

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра Вычислительной техники
(полное название кафедры)

Утверждаю

Зав. кафедрой Якименко А.А.

(подпись, инициалы, фамилия)

«__» _____ 2021 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Бердников Константин Сергеевич

(фамилия, имя, отчество студента – автора работы)

Web-приложение для автоматизации заказов интернет-магазина

(тема работы)

Факультет автоматики и вычислительной техники

(полное название факультета)

Направление подготовки 09.03.01 Информатика и вычислительная техника

(код и наименование направления подготовки бакалавра)

**Руководитель
от НГТУ**

Дубков Илья Сергеевич

(фамилия, имя, отчество)

Старший преподаватель

(ученая степень, ученое звание)

(подпись, дата)

**Автор выпускной
квалификационной работы**

Бердников Константин Сергеевич

(фамилия, имя, отчество)

АВТФ, АВТ-710

(факультет, группа)

(подпись, дата)

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра Вычислительной техники
(полное название кафедры)

УТВЕРЖДАЮ

Зав. кафедрой Якименко А.А.
(фамилия, имя, отчество)

(подпись, дата)

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА**

студенту Бердникову Константин Сергеевичу
(фамилия, имя, отчество)

Направление подготовки 09.03.01 Информатика и вычислительная техника
(код и наименование направления подготовки бакалавра)

Факультет автоматики и вычислительной техники
(полное название факультета)

Тема Web-приложение для автоматизации заказов интернет-магазина
(полное название темы выпускной квалификационной работы бакалавра)

Цель работы: разработать интернет-магазин, который специализируется на
продаже книг.

Структурные части работы _____

1. Аннотация
2. Содержание
3. Введение
4. Постановка задачи

5. Обзор аналогов приложений для автоматизации заказов

6. Выбор архитектуры (структуры) приложения.

7. Инфологическая модель базы данных

8. Создание серверной части

9. Создание клиентской части

10. Заключение

11. Список использованных источников

12. Приложение

Задание согласовано и принято к исполнению.

**Руководитель
от НГТУ**

Дубков Илья Сергеевич

(фамилия, имя, отчество)

Старший преподаватель

(ученая степень, ученое звание)

(подпись, дата)

Старший преподаватель

Студент

Бердников Константин Сергеевич

(фамилия, имя, отчество)

АВТФ, АВТ-710

(факультет, группа)

(подпись, дата)

Тема утверждена приказом по НГТУ № 737/2 от «25» февраля 2021 г.

ВКР сдана в ГЭК № _____, тема сверена с данными приказа

(подпись секретаря государственной экзаменационной комиссии по защите ВКР, дата)

(фамилия, имя, отчество секретаря государственной
экзаменационной комиссии по защите ВКР)

АННОТАЦИЯ

Пояснительная записка к бакалаврской работе Бердникова Константина Сергеевича «Web-приложение для автоматизации заказов интернет-магазина» представлена на 74 страницах. Пояснительная записка включает 23 рисунков, 13 таблиц, 20 листингов и 3 приложения. Список литературы содержит 14 библиографических источников.

Целью бакалаврской работы является разработка интернет-магазина, который специализируется на продаже книг.

В процессе выполнения работы были выбраны средства разработки, рассмотрены аналоги веб-приложений, создана серверная и клиентская часть.

В результате выполнения работы на языке программирования Ruby, было разработано веб-приложение по продаже книг.

СОДЕРЖАНИЕ

СПИСОК ИСПОЛЬЗУЕМЫХ ТЕРМИНОВ	7
ВВЕДЕНИЕ	8
1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ	9
2 ВЫБОР АРХИТЕКТУРЫ (СТРУКТУРЫ) ПРИЛОЖЕНИЯ	11
3 ИНФОЛОГИЧЕСКАЯ МОДЕЛЬ БАЗЫ ДАННЫХ	13
4 ОБОСНОВАНИЕ ВЫБОРА СРЕДСТВ ПРОГРАММНОЙ РАЗРАБОТКИ	15
5 ФОРМИРОВАНИЕ ЦЕЛЕЙ И ЗАДАЧ	17
6 СОЗДАНИЕ СЕРВЕРНОЙ ЧАСТИ	18
6.1 Генерация приложения	18
6.2 Создание контроллера статических страниц	20
6.3 Создание авторизации пользователя	21
6.4 Создание сущности “Каталог”	23
6.5 Создание сущности “Категория”	24
6.6 Создание сущности “Продукт”	25
6.7 Создание сущности “Корзина”	27
6.8 Создание сущности “Продукт в корзине”	28
6.9 Создание сущности “Заказ”	31
6.10 Создание сущности “Комментарий”	32
6.11 Создание сущности “Хранимые файлы”	34
6.12 Регистрация просмотров страниц	35
6.13 Распределение прав доступа	36
7 СОЗДАНИЕ КЛИЕНТСКОЙ ЧАСТИ	38
7.1 Шапка сайта	38
7.2 Блок продукта	39
7.3 Главная страница	40

7.4 Страница создания каталога	41
7.5 Страница каталога.....	42
7.6 Страница создания категории.....	43
7.7 Страница категории	43
7.8 Страница создания продукта	44
7.9 Страница продукта.....	45
7.10 Страница корзины	46
7.11 Страница оформления заказа	47
7.12 Страница оформленного заказа	48
7.13 Страница авторизации	48
7.14 Страница регистрации	49
ЗАКЛЮЧЕНИЕ	50
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	51
ПРИЛОЖЕНИЕ	52

СПИСОК ИСПОЛЬЗУЕМЫХ ТЕРМИНОВ

Framework (фреймворк) — программная платформа, определяющая структуру программной системы.

Gem (гем) — система управления пакетами для языка программирования Руби, которая предоставляет стандартный формат для программ и библиотек Руби.

Шаблонизатор — программное обеспечение, позволяющее использовать html-шаблоны для генерации конечных html-страниц;

Генератор - встроенное средство для создания заранее заготовленных шаблонов;

CRUD — акроним, обозначающий четыре базовые функции, используемые при работе с базами данных: создание (англ. create), чтение (read), модификация (update), удаление (delete);

Реактивность — этот декларативный подход предлагает описывать данные в виде набора утверждений или формул;

Масштабируемость — в электронике и информатике означает способность системы, сети или процесса справляться с увеличением рабочей нагрузки (увеличивать свою производительность) при добавлении ресурсов (обычно аппаратных);

Action (экшен) — это независимый пакет или класс, который можно использовать с любой библиотекой Ruby;

Routes (пути) — роутер Rails распознает URL и направляет его в экшн контроллера;

Переадресация — смена URL-адреса текущей веб-страницы;

Рендер (визуализация) — в общем смысле под рендирингом подразумевается перевод объектов, не имеющих визуального представления (структуры в памяти компьютера) в объекты это представление имеющие.

ВВЕДЕНИЕ

В настоящее время интернет очень популярен, поэтому широкое распространение получили веб-приложения. Один из видов веб-приложения – это интернет-магазин, они также стали невероятно популярными, а обусловлено это удобством покупки товара. Можно не выходя из дома делать покупки, ознакомиться с продуктом, конкретнее узнать о его спецификациях. Поэтому разработка интернет-магазинов стала очень актуальной.

Основная цель интернет-магазина заключается в предоставлении возможностей сделать электронную покупку, а для этого существует множество технологий. Некоторые технологии позволяют без знаний программирования создать свой сайт, в основном это конструкторы сайтов, такие как: WordPress, Joomla, MODX, OpenCart. А также существуют множество фреймворков для создания веб-приложений, например: Rails, Django, Laravel. Для выпускной квалификационной работы был выбран фреймворк Rails.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Web-приложение – это клиент-серверное приложение, взаимодействие клиента происходит через браузер. Преимуществом данного приложения является доступность, для доступа необходим лишь интернет, а браузер доступен в множестве операционных систем. А также приложения не нуждается в оптимизации под различные устройства.

Интернет-магазин – это непосредственно один из видов web-приложения. Клиенты смогут в режиме реального времени делать заказы, но оформление заказа происходит дистанционным способом, что позволяет не контактировать с персоналом магазина, что является актуальным во время пандемии.

Целью интернет-магазина, является минимизация затрат и получение максимальной прибыли. Интернет-магазины зарабатывают не на продаже рекламных мест на сайте, а стараются обеспечивать стабильный приток клиентов, приобретающих товары.

Интернет-магазин в дипломной работе будет ориентироваться на продажу книг.

Для оформления заказа будет необходимо выбрать понравившуюся книгу или несколько из каталога, добавить в корзину, а после оформить его, заполнив несколько форм.

В настоящее время на замену книг пытаются прийти аудиокниги, электронные книги, но тем не менее книги до сих пор являются актуальными. Поэтому интернет-магазинов по продажам книг много.

Для сравнения были выбраны следующие интернет-магазины:

- Chitai-gorod.ru;
- Eksmo.ru;
- Bookvoed.ru.

Все три магазина объединяет то, что для каждой книги имеется система рейтингов. Можно оставить свой отзыв о книге. Имеется строка поиска. А также предлагаются похожие книги, которые могут подойти читателю.

Интернет-магазин Chitai-gorod.ru – самая большая сеть книжных магазинов в России. Представляет краткое описание книги. Для каждой книги имеется несколько фотографий. Можно проверить в каком из магазинов книга в наличии.

Сайт Eksmo.ru – это лучшее российское издание книг, а также один из лидеров Европы. Магазин предлагает не только бумажное издание книг, но также есть возможность купить электронную версию книги. Страницы книжного-магазина перегружены и имеют множество ненужной информации для рядового покупателя, которая занимает достаточно места на экране. Но плюсом является то, что можно прочитать фрагмент книги онлайн.

Bookvoed.ru -крупнейшая сеть магазинов в Санкт-Петербурге. Имеет лаконичный дизайн, но совсем не интуитивную навигацию по каталогу книг. Относительно других сайтов, то здесь отзывы об книгах не спрятаны за кучей предложений других книг. Но как и на chita-gorod.ru мы не можем прочитать фрагмент книги.

2 ВЫБОР АРХИТЕКТУРЫ (СТРУКТУРЫ) ПРИЛОЖЕНИЯ

Веб-приложение всегда состоит из двух частей. Первая – это серверная часть (backend). Вторая – это клиентская часть (frontend).

Клиентскую часть иногда называют интерфейсом. Это визуальная часть, с которой взаимодействует клиент, взаимодействие происходит посредством использования браузера. Также клиентская часть отправляет HTTP-запросы на сервер и получает ответ.

Серверную часть клиент не видит. Она занимается бизнес-логикой и отвечает на отправленные запросы с клиентской части. Например, заполненную форму регистрации обрабатывает серверная часть и она хранит все данные о пользователе. А клиентская часть только отправляет запрос с указанием данных пользователя.

Существует несколько типов веб-приложений:

- одностраничные веб-приложения;
- многостраничные веб-приложения;
- микросервисы;
- прогрессивные веб-приложения;

Существует несколько типов веб-приложений:

Данное приложение будет многостраничным, при переходе по ссылкам страница будет перезагружаться. А информация будет генерироваться на сервере.

Код будет организован с помощью MVC паттерна. MVC можно расшифровать как модель-представление-контроллер (от англ. Model-view-controller).

Каждый компонент отвечает за следующие задачи:

- модель – отвечает за работу с данными. определяет связи таблиц в базе данных, а также их валидацию;
- представление – компонент отвечает за работу с пользователем, то, как будет клиент пользоваться сайтом, а также его внешний вид;

– контроллер – связывает модель и представление. принимает запросы с клиента и перенаправляет к нужной модели, а далее уже сгенерированное представление с нужными данными возвращается клиенту.

Схема структуры MVC-паттерна имеет внешний вид, изображенный на рисунке 2.1.

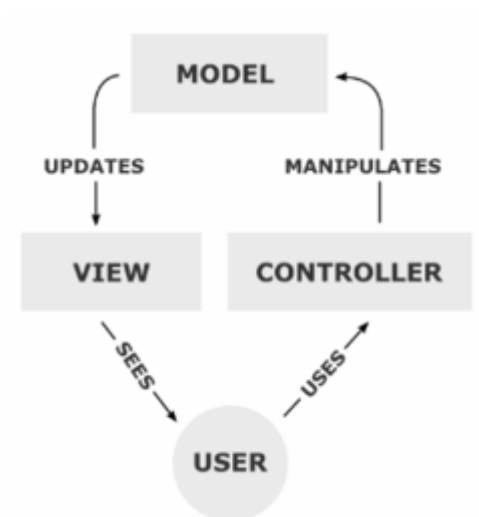


Рисунок 2.1 — Схема MVC паттерна

3 ИНФОЛОГИЧЕСКАЯ МОДЕЛЬ БАЗЫ ДАННЫХ

Инфологическое моделирование выполняется с целью обеспечения естественных для человека способов представления и сбора информации, которая будет храниться в создаваемой БД.

Инфологическая модель данных строится в соответствии с естественным языком, который невозможно использовать в чистом виде в виду сложности обработки текстов с помощью компьютера и неоднозначности естественного языка.

Инфологическая модель – это потоки информации, сущности и связи данной области. В такой модели указываются связи между сущностями данной предметной области.

Сущность – это любой объект, отличающийся от другого, информацию о котором необходимо сохранить.

Связь – это ассоциирование нескольких сущностей с целью отыскания одних из них по значениям других.

База данных может содержать неограниченное количество сущностей и такое же количество связей между ними, что определяет сложность инфологических моделей.

Атрибут – это характеристика сущности. Это может быть числовой характеристикой, классификацией, идентификацией. Его наименование должно быть уникальным для конкретного типа сущностей и может совпадать с атрибутами других сущностей.

Диаграмма «Сущность-связь» имеет внешний вид, изображенный на рисунке 3.1.

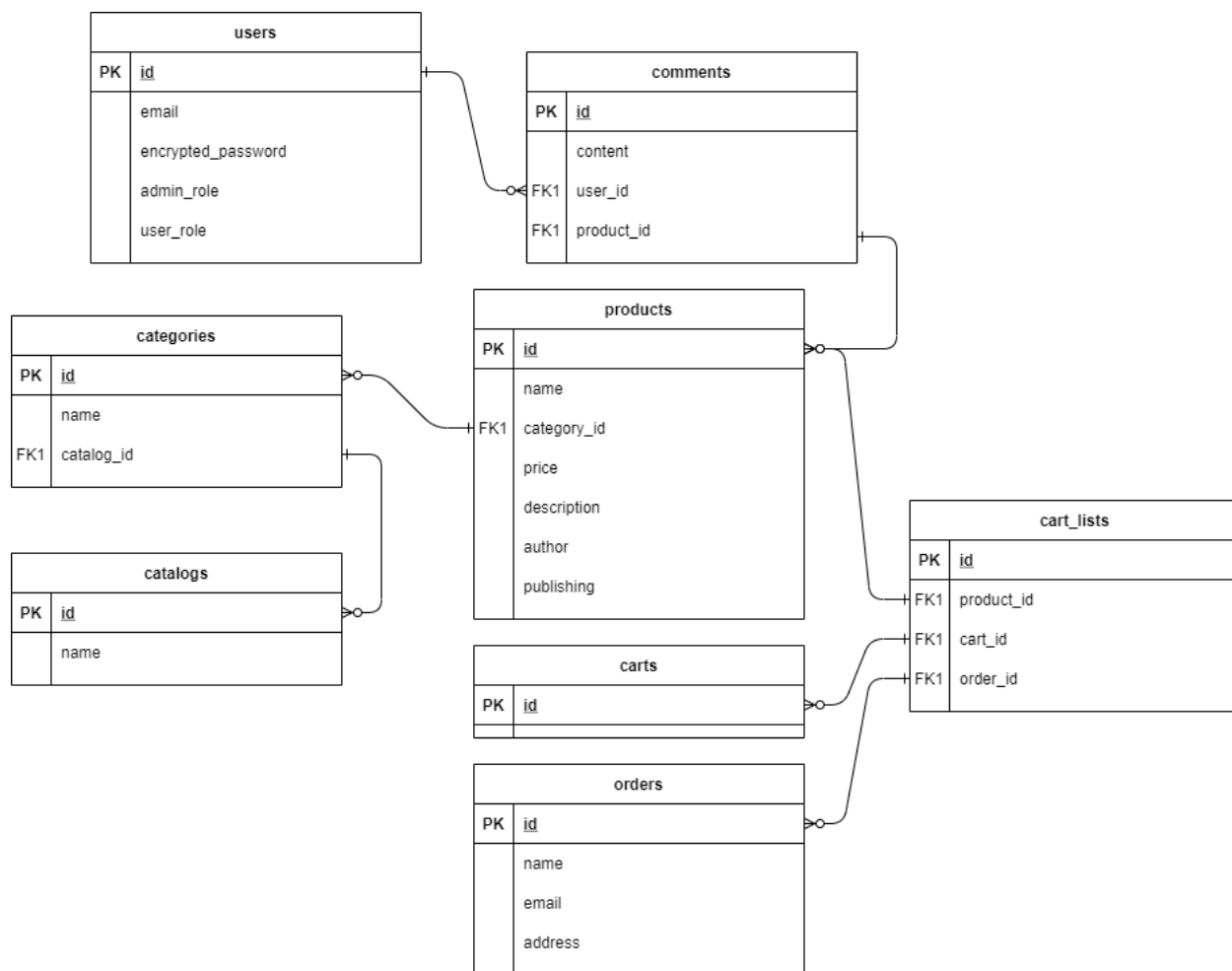


Рисунок 3.1 - Диаграмма «Сущность-связь»

4 ОБОСНОВАНИЕ ВЫБОРА СРЕДСТВ ПРОГРАММНОЙ РАЗРАБОТКИ

Для разработки серверной части будет использоваться фреймворк Ruby on Rails (Rails), который написан на языке Ruby. Основные причины выбора:

- разрабатывать приложения на Rails гораздо эффективнее, чем на других фреймворках. По статистике скорость разработки выше на 20% - 30% относительно других фреймворков. Причиной высокой эффективности является то, что он имеет множество гибких инструментов и удобную модель работы с базами данных.

- rails имеет множество стандартов написания кода, соблюдение которых помогает очень просто поддерживать и расширять проект. Если веб-приложение становится успешным, то новым разработчикам будет проще вникнуть в происходящее.

- фреймворк имеет высокую надежность, в нем много инструментов для тестирования написанного кода, сам же фреймворк также покрыт тестами, что позволяет быть уверенным, что при обновлении на новую версию ничего не сломается. Стабильность – это очень важный фактор для любого приложения.

- rails отлично справляется с масштабируемостью и высокими нагрузками, приложение можно развернуть на нескольких кластерах и разделять нагрузку между ними. А вместе с надежностью это весомый аргумент в сторону выбора данного фреймворка.

Список популярных сайтов, написанных на фреймворке Rails:

- Github;
- AirBnB;
- Twitch;
- Aviasales.

Клиентская часть будет разрабатываться с помощью шаблонного языка Slim, цель которого состоит в том, чтобы свести синтаксис к основным частям и не делая трудночитаемым. А совместно с фреймворком Ruby on Rails он

работает быстрее стандартного шаблонизатора ERB. Написанный код на Slim преобразовывается в HTML. Также будет использоваться препроцессор SCSS, который упрощает написание CSS кода. Для того чтобы клиентская часть была более отзывчивой, будет использоваться JavaScript.

Для «реактивности» приложения будет использоваться фреймворк StimulusJS, который поможет «оживить» приложение. Данная библиотека поможет дополнить HTML-код. А также он отлично работает с библиотекой turbolinks. Сравнивая с другими фреймворками, он является легковесным.

5 ФОРМИРОВАНИЕ ЦЕЛЕЙ И ЗАДАЧ

Целью работы является разработка интернет-магазина, который специализируется на продаже книг.

Реализация цели требует решения следующих задач:

1. Изучить фреймворк Rails;
2. Разработать интернет-магазин с использованием базы данных PostgreSQL;
3. Отображать каталог товаров, популярные товары, а также новые товары;
4. Возможность сортировки товаров по заданным категориям;
5. Реализовать регистрацию пользователя, чтобы была возможность оставить отзывы о товаре.

6 СОЗДАНИЕ СЕРВЕРНОЙ ЧАСТИ

6.1 ГЕНЕРАЦИЯ ПРИЛОЖЕНИЯ

Чтобы приступить к работе, генерируем базовую структуру приложения с помощью следующей команды:

Листинг 6.1.1 — Команда генерации веб-приложения

```
rails new shop-online -d postgresql
```

В команде видно, чтобы управлять базами данных выбрана объектно-ориентированная система управления postgresql.

После генерации, видим следующую структуру, больше всего интересует папка app. В ней будет происходить основная работа.

Данная структура имеет внешний вид, изображенный на рисунке 6.1.1.

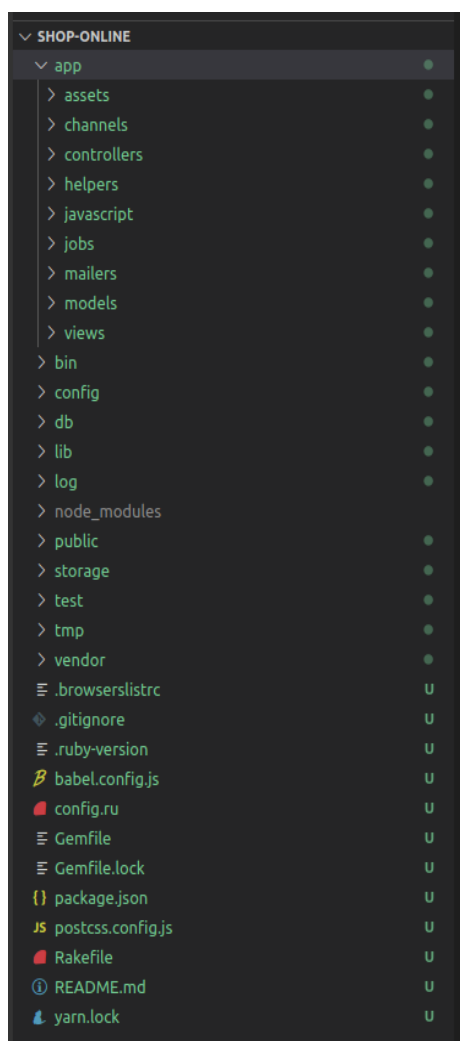


Рисунок 6.1.1 - Результат генерации.

Рассмотрим вложенные папки в папку `app` по назначению:

- `assets` - используется для хранения файлов типа `css`, `js`, `img`. Файлы в данной папке будут сжаты, чтобы конечный пользователь получил оптимизированный код, который занимает меньше места, а в свою очередь имеем более быструю загрузку страницы, что поднимает нас в выдаче поисковых запросов. Также в данной папке можно использовать препроцессоры такие как `scss` или же `CoffeeScript`, которые упрощают написание, а код получается более лаконичным;

- `channels` - папка, которая содержит в себе файлы для работы с каналами. Каналы в `Ruby on Rails` — это способ работы с `WebSockets`. Позволяют писать функции реального времени в том же стиле, что и все приложение rails;

- `controllers` - все действия после маршрутизатора попадают в контроллеры. Он отвечает за понимание запросы и соответствующую выдачу. Контроллер будет получать запросы и взаимодействовать с данными из моделей, а также использовать представление для создания возвращаемого `HTML`. Рассматривать контроллер стоит как посредника между представлением и моделями. Отвечает за бизнес-логику;

- `helpers` - являются помощниками для генерации `HTML` кода, связывает представление с файлами (изображения, `js`);

- `javascripts` - здесь хранятся файлы, которые будут работать с представлением. В данной работе будут храниться файлы с фреймворка `StimulusJS`;

- `jobs` - платформа для работы с фоновыми серверными задачами. Помогает управлять очередью, а также выполняем работу `ruby` кода в асинхронном режиме. Работы могут быть различными, начиная от рассылки сообщений пользователям или же сложными расчетами, результат которых требуются через определенное время;

- `mailers` - здесь хранятся структуры отправляемых сообщений;

- `models` - содержит различные модули, используемые при разработке классов, которые взаимодействуют с базами данных;

– views - представления, которые преобразуются в HTML код. Для представлений будет использоваться препроцессор SLIM.

6.2 СОЗДАНИЕ КОНТРОЛЛЕРА СТАТИЧЕСКИХ СТРАНИЦ

Контроллер статических страниц позволяет возвращать страницы без динамического содержимого, данные страницы будут написаны на чистом HTML. В этом контроллере будут созданы такие страницы как:

- домашняя;
- о нас;
- помощь.

Был создан контроллер с помощью использования генератора. Генераторы Rails — это удобный инструмент, который помогает оптимизировать рабочий процесс. Генераторы упрощают рутину, больше не занимаемся созданием и объявлением классов простых CRUD моделей.

```
rails generate controller StaticPage
```

После этого получены следующие новые файлы:

Листинг 6.2.1 — Отчет о созданных файлах

```
create app/controllers/static_pages_controller.rb # Сам controller
invoke erb
create app/views/static_pages # Представление
invoke test_unit
create test/controllers/static_pages_controller_test.rb # Тест для
контроллера
invoke helper
create app/helpers/static_pages_helper.rb # Помощник
invoke test_unit
invoke assets
invoke scss
create app/assets/stylesheets/static_pages.scss # Scss, преобразу-
ется в CSS
```

Добавим пути в файл маршрутизатор, расположенный в папке с конфигурациями ./config/routes.rb

Листинг 6.2.2 — Роуты статических страниц

```
root 'static_pages#home' # Домашняя страница приложения поме-
чена как root

get 'static_pages#help', as: 'help' # Страница с помощью на сайте
get 'static_pages#about', as: 'about' # Страница содержит информацию
о магазине
```

Теперь следует связать маршруты с действиями в контроллерах.

Листинг 6.2.3 — Методы статического контроллера

```
class ApplicationController < ActionController::Base
  def home; end

  def help; end

  def about; end
end
```

Осталось только добавить представление. В папке `views` будет добавлено три пустых файла, позже они будут заполнены.

- `home.html.slim`;
- `about.html.slim`;
- `help.html.slim`.

6.3 СОЗДАНИЕ АВТОРИЗАЦИИ ПОЛЬЗОВАТЕЛЯ

Для создания сущности пользователя была использована библиотека `'devise'`, это ruby gem, который позволяет сгенерировать аутентификацию в rails-приложениях.

Добавлена библиотека `devise` к списку других гемов. Все гемы располагаются в корне проекта в файле `Gemfile`.

Установлены все представления в приложении, это позволяет в дальнейшем их использовать.

Листинг 6.3.1 — Команда установки представлений devise

```
rails generate devise:install
```

После первичной настройки. В конфигурации среды разработчика укажем локальный адрес откуда будут отправляться сообщения на почту с подтверждением о регистрации.

Листинг 6.3.2 — Конфигурация отправки сообщений на почту

```
config.action_mailer.default_url_options = { host: 'localhost', port: 3000 }
```

Создана сущность пользователя в базе данных. Сущность будет иметь следующий вид.

Таблица 6.3.1 — Сущность «Пользователи» (users)

Название поля	Тип данных	Описание
id	bigint	Идентификатор пользователя
email	string	Почтовый ящик пользователя
encrypted_password	string	Пароль в зашифрованном виде
admin role	boolean	Флаг, является ли пользователь администратором
user_role	boolean	Флаг, имеет ли пользователь права пользователя. Если значение false, учетная запись заблокирована
created_at	datetime	Время создания записи
updated_at	datetime	Время последнего обновления записи

Также провалидированы данные, которые будут записываться в базу данных. Были стандартизуемы данные почтовых ящиков перед сохранением, данные сохраняются в нижнем регистре, а также добавлена проверка, что был введен именно почтовый ящик с помощью регулярного выражения.

Листинг 6.3.3 — Валидация сущности пользователя

```
before_save { email.downcase! }

VALID_EMAIL_REGEX = /\A[\w+\-\.]+\@[a-z\-\.\.][a-z]+\z/i

validates :email, presence: true, length: { maximum: 255 },
              format: { with: VALID_EMAIL_REGEX }
```

Контроллеры были сгенерированы автоматически с помощью библиотеки devise.

6.4 СОЗДАНИЕ СУЩНОСТИ “КАТАЛОГ”

Создана модель “Каталог”, которая необходима для того, чтобы хранить разделы товаров на сайте. Например книги, канцтовары, творчество и т.д. Также свяжем с категориями.

Листинг 6.4.1 — Связи модели каталога

```
has_many :categories, dependent: :destroy # Каталог может иметь множество
категорий, после удаления каталога, также будет удалена и категории
```

Таблица 6.4.1 — Сущность «Каталог» (catalogs)

Название поля	Тип данных	Описание
id	bigint	Идентификатор каталога
name	string	Название раздела каталога
created_at	datetime	Время создания записи
updated_at	datetime	Время последнего обновления записи

Была сделана валидация категории, которая не позволяет иметь одинаковых названий независимо от регистра слов.

Листинг 6.4.2 — Валидация модели каталога

```
validates :name, presence: true, uniqueness: { case_sensitive: false }
```

Добавим пути к контроллеру.

Листинг 6.4.3 — Роуты каталога

```
resources :catalogs # Создаст несколько путей к методам контроллера
```

Для работы с каталогом созданы следующие методы в контроллере.

Публичные методы:

- index - страница со всем списком созданных каталогов;
- show - страница просмотра одного каталога;
- new - страница, на которой вводим данные для создания каталога;

- create - метод, который отвечает на post запрос и записывает данные в БД;
- edit - страница с редактированием каталога;
- update - метод, который отвечает на patch запрос и обновляет данные каталога в БД;
- destroy - отвечает на delete запрос и удаляет запись из БД.

Приватные методы:

- catalog_params - здесь указываются параметры, которые разрешено передавать в запросах;
- current_catalog - данный метод устанавливает запись каталога в методах show, edit, update, destroy. Необходим для поддержания чистоты кода.

Код контроллера будет размещаться в приложении.

6.5 СОЗДАНИЕ СУЩНОСТИ “КАТЕГОРИЯ”

В модели “Категория” хранится ее название, и она принадлежит каталогу. А также иметь множественную связь с продуктами.

Листинг 6.5.1 — Связи модели категории

```
belongs_to :catalog
has_many :products, dependent: :destroy
```

Таблица 6.5.1 — Сущность «Категория» (categories)

Название поля	Тип данных	Описание
id	bigint	Идентификатор категории
name	string	Название раздела каталога
catalog_id	bigint	Идентификатор каталога к которому принадлежит категория
created_at	datetime	Время создания записи
updated_at	datetime	Время последнего обновления записи

Как и с каталогом были провалидированы ячейки с названием перед сохранением в базу данных.

Листинг 6.5.2 — Валидация модели категории

```
validates :name, presence: true, uniqueness: { case_sensitive: false }
```

Так как категории являются вложенными в каталог, то это отразилось и на путях, которые были созданы.

Листинг 6.5.3 — Роуты категорий

```
resources :catalogs, shallow: true do
  resources :categories
end
end
```

Методы в контроллере категории будут схожи с методами каталогов, однако основное различие будет состоять в том, что прежде, чем захотим создать или обновить категорию, должны будем построить связь с каталогом.

Листинг 6.5.4 — Метод создания категории

```
def new
  @catalog = Catalog.find(params[:catalog_id])
  @category = @catalog.categories.build
end
```

В коде, приведенном выше, сперва находим каталог по переданном параметру, а после выстраиваем связь каталога с категорией.

6.6 СОЗДАНИЕ СУЩНОСТИ “ПРОДУКТ”

Создана модель “Продукт”, в которой хранятся все данные о продукте, например такие как цена, название и описание. Продукт будет принадлежать категории, а также иметь один каталог через связь с категорией. Также модель имеет множество связей с продуктом в корзине.

Листинг 6.6.1 — Связи модели продукта

```
belongs_to :category
has_one :catalog, through: :category
has_many :cart_lists, dependent: :destroy
```

Таблица 6.6.1 — Сущность «Продукт» (products)

Название поля	Тип данных	Описание
id	bigint	Идентификатор продукта
name	string	Название продукта
price	decimal	Цена продукта
description	text	Описание продукта
author	string	Автор продукта
publishing	string	Издатель продукта
category_id	bigint	Идентификатор категории, к которому принадлежит продукта
created_at	datetime	Время создания записи
updated_at	datetime	Время последнего обновления записи

Каждый продукт должен иметь изображение. Для загрузки изображений использовался гем 'image_processing'. Который позволяет хранить данные в 'active storage'. Поэтому была сгенерирована ещё одна сущность 'Хранение файлы, которая будет описана в пункте "Создание сущности 'Хранимые файлы"'.

Теперь необходимо провалидировать загружаемое изображение на сервер. Изображение должно быть загружено для каждого продукта, размер изображения не должен превышать 1МБ, а тип изображения может быть только jpeg или png.

Листинг 6.6.2 — Валидация изображения продукта

```
has_one_attached :image
validate :image_product
def image_product
  errors.add(:image, 'is missing!') if image.attached? == false

  errors.add(:image, 'is too big') unless image.byte_size <= 1.megabyte
  unless image.content_type.in?(%('image/jpeg image/png'))
    errors.add(:image, 'needs to be a jpeg or png!')
  end
end
end
```

При разработке контроллера использовалась такая же логика, как и в случае с категориями, однако разница лишь в построении связей. А также пути вложены в категорию.

Листинг 6.6.3 — Роуты продуктов

```
resources :categories, shallow: true do
  resources :products
end
```

6.7 СОЗДАНИЕ СУЩНОСТИ “КОРЗИНА”

Была сгенерирована модель “Корзина”. Корзина будет иметь связи с предметами в корзине.

Листинг 6.7.1 — Связи модели корзины

```
has_many :cart_lists, dependent: :destroy
has_many :products, through: :cart_lists
```

Таблица 6.7.1 — Сущность «Корзина» (carts)

Название поля	Тип данных	Описание
id	bigint	Идентификатор каталога
created_at	datetime	Время создания записи
updated_at	datetime	Время последнего обновления записи

Был создан метод, который каждый раз, когда пользователь заходит на сайт, вызывается, независимо от действия. Метод создает в БД запись с корзиной, а с сервера возвращаются данные с идентификатором корзины и записываются в куки. При повторном заходе на сайт, будет получен идентификатор корзины от пользователя, а если его не будет, то создается новая корзина.

Листинг 6.7.2 — Метод создания корзины

```
before_action :current_cart

private

def current_cart
```

```

    @current_cart = Cart.find(session[:cart_id]) # Поиск корзины по идентификатору
  rescue ActiveRecord::RecordNotFound # В случае если не найдена, создаем новую
    @current_cart = Cart.create
    session[:cart_id] = @current_cart.id

    @current_cart # Возвращаем корзину
  end

```

Создадим путь для просмотра корзины, будут сгенерированы пути только для метода show в контроллере CartsController.

Листинг 6.7.3 — Роуты корзины

```
resources :carts, only: %i[show]
```

В методе show создано две переменные класса, которые позволяют получать список товаров в корзине, а также переменную, которая хранит итоговую стоимость товаров в корзине. Данные переменные будут использоваться при создании клиентской части.

Листинг 6.7.4 — Метод отображения корзины

```

def show
  @cart_lists = @current_cart.cart_lists
  @total_price = 0
  @cart_lists.each do |cart_list|
    @total_price += cart_list.quantity * cart_list.product.price
  end
end

```

6.8 СОЗДАНИЕ СУЩНОСТИ “ПРОДУКТ В КОРЗИНЕ”

Сущность “Продукт в корзине” будет принадлежать самой корзине, а также продукту.

Листинг 6.8.1 — Связи модели «Продукт в корзине»

```

belongs_to :cart, optional: true
belongs_to :product
belongs_to :order, optional: true

```

Таблица 6.8.1 — Сущность «Продукт в корзине» (cart_lists)

Название поля	Тип данных	Описание
id	bigint	Идентификатор продукта в корзине
product_id	bigint	Идентификатор продукта
cart_id	bigint	Идентификатор корзины, к которому принадлежит товар в корзине, опционально
order_id	bigint	Идентификатор заказа, опционально
created_at	datetime	Время создания записи
updated_at	datetime	Время последнего обновления записи

Для работы с предметами в корзине было создано 4 метода, а также пути для них.

Листинг 6.8.2 — Роуты «Продукт в корзине»

```
resources :cart_lists, only: %i[destroy]
post 'cart_lists' => 'cart_lists#create'
post 'cart_lists/:id/add' => "cart_lists#add_quantity", as:
"cart_lists_add"
post 'cart_lists/:id/reduce' => "cart_lists#reduce_quantity", as:
"cart_lists_reduce"
```

Метод ‘create’ отвечает за добавление предмета в корзину. Сперва находим продукт, после проверяем, содержится ли данный продукт в корзине, если находится, то увеличивается число продуктов в корзине, а иначе добавляем продукт в корзину увеличивая количество предметов на 1. После того как будет продукт добавлен в корзину, происходит переадресация в корзину.

Листинг 6.8.3 — Метод добавления продукта в корзину

```
def create
  chosen_product = Product.find(params[:product_id])
  current_cart = @current_cart

  if current_cart.products.include?(chosen_product)
    @cart_list = current_cart.cart_lists.find_by(product_id: chosen_product)
    @cart_list.quantity += 1
  end
end
```

```

else
  @cart_list = CartList.new
  @cart_list.cart = current_cart
  @cart_list.product = chosen_product
  @cart_list.quantity = 1
end

@cart_list.save
redirect_to current_cart
end

```

Метод ‘delete’ отвечает за удаления предмета из корзины. Первым делом находим продукт, а после удаляем, а далее переадресуемся в корзину.

Листинг 6.8.4 — Метод удаления продукта из корзины

```

def destroy
  @cart_list = CartList.find(params[:id])
  @cart_list.destroy
  redirect_to current_cart
end

```

Метод ‘add_quantity’ увеличивает количество продуктов в корзине. Находим продукт в корзине и увеличиваем число на 1.

Листинг 6.8.5 — Метод увеличения количества продуктов в корзине

```

def add_quantity
  @cart_lists = CartList.find(params[:id])
  @cart_lists.quantity += 1
  @cart_lists.save
  redirect_to current_cart
end

```

Метод ‘reduce_quantity’ увеличивает количество продуктов в корзине. Находим продукт в корзине и уменьшаем число на 1, но только в том случае, если количество продуктов в корзине больше одного.

Листинг 6.8.6 — Метод уменьшения количества продуктов в корзине

```

def reduce_quantity
  @cart_lists = CartList.find(params[:id])
  if @cart_lists.quantity > 1
    @cart_lists.quantity -= 1
  end
end

```

```
@cart_lists.save
redirect_to current_cart
end
```

6.9 СОЗДАНИЕ СУЩНОСТИ “ЗАКАЗ”

Создана модель “Заказ”. Модель будет иметь множество связей с продуктами в корзине.

Листинг 6.9.1 — Связи модели заказа

```
has_many :cart_lists, dependent: :destroy
```

Таблица 6.9.1 — Сущность «Заказ» (orders)

Название поля	Тип данных	Описание
id	bigint	Идентификатор заказа
name	string	Имя заказчика
email	string	Email заказчика
address	text	Адрес заказчика
created_at	datetime	Время создания записи
updated_at	datetime	Время последнего обновления записи

Был разработан метод создания заказа. Устройство заказа очень просто, как только пользователь оформляет заказ, ко всем товарам в корзине в поле `order_id` добавляется значение только что созданного заказа, а идентификатор корзины обнуляется. После происходит удаление корзины.

Листинг 6.9.2 — Код метода создания заказа

```
def create
  @order = Order.new(order_params)
  @current_cart.cart_lists.each do |item|
    @order.cart_lists << item
    item.cart_id = nil
  end
  @order.save

  Cart.destroy(session[:cart_id])
end
```

```

    session[:cart_id] = nil
    redirect_to root_path
  end

```

Пути к методам контроллера не являются вложенным.

Листинг 6.9.3 — Роуты заказов

```
resources :orders
```

6.10 СОЗДАНИЕ СУЩНОСТИ “КОММЕНТАРИЙ”

Создана модель “Комментарий”, в ней хранятся данные о создателе комментария, а также продукт, к которому оставлен комментарий. Также храним содержимое комментария.

Листинг 6.10.1 — Связи модели комментариев

```

belongs_to :product
belongs_to :user

```

Таблица 6.10.1 — Сущность «Комментарий» (comments)

Название поля	Тип данных	Описание
id	bigint	Идентификатор комментария
content	text	Содержимое комментария
user_id	bigint	Идентификатор пользователя создавшего комментарий
product_id	bigint	Идентификатор продукта к которому оставили комментарий
created_at	datetime	Время создания записи
updated_at	datetime	Время последнего обновления записи

Созданы два пути, которые позволяют обращаться к методам ‘create’ и ‘destroy’ в контроллере.

Листинг 6.10.2 — Роуты комментариев

```

resources :products, shallow: true do
  resources :comments, only: [:create, :destroy]
end

```



```
end
```

Метод ‘create’ отвечает соответственно за создание комментария. Для создания комментария необходимо найти продукт, к которому мы оставляем комментарий, идентификатор продукта берем из параметров, переданных в ссылке. Строится связь с продуктом и создается запись комментария. Владелец комментария назначаем текущего пользователя. Текущий пользователь берётся из сессии. В случае если все заполнено верно и запись прошла валидацию, то она сохраняется в БД, а иначе мы выводим список ошибок, которые видит пользователь.

Листинг 6.10.3 — Код метода создания комментария

```
def create
  @product = Product.find(params[:product_id])
  @comment = @product.comments.build(comment_params)
  @comment.user = current_user

  respond_to do |format|
    if @comment.save
      format.html { redirect_to @product, notice: 'Комментарий
успешно создана.' }
    else
      format.html { render :new }
      format.json { render json: @comment.errors, status: :unprocessable_entity }
    end
  end
end
```

Метод ‘destroy’ удаляет комментарий. Удалить комментарий может только его создатель или же администратор сайта.

Листинг 6.10.4 — Код метода удаления комментария

```
def destroy
  @comment = Comment.find(params[:id])
  @comment.destroy
  respond_to do |format|
    format.html { redirect_back(fallback_location: root_path) }
    format.json { head :no_content }
  end
end
```

```
end  
end
```

Перед тем, как пользователь получит html структуру сайта, на сервере происходит рендер комментария, который проверяет, может ли данный пользователь удалить комментарий. Права пользователей описаны в разделе “Распределение прав доступа”

Листинг 6.10.5 — Код проверки прав доступа

```
- if can? :manage, comment  
  .manage  
    = link_to 'Удалить', comment, method: :delete, data: { confirm:  
    'Вы уверены?' }
```

6.11 СОЗДАНИЕ СУЩНОСТИ “ХРАНИМЫЕ ФАЙЛЫ”

Для прикрепления файлов к моделям были использованы несколько гемов. Первый - ‘active-storage’, который упрощает загрузку файлов и закрепляет их за моделями. Данный гем работает в паре с ‘image_processing’. Это мощный помощник в преобразовании изображений. Это хороший выбор по умолчанию, она является более новой по сравнению с уже имеющимися, а также быстрее обрабатывает изображения.

Таблица 6.11.1 — Сущность «Хранимые файлы» (active_storage_blobs)

Название поля	Тип данных	Описание
id	bigint	Идентификатор файла
key	string	Ключ в названии файла
filename	string	Название файла
content_type	string	Тип хранимого файла
byte_size	bigint	Размер файла
checksum	string	Хэш контрольной суммы файла
created_at	datetime	Время создания записи

6.12 РЕГИСТРАЦИЯ ПРОСМОТРОВ СТРАНИЦ

Одной из немало важных вещей — это сбор статистики в веб приложении. Будем собирать статистику просмотров продуктов, которая в дальнейшем на позволит выдавать самые популярные товары за все время или за определенный срок.

Чтобы организовать сбор статистики, был использован гем ‘impressionist’. Данный гем регистрирует количество показов за действие с контроллером или же настраивается вручную.

Использование данного гема даст нам статистику без подключения сторонних плагинов. Он сможет заменить нам Google Analytics.

Была сгенерирована таблица в БД.

Таблица 6.12.1 — Сущность «Отпечатки» (impressions)

Название поля	Тип данных	Описание
impressionable_id	bigint	Идентификатор отпечатка
user_id	bigint	Идентификатор текущего пользователя, если посетитель авторизован
controller_name	string	Название контроллера с которого снят отпечаток
action_name	string	Название действия с которого снят отпечаток
ip_address	string	IP адрес пользователя
created_at	datetime	Время создания записи

Добавлено в модель объявление о том, что будет использоваться данный гем.

Листинг 6.12.1 — Код объявления использования гема

```
Class Product < ApplicationRecord
  is_impressionable
  ...
end
```

А также вызывать метод отслеживание в методе контроллера, о котором хотим собирать данные. В данном случае это метод ‘show’ в контроллере продуктов.

Листинг 6.12.2 — Код отслеживания действий с продуктом

```
class ProductsController < ApplicationController
  ...
  def show
    @product = Product.find(params[:id])
    impressionist(@product)
  end
end
```

Полученную статистику используем для получения самых просматриваемых продуктов. Данные продукты выведены на главной странице, на странице каталогов и категорий. Для этого был написан небольшой SQL запрос.

Листинг 6.12.3 — Код запроса популярных продуктов

```
@most_vieved = Product
  .joins(:impressions, :catalog)
  .group('products.id')
  .order('count(impressions.impressionable_id) DESC')
  .limit(4)
```

6.13 РАСПРЕДЕЛЕНИЕ ПРАВ ДОСТУПА

Для того чтобы управлять правами доступа пользователей, будет использован гем ‘CanCanCan’. CanCanCan - это гем авторизации для Ruby и Ruby on Rails, который ограничивает к каким ресурсам разрешен доступ данному пользователю.

Все разрешения могут быть определены в одном или нескольких файлах “возможностей”.

Установив, были настроены права доступа. Любой посетитель может управлять корзиной и списком продуктов в корзине, а также просматривать все страницы. Пользователь с ролью администратора имеет доступ к любому

контроллеру, а также создавать\удалять\редактировать все записи. Пользователь с user_role имеет право создавать комментарии.

Листинг 6.13.1 — Распределение прав доступа

```
class Ability
  include CanCan::Ability

  def initialize(user)
    can :read, :all
    can :manage, [CartList]

    return unless user.present?

    if user.admin_role?
      can :manage, :all
      return
    end

    if user.user_role?
      can :manage, Comment, user_id: user.id
    end
  end
end
```

7 СОЗДАНИЕ КЛИЕНТСКОЙ ЧАСТИ

7.1 ШАПКА САЙТА

Данный элемент страниц имеет внешний вид, изображенный на рисунках 7.1.1, 7.1.2.



Рисунок 7.1.1 – Шапка авторизованного пользователя



Рисунок 7.1.2 – Шапка неавторизованного пользователя

Данный элемент содержит следующее:

1. Логотип веб-приложения;
2. Ссылки на статические страницы;
 - главная;
 - о нас;
 - помощь.
3. Ссылка на страницу с корзиной;
4. Ссылки управления сессией;
 - если пользователь авторизован;
 - i. ссылка с запросом удаления сессии;
 - если пользователь не авторизован.
 - i. войти;
 - ii. регистрация.
5. Блок с ссылками к каталогам (Книги, канцтовары, сувениры, творчество).

7.2 БЛОК ПРОДУКТА

Данный элемент страниц имеет внешний вид, изображенный на рисунке 7.2.1.



Рисунок 7.2.1 – Внешний вид блока продукта

Данный элемент содержит следующее:

1. Изображение продукта;
2. Название продукта;
3. Цена продукта;
4. Кнопка приобретения продукта.

Данный элемент представляет собой “частицу” в *ruby on rails*. Его название начинается с нижнего подчеркивания, а само название описывает объект внутри кусочка.

Отображается “частица” следующим образом. К переменной класса, которая объявлена в контроллере вызывается метод перебора, на каждой итерации вызывается рендер этой “частицы”.

Листинг 7.2.1 — Рендер «частицы» продукта

```
- @products.each do |product|  
  = render 'products/product', product: product
```

Разметка “частицы” имеет следующий вид:

Листинг 7.2.2 — Разметка «частицы» продукта

```
.product-block  
.product-card  
  = link_to product do  
    .product-image  
      = image_tag(url_for(product.image.variant(resize_to_limit: [165,  
250])))  
    = link_to product do  
      .product-link-title = product.name  
      .product-buy-block  
      .product-price = product.price.to_s + '₽'  
      = link_to cart_lists_path(product_id: product.id), method: :post do  
        .button.button-blue Купить
```

7.3 ГЛАВНАЯ СТРАНИЦА

Данная страница имеет внешний вид, изображенный на рисунке 7.3.1.

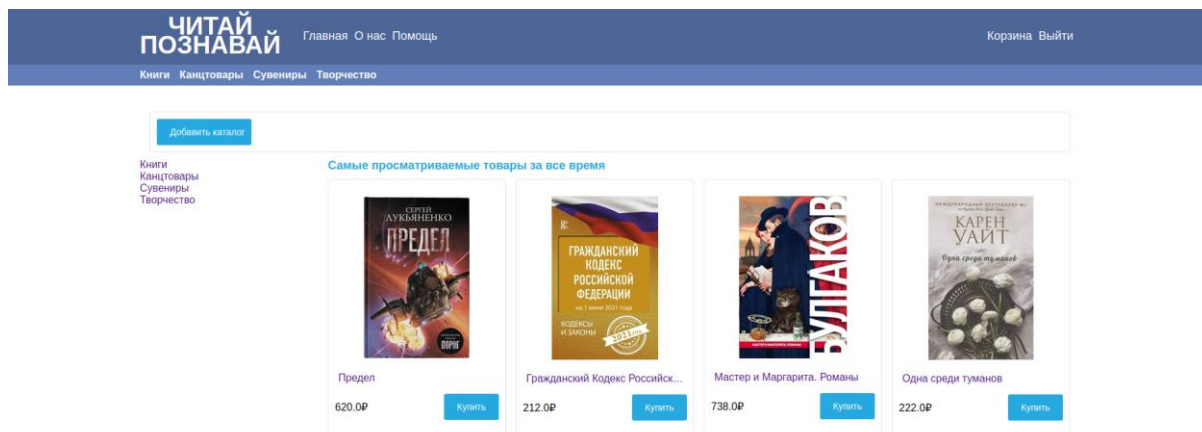


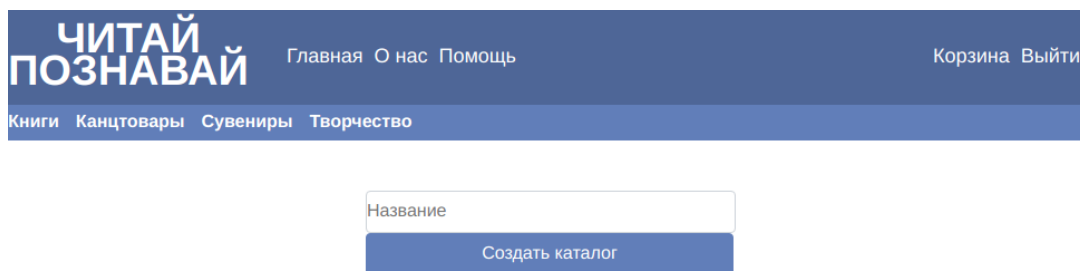
Рисунок 7.3.1 – Внешний вид главной страницы

1. Блок доступный для администратора, который содержит:
 - кнопка “добавить каталог”

2. Блок с ссылками на каталоги (Книги, канцтовары, сувениры, творчество);
3. Блок с самыми просматриваемыми товарами за все время, который содержит 4 блока с продуктом.

7.4 СТРАНИЦА СОЗДАНИЯ КАТАЛОГА

Данная страница имеет внешний вид, изображенный на рисунке 7.4.1.



ЧИТАЙ
ПОЗНАВАЙ

Главная О нас Помощь Корзина Выйти

Книги Канцтовары Сувениры Творчество

Название

Создать каталог

Рисунок 7.4.1 – Внешний вид формы создания каталога

Страница содержит процедуру создания каталога, а также следующие элементы:

1. Поле ввода для названия;
2. Кнопка подтверждения создания каталога.

7.5 СТРАНИЦА КАТАЛОГА

Данная страница имеет внешний вид, изображенный на рисунке 7.5.1.

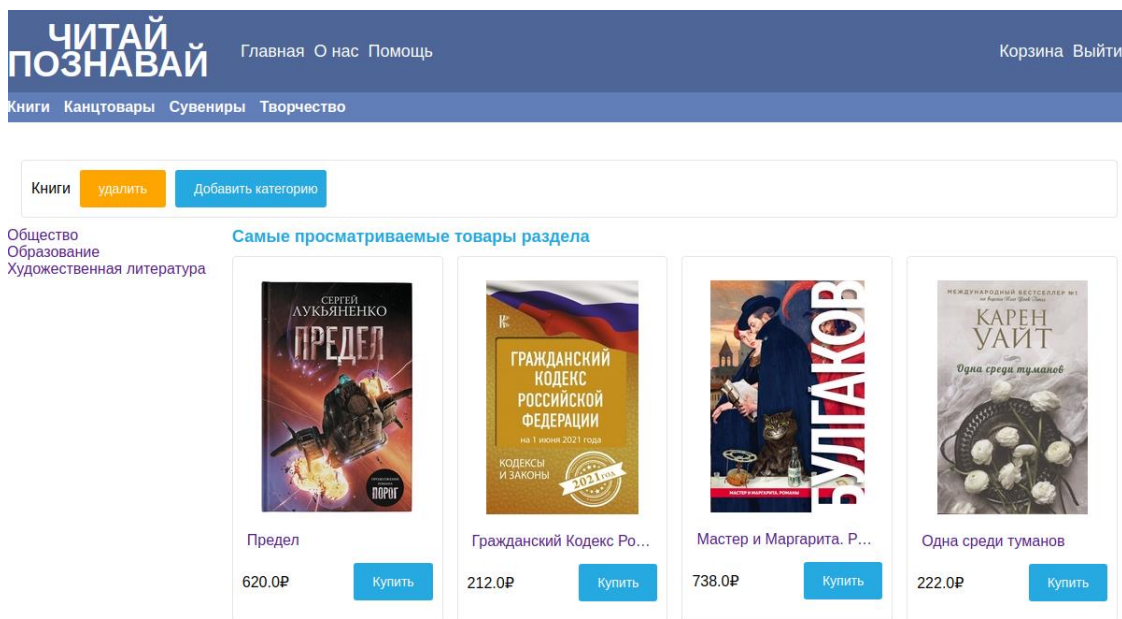


Рисунок 7.5.1 – Внешний вид страницы каталога

Страница содержит следующие элементы:

1. Блок доступный для администратора, который содержит:
 - название каталога;
 - кнопка “добавить категорию”;
 - кнопка “удалить”, которая удаляет каталог.
2. Блок с ссылками на категории (Общество, Образование, Художественная литература);
3. Блок с самыми просматриваемыми товарами раздела, который содержит 4 блока с продуктом.

7.6 СТРАНИЦА СОЗДАНИЯ КАТЕГОРИИ

Данная страница имеет внешний вид, изображенный на рисунке 7.6.1.

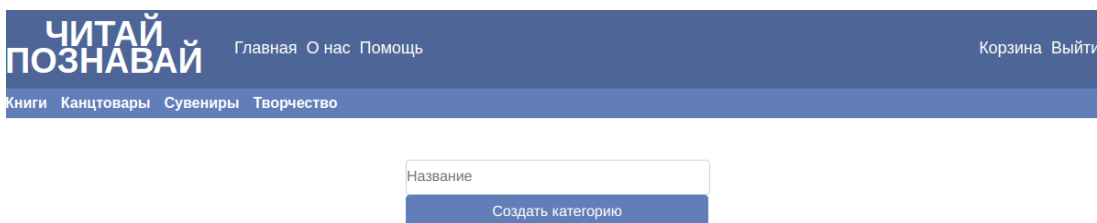


Рисунок 7.6.1 – Внешний вид формы создания категории

Страница содержит процедуру создания категории, а также следующие элементы:

1. Поле ввода названия категории;
2. Кнопка создания категории.

7.7 СТРАНИЦА КАТЕГОРИИ

Данная страница имеет внешний вид, изображенный на рисунке 7.7.1.

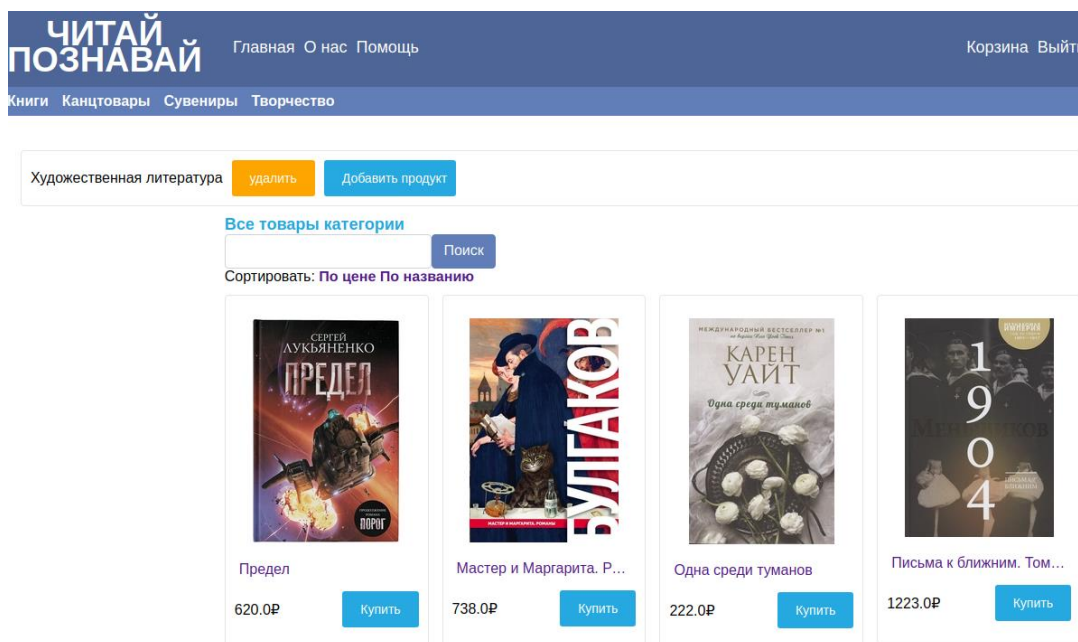


Рисунок 7.7.1 – Внешний вид страницы категории

Страница содержит следующие элементы:

2. Блок доступный для администратора, который содержит:
 - название категории;

- кнопка удаления категории;
 - кнопка добавления продукта к категории.
3. Поиск товара по названию;
 4. Фильтры порядка выдачи продуктов;
 - по цене;
 - по названию.
 5. Блоки с множеством продуктов доступных в категории.

7.8 СТРАНИЦА СОЗДАНИЯ ПРОДУКТА

Данная страница имеет внешний вид, изображенный на рисунке 7.8.1.

ЧИТАЙ
ПОЗНАВАЙ

Главная О нас Помощь

Корзина Выйти

Книги Канцтовары Сувениры Творчество

Название

Цена

Описание

Автор

Издательство

Изображение товара

Choose File No file chosen

Создать продукт

Рисунок 7.8.1 – Внешний вид страницы категории

Страница содержит следующие элементы:

1. Поле ввода название;
2. Поле ввода цены;
3. Текстовое поле ввода описания;
4. Поле ввода автора;
5. Поле ввода издательства;
6. Кнопка загрузки изображения продукта;
7. Кнопка создания продукта.

7.9 СТРАНИЦА ПРОДУКТА

Данная страница имеет внешний вид, изображенный на рисунке 7.9.1.

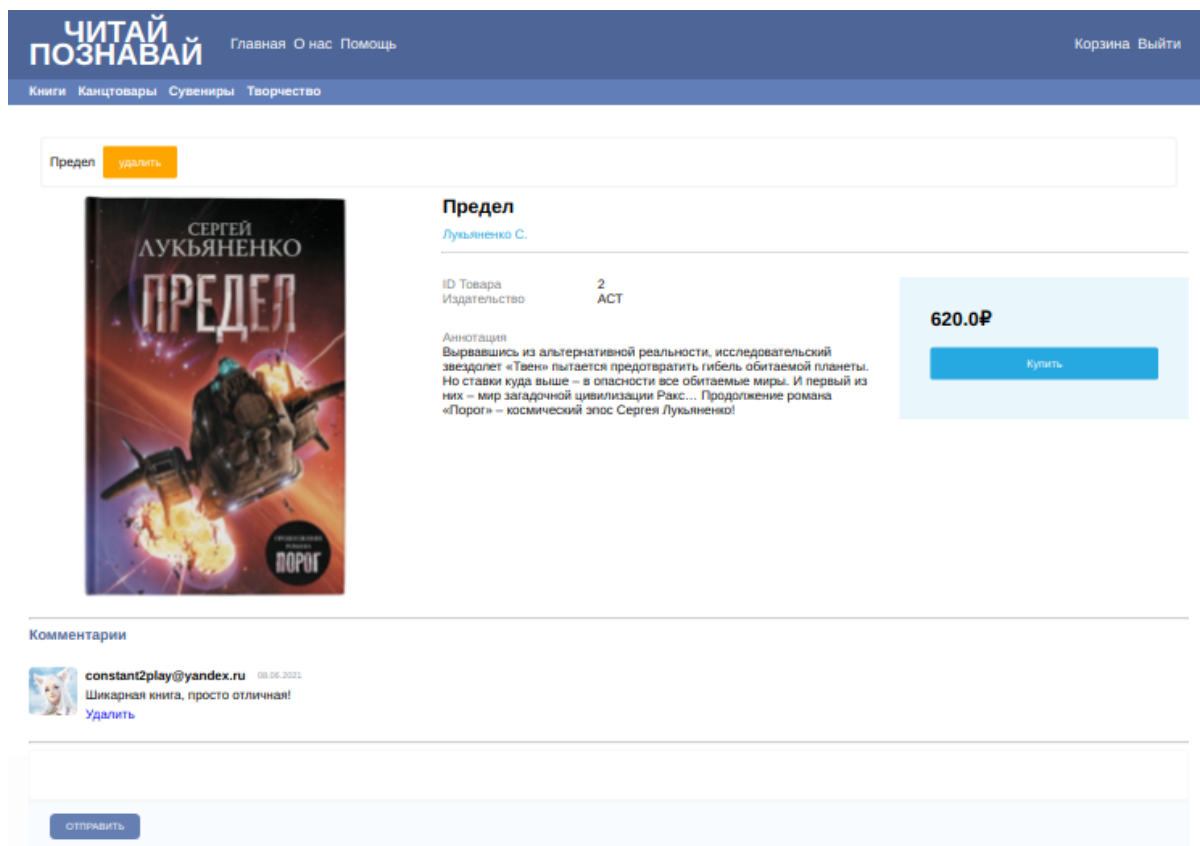


Рисунок 7.9.1 – Внешний вид страницы продукта

Страница содержит следующие элементы:

1. Блок доступный для администратора, который содержит:
 - Название продукта;
 - Кнопка удаления продукта.
2. Блок с изображением товара
3. Блок с информацией о продукте:
 - Название продукта;
 - Автор произведения;
 - Идентификатора продукта;
 - Название издательства;
 - Аннотация к продукту.
4. Блок приобретения товара:

- цена товара;
 - кнопка покупки товара.
5. Комментарии к товару;
 6. Поле ввода комментария. Доступно только авторизованному пользователю.

7.10 СТРАНИЦА КОРЗИНЫ

Данная страница имеет внешний вид, изображенный на рисунке 7.10.1.

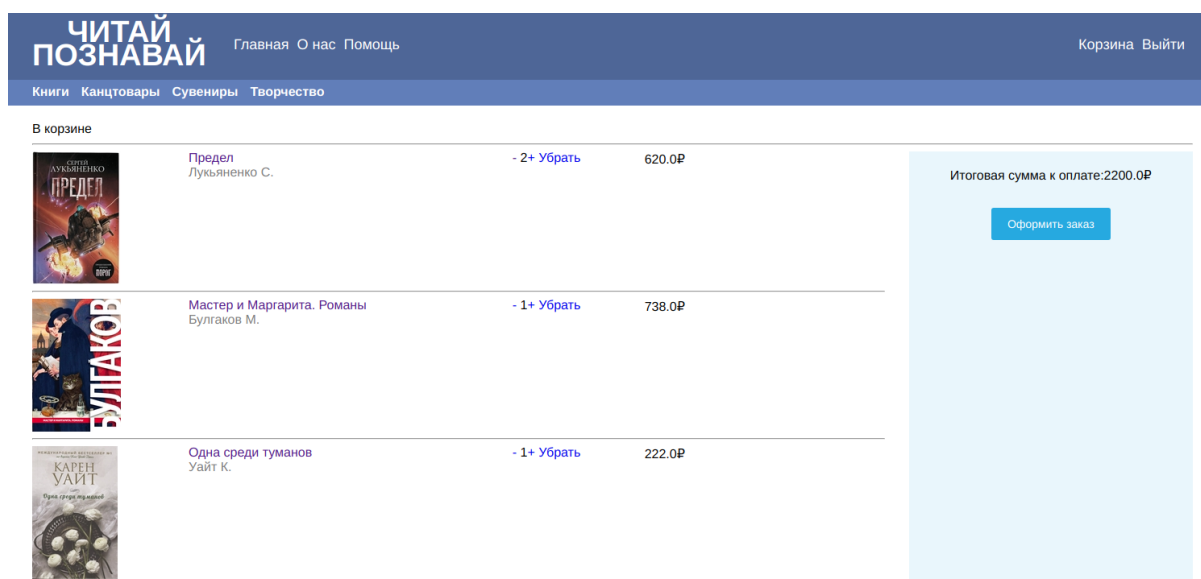


Рисунок 7.10.1 – Внешний вид страницы корзины

Страница содержит следующие элементы:

1. Продукты в корзине:
 - Изображение продукта;
 - Название и автор продукта;
 - Количество приобретаемого продукта;
 - Кнопка “убрать” продукт из корзины;
 - Кнопка добавить\убавить количество товаров в корзине;
 - Цена продукта.
2. Блок с оформлением заказа:
 - Итоговая стоимость заказа;
 - Кнопка оформить заказ.

7.11 СТРАНИЦА ОФОРМЛЕНИЯ ЗАКАЗА

Данная страница имеет внешний вид, изображенный на рисунке 7.11.1.

Сводка заказа:

Предел (X2) ID: 2	1240.0 Руб.
Мастер и Маргарита. Романы (X1) ID: 3	738.0 Руб.
Одна среди туманов (X1) ID: 4	222.0 Руб.
Итого:	2200.0 Руб.

Ваше имя

Email

Адрес доставки

Создать заказ

Рисунок 7.11.1 – Внешний вид страницы оформления заказа

Страница содержит следующие элементы:

1. Перечень товаров;
2. Поле ввода имени заказчика;
3. Поле ввода email, если пользователь авторизован, значение подставляется автоматически;
4. Адрес доставки;
5. Кнопка создания заказа.

7.12 СТРАНИЦА ОФОРМЛЕННОГО ЗАКАЗА

Данная страница имеет внешний вид, изображенный на рисунке 7.12.1.

Благодарим Вас за заказ

Номер вашего заказа #6

Для подтверждения указанной вами при оформлении заказа электронной почты мы отправили на неё детали вашего заказа. В скором времени наш оператор свяжется с вами по телефону для подтверждения заказа. После оплаты заказа на указанную вами почту мы вышлем данные для получения заказа

Рисунок 7.12.1 – Внешний вид страницы оформленного заказа

На странице содержится благодарственное письмо.

7.13 СТРАНИЦА АВТОРИЗАЦИИ

Данная страница имеет внешний вид, изображенный на рисунке 7.13.1.

Войти

Email

admin@example.com

Пароль

.....

☐ Запомнить меня

Войти

[Регистрация](#)

[Забыли пароль?](#)

Рисунок 7.13.1 – Внешний вид страницы авторизации

Страница содержит следующие элементы:

1. Поле ввода email'а;
2. Поле ввода пароля;
3. Чек-бокс “запомнить меня”;

4. Кнопка “Войти”;
5. Ссылка на страницу восстановления пароля.

В случае неверно введенных данных, пользователь получает уведомление об этом.

7.14 СТРАНИЦА РЕГИСТРАЦИИ

Данная страница имеет внешний вид, изображенный на рисунке 7.14.1.

Регистрация

Email

Пароль (минимум 6 знаков)

Подтверждение пароля

Регистрация

Рисунок 7.14.1 – Внешний вид страницы регистрации

Страница содержит следующие элементы:

1. Поле ввода email’а;
2. Поле ввода пароля;
3. Поле подтверждения пароля;
4. Кнопка завершения регистрации.

В случае если email не удовлетворяет валидации, пользователь получит об этом уведомление, также как если пользователь введен два разных пароля.

ЗАКЛЮЧЕНИЕ

Выпускная квалификационная работа на тему «Web-приложение для автоматизации заказов интернет-магазина» была выполнена в соответствии с поставленной задачей. В ходе данной работы были проведены: обзор аналогов для автоматизации заказов, выбрана архитектура приложения, составлена инфологическая модель базы данных, создана серверная и клиентская часть.

Для реализации проекта были выбраны такие средства, как язык программирования Ruby, фреймворк Ruby on Rails, база данных PostgreSQL, препроцессор SLIM, а также препроцессор SCSS.

Было спроектировано и разработано веб-приложение, реализующее работу интернет-магазина по продаже книг. Реализованное веб-приложение удовлетворяет всем требованиям, выдвинутым в поставленной задаче.

Веб-приложение разработано с учетом следующих особенностей:

- понятный интерфейс;
- защита от межсайтового скриптинга;
- защита от межсайтовой подделки запросов;
- высокая скорость функционирования;
- фильтрация товаров;
- распределение ролей в приложении;
- сбор статистики от пользователей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Официальная документация по Ruby on Rails. Обучающие материалы. URL: <https://guides.rubyonrails.org/> (дата обращения: 14.04.2021).
2. StimulusJS. Скромный JavaScript-фреймворк для того HTML, который у вас уже есть. URL: <https://stimulus.hotwire.dev/> (дата обращения: 15.04.2021).
3. Web-creator. Почему стоит выбрать Ruby on Rails для разработки веб-приложения или сайта. URL: https://web-creator.ru/articles/why_ruby_on_rails (дата обращения: 04.05.2021).
4. Mkdev. Как работает Ruby on Rails и о том, насколько важно понимать, что значит Ruby в названии этого фреймворка. Rails это просто Ruby. URL: <https://mkdev.me/posts/rails-eto-prosto-ruby> (дата обращения: 08.05.2021).
5. Интернет-технологии. Схема MVC в Ruby on Rails. URL: <https://www.internet-technologies.ru/articles/shema-mvc-v-ruby-on-rails.html> (дата обращения: 08.05.2021).
6. Студия Михаила Кечинова. 10 наиболее частых ошибок на Ruby on Rails. URL: https://mkechinov.ru/rails_mistakes.html (дата обращения: 12.05.2021).
7. Beget. Полезные статьи. Установка и настройка Ruby on Rails. URL: <https://beget.com/ru/kb/how-to/web-apps/ruby> (дата обращения: 12.05.2021).
8. Хекслет. Язык программирования Ruby: особенности, перспективы, рынок труда. URL: <https://ru.hexlet.io/blog/posts/yazyk-programmirovaniya-ruby-osobennosti-perspektivy-rynok-truda> (дата обращения: 16.05.2021).
9. Медиум. Ruby on Rails меняет всё. URL: <https://medium.com/nuances-of-programming/ruby-on-rails-%D0%BC%D0%B5%D0%BD%D1%8F%D0%B5%D1%82-%D0%B2%D1%81%D1%91-87fdc4ce79c5> (дата обращения: 21.05.2021).
10. Kverner. Для чего нужен Ruby on Rails: Советы от Back-end разработчиков. URL: <https://www.kverner.ru/dlya-chego-nuzhen-ruby-on-rails-sovety-ot-back-end-razrabotchikov/> (дата обращения: 20.05.2021).

ПРИЛОЖЕНИЕ

Исходные коды выпускной квалификационной работы бакалавра представлены на приложенном USB-флэш-накопителе.