МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**

Кафедра дифференциальных уравнений и
математической экономики

# РАЗРАБОТКА И ВНЕДРЕНИЕ ВЕБ-САЙТА ДЛЯ МАГАЗИНА ОБУВИ

АВТОРЕФЕРАТ МАГИСТЕРСКОЙ РАБОТЫ

Студентки 2 курса 248 группы
направления 09.04.03 — Прикладная информатика

механико-математического факультета
Сережкиной Софии Александровны

Научный руководитель
доцент, к. э. н. _____ С. В. Иванилова

Заведующий кафедрой
зав. кафедрой, д. ф.-м. н., доцент _____ В. С. Рыхлов

Саратов 2026

**Введение.** Информационная система (ИС) - совокупность средств, методов и персонала, используемых для хранения, обработки и выдачи информации в интересах достижения поставленной цели. Она предназначена для обеспечения людей необходимой информацией, то есть для удовлетворения информационных потребностей в рамках конкретной предметной области. Результатом работы информационных систем является информационная продукция — документы, информационные массивы, базы данных и информационные услуги.

**Актуальность данной работы** определена тем, что в современном мире любому частному предприятию, особенно из области продаж товаров, необходима веб-платформа, на которой был бы размещен товар организации. Сайт магазина значительно упрощает не только работу продавцов, но и действия покупателей. Наличие каталога товаров, в котором по заданному фильтру можно найти необходимый продукт, возможность оформления заказа, получение новостных рассылок с выгодными предложениями - данный функционал становится доступным для клиентов с появлением сайта магазина.

Продавец имеет возможность размещать товар на платформе, собирать обратную связь от клиентов, оформлять заказы. В век развития технологий предприятия без информационной системы могут проиграть конкурентам. Например, в сфере торговли пользователи в большинстве случаев совершают покупки одежды и обуви через сайт магазина.

**Целью магистерской работы** является разработка веб-сайта для магазина обуви в городе Саратов “ТуКикс”. Магазин открылся летом 2024 года, каталог товаров размещается в мессенджере “Телеграм”, что является непрактичным.

В рамках данной магистерской работы планируется реализовать следующие **задачи**:

— изучить технологии веб-разработки;
— провести сравнительный анализ технологий;
— осуществить сбор и анализ требований заказчика;
— смоделировать работу сайта;
— реализовать клиентскую и серверную части сайта.

**Основное содержание работы.** Работа состоит из введения, четырех разделов, заключения и списка использованных источников, содержащего 23 наименования. Первый раздел магистерской работы посвящен технологиям создания сайта и инструментам веб-разработки, второй - описанию процесса сбора и анализа требований, а также построению UML-диаграмм работы сайта, третий и четвертый разделы - разработке клиентской и серверной частей разрабатываемого сайта.

В **первом разделе** даются определения таким понятиям, как front-end и back-end разработка, описывается основной стек технологий, используемых в проекте, а также проводится сравнительный анализ веб-фреймворков.

Frontend-разработка — это создание внешней, визуальной части веб-сайтов и приложений, с которой взаимодействует пользователь (кнопки, меню, анимации). По итогам клиентской разработки сайт должен быть привлекательным и удобным в использовании. Для этого используются различные технологии и инструменты, которые позволяют создавать интерактивные и динамичные веб-страницы.

JavaScript — язык программирования, который позволяет создавать динамически обновляемый контент, управлять мультимедиа, анимировать изображения. Также JavaScript используется для разработки одностраничных приложений, где весь контент загружается на одной странице, а переходы между разделами происходят без перезагрузки страницы.

React - это JavaScript-библиотека для создания пользовательских интерфейсов. Она основана на декларативном подходе, то есть в ее основе лежит описание изменения интерфейса в зависимости от событий. Данное описание представлено в виде компонента, где компонент - набор Javascript-функций, которые возвращают фрагмент страницы. Компонентами являются формы, кнопки, поля приложения.

Back-end технологии - это технологии, которые обеспечивают логику работы приложения, обработку данных, связь с сервером и управление запросами. В ключевые технологии данной группы входят языки программирования, базы данных, фреймворки, работа серверов, использование контейнеризации, брокеры сообщений.

Django Framework — это высокоуровневый Python веб-фреймворк, ко-

торый позволяет быстро создавать безопасные и поддерживаемые веб-сайты. Данный фреймворк имеет следующие преимущества:

1. настроенный веб-сервер, который обрабатывает запросы от пользователей к веб-сервису;
2. готовые механизмы для авторизации пользователей;
3. административный интерфейс для управления используемыми данными;
4. система кэширования для увеличения скорости загрузки и открытия страниц через браузеры, внешние клиенты или приложения;
5. интерфейсы и адаптеры для подключения к различным типам баз данных.

Брокер сообщений - программное обеспечение и архитектурный паттерн, который обеспечивает асинхронный обмен сообщениями между сервисами. Сервис, отправляющий данные, называется продюсером (producer), а потребляющий — потребителем (consumer).

RabbitMQ — это один из популярных брокеров, написанный на языке Erlang, который служит посредником для обмена информацией между различными системами. Он реализует протокол AMQP (Advanced Message Queuing Protocol) — открытый стандарт для передачи сообщений между приложениями. AMQP определяет такие понятия, как обменники (exchanges), очереди (queues) и привязки (bindings).

Результат сравнительного анализа веб-фремворков представлен на таблице 1.

**Второй раздел** посвящен проектированию информационной системы - а именно сбору и анализу требований заказчика и моделированию работы сайта.

В ходе обсуждения с заказчиком, какие функции на сайте он хотел бы видеть, были определены следующие требования к пользовательским функциям:

— сайт должен обеспечить возможность регистрации и авторизации пользователей;
— после регистрации должен создаваться личный кабинет пользователя;
— пользователь должен иметь возможность добавлять и удалять товары

Таблица 1 – Сравнительный анализ веб-фреймворков

| Название фреймворка | Производительность | Легкость в изучении | Архитектура | Безопасность |
|---|---|---|---|---|
| Django | высокая производительностью | легкий в изучении из-за языка Python | Model-Template-View | CSRF токен, Django ORM |
| Laravel | может работать медленно | на обучение может уйти больше времени из-за основы языка PHP | Model-View-Controller | алгоритм хеширования Bcrypt |
| Angular | высокая производительность благодаря использованию виртуального DOM | сложный для новичков из-за концепций реактивного программирования | Model-View-Controller | включает встроенные функции безопасности |
| Ruby on Rails | невысокая производительность по сравнению с языками Java и C++ | легкий в изучении | Model-View-Controller | Sandboxing, безопасные API, кодировка строк |
| Flask | хорошая производительность для небольших приложений | легкий в изучении | гибкость в выборе архитектуры | средний уровень безопасности |

из корзины;

— в корзине пользователя должна выводиться итоговая сумма товаров, находящихся в корзине;
— в корзине пользователя должна быть предусмотрена возможность оформления заказа;
— сайт должен включать каталог товаров с возможностью фильтрации по цене, размеру и бренду;
— каждый товар, представленный в каталоге, должен иметь карточку, включающую описание и характеристики товара;
— пользователь должен иметь возможность оставлять отзыв на сайте;
— пользователь должен иметь возможность подписываться на рассылку с новостями магазина.

Анализируя требования к пользовательским функциям, можно сделать

вывод, что сайт должен включать такие страницы веб-интерфейса, как страница авторизации, регистрации, каталога товаров, детальной информации о товаре, личного кабинета пользователя, корзины пользователя с возможностью оформления заказа, страница отзывов и подписки на новости магазина.

Требования к административным функциям включают возможность добавления, удаления и редактирования товаров, отображаемых в каталоге, просмотр созданных заказов и отзывов пользователей, обеспечение рассылки новостей всем подписчикам.

Также были построены и описаны UML диграммы. Диаграмма вариантов использования содержит действия, которые доступны пользователям на сайте, диаграмма активности отражает процесс оформления заказа с альтернативными потоками, диаграмма компонентов содержит архитектуру программной системы.

**Третий раздел** посвящен разработке клиентской части сайта, а именно описанию организации компонентов, маршрутизации, разработке необходимых компонентов. Для реализации использовался язык JavaScript с фреймворком React, который взаимодействует с Django REST API.

К ключевым задачам клиентской части относятся отображение каталога товаров, корзины, личного кабинета, отправка отзывов, подписка на новости. Также к набору инструментов разработки относятся JSX, Fetch API, маршрутизация осуществляется через window.location.pathname, управление состоянием через хуки (useState, useEffect).

Маршрутизация построена на анализе свойства window.location.pathname, которое содержит путь текущей страницы. Главный компонент приложения App, рендеринг которого осуществляется в файле index.jsx, выполняет последовательную проверку пути и возвращает соответствующий компонент.

Директория components содержит все переиспользуемые компоненты React приложения. Каждый компонент реализует определенную функциональную часть интерфейса и может быть как независимым, так и встроенным в более сложные компоненты. Описание компонентов приложения приведены в таблице 2.

В **четвертом разделе** описывается реализация серверной части сайта - проектирование базы данных, разработка API, подключение брокера со-

Таблица 2 – Компоненты приложения React и их назначение

| Компонент | Файл | Назначение |
|---|---|---|
| Catalog | Catalog.jsx | Отображение каталога товаров с возможностью фильтрации по бренду, цене, размеру и цвету. Управляет загрузкой данных с API и состоянием фильтров |
| Cart | Cart.jsx | Отображение корзины покупок: список добавленных товаров, изменение количества, удаление позиций, подсчет итоговой суммы |
| ProductDetail | ProductDetail.jsx | Детальная страница конкретного товара с полным описанием, характеристиками и возможностью добавления в корзину с выбором количества. |
| Profile | Profile.jsx | Личный кабинет пользователя: просмотр и редактирование личных данных, смена пароля |
| Response | Response.jsx | Форма для отправки отзывов о товарах (текстовое поле и кнопка отправки) |
| Subscribe | Subscribe.jsx | Форма подписки на новости магазина (поле ввода адреса и кнопка) |
| CheckoutModal | CheckoutModal.jsx | Модальное окно для оформления заказа |

общений для создания асинхронной рассылки, работа с административной панелью, развертывание сайта на хостинге.

Для создания базы данных (и всех взаимодействий с ней) Django использует ORM. ORM (Object-Relational Mapping или Объектно-Реляционное Представление) в Django framework - технология программирования, которая связывает отношения базы данных с классами объектно-ориентированного программирования. ORM позволяет работать с базой данных, используя объекты Python вместо написания прямых SQL-запросов. По умолчанию Django сконфигурирован для работы с базой данных SQLite.

Спроектированная база данных содержит 12 таблиц, основная часть которых связана с таблицей товаров. Три таблицы (подписчиков, рассылки, отзывов) не имеют внешних ключей, так как по логике работы подписка и отзывы не требуют привязки к товару или к аккаунту пользователя.

Для корзины пользователя и оформления заказа были реализованы следующие эндпоинты:

— **GET /products/api/cart/** - возвращение содержимого корзины авто-

ризованного пользователя, а также общего количества товаров и итоговой суммы корзины;

— **POST /products/api/cart/add/** - добавление товара в корзину. Если товар уже присутствует в корзине, количество увеличивается, если отсутствует — создается новая запись;
— **POST /products/api/cart/update/** - изменение количества товара в корзине. Перед сохранением выполняется проверка на количество товаров - значение не может стать меньше единицы;
— **DELETE /products/api/cart/remove/<id>/** - удаление позиции из корзины по её идентификатору;
— **POST /products/api/orders/create/** - создание заказа и привязка к нему позиции из корзины пользователя;
— **GET /products/api/orders/my/** - получение списка заказа текущего пользователя;
— **GET /products/api/orders/id/** - получение детальной информации о конкретном заказе.

Эндпоинты подписки, отзывов и профиля пользователя имеют вид:

— **POST /products/api/subscribe/** - добавление адреса почты в базу данных подписчиков. При повторной подписке ранее отписавшегося пользователя его статус восстанавливается;
— **POST /products/api/responses/** - создание нового отзыва. Принимает текст отзыва, автоматически проставляя дату создания;
— **GET /products/api/profile/** - данные текущего авторизованного пользователя;
— **PUT /products/api/profile/** - данные пользователя (имя, фамилия, email, телефон);
— **POST /products/api/change-password/** - изменение пароля пользователя. Требует указания текущего пароля для подтверждения.

Для защиты эндпоинтов, изменяющих состояние системы (добавление в корзину, удаление, обновление профиля), используется декоратор проверки наличия авторизации у пользователя - @permission_classes([IsAuthenticated]). Если запрос поступает от неавторизованного пользователя, сервер возвращает ошибку 401 (Unauthorized) или 403 (Forbidden)

Для реализации асинхронной рассылки пользователя использовался встроенный модуль django.core.mail, который позволяет работать с отправкой электронных писем. Модуль предоставляет функционал для работы с различными SMTP-серверами.

После настройки и успешного тестирования синхронной отправки писем были подключены приложение Celery и брокер сообщений RabbitMq для реализации асинхронной отправки. Алгоритм рассылки писем выглядит следующим образом:

1. Пользователь оставляет адрес своей электронной почты в специальном окошке на странице «Подписка на новости»;
2. Выполняется первая задача, отвечающая за отправку приветственного письма. Также указанный электронный адрес сохраняется в базу данных подписчиков;
3. Администратор в панели администратора публикует массовую рассылку;
4. Создается вторая задача, отвечающая за рассылку писем на все адреса, существующие в базе данных.

Были разработаны две асинхронные задачи с использованием библиотеки Celery. Задача send_subscription_email отвечает за отправку приветственного письма новому подписчику. Она вызывается сразу после того, как пользователь ввел свой адрес электронной почты в форму подписки и нажал кнопку «Подписаться».

Вторая задача send_newsletter_to_all осуществляет массовую рассылку писем всем активным подписчикам. Она вызывается из административной панели Django, когда администратор создает новую рассылку и инициирует её отправку. Для каждого письма формируется HTML-версия письма с использованием переданного содержимого. HTML-письмо позволяет использовать заголовки, абзацы, встроенные стили и другие элементы форматирования, что делает рассылку удобной для восприятия.

В панели администрирования администратору доступно добавление товаров и их редактирование, оформление заказов, создание рассылки пользователям, просмотр отзывов пользователей.

Для размещения веб-сайта на хостинге был выбран VPS-хостинг (Virtual

Private Server) от Reg.ru. После настройки WSGI сервера с параметрами хостинга сайт стал доступен по адресу http://dva.kik.ru.

**Заключение.** Таким образом, был разработан веб-сайт для магазина обуви в Саратове, который удовлетворяет требованиям заказчика. На сайте был реализован такой функционал, как просмотр каталога товара, добавление товаров в корзину, оформление заказов, авторизация и регистрация пользователей, создание личного кабинета, оставление отзывов, подписка на новости магазина.

Были описаны основные технологии разработки сайта, проведен сравнительный анализ фреймворков для выбора наиболее подходящего инструмента под цели проекта.

Сформулировано техническое задание на разработку сайта с перечнем функций, которые клиент хотел бы видеть на сайте.

Разработка клиентской части сайта включала организацию компонентов приложения, обеспечение взаимодействия компонентов с серверной частью посредством API запросов. Были реализованы пользовательские страницы с использованием JavaScript, HTML, CSS.

Для реализации серверной части была спроектирована база данных сайта, API эндпоинты, подключен брокер сообщений RabbitMq для отправки асинхронной рассылки пользователям, разработан функционал для работы в панели администратора. По завершении разработки сайт был развернут на хостинге.