МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА СИСТЕМЫ УПРАВЛЕНИЯ ДОСТАВКОЙ ТОВАРОВ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Мустафаева Амирхана Радифовича

Научный руководитель
зав. каф. к. ф.-м. н. _____ С. В. Миронов

Зав.кафедрой,
к. ф.-м. н. _____ С. В. Миронов

Саратов 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире, где информационные технологии занимают центральное место в различных сферах деятельности, управление логистическими операциями становится все более сложным и критически важным процессом. Для компаний малого и среднего бизнеса, эффективное и точное управление заявками на доставку является неотъемлемой частью их операционного здоровья и успеха. Тем не менее, традиционные методы ведения учета часто оказываются недостаточно гибкими и удобными, что приводит к потерям времени и ресурсов. В таких условиях на первый план выходит необходимость разработки специализированных программных решений, способных автоматизировать и оптимизировать процесс управления заявками на доставку.

Целью данной дипломной работы является разработка веб-приложения для управления заявками на доставку, которое будет удовлетворять потребности малого и среднего бизнеса. Приложение должно предоставлять работодателям и рабочим интуитивно понятный интерфейс для создания, управления и мониторинга заявок на доставку в режиме реального времени.

Задачи, поставленные в рамках дипломной работы:

- изучение существующих решений и технологий
- разработка архитектуры и функционала приложения
- обеспечение безопасности данных
- разработка удобного пользовательского интерфейса
- тестирование работы приложения

Практическая значимость данного исследования заключается в создании эффективного инструмента, способного существенно упростить и ускорить процесс управления заявками на доставку, снизить количество ошибок и улучшить общую продуктивность работы компаний.

Таким образом, данная дипломная работа направлена на создание надежного и удобного веб-приложения для управления заявками на доставку, что будет способствовать повышению эффективности бизнес-процессов и конкурентоспособности компаний малого и среднего бизнеса на рынке.

1 Описание продукта

Разрабатываемый продукт представляет собой веб-приложение для управления заявками на доставку, предназначенное для использования в малом и среднем бизнесе. Основной функционал приложения включает создание, редактирование, просмотр и управление статусами заявок на доставку. Система предоставляет возможность пользователям с различными ролями, такими как работодатели и рабочие, взаимодействовать с учетными записями и осуществлять свои обязанности в рамках четко определенных прав доступа.

Одним из ключевых преимуществ нашего веб-приложения является его интуитивно понятный интерфейс, который позволяет пользователям быстро освоиться и начать работать без необходимости прохождения длительного обучения.

Важное место занимает разработка подсистемы авторизации, которая реализована с использованием Spring Security. Это позволяет надежно защитить данные пользователей и предотвратить несанкционированный доступ к информации о доставках.

Кроме того, приложение поддерживает возможность масштабирования и добавления новых функций. Это достигается благодаря модульной архитектуре, что делает его гибким и легко адаптируемым к изменяющимся требованиям пользователей.

2 Выбор технологий

При выборе технологического стека для разработки системы управления заявками на доставку стояло несколько ключевых задач, связанных с обеспечением надежности, производительности и расширяемости приложения.

В первую очередь было принято решение использовать Java в качестве основного языка программирования. Java предоставляет широкие возможности для разработки масштабируемых и надежных приложений благодаря своей кроссплатформенности, строгой типизации, обширному сообществу разработчиков и богатому набору библиотек .

В качестве фреймворка для разработки был выбран Spring Framework. Spring обеспечивает мощные инструменты для создания Enterprise-уровневых приложений, включая управление зависимостями, внедрение зависимостей (DI), управление транзакциями, обеспечение безопасности и многое другое. В частности, использование Spring Boot значительно упрощает настройку и развертывание приложения, предоставляя конвенции по умолчанию и автоматическую конфигурацию .

Для обеспечения асинхронной обработки и взаимодействия с клиентами было решено использовать RESTful API. REST позволяет создавать гибкие и расширяемые интерфейсы, обеспечивающие прозрачную интеграцию с другими системами.

Важным аспектом выбора технологий является поддержка базы данных PostgreSQL. PostgreSQL обеспечивает высокую производительность, масштабируемость и надежность при работе с большими объемами данных. Его поддержка транзакций, расширяемость и возможности для полнотекстового поиска делают его оптимальным выбором для хранения и обработки данных, включая заявки на доставку .

3 Технические и программные средства

Для разработки веб-приложения управления заявками на доставку были использованы современные технические и программные средства, обеспечивающие высокую производительность, безопасность и удобство использования. Основной технологической платформой для разработки серверной части приложения был выбран фреймворк Spring Boot, который предоставляет мощные инструменты для создания масштабируемых и легко поддерживаемых приложений на языке Java. Spring Boot упрощает настройку и разработку, обеспечивая встроенную поддержку для работы с базами данных, аутентификации и авторизации пользователей, а также интеграции с другими сервисами.

Для обеспечения безопасности была использована библиотека Spring Security, которая предоставляет мощные средства для защиты приложений. Spring Security позволяет легко настроить различные уровни доступа для разных ролей пользователей, обеспечивает надежную аутентификацию и авторизацию.

Для работы с базами данных использовалась технология JPA , которая упрощает процесс взаимодействия с реляционными базами данных, такими как PostgreSQL. PostgreSQL был выбран в качестве СУБД благодаря его надежности, высокой производительности и богатому набору функций для обеспечения целостности данных. JPA позволяет использовать объектно-ориентированный подход к работе с данными, что значительно упрощает разработку и поддержку приложения.

Фронтенд-часть приложения была реализована с использованием фреймворка Vue.js, который позволяет создавать динамичные и интерактивные пользовательские интерфейсы. Vue.js отличается простотой использования и высокой производительностью, что делает его идеальным выбором для разработки современных веб-приложений.

Таким образом, комбинация современных технологий и инструментов, таких как Spring Boot, Spring Security, JPA и Vue.js, позволяет создать надежное, безопасное и удобное в использовании веб-приложение для управления заявками на доставку. Эти технические и программные средства обеспечивают высокую производительность и масштабируемость приложения, удовлетворяя потребности малого и среднего бизнеса в эффективном управлении процессом доставки.

4 Аутентификация

Аутентификация в проекте, построенном с использованием Spring Boot, реализована через REST API и JWT (JSON Web Token). Основной целью системы аутентификации является обеспечение безопасного доступа пользователей к ресурсам и функциональности приложения. Для этого используются несколько ключевых компонентов, включая контроллеры, сервисы и фильтры, которые обеспечивают регистрацию и аутентификацию пользователей.

4.1 Контроллер аутентификации

Контроллер аутентификации AuthenticationController отвечает за обработку HTTP-запросов, связанных с регистрацией и аутентификацией пользователей. Он определяет два основных эндпоинта: /register и /authenticate. Эти эндпоинты принимают POST-запросы с телом, содержащим данные для регистрации или аутентификации пользователя. Контроллер взаимодействует с сервисом аутентификации AuthenticationService для выполнения необходимых операций.

4.2 Модель запросов и ответов

Для обмена данными между клиентом и сервером используются модели RegisterRequest, AuthenticationRequest и AuthenticationResponse. Эти классы содержат поля, необходимые для передачи информации о пользователе и токене аутентификации.

4.3 Сервис аутентификации

Сервис аутентификации AuthenticationService реализует основную логику для регистрации и аутентификации пользователей. При регистрации новый пользователь сохраняется в базе данных после хэширования его пароля. Затем для пользователя генерируется JWT, который возвращается клиенту. В процессе аутентификации сервис проверяет предоставленные учетные данные, а затем также генерирует и возвращает JWT при успешной аутентификации.

В методе authenticate сервис проверяет учетные данные пользователя, используя AuthenticationManager. Если аутентификация успешна, сервис загружает данные пользователя из базы данных через UserRepository. В случае успешного нахождения пользователя для него генерируется JWT с помощью JwtService.

Этот токен затем возвращается клиенту в ответе AuthenticationResponse. Если учетные данные неверны, выбрасывается исключение AuthenticationException.

4.4 JWT и фильтрация запросов

Для управления JWT используется сервис JwtService, который предоставляет методы для генерации, проверки и извлечения информации из токенов. JwtAuthenticationFilter является фильтром, который перехватывает каждый HTTP-запрос и проверяет наличие и валидность JWT. Если токен валиден, фильтр устанавливает аутентификацию в контексте безопасности.

Если имя пользователя извлечено из токена и пользователь еще не аутентифицирован, фильтр загружает детали пользователя из userDetailsService. Затем он проверяет валидность токена с помощью jwtService. Если токен действителен, создается объект UsernamePasswordAuthenticationToken, содержащий детали пользователя и его полномочия. Этот объект используется для установки аутентификации в контексте безопасности SecurityContextHolder.В конце фильтр пропускает запрос и ответ дальше по цепочке фильтров, независимо от того, была ли аутентификация успешно выполнена или нет.

4.5 Конфигурация безопасности

Конфигурация безопасности в данном проекте управляется с помощью класса SecurityConfiguration. Этот класс предназначен для настройки правил доступа к различным URL-адресам и определения использования JWT-фильтра и провайдера аутентификации.

Прежде всего, класс аннотируется @Configuration и @EnableWebSecurity, что указывает Spring, что это класс конфигурации безопасности. Аннотация @RequiredArgsConstructor автоматически создает конструктор с параметрами для всех final полей, что упрощает внедрение зависимостей.

Настройка фильтра цепочки безопасности Основной метод в этом классе — securityFilterChain, который возвращает объект SecurityFilterChain. Внутри этого метода производится настройка безопасности.

4.6 Настройка CORS

Метод corsConfigurationSource настраивает CORS, позволяя клиентам из других доменов взаимодействовать с сервером. Здесь указывается, какие домены, методы и заголовки разрешены для междоменных запросов.

Таким образом, данная конфигурация обеспечивает безопасность доступа к API, ограничивая доступ к определенным эндпоинтам только авторизованным пользователям с определенными ролями и применяя JWT токены для аутентификации. Эти меры позволяют обеспечить надежную защиту и контролируемый доступ к ресурсам вашего приложения.

5 База данных

Для разработки базы данных в рамках дипломного проекта была спроектирована структура, включающая несколько ключевых таблиц для управления пользователями, заявками и товарами.

Таблица _user предназначена для хранения информации о пользователях системы. В данной таблице определены следующие поля: id (идентификатор пользователя), username (имя пользователя для входа), email (электронная почта), phone_number (номер телефона), password (захешированный пароль), full_name (полное имя пользователя) и role (роль пользователя в системе). Поле role используется для определения прав доступа пользователя, что позволяет различать администраторов, менеджеров и обычных пользователей.

Таблица Supplier представляет собой сущность поставщика товаров. Она включает в себя поля id (идентификатор поставщика), name (название компании поставщика), address (адрес) и phone (контактный телефон). Данная таблица служит для хранения информации о компаниях, от которых закупаются товары.

Таблица Customer предназначена для хранения данных о клиентах. Она содержит аналогичные поля, как и таблица поставщиков, включая id, name, address и phone. Эта таблица используется для хранения информации о компаниях или частных лицах, которые являются клиентами вашей системы.

Таблица Product отвечает за хранение информации о продуктах, которые могут быть включены в заявку. Она включает в себя поля id (идентификатор продукта), name (название продукта), price (цена продукта), weight (вес продукта) и article (артикул или уникальный идентификатор продукта). Эта таблица предоставляет данные о товарах, которые могут быть приобретены или проданы через вашу систему.

Таблица Invoice используется для представления заявок на доставку. Она включает в себя поля id (идентификатор заявки), totalCost (общая стоимость заявки), supplier_id (идентификатор поставщика, связанный с таблицей Supplier), customer_id (идентификатор клиента, связанный с таблицей Customer), а также status (статус заявки, представленный в виде булевого значения). Эта таблица позволяет хранить информацию о финансовых операциях, связанных с покупками и продажами товаров между компаниями.

Таблица InvoiceProduct является связующей таблицей между таблицами Invoice и Product. Она содержит поля id (идентификатор записи), invoice_id

(идентификатор заявки, связанный с таблицей Invoice), product_id (идентификатор продукта, связанный с таблицей Product) и count (количество единиц продукта в заявке). Эта таблица позволяет организовать множественные связи между заявками и товарами, что позволяет включать множество товаров в одну заявку на доставку.

Каждая из этих таблиц спроектирована с учетом своей специфики и предназначения, обеспечивая эффективное хранение и управление данными.

6 Заявка на доставку

Заявка на доставку представляет собой ключевой элемент в бизнес-процессах, отражающий финансовые транзакции между поставщиками и покупателями. В проекте заявки на доставку моделируются с использованием Spring Data JPA, что обеспечивает удобство работы с базой данных и управление отношениями между сущностями.

6.1 Создание заявки

Метод контроллера принимает POST-запрос для создания новой заявки на доставку на основе данных, переданных в теле запроса (InvoiceRequest). В сервисе InvoiceService происходит сохранение поставщика, покупателя и заявки на доставку, а также связанных с ней продуктов (InvoiceProduct). Вычисляется общая стоимость заявки на основе цен продуктов и их количества.

Полный код файла контроллера заявок на доставку представлен в приложении

6.2 Получение заявки на доставку по ID

Метод контроллера позволяет получить заявку на доставку по ее уникальному идентификатору (id). Если заявка найдена в базе данных, она возвращается клиенту в теле ответа. В противном случае возвращается код 404 Not Found.

6.3 Обновление статуса заявки

Метод контроллера обновляет статус заявки по ее идентификатору (id). Новый статус передается в теле запроса в формате JSON. В сервисе InvoiceService обновляется статус заявки на доставку в базе данных.

Совокупность вышеуказанных методов позволяет эффективно управлять жизненным циклом заявки на доставку в вашем приложении. Создание новой заявки, получение существующей по идентификатору и обновление ее статуса позволяют полностью охватить основные операции, необходимые для работы с данными о финансовых транзакциях. Использование Spring Data JPA упрощает взаимодействие с базой данных, предоставляя мощные инструменты для управления персистентными объектами и их отношениями.

ЗАКЛЮЧЕНИЕ

В результате выполнения данной дипломной работы было разработано веб-приложение для управления заявками на доставку, которое обеспечивает удобный и интуитивно понятный интерфейс для малого и среднего бизнеса. Приложение включает следующий функционал:

- регистрация и аутентификация пользователей;
- создание и редактирование заявок на доставку;
- просмотр и управление списком заявок на доставку;
- обновление статуса заявок на доставку;

В ходе работы были рассмотрены и применены современные технологии и инструменты разработки веб-приложений. Было изучено и внедрено использование фреймворка Spring с его основными компонентами: Spring Data, Spring Boot, Spring Core и Spring Security. Применение этих технологий позволило создать надежную и масштабируемую архитектуру приложения. Также был изучен и реализован процесс настройки CORS для обеспечения безопасности междоменных запросов.