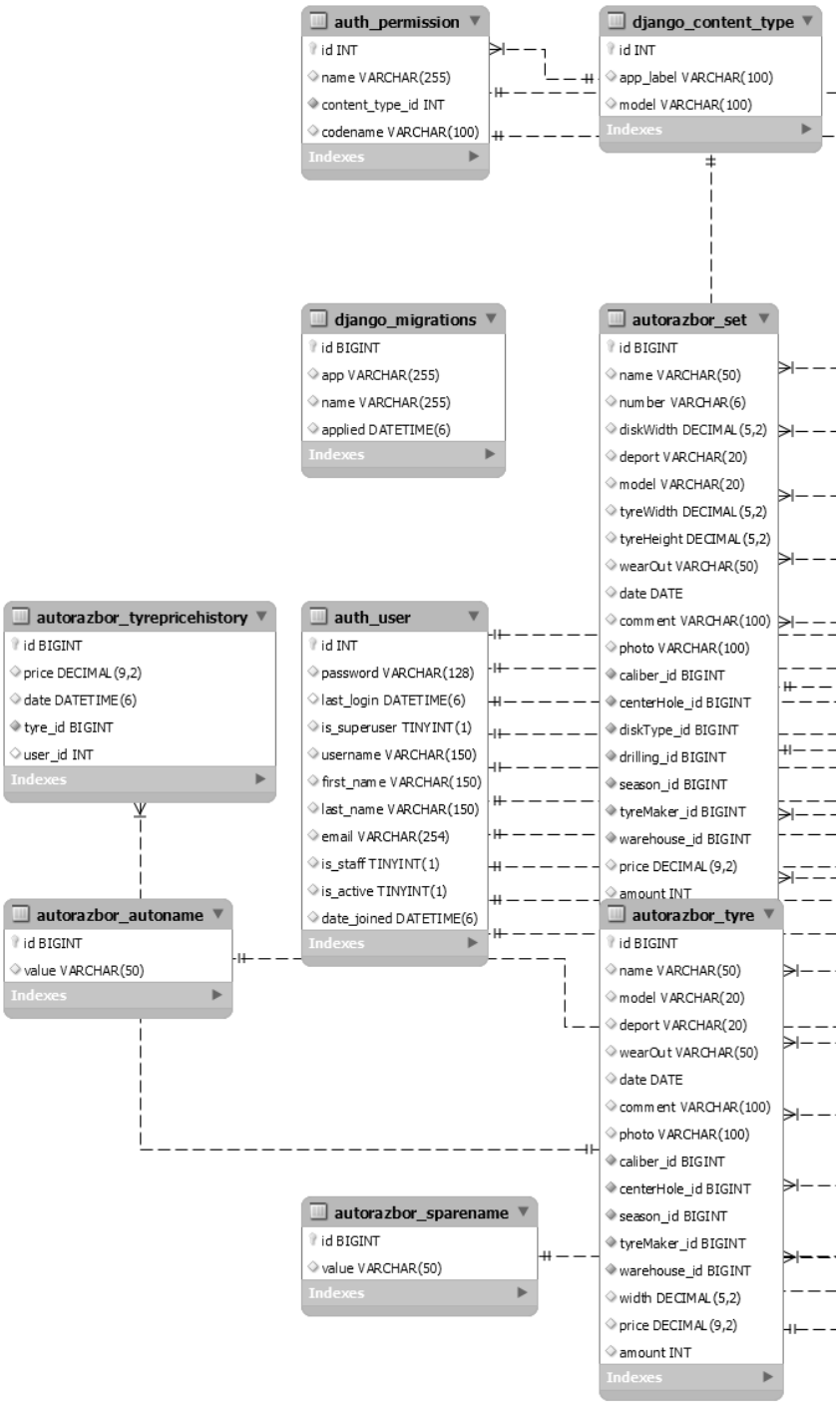

Рисунок 7 – Схема данных

Вторая часть схемы данных показана на рисунке 8.

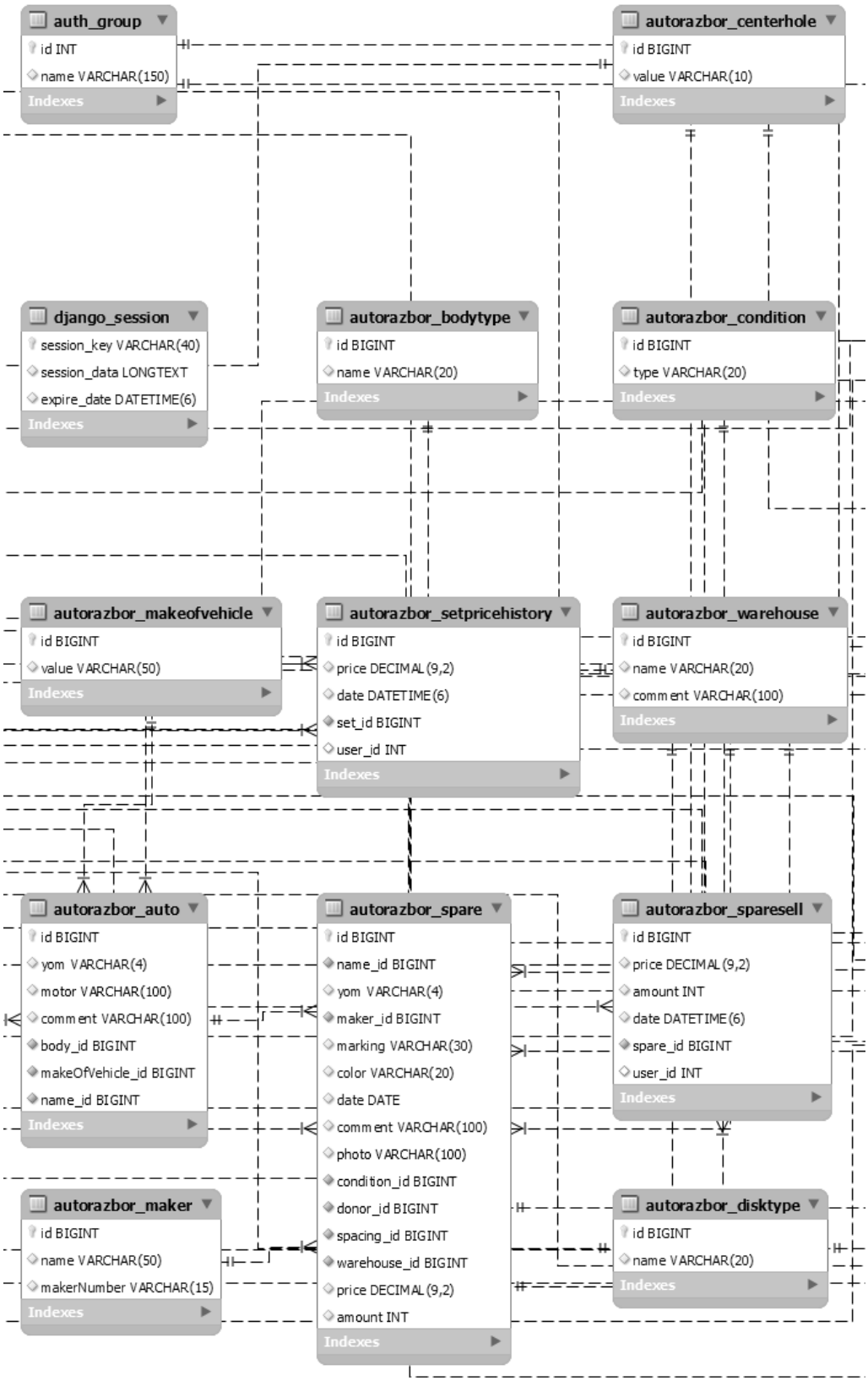

Рисунок 8 – Схема данных

Третья часть схемы данных показана на рисунке 9.

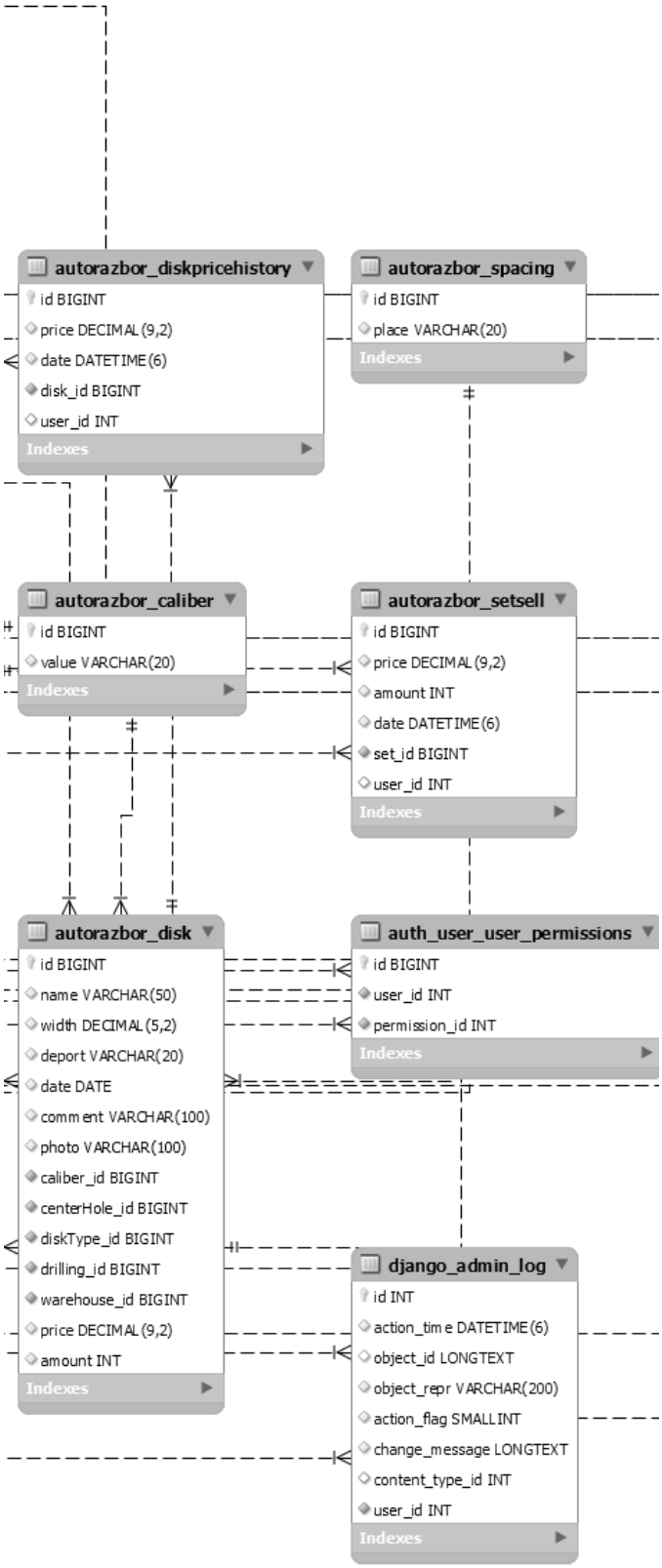

Рисунок 9 – Схема данных

Четвертая часть схемы данных показана на рисунке 10.

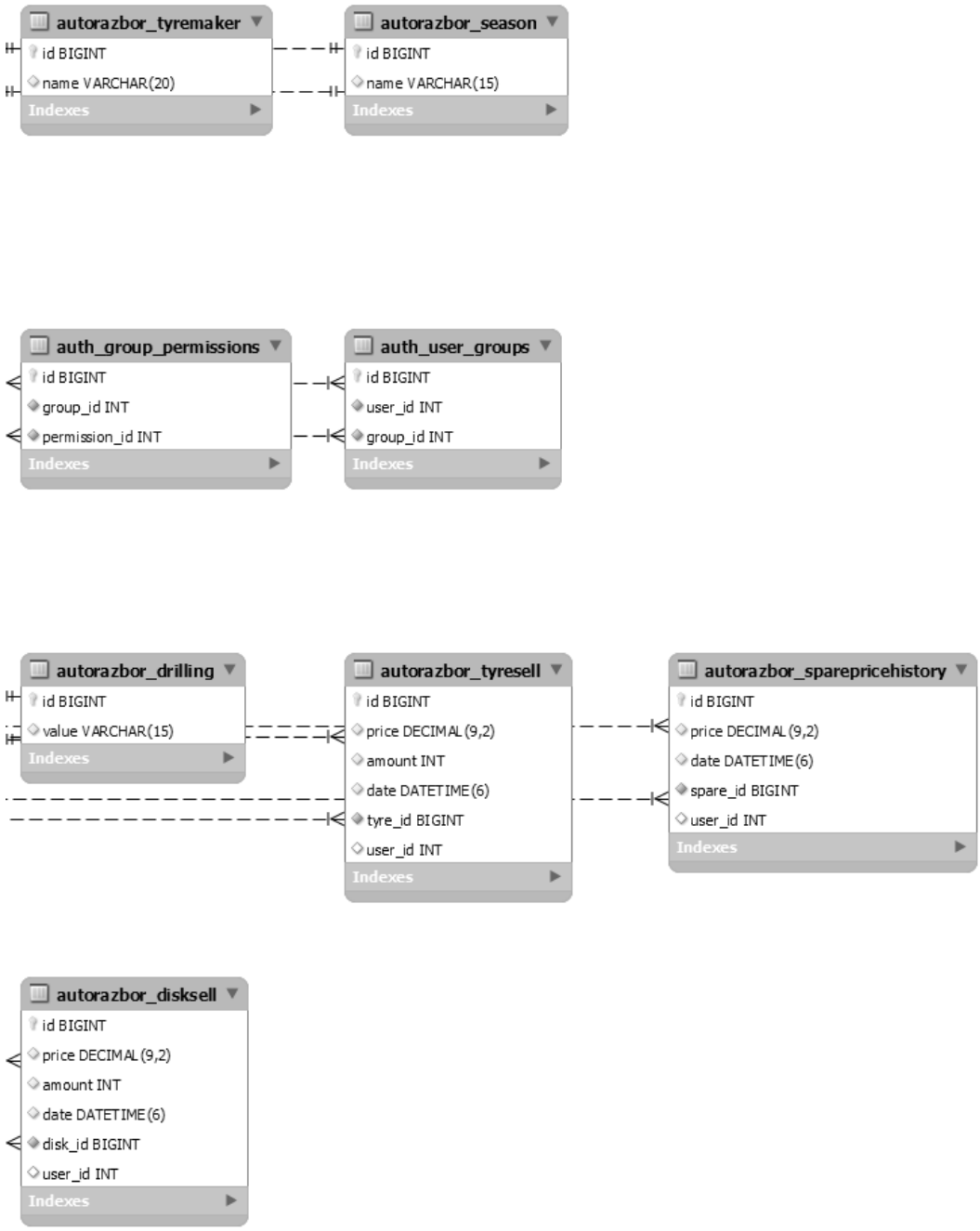

Рисунок 10 – Схема данных

На схеме данных показано множество взаимосвязанных таблиц. Каждая таблица отвечает за хранение определенной информации, необходимой для корректного отображения данных. Все таблицы приведены к третьей нормальной форме. Некоторые таблицы на схеме созданы фреймворком автоматически во время первой миграции.

3.2 Описание таблиц базы данных

Далее приведено описание самостоятельно-созданных таблиц базы данных.

В таблице 4 представлено описание таблицы «Season».

Таблица 4 – Season (Сезон, Сезоны)

Имя поля	Отображение	Тип поля	Что хранится?
name	Название	CharField(max_length=15)	Название сезона

В таблице 5 представлено описание таблицы «Condition».

Таблица 5 – Condition (Состояние, Состояния)

Имя поля	Отображение	Тип поля	Что хранится?
type	Тип	CharField(max_length=18)	Тип состояния

В таблице 6 представлено описание таблицы «Warehouse».

Таблица 6 – Warehouse (Склад, Склады)

Имя поля	Отображение	Тип поля	Что хранится?
name	Название	CharField(max_length=20)	Название склада
comment	Комментарий	CharField(max_length=50)	Комментарий

В таблице 7 представлено описание таблицы «Drilling».

Таблица 7 – Drilling (Сверление, Сверления)

Имя поля	Отображение	Тип поля	Что хранится?
value	Значение	CharField(max_length=15)	Значение сверловки

В таблице 8 представлено описание таблицы «Spacing».

Таблица 8 – Spacing (Расположение, Расположения)

Имя поля	Отображение	Тип поля	Что хранится?
place	Место	CharField(max_length=10)	Расположение запчасти

В таблице 9 представлено описание таблицы «CenterHole».

Таблица 9 – CenterHole (Центральное отверстие, Центральные отверстия)

Имя поля	Отображение	Тип поля	Что хранится?
value	Значение	CharField(max_length=10)	Значение центрального отверстия

В таблице 10 представлено описание таблицы «BodyType».

Таблица 10 – BodyType (Тип кузова, Типы кузовов)

Имя поля	Отображение	Тип поля	Что хранится?
name	Название	CharField(max_length=12)	Название типа кузова

В таблице 11 представлено описание таблицы «DiskType».

Таблица 11 – DiskType (Тип диска, Типы дисков)

Имя поля	Отображение	Тип поля	Что хранится?
name	Название	CharField(max_length=10)	Название типа диска

В таблице 12 представлено описание таблицы «TyreMaker».

Таблица 12 – TyreMaker (Производитель шин, Производители шин)

Имя поля	Отображение	Тип поля	Что хранится?
name	Название	CharField(max_length=20)	Название производителя шин

В таблице 13 представлено описание таблицы «Caliber».

Таблица 13 – Caliber (Посадочный диаметр, Посадочные диаметры)

Имя поля	Отображение	Тип поля	Что хранится?
value	Значение	CharField(max_length=20)	Значение диаметра шины

В таблице 14 представлено описание таблицы «AutoName».

Таблица 14 – AutoName (Название автомобиля, Названия автомобилей)

Имя поля	Отображение	Тип поля	Что хранится?
value	Значение	CharField(max_length=50)	Название автомобиля

В таблице 15 представлено описание таблицы «SpareName».

Таблица 15 – SpareName (Название запчасти, Названия запчастей)

Имя поля	Отображение	Тип поля	Что хранится?
value	Значение	CharField(max_length=50)	Название запчасти

В таблице 16 представлено описание таблицы «MakeOfVehicle».

Таблица 16 – MakeOfVehicle (Марка автомобиля, Марки автомобилей)

Имя поля	Отображение	Тип поля	Что хранится?
value	Значение	CharField(max_length=50)	Марка автомобиля

В таблице 17 представлено описание таблицы «Auto».

Таблица 17 – Auto (Автомобиль, Автомобили)

Имя поля	Отображение	Тип поля	Что хранится?
name	Название	ForeignKey(AutoName)	Название автомобиля
makeOfVehicle	Марка	ForeignKey(MakeOfVehicle)	Марка автомобиля
yom	Год выпуска	CharField(max_length=4)	Год выпуска автомобиля

Продолжение таблицы 17

Имя поля	Отображение	Тип поля	Что хранится?
body	Кузов	ForeignKey(BodyType)	Кузов автомобиля
motor	Двигатель	CharField(max_length=20)	Двигатель автомобиля
comment	Комментарий	CharField(max_length=50)	Комментарий

В таблице 18 представлено описание таблицы «Maker».

Таблица 18 – Maker (Производитель запчасти, Производители запчастей)

Имя поля	Отображение	Тип поля	Что хранится?
Name	Название	CharField(max_length=50)	Название производителя запчасти
mak-erNumber	Номер производителя	CharField(max_length=15)	Номер производителя запчасти

В таблице 19 представлено описание таблицы «Spare».

Таблица 19 – Spare (Запчасть, Запчасти)

Имя поля	Отображение	Тип поля	Что хранится?
name	Название	ForeignKey(SpareName)	Название запчасти
price	Цена	DecimalField(max_dig-its=9, decimal_places=2)	Цена запчасти
donor	Авто-донор	ForeignKey(Auto)	Авто-донор
yom	Год выпуска	CharField(max_length=4)	Год выпуска запчасти
maker	Производитель	ForeignKey(Maker)	Название производителя
marking	Маркировка	CharField(max_length=30)	Значение маркировки
condition	Состояние	ForeignKey(Condition)	Состояние запчасти
spacing	Расположение	ForeignKey(Spacing)	Расположение запчасти
color	Цвет	CharField(max_length=20)	Цвет запчасти
warehouse	Склад	ForeignKey(Warehouse)	Склад хранения
date	Дата	DateField()	Дата
comment	Комментарий	CharField(max_length=50)	Комментарий
photo	Фото	ImageField()	Фото запчасти
amount	Количество	IntegerField()	Количество запчастей на складе

В таблице 20 представлено описание таблицы «Disk».

Таблица 20 – Disk (Диск, Диски)

Имя поля	Отображение	Тип поля	Что хранится?
name	Название	CharField(max_length=50)	Название диска

Продолжение таблицы 20

Имя поля	Отображение	Тип поля	Что хранится?
price	Цена	DecimalField(max_digits=9, decimal_places=2)	Цена диска
caliber	Диаметр	ForeignKey(Caliber)	Посадочный диаметр диска
diskType	Тип	ForeignKey(DiskType)	Тип диска
width	Ширина	DecimalField(max_digits=5, decimal_places=2)	Значение ширины
drilling	Сверловка	ForeignKey(Drilling)	Значение сверления
deport	Вылет	CharField(max_length=20)	Значение вылета
centerHole	Центральное отверстие	ForeignKey(CenterHole)	Значение центрального отверстия
warehouse	Склад	models.ForeignKey(Warehouse)	Склад хранения
date	Дата	DateField()	Дата
comment	Комментарий	CharField(max_length=50)	Комментарий
photo	Фото	ImageField()	Фото диска
amount	Количество	IntegerField()	Количество дисков на складе

В таблице 21 представлено описание таблицы «Tyre».

Таблица 21 – Tyre (Шина, Шины)

Имя поля	Отображение	Тип поля	Что хранится?
name	Название	CharField(max_length=50)	Название шины
price	Цена	DecimalField(max_digits=9, decimal_places=2)	Цена шины
caliber	Диаметр	ForeignKey(Caliber)	Посадочный диаметр шины
season	Сезон	ForeignKey(Season)	Сезон
tyreMaker	Производитель	ForeignKey(TyreMaker)	Название производителя
model	Модель	CharField(max_length=20)	Модель шины
width	Ширина	DecimalField(max_digits=5, decimal_places=2)	Значение ширины
deport	Вылет	CharField(max_length=20)	Вылет шины
wearOut	Износ	CharField(max_length=50)	Износ шины
centerHole	Центральное отверстие	ForeignKey(CenterHole)	Значение центрального отверстия

Продолжение таблицы 21

Имя поля	Отображение	Тип поля	Что хранится?
warehouse	Склад	models.ForeignKey(Warehouse)	Склад хранения
date	Дата	DateField()	Дата
comment	Комментарий	CharField(max_length=50)	Комментарий
photo	Фото	ImageField()	Фото шины
amount	Количество	IntegerField()	Количество шин на складе

В таблице 22 представлено описание таблицы «Set».

Таблица 22 – Set (Комплект, Комплекты)

Имя поля	Отображение	Тип поля	Что хранится?
name	Название	CharField(max_length=50)	Название комплекта
number	Номер	CharField(max_length=6)	Номер комплекта
price	Цена	DecimalField(max_digits=9, decimal_places=2)	Цена комплекта
caliber	Диаметр	ForeignKey(Caliber)	Посадочный диаметр
season	Сезон	ForeignKey(Season)	Сезон
diskType	Тип диска	ForeignKey(DiskType)	Тип диска
diskWidth	Ширина диска	DecimalField(max_digits=5, decimal_places=2)	Значение ширины диска
drilling	Сверление	ForeignKey(Drilling)	Значение сверления
tyreMaker	Производитель шин	ForeignKey(TyreMaker)	Название производителя шин
model	Модель	CharField(max_length=20)	Модель шины
tyreWidth	Ширина шины	DecimalField(max_digits=5, decimal_places=2)	Значение ширины шины
tyreHeight	Высота шины	DecimalField(max_digits=5, decimal_places=2)	Значение высоты шины
deport	Вылет	CharField(max_length=20)	Вылет
wearOut	Износ	CharField(max_length=50)	Износ шины
centerHole	Центральное отверстие	ForeignKey(CenterHole)	Значение центрального отверстия
warehouse	Склад	models.ForeignKey(Warehouse)	Склад хранения

Продолжение таблицы 22

Имя поля	Отображение	Тип поля	Что хранится?
date	Дата	DateField()	Дата
comment	Комментарий	CharField(max_length=50)	Комментарий
photo	Фото	ImageField()	Фото шины
amount	Количество	IntegerField()	Количество комплектов на складе

В таблице 23 представлено описание таблицы «SparePriceHistory».

Таблица 23 – SparePriceHistory (История цен (запчасти), История цен (запчасти))

Имя поля	Отображение	Тип поля	Что хранится?
spare	Запчасть	ForeignKey(Spare)	Запчасть
price	Цена	DecimalField(max_digits=9, decimal_places=2)	Цена запчастиЫ
date	Дата	DateTimeField()	Дата изменения цены запчасти
user	Пользователь	ForeignKey(User)	Логин пользователя, который изменил цену

В таблице 24 представлено описание таблицы «TyrePriceHistory».

Таблица 24 – TyrePriceHistory (История цен (шины), История цен (шины))

Имя поля	Отображение	Тип поля	Что хранится?
tyre	Шина	ForeignKey(Tyre)	Шина
price	Цена	DecimalField(max_digits=9, decimal_places=2)	Цена шины
date	Дата	DateTimeField()	Дата изменения цены шины
user	Пользователь	ForeignKey(User)	Логин пользователя, который изменил цену

В таблице 25 представлено описание таблицы «DiskPriceHistory».

Таблица 25 – DiskPriceHistory (История цен (диски), История цен (диски))

Имя поля	Отображение	Тип поля	Что хранится?
disk	Диск	ForeignKey(Disk)	Диск
price	Цена	DecimalField(max_digits=9, decimal_places=2)	Цена диска
date	Дата	DateTimeField()	Дата изменения цены диска
user	Пользователь	ForeignKey(User)	Логин пользователя, который изменил цену

В таблице 26 представлено описание таблицы «SetPriceHistory».

Таблица 26 – SetPriceHistory (История цен (комплекты), История цен (комплекты))

Имя поля	Отображение	Тип поля	Что хранится?
set	Комплект	ForeignKey(Tyre)	Комплект
price	Цена	DecimalField(max_digits=9, decimal_places=2)	Цена комплекта
date	Дата	DateTimeField()	Дата изменения цены комплекта
user	Пользователь	ForeignKey(User)	Логин пользователя, который изменил цену

В таблице 27 представлено описание таблицы «SpareSell».

Таблица 27 – SpareSell (Продажа (запчасти), Продажа (запчасти))

Имя поля	Отображение	Тип поля	Что хранится?
spare	Запчасть	ForeignKey(Spare)	Запчасть
price	Цена	DecimalField(max_digits=9, decimal_places=2)	Цена запчасти
amount	Количество	IntegerField()	Количество проданных запчастей
date	Дата	DateTimeField()	Дата продажи запчасти
user	Пользователь	ForeignKey(User)	Логин пользователя, который продал запчасть

В таблице 28 представлено описание таблицы «TyreSell».

Таблица 28 – TyreSell (Продажа (шины), Продажа (шины))

Имя поля	Отображение	Тип поля	Что хранится?
tyre	Шина	ForeignKey(Tyre)	Шина
price	Цена	DecimalField(max_digits=9, decimal_places=2)	Цена шины
amount	Количество	IntegerField()	Количество проданных шин
date	Дата	DateTimeField()	Дата продажи шины
user	Пользователь	ForeignKey(User)	Логин пользователя, который продал шину

В таблице 29 представлено описание таблицы «DiskSell».

Таблица 29 – DiskSell (Продажа (диски), Продажа (диски))

Имя поля	Отображение	Тип поля	Что хранится?
disk	Диск	ForeignKey(Disk)	Диск
price	Цена	DecimalField(max_digits=9, decimal_places=2)	Цена диска
amount	Количество	IntegerField()	Количество проданных дисков
date	Дата	DateTimeField()	Дата продажи диска
user	Пользователь	ForeignKey(User)	Логин пользователя, который продал диск

В таблице 30 представлено описание таблицы «SetSell».

Таблица 30 – SetSell (Продажа (комплекты), Продажа (комплекты))

Имя поля	Отображение	Тип поля	Что хранится?
set	Комплект	ForeignKey(Tyre)	Комплект
price	Цена	DecimalField(max_digits=9, decimal_places=2)	Цена комплекта
amount	Количество	IntegerField()	Количество проданных комплектов
date	Дата	DateTimeField()	Дата продажи комплекта
user	Пользователь	ForeignKey(User)	Логин пользователя, который продал комплект

3.3 Описание работы программы

Открыв веб-приложение, пользователь попадает на главную страницу. На главную страницу можно также попасть, нажав на логотип RAZBORIN. На этой странице размещен текст, который советует пользователю войти в учетную запись с помощью административной панели. Переходя на разные пункты меню, они выделяются красным цветом, чтобы пользователю было легче ориентироваться на страницах.

Главная страница изображена на рисунке 11.

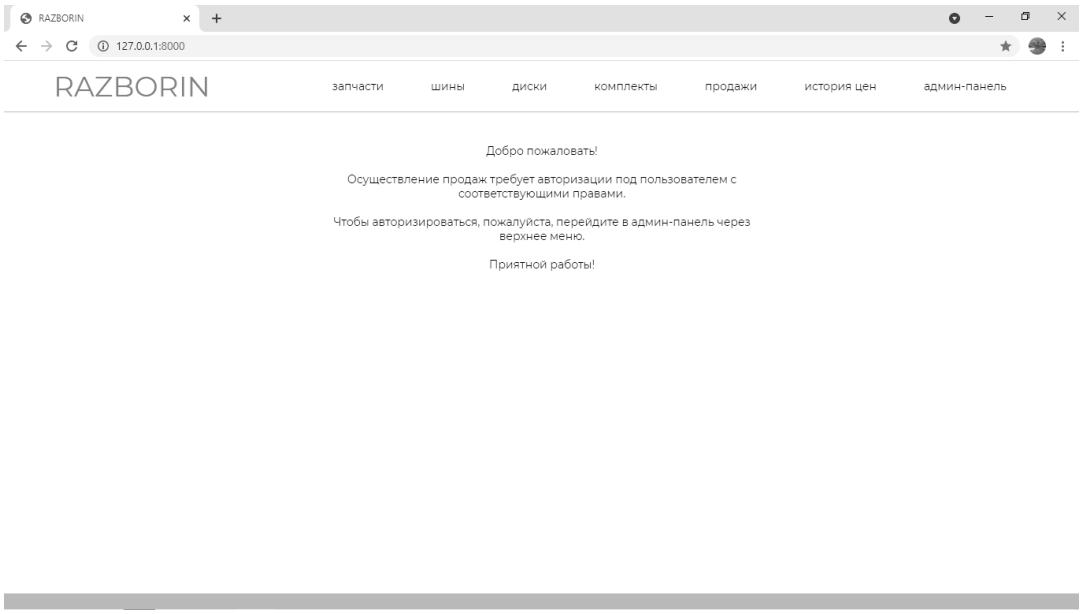

Рисунок 11 – Главная страница

Следующими по порядку идут страницы запчастей, шин, дисков и комплектов, на которые можно попасть, нажав на соответствующие пункты меню в верхней части экрана. Страница с запчастями показана на рисунке 12, страницы с шинами, дисками и комплектами выполнены аналогично, отличаются лишь фильтры.

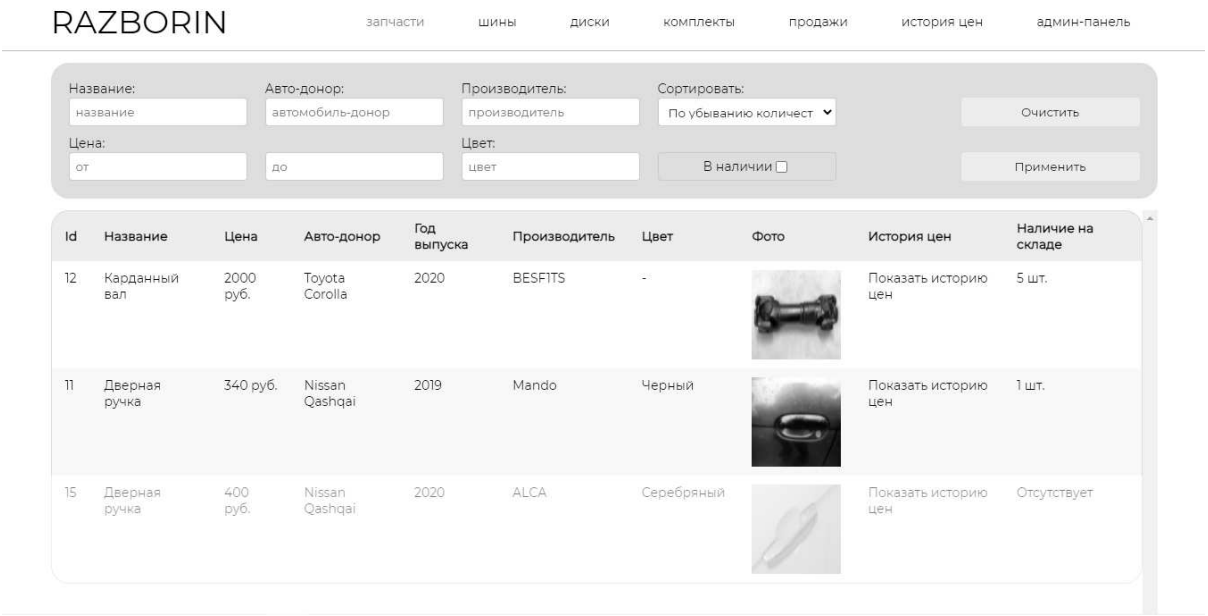

Рисунок 12 – Страница «запчасти»

На каждой из этих страниц можно фильтровать и сортировать данные по нескольким параметрам. Фильтры и сортировка применяются при нажатии на

соответствующую кнопку. Форму с фильтрами можно быстро очистить с помощью соответствующей кнопки, что экономит время пользователя. Пример отфильтрованных данных можно увидеть на рисунке 13.

Рисунок 13 – Отфильтрованные данные

Хочется отметить, что, если детали нет в наличии, то она выглядят более прозрачной, чем детали в наличии. Если навести на фотографию детали, то она увеличится в размере, что позволит лучше ее рассмотреть.

Далее в верхнем меню размещается пункт меню «продажи». Этот пункт содержит выпадающий список, в котором можно выбрать, какие продажи надо отобразить. Остальные три пункта выпадающего списка выполнены точно так же. Для каждой таблицы можно задать фильтрацию, как и в таблицах запчастей, шин, дисков и комплектов. Кнопки «Применить» и «Очистить» также остались на месте. Для каждой продажи считается сумма, которая не хранится в базе данных, потому что ее можно вычислить, зная цену и количество проданного товара.

Страница «продажи» для запчастей продемонстрирована на рисунке 14.

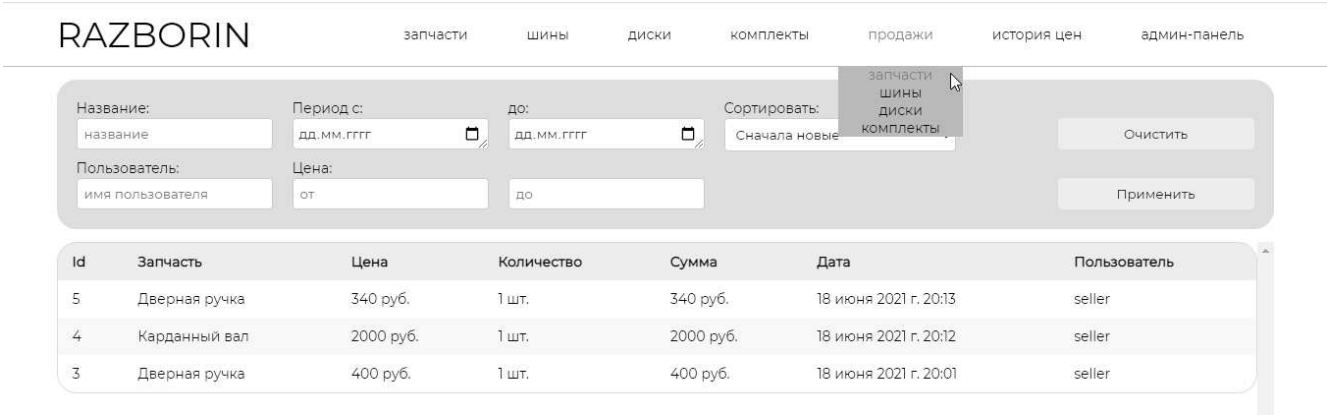

Рисунок 14 – Страница «продажи»

Предпоследний пункт меню – это «история цен». История цен хранит все цены товаров (для каждого подпункта выпадающего списка (запчасти, шины, диски, комплекты) отдельно). Таблица пополняется данными при добавлении детали, либо при изменении ее цены. Как и в продажах, в истории цен отмечается пользователь, который добавил деталь, либо изменил ее цену, а также дата и время, когда действие было совершено. Данные можно фильтровать и сортировать, как и на предыдущих страницах.

Страница истории цен запчастей показана на рисунке 15.

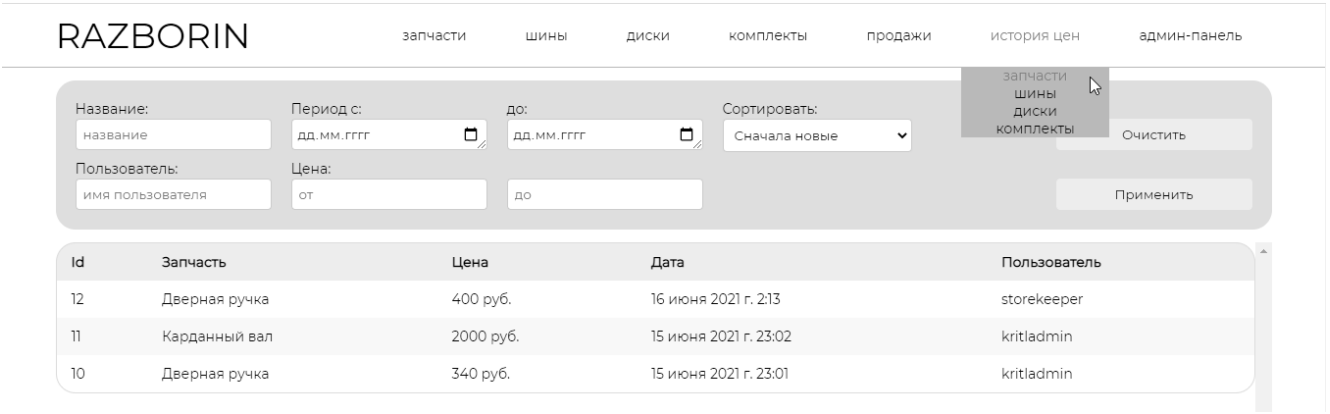

Рисунок 15 – Страница «история цен»

Последний пункт меню – это «админ-панель». При переходе в админ-панель пользователь попадает в панель администрирования приложения. Сначала пользователя, если он еще не авторизован, попросят авторизоваться.

Авторизация показана на рисунке 16.

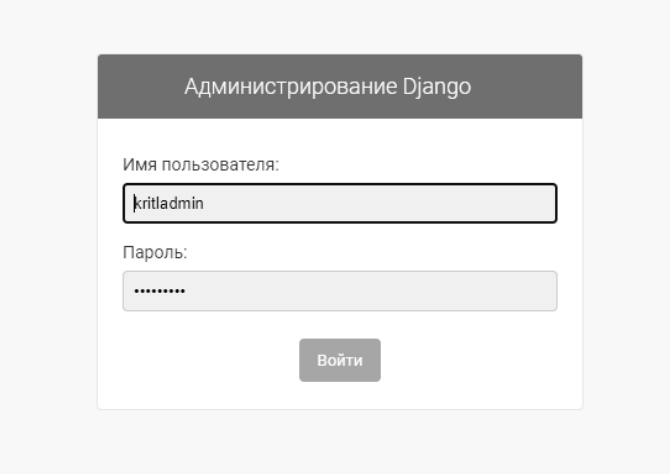

Рисунок 16 – Авторизация

После авторизации пользователь попадает на страницу администрирования Django. Здесь настроены различные таблицы базы данных. В зависимости от группы пользователей, а, следовательно, от прав доступа, различаются функции для административной панели.

Эта страница показана на рисунке 17.

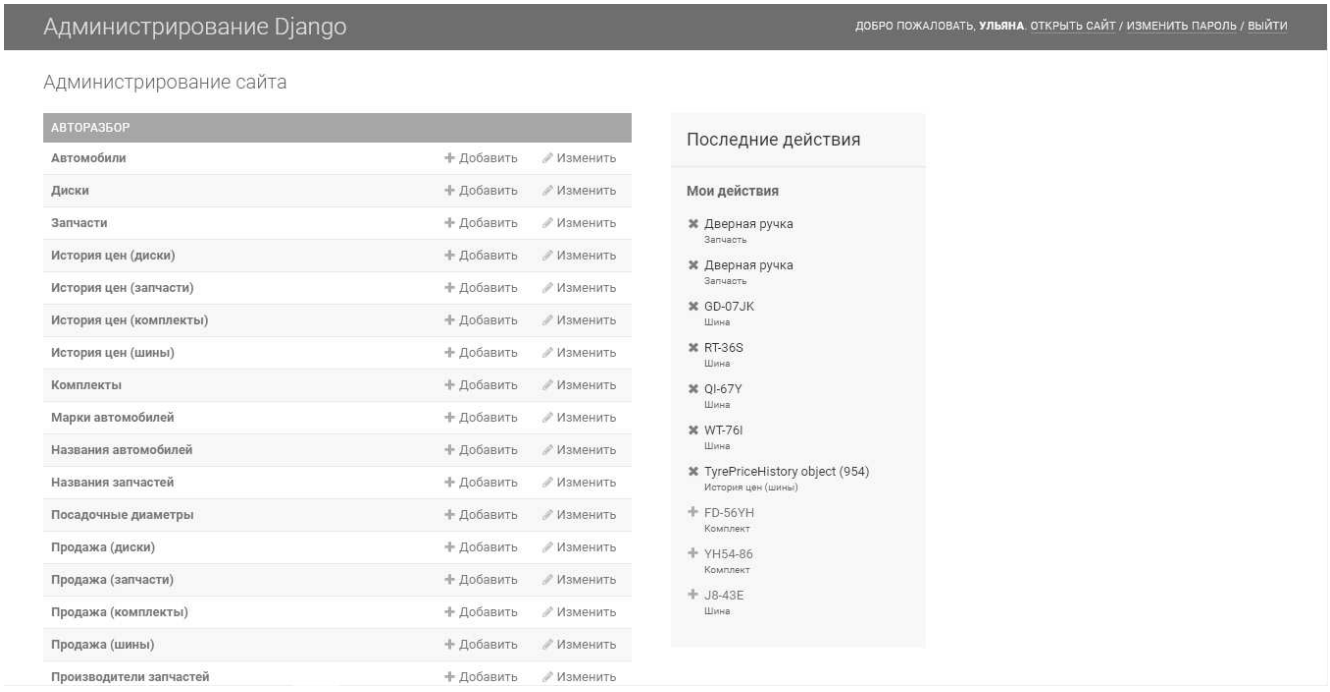

Рисунок 17 – Административная панель

Пользователи, состоящие в группе «Администратор», имеют все права (на добавление, изменение и удаление данных всех таблиц). Пользователи из группы

«Кладовщик» могут добавлять, изменять и удалять данные из всех таблиц, кроме таблиц, в которых содержатся данные историй цен, продаж, пользователей и групп пользователей, их они могут лишь просматривать (кроме пользователей и групп пользователей). Пользователи из группы «Продавец» могут только просматривать все таблицы, кроме таблицы, содержащей информацию о пользователях, и таблицы с группами пользователей. Группы пользователей показаны на рисунке 18.

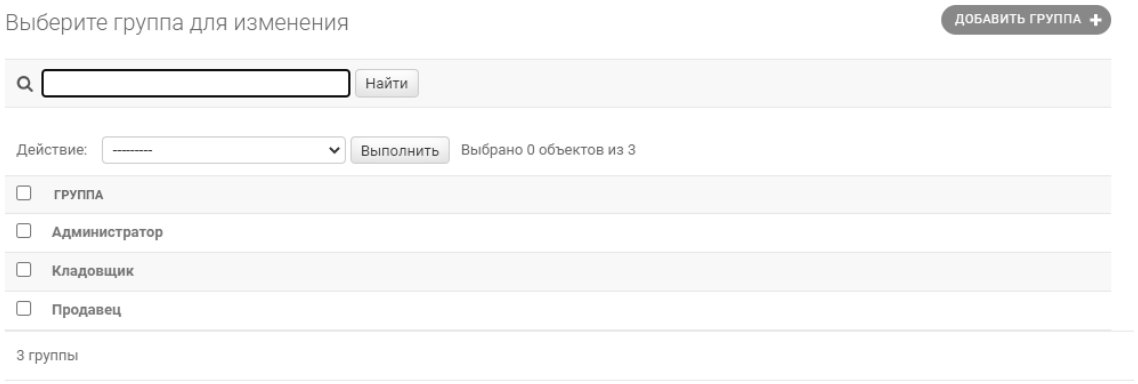

Рисунок 18 – Группы пользователей

Таблицы в панели администрирования сделаны специально для добавления, изменения и удаления информации. Пример редактирования запчасти показан на рисунке 19.

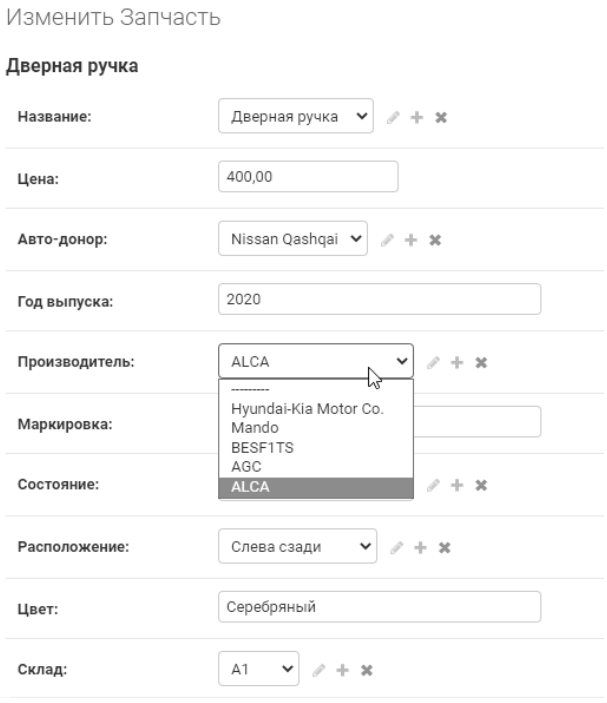

Рисунок 19 – Редактирование запчасти

Если пользователю необходимо найти шину по какому-либо параметру, то для этого предусмотрена строка для поискового запроса. Также можно массово удалять детали, отметив несколько деталей и выбрав действие удаления. Если не заполнить обязательное поле, либо допустить в нем ошибку, то сохранить изменения не получится, потому что появится сообщение об ошибке.

Сообщение об ошибке показано на рисунке 20.

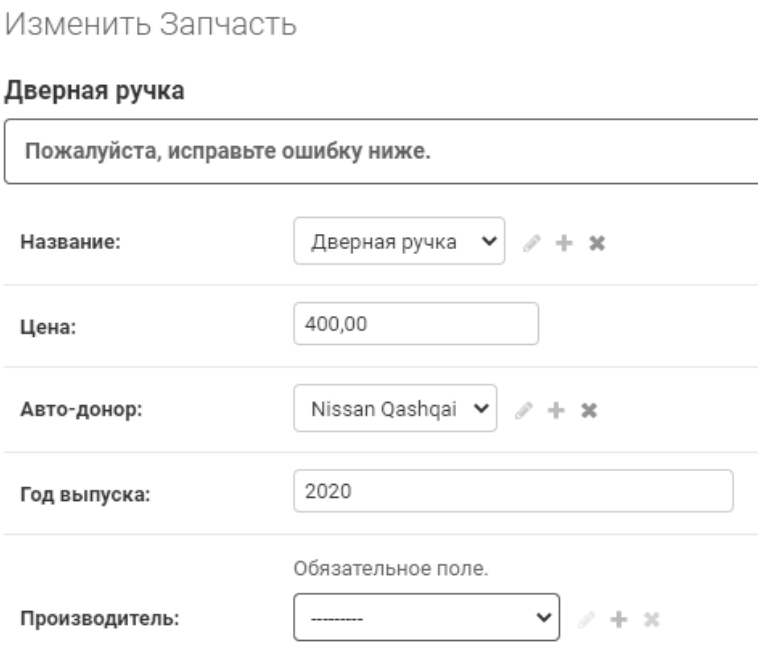

Рисунок 20 – Сообщение об ошибке в админ-панели

Далее рассмотрена функция веб-приложения, которая доступна только для пользователя из группы «Продавец» – функция продажи. Если на любой из страниц нажать на название детали, то появится страница с полной информацией о ней. Для удобства страница открывается в новом окне в половину размера. Пример страницы изображен на рисунке 21.

Id	Название	Цена	Наличие на складе	Авто-донор	Год выпуска	Производитель	Ном про
12	Карданный вал	2000,00 руб.	4 шт.	Toyota Corolla	2020	BESF1TS	8746

Рисунок 21 – Детальная запись

Если пользователь не входит в группу «Продавец», то форма для продажи не появится. Если ввести число меньше, либо равное нулю, то пользователю будет показано сообщение о том, что количество меньше единицы, и продажа не может быть совершена.

Такое сообщение показано на рисунке 22.

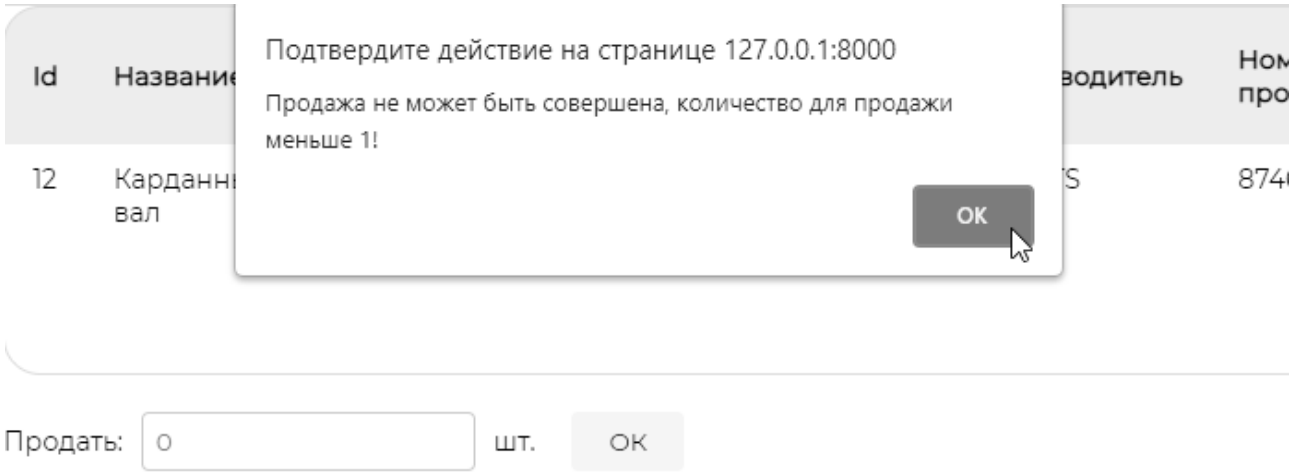

Рисунок 22 – Количество меньше единицы

Если ввести количество, которое будет больше имеющегося на складе, то программа оповестит об этом пользователя, продажа не будет совершена. Сообщение с этой ошибкой есть на рисунке 23.

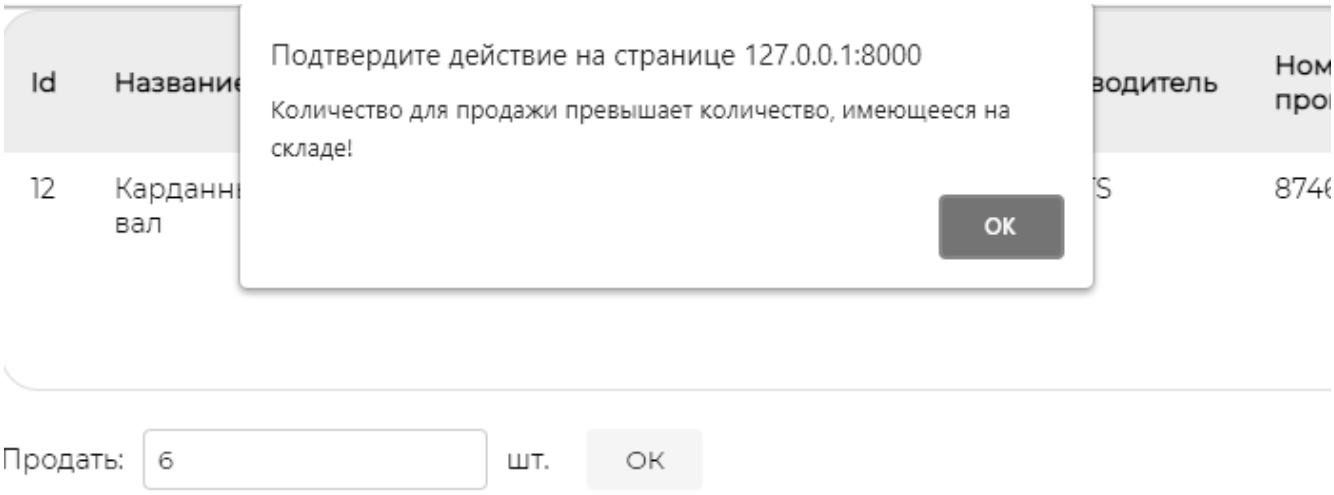

Рисунок 23 – Количество больше имеющегося на складе

Если же все введено верно, товар в прописанном количестве присутствует на складе, то приложение запросит подтверждение продажи, которое есть на рисунке 24.

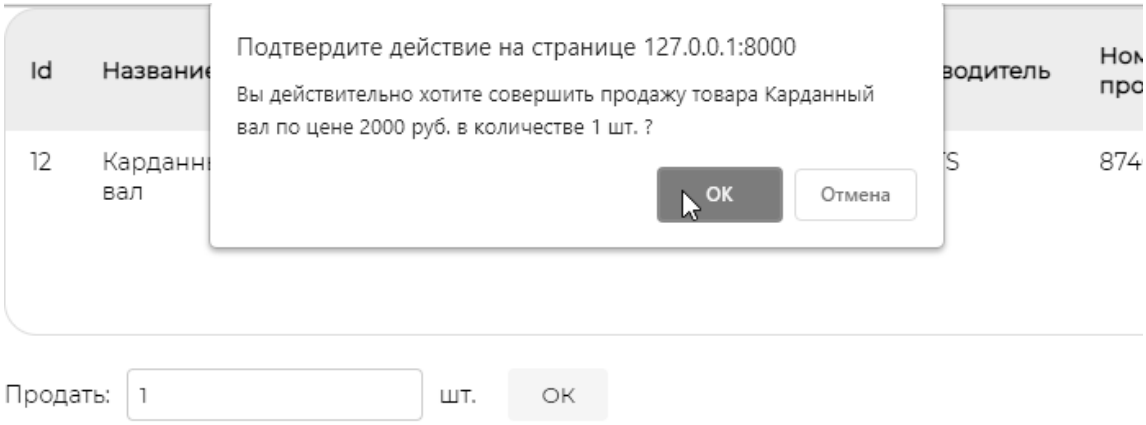

Рисунок 24 – Подтверждение продажи

Если ответ пользователя – «Отмена», то пользователь увидит сообщение о том, что продажа была отменена, как на рисунке 25.

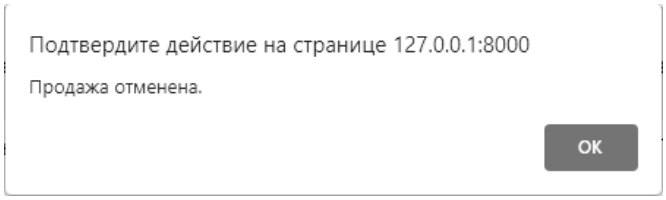

Рисунок 25 – Отмена продажи

Если же ответ – «ОК», то продажа будет произведена, а пользователь увидит сообщение с информацией о продаже, как на рисунке 26. Информация о продаже автоматически занесется в соответствующую таблицу, а количество выбранной детали на складе сразу же изменится.

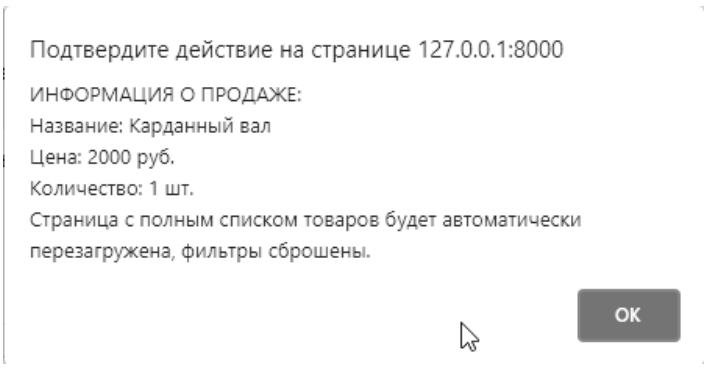

Рисунок 26 – Информация о продаже

3.4 Структура проекта

На рисунке 27 изображена структура проекта.

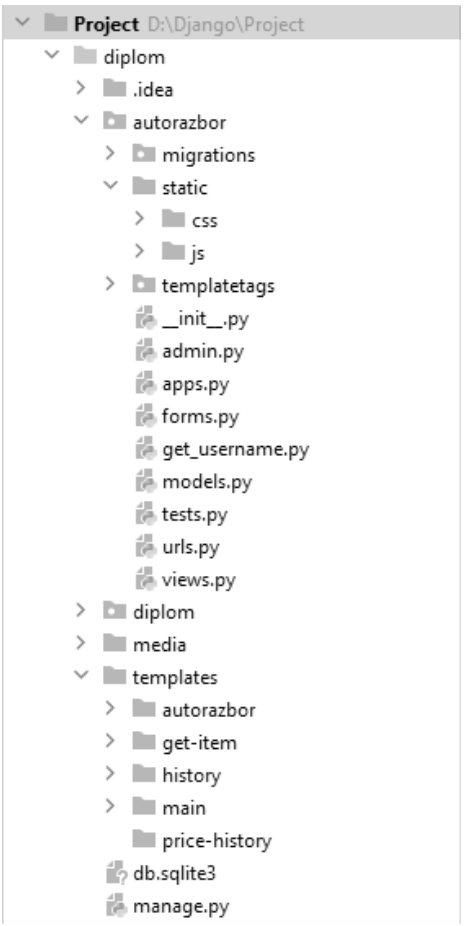

Рисунок 27 – Структура проекта

Папка migrations

Директория «migrations» хранит все файлы миграций проекта.

Django применяет миграции для переноса изменений, происходящих в моделях (изменение названия поля, добавление модели и так далее) на структуру базы данных. Миграции – это своеобразная система контроля версий для базы данных.

Папка static

Веб-приложение использует различные статические файлы – файлы стилей (css), скриптов (javascript) и так далее. Для хранения файлов такого типа и создана директория «static».

Файл pagination.js

В файле pagination.js находится функция для пагинации страниц. Листинг кода файла находится в приложении А. С помощью get-запроса осуществляется запрос к серверу без перезагрузки страницы, благодаря этому меняется лишь часть страницы, а именно, таблица, содержащая строки со следующими в базе данными.

Пагинация – это разделение большого массива данных, имеющихся на сайте, на отдельные страницы для удобства использования.

Также в данном файле находится функция для определения текущей страницы, которая после определения страницы добавляет элементу меню, соответствующему url текущей страницы, дополнительный еще один класс, благодаря которому этот элемент меняет цвет на красный, таким образом помечая текущий раздел меню.

Файл sell.js

В этом файле хранится код функции, позволяющей пользователю группы «Продавец» совершать продажу деталей. Благодаря post-запросу функция отправляет данные для продажи на сервер, где используется функция добавления в базу данных этой информации. Листинг кода функции приведен в приложении Б.

Файлы tyre_filter.js, spare_filter.js, disk_filter.js, set_filter.js, pricehistory_sell_filter.js

Во всех этих файлах хранится код для фильтрации соответствующих таблиц. Рассмотрим фильтрацию на примере файла tyre_filter.js, листинг кода которого приведен в приложении В. Код остальных файлов аналогичен.

Сначала берутся данные с текущей страницы, затем, с помощью get-запроса берутся данные, подходящие требуемой фильтрации. Затем часть страницы меняется в соответствии с полученными данными.

Папка templatetags

Эта директория должна содержать собственные шаблонные теги и фильтры.

Шаблон – это текстовый файл. Он может генерировать любой текстовый формат (HTML, XML, CSV и т.д.).

Шаблон содержит переменные, которые заменяются значениями при оценке шаблона, и теги, которые управляют логикой шаблона.

Когда механизм шаблонов встречает переменную, он вычисляет эту переменную и заменяет ее результатом.

Но также можно изменять отображаемые переменные с помощью фильтров.

Фильтры отображают значение переменной после фильтрации.

Файл filters_extras.py

Этот файл содержит собственный шаблонный фильтр, который определяет, есть ли группа пользователей, которую указали в фильтре. Листинг кода фильтра приведен в приложении Г.

Папка media

В этой папке хранятся все добавленные в приложение изображения.

Папка templates

Эта папка содержит различные html-файлы, являющиеся стандартными веб-страницами. С их помощью веб-приложение отображается в браузере.

Папка Diplom

Эта директория содержит файлы проекта.

Файл manage.py

Этот файл выполняет различные команды проекта, например, запускает приложение, создает миграции.

Файл __init__.py

Данный файл указывает, что папка, в которой он находится, будет рассматриваться как модуль. Это стандартный файл для Python.

Файл settings.py:

Файд settings.py содержит настройки конфигурации проекта.

Файл urls.py

Этот файл содержит шаблоны URL-адресов, по сути, он определяет систему маршрутизации проекта

Файл wsgi.py

Файл содержит свойства конфигурации WSGI (Web Server Gateway Inerface). Он используется при развертывании проекта.

Файл asgi.py

Файл содержит свойства конфигурации ASGI (Asynchronous Server Gateway Inerface). Он также используется при развертывании проекта.

Папка autorazbor

Веб-приложение или проект Django состоит из отдельных приложений. Вместе они образуют полноценное веб-приложение. Каждое приложение представляет какую-то определенную функциональность или группу функциональностей.

При создании проекта он уже содержит несколько приложений по умолчанию: django.contrib.admin, django.contrib.auth, django.contrib.contenttypes, django.contrib.sessions, django.contrib.messages, django.contrib.staticfiles.

Список всех приложений можно найти в проекте в файле settings.py в переменной INSTALLED_APPS.

Директория autorazbor содержит файлы веб-приложения.

Файл urls.py

Чтобы функции сопоставлялись с запросами, надо определить для них маршруты в файле urls.py.

Переменная urlpatterns определяет набор сопоставлений функций обработки с определенными строками запроса. Листинг кода файла показан в приложении Д.

Файл models.py

Модели в Django описывают структуру используемых данных. Используемые в программе данные хранятся в базах данных, а с помощью моделей осуществляется взаимодействие с базой данных. Листинг кода модели Tyre находится в приложении Е. Остальные модели похожи по реализации.

Для некоторых моделей переопределено сохранение, так как для таблиц продаж и историй цен необходимо добавлять данные во время сохранения данных других моделей.

По умолчанию Django в качестве базы данных использует SQLite. Но для данного проекта было принято решение использовать СУБД MySQL.

Файл admin.py

В этом файле определяется, какие модели будут редактироваться через интерфейс администратора.

Для каждой модели создается класс ModelAdmin, который инкапсулирует настройки интерфейса администратора для конкретной модели.

Листинг кода TyreAdmin приведен в приложении Ж.

Файл get_username.py

Middleware – это промежуточный слой между запросом(request) и ответом (response). В фреймворке Django уже по умолчанию содержатся наиболее важные и нужные middleware. Каждый компонент промежуточного слоя отвечает за определенный функционал.

В этом файле содержится middleware, который используется для получения логина пользователя. Листинг кода файла находится в приложении З.

Файл views.py

Центральным моментом любого веб-приложения является обработка запроса, который отправляет пользователь. В Django за обработку запроса отвечают представления или views. По сути, представления представляют функции обработки, которые принимают данные запроса в виде объекта request и генерируют некоторый результат, который затем отправляется пользователю.

По умолчанию представления размещаются в приложении в файле views.py. Данный файл имеет представления для обработки всех запросов приложения.

Листинг кода представлений TyreListView, get_tyre и get_tyre_history расположен в приложении И.

3.5 Описание работы телеграм-бота

Первый раз при входе в диалог к боту, пользователь увидит экран с описанием того, что может бот. Этот экран показан на рисунке 28. Диалог начинается с помощью кнопки «Старт».

Рисунок 28 – Начальный экран

Сначала бот предлагает на выбор четыре варианта для добавления: запчасть, шина, диск, комплект. В зависимости от выбранного варианта, будут различаться добавляемые данные. Начало диалога с ботом показано на рисунке 29.

Рисунок 29 – Начало диалога

Сообщения условно можно поделить на 2 вида.

1. Сообщения с вводом ответа.
2. Сообщения с выбором ответа.

На сообщения с вводом ответа надо отвечать вводом чисел, слов, даты или фотографии. Пример такого типа сообщений показан на рисунке 30.

Рисунок 30 – Ввод данных

Если ввести данные не так, как рекомендует бот, то данные добавлены не будут. Бот отправит сообщение об ошибке и попросит ввести данные заново. Сообщение об ошибке представлено на рисунке 31.

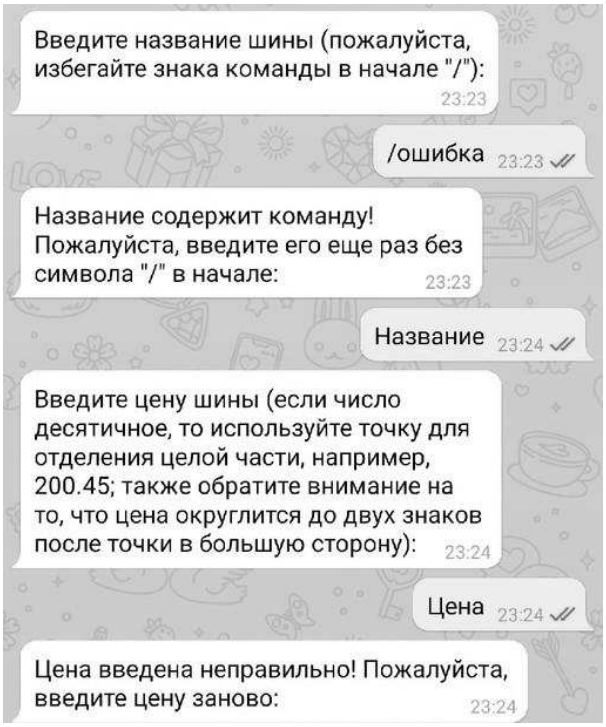

Рисунок 31 – Ошибка ввода данных

Сообщения с выбором ответа позволяют пользователю выбрать существующие варианты. Если данных в таблице меньше количества, определенного в программе, то выведется список всех вариантов ответа. Пример такого типа сообщения и ответа на него представлен на рисунке 32.

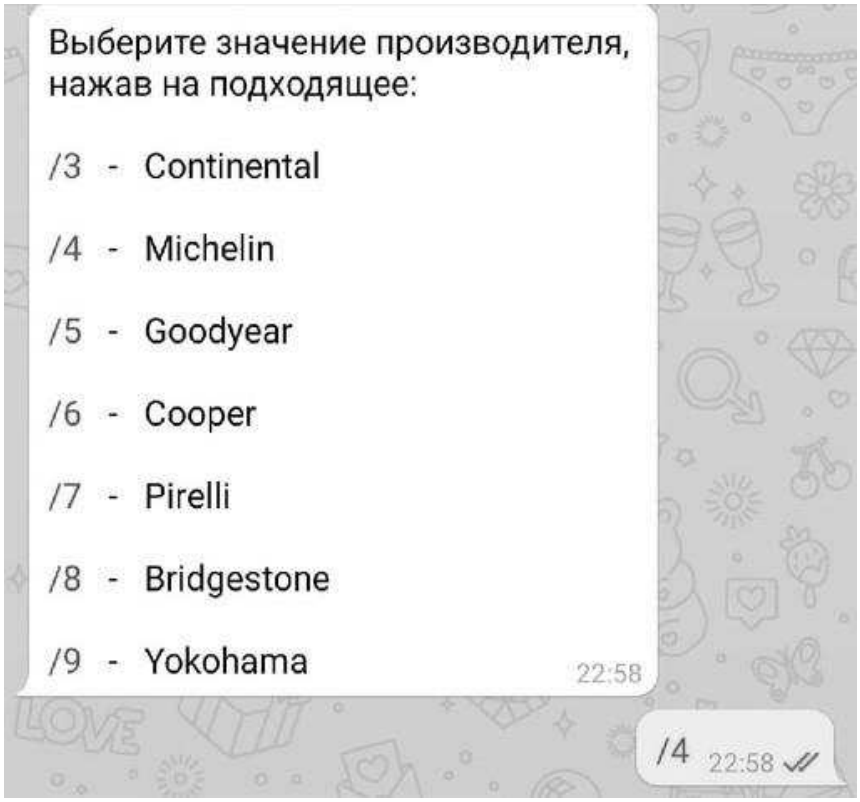

Рисунок 32 – Вывод всех данных

Если в нужной таблице данных больше, то пользователь увидит сообщение о том, сколько всего записей найдено и предложит отправить слово «да» для вывода полного списка вариантов и значение для поиска по таблице, если не требуется вывод полного списка, либо записей слишком много. Данная функция была предусмотрена для удобства пользователя. После того, как пользователь нашел подходящий вариант, ему необходимо нажать на число слева от значения, тем самым, будет сделан выбор. Стоит обратить внимание на то, что, пока не сделан выбор, действия поиска и вывода полного списка значений можно повторять столько, сколько нужно пользователю.

Пример такого сообщения с выбором варианта «да» приведен на рисунке 33.

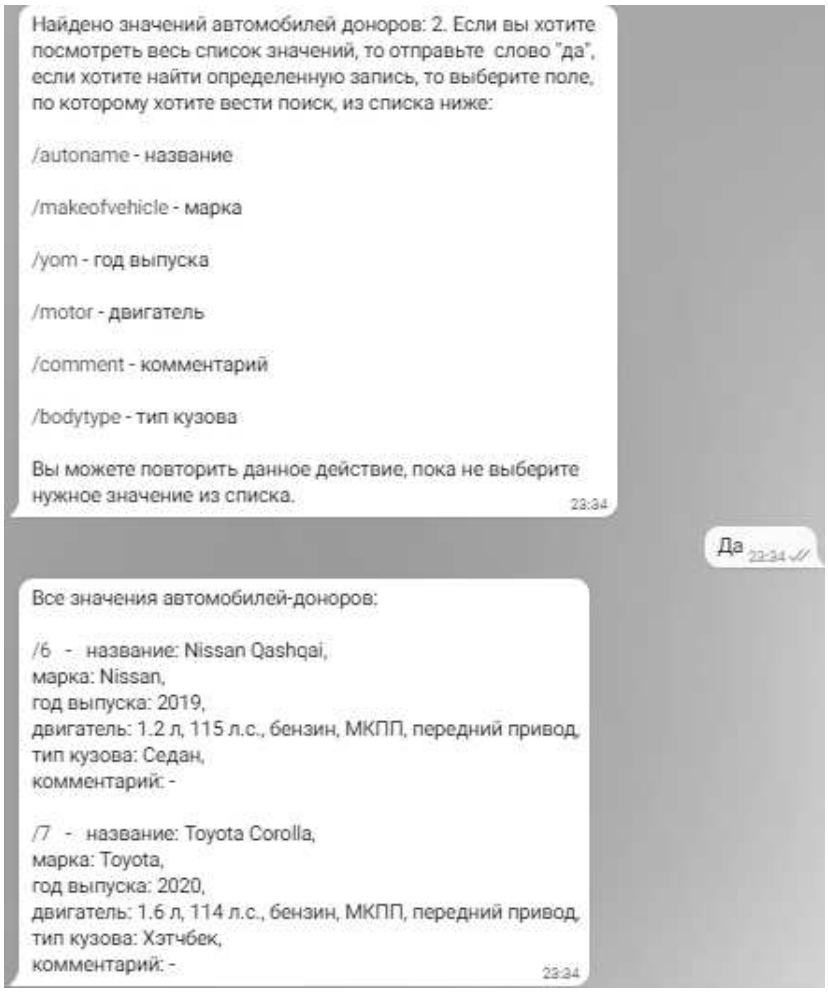

Рисунок 33 – Выбор варианта «да»

Пример поиска значения показан на рисунке 34.

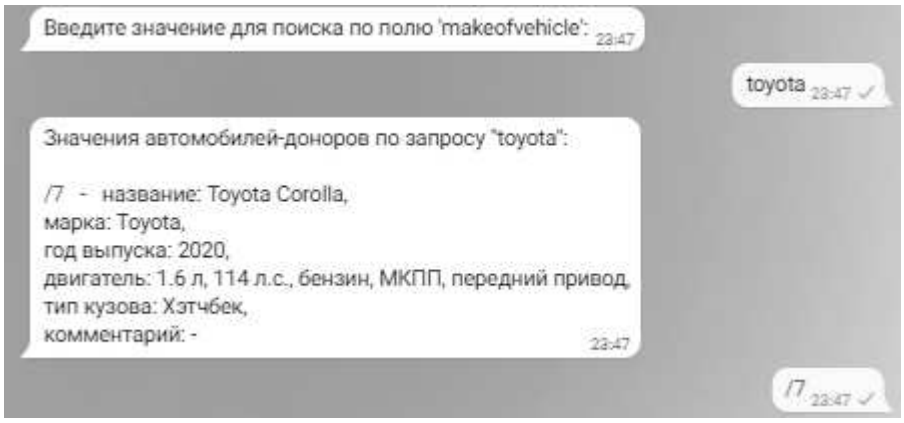

Рисунок 34 – Поиск значения

При выборе значения тоже могу присутствовать ошибки, поэтому и здесь предусмотрена защита от ввода некорректных данных. Пример защиты приведен на рисунке 35.

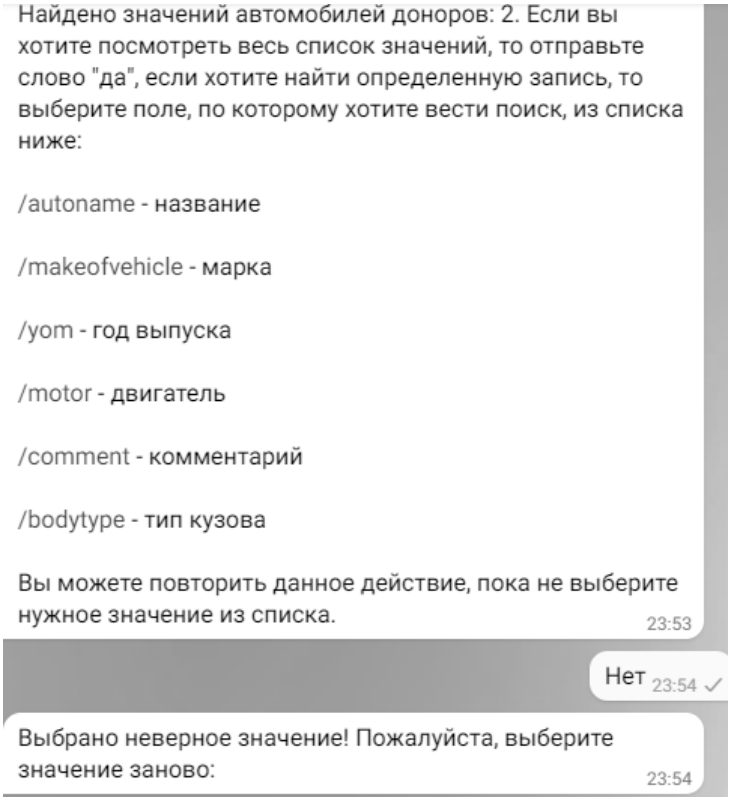

Рисунок 35 – Ошибка во втором типе сообщений

Если пользователь по какой-то причине решил не продолжать добавление текущих данных в базу, то для этого предусмотрена команда «/anew». Она прерывает добавление и предлагает пользователю заново выбрать то, что он планирует добавить. Пример работы команды показан на рисунке 36.

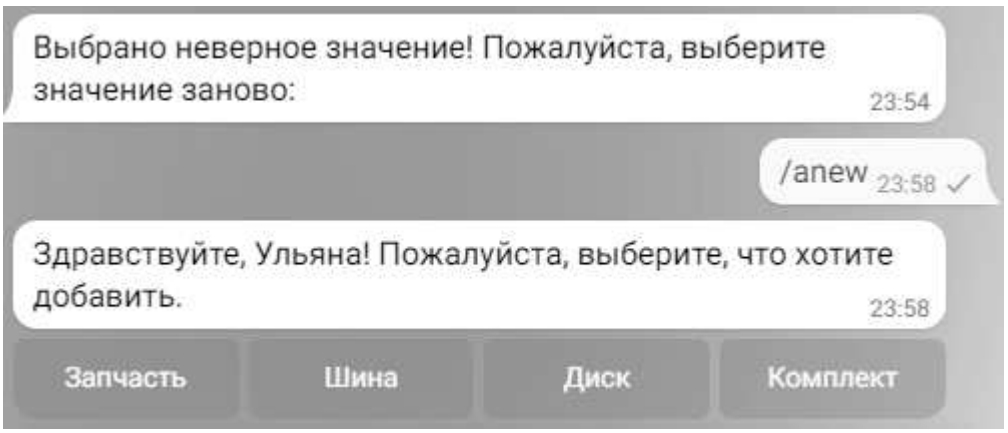

Рисунок 36 – Команда «/anew»

Если все данные были заполнены верно, то в конце пользователь увидит сообщение об успешном добавлении выбранной детали в базу. Он может сразу начать добавлять что-то другое, воспользовавшись командой «/start». Сообщение об успешном добавлении показано на рисунке 37.

Рисунок 37 – Успешное добавление

3.6 Структура телеграм-бота

На рисунке 38 изображена структура телеграм-бота.

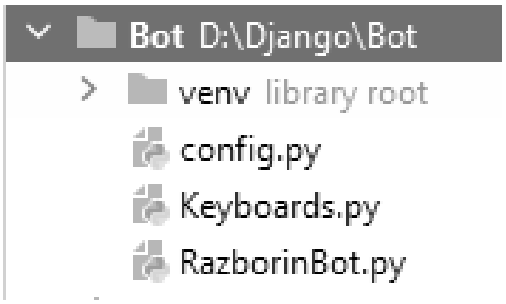

Рисунок 38 – Структура бота

Файл config.py

Этот файл содержит токен (ключ авторизации), с помощью которого бот подключается к API телеграмма. Токен бота показан на рисунке 39.

```
token = '1504821324:AAGR2-K1eTS9FkVeXi2AGlwJ1GJqdduD-Bo'
```

Рисунок 39 – Токен

Файл Keyboards.py

Этот файл содержит код для генерации клавиатуры с выбором детали. Листинг кода файла приведен в приложении К.

Файл RazborinBot.py

Этот файл содержит все функции для работы бота, а также различные словари, необходимые для его работы. Листинг кода добавления запчасти продемонстрирован в приложении Л.

Вывод по третьему разделу

Рассмотрена разработка базы данных, самого веб-приложения и телеграм-бота. Приведено полное описание работы системы учета запчастей и телеграм-бота. Определена структура основного проекта и структура проекта бота.

4 ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ

Затраты на заработную плату программисту составляют: 52.000 руб в месяц, согласно средней заработной плате программиста в Челябинске.

Затраты на электроэнергию составляют около 260 руб в месяц – средняя стоимость электроэнергии в Челябинске.

Затраты на оборудование:

- ПК – 40.000 руб.

Длительность разработки составляет 6 месяцев.

Вычисляем итоговую сумму, сложив затраты на заработную плату и электроэнергию, затем умножив на количество месяцев, а после этого сложив с затратами на оборудование: (52.000 + 260) * 6 + 40.000 = 353.560 руб.

Дальнейшее обслуживание и доработки: по договоренности от 5.000 руб.

В проекте окупаемость заменена ресурсами, которые являются для предприятия более важными, чем денежный. Главным эффектом от создания и внедрения приложения является увеличение экономических и хозяйственных показателей. В разы уменьшается время, потраченное на учет деталей, вследствие чего уменьшается количество работников, вовлеченных в учет. Затраты на заработную плату и страховые отчисления уменьшаются. За счет роста продуктивности увеличивается и прибыль, так как клиентам не приходится долго ждать выполнения заказа.

Вывод по четвертому разделу

Рассчитаны затраты на разработку проекта, обоснована экономическая эффективность проекта.

ЗАКЛЮЧЕНИЕ

В процессе выполнения выпускной квалификационной работы поставлены цели и задачи, обоснована актуальность и необходимость выбранной темы. Проведен анализ предметной области. Проведен сравнительный анализ существующих систем.

Проанализирован и обоснован выбор средств и языков разработки. Реализована база данных для приложения, а также различные функции и интерфейсы приложения и телеграм-бота. Описаны модели приложения и функционал.

Цель работы достигнута.

В дальнейшем планируются такие доработки системы, как, например, отслеживание поступлений количества деталей на склад и возможность добавления нескольких фотографий за один раз.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Пошаговая инструкция автоматизации складского учета товаров: что такое и как работает. – www.ekam.ru/blogs/pos/avtomatizatsiya-skladskogo-ucheta-tovarov;
2. Миндалев, И.В. Моделирование бизнес-процессов с помощью IDEF0, DFD, BPMN за 7 дней: учеб. пособие / И.В. Миндалев; Краснояр. гос. аграр. ун-т. – Красноярск, 2016. – 123 с.
3. 3 способа разработки веб-сайта. – www.vc.ru/dev/78714-3-sposoba-razrabotki-veb-sayta;
4. Фреймворки в веб-разработке – www.web-creator.ru/articles/ about_frameworks;
5. Форсье, Дж. Django. Разработка веб-приложений на Python / Джефф Форсье, Пол Биссекс,
6. Руководство по веб-фреймворку Django. – www.metanit.com/python/django/;
7. Документация Django на русском. – www.djbook.ru.
8. Python. – www.python.org;
9. Сравнение HTML и XHTML. – www.internet-technologies.ru/articles/sravnenie-html-i-xhtml.html.
10. Справочник HTML. – www.webref.ru/html.
11. Справочник CSS. – www.webref.ru/css;
12. JavaScript Учебные материалы. – www.developer.mozilla.org/ru/docs/Web/JavaScript;
13. jQuery.ajax(). – www.jquery-docs.ru/jQuery.ajax/;
14. AJAX. – www.developer.mozilla.org/ru/docs/Web/Guide/AJAX;

ПРИЛОЖЕНИЕ А

Пагинация

Листинг 1 – Код файла pagination.js

```
function ajaxPagination()
{
    $('#pagination a.page-link').each((index,el) => {
        $(el).click((e) => {
            e.preventDefault();
            let page_url = $(el).attr('href');

            $.ajax({
                url: page_url,
                type: 'GET',
                success: (data) => {
                    $("#main_pag").empty();

$("#main_pag").html($(data).find("#main_pag").html());
                    $("#pagination").empty();
                    $("#pagination").html($(data).find("#pagi-
nation").html());
                }
            });
        });
    });
}

function currentPage()
{
    path = decodeURI(window.location.pathname).toLowerCase();
    if (path.includes("главная") || path=="/")
    {
        var elem = document.querySelector(".razborin")
        elem.classList.add("active-page");
    }
    if (path.includes("история-цен"))
    {
        $('.main-menu__item-link').each((index,el) => {
            if (el.innerHTML === "история цен")
            {
                el.classList.add("active-page");
                $('.history').each((index,el) => {
                   if (path.includes(el.innerHTML))
```

```
                    {
                        el.classList.add("active-page");
                    }
                })
            }
        })
    }
    else
    {
        if (path.includes("продажи"))
        {
            $('.main-menu__item-link').each((index,el) => {
if (el.innerHTML === "продажи")
                {
                    el.classList.add("active-page");
                    $('.sell').each((index,el) => {
                        if (path.includes(el.innerHTML))
                         {
                            el.classList.add("active-page");
                         }
                    })
                }
            })
        }
        else
        {
            $('.main-menu__item-link').each((index,el) => {
                if ("/"+el.innerHTML+"/" === path)
                    el.classList.add("active-page");
            })
        }
    }
};

$(document).ready(function() {
    ajaxPagination();
    currentPage();
});

$(document).ajaxStop(function() {
    ajaxPagination();
});
```

ПРИЛОЖЕНИЕ Б

Продажа

Листинг 2 – Код файла sell.js

```
$(document).ready( function()
{
    $( "#sell" ).submit(function ajaxSell(e)
    {
       e.preventDefault();
       amount = $('#sell-amount').val();
       am = document.getElementById('amount').textContent;
       am = am.replace(' шт.', '');
       if(amount<=am)
       {
           if (Number(amount)<=0)
           {
               alert("Продажа не может быть совершена, количе-
ство для продажи меньше 1!");
           }
           else
           {
               var data = {};
               var csrf_token = $('#sell
[name="csrfmiddlewaretoken"]').val();
               url=$("#sell").attr("action");
               path = decodeURI(window.location.pathname);
               prod_id = path.replace(/[^0-9]/g, '');
               price = document.getEle-
mentById('price').textContent;
               name = document.getElementById('name').textCon-
tent;
               price = parseFloat(price.replace(' руб.', ''));
               let page_url = url+"Sell";
               var data = {'price':price, 'amount':amount,
'csrfmiddlewaretoken':csrf_token};
               var conf = confirm("Вы действительно хотите со-
вершить продажу товара " + name + " по цене " + price + " руб.
в количестве " + amount + " шт. ?");
               if(conf)
               {
                   $.ajax({
                      type: "POST",
                      url: page_url,
```

```
                        data: data,
                        success: function(response)
                        {
                           if(response == "OK")
                           {
                              alert("ИНФОРМАЦИЯ О ПРОДАЖЕ:
\nНазвание: "+ name + "\nЦена: " + price + " руб. \nКоличе-
ство: " + amount + " шт. \nСтраница с полным списком товаров
будет автоматически перезагружена, фильтры сброшены.");
                              location.reload();
                              if(window.opener && !win-
dow.opener.closed)
                              {
                                  window.opener.location.re-
load();
                              }
                           }
                           else
                           alert("Ошибка в запросе! Ответ сер-
вера: " + response);
                        }
                    });
                }
                else
                {
                    alert("Продажа отменена.");
                }
            }
        }
        else
        {
            alert("Количество для продажи превышает количество,
имеющееся на складе!");
        }
    });
});
```

ПРИЛОЖЕНИЕ В

Фильтрация

Листинг 3 – Код файла tyre_filter.js

```
$(document).ready(function()
{
    $( "#filter_form" ).submit(function ajaxFilter(e)
    {
        e.preventDefault();
        url=$(location).attr('href');
        f_name =  $('#filter_name').val();
        f_maker =  $('#filter_maker').val();
        f_model =  $('#filter_model').val();
        f_season = $('#filter_season').val();
        f_gte =  $('#filter_price_gte').val();
        f_lte = $('#filter_price_lte').val();
        f_stock = $('#filter_stock').is(':checked');
        order = $('#order').val();
        let page_url =
url+"?name="+f_name+"&maker="+f_maker+"&model="+f_model+"&sea-
son="+f_sea-
son+"&gte="+f_gte+"&lte="+f_lte+"&stock="+f_stock+"&or-
der="+order;

        $.ajax({
            url: page_url,
            type: 'GET',
            success: (data) => {
                $("#main_pag").empty();

$("#main_pag").html($(data).find("#main_pag").html());
                $("#pagination").empty();
                $("#pagination").html($(data).find("#pagina-
tion").html());
            }
        });
    });
});
```

ПРИЛОЖЕНИЕ Г

Собственный фильтр

Листинг 4 – Код файла filters_extras.py

```
from django import template

register = template.Library()


@register.filter(name='usergroup')
def usergroup(user, group_name):
    return user.groups.filter(name=group_name).exists()
```

ПРИЛОЖЕНИЕ Д

Роутинг приложения

Листинг 5 – Код файла diplom.urls.py

```
from django.urls import path
from .views import *

urlpatterns = [
    path('', index),
    path('Склад/<int:id>/', get_warehouse, name='get_ware-
house'),
    path('Донор/<int:id>/', get_donor, name='get_donor'),
    path('История-цен/Шины/<int:id>/', get_tyre_history,
name='get_tyre_history'),
    path('История-цен/Диски/<int:id>/', get_disk_history,
name='get_disk_history'),
    path('История-цен/Запчасти/<int:id>/', get_spare_history,
name='get_spare_history'),
    path('История-цен/Комплекты/<int:id>/', get_set_history,
name='get_set_history'),
    path('Шины/<int:id>/Sell', tyre_sell_post,
name='tyre_sell'),
    path('Диски/<int:id>/Sell', disk_sell_post,
name='disk_sell'),
    path('Запчасти/<int:id>/Sell', spare_sell_post,
name='spare_sell'),
    path('Комплекты/<int:id>/Sell', set_sell_post,
name='set_sell'),
    path('Шины/<int:id>/', get_tyre, name='get_tyre'),
    path('Диски/<int:id>/', get_disk, name='get_disk'),
    path('Запчасти/<int:id>/', get_spare, name='get_spare'),
    path('Комплекты/<int:id>/', get_set, name='get_set'),
    path('История-цен/Запчасти/', SparePriceHistory-
ListView.as_view(), name='sparesPriceHistory'),
    path('История-цен/Шины/', TyrePriceHistory-
ListView.as_view(), name='tyresPriceHistory'),
    path('История-цен/Диски/', DiskPriceHistory-
ListView.as_view(), name='disksPriceHistory'),
    path('История-цен/Комплекты/', SetPriceHistory-
ListView.as_view(), name='setsPriceHistory'),
    path('Продажи/Запчасти/', SpareSellListView.as_view(),
name='sparesSell'),
    path('Продажи/Шины/', TyreSellListView.as_view(),
```

```
name='tyresSell'),
    path('Продажи/Диски/', DiskSellListView.as_view(),
name='disksSell'),
    path('Продажи/Комплекты/', SetSellListView.as_view(),
name='setsSell'),
    path('Запчасти/', SpareListView.as_view(), name='spares'),
    path('Шины/', TyreListView.as_view(), name='tyres'),
    path('Диски/', DiskListView.as_view(), name='disks'),
    path('Комплекты/', SetListView.as_view(), name='sets'),
    path('Главная/', index, name='index'),
]
```

ПРИЛОЖЕНИЕ Е

Модели

Листинг 6 – Код файла models.py

```
class Tyre(models.Model):
    name = models.CharField(max_length=50, ver-
bose_name='Название')
    price = models.DecimalField(max_digits=9, deci-
mal_places=2, verbose_name='Цена')
    caliber = models.ForeignKey(Caliber, on_delete=models.PRO-
TECT, verbose_name='Посадочный диаметр')
    season = models.ForeignKey(Season, on_delete=models.PRO-
TECT, verbose_name='Сезон')
    tyreMaker = models.ForeignKey(TyreMaker, on_delete=mod-
els.PROTECT, verbose_name='Производитель')
    model = models.CharField(max_length=20, ver-
bose_name='Модель')
    width = models.DecimalField(max_digits=5, deci-
mal_places=2, verbose_name='Ширина')
    deport = models.CharField(max_length=20, ver-
bose_name='Вынос')
    wearOut = models.CharField(max_length=50, ver-
bose_name='Износ')
    centerHole = models.ForeignKey(CenterHole, on_delete=mod-
els.PROTECT, verbose_name='Центральное отверстие')
    warehouse = models.ForeignKey(Warehouse, on_delete=mod-
els.PROTECT, verbose_name='Склад')
    date = models.DateField(verbose_name='Дата')
    comment = models.CharField(max_length=100, ver-
bose_name='Комментарий')
    photo = models.ImageField(verbose_name='Фото')
    amount = models.IntegerField(verbose_name='Количество',
default=1)

    class Meta:
        verbose_name = 'Шина'
        verbose_name_plural = 'Шины'

    def __str__(self):
        return self.name

    def save(self, *args, **kwargs):
```

```
        if not self._state.adding:            super(Tyre,
self).save(*args, **kwargs)
        else:
            super(Tyre, self).save(*args, **kwargs)
            a_record = TyrePriceHistory(tyre=self,
price=self.price, date=datetime.datetime.now(),
                                        user=current_re-
quest().user)
            a_record.save()


@receiver(models.signals.pre_save, sender=Tyre)
def prepare_save(sender, instance, **kwargs):
    try:
        current_instance = sender.objects.get(pk=instance.pk)

        if current_instance.price != instance.price:
            a_record = TyrePriceHistory(tyre=instance,
price=instance.price, date=datetime.datetime.now(),
                                    user=current_re-
quest().user)
        a_record.save()
except sender.DoesNotExist:
    pass
```

ПРИЛОЖЕНИЕ Ж

Настройка админ-панели

Листинг 7 – Код файла admin.py

```
class TyreSellAdmin(admin.ModelAdmin):
    list_display = ('id', 'tyre', 'price', 'amount', 'date',
'user')
    list_display_links = ('id', 'tyre')
    search_fields = ('id', 'tyre__name', 'amount', 'date',
'user')

admin.site.register(Tyre, TyreAdmin)
```

ПРИЛОЖЕНИЕ 3

Получение текущего пользователя

Листинг 8 – Код файла get_username.py

```
from threading import current_thread

from django.utils.deprecation import MiddlewareMixin


_requests = {}


def current_request():
    return _requests.get(current_thread().ident, None)


class RequestMiddleware(MiddlewareMixin):

    def process_request(self, request):
        _requests[current_thread().ident] = request

    def process_response(self, request, response):
        _requests.pop(current_thread().ident, None)
        return response

    def process_exception(self, request, exception):
        _requests.pop(current_thread().ident, None)
        raise exception
```

ПРИЛОЖЕНИЕ И

Представления

Листинг 9 – Код файла views.py

Листинг 9.1 – Код представления TyreListView

```
class TyreListView(ListView):
    model = Tyre

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)

        if (not self.request.GET.get('order')) or self.re-
quest.GET.get('order') == "" or self.request.GET.get(
                'order') is None:
            order_query = "-id"
        else:
            order_query = self.request.GET.get('order')
        context['order'] = order_query

        filter_query_stock = self.request.GET.get('stock')
        context['filter_stock'] = filter_query_stock

        if not self.request.GET.get('name'):
            filter_query_name = ""
        else:
            filter_query_name = self.request.GET.get('name')
        context['filter_name'] = filter_query_name

        if not self.request.GET.get('maker'):
            filter_query_maker = ""
        else:
            filter_query_maker = self.request.GET.get('maker')
        context['filter_maker'] = filter_query_maker

        if not self.request.GET.get('model'):
            filter_query_model = ""
        else:
            filter_query_model = self.request.GET.get('model')
        context['filter_model'] = filter_query_model

        if (not self.request.GET.get('gte')) or (self.re-
quest.GET.get('gte') is None) or (
                self.request.GET.get('gte') == ""):
```

```
            filter_query_gte = 0
        else:
            filter_query_gte = self.request.GET.get('gte')

        if (not self.request.GET.get('lte')) or (self.re-
quest.GET.get('lte') is None) or (
                self.request.GET.get('lte') == ""):
            filter_query_lte = 1000000
        else:
    filter_query_lte = self.request.GET.get('lte')

context['filter_price_gte'] = filter_query_gte
context['filter_price_lte'] = filter_query_lte

filter_query_season = self.request.GET.get('season')
if filter_query_season is None or filter_query_season ==
"Любой":
    filter_query_season = ""
context['filter_season'] = filter_query_season

if filter_query_stock == "true":
    context['object_list'] = Tyre.objects.filter(name__icon-
tains=filter_query_name,

tyreMaker__name__icontains=filter_query_maker,
                                                 model__icon-
tains=filter_query_model,
                                                 sea-
son__name__icontains=filter_query_season,

price__gte=filter_query_gte,

price__lte=filter_query_lte,

amount__gte=1).order_by(order_query)
else:
    context['object_list'] = Tyre.objects.filter(name__icon-
tains=filter_query_name,

tyreMaker__name__icontains=filter_query_maker,
                                                 model__icon-
tains=filter_query_model,
```

```
season__name__icontains=filter_query_season,

price__gte=filter_query_gte,

price__lte=filter_query_lte).order_by(order_query)
paginator = Paginator(context['object_list'], 5)
page = self.request.GET.get('page')
try:
    context['object_list'] = paginator.page(page)
except PageNotAnInteger:
    context['object_list'] = paginator.page(1)
except EmptyPage:
    context['object_list'] = paginator.page(pagina-
tor.num_pages)
return context
```

Листинг 9.2 – Код представления get_tyre

```
def get_tyre(request, id):
    context = {
        "tyre": Tyre.objects.filter(id=id)
    }
    return render(request, 'get-item/get-tyre.html', context)
```

Листинг 9.3 – Код представления get_tyre_history

```
def get_tyre_history(request, id):
    contact_list = TyrePriceHistory.objects.filter(tyre=id)
    paginator = Paginator(contact_list, 15)
    page_number = request.GET.get('page')
    page_obj = paginator.get_page(page_number)
    context = {
        "page_obj": page_obj,
        "tyreHistory": TyrePriceHistory.objects.fil-
ter(tyre=id)
    }
    return render(request, 'history/tyres-history.html', con-
text)
```

ПРИЛОЖЕНИЕ К

Создание клавиатуры

Листинг 10 – Код создания клавиатуры

```
from telebot import types


def create_keyboard_main(r, ch):
    keyboard = types.InlineKeyboardMarkup(row_width=r)
    buttons = [types.InlineKeyboardButton(text=c,
callback_data=c) for c in ch]
    keyboard.add(*buttons)
    return keyboard
```

ПРИЛОЖЕНИЕ Л

Добавление запчасти

Листинг 11 – Код добавления запчасти через телеграм-бот

```
def spare_sparename(message):
    s = message.text.replace('/', '')
    s = int(s)
    SpareData[message.from_user.id] = Spare(s)
    sent = bot.send_message(message.chat.id, 'Введите цену за-
пчасти (если число десятичное, то используйте точку '
                                             'для отделения
целой части, например, 200.45; также обратите '
                                             'внимание на то,
что цена округлится до двух знаков после точки '
                                             'в большую сто-
рону):')
    bot.register_next_step_handler(sent, spare_price)


def spare_price(message):
    if message.text == '/anew':
        update_state(message, START)
        f = eval('start')(message)
    else:
        try:
            float(message.text)
        except ValueError:
            sent = bot.send_message(message.chat.id, 'Цена
введена неправильно! Пожалуйста, введите цену заново:')
            bot.register_next_step_handler(sent, spare_price)
        else:
            spare = SpareData[message.from_user.id]
            spare.price = Decimal(message.text)
            spare.price = spare.price.quantize(Deci-
mal("1.00"), ROUND_HALF_DOWN)
            check_auto(message)


def spare_auto(message):
    s = message.text.replace('/', '')
    s = int(s)
    spare = SpareData[message.from_user.id]
    spare.donor = s
```

```
    sent = bot.send_message(message.chat.id, 'Введите год
выпуска запчасти:')
    bot.register_next_step_handler(sent, spare_yom)


def spare_yom(message):
    if message.text == '/anew':
        update_state(message, START)
        f = eval('start')(message)
    else:
        try:
            int(message.text)
        except ValueError:
            sent = bot.send_message(message.chat.id,
                                    'Год введен неправильно!
Пожалуйста, введите его заново:')
            bot.register_next_step_handler(sent, spare_yom)
        else:
            if 1900 < int(message.text) <
int(datetime.date.today().year):
                spare = SpareData[message.from_user.id]
                spare.yom = message.text
                check(message, 'maker', 'id', 'name',
'производителей запчастей', choice_maker, 'spare')
            else:
                sent = bot.send_message(message.chat.id,
                                        'Год введен
неправильно! Пожалуйста, введите его заново:')
                bot.register_next_step_handler(sent,
spare_yom)


def spare_maker(message):
    s = message.text.replace('/', '')
    s = int(s)
    spare = SpareData[message.from_user.id]
    spare.maker = s
    sent = bot.send_message(message.chat.id, 'Введите
маркировку запчасти:')
    bot.register_next_step_handler(sent, spare_marking)


def spare_marking(message):
```

```
    if message.text.find('/') == 0:
        if message.text == '/anew':
            update_state(message, START)
            f = eval('start')(message)
        else:
            sent = bot.send_message(message.chat.id, 'Значение
содержит команду! Пожалуйста, введите его еще раз без '
                                                    'символа
"/" в начале:')
            bot.register_next_step_handler(sent, spare_mark-
ing)
    else:
        spare = SpareData[message.from_user.id]
        spare.marking = message.text
        check(message, 'condition', 'id', 'type', 'состояния',
choice_condition, 'spare')

def spare_condition(message):
    s = message.text.replace('/', '')
    s = int(s)
    spare = SpareData[message.from_user.id]
    spare.condition = s
    check(message, 'spacing', 'id', 'place', 'расположения',
choice_spacing, 'spare')
def spare_spacing(message):
    s = message.text.replace('/', '')
    s = int(s)
    spare = SpareData[message.from_user.id]
    spare.spacing = s
    sent = bot.send_message(message.chat.id, 'Введите цвет
запчасти:')
    bot.register_next_step_handler(sent, spare_color)


def spare_color(message):
    if message.text.find('/') == 0:
        if message.text == '/anew':
            update_state(message, START)
            f = eval('start')(message)
        else:
            sent = bot.send_message(message.chat.id, 'Значение
содержит команду! Пожалуйста, введите его еще раз без '
                                    'символа "/" в начале:')
```

```
            bot.register_next_step_handler(sent, spare_color)
    else:
        spare = SpareData[message.from_user.id]
        spare.color = message.text
    check(message, 'warehouse', 'id', 'name', 'склада',
choice_warehouse, 'spare')


def spare_warehouse(message):
    s = message.text.replace('/', '')
    s = int(s)
    spare = SpareData[message.from_user.id]
    spare.warehouse = s
    sent = bot.send_message(message.chat.id, 'Введите дату в
формате год-месяц-день (пример: 2021-01-01):')
    bot.register_next_step_handler(sent, spare_date)


def spare_date(message):
    if message.text == '/anew':
        update_state(message, START)
        f = eval('start')(message)
    else:
        try:
            datetime.datetime.strptime(message.text, "%Y-%m-
%d")
        except ValueError:
            sent = bot.send_message(message.chat.id,
                                    'Дата введена неправильно!
Пожалуйста, введите её заново:')
            bot.register_next_step_handler(sent, spare_date)
        else:
            spare = SpareData[message.from_user.id]
            spare.date = datetime.datetime.strptime(mes-
sage.text, "%Y-%m-%d")
            sent = bot.send_message(message.chat.id, 'Введите
комментарий:')
            bot.register_next_step_handler(sent, spare_com-
ment)


def spare_comment(message):
    if message.text.find('/') == 0:
```

```
        if message.text == '/anew':
            update_state(message, START)
            f = eval('start')(message)
        else:
            sent = bot.send_message(message.chat.id, 'Значение
содержит команду! Пожалуйста, введите его еще раз без '
                                                    'символа
"/" в начале:')
            bot.register_next_step_handler(sent, spare_com-
ment)
    else:
        spare = SpareData[message.from_user.id]
        spare.comment = message.text
        sent = bot.send_message(message.chat.id, 'Введите
количество:')
        bot.register_next_step_handler(sent, spare_amount)


def spare_amount(message):
    if message.text == '/anew':
        update_state(message, START)
        f = eval('start')(message)
    else:
        try:
            int(message.text)
        except ValueError:
            sent = bot.send_message(message.chat.id,
                                    'Количество введено
неправильно! Пожалуйста, введите его заново:')
            bot.register_next_step_handler(sent, spare_amount)
        else:
            spare = SpareData[message.from_user.id]
            spare.amount = int(message.text)
            sent = bot.send_message(message.chat.id,
'Загрузите фото:')
            bot.register_next_step_handler(sent, spare_photo)


def spare_photo(message):
    if message.text == '/anew':
        update_state(message, START)
        f = eval('start')(message)
    else:
```

```
        try:
            bot.get_file(message.photo[len(message.photo) -
1].file_id)
        except TypeError:
            sent = bot.send_message(message.chat.id,
                                    'Фото не добавлено!
Пожалуйста, добавьте его заново:')
            bot.register_next_step_handler(sent, spare_photo)
        else:
            try:
                file_info = bot.get_file(mes-
sage.photo[len(message.photo) - 1].file_id)
                downloaded_file = bot.down-
load_file(file_info.file_path)
                spare = SpareData[message.from_user.id]
                fpath = file_info.file_path.split('/')
                spare.photo = 'D:/Django/Project/diplom/me-
dia/' + fpath[1]
                with open(spare.photo, 'wb') as new_file:
                    new_file.write(downloaded_file)
                sql = "INSERT INTO autorazbor_spare (name_id,
price, donor_id, yom, maker_id, marking, condition_id, spac-
ing_id, " \
                      "color, warehouse_id, date, comment,
photo, amount) VALUES (%s, %s, " \
                      "%s, %s, %s, %s, %s, %s, %s, %s, %s, %s,
%s, %s) "
                values = (spare.name, spare.price, spare.do-
nor, spare.yom,
                          spare.maker, spare.marking,
spare.condition, spare.spacing, spare.color,
                          spare.warehouse, spare.date,
spare.comment, spare.photo, spare.amount)
                cursor.execute(sql, values)
                db.commit()
                sent = bot.send_message(message.chat.id,
                                        'Запчасть успешно
добавлена! Вы можете начать сначала, используя команду
/start')
                cursor.execute(
                    "SELECT id FROM autorazbor_spare WHERE
name_id ='" + str(spare.name) + "'")
                search_result = cursor.fetchall()
```

```
                for row in search_result:
                    id = row['id']
                sql = "INSERT INTO autorazbor_sparepricehis-
tory (price, date, spare_id, user_id) VALUES " \
                      "(%s, %s, %s, %s) "
                values = (spare.price,
datetime.datetime.now().strftime('%Y/%m/%d %H:%M'), id, '2')
                cursor.execute(sql, values)
                db.commit()
            except Exception as e:
                sent = bot.send_message(message.chat.id,
'Произошел сбой! Попробуйте сначала! Ошибка: ' + e.__str__())
                bot.register_next_step_handler(sent,
spare_photo)
```