



Geekbrains

Дипломная работа:

**Разработка CRM-системы для компании по установке натяжных
ПОТОЛКОВ**

(на базе монолитной архитектуры)

Программа: Разработчик

Специализация: Веб-разработка на Java

Соловьев Евгений Олегович

Сочи
2024

Содержание

1. Введение

- 1.1. Описание предметной области проекта
- 1.2. Цели и задачи проекта
- 1.3. Обзор используемых технологий и инструментов

2. Анализ требований

- 2.1. Функциональные требования
- 2.1.1. Нефункциональные требования

3. Архитектура проекта

- 3.1. Обзор архитектуры приложения
- 3.2. Описание слоев приложения
- 3.3. Use-Case диаграмма

4. Проектирование базы данных

- 4.1. Модель данных CRM-системы
- 4.2. ER-диаграмма базы данных
- 4.3. Использование JPA/Hibernate для доступа к данным

5. UML диаграмма классов

6. Реализация

- 6.1. Дизайн сайта
- 6.2. Общий дизайн
- 6.3. Навигационная панель
- 6.4. Каталог
- 6.5. Страница товара
- 6.6. Реализация нефункционального блока
- 6.7. Конфигурация безопасности Spring Security

7. Мониторинг приложения с помощью Actuator, Prometheus, Grafana

- 7.1. Actuator
- 7.2. Prometheus
- 7.3. Grafana

8. Заключение

- 8.1. Общий обзор выполненной работы

1. Введение

1.1. Описание предметной области проекта

Компании, занимающиеся установкой натяжных потолков, нуждаются в эффективной системе управления взаимоотношениями с клиентами (CRM). Такая система должна упрощать процесс обработки заявок, планирования задач, управления клиентской базой и анализа продаж. Автоматизация этих процессов позволяет повысить эффективность работы компании, улучшить качество обслуживания клиентов и увеличить продажи.

Эти компании часто сталкиваются с рядом проблем, которые могут быть решены с помощью CRM-системы:

- **Управление заявками:** Компании получают множество заявок от клиентов, и важно, чтобы эти заявки обрабатывались быстро и эффективно. CRM-система позволяет автоматизировать процесс приема и распределения заявок, что снижает вероятность ошибок и ускоряет время обработки.
- **Планирование задач:** Процесс установки натяжных потолков требует тщательного планирования и координации между различными командами (например, монтажниками и складскими работниками). CRM-система позволяет назначать задачи, отслеживать их выполнение и контролировать сроки.
- **Управление клиентской базой:** Ведение подробной базы данных клиентов помогает компании лучше понимать потребности своих клиентов, предлагать персонализированные услуги и улучшать общение с клиентами. CRM-система обеспечивает централизованное хранение всей информации о клиентах, включая контактные данные, историю взаимодействий и предпочтения.
- **Анализ продаж:** Для улучшения бизнес-процессов и увеличения продаж важно анализировать данные о продажах. CRM-система предоставляет инструменты для анализа данных, создания отчетов и визуализации ключевых показателей производительности.
- **Улучшение обслуживания клиентов:** Автоматизация процессов позволяет сотрудникам больше времени уделять общению с клиентами и решению их проблем. CRM-система помогает обеспечить высокое качество обслуживания за счет своевременного выполнения задач и более точного учета потребностей клиентов.
- **Снижение операционных расходов:** Автоматизация рутинных задач снижает необходимость в дополнительном персонале и уменьшает вероятность ошибок, что в конечном итоге ведет к снижению затрат.

- **Повышение конкурентоспособности:** Компании, использующие современные CRM-системы, имеют преимущество перед конкурентами, так как могут предоставлять услуги более высокого качества и более эффективно управлять своим бизнесом.

Использование CRM-системы в компании по установке натяжных потолков способствует оптимизации бизнес-процессов, улучшению взаимодействия с клиентами и повышению общей эффективности работы компании.

1.2. Цели и задачи проекта

Целью проекта является разработка CRM-системы для компании по установке натяжных потолков, которая обеспечит автоматизацию и оптимизацию основных бизнес-процессов, связанных с управлением клиентами и продажами. Основные задачи включают:

1. Управление клиентской базой:

- Создание централизованной базы данных клиентов, содержащей контактную информацию, историю взаимодействий, предпочтения и другую релевантную информацию.
- Обеспечение быстрого доступа к информации о клиентах для сотрудников компании, что позволит улучшить качество обслуживания и повысить лояльность клиентов.

2. Управление задачами и их назначение сотрудникам:

- Разработка системы управления задачами, которая позволит создавать, назначать и отслеживать задачи, связанные с установкой натяжных потолков.
- Внедрение механизма уведомлений и напоминаний, чтобы сотрудники не пропускали важные задания и соблюдали установленные сроки.

3. Отслеживание статусов заявок и задач:

- Внедрение функционала для мониторинга статусов заявок и задач в реальном времени, что позволит оперативно реагировать на изменения и предотвращать задержки.
- Разработка системы уведомлений для клиентов, информирующих их о статусе их заявок и сроках выполнения работ.

4. Анализ и отчетность по продажам и выполненным задачам:

- Создание инструментов для анализа данных о продажах, включающих визуализацию ключевых показателей и создание подробных отчетов.
- Внедрение аналитических модулей, позволяющих отслеживать эффективность работы сотрудников, выявлять узкие места и принимать обоснованные управленческие решения.

5. Повышение эффективности взаимодействия между отделами:

- Обеспечение интеграции между различными отделами компании (например, продажи, монтаж, склад), что позволит улучшить координацию и сократить время выполнения задач.
- Разработка функционала для совместного использования информации и задач между отделами, что способствует более слаженной работе и снижению вероятности ошибок.

6. Повышение прозрачности и контроля над процессами:

- Внедрение системы контроля и аудита, позволяющей отслеживать все действия сотрудников в системе, что способствует повышению прозрачности и снижению рисков.
- Разработка функционала для генерации отчетов по различным аспектам работы компании, что позволит руководству иметь полное представление о текущем состоянии дел и оперативно принимать решения.

7. Обеспечение безопасности данных:

- Внедрение современных методов защиты данных, включая шифрование, контроль доступа и регулярное резервное копирование, чтобы гарантировать безопасность и целостность клиентской информации.
- Разработка политики безопасности, регулирующей доступ сотрудников к различным разделам системы в зависимости от их роли и полномочий.

Реализация данных задач обеспечит повышение эффективности работы компании, улучшение качества обслуживания клиентов и увеличение объема продаж.

1.3. Обзор используемых технологий и инструментов

Для разработки CRM-системы для компании по установке натяжных потолков будет использован широкий спектр современных технологий и инструментов, которые обеспечат надежность, производительность и масштабируемость приложения. Основные используемые технологии и инструменты включают:

1. Языки программирования:

- **Java:** основной язык программирования для разработки серверной части приложения. Java обеспечивает высокую производительность, безопасность и широкие возможности для разработки корпоративных приложений.

2. Фреймворки и библиотеки:

- **Spring Boot:** фреймворк для создания производительных и масштабируемых приложений на языке Java. Spring Boot упрощает конфигурацию и разработку, обеспечивая встроенные средства для работы с базами данных, безопасности и управления конфигурацией.
- **Spring Security:** модуль для обеспечения безопасности в приложениях Spring, предоставляющий средства для аутентификации и авторизации пользователей.
- **Hibernate:** библиотека для объектно-реляционного отображения (ORM), которая упрощает работу с базами данных, позволяя взаимодействовать с ними через объектно-ориентированные модели.
- **Thymeleaf:** шаблонизатор для создания динамических веб-страниц, который легко интегрируется с Spring и поддерживает двустороннюю привязку данных.

3. Базы данных:

- **PostgreSQL:** реляционная база данных с поддержкой расширенных возможностей SQL, высокой производительностью и надежностью. PostgreSQL будет использоваться для хранения всех данных приложения.
- **H2 Database:** представляет собой легковесную встроенную базу данных, которая идеально подходит для разработки и тестирования приложений. Она обеспечивает высокую производительность и поддерживает стандарт SQL, что делает ее удобной для использования в различных проектах.
- **Liquibase:** инструмент для управления изменениями схемы базы данных, который позволяет отслеживать и автоматически применять изменения в базе данных.

4. Веб-технологии:

- **HTML, CSS, JavaScript:** основные технологии для разработки клиентской части веб-приложения, обеспечивающие создание интерактивных и удобных пользовательских интерфейсов.

5. Инструменты для разработки и управления проектом:

- **IntelliJ IDEA:** интегрированная среда разработки (IDE) для языка Java, обеспечивающая мощные инструменты для написания, тестирования и отладки кода.
- **Git:** система контроля версий, позволяющая отслеживать изменения в коде, работать в команде и управлять ветками разработки.
- **GitHub:** платформа для хостинга репозитория Git и совместной работы над проектом, обеспечивающая возможности для управления задачами и интеграции с другими инструментами.
- **Jenkins:** инструмент для автоматизации сборки, тестирования и развертывания приложений, который позволяет организовать непрерывную интеграцию и доставку (CI/CD).

6. Инструменты для тестирования:

- **JUnit:** фреймворк для модульного тестирования на языке Java, который позволяет писать и запускать автоматические тесты для проверки корректности работы кода.
- **Mockito:** библиотека для создания мок-объектов и проведения юнит-тестирования, которая помогает имитировать поведение зависимостей и тестировать компоненты в изоляции.

7. Инструменты для мониторинга и анализа:

- **Spring Boot Actuator:** модуль, который предоставляет готовые конечные точки для мониторинга и управления приложением, включая информацию о состоянии системы, метрики и трассировку запросов.
- **Micrometer:** библиотека для сбора и экспорта метрик из приложений Spring Boot в различные системы мониторинга, такие как Prometheus и Grafana

- **Prometheus:** система мониторинга и алертинга с временными рядами, которая позволяет собирать и анализировать метрики приложения.

Использование этих технологий и инструментов позволит создать надежную, производительную и масштабируемую CRM-систему, удовлетворяющую требованиям компании по установке натяжных потолков.

2. Анализ требований

2.1. Функциональные требования

Функциональные требования описывают, что система должна делать. Для CRM-системы это:

1. Управление клиентской базой:

- Создание, чтение, обновление и удаление (CRUD) записей клиентов.
- Хранение информации о клиентах, включая контактные данные, историю взаимодействий и предпочтения.
- Поиск и фильтрация клиентов по различным параметрам (имя, контактные данные, статус и т.д.).
- Возможность импорта и экспорта клиентских данных в различных форматах (CSV, Excel).

2. Управление заявками:

- Создание, чтение, обновление и удаление заявок.
- Назначение заявок на конкретных сотрудников.
- Отслеживание статусов заявок (новая, в работе, завершена и т.д.).
- Уведомления о статусе заявок для сотрудников и клиентов.
- Возможность добавления комментариев и вложений к заявкам.

3. Управление задачами:

- Создание, чтение, обновление и удаление задач.
- Назначение задач на сотрудников и контроль их выполнения.
- Приоритизация задач и установка дедлайнов.
- Уведомления о назначении и сроках выполнения задач.

4. Управление продажами:

- Создание, чтение, обновление и удаление записей о продажах.
- Связывание продаж с конкретными клиентами и заявками.
- Ведение истории продаж и аналитика по продажам.
- Отслеживание этапов продаж (предварительная консультация, замер, установка, завершение).

5. Анализ и отчетность:

- Генерация отчетов по различным аспектам деятельности компании (продажи, заявки, задачи, клиенты).
- Визуализация данных в виде графиков и диаграмм.
- Возможность настройки параметров отчетов и фильтрации данных.
- Экспорт отчетов в различные форматы (PDF, Excel).

6. Управление пользователями и ролями:

- Регистрация и управление учетными записями пользователей (сотрудников).
- Назначение ролей и прав доступа (администратор, менеджер, сотрудник).
- Аутентификация и авторизация пользователей.
- Управление правами доступа к различным модулям и функциям системы.

7. Интеграция с внешними системами:

- Импорт данных из внешних источников (например, из существующих систем учета клиентов).

- Экспорт данных для использования в других системах (например, бухгалтерских или аналитических системах).
- Интеграция с почтовыми сервисами для отправки уведомлений и писем клиентам.

8. Безопасность данных:

- Регулярное создание резервных копий данных.
- Защита от несанкционированного доступа и злоупотреблений.

9. Пользовательский интерфейс:

- Интуитивно понятный и удобный интерфейс для всех категорий пользователей.
- Поддержка различных устройств и экранов (адаптивный дизайн).
- Локализация интерфейса на разные языки при необходимости.

Эти функциональные требования обеспечат создание комплексной и эффективной CRM-системы для компании по установке натяжных потолков, способной удовлетворить все ключевые потребности и задачи бизнеса.

2.2. Нефункциональные требования

Нефункциональные требования определяют, как система должна выполнять свои функции, включая требования к производительности, безопасности и надежности. Для CRM-системы, нефункциональные требования включают:

1. Производительность:

- Система должна быть отзывчивой и быстро реагировать на запросы пользователей.
- Время отклика системы на действия пользователя должно быть минимальным.
- Максимальное время загрузки страниц и выполнения операций не должно превышать установленных значений.

2. Масштабируемость:

- Система должна быть способной масштабироваться в соответствии с увеличением количества клиентов, заявок и задач.
- При увеличении нагрузки на систему необходимо предусмотреть возможность горизонтального и вертикального масштабирования аппаратного и программного обеспечения.

3. Безопасность:

- Доступ к данным должен быть защищен от несанкционированного доступа.
- Передача данных между клиентом и сервером должна осуществляться по защищенным протоколам (например, HTTPS).
- Идентификация и аутентификация пользователей должны быть надежными.
- Система должна иметь механизмы резервного копирования данных и восстановления в случае сбоев.

4. Надежность:

- Система должна быть стабильной и работоспособной даже при высоких нагрузках.
- Максимальное время простоя системы должно быть минимальным.
- В случае возникновения ошибок или сбоев, система должна иметь механизмы автоматического восстановления и обработки ошибок.

5. Доступность:

- Система должна быть доступной для пользователей в любое время суток.
- Максимальное время простоя системы для планового технического обслуживания должно быть минимальным и заранее объявленным.
- Система должна иметь механизмы отказоустойчивости для предотвращения простоев из-за сбоев в работе оборудования или программного обеспечения.

6. Удобство использования:

- Интерфейс системы должен быть интуитивно понятным и удобным для всех категорий пользователей.
- Система должна иметь поддержку адаптивного дизайна для корректного отображения на различных устройствах и разрешениях экранов.

7. Совместимость:

- Система должна быть совместима с различными браузерами (Google Chrome, Mozilla Firefox, Microsoft Edge, Safari) и операционными системами (Windows, macOS, Linux).
- Система должна поддерживать работу на различных типах устройств (персональные компьютеры, ноутбуки, планшеты, смартфоны).

Эти нефункциональные требования обеспечат эффективное и надежное функционирование CRM-системы для компании по установке натяжных потолков, удовлетворяя требованиям производительности, безопасности, надежности и удобства использования.

3. Архитектура проекта

3.1. Обзор архитектуры приложения

Архитектура CRM-системы для компании по установке натяжных потолков будет построена с использованием монолитной архитектуры. Это означает, что все компоненты и функциональные модули системы будут размещены в едином исполняемом файле.

Компоненты архитектуры:

1. Пользовательский интерфейс (UI):

- Взаимодействие с пользователями происходит через веб-интерфейс.
- Веб-интерфейс будет построен с использованием современных фреймворков для создания удобного и интуитивно понятного пользовательского интерфейса.

2. Бизнес-логика:

- Содержит основные функции и алгоритмы обработки данных, необходимые для работы CRM-системы.

- Обработка запросов клиентов, управление задачами и заявками, анализ данных о клиентах и продажах, генерация отчетов и другие бизнес-процессы осуществляются в этом компоненте.

3. Доступ к данным:

- Отвечает за взаимодействие с базой данных.
- Включает в себя средства для выполнения операций CRUD (Create, Read, Update, Delete) с данными клиентов, задач, заявок, продаж и другой информации, хранимой в базе данных.

4. Преимущества монолитной архитектуры:

- Простота развертывания: Монолитное приложение разворачивается как единое целое, что упрощает процесс развертывания и управления системой.
- Удобство разработки: Все компоненты приложения находятся в одном проекте, что облегчает процесс разработки и поддержки кода.
- Меньшие затраты на инфраструктуру: Запуск монолитного приложения требует меньше ресурсов и затрат по сравнению с микросервисной архитектурой.
- Проще масштабирование: При необходимости масштабировать систему, можно масштабировать все ее компоненты одновременно.

Хотя монолитная архитектура имеет свои преимущества, она также может иметь недостатки, такие как более сложное масштабирование и поддержка при росте системы. Однако, для небольших и средних компаний, монолитная архитектура часто является оптимальным выбором из-за своей простоты и экономической эффективности.

3.2. Описание слоев приложения

Архитектура монолитного приложения включает три основных слоя, каждый из которых выполняет свою уникальную функцию:

1. Представление (UI Layer):

- Этот слой отвечает за отображение информации пользователю и обеспечивает интерфейс взаимодействия с системой.
- Включает в себя веб-интерфейс, который пользователи используют для взаимодействия с CRM-системой.

- Взаимодействие пользователей с приложением происходит через веб-страницы, формы и элементы управления, предоставляемые веб-интерфейсом.

2. Бизнес-логика (Service Layer):

- Этот слой содержит основные функции и алгоритмы, обрабатывающие бизнес-логику приложения.
- Включает в себя обработку запросов, управление задачами и заявками, анализ данных о клиентах и продажах, а также другие бизнес-процессы.
- Бизнес-логика реализуется в виде сервисов, которые обрабатывают запросы и возвращают соответствующие результаты.

3. Доступ к данным (Data Access Layer):

- Этот слой обеспечивает взаимодействие с базой данных и выполнение операций с данными.
- Включает в себя методы для создания, чтения, обновления и удаления данных из базы данных.
- Взаимодействие с базой данных происходит через объекты доступа к данным (Data Access Objects, DAO), которые предоставляют интерфейс для выполнения операций с данными.

Каждый из этих слоев играет важную роль в функционировании CRM-системы. Они взаимодействуют друг с другом для обеспечения работы приложения и достижения его основных целей. Вместе они обеспечивают эффективное управление клиентскими данными, задачами и заявками, а также обеспечивают анализ и отчетность по продажам и выполненным задачам.

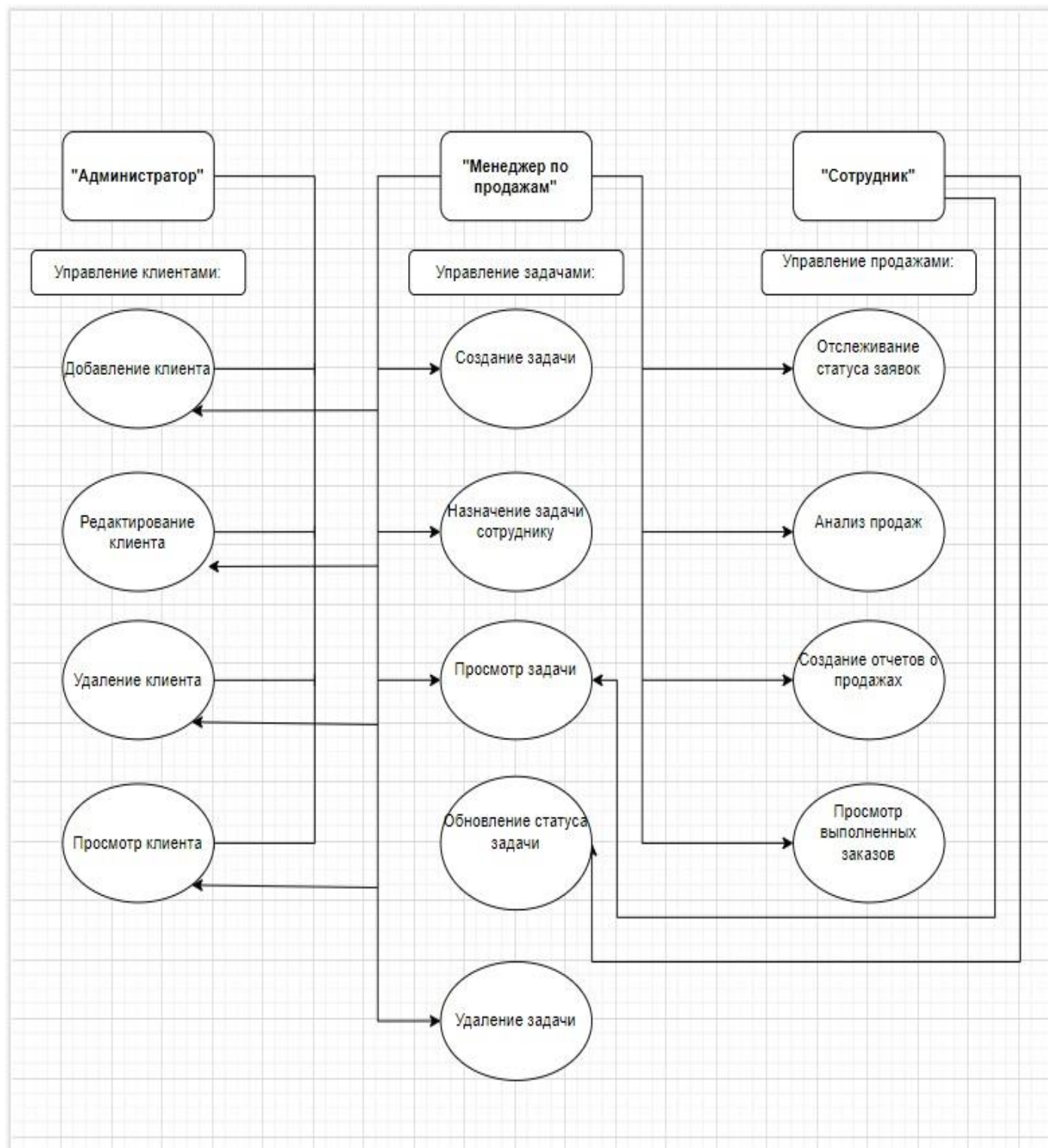
3.3. Use-Case диаграмма

Основные актеры:

1. **Администратор:** Пользователь с полным доступом к функционалу системы, включая управление пользователями, настройку системы и просмотр отчетов.
2. **Менеджер по продажам:** Пользователь, который отвечает за управление клиентской базой, обработку заявок и задач, а также анализ продаж.

3. **Сотрудник:** Пользователь, который выполняет задачи, связанные с установкой натяжных потолков, такие как проведение замеров, установка потолков и обслуживание клиентов.

Примеры сценариев использования:



Сценарии использования:

1. Управление клиентами:

Администратор и менеджер по продажам могут добавлять, редактировать и удалять клиентов из базы данных, просматривать их контактные данные и историю взаимодействия.

2. Управление задачами:

Менеджер по продажам и сотрудники могут просматривать, назначать и отслеживать задачи, связанные с установкой натяжных потолков, такие как замеры, установка и обслуживание.

3. Управление продажами:

Менеджер по продажам может отслеживать статусы заявок, анализировать данные о продажах, создавать отчеты и предоставлять информацию о выполненных заказах.

- Менеджер по продажам добавляет нового клиента в систему и создает для него задачу на замер потолка.
- Сотрудник проводит замер потолка, фиксирует результаты и отмечает задачу как выполненную.
- Администратор генерирует отчет о продажах за месяц и анализирует эффективность работы команды.

4. Проектирование базы данных

4.1. Модель данных CRM-системы

Модель данных для CRM-системы, разрабатываемой для компаний, занимающихся установкой натяжных потолков, должна учитывать основные аспекты взаимодействия с клиентами, управления задачами и отслеживания продаж.

Таблица "Клиенты":

- **ID клиента (Primary Key):** Уникальный идентификатор клиента.
- **Имя:** Имя клиента.
- **Фамилия:** Фамилия клиента.
- **Контактная информация:** Адрес электронной почты и номер телефона клиента.

- **Адрес:** Фактический адрес клиента.
- **История обращений:** Связанный список с обращениями клиента, что позволяет отслеживать всю историю взаимодействия с клиентом, включая заявки, заказы и другие обращения.

•

Таблица "Задачи":

- **ID задачи (Primary Key):** Уникальный идентификатор задачи.
- **Описание:** Описание задачи.
- **Статус:** Текущий статус задачи (например, "выполнена", "в ожидании", "в работе").
- **ID клиента (Foreign Key):** Ссылка на соответствующего клиента, к которому относится задача.
- **Сотрудник, ответственный за задачу (Foreign Key):** Ссылка на сотрудника, ответственного за выполнение задачи.
- **Дата создания:** Дата создания задачи.
- **Срок выполнения:** Планируемая дата завершения задачи.

Таблица "Продажи":

- **ID продажи (Primary Key):** Уникальный идентификатор продажи.
- **ID клиента (Foreign Key):** Ссылка на соответствующего клиента.
- **Сотрудник, ответственный за продажу (Foreign Key):** Ссылка на сотрудника, который провел продажу.
- **Дата продажи:** Дата совершения продажи.
- **Сумма:** Сумма продажи.
- **Товары:** Список товаров, включенных в продажу, что позволяет отслеживать, какие товары были приобретены клиентом.

4.2. ER-диаграмма базы данных

ER-диаграмма представляет собой визуализацию структуры базы данных, отражая сущности (таблицы), их атрибуты и связи между ними. В случае CRM-системы для компаний по установке натяжных потолков, ER-диаграмма поможет понять, какую информацию мы храним и как она связана.

Таблицы базы данных:

1. Client (Клиент)

- *client_id*: int (Primary Key) - Уникальный идентификатор клиента.
- *first_name*: varchar - Имя клиента.
- *last_name*: varchar - Фамилия клиента.
- *email*: varchar - Адрес электронной почты клиента.
- *phone*: varchar - Номер телефона клиента.
- *address*: varchar - Фактический адрес клиента.

Task (Задача)

- *task_id: int* (Primary Key) - Уникальный идентификатор задачи.
- *description: varchar* - Описание задачи.
- *status: varchar* - Статус задачи.
- *client_id: int* (Foreign Key) - Идентификатор клиента, связанный с задачей.
- *employee_id: int* (Foreign Key) - Идентификатор сотрудника, ответственного за задачу.
- *created_date: datetime* - Дата создания задачи.
- *due_date: datetime* - Срок выполнения задачи.

Sale (Продажа)

- *sale_id: int* (Primary Key) - Уникальный идентификатор продажи.
- *client_id: int* (Foreign Key) - Идентификатор клиента, совершившего покупку.
- *employee_id: int* (Foreign Key) - Идентификатор сотрудника, проводшего продажу.
- *sale_date: datetime* - Дата продажи.
- *amount: decimal* - Сумма продажи.

Employee (Сотрудник)

- *employee_id: int* (Primary Key) - Уникальный идентификатор сотрудника.
- *first_name: varchar* - Имя сотрудника.
- *last_name: varchar* - Фамилия сотрудника.
- *position: varchar* - Должность сотрудника.

Связи между таблицами:

- Каждая задача (Task) привязана к определенному клиенту (Client) и определенному сотруднику (Employee).
- Каждая продажа (Sale) также привязана к определенному клиенту (Client) и определенному сотруднику (Employee).

Эта структура базы данных обеспечивает эффективное хранение и управление информацией о клиентах, задачах и продажах в CRM-системе для компаний по установке натяжных потолков.

4.3. Использование JPA/Hibernate для доступа к данным

4.4.

JPA (Java Persistence API) и Hibernate - это мощные инструменты для работы с базами данных в Java-приложениях. Они позволяют разработчикам взаимодействовать с базой

данных, используя объектно-ориентированный подход, что делает работу с данными более удобной и эффективной.

Преимущества использования JPA/Hibernate:

1. **ORM (Object-Relational Mapping):** JPA и Hibernate позволяют отображать объекты Java на записи в базе данных и наоборот, избавляя разработчиков от необходимости писать SQL-запросы вручную и обеспечивая более простой и интуитивный способ работы с данными.
2. **Уменьшение дублирования кода:** Благодаря использованию аннотаций и конфигураций, разработчики могут сократить объем кода, необходимого для доступа к данным, и избежать повторного написания стандартных CRUD (Create, Read, Update, Delete) операций.
3. **Управление транзакциями:** JPA и Hibernate автоматически управляют транзакциями базы данных, обеспечивая целостность данных и предотвращая возможные ошибки при работе с базой данных.

Встроенные методы JPA/Hibernate:

JPA и Hibernate предоставляют ряд встроенных методов для работы с базой данных:

- **save(entity):** сохраняет сущность в базе данных. Если сущность уже существует в базе данных, то она обновляется, в противном случае создается новая запись.
- **findById(id):** ищет сущность в базе данных по ее идентификатору.
- **findAll():** извлекает все сущности из базы данных.
- **delete(entity):** удаляет сущность из базы данных.
- **deleteById(id):** удаляет сущность из базы данных по ее идентификатору.
- **count():** возвращает количество сущностей в базе данных.
- **existsById(id):** проверяет, существует ли сущность с указанным идентификатором в базе данных.

5. UML диаграмма классов

Диаграмма классов (UML Class Diagram) - это один из ключевых инструментов для визуализации структуры классов в программном обеспечении. Она помогает определить основные сущности системы, их атрибуты и методы, а также взаимоотношения между этими сущностями.

Для нашего CRM-приложения, предназначенного для управления компанией, основные классы включают:

- **Client:** класс для представления клиентов.
- **Task:** класс для представления задач.
- **Employee:** класс для представления сотрудников.
- **Sale:** класс для представления продаж.

Описание классов:

1. Client:

- Атрибуты:
 - **id:** Long
 - **firstName:** String
 - **lastName:** String
 - **email:** String
 - **phoneNumber:** String
 - **address:** String
- Методы:
 - Геттеры и сеттеры для каждого атрибута

2. Task:

- Атрибуты:
 - **id:** Long
 - **title:** String
 - **description:** String
 - **status:** String (e.g., "Pending", "In Progress", "Completed")

- **assignedTo:** Employee
- **client:** Client
- Методы:
 - Геттеры и сеттеры для каждого атрибута

3. Employee:

- Атрибуты:
 - **id:** Long
 - **firstName:** String
 - **lastName:** String
 - **position:** String
 - **email:** String
 - **phoneNumber:** String
- Методы:
 - Геттеры и сеттеры для каждого атрибута

4. Sale:

- Атрибуты:
 - **id:** Long
 - **amount:** Double
 - **date:** LocalDate
 - **client:** Client
- Методы:
 - Геттеры и сеттеры для каждого атрибута

Связи между классами:

Client имеет многие ко многим (Many-to-Many) отношения с **Task**, так как один клиент может иметь несколько задач, и одна задача может быть связана с несколькими клиентами (в случае коллективных заказов).

Task имеет многие к одному (Many-to-One) отношение с **Employee**, так как одна задача может быть назначена только одному сотруднику.

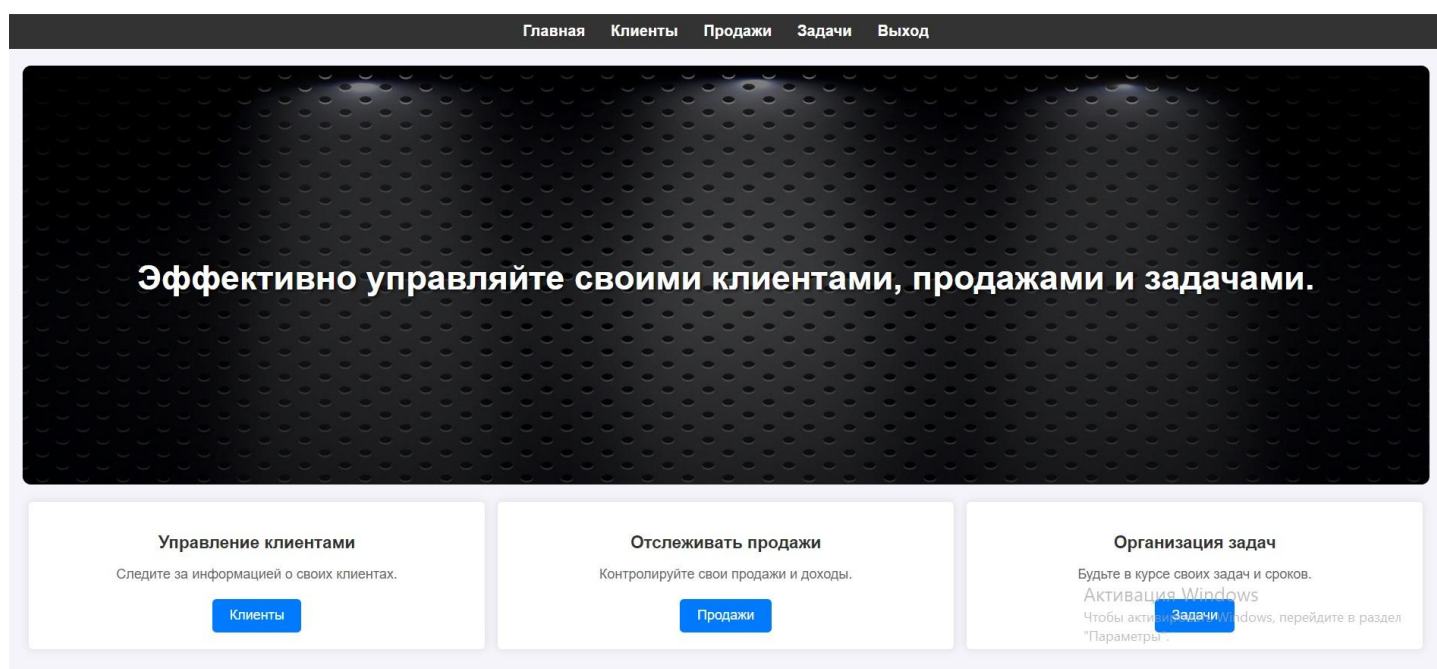
Sale имеет многие к одному (Many-to-One) отношение с Client, так как одна продажа связана с одним клиентом.

6. Реализация

Разработка CRM-системы для компании, занимающейся установкой натяжных потолков, включала несколько этапов реализации. В этом разделе мы рассмотрим дизайн сайта и ключевые аспекты реализации.

6.1. Дизайн сайта

Дизайн сайта играет ключевую роль в обеспечении удобства использования и эффективности работы с системой. В нашем проекте дизайн был разработан с учетом современных стандартов и требований, что обеспечило интуитивно понятный интерфейс и удобство работы для пользователей.



6.1.1. Общий дизайн

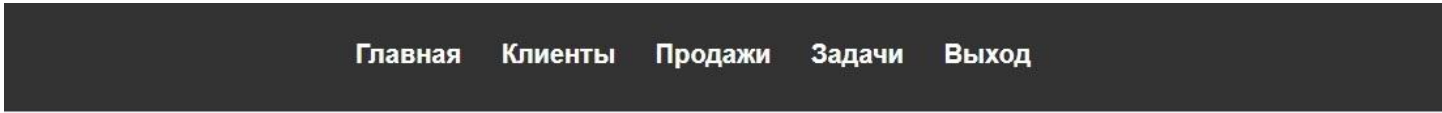
Общий дизайн сайта основывался на принципах минимализма и функциональности. Основные элементы дизайна включали:

- **Чистый и понятный интерфейс:** Использование простых и четких линий, нейтральных цветов, чтобы сосредоточить внимание пользователя на содержимом.

- **Адаптивный дизайн:** Обеспечение корректного отображения на различных устройствах (ПК, планшеты, смартфоны) с помощью CSS-медиа-запросов.
- **Единый стиль:** Соблюдение единого стиля на всех страницах для поддержания визуальной консистентности.

6.1.2. Навигационная панель

Навигационная панель обеспечивает доступ ко всем основным разделам системы и включает:



Главная страница: Отображает ключевые показатели и быстрые ссылки на основные разделы.

Клиенты: Раздел для управления клиентской базой.

Задачи: Раздел для создания, назначения и отслеживания задач.

Продажи: Раздел для управления и анализа продаж.

Отчеты (будет) Раздел для генерации и просмотра отчетов по различным параметрам.

Настройки: (будет) Раздел для управления настройками системы и профилями пользователей.

6.1.3. Продажи

ГлавнаяКлиентыПродажиЗадачиВыход

Продажи

Добавить Продажу

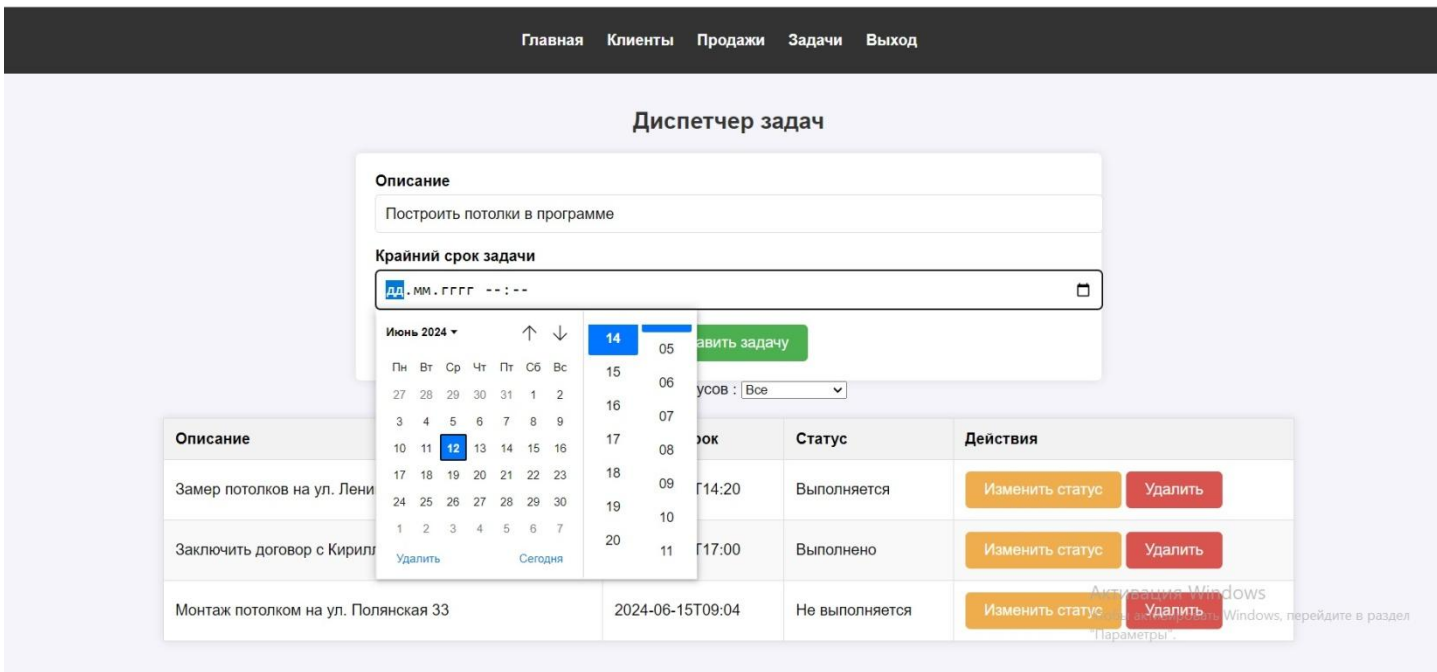
ID	Наименование:	Цена:	Дата продажи	Действия
3	Натяжной потолок в Квартире студии 44.5 м2	32000.0	2024-06-03	<div>РедактироватьУдалитьПодробнее</div>
4	Натяжной потолок в Квартире студии 34 м2	2000.0	2024-06-12	<div>РедактироватьУдалитьПодробнее</div>
6	Натяжной потолок в Квартире студии 22 м2	16600.0	2024-04-10	<div>РедактироватьУдалитьПодробнее</div>
7	Матовый натяжной потолок 32 м2.	54000.0	2024-01-11	<div>РедактироватьУдалитьПодробнее</div>
8	Натяжной потолок на Кухне 12 м2	24300.0	2024-02-10	<div>РедактироватьУдалитьПодробнее</div>
9	Натяжной потолок в Квартире студии 34 м2	11220.0	2024-06-21	<div>РедактироватьУдалитьПодробнее</div>
10	Матовый натяжной потолок 32 м2.	41000.0	2024-06-07	<div>РедактироватьУдалитьПодробнее</div>
11	Глянцевый натяжной потолок 32 м2.	12000.0	2024-06-28	<div>РедактироватьУдалитьПодробнее</div>

Каталог предназначен для отображения списка клиентов, задач и продаж. Основные особенности:

- **Поиск и фильтрация:** Возможность поиска по различным параметрам (имя клиента, статус задачи, дата продажи) и использования фильтров для уточнения результатов.
- **Сортировка:** Возможность сортировки данных по различным критериям (по алфавиту, дате, статусу).
- **Пагинация:** Разбиение данных на страницы для удобства просмотра больших объемов информации.

6.1.4. Страница задачи

Страница задачи предоставляет пользователям доступ к информации о конкретной задаче, связанной с установкой натяжных потолков. Включает следующие компоненты:



- **Описание задачи:** Подробное описание работы или задания, которое должно быть выполнено.
- **Статус задачи:** Текущий статус задачи (например, "выполнена", "в ожидании", "в работе").

- Дата создания: Дата, когда задача была добавлена в систему.
- Срок выполнения: Планируемая дата завершения задачи.
- Клиент: Связанный клиент, для которого создана задача.
- Ответственный сотрудник: Сотрудник, которому назначена данная задача.
- Комментарии: Возможность добавления и просмотра комментариев, связанных с данной задачей.
- История изменений: Лог изменений, отражающий все действия, совершенные над данной задачей (например, изменение статуса, назначение ответственного и т. д.).

На странице задачи пользователи могут отслеживать текущий статус задачи, добавлять комментарии, обмениваться информацией и принимать необходимые меры для выполнения задания в срок.

6.1.5. Страница клиента

Главная
Клиенты
Продажи
Задачи
Выход

Клиенты

Добавить нового клиента

ID	Имя	Фамилия	Электронная почта	Действия
3	Евгений	Соловьев	22potolok@gmail.com	<div>Редактировать</div> <div>Удалить</div>
4	Оксана	Пушкина	sdgh@list.ru	<div>Редактировать</div> <div>Удалить</div>
5	Алексей	Кудыкин	dfgjkui@gmail.com	<div>Редактировать</div> <div>Удалить</div>
6	Татьяна	Митюхина	zzx777@gmail.com	<div>Редактировать</div> <div>Удалить</div>
7	Светлана	Осипова	56dfghy@gmail.com	<div>Редактировать</div> <div>Удалить</div>
8	Сергей	Жарков	2ff44k@gmail.com	<div>Редактировать</div> <div>Удалить</div>
9	Антон	Перогов	tolok@gmail.com	<div>Редактировать</div> <div>Удалить</div>
10	Егор	Тарасов	88ok@gmail.com	<div>Редактировать</div> <div>Удалить</div>

Активация Windows

Чтобы активировать Windows, перейдите в раздел "Параметры".

Страница клиента включает:

- **Основную информацию:** Имя, контактные данные, адрес.
- **Историю взаимодействий:** Задачи, связанные с клиентом, и их статусы.
- **Продажи:** История покупок клиента.
- **Комментарии:** Возможность добавления и просмотра комментариев, связанных с клиентом.

1. Главная страница: Содержит виджеты с ключевыми показателями, такими как количество клиентов, активных задач и текущих продаж. Примеры метрик:

- Количество новых клиентов за последний месяц.
- Статус выполнения текущих задач.
- Общая сумма продаж за выбранный период.

Страница клиентов: Таблица с данными о клиентах, функциональность для добавления, редактирования и удаления записей, возможность просмотра детальной информации о каждом клиенте.

2. Страница задач: Список задач с возможностью фильтрации по статусу, назначению сотрудникам и срокам выполнения. Детальная информация о каждой задаче включает описание, ответственного сотрудника и сроки выполнения.

3. Страница продаж: Отображение истории продаж с деталями каждой сделки, включая дату, сумму и клиента.

Заключение

Разработанный дизайн сайта CRM-системы обеспечивает удобный и интуитивно понятный интерфейс, который способствует эффективной работе пользователей. Благодаря адаптивному дизайну система доступна для использования на различных устройствах, что повышает гибкость и доступность для сотрудников компании.

6.2. Реализация нефункционального блока

Нефункциональный блок проекта включает в себя ряд компонентов и настроек, которые обеспечивают общую работоспособность и надежность приложения, но не прямо связаны с его функциональностью. В этом блоке определяются основные параметры, а также настраивается инфраструктура для поддержки функциональных модулей приложения.

6.2.2. Конфигурация безопасности Spring Security

Spring Security используется для аутентификации и авторизации пользователей, защиты данных и управления доступом к различным частям системы. Конфигурация безопасности включает следующие аспекты:

1. Настройка безопасности на уровне HTTP

Spring Security позволяет контролировать доступ к URL-адресам приложения на основе ролей пользователей. В нашем приложении используется метод `http.authorizeRequests()` для определения прав доступа к различным URL. Например, доступ к административной панели должен быть разрешен только администраторам, в то время как доступ к страницам просмотра задач и клиентов может быть предоставлен менеджерам по продажам и другим авторизованным сотрудникам.

```
Explain | Test | Document | Fix | Ask
no usages  Solov18 *
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
    http
        .csrf(csrf -> csrf.disable())
        .authorizeHttpRequests(authorizeRequests ->
            authorizeRequests
                .requestMatchers(...patterns: "/admin/**").hasRole("ADMIN")
                .requestMatchers(...patterns: "/admin/**").hasRole("ADMIN")
                .requestMatchers(...patterns: "/tasks/**").hasAnyRole(...roles: "ADMIN", "USER")
                .requestMatchers(...patterns: "/clients/**").hasAnyRole(...roles: "ADMIN", "USER")
                .requestMatchers(...patterns: "/login").permitAll()
                .anyRequest().authenticated()
            )
        .formLogin(formLogin ->
            formLogin
                .defaultSuccessUrl( defaultSuccessUrl: "/representative/listall", alwaysUse: true)
            )
        .httpBasic(Customizer.withDefaults())
        .logout(logout ->
            logout
                .permitAll()
        );

    return http.build();
}
```

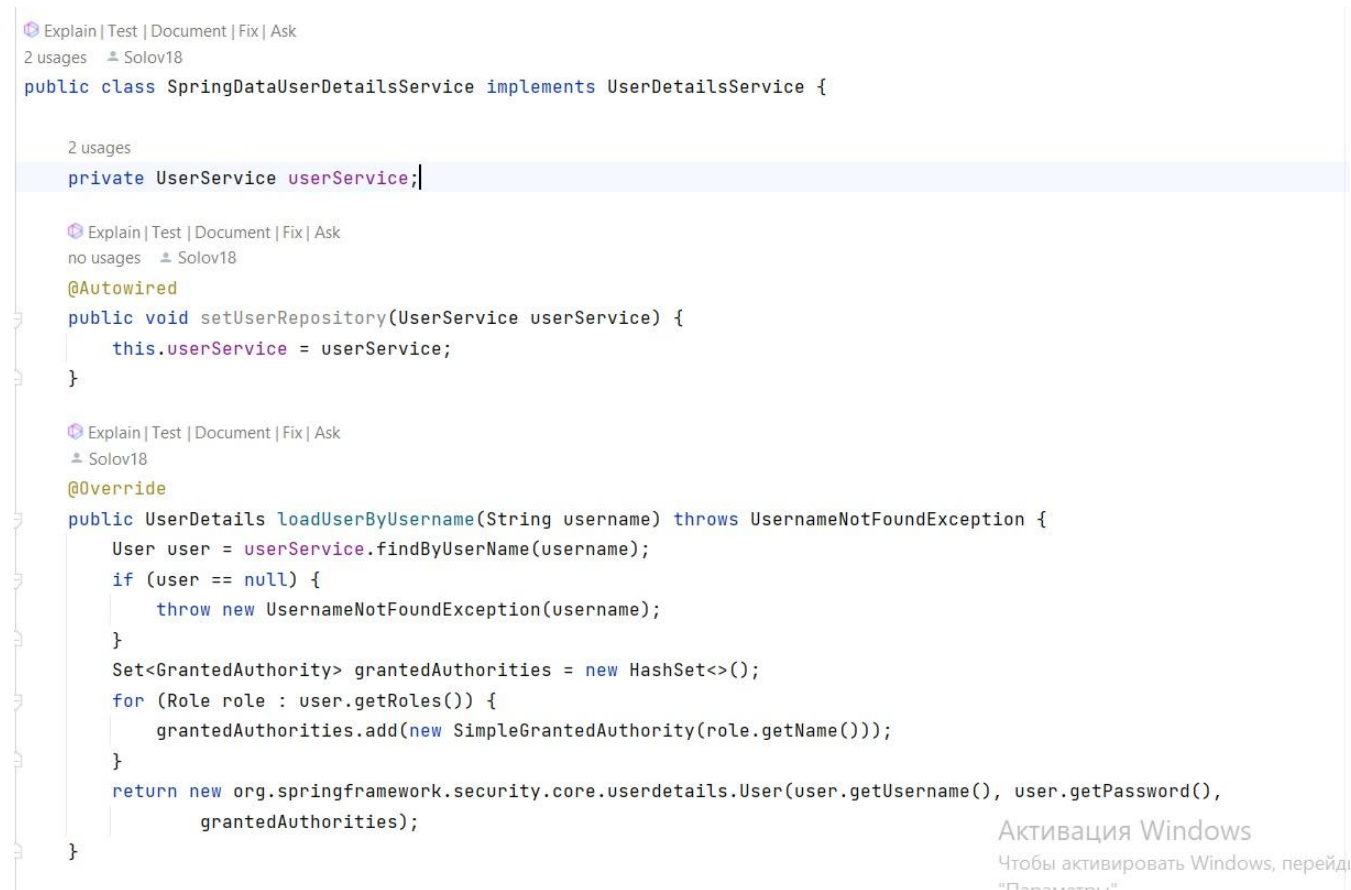
2. Конфигурация шифрования паролей

Для обеспечения безопасности паролей используется шифрование с помощью **BCryptPasswordEncoder**. Это позволяет безопасно хранить пароли в базе данных и защищает их от несанкционированного доступа.

```
Explain | Test | Document | Fix | Ask
no usages  Solov18
@Bean
public BCryptPasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
}
```

3. Настройка аутентификации пользователей

Для аутентификации пользователей в нашем приложении мы используем `UserDetailsService`, который загружает данные пользователей из базы данных. Пользователи и их роли хранятся в таблицах базы данных, что позволяет динамически управлять доступом.



```
public class SpringDataUserDetailsService implements UserDetailsService {

    2 usages
    private UserService userService;

    Explain | Test | Document | Fix | Ask
    no usages Solov18
    @Autowired
    public void setUserRepository(UserService userService) {
        this.userService = userService;
    }

    Explain | Test | Document | Fix | Ask
    Solov18
    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        User user = userService.findByUserName(username);
        if (user == null) {
            throw new UsernameNotFoundException(username);
        }
        Set<GrantedAuthority> grantedAuthorities = new HashSet<>();
        for (Role role : user.getRoles()) {
            grantedAuthorities.add(new SimpleGrantedAuthority(role.getName()));
        }
        return new org.springframework.security.core.userdetails.User(user.getUsername(), user.getPassword(),
            grantedAuthorities);
    }
}
```

Заключение

Конфигурация безопасности Spring Security обеспечивает высокий уровень защиты приложения, контролируя доступ пользователей к различным частям системы, управляя аутентификацией и шифрованием паролей, а также защищая сессии пользователей. Эта комплексная защита помогает предотвратить несанкционированный доступ и гарантирует безопасность данных в CRM-системе.

7. Мониторинг приложения с помощью Actuator, Micrometer, Prometheus, Grafana

Мониторинг приложения является критически важным аспектом для обеспечения его надежности, производительности и безопасности. В данном проекте для реализации мониторинга были выбраны инструменты Spring Boot Actuator, Micrometer,

Prometheus и Grafana. Данная глава описывает использование этих инструментов и их конфигурацию для мониторинга CRM-системы.

7.1. Spring Boot Actuator

Spring Boot Actuator представляет собой набор инструментов, предоставляющих доступ к метрикам, данным о здоровье приложения и другой диагностической информации. Включение Actuator в проект позволяет разработчикам и администраторам отслеживать состояние и производительность приложения в реальном времени.

Основные конечные точки Spring Boot Actuator

- **/actuator/health:** Проверка состояния приложения.
- **/actuator/metrics:** Доступ к различным метрикам приложения.
- **/actuator/env:** Информация о переменных окружения.
- **/actuator/loggers:** Управление уровнями логирования.

Пример конфигурации Actuator:

```
no usages new *
@Configuration
public class ActuatorConfig {

    @Bean
    public ServletRegistrationBean<DispatcherServlet> actuatorServlet() {
        ServletRegistrationBean<DispatcherServlet> servlet = new ServletRegistrationBean<>(new DispatcherServlet(),
            ...urlMappings: "/actuator/*");
        servlet.setLoadOnStartup(1);
        servlet.setName("ActuatorServlet");
        return servlet;
    }
}
```

Настройки в `application.properties`:

```
management.endpoints.web.exposure.include=*  
management.endpoint.health.show-details=always
```

7.2. Micrometer

Micrometer является фасадом для различных систем мониторинга и позволяет интегрировать их с Spring Boot. Он предоставляет единый интерфейс для работы с метриками, что упрощает процесс их сбора и обработки.

Основные методы Micrometer:

- **Counter:** Счетчик событий.
- **Gauge:** Метрика, отображающая текущее состояние.
- **Timer:** Измерение времени выполнения операций.

Пример использования Micrometer:

Explain | Test | Document | Fix | Ask

no usages Solov18

@RestController

```
public class MetricsController {
```

2 usages

```
private final Counter requestCounter;
```

Explain | Test | Document | Fix | Ask

no usages Solov18

@Autowired

```
public MetricsController(MeterRegistry meterRegistry) {  
    this.requestCounter = meterRegistry.counter(name: "http.requests");  
}
```

Explain | Test | Document | Fix | Ask

no usages Solov18

@GetMapping("/metrics")

```
public String getMetrics() {  
    requestCounter.increment();  
    return "Metrics collected";  
}
```

7.3. Prometheus

Prometheus используется для сбора, хранения и анализа метрик. Он опрашивает конечные точки, предоставляемые Micrometer, и сохраняет метрики в своей базе данных. Prometheus позволяет гибко настраивать частоту сбора данных и обеспечивает мощные средства для анализа и агрегации метрик.

Конфигурация Prometheus:

```
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'crm-application'
    static_configs:
      - targets: ['localhost:8080']
```

7.4. Grafana

Grafana используется для визуализации метрик, собранных Prometheus. Он предоставляет мощный и гибкий интерфейс для создания дашбордов и отчетов, что позволяет быстро получать наглядную информацию о состоянии и производительности приложения.

Настройка Grafana:

1. Установить Grafana и запустите его.
2. Откройте веб-интерфейс Grafana (обычно по адресу <http://localhost:3000>).
3. Добавить источник данных Prometheus, указав URL Prometheus-сервера (например, <http://localhost:9090>).
4. Создайте новый дашборд и добавьте панели для отображения метрик.

Пример дашборда в Grafana:

Панель 1: Количество запросов

Запрос: `http_requests_total`

Панель 2: Время ответа

Запрос: `http_server_requests_seconds_count`

Заключение

Интеграция Spring Boot Actuator, Micrometer, Prometheus и Grafana позволяет обеспечить полный цикл мониторинга CRM-системы. Эти инструменты предоставляют детализированные данные о состоянии и производительности приложения, что помогает

своевременно выявлять и устранять проблемы. Такой подход к мониторингу значительно повышает надежность и устойчивость системы, обеспечивая высокое качество обслуживания клиентов и стабильную работу компании.

8. Заключение

8.1. Общий обзор выполненной работы

Разработка CRM-системы для компании по установке натяжных потолков представляла собой многогранный и комплексный проект, требующий всестороннего подхода к проектированию и реализации. В ходе работы были последовательно пройдены несколько ключевых этапов, каждый из которых внёс существенный вклад в конечный продукт. В этом разделе мы рассмотрим общий обзор выполненной работы, описывающий основные этапы и подходы, использованные при создании приложения.

Этап 1: Анализ требований

Проект начался с детального анализа требований. На данном этапе были определены функциональные и нефункциональные требования к системе. Функциональные требования включали управление клиентской базой, управление задачами и их назначение сотрудникам, отслеживание статусов заявок и задач, а также анализ и отчетность по продажам и выполненным задачам. Нефункциональные требования касались производительности, безопасности, надежности и удобства использования системы.

Этап 2: Разработка архитектуры

На следующем этапе была разработана архитектура проекта, основанная на монолитной модели. Эта модель была выбрана для упрощения разработки и развертывания на начальном этапе. Архитектура включала три основных слоя:

- **Представление (UI Layer):** Отвечает за отображение данных пользователю и взаимодействие с ним.
- **Бизнес-логика (Service Layer):** Обработывает бизнес-логику и правила приложения.

- **Доступ к данным (Data Access Layer):** Обеспечивает взаимодействие с базой данных.

Такая структура позволила четко разделить ответственность между различными частями системы, что способствовало упрощению разработки и поддержки приложения.

Этап 3: Проектирование базы данных

На этапе проектирования базы данных была создана модель данных CRM-системы. Она включала таблицы для хранения информации о клиентах, задачах, продажах и других аспектах, необходимых для функционирования системы. Основные сущности и их связи были отображены с помощью ER-диаграммы, что позволило визуализировать структуру базы данных и упростить её дальнейшую реализацию.

Использование JPA/Hibernate для доступа к данным обеспечило удобное и эффективное взаимодействие с базой данных. Эти инструменты позволили маппировать объекты Java на записи в базе данных, что значительно упростило процесс разработки и тестирования.

Этап 4: Реализация функционала

Реализация функционала включала разработку основных модулей системы, таких как управление клиентами, управление задачами, управление продажами и аналитика. Для этого использовались различные технологии и инструменты, включая Spring Boot, Spring Data JPA, Hibernate и другие.

Одним из важных аспектов было обеспечение безопасности системы. Для этого использовалась Spring Security, которая обеспечивала аутентификацию и авторизацию пользователей, защиту данных и управление доступом к различным частям системы. Конфигурация безопасности включала настройку прав доступа для различных ролей пользователей, таких как администратор, менеджер по продажам и сотрудники.

Этап 5: Дизайн и пользовательский интерфейс

Дизайн и пользовательский интерфейс системы играли ключевую роль в обеспечении удобства использования приложения. Включение отзывчивого дизайна и интуитивно

понятной навигации способствовало улучшению взаимодействия пользователей с системой. Были разработаны интерфейсы для управления клиентами, задачами и продажами, а также для просмотра аналитических данных и отчетов.

Этап 6: Мониторинг и поддержка

Мониторинг приложения является критически важным для обеспечения его надежности и производительности. В этом проекте были использованы инструменты Spring Boot Actuator, Micrometer, Prometheus и Grafana для реализации мониторинга. Actuator предоставлял информацию о состоянии приложения, Micrometer собирал метрики, Prometheus обеспечивал хранение метрик, а Grafana использовалась для визуализации данных. Такая интеграция позволила обеспечить полный цикл мониторинга, что способствовало своевременному выявлению и устранению проблем, повышая надежность и устойчивость системы.

Этап 7: Тестирование и отладка

На протяжении всего процесса разработки проводилось тестирование и отладка системы. Были использованы различные виды тестирования, включая модульное тестирование, интеграционное тестирование и тестирование производительности. Это позволило выявить и исправить ошибки, а также убедиться в корректной работе всех компонентов системы.

Заключение

Выполненная работа по разработке CRM-системы для компании по установке натяжных потолков охватывала весь жизненный цикл создания программного обеспечения, начиная с анализа требований и проектирования, и заканчивая реализацией, тестированием и мониторингом. Благодаря структурированному подходу и использованию современных технологий удалось создать надежную и эффективную систему, которая значительно улучшает управление клиентскими отношениями и бизнес-процессами компании.

Основные достижения проекта включают:

- Разработку гибкой и масштабируемой архитектуры системы.
- Создание удобного и интуитивно понятного пользовательского интерфейса.
- Обеспечение высокой безопасности и производительности приложения.
- Внедрение эффективных инструментов мониторинга и анализа данных.

Эти достижения создают прочную основу для дальнейшего развития и расширения функциональности CRM-системы, что позволит компании улучшить качество обслуживания клиентов и повысить свою конкурентоспособность на рынке.

8.2. Полученные результаты и достижения

Разработка CRM-системы для компании по установке натяжных потолков была сложной и многогранной задачей, которая включала в себя множество этапов и аспектов. В результате выполнения проекта были достигнуты значительные успехи, которые можно разделить на несколько ключевых областей:

1. Разработка и успешное развертывание CRM-системы

Одним из главных достижений проекта стало создание полнофункциональной CRM-системы, которая была успешно разработана, протестирована и развернута. В ходе работы была выбрана и реализована монолитная архитектура, обеспечивающая простоту развертывания и управления на начальном этапе. Это решение позволило оперативно внести необходимые изменения и обеспечить минимальные накладные расходы на управление инфраструктурой.

2. Реализация основных функций управления клиентами, задачами и продажами

Основные функциональные блоки системы были реализованы и включают в себя следующие компоненты:

- **Управление клиентами:** Система позволяет создавать и хранить информацию о клиентах, включая контактные данные, историю взаимодействий, предпочтения и потребности. Это позволяет сотрудникам компании эффективно управлять взаимоотношениями с клиентами и предоставлять персонализированные услуги.

- **Управление задачами:** Внедрен функционал для создания, назначения и отслеживания задач. Менеджеры могут назначать задачи сотрудникам, устанавливать сроки выполнения и контролировать статус выполнения задач. Это способствует улучшению планирования и координации работы команды.
- **Управление продажами:** Реализован модуль для управления процессом продаж, включающий создание и обработку заявок, отслеживание статусов заказов и генерацию отчетов по продажам. Это позволяет компании анализировать эффективность продаж, выявлять ключевые тенденции и принимать обоснованные решения для повышения продаж.

3. Обеспечение безопасности и надежности системы

Безопасность и надежность системы были приоритетными задачами на всех этапах разработки. Для обеспечения этих аспектов были реализованы следующие меры:

- **Spring Security:** Система безопасности включала в себя аутентификацию и авторизацию пользователей, защиту данных и управление доступом к различным частям системы. Были настроены права доступа для различных ролей пользователей, таких как администраторы, менеджеры по продажам и сотрудники, что обеспечило контроль и защиту данных.
- **Мониторинг и управление производительностью:** Интеграция инструментов мониторинга, таких как Spring Boot Actuator, Micrometer, Prometheus и Grafana, позволила обеспечить полный цикл мониторинга состояния приложения. Это включало сбор и анализ метрик производительности, визуализацию данных и своевременное выявление проблем. В результате система стала более надежной и устойчивой к нагрузкам.

4. Пользовательский интерфейс и удобство использования

Дизайн и реализация пользовательского интерфейса были выполнены с учетом современных стандартов и требований. Включение отзывчивого дизайна и интуитивно понятной навигации способствовало улучшению взаимодействия пользователей с

системой. Основные интерфейсы для управления клиентами, задачами и продажами были разработаны с акцентом на удобство использования и функциональность.

5. Тестирование и качество

На протяжении всего проекта проводилось тщательное тестирование, включая модульное, интеграционное и нагрузочное тестирование. Это позволило выявить и исправить ошибки на ранних этапах разработки, обеспечивая высокое качество конечного продукта. Все ключевые компоненты системы были протестированы на соответствие функциональным и нефункциональным требованиям, что гарантировало стабильную и корректную работу приложения.

6. Анализ и отчетность

Реализованные функции анализа и отчетности позволили компании получать детализированные отчеты по различным аспектам своей деятельности, включая продажи, выполнение задач и управление клиентами. Это способствовало улучшению принятия управленческих решений, планированию и стратегическому развитию компании.

Выводы

В результате выполнения данного проекта была создана высококачественная CRM-система, полностью соответствующая требованиям заказчика и способная значительно улучшить процессы управления взаимоотношениями с клиентами, задачами и продажами. Реализация системы позволила компании повысить свою эффективность, улучшить качество обслуживания клиентов и увеличить продажи. Основные достижения проекта включают:

- Разработку и успешное развертывание полнофункциональной CRM-системы.
- Реализацию основных функций управления клиентами, задачами и продажами.
- Обеспечение высокой безопасности и надежности системы.
- Создание удобного и интуитивно понятного пользовательского интерфейса.
- Внедрение эффективных инструментов мониторинга и управления производительностью.

Эти результаты создают прочную основу для дальнейшего развития системы и её адаптации к изменяющимся потребностям компании, что позволит поддерживать высокий уровень конкурентоспособности и удовлетворенности клиентов.

8.3. Выводы и дальнейшие перспективы развития проекта

Проект разработки CRM-системы для компании по установке натяжных потолков достиг значительных результатов. Созданная система успешно удовлетворяет основные потребности компании, обеспечивая эффективное управление клиентской базой, задачами и продажами. Однако для поддержания конкурентоспособности и дальнейшего улучшения системы важно рассмотреть возможные направления её развития.

Анализ достижений

1. Разработка и развертывание CRM-системы:

- Полнофункциональная система была успешно разработана и внедрена.
- Реализованы основные модули для управления клиентами, задачами и продажами.

2. Обеспечение безопасности:

- Внедрены механизмы аутентификации и авторизации с использованием Spring Security.
- Настроены права доступа для различных ролей пользователей, обеспечивая защиту данных.

3. Мониторинг и управление производительностью:

- Использованы Spring Boot Actuator, Micrometer, Prometheus и Grafana для мониторинга состояния приложения и визуализации данных.

4. Пользовательский интерфейс:

- Разработан интуитивно понятный и отзывчивый пользовательский интерфейс, обеспечивающий удобное взаимодействие с системой.

5. Качество и тестирование:

- Проведено тщательное тестирование на всех этапах разработки, что обеспечило высокое качество конечного продукта.

Дальнейшие перспективы развития проекта

1. Расширение функционала

Для удовлетворения растущих потребностей компании и её клиентов важно постоянно расширять функционал CRM-системы. Возможные направления включают:

- Автоматизация маркетинговых кампаний: Внедрение инструментов для планирования и проведения маркетинговых кампаний, а также анализа их эффективности.
- Управление складскими запасами: Добавление функционала для отслеживания и управления складскими запасами, что позволит оптимизировать процесс закупок и сократить издержки.
- Интеграция с электронной коммерцией: Разработка модулей для интеграции с платформами электронной коммерции, что упростит процесс продаж и повысит удобство для клиентов.

2. Интеграция с другими системами

Интеграция CRM-системы с другими системами компании и внешними сервисами позволит значительно расширить её возможности и улучшить взаимодействие между различными подразделениями:

- Интеграция с ERP-системами: Обмен данными с ERP-системами поможет синхронизировать бизнес-процессы, улучшить управление ресурсами и повысить общую эффективность компании.

- **Интеграция с системами телефонии:** Внедрение интеграции с системами IP-телефонии позволит улучшить качество обслуживания клиентов за счет автоматического ведения журнала звонков и записи разговоров.
- **API для внешних сервисов:** Разработка API для интеграции с внешними сервисами, такими как платежные системы и службы доставки, упростит обработку заказов и улучшит пользовательский опыт.

3. Улучшение производительности и масштабируемости

Для обеспечения стабильной работы CRM-системы при увеличении нагрузки и объема данных необходимо провести оптимизацию производительности и масштабируемости:

- **Оптимизация запросов к базе данных:** Анализ и оптимизация наиболее ресурсоемких запросов к базе данных для сокращения времени отклика и уменьшения нагрузки на сервер.
- **Использование кэширования:** Внедрение кэширования данных на различных уровнях системы (например, с использованием Redis или Memcached) для уменьшения количества обращений к базе данных.
- **Масштабирование приложения:** Рассмотрение возможности перехода от монолитной архитектуры к микросервисной архитектуре для более гибкого масштабирования и повышения отказоустойчивости системы.
- **Облачные технологии:** Перенос системы в облачные среды (например, AWS, Azure или Google Cloud) для обеспечения высокой доступности и масштабируемости, а также оптимизации затрат на инфраструктуру.

Заключение

Достигнутые результаты и описанные перспективы развития CRM-системы демонстрируют успешность проекта и его значимость для компании. Внедренная система уже сейчас обеспечивает повышение эффективности работы, улучшение качества обслуживания клиентов и увеличение продаж. Однако, для поддержания конкурентоспособности и дальнейшего роста компании необходимо продолжать развивать и улучшать систему, учитывая меняющиеся потребности рынка и технологические возможности.

9. Приложения

Приложения включают в себя вспомогательные материалы и исходный код, необходимые для полного понимания и воссоздания CRM-системы. Эти разделы содержат конкретные примеры и технические детали, которые поддерживают основные разделы дипломной работы.

9.1. Код приложения «CRM»

Кода приложения, включая основные модули и компоненты, реализованные в проекте прикладывается к архиву.