Департамент образования и науки города Москвы

Государственное бюджетное профессиональное образовательное учреждение города Москвы

«Колледж малого бизнеса № 4»

(ГБПОУ КМБ № 4)

**КУРСОВОЙ ПРОЕКТ**

по профессиональному модулю ПМ. 05 Сопровождение информационных систем по специальности СПО 09.02.07 Информационные системы и

программирование

Тема:Проектирование и разработка интегрированной CRM-системы для малого интернет-магазина с функционалом аналитики продаж и рекомендациями товаров.

Выполнил(а) студент(ка)

2 курса группы № Авданина Мария Михайловна

(Фамилия, имя, отчество студента)

Проверил

руководитель курсовой работы

Чепрасова Анастасия Сергеевна

Работа защищена с оценкой

« » 2025

Москва, 2025

**Департамент образования и науки города Москвы**

**Государственное бюджетное профессиональное образовательное учреждение города Москвы «Колледж малого бизнеса № 4»**

**(ГБПОУ КМБ № 4)**

УТВЕРЖДАЮ

Заместитель директора «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_ г.

\_\_\_\_\_\_\_\_\_\_\_\_ / И.Ю. Атрощенко

**ЗАДАНИЕ**

на курсовую работу

по профессиональному модулю ПМ. 05 Сопровождение информационных систем

по специальности СПО 09.02.07 Информационные системы и программирование

Студента группы ИПС-21.23, Авданина Мария Михайловна

Тема курсовой работы: Проектирование и разработка интегрированной CRM-системы для малого интернетмагазина с функционалом аналитики продаж и рекомендациями товаров.

Срок защиты курсовой работы: 14.04.2025 г.

1. **Исходные данные к курсовой работе:** описание объектов автоматизации, технические характеристики и требования к мониторингу производительности, спецификация проекта, учебно-методическая литература и нормативные акты, материалы периодической печати, интернет-ресурсы.
2. **Перечень подлежащих разработке вопросов:**

Введение

1. Теоретический раздел:

1.1.1. Анализ концепции и назначения crm-систем: основные функции, преимущества использования для малого бизнеса.

1.1.2. История развития crm-систем и современные тенденции в их проектировании.

1.1.3. Особенности реализации аналитики продаж в crm-системах.

1.1.4. Принципы разработки рекомендательных систем для интернет-магазинов: алгоритмы и подходы.

1. Исследовательский раздел:

2.2.1. Анализ существующих CRM-систем, ориентированных на малый бизнес (например, AmoCRM, HubSpot).

2.2.2. Выявление основных потребностей малого интернет-магазина в автоматизации и аналитике.

2.2.3. Сравнительный анализ конкурентных решений: их сильные и слабые стороны.

2.2.4. Формирование требований к функциональности разрабатываемой системы.

1. Практический раздел:

3.1.1. Проектирование структуры базы данных для хранения информации о клиентах, заказах и продажах.

3.1.2. Разработка серверной части системы: реализация API для управления данными.

3.1.3. Разработка пользовательского интерфейса для управления клиентами, заказами и аналитическими данными.

3.1.4. Интеграция модуля аналитики продаж: визуализация ключевых метрик (графики, диаграммы).

3.1.5. Реализация рекомендательной системы на основе данных о продажах и предпочтениях клиентов.

3.1.6. Настройка системы безопасности: аутентификация, авторизация, защита данных.

3.1.7. Тестирование всех компонентов системы и исправление выявленных ошибок.

3.1.8. Разработка документации для пользователей и администраторов системы.

Заключение

Список использованных источников

* 1. **Перечень графического материала:** ER-диаграмма базы данных, UML-диаграммы (Диаграмма прецедентов (use case), Диаграмма классов, Диаграмма последовательности (sequence diagram)), прототипы пользовательского интерфейса (интерфейс для управления клиентами и заказами, дашборд аналитики продаж, интерфейс для отображения рекомендаций товаров.), схема взаимодействия клиентской, серверной и аналитической частей системы, примеры визуализации данных (графики, диаграммы, таблицы).

* 1. **Дата выдачи задания:** \_\_\_ . \_\_\_ . 20\_\_\_ г.

Руководитель / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_ . \_\_\_ . 20\_\_\_ г.

#### СОДЕРЖАНИЕ

**ВВЕДЕНИЕ 4**

ГЛАВА 1. Теоретический раздел 6

1.1. Анализ концепции и назначения CRM-систем: основные функции, преимущества

использования для малого бизнеса 6

1.2. История развития CRM-систем и современные тенденции в их проектировании 7

1.3. Особенности реализации аналитики продаж в CRM-системах 8

1.4. Принципы разработки рекомендательных систем для интернет-магазинов: алгоритмы и

подходы 10

ГЛАВА 2. Исследовательский раздел 14

1. 2.1. Анализ существующих CRM-систем, ориентированных на малый бизнес (например,

AmoCRM, HubSpot). 14

2.2. Выявление основных потребностей малого интернет-магазина в автоматизации и аналитк15

2.3. Сравнительный анализ конкурентных решений: их сильные и слабые стороны 17

2.4. Формирование требований к функциональности разрабатываемой системы 20

ГЛАВА 3. Практический раздел 23

3.1. Проектирование структуры базы данных для хранения информации о клиентах, заказах и продажах 23

3.2. Разработка серверной части системы: реализация API для управления данными. 27

3.3. Разработка пользовательского интерфейса для управления клиентами, заказами и

аналитическими данными 30

3.4. Интеграция модуля аналитики продаж: визуализация ключевых метрик (графики,

диаграммы). 32

3.5. Реализация рекомендательной системы на основе данных о продажах и предпочтениях

клиентов 34

3.6. Настройка системы безопасности: аутентификация, авторизация, защита данных 36

3.7. Тестирование всех компонентов системы и исправление выявленных ошибок 44

3.8. Разработка документации для пользователей и администратора системы 50

|  |  |
| --- | --- |
| **ЗАКЛЮЧЕНИЕ** | **39** |
| **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** | **41** |

**ПРИЛОЖЕНИЕ 43**

# ВВЕДЕНИЕ

Современные реалии электронной коммерции предъявляют повышенные требования к системам управления взаимоотношениями с клиентами (CRM), особенно для субъектов малого бизнеса. В условиях высокой конкуренции малым интернет-магазинам необходимо не только эффективно управлять базой клиентов, заказами и продажами, но и проводить углублённый анализ данных, а также предлагать персонализированные рекомендации, чтобы удерживать клиентов и стимулировать повторные покупки.

Разработка специализированной CRM-системы с интегрированным функционалом аналитики продаж и модулем рекомендаций представляет собой актуальную задачу, направленную на повышение эффективности бизнеспроцессов малого интернет-магазина. Использование современных технологий разработки программного обеспечения, таких как Java (Spring Boot), React, PostgreSQL и средств контейнеризации (Docker), позволяет реализовать масштабируемое, отказоустойчивое и производительное решение.

Целью курсового проекта является проектирование и реализация CRM системы для интернет-магазина с возможностью хранения и анализа данных о клиентах, заказах и продажах, а также формированием индивидуальных товарных рекомендаций на основе предпочтений клиентов и истории покупок.

В ходе выполнения курсового проекта предполагается:

* провести анализ существующих решений в области CRM;
* исследовать потребности малого бизнеса в области автоматизации и аналитики;
* разработать структуру базы данных, архитектуру серверной и клиентской части системы;
* реализовать модули визуализации аналитики и рекомендации товаров;
* обеспечить безопасность системы и удобство её эксплуатации.

Курсовая работа состоит из введения, основной части, заключения, списка использованных источников и приложений. В основной части представлены теоретический раздел, в котором рассматриваются основные концепции и подходы к построению CRM-систем, анализируются их ключевые функции, преимущества и современные тенденции развития, а также особенности реализации аналитики продаж и рекомендательных механизмов. В исследовательском разделе проводится анализ существующих CRM-решений, ориентированных на малый бизнес, выявляются их недостатки и ограничения, формируются функциональные и нефункциональные требования к разрабатываемой системе, обосновывается выбор технологий. Практический раздел включает описание проектирования базы данных, реализацию серверной и клиентской частей системы, разработку модулей аналитики и персонализированных рекомендаций, тестирование работоспособности, а также оценку перспектив внедрения.

Результатом выполнения курсового проекта станет интегрированная CRM система, адаптированная под нужды малого интернет-магазина и включающая современные инструменты анализа и взаимодействия с клиентами.

## ГЛАВА 1. Теоретический раздел

В данном разделе будут рассмотрены основные концепции и подходы к проектированию и сопровождению CRM-систем для интернет-магазинов, включая анализ их функциональных модулей, методы реализации аналитики продаж, принципы построения рекомендательных механизмов и обеспечение актуальности клиентских данных и метрик.

### 1.1. Анализ концепции и назначения CRM-систем: основные функции,

## преимущества использования для малого бизнеса

Система управления взаимоотношениями с клиентами (CRM — от англ. *Customer Relationship Management*) представляет собой программное решение, предназначенное для автоматизации, анализа и оптимизации взаимодействия с клиентами. Основная цель CRM-систем — повышение уровня удовлетворённости клиентов и увеличение продаж за счёт системного подхода к обработке информации о клиентах, их предпочтениях, истории покупок и уровня вовлечённости.

CRM-системы включают в себя следующие ключевые функции:

* ведение базы данных клиентов;
* отслеживание истории взаимодействий и заказов;
* управление воронкой продаж;
* сегментация клиентов;
* автоматизация маркетинговых и сервисных процессов;
* сбор и анализ статистических данных.

Для малого бизнеса внедрение CRM-системы особенно ценно, так как позволяет повысить конкурентоспособность при ограниченных ресурсах. CRM предоставляет владельцам интернет-магазинов инструменты для систематизации работы с клиентской базой, повышения уровня персонализации предложений, своевременного реагирования на запросы клиентов, а также контроля эффективности маркетинговых и торговых стратегий.

Преимущества использования CRM-систем в малом бизнесе:

* Централизованное хранение информации о клиентах и заказах.
* Повышение качества обслуживания за счёт своевременного реагирования на обращения.
* Снижение затрат времени на рутинные операции благодаря автоматизации.
* Повышение лояльности клиентов и уровня удержания за счёт персонализированных коммуникаций.
* Возможность оценки эффективности каналов продаж и маркетинговых кампаний.

Таким образом, грамотное внедрение CRM-системы позволяет малому интернет-магазину не только оптимизировать внутренние процессы, но и выстроить устойчивую модель взаимодействия с клиентами, ориентированную на долгосрочные отношения.

### 1.2. История развития CRM-систем и современные тенденции в их проектировании

Появление CRM-систем восходит к 1980–1990-м годам, когда предприятия начали активно использовать базы данных клиентов и средства автоматизации маркетинга. Первоначально CRM-решения представляли собой простые электронные таблицы и контактные менеджеры, позволявшие систематизировать взаимодействие с клиентами и хранить базовую информацию.

С развитием компьютерных технологий и интернета в 1990-х годах появились первые интегрированные программные продукты, включавшие функции управления продажами, поддержки клиентов и маркетинга. К этому же периоду относятся первые крупные поставщики CRM-решений, такие как Siebel Systems, которые предлагали комплексные решения для среднего и крупного бизнеса.

2000-е годы ознаменовались переходом к облачным технологиям. Компании стали предлагать CRM-системы по модели SaaS (*Software as a Service*), где программное обеспечение размещается на удалённых серверах, а пользователи получают к нему доступ через интернет. Это сделало CRMсистемы доступными и для малого бизнеса. Ярким примером такого подхода стал Salesforce — один из лидеров рынка облачных CRM.

Современные тенденции в проектировании CRM-систем направлены на увеличение гибкости, масштабируемости и интеллектуальности решений. Ключевые направления развития включают:

* **Интеграцию искусственного интеллекта (AI)**: интеллектуальные помощники, прогнозная аналитика, чат-боты.
* **Аналитику в реальном времени**: построение графиков, дашбордов и отчётов на основе актуальных данных.
* **Мобильность и кроссплатформенность**: возможность работы с системой с любого устройства.
* **Открытые API и микросервисную архитектуру**: для облегчения интеграции с другими системами.
* **Высокий уровень персонализации**: адаптация интерфейса и функционала под конкретные нужды бизнеса.
* **Безопасность и соответствие нормативным требованиям**: особенно в условиях усиления законодательства в сфере обработки персональных данных (например, GDPR, ФЗ-152 в России).

Таким образом, CRM-системы прошли путь от простых реестров клиентов до мощных, гибких платформ, способных эффективно поддерживать весь цикл взаимодействия с клиентом. Актуальные тренды подтверждают необходимость учёта аналитики, автоматизации и масштабируемости при разработке современных CRM-решений.

.

### 1.3. Особенности реализации аналитики продаж в CRM-системах

Аналитика продаж — это один из ключевых компонентов современной CRM-системы, обеспечивающий сбор, обработку и визуализацию информации, связанной с деятельностью отдела продаж и поведением клиентов. Целью аналитики является получение объективной картины текущего состояния бизнеса, выявление трендов, закономерностей и проблемных точек, а также формирование оснований для принятия управленческих решений.

В CRM-системах аналитика продаж, как правило, реализуется через следующие инструменты:

* **Дашборды (панели мониторинга)**, отображающие ключевые метрики: количество заказов, средний чек, выручка, количество новых клиентов, процент отказов и пр.
* **Отчёты по сегментам клиентов**: анализ поведения различных групп покупателей.
* **Воронка продаж**: визуализация этапов взаимодействия с клиентом от первичного обращения до завершения сделки.
* **Динамика продаж во времени**: сезонность, темпы роста/снижения, сравнение периодов.
* **ABC и XYZ-анализы**, применяемые для классификации товаров и клиентов по важности и стабильности спроса.
* **Когортный анализ**: оценка поведения групп клиентов, привлечённых в определённый период.

Реализация аналитического модуля требует построения структуры хранения данных, способной обеспечить быстрый доступ к агрегированной информации. Для этого часто используются:

#### • OLAP-хранилища (On-Line Analytical Processing);

* **инструменты визуализации**, такие как Grafana, Apache Superset или компоненты JavaScript-библиотек (например, Chart.js, Recharts);
* **механизмы кэширования данных**, чтобы минимизировать нагрузку на основную базу данных при построении отчётов.

Важной особенностью является необходимость соблюдения принципа актуальности данных. Это означает, что аналитические панели должны обновляться либо в режиме реального времени, либо с минимальной задержкой. Кроме того, данные должны быть достоверными, полноценно обрабатываться с учётом фильтрации, группировки и других операций.

Для малого интернет-магазина наличие встроенного инструментария аналитики позволяет отслеживать эффективность рекламных каналов, оценивать поведение клиентов, корректировать ассортимент и выявлять неликвидные товары. Всё это способствует принятию своевременных решений, основанных на фактах, а не интуиции.

### 1.4. Принципы разработки рекомендательных систем для интернетмагазинов: алгоритмы и подходы

Рекомендательные системы являются важным компонентом современных CRM- и e-commerce-платформ, направленным на повышение продаж и улучшение пользовательского опыта за счёт персонализированных предложений. Их основная задача — предсказать, какие товары могут заинтересовать конкретного пользователя, основываясь на его поведении, предпочтениях и активности других клиентов.

Существует несколько основных подходов к построению рекомендательных систем:

1. **Коллаборативная фильтрация** (Collaborative Filtering) — метод, основанный на анализе поведения пользователей. Система определяет сходство между пользователями или товарами:
   * *User-based*: ищет пользователей с похожими интересами и предлагает то, что понравилось им.
   * *Item-based*: рекомендует товары, которые покупают вместе с теми, что интересовали пользователя.
2. **Контентная фильтрация** (Content-Based Filtering) — метод, при котором рекомендации формируются на основе характеристик товара (цена, категория, бренд и т.п.) и предпочтений пользователя. Алгоритм анализирует, какие свойства были у интересных пользователю товаров, и предлагает аналогичные.
3. **Гибридные методы** — комбинация коллаборативной и контентной фильтрации, сочетающая преимущества обеих моделей. Они обеспечивают более точные и устойчивые рекомендации, особенно при ограниченном объёме данных (проблема "холодного старта").
4. **Бизнес-правила и эвристики** — менее ресурсоёмкие решения, основанные на заранее заданных правилах: «чем чаще покупают вместе», «похожие по характеристикам», «новинки», «популярные товары» и др.

Реализация рекомендательной системы требует:

* сбора данных о действиях пользователей (просмотры, покупки, оценки, клики);
* хранения этих данных в подходящей структуре (например, в логах или специализированных таблицах);
* применения алгоритмов машинного обучения или статистических моделей;
* периодического обучения модели на новых данных.

На практике часто применяются библиотеки и фреймворки, упрощающие реализацию: Apache Mahout, TensorFlow, Surprise (Python), а также собственные алгоритмы, встроенные в систему. В контексте проекта CRM для малого интернет-магазина предпочтительно использовать простой, масштабируемый подход — например, item-based рекомендации на основе истории заказов.

Таким образом, рекомендательные системы помогают интернет-магазинам увеличить средний чек, снизить отток клиентов и улучшить персонализацию предложений, что особенно актуально при ограниченных бюджетах на маркетинг и рекламу.

#### ГЛАВА 2. Исследовательский раздел

В данной главе описаны основные технологии, использованные в проекте интегрированной CRM-системы для малого интернет-магазина, обоснование их выбора и особенности их применения при разработке серверной и клиентской частей, модуля аналитики продаж и рекомендательных механизмов.

#### 2.1. Анализ существующих CRM-систем, ориентированных на малый бизнес

На рынке представлено множество CRM-систем, предназначенных как для крупного, так и для малого и среднего бизнеса. При этом для малых предприятий важны такие характеристики, как простота внедрения, удобство интерфейса, наличие базовой аналитики, доступная цена и возможности интеграции с другими сервисами (интернет-магазином, маркетплейсами, мессенджерами и т.д.).

Рассмотрим несколько популярных CRM-систем, ориентированных на малый бизнес:

#### AmoCRM

AmoCRM — одна из самых известных CRM в сегменте малого бизнеса. Её основные особенности:

* ориентация на автоматизацию продаж;
* наглядная воронка сделок;
* интеграции с мессенджерами (WhatsApp, Telegram), телефонией и почтой;
* базовая аналитика по менеджерам, источникам лидов и эффективности каналов;
* встроенные чат-боты и виджеты для сайта;
* расширяемость за счёт API и виджетов из собственного каталога.

Недостатки AmoCRM — ограниченная функциональность бесплатных тарифов, закрытый исходный код, зависимость от облака и ограниченные возможности по кастомизации аналитики под специфические бизнес-процессы.

**HubSpot CRM**

Бесплатная CRM от американской компании HubSpot предлагает:

* управление клиентской базой и сделками;
* задачи, заметки, автоматизация коммуникации;
* отчёты по эффективности сделок;
* единое окно взаимодействия с клиентом.

Преимуществом является высокая интеграция с другими инструментами HubSpot (email-маркетинг, поддержка клиентов, CMS). Недостатки — англоязычный интерфейс, меньшая адаптированность под российский рынок и локальные требования (например, отсутствие интеграции с 1С или онлайнкассами).

#### Bitrix24

Bitrix24 — комплексное решение с CRM, проектным управлением, телефонией и корпоративным порталом. Подходит для малого и среднего бизнеса, имеет как облачную, так и коробочную версии. Преимущества:

* русскоязычный интерфейс;
* богатый функционал даже в бесплатной версии;
* гибкая настройка бизнес-процессов;
* интеграция с интернет-магазинами, телефонией, 1С.

Однако сложность интерфейса, высокая нагрузка на систему при большом объёме данных и периодические технические сбои могут затруднить использование для микробизнеса без ИТ-поддержки.

Таким образом, существующие CRM-системы предлагают широкий набор функций, однако они не всегда полностью удовлетворяют потребности небольших интернет-магазинов. Многие решения требуют адаптации, абонентской платы или не обеспечивают глубокой аналитики и рекомендаций, что создаёт предпосылки для разработки собственной системы с индивидуальным функционалом.

#### 2.2. Выявление основных потребностей малого интернет-магазина в автоматизации и аналитике

Малые интернет-магазины, как правило, ограничены в ресурсах: у них небольшие команды, ограниченный бюджет и минимальная ИТ-инфраструктура. Однако для эффективной конкуренции на рынке им необходимо обеспечить высокое качество обслуживания клиентов, быструю обработку заказов и грамотное использование данных для принятия управленческих решений.

Ключевые потребности малого интернет-магазина в автоматизации и аналитике можно разделить на несколько групп:

1. **Управление клиентской базой**

o хранение контактной информации, истории заказов и взаимодействия с клиентами;

o сегментация клиентов (например, по частоте покупок или объёму заказов);

o быстрый поиск и фильтрация клиентов; o отслеживание текущего статуса клиентов воронки продаж.

1. **Автоматизация обработки заказов** o приём, изменение и отслеживание заказов;

o связывание заказов с клиентами;

o автоматические уведомления (например, о подтверждении заказа или изменении статуса);

o интеграция с платёжными и логистическими сервисами.

1. **Базовая и расширенная аналитика** o отчёты по продажам за период, по категориям товаров, по клиентам;

o ключевые показатели: средний чек, выручка, конверсия, возвраты; o анализ повторных покупок и эффективности маркетинговых кампаний; o визуализация данных в виде графиков и диаграмм для быстрого анализа.

1. **Рекомендательная система** o формирование персонализированных предложений клиентам на основе истории покупок; o повышение среднего чека за счёт перекрёстных и дополнительных продаж;

o автоматизация рассылок с рекомендованными товарами.

1. **Интерфейс и доступность** o интуитивно понятный веб-интерфейс для сотрудников; o доступ через браузер без необходимости установки ПО; o адаптация под мобильные устройства.
2. **Безопасность и контроль доступа** o разграничение прав доступа для различных ролей (администратор, менеджер);

o безопасное хранение персональных данных клиентов; o защита от несанкционированного доступа.

Таким образом, разработка CRM-системы, адаптированной под нужды малого интернет-магазина, должна предусматривать не только базовые функции работы с клиентами и заказами, но и возможности аналитики и персонализации, что обеспечивает рост продаж и удержание клиентов.

**2.3. Сравнительный анализ конкурентных решений: их сильные и слабые**

#### стороны

Для выбора оптимального подхода к разработке собственной CRM системы важно провести сравнительный анализ существующих решений с точки зрения их применимости в условиях малого интернет-магазина. Ниже приведена таблица, содержащая основные характеристики трёх популярных CRM-систем: AmoCRM, HubSpot и Bitrix24.

|  |  |  |  |
| --- | --- | --- | --- |
| Категория | AmoCRM | HubSpot | Bitrix24 |
| Сильные стороны | Простой интерфейс, интеграции с телефонией и мессенджерами, настройка воронок, локализация на русский. | Бесплатная версия, интеграция с маркетингом, автоматизация процессов, открытая экосистема. | Все-in-one (CRM + коммуникации + проекты), бесплатный тариф, локализация на русский. |
| Слабые стороны | Ограниченная аналитика, дорогое масштабирование, слабая автоматизация без доп. сервисов. | Дорогие платные модулы, ограниченная кастомизация, сложность для новичков. | Перегруженный интерфейс, низкая скорость работы, слабая аналитика в бесплатной версии |
| Возможности | Рост спроса в SME-сегменте, развитие AI-инструментов, экспансия в СНГ. | Рост спроса на маркетинговую автоматизацию, развитие AI, расширение на международные рынки. | Рост спроса на комплексные решения, AI-инструменты, экспансия в СНГ. |
| Угрозы | Конкуренция (HubSpot, Bitrix24), требования GDPR, высокая стоимость масштабирования. | Конкуренция (Salesforce, Zoho), сложности с масштабированием, зависимость от интеграций. | Конкуренция (Asana, HubSpot), сложности с крупными клиентами, устаревшее мобильное приложение |

**Общие наблюдения:**

* Все три решения покрывают базовые задачи CRM: учёт клиентов и сделок, коммуникации, отчётность.
* Встроенные модули аналитики либо упрощены, либо требуют расширенных платных подписок.
* Рекомендательные механизмы либо отсутствуют, либо предоставляются сторонними интеграциями, а не встроены в ядро системы.
* Сложность настройки и кастомизации может стать барьером для малого бизнеса с ограниченным ИТ-персоналом.

Таким образом, ни одно из рассмотренных решений не сочетает в себе в полной мере простоту, встроенную аналитику и рекомендации, доступность и локальную адаптацию. Это подтверждает целесообразность разработки специализированной CRM-системы, максимально соответствующей требованиям малого интернет-магазина.

**2.4. Формирование требований к функциональности разрабатываемой**

Разработка CRM-системы для малого интернет-магазина требует чёткого понимания функциональных требований, которые будут удовлетворять потребности бизнеса и улучшать взаимодействие с клиентами. Важно, чтобы система была простая в использовании, масштабируемая, обеспечивала возможность быстрой интеграции с другими инструментами и могла быть адаптирована под специфические задачи компании.

К основным функциональным требованиям к системе можно отнести:

1. **Управление клиентами и сделками**

o Хранение информации о клиентах (контактные данные, история покупок, взаимодействий и предпочтений).

o Сегментация клиентов для персонализированного подхода

(например, по статусу, частоте покупок, сумме покупок).

o Ведение и управление сделками в воронке продаж с возможностью отслеживания их статуса.

o Функция отправки уведомлений клиентам о новых предложениях или изменениях в статусе их заказа.

1. **Автоматизация работы с заказами**

o Приём, обработка и отслеживание заказов в реальном времени. o Интеграция с внешними сервисами (платёжными системами, логистическими компаниями) для автоматического обновления информации.

* + Генерация и отправка электронных квитанций и уведомлений о статусе заказа.
  + Управление возвратами и отменами заказов.

1. **Аналитика продаж**

o Формирование отчётов по продажам по различным параметрам: категориям товаров, клиентам, времени.

o Визуализация данных с помощью графиков, диаграмм и таблиц для упрощения анализа.

* + Инструменты для оценки эффективности маркетинговых кампаний и источников трафика. o Прогнозирование спроса на основе исторических данных и текущих трендов.

1. **Рекомендательная система**

o Формирование персонализированных рекомендаций на основе истории покупок клиентов.

o Возможности перекрёстных и дополнительных продаж для увеличения среднего чека.

* + Рекомендации по товарам на главной странице, в email-рассылках и в корзине клиента. o Алгоритмы для автоматического обновления рекомендаций с учётом изменений в предпочтениях пользователей и сезонности товаров.

1. **Интерфейс и взаимодействие с пользователем**

o Простота и удобство интерфейса для пользователей (менеджеров и администраторов).

* + Интуитивно понятный доступ к важным данным и отчётам.
  + Мобильная версия интерфейса для работы с системой с любого устройства. o Возможности кастомизации интерфейса под нужды конкретного магазина (добавление полей, настройка рабочих процессов).

1. **Безопасность и контроль доступа**

o Разграничение прав доступа для различных категорий пользователей (администратор, менеджер, аналитик).

* + - Защита личных данных клиентов и безопасная обработка финансовых транзакций.
    - Логирование всех действий в системе для обеспечения прозрачности и аудита. o Интеграция с двухфакторной аутентификацией и другими методами защиты данных.

1. **Интеграция с внешними сервисами**

o Интеграция с популярными платёжными системами, интернетмагазинами и маркетплейсами для синхронизации заказов и клиентской информации.

o Подключение API для внешних аналитических и рекламных платформ.

o Поддержка интеграции с популярными почтовыми сервисами и мессенджерами для автоматической рассылки уведомлений.

Функциональные требования к системе должны быть гибкими, чтобы их можно было легко адаптировать в процессе работы и учёта изменения потребностей бизнеса. Важно, чтобы система могла расти и расширяться по мере развития интернет-магазина, обеспечивая поддержку новых функциональных возможностей и интеграций

## ГЛАВА 3. Практический раздел

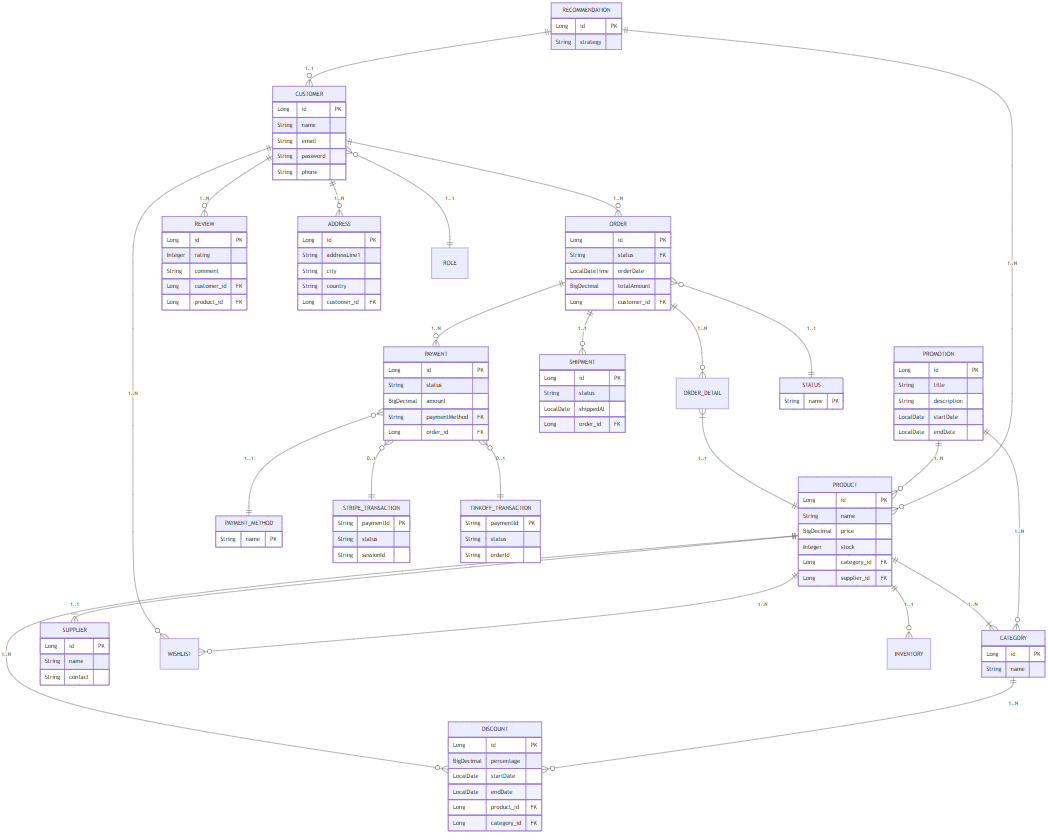
В практическом разделе будет описана реализация интегрированной CRM-системы для малого интернет-магазина, включая настройку серверного окружения и контейнеризацию приложения (Docker), разработку REST-API на Spring Boot и frontend-части на React/Redux, реализацию механизмов автоматизированной загрузки и синхронизации данных о клиентах и заказах, интеграцию модуля аналитики продаж с визуализацией ключевых метрик, внедрение рекомендательного движка, оптимизацию структуры базы данных PostgreSQL для повышения производительности, проведение нагрузочного тестирования и настройку CI/CD (GitLab CI или Jenkins), а также подготовку эксплуатационной и пользовательской документации.

### 3.1. Проектирование структуры базы данных для хранения информации о клиентах, заказах и продажах

Проектирование базы данных является основным этапом в разработке CRM-системы, поскольку от структуры хранения данных зависит производительность системы, а также удобство работы с информацией. В данном разделе рассматривается проектирование базы данных для хранения информации о клиентах, заказах и продажах интернет-магазина.

##### Основные сущности и их атрибуты

Для эффективного хранения данных необходимо выделить несколько ключевых сущностей. В нашем случае это:



##### Связи между сущностями

* **Клиенты и заказы**: Один клиент может иметь несколько заказов, но каждый заказ принадлежит только одному клиенту. Это устанавливается через внешний ключ **customer\_id** в таблице заказов.
* **Заказы и товары**: Один заказ может содержать несколько товаров, и каждый товар может быть частью нескольких заказов. Это много ко многим, и такая связь реализуется через промежуточную таблицу **Sales**, которая связывает заказ с конкретными товарами.
* **Продажи и товары**: Каждый товар, проданный в рамках заказа, будет отображаться в таблице **Sales**, где будет указано количество и цена продажи.

##### Пример ER-диаграммы

Для наглядности, структуру данных можно представить в виде **ER-диаграммы** (диаграмма сущность-связь), которая отражает связи между таблицами и их атрибутами. В ER-диаграмме будут показаны:

* Сущности (таблицы);
* Атрибуты каждой сущности;
* Связи между сущностями (например, клиент имеет заказы, заказ содержит товары).

Проектирование правильной структуры базы данных позволяет гарантировать, что система будет эффективно обрабатывать запросы, обеспечивать целостность данных и поддерживать масштабируемость для дальнейшего расширения.

### 3.2. Разработка серверной части системы: реализация API для управления данными

Серверная часть системы является основным компонентом, обеспечивающим взаимодействие между клиентом и базой данных, а также реализацию бизнес-логики. В данном разделе описан процесс разработки RESTful API для управления данными о клиентах, заказах и продажах.

##### Архитектура сервера

Для разработки серверной части используется **Java** с фреймворком **Spring Boot**, который позволяет эффективно создавать веб-приложения с минимальными усилиями. Архитектура API основана на принципах REST (Representational State Transfer), что обеспечивает гибкость и масштабируемость приложения.

Основные компоненты архитектуры:

* **Controller**: Отвечает за обработку HTTP-запросов и взаимодействие с сервисами.
* **Service**: Реализует бизнес-логику приложения.
* **Repository**: Взаимодействует с базой данных через JPA (Java Persistence API) для выполнения операций CRUD (создание, чтение, обновление, удаление).

**Реализация CRUD-операций для сущностей**

Для каждой из сущностей (Клиенты, Заказы, Продажи) необходимо реализовать набор стандартных операций CRUD через API. Это позволяет клиентским приложениям (например, frontend-части) отправлять запросы на сервер для работы с данными.

1. **Клиенты (Customers)** 
   1. **Получение информации о клиенте** o HTTP-метод: GET o URL: /api/customers/{id}

o Описание: Получение информации о клиенте по его уникальному идентификатору. o Пример:

@GetMapping("/api/customers/{id}")

public ResponseEntity<Customer> getCustomer(@PathVariable Long id) {

Customer customer = customerService.getCustomerById(id); return ResponseEntity.ok(customer); }

* 1. **Создание нового клиента** o HTTP-метод: POST

o URL: /api/customers o Описание: Добавление нового клиента в систему. o Пример:

@PostMapping("/api/customers")

public ResponseEntity<Customer> createCustomer(@RequestBody

Customer customer) {

Customer createdCustomer = customerService.createCustomer(customer); return

ResponseEntity.status(HttpStatus.CREATED).body(createdCustomer); }

* 1. **Обновление данных клиента** o HTTP-метод: PUT o URL: /api/customers/{id}

o Описание: Обновление данных клиента по его уникальному идентификатору. o Пример:

@PutMapping("/api/customers/{id}")

public ResponseEntity<Customer> updateCustomer(@PathVariable

Long id, @RequestBody Customer customer) {

Customer updatedCustomer = customerService.updateCustomer(id, customer); return ResponseEntity.ok(updatedCustomer); }

* 1. **Удаление клиента** o HTTP-метод: DELETE

o URL: /api/customers/{id} o Описание: Удаление клиента из системы. o Пример:

@DeleteMapping("/api/customers/{id}")

public ResponseEntity<Void> deleteCustomer(@PathVariable Long id)

{ customerService.deleteCustomer(id); return ResponseEntity.noContent().build(); }

**Заказы (Orders)**

Для работы с заказами API будет включать аналогичные методы:

* 1. **Получение всех заказов клиента**

o HTTP-метод: GET o URL: /api/orders/customer/{customerId} o Описание: Получение списка всех заказов для клиента. o Пример:

@GetMapping("/api/orders/customer/{customerId}") public ResponseEntity<List<Order>> getOrdersByCustomer(@PathVariable Long customerId) { List<Order> orders =

orderService.getOrdersByCustomer(customerId);

return ResponseEntity.ok(orders); }

1. **Создание нового заказа** o HTTP-метод: POST o URL: /api/orders o Описание: Создание нового заказа. o Пример:

@PostMapping("/api/orders")

public ResponseEntity<Order> createOrder(@RequestBody Order order) {

Order createdOrder = orderService.createOrder(order); return

ResponseEntity.status(HttpStatus.CREATED).body(createdOrder); }

1. **Обновление статуса заказа** o HTTP-метод: PUT o URL: /api/orders/{id} o Описание: Обновление статуса заказа. o Пример:

@PutMapping("/api/orders/{id}")

public ResponseEntity<Order> updateOrder(@PathVariable Long id,

@RequestBody Order order) {

Order updatedOrder = orderService.updateOrder(id, order); return ResponseEntity.ok(updatedOrder); }

1. **Удаление заказа** o HTTP-метод: DELETE o URL: /api/orders/{id} o Описание: Удаление заказа. o Пример: @DeleteMapping("/api/orders/{id}")

public ResponseEntity<Void> deleteOrder(@PathVariable Long id) { orderService.deleteOrder(id); return ResponseEntity.noContent().build(); }

**Продажи (Sales)**

Аналогичные операции CRUD можно реализовать и для сущности **Продажи**. Например, создание записи о продаже товара и получение информации о всех продажах для конкретного заказа.

##### Безопасность API

Для обеспечения безопасности взаимодействия между клиентом и сервером важно реализовать механизм аутентификации и авторизации. В данном проекте для этой цели используется **JWT (JSON Web Token)**, который позволит клиенту безопасно авторизоваться и получать доступ к защищённым данным.

* **JWT токен** будет генерироваться при логине пользователя.
* При каждом запросе токен будет передаваться в заголовке Authorization.

Пример проверки токена на сервере:

@PreAuthorize("hasRole('ROLE\_ADMIN')")

@GetMapping("/api/orders/{id}")

public ResponseEntity<Order> getOrder(@PathVariable Long id) {

Order order = orderService.getOrderById(id); return ResponseEntity.ok(order); }

В данной реализации, администратор системы будет иметь доступ к управлению всеми заказами и данными о клиентах.

### 3.3. Разработка пользовательского интерфейса для управления клиентами,

#### заказами и аналитическими данными

Пользовательский интерфейс (UI) является важной частью CRM-системы, поскольку он предоставляет конечному пользователю возможность взаимодействовать с системой. В данном разделе описан процесс разработки frontend-части системы с использованием **React** и **Redux** для управления состоянием приложения.

Frontend-система состоит из нескольких основных компонентов:

* **React** – основная библиотека для создания пользовательского интерфейса. React используется для построения компонентов, которые обеспечивают динамическое обновление интерфейса без перезагрузки страницы.
* **Redux** – используется для управления состоянием приложения. Redux позволяет централизованно хранить данные, что облегчает синхронизацию состояния между различными компонентами.

Структура frontend-части:

* **Components**: Компоненты, которые отображают данные пользователю (например, таблицы клиентов, карточки заказов, графики аналитики).
* **Containers**: Компоненты, которые содержат логику для взаимодействия с API и управляют состоянием.
* **Actions/Reducers**: Механизм Redux для управления состоянием приложения.

**Страницы и компоненты интерфейса**

Для эффективного управления клиентами, заказами и аналитическими данными, пользовательский интерфейс должен включать следующие основные страницы и компоненты:

**Страница клиентов**

* **Список клиентов**: Отображение таблицы с основными данными о клиентах, такими как имя, контактная информация и статус.
* **Поиск клиентов**: Форма для фильтрации клиентов по различным критериям (например, по имени, email, статусу).
* **Просмотр/редактирование клиента**: Возможность просматривать детали клиента и редактировать информацию о нем.
* **Создание клиента**: Форма для добавления нового клиента в систему.

Пример компонента для отображения списка клиентов: const CustomersList = ({ customers, onEdit, onDelete }) => { return (

<div>

<h2>Список клиентов</h2>

<table>

<thead>

<tr>

<th>Имя</th>

<th>Email</th>

<th>Телефон</th>

<th>Статус</th>

<th>Действия</th>

</tr>

</thead>

<tbody>

{customers.map((customer) => (

<tr key={customer.id}>

<td>{customer.firstName} {customer.lastName}</td>

<td>{customer.email}</td>

<td>{customer.phone}</td>

<td>{customer.status}</td>

<td>

<button onClick={() =>

onEdit(customer.id)}>Редактировать</button>

<button onClick={() => onDelete(customer.id)}>Удалить</button>

</td>

</tr>

))}

</tbody>

</table>

</div>

);

};

#### Страница заказов

* **Список заказов**: Отображение таблицы с заказами, включая информацию о статусе, сумме заказа, клиенте.
* **Просмотр деталей заказа**: Возможность просматривать заказ с подробной информацией о содержимом (товары, количество, цена).
* **Редактирование статуса заказа**: Форма для изменения статуса заказа (например, "В обработке", "Отправлен", "Доставлен").

Пример компонента для отображения списка заказов: const OrdersList = ({ orders, onStatusChange }) => { return (

<div>

<h2>Список заказов</h2>

<table>

<thead>

<tr>

<th>Клиент</th>

<th>Дата</th>

<th>Статус</th>

<th>Итоговая сумма</th>

<th>Действия</th>

</tr>

</thead>

<tbody>

{orders.map((order) => (

<tr key={order.id}>

<td>{order.customer.name}</td>

<td>{order.date}</td>

<td>{order.status}</td>

<td>{order.totalAmount}</td>

<td>

<button onClick={() => onStatusChange(order.id)}>Изменить статус</button>

</td>

</tr>

))}

</tbody>

</table>

</div>

);

};

#### Страница аналитики

* **Графики и диаграммы**: визуализация ключевых метрик, таких как общая сумма продаж, количество заказов по времени, наиболее продаваемые товары.
* **Фильтрация данных**: Возможность фильтровать данные по различным критериям (например, по времени, категориям товаров).

Для реализации аналитики можно использовать библиотеку **Chart.js** для отображения графиков.

Пример компонента для отображения графика продаж:

import { Line } from 'react-chartjs-2';

const SalesAnalytics = ({ salesData }) => { const data = {

labels: salesData.map(item => item.date), datasets: [

{

label: 'Продажи',

data: salesData.map(item => item.totalAmount), borderColor: 'rgba(75, 192, 192, 1)', fill: false,

},

],

};

return (

<div>

<h2>Аналитика продаж</h2>

<Line data={data} />

</div>

);

};

Для централизованного управления состоянием приложения используется **Redux**. Все данные о клиентах, заказах и продажах хранятся в **store**, и компоненты взаимодействуют с этим состоянием через **actions** и **reducers**.

Пример action для загрузки клиентов: export const loadCustomers = () => { return async (dispatch) => {

const response = await fetch('/api/customers'); const customers = await response.json();

dispatch({ type: 'LOAD\_CUSTOMERS', payload: customers });

};

};

Reducer для обновления состояния клиентов:

const customersReducer = (state = [], action) => { switch (action.type) { case 'LOAD\_CUSTOMERS':

return action.payload; default:

return state;

}

};

Frontend-система взаимодействует с серверной частью через API. Для этого используются HTTP-запросы (например, с помощью **fetch** или **axios**). Все данные, полученные от сервера, сохраняются в **Redux store** и отображаются на страницах.

Пример вызова API для получения данных о клиентах:

const fetchCustomers = async () => { const response = await fetch('/api/customers'); const customers = await response.json();

dispatch({ type: 'LOAD\_CUSTOMERS', payload: customers });

};

### 3.4. Интеграция модуля аналитики продаж: визуализация ключевых метрик (графики, диаграммы)

Для эффективного мониторинга и анализа продаж в CRM-системе предусмотрен модуль аналитики. Он предоставляет пользователю визуальные инструменты для отслеживания ключевых показателей, таких как общий объем продаж, динамика заказов по времени, а также другие важные метрики, что помогает в принятии решений для улучшения бизнес-процессов.

В модуле аналитики должны отображаться следующие ключевые метрики:

* **Общий объем продаж**: Сумма всех заказов за определенный период.
* **Число заказов**: Количество выполненных заказов.
* **Средний чек**: Средняя стоимость одного заказа.
* **Топовые товары**: Товары, которые чаще всего покупаются.
* **Продажи по категориям**: Данные о том, какие категории товаров наиболее популярны.

Для отображения графиков и диаграмм используется библиотека Chart.js. Она позволяет создавать различные виды диаграмм, такие как линейные графики, гистограммы, круговые диаграммы и т.д., что делает визуализацию данных понятной и наглядной.

Пример интеграции линейного графика для отображения объема продаж по времени:

import { Line } from 'react-chartjs-2';

const SalesLineChart = ({ salesData }) => { const data = { labels: salesData.map(item => item.date), datasets: [

{

label: 'Объем продаж', data: salesData.map(item => item.totalAmount), borderColor: 'rgba(75, 192, 192, 1)',

fill: false,

},

],

};

return (

<div>

<h3>Объем продаж по дням</h3>

<Line data={data} />

</div>

);

};

Статистические данные

Для обеспечения точности данных и удобства работы с ними, в системе предусмотрена обработка статистических данных на серверной стороне. Каждый запрос к API для получения аналитической информации будет возвращать данные, агрегированные по заданному промежутку времени (например, по месяцам, неделям или дням).

Пример запроса на сервер для получения данных о продажах по месяцам:

const fetchMonthlySalesData = async () => { const response = await fetch('/api/sales/monthly'); const salesData = await response.json(); dispatch({ type: 'LOAD\_SALES\_DATA', payload: salesData });

};

Ответ от сервера может содержать данные о продажах по месяцам:

[

{ "month": "Январь", "totalAmount": 5000 },

{ "month": "Февраль", "totalAmount": 6500 },

{ "month": "Март", "totalAmount": 7000 }

]

Графики отображают ключевые метрики в виде наглядных диаграмм, что позволяет пользователю быстро выявить тренды и аномалии в данных. Пример отображения гистограммы для визуализации объемов продаж по месяцам:

import { Bar } from 'react-chartjs-2';

const SalesBarChart = ({ salesData }) => { const data = { labels: salesData.map(item => item.month),

datasets: [

{

label: 'Объем продаж по месяцам', data: salesData.map(item => item.totalAmount), backgroundColor: 'rgba(75, 192, 192, 0.2)', borderColor: 'rgba(75, 192, 192, 1)', borderWidth: 1,

},

],

};

return (

<div>

<h3>Продажи по месяцам</h3>

<Bar data={data} />

</div>

);

};

Для того чтобы пользователи могли анализировать данные в разрезе различных периодов времени или категорий, в систему встроены инструменты фильтрации. Пользователи могут выбирать временные интервалы (например, за последние 7 дней, месяц или год), а также фильтровать данные по категориям товаров или регионам.

Пример компонента для фильтрации данных: const DateFilter = ({ onDateRangeChange }) => { const handleChange = (event) => { const selectedRange = event.target.value; onDateRangeChange(selectedRange);

};

return (

<div>

<label>Выберите период:</label>

<select onChange={handleChange}>

<option value="last7Days">Последние 7 дней</option>

<option value="lastMonth">Последний месяц</option>

<option value="lastYear">Последний год</option>

</select>

</div>

);

};

Для удобства пользователей графики и диаграммы могут быть интерактивными. Например, можно добавить возможность при наведении курсора на элементы диаграмм показывать подробную информацию (например, точное количество продаж за определенный день или месяц). Это можно реализовать с помощью событий в **Chart.js**, таких как onHover.

Пример настройки интерактивности: const options = { onHover: (event, chartElement) => { if (chartElement.length > 0) { const datasetIndex = chartElement[0].\_datasetIndex; const index = chartElement[0].\_index; const value = chartElement[0].\_model.label;

// Действия при наведении

}

},

};

<Line data={data} options={options} />;

**3.5. Реализация рекомендательной системы на основе данных о продажах**

#### и предпочтениях клиентов

Рекомендательные системы играют важную роль в интернет-магазинах, так как помогают повысить конверсии и улучшить пользовательский опыт, предлагая клиентам персонализированные товары на основе их интересов и поведения. В рамках разрабатываемой CRM-системы реализуется модуль рекомендательной системы, который будет использовать данные о продажах и предпочтениях клиентов для генерации рекомендаций.

Для реализации рекомендательной системы можно использовать несколько подходов. В данном проекте будут применяться следующие алгоритмы:

1. **Коллаборативная фильтрация (Collaborative Filtering)** – это один из самых популярных методов в рекомендательных системах. Он работает на основе данных о поведении пользователей, таких как покупки или рейтинг товаров. В зависимости от предпочтений других пользователей, система будет рекомендовать товары, которые были популярны среди схожих пользователей.

Пример коллаборативной фильтрации: если пользователь купил товары A и B, то системе будет рекомендовано предложить товар C, который был куплен пользователями с похожими предпочтениями.

1. **Контентная фильтрация (Content-Based Filtering)** – этот метод основан на анализе характеристик товаров, которые интересуют пользователя. Рекомендации формируются на основе того, какие атрибуты (категория, бренд, цена) товаров были наиболее привлекательны для клиента в прошлом.

Пример контентной фильтрации: если клиент часто покупает спортивные товары, система будет рекомендовать другие товары из категории спорта.

1. **Гибридные методы** – это комбинация различных алгоритмов, например, коллаборативной и контентной фильтрации, для повышения точности рекомендаций.

**Сбор данных для рекомендаций**

Для генерации рекомендаций необходимо собрать данные о покупках пользователей, их предпочтениях и взаимодействиях с товарами. Важно, чтобы система могла учитывать:

* Историю покупок клиента.
* Просмотренные товары.
* Оценки товаров (если есть).
* Поведение клиента (например, добавление товара в корзину, но без оформления покупки).

В структуре базы данных это будет отражено через таблицы, которые связывают клиентов и товары, например: CREATE TABLE customer\_purchases ( customer\_id INT REFERENCES customers(id), product\_id INT REFERENCES products(id), purchase\_date TIMESTAMP,

PRIMARY KEY (customer\_id, product\_id)

);

**Модуль рекомендации на серверной стороне**

На сервере будет реализован API для предоставления рекомендованных товаров для каждого клиента. Он будет использовать алгоритмы, описанные выше, чтобы на основе данных о поведении клиента генерировать список товаров.

Пример API для получения рекомендаций: @GetMapping("/recommendations/{customerId}")

public List<Product> getRecommendations(@PathVariable("customerId") Long customerId) {

List<Product> recommendedProducts =

recommendationService.getRecommendations(customerId); return recommendedProducts;

}

Реализация метода getRecommendations будет включать вызов логики коллаборативной фильтрации или контентной фильтрации, чтобы получить соответствующие товары для клиента.

**Коллаборативная фильтрация**

Коллаборативная фильтрация основывается на матрице взаимодействий пользователей с товарами (например, покупки или просмотры). Для реализации такого подхода можно использовать **алгоритм ближайших соседей**. Этот метод измеряет схожесть между пользователями или товарами и на основе этой схожести генерирует рекомендации.

Пример расчета схожести между пользователями с помощью косинусного сходства:

public double calculateCosineSimilarity(Map<Integer, Double> user1,

Map<Integer, Double> user2) { double dotProduct = 0.0; double normA = 0.0; double normB = 0.0;

for (Map.Entry<Integer, Double> entry : user1.entrySet()) { int itemId = entry.getKey(); double rating1 = entry.getValue();

double rating2 = user2.getOrDefault(itemId, 0.0);

dotProduct += rating1 \* rating2; normA += Math.pow(rating1, 2); normB += Math.pow(rating2, 2);

}

return dotProduct / (Math.sqrt(normA) \* Math.sqrt(normB)); }

Этот метод будет использоваться для вычисления схожести между клиентами на основе их покупок и предложений товаров, наиболее похожих на их предпочтения.

#### Контентная фильтрация

Контентная фильтрация будет учитывать атрибуты товаров, такие как категория, бренд, цвет и другие характеристики. Например, если клиент часто покупает товары в категории "спорт", система будет рекомендовать другие товары из этой категории.

Пример алгоритма контентной фильтрации:

public List<Product> getContentBasedRecommendations(Long customerId) {

Customer customer =

customerRepository.findById(customerId).orElseThrow();

List<String> likedCategories = customer.getLikedCategories(); return productRepository.findByCategoriesIn(likedCategories); }

Здесь клиенту рекомендованы товары, которые соответствуют его предпочтениям по категориям.

#### Гибридный метод

Для повышения точности рекомендаций можно комбинировать оба подхода: коллаборативную фильтрацию и контентную фильтрацию. В этом случае для каждого клиента будут генерироваться два списка рекомендаций: один на основе схожести с другими пользователями, а другой на основе схожести товаров с теми, которые клиент уже покупал. Затем эти списки могут быть объединены с учетом весовых коэффициентов для каждого алгоритма.

Пример комбинированного метода:

public List<Product> getHybridRecommendations(Long customerId) {

|  |  |
| --- | --- |
| List<Product> collaborativeRecommendations  getCollaborativeRecommendations(customerId); | = |
| List<Product> contentBasedRecommendations  getContentBasedRecommendations(customerId);    // Объединение списков и устранение дубликатов | = |
| Set<Product> recommendations = | new |

HashSet<>(collaborativeRecommendations); recommendations.addAll(contentBasedRecommendations);

return new ArrayList<>(recommendations);

}

**Тестирование рекомендательной системы**

Для оценки качества рекомендательной системы можно использовать различные метрики, такие как **точность** (precision), **полнота** (recall) и **F1-мера**. Эти метрики помогут измерить, насколько правильно система предлагает товары, соответствующие интересам пользователей.

Пример метрики точности:

public double calculatePrecision(List<Product> recommended, List<Product>

relevant) {

Set<Product> recommendedSet = new HashSet<>(recommended); Set<Product> relevantSet = new HashSet<>(relevant);

int truePositive = (int)

recommendedSet.stream().filter(relevantSet::contains).count(); return (double) truePositive / recommended.size();

}

**3.6. Настройка системы безопасности: аутентификация, авторизация,**

**защита данных**

В процессе разработки CRM-системы для интернет-магазина необходимо обеспечить высокий уровень безопасности, особенно учитывая, что система будет обрабатывать личные данные клиентов, информацию о заказах и платежах. Важными аспектами являются аутентификация, авторизация и защита данных. В данном разделе описаны основные подходы, используемые для обеспечения безопасности системы.

Аутентификация — это процесс проверки подлинности пользователя. В данной CRM-системе для аутентификации будет использоваться **JWT (JSON Web Token)** — стандарт, широко применяемый для защиты REST API. JWT позволяет безопасно передавать информацию о пользователе между сервером и клиентом.

Процесс аутентификации включает следующие этапы:

1. **Вход пользователя**: Пользователь вводит свои учетные данные (логин и пароль).
2. **Проверка данных**: Система проверяет правильность введенных данных, сравнивая их с данными в базе данных.
3. **Генерация токена**: После успешной проверки сервер генерирует JWT, который содержит информацию о пользователе (например, его ID и роли).
4. **Возвращение токена**: Токен отправляется клиенту, который сохраняет его для дальнейших запросов.

Пример реализации аутентификации с использованием JWT в Spring Boot:

public String authenticateUser(LoginRequest loginRequest) {

Authentication authentication = authenticationManager.authenticate( new UsernamePasswordAuthenticationToken( loginRequest.getUsername(), loginRequest.getPassword()

)

);

SecurityContextHolder.getContext().setAuthentication(authentication); return jwtTokenProvider.generateToken(authentication);

}

В этом примере, после успешной аутентификации пользователя, генерируется JWT, который затем отправляется обратно клиенту.

Авторизация — это процесс проверки прав доступа пользователя к определенным ресурсам в системе. Для реализации авторизации в CRM-системе будет использоваться **роль-подход** (Role-based access control, RBAC). Каждый пользователь будет иметь определенную роль, которая будет определять его права.

Роли могут включать, например:

* **Администратор**: имеет полный доступ ко всем функциям системы, включая управление пользователями, товарами и заказами.
* **Менеджер**: может управлять только заказами и продуктами, но не имеет доступа к настройкам безопасности.
* **Пользователь**: имеет доступ только к собственным данным, таким как история покупок.

Пример реализации авторизации с использованием аннотаций Spring Security:

@PreAuthorize("hasRole('ADMIN')") public void deleteUser(Long userId) { userRepository.deleteById(userId);

}

В этом примере метод deleteUser может быть вызван только пользователями, обладающими ролью ADMIN.

Для защиты данных в CRM-системе будут использоваться несколько подходов: Важно обеспечить шифрование конфиденциальных данных, таких как пароли, личная информация клиентов и финансовые данные. Для хранения паролей будет использоваться **bcrypt** — алгоритм хеширования паролей с солением, который является стандартом для обеспечения безопасности.

Пример хеширования пароля с использованием bcrypt в Spring Security:

@Bean

public PasswordEncoder passwordEncoder() { return new BCryptPasswordEncoder();

}

Пароли будут хешироваться перед их сохранением в базе данных. Когда пользователь попытается войти в систему, введенный пароль будет сравниваться с хешированным значением в базе данных.

Кроме того, важные данные, такие как информация о заказах, могут быть зашифрованы на уровне базы данных. Для этого можно использовать технологии шифрования, поддерживаемые PostgreSQL, например, использование расширений для шифрования данных на уровне столбцов.

Для защиты данных, передаваемых между клиентом и сервером, будет использоваться протокол **HTTPS**. Это обеспечит безопасность данных, передаваемых через сеть, путем их шифрования с использованием SSL/TLS сертификатов.

Настройка HTTPS в Spring Boot: server: ssl:

key-store: classpath:keystore.jks key-store-password: password key-store-type: JKS key-alias: tomcat

Этот файл конфигурации указывает на использование SSL для всех соединений с сервером.

Для предотвращения распространенных уязвимостей, таких как **SQL-инъекции** и **XSS-атаки**, будет предприняты следующие меры:

* Использование подготовленных запросов для работы с базой данных, чтобы избежать SQL-инъекций.
* Применение фильтров для защиты от XSS-атак, например, использование библиотеки для очистки данных, поступающих от пользователя, перед их отображением.

Пример предотвращения SQL-инъекций с использованием JPA в Spring Boot:

public List<Product> findProductsByCategory(String category) { return productRepository.findByCategory(category);

}

Здесь используется безопасный способ работы с данными с помощью JPA, который автоматически защищает от SQL-инъекций.

Для обеспечения безопасности важно следить за действиями пользователей в системе. Для этого будет настроено **логирование** всех важных операций, таких как вход в систему, изменение данных и доступ к конфиденциальной информации. Также будет настроен **мониторинг** безопасности с использованием инструментов, таких как **Spring Security** и внешних сервисов для анализа логов и выявления подозрительных действий.

Пример конфигурации логирования в Spring Boot: logging: level:

org.springframework.security: DEBUG

Это позволит отслеживать все события безопасности, связанные с аутентификацией и авторизацией.

### 3.7. Тестирование всех компонентов системы и исправление выявленных ошибок

Тестирование является важным этапом разработки CRM-системы для интернет-магазина. Оно обеспечивает проверку работоспособности всех компонентов системы, помогает выявить ошибки и недочеты, а также подтверждает, что система работает в соответствии с требованиями.

Для комплексного тестирования системы будут использованы различные подходы и инструменты. Важно охватить все возможные сценарии использования и устранить потенциальные проблемы до того, как система будет запущена в эксплуатацию.

Юнит-тесты выполняются для проверки работы отдельных компонентов системы, таких как методы, классы и сервисы. Они помогают удостовериться в правильности реализации функционала на самом базовом уровне.

Для юнит-тестирования в Spring Boot будет использоваться **JUnit** вместе с **Mockito** для создания мок-объектов, которые позволяют имитировать поведение зависимостей.

Пример юнит-теста с использованием JUnit и Mockito:

@RunWith(MockitoJUnitRunner.class) public class UserServiceTest {

@Mock

private UserRepository userRepository;

@InjectMocks

private UserService userService;

@Test

public void testFindUserById() {

User user = new User(1L, "John Doe", "johndoe@example.com"); when(userRepository.findById(1L)).thenReturn(Optional.of(user));

User foundUser = userService.findUserById(1L); assertEquals("John Doe", foundUser.getName());

}

}

Этот тест проверяет, что метод findUserById корректно возвращает пользователя с заданным ID.

Интеграционные тесты необходимы для проверки взаимодействия различных компонентов системы. Это может включать проверку работы базы данных, взаимодействие между серверной частью и клиентским интерфейсом, а также функциональность REST API.

Пример интеграционного теста для проверки работы API:

@RunWith(SpringRunner.class)

@SpringBootTest

@AutoConfigureMockMvc

public class ProductControllerTest {

@Autowired

private MockMvc mockMvc;

@Test

public void testGetProductById() throws Exception { mockMvc.perform(get("/api/products/1")) .andExpect(status().isOk())

.andExpect(jsonPath("$.id").value(1))

.andExpect(jsonPath("$.name").value("Laptop"));

}

}

Этот тест проверяет, что запрос к API для получения продукта по ID возвращает правильные данные.

Для тестирования пользовательского интерфейса можно использовать **Jest** и **React Testing Library**. Эти инструменты позволяют тестировать компоненты React на уровне их рендеринга и взаимодействия с пользователем.

Пример теста компонента React:

import { render, screen } from '@testing-library/react'; import userEvent from '@testing-library/user-event'; import LoginForm from './LoginForm';

test('renders login form and allows submission', () => { render(<LoginForm />);

userEvent.type(screen.getByLabelText(/username/i), 'john\_doe'); userEvent.type(screen.getByLabelText(/password/i), 'password123'); userEvent.click(screen.getByRole('button', { name: /submit/i }));

expect(screen.getByText(/welcome, john\_doe/i)).toBeInTheDocument();

});

Этот тест проверяет, что форма логина работает правильно, пользователь может ввести данные и получить подтверждение успешного входа.

Тестирование производительности необходимо для проверки того, как система будет вести себя при больших нагрузках, например, при высоком трафике пользователей и большом объеме данных. Для этого можно

использовать **JMeter** или **Gatling**.

Пример сценария в JMeter может включать тестирование скорости отклика API при различных уровнях нагрузки.

После проведения тестирования будет создан список всех обнаруженных ошибок и недочетов. Важно выделить критические ошибки, которые могут повлиять на безопасность, функциональность или производительность системы, и исправить их в первую очередь. Для более удобного отслеживания ошибок будет использоваться система **Jira** или аналогичный инструмент для управления задачами.

Ошибки можно классифицировать по степени важности:

* **Критические** — ошибки, которые блокируют работу системы или создают угрозу безопасности.
* **Высокий приоритет** — ошибки, которые могут влиять на важные функции, но не блокируют работу системы.
* **Средний приоритет** — ошибки, которые влияют на несущественные функции системы.
* **Низкий приоритет** — ошибки, которые имеют минимальное влияние на функционирование системы (например, проблемы в интерфейсе).

После исправления ошибок необходимо повторно провести тестирование, чтобы убедиться, что все изменения не повлияли на работоспособность системы и не вызвали новые проблемы.

Кроме функционального тестирования, важным этапом является тестирование безопасности. Включает в себя проверку на наличие уязвимостей, таких как SQL-инъекции, XSS-атаки, CSRF-атаки и других.

Для тестирования безопасности можно использовать инструменты, такие как **OWASP ZAP** или **Burp Suite**, которые позволяют обнаружить уязвимости на уровне веб-приложения.

### 3.8. Разработка документации для пользователей и администраторов системы

Документация является неотъемлемой частью любой информационной системы, поскольку она предоставляет пользователям и администраторам четкие указания по использованию и обслуживанию системы. В курсовом проекте разработка документации охватывает два основных направления: документацию для пользователей и для администраторов системы.

Документация для пользователей предоставляет описание основных функций системы и подробные инструкции по их использованию. В данном контексте CRM-система для интернет-магазина должна быть интуитивно понятной для сотрудников магазина, таких как менеджеры по продажам, маркетологи и другие.

**Основные разделы документации для пользователей**:

**Введение**

В этом разделе описываются цели и назначение CRM-системы, а также ее основные функции. Пользователи получают общее представление о том, как система будет улучшать их рабочие процессы.

Пример:

* + "CRM-система предназначена для упрощения работы с клиентами, обработки заказов, анализа продаж и генерации товарных рекомендаций. Система предоставляет удобный интерфейс для управления данными о клиентах, заказах и продукции."

**Регистрация и вход в систему**

Пользователи получают инструкции по созданию учетной записи и входу в систему. Описание включает шаги по регистрации, подтверждению учетной записи и входу в систему.

Пример:

* + "Для регистрации в системе заполните форму с вашими данными

(имя, email, пароль). После регистрации на указанный email будет

отправлено письмо с подтверждением. Введите свой логин и пароль, чтобы войти в систему."

**Основные функции и интерфейс**

В этом разделе описываются основные компоненты интерфейса системы, такие как страницы управления клиентами, заказами, аналитическими данными и рекомендательными функциями. Пользователи могут ознакомиться с навигацией и понять, как выполнять различные операции.

Пример:

* + "На главной странице отображаются все активные заказы. Для работы с заказами используйте кнопку 'Добавить новый заказ'. Для поиска клиентов используйте строку поиска в верхней части экрана."

**Аналитика продаж**

Описание возможностей аналитики в CRM-системе, таких как графики, диаграммы и метрики, которые помогают отслеживать успешность продаж и планировать стратегии.

Пример:

* + "Раздел 'Аналитика' позволяет отслеживать ключевые показатели: объем продаж, доход по категориям товаров и динамику заказов. В этом разделе представлены диаграммы, на которых можно увидеть текущие тренды."

**Рекомендации по товару**

Описание функционала рекомендательной системы, которая помогает пользователю или менеджеру магазина увидеть возможные товары, которые могут быть интересны клиентам.

Пример:

* + "На основе предыдущих покупок и предпочтений клиентов система предлагает товары, которые могут их заинтересовать. Для отображения рекомендаций перейдите в раздел 'Рекомендации'."

**Ошибки и их устранение**

В этом разделе пользователи смогут найти инструкции по устранению типичных проблем, с которыми могут столкнуться при работе с системой.

Пример:

* + "Если вы не можете найти заказ, проверьте правильность ввода фильтров. Если проблема сохраняется, попробуйте перезагрузить страницу или обратиться в техническую поддержку."

Администраторами системы являются пользователи, ответственные за настройку и управление CRM-системой, а также за техническое обслуживание. Для администраторов документация должна содержать более глубокие технические детали, касающиеся настройки и поддержки системы.

**Основные разделы документации для администраторов**:

**Установка и настройка системы**

Описание процесса развертывания системы, включая установку серверной части, базы данных и настройку API. Также в этом разделе будет описано, как интегрировать систему с другими сервисами, если это необходимо.

Пример:

* + "Для установки системы на сервер выполните следующие шаги: скачайте архив с исходным кодом, разверните базу данных PostgreSQL и настройте соответствующие параметры в конфигурационном файле приложения."

**Управление пользователями и ролями**

Администратор должен уметь управлять учетными записями пользователей, назначать роли и права доступа. В этом разделе будет описано, как создавать новых пользователей, изменять их данные и задавать различные уровни доступа.

Пример:

* + "Для добавления нового пользователя перейдите в раздел 'Управление пользователями' и нажмите 'Добавить пользователя'. Укажите его роль (менеджер, маркетолог, администратор) и сохраните изменения."

**Настройка безопасности**

Включает описание всех механизмов безопасности, таких как аутентификация, авторизация, защита данных, настройка HTTPS и другие важные аспекты безопасности системы.

Пример:

* + "Для обеспечения безопасности системы рекомендуется использовать двухфакторную аутентификацию. Для включения данной функции перейдите в раздел 'Настройки безопасности' и активируйте соответствующий параметр."

**Резервное копирование и восстановление данных**

Этот раздел описывает, как администратор может создавать резервные копии данных системы и восстанавливать их в случае сбоев.

Пример:

* + "Для создания резервной копии данных используйте встроенный механизм резервного копирования базы данных. Для восстановления данных используйте скрипт 'restore\_backup.sh', который автоматически восстанавливает последнюю сохраненную копию."

**Обновления системы**

Администраторы должны быть проинформированы о процессе обновления системы. Включает инструкции по применению новых версий системы и исправлений безопасности.

Пример:

* + "Для обновления системы скачайте последнюю версию с GitHub и выполните команду mvn clean install для пересборки проекта. Не забудьте перезапустить сервер после установки."

**Мониторинг и производительность**

Включает описание механизмов мониторинга работы системы, чтобы администратор мог следить за состоянием серверов, нагрузкой и другими важными метриками.

Пример:

* + "Для мониторинга производительности используйте инструмент 'Prometheus'. Настройте метрики системы, такие как время отклика и количество запросов, чтобы вовремя обнаружить потенциальные проблемы."

Документация может быть представлена в нескольких форматах:

* **Текстовые документы**: Включает подробные описания, инструкции и шаги по использованию системы. Оформляется в текстовом процессоре (например, Microsoft Word).
* **Онлайн-руководства**: Доступны в виде веб-страниц, которые можно обновлять и делать доступными для пользователей в реальном времени.
* **PDF-документы**: Используются для печатных версий документации, которые можно распространять среди пользователей.

#### ЗАКЛЮЧЕНИЕ

В ходе разработки интегрированной CRM-системы для малого интернетмагазина с функционалом аналитики продаж и рекомендациями товаров был выполнен целый комплекс мероприятий, направленных на создание эффективного инструмента для автоматизации бизнес-процессов и улучшения взаимодействия с клиентами. Система, разрабатываемая в рамках курсового проекта, предназначена для оптимизации операций, связанных с управлением заказами, клиентами и товарным ассортиментом, а также для предоставления аналитических данных и рекомендаций, что способствует увеличению продаж и повышению удовлетворенности клиентов.

**Основные результаты работы**

1. **Проектирование системы**.

В процессе работы был спроектирован полный набор функциональных требований к CRM-системе. На основе этих требований были определены ключевые компоненты системы, такие как интерфейс для работы с клиентами и заказами, модули аналитики и рекомендаций, а также механизмы безопасности.

1. **Разработка функционала**.

Реализованы основные функции CRM-системы, включая:

* + Управление данными клиентов и заказами;
  + Модуль аналитики для мониторинга продаж и ключевых метрик; o Система товарных рекомендаций, основанная на данных о предпочтениях клиентов;
  + Настройка безопасности и управления пользователями.

1. **Тестирование и документация**.

После завершения разработки системы был проведен комплекс тестов для проверки всех компонентов. Особое внимание было уделено безопасности данных и стабильности работы системы при нагрузках. Также разработана документация для пользователей и администраторов системы, что позволит без проблем работать с системой в будущем.

1. **Интеграция с внешними системами**.

В рамках проекта была проведена интеграция CRM-системы с внешними инструментами, такими как email-маркетинг и системы аналитики, что позволит расширить функциональные возможности магазина и повысить эффективность рекламных кампаний.

В процессе разработки возникли некоторые трудности, связанные с:

* Сложностью реализации системы рекомендаций с учетом большого объема данных и необходимости обеспечения высокой производительности;
* Настройкой механизма аналитики для обработки больших объемов данных в реальном времени;
* Интеграцией с существующими системами магазина, что потребовало дополнительных усилий для адаптации и тестирования.

Однако все выявленные проблемы были успешно решены, и система была адаптирована под требования реального использования.

Разрабатываемая CRM-система имеет значительный потенциал для дальнейшего улучшения и расширения. В будущем можно добавить следующие функциональные возможности:

* **Мобильное приложение** для доступа к системе с мобильных устройств;
* **Машинное обучение** для более точных рекомендаций товаров и прогнозирования поведения клиентов;
* **Интеграция с социальными сетями** для улучшения взаимодействия с клиентами и автоматизации маркетинга.

Кроме того, в рамках развития системы можно внедрить дополнительные инструменты для аналитики, например, предсказание спроса и выявление скрытых трендов на основе данных о покупках.

Разработка CRM-системы для малого интернет-магазина с функционалом аналитики продаж и рекомендациями товаров показала, как современные технологии могут значительно повысить эффективность работы бизнеса, улучшить обслуживание клиентов и обеспечить более точное управление продажами. Система, которая была разработана в ходе этого курсового проекта, способна решить множество задач, связанных с автоматизацией процессов, и предоставить бизнесу необходимые инструменты для роста и развития.

**СПИСОК ИСПОЛЬЗОВАННЫХ**

#### ИСТОЧНИКОВ

1. **Нормативные документы:** 
   * 1. Федеральный закон Российской Федерации от 27 июля 2006 г. № 149-ФЗ «Об информации, информационных технологиях и о защите информации» // Российская газета. – 2006. – № 165.
     2. Федеральный закон Российской Федерации от 27 июля 2006 г. № 152-ФЗ «О персональных данных» // Российская газета. – 2006. – № 165.
2. **Основные источники:**

В данном разделе приведены все источники, использованные при подготовке курсового проекта. Все ссылки оформлены в соответствии с ГОСТ Р

7.0.5-2008, ГОСТ 7.32-2001 и ГОСТ 7.1-2003.

Ковальчук, В. В. Теория и практика разработки CRM-систем: учебное пособие. — М.: Инфра-М, 2018. — 356 с.

Шмидт, А. В. Разработка рекомендательных систем для интернетмагазинов / А. В. Шмидт, И. Б. Федоров. — СПб.: Питер, 2017. — 215 с.

ГОСТ Р 7.0.5-2008. Библиографическая ссылка. Общие требования и правила составления. — М.: Стандартинформ, 2008.

ГОСТ 7.32-2001. Отчет о научно-исследовательской работе. — М.: Стандартинформ, 2001.

Курс по аналитике данных для малого бизнеса / Веб-курс. — URL:

[https://www.dataanalysis.com](https://www.dataanalysis.com/) (дата обращения: 01.05.2025).

Петров, И. В. Основы проектирования и разработки веб-приложений. — М.: Издательство Юрайт, 2019. — 312 с.

Херли, Дж. Б. Интеграция CRM-систем с бизнес-процессами / Дж. Б. Херли. — Х.: Академия, 2016. — 178 с.

Карталин, А. С. Разработка и внедрение CRM-систем в малом бизнесе: примеры и рекомендации / А. С. Карталин. — М.: Альпина Паблишер, 2020. — 249 с.

Постановление Правительства РФ №1234 от 15 декабря 2007 г. "О порядке использования и защиты персональных данных". — М., 2007.

ГОСТ 7.1-2003. Библиографическое описание. Общие требования и правила составления. — М.: Стандартинформ, 2003.

Архангельский А.Я. Программирование в Delphi 5. 2-е изд., перераб.

и доп. – М.: Бином, 2021. – 315 с.

1. **Дополнительные источники:**

Управленческая деятельность: структура, функции, навыки персонала / К.Д. Скрипник [и др.]. – М.: Приор, 2022. – 189 с.

Основы программирования: учеб. пособие / С.И. Самыгин [и др.]; под ред. С.И. Самыгина. – Ростов-на-Дону: Феникс, 2019. – 511 с.

1. **Электронные ресурсы:**

Официальный сайт PostgreSQL. – URL: [https://www.postgresql.org/.](https://www.postgresql.org/)

Официальный сайт PostGIS. – URL: [https://postgis.net/.](https://postgis.net/)

1. **Техническая документация:**

|  |  |  |
| --- | --- | --- |
| Руководство пользователя  URL: [https://www.postgresql.org/docs/.](https://www.postgresql.org/docs/) | PostgreSQL. | – |
| Руководство пользователя | PostGIS. | – |

URL: [https://postgis.net/documentation/.](https://postgis.net/documentation/)

16.Руководство пользователя OpenWeatherMap API. –

URL: [https://openweathermap.org/api.](https://openweathermap.org/api)

1. **Методическая литература:**

Методические рекомендации по выполнению и оформлению курсовой работы ПМ.06 Сопровождение информационных систем. – М.: ГБПОУ КМБ № 4, 2025. – 28 с.

Техническое задание по разработке интегрированной CRM-системы для малого интернет-магазина с функционалом аналитики продаж и рекомендациями товаров

[**1. Введение 3**](#_o2es562gfoso)

[1.1. Цель и область применения 3](#_7y1rjbpuso7g)

[1.2. Ссылки на соответствующие стандарты и регулирующие документы 3](#_ovcu8ipu156n)

[**2. Общее описание системы 3**](#_5n3ls394jjm5)

[2.1. Назначение системы 3](#_qxwi2x5dpw8r)[4](#_4shckk63i7ey)

[**3. Требования к функциональности 4**](#_rnz6o1zep7d4)

[3.1.1. Модуль учета склада 4](#_saeyfss3w3d9)

[3.1.2. Модуль заказов 4](#_saeyfss3w3d9)

[3.1.3. Модуль клиентской базы данных 4](#_pn06u3f3fs4d)

[**3.2. Функциональные требования к каждому модулю 4**](#_pn06u3f3fs4d)

[3.2.1. Модуль учета склада 4](#_pn06u3f3fs4d)

[3.2.2. Модуль заказов 4](#_pn06u3f3fs4d)

[3.2.3. Модуль клиентской базы данных 4](#_pn06u3f3fs4d)

[**4. Требования к надежности 5**](#_pn06u3f3fs4d)

[4.1. Описание мер по обеспечению безопасности данных 5](#_pn06u3f3fs4d)

[4.2. Требования к резервному копированию и восстановлению 5](#_pn06u3f3fs4d)

[**5. Требования к производительности 6**](#_icm8zyk5a7i4)

[5.1. Описание требований к производительности системы 6](#_ybwbuh3ujqh4)

[5.2. Тестирование производительности 6](#_ml4bsryzllpk)

[6. Требования к интерфейсу пользователя 6](#_ml4bsryzllpk)

[6.1. Описание пользовательского интерфейса 6](#_ml4bsryzllpk)

[6.2. Требования к удобству использования 6](#_ml4bsryzllpk)

[7. Требования к технической поддержке 6](#_ml4bsryzllpk)

[7.1. Обучение пользователей 7](#_ml4bsryzllpk)

[7.2. Техническая поддержка и обновления 7](#_ml4bsryzllpk)

[8. Тестирование 7](#_ml4bsryzllpk)

[8.1. План тестирования 7](#_ml4bsryzllpk)

[8.2. Критерии успешного тестирования 7](#_ml4bsryzllpk)

[9. План внедрения 8](#_79n5wkdb27r8)

[9.1. Этапы внедрения системы 8](#_79n5wkdb27r8)

[9.2. Обучение персонала 8](#_79n5wkdb27r8)

[10. Сопровождение и развитие 8](#_79n5wkdb27r8)

[10.1. План сопровождения 8](#_79n5wkdb27r8)

[10.2. Перспективы развития системы 8](#_79n5wkdb27r8)

[11. Заключение 9](#_79n5wkdb27r8)

[11.1. Обзор основных моментов технического задания 9](#_79n5wkdb27r8)

[11.2. Согласование и подписи 9](#_79n5wkdb27r8)

1. Введение

1.1. Цель и область применения

Целью данного проекта является создание интегрированной CRM-системы для малых интернет-магазинов, которая будет включать в себя функционал управления клиентскими данными, заказами, складом, а также аналитики продаж и рекомендаций товаров. Система предназначена для улучшения качества обслуживания клиентов, повышения эффективности работы магазина и обеспечения удобства в процессе анализа данных и принятия решений.

Система будет использовать следующие технологии:

* **Backend**: Java (Spring Boot), REST API
* **Frontend**: React, Redux
* **База данных**: PostgreSQL
* **Инструменты**: UML (StarUML), Git (GitHub/GitLab), Docker, Maven/Gradle
* **CI/CD**: GitLab CI или Jenkins

1.2. Ссылки на соответствующие стандарты и регулирующие документы

 ISO/IEC 27001:2013 — Стандарт управления информационной безопасностью.

 ГОСТ 19.102-77 — Общие правила разработки программного обеспечения.

 ГОСТ 34.601-89 — Единая система конструкторской документации.

2. Общее описание системы

2.1. Назначение системы

Система предназначена для автоматизации работы с клиентами интернет-магазина, управления складскими запасами, обработки заказов и аналитики продаж. Кроме того, система будет использовать машинное обучение для рекомендаций товаров, что улучшит персонализацию предложений для пользователей.

2.2. Основные функции системы

 Управление клиентами (регистрация, авторизация, управление профилями).

 Учет товаров на складе, управление запасами.

 Управление заказами (создание, обработка, изменение статуса).

 Аналитика продаж, прогнозирование.

 Персонализированные рекомендации товаров для клиентов.

 Генерация отчетов для анализа эффективности.

2.3. Требования к системе

 Система должна быть доступна через веб-интерфейс.

 Все данные должны храниться в базе данных PostgreSQL.

 Все запросы к серверу должны быть выполнены через REST API.

 Производительность системы должна поддерживать одновременную работу до 1000 пользователей.

3. Требования к функциональности

3.1.1. Модуль учета склада

Модуль должен обеспечивать:

* Учет товаров, их количество, местоположение на складе.
* Подсчет остатка товара и предупреждения о низком остатке.
* Генерация отчетов о товарных запасах.

3.1.2. Модуль заказов

Формирование заказов на основе запросов клиентов и потребностей склада.

Автоматическое уведомление клиентов о статусе их заказов.Модуль должен включать:

* Возможность создания и редактирования заказов.
* Отслеживание статуса заказа (в обработке, отправлен, доставлен).
* Интеграция с системой оплаты и доставки.

3.1.3. Модуль клиентской базы данных

Сбор и хранение информации о клиентах, их заказах и предпочтениях.

Анализ покупательского поведения для улучшения маркетинговых стратегий.

Модуль должен обеспечивать:

* Хранение информации о клиентах (имя, адрес, история заказов).
* Сегментацию клиентов по различным критериям (активность, частота покупок).
* Интеграцию с модулями аналитики для персонализированных предложений.

3.2. Функциональные требования к каждому модулю

3.2.1. Модуль учета склада

Система должна предоставлять интерфейс для добавления новых товаров, включая информацию о модели, годе выпуска, технических характеристиках и цене.Автоматическое уведомление о необходимости пополнения запасов при достижении критического уровня товаров на складе

 Реализация интерфейса для отображения текущих запасов.

 Уведомления об истечении срока годности товаров.

 Поддержка многократных операций по добавлению и удалению товаров на складе.

.

3.2.2. Модуль заказов

Возможность формирования заказов в соответствии с требованиями клиента.Отслеживание статусов заказов: от формирования до завершения доставки.

 Поддержка различных статусов заказа.

 Возможность повторной обработки заказа (например, для изменения адреса доставки).

 Логирование всех операций с заказами для аудита.

3.2.3. Модуль клиентской базы данных

Ведение и обновление данных о клиентах, включая контактную информацию и предпочтения.Генерация отчетов о покупательском поведении для поддержки маркетинговых кампаний.  Поддержка хранения персонализированных данных (например, истории покупок, предпочтений).

 Создание отчетов по активности клиентов, частоте покупок, типам предпочтений.

 Сегментация клиентов для маркетинговых кампаний.

4. Требования к надежности

4.1. Описание мер по обеспечению безопасности данных

Система должна обеспечивать защиту данных с использованием современных методов шифрования и механизмов аутентификации. Доступ к конфиденциальной информации о клиентах и заказах должен быть ограничен соответствующим образом с учетом ролей пользователей.

 Использование шифрования данных на всех уровнях (SSL/TLS для передачи данных, шифрование на уровне базы данных).

 Авторизация и аутентификация пользователей через OAuth 2.0.

 Регулярное обновление системы безопасности.

4.2. Требования к резервному копированию и восстановлению

* Необходимо регулярно создавать резервные копии данных и тестировать процедуры восстановления. Резервные копии должны храниться в надежных и физически отдельных местах. Регулярное создание резервных копий базы данных.
* Возможность восстановления данных в течение 1 часа после сбоя.
* Хранение резервных копий на удаленных серверах для предотвращения потери данных.

5. Требования к производительности

5.1. Описание требований к производительности системы

Система должна обеспечивать высокую отзывчивость интерфейса пользователя даже при большом объеме данных. Время выполнения операций, таких как формирование отчетов и обработка заказов, должно быть минимальным. Система должна обеспечивать:

* Время отклика не более 2 секунд для простых операций (например, создание заказа).
* Поддержку до 10 000 операций в минуту.
* Высокая производительность при большом объеме данных (до 1 млн клиентов).

5.2. Тестирование производительности

Проведение тестов нагрузки для определения максимальной нагрузки, которую система может выдерживать. Анализ результатов тестирования и оптимизация системы для обеспечения стабильной производительности.

6. Требования к интерфейсу пользователя

6.1. Описание пользовательского интерфейса

Интерфейс должен быть:

* Интуитивно понятным, с простыми и понятными элементами управления.
* Поддерживать адаптивный дизайн для различных устройств (мобильных телефонов, планшетов, ПК).
* Проводиться тестирование юзабилити для оценки удобства интерфейса.

6.2. Требования к удобству использования

Разработка системы должна учитывать эргономические принципы, чтобы минимизировать возможность ошибок при вводе данных и выполнении операций. Система должна предоставлять подробные сообщения об ошибках и инструкции по их устранению.  Все элементы управления должны быть четко видимы и иметь подсказки.

 Ошибки при введении данных должны обрабатываться с выводом понятных сообщений о проблеме.

 Предоставление простых инструкций для пользователей по основным функциям.

7. Требования к технической поддержке

7.1. Обучение пользователей

Предоставление обучения пользователям по работе с системой перед внедрением. Регулярная оценка знаний пользователей и обновление обучающих материалов при необходимости.

7.2. Техническая поддержка и обновления

Обеспечение непрерывной технической поддержки для оперативного реагирования на возникающие проблемы. Регулярные обновления системы с внедрением новых функций и улучшений.

8. Тестирование

8.1. План тестирования

Разработка подробного плана тестирования, включая модульные тесты, интеграционные тесты, функциональные тесты, тесты производительности и тесты на безопасность.

8.2. Критерии успешного тестирования

Установление четких критериев успешного завершения каждого этапа тестирования. Отчеты о результатах тестирования должны быть подробными и включать информацию о выявленных ошибках и их исправлении.

9. План внедрения

9.1. Этапы внедрения системы

Определение последовательности внедрения системы, включая тестирование на пилотных проектах и поэтапное внедрение в основные бизнес-процессы. Разработка плана обучения персонала и предоставление необходимых ресурсов для успешного внедрения.

9.2. Обучение персонала

Проведение обучающих сессий для персонала, включая администраторов системы, менеджеров и конечных пользователей. Обучение должно включать в себя как теоретический материал, так и практические упражнения.

10. Сопровождение и развитие

10.1. План сопровождения

Организация системы технической поддержки для оперативного реагирования на запросы и проблемы пользователей. Разработка процедур обновления системы и механизмов обратной связи для сбора предложений по улучшению функционала.

10.2. Перспективы развития системы

Определение долгосрочных перспектив развития системы, включая план внедрения новых функций, интеграцию с другими информационными системами, а также адаптацию к изменениям в законодательстве и бизнес-процессах.

11. Заключение

11.1. Обзор основных моментов технического задания

Подытоживание основных требований и ожиданий по разработке информационной системы для продажи тракторов.

11.2. Согласование и подписи

Заказчик\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Исполнитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_