Geekbrains

**Разработка приложения «Служба доставки»**

Программист Java. Цифровые профессии

Т.Д. Саркисян

Москва

2024

**Дипломный проект**

**«Разработка приложения «Служба доставки»**

**Содержание**

|  |  |
| --- | --- |
| Введение……………………………………………………………………. | 3 |
| Глава 1. Предпроектная стадия…………………………………………… | 4 |
| 1.1. Порядок проведения работ…………………………………………… | 4 |
| 1.2. Определение требований к приложению…………………………… | 5 |
| 1.3. Выбор стека технологий……………………………………………… | 8 |
| Глава 2. Проектная стадия………………………………………………… | 11 |
| 2.1. Проектирование общей архитектуры приложения………………… | 11 |
| 2.2. Проектирование web-интерфейса приложения…………………….. | 13 |
| 2.3. Проектирование базы данных……………………………………….. | 14 |
| 2.4. Безопасность приложения……………………………………………. | n |
| Заключение………………………………………………………………… | n |
| Список использованной литературы…………………………………….. | n |
|  |  |

**Введение**

Часто для успешного функционирования различным организациям требуется развитая информационная система, реализующая автоматизированный процесс сбора, манипулирования и обработки данных. Одним из наиболее удобного средства для рационального и эффективного хранения информации являются базы данных. Приложения базы данных обеспечивают надежную защиту данных от случайной потери или порчи, экономно используют ресурсы, как людские, так и технические и снабжены механизмами поиска информации, удовлетворяющим требованиям   
к производительности системы.

В связи с этим было решено создать веб приложение службы доставки.

Приложение представляет собой удобный инструмент для отслеживания и анализа оборота товаров. С его помощью пользователь заказать и отследить доставку товара.

Пользователь может просматривать статистику своих заказов   
за определенный период времени.

**Уведомления и напоминания**: Приложение может отправлять пользователю уведомления, напоминая о статусе заказа.

**Конфиденциальность и безопасность:** Для защиты личных данных пользователей приложение обеспечивает возможность установки пароля.

**Глава 1. Предпроектная стадия**

**1.1. Порядок проведения работ**

В ходе создания проекта были проведены следующие виды работ:

1. **Планирование проекта**:
   * Определение функциональных и нефункциональных требований к приложению.
   * Проектирование архитектуры приложения и выбор используемых технологий.
2. **Настройка окружения разработки**:
   * Установка и настройка JavaDevelopmentKit (JDK), ApacheMaven (для сборки проекта), и SpringBoot (для создания и запуска приложения).
3. **Создание проекта**:
   * Создан новый проект с использованием SpringInitializr.
   * Выбраны необходимые зависимости, такие как SpringWeb, SpringDataJPA, SpringSecurity, Thymeleaf, для поддержки функциональности приложения.
4. **Разработка модели данных, репозитория, сервисов, контроллера.**
5. **Разработка пользовательского интерфейса**:
   * Создание HTML-шаблонов с использованием Thymeleaf (или других шаблонизаторов) для отображения данных и взаимодействия с пользователем.
6. **Отладка и оптимизация**:
   * Отладка кода для исправления ошибок и улучшения производительности.
   * Оптимизация кода и конфигурации приложения для улучшения скорости работы и эффективности использования ресурсов.
7. **Документирование**:
   * Создание документации по проекту, включая описание архитектуры, инструкции по установке и запуску, описание API (если применимо) и другие справочные материалы.

**1.2. Определение требований к приложению**

Для разработки приложения «Служба доставки» были определены следующие требования:

1. Функциональные требования:
   * Пользователи должны иметь возможность создавать заказы.
   * Приложение должно предоставлять возможность категоризации товаров для удобного анализа и отслеживания.
   * Пользователи должны иметь возможность просматривать и редактировать свои заказы.
   * Возможность получения советов и рекомендаций по управлению эмоциями на основе данных пользователя.
2. Нефункциональные требования:
   * Удобство использования: Интерфейс должен быть интуитивно понятным и привлекательным для пользователя.
   * Безопасность: Данные пользователей должны храниться и передаваться в зашифрованном виде.
   * Производительность: Приложение должно обеспечивать быстрый доступ к данным и отзывчивый интерфейс.
   * Масштабируемость: Возможность расширения функциональности и увеличения числа пользователей без значительного ухудшения производительности.
3. Требования к интерфейсу пользователя:
   * Интуитивно понятный интерфейс, который позволяет пользователям легко создавать и просматривать записи о своих чувствах.
   * Возможность выбора цветовой схемы или темы интерфейса для создания комфортной атмосферы.
4. Требования к безопасности:
   * Аутентификация и авторизация пользователей.
   * Защита данных пользователей от несанкционированного доступа.
5. Требования к производительности:
   * Быстрый доступ к данным и отзывчивый интерфейс приложения.
6. Требования к масштабируемости:
   * Возможность добавления новых функций и поддержка роста числа пользователей.
7. Требования к поддержке и обновлениям:
   * Поддержка приложения и обновления для улучшения функциональности и безопасности.

**1.3. Выбор стека технологий**

1. **Язык программирования**: Java для разработки серверной части приложения так как этот язык хорошо подходит для создания масштабируемых и надежных веб-приложений.
2. **Фреймворк для веб-разработки**: SpringBoot с модулями Spring MVC и SpringSecurity для быстрой и удобной разработки серверной части приложения. SpringSecurity обеспечит безопасность приложения, а Spring MVC позволит создать удобный интерфейс.
3. **База данных**: MySQL для хранения данных пользователей и их эмоций. Обе системы хорошо масштабируются и обеспечивают надежное хранение данных.
4. **Хранение данных о чувствах**: для хранения данных о чувствах и эмоциях пользователей используется **MySQL.**
5. **Шаблонизатор**: Thymeleaf для генерации HTML-страниц на сервере. Thymeleaf интегрируется хорошо со Spring и позволяет удобно работать с данными из Java кода.
6. **Аутентификация и авторизация**: SpringSecurity используется для обеспечения безопасности приложения, включая аутентификацию пользователей и управление доступом к ресурсам.
7. **Среда разработки**: IntelliJ IDEA для разработки серверной части приложения.   
   Архитектура приложения в рамках JavaSpring основывается на паттерне проектирования MVC (Model-View-Controller), который позволяет разделить приложение на три основных компонента: модель (Model), представление (View) и контроллер (Controller). Наше приложение состоит из данных составных частей:
8. **Модель (Model)**:
   * Модель представляет собой структуру данных приложения.
   * Включает классы, описывающие сущности приложения ("Пользователь", "Запись чувств").
9. **Представление (View)**:
   * Представление отвечает за отображение данных пользователю.
   * В веб-приложениях на JavaSpring представление часто реализуется с использованием шаблонизаторов, которые позволяют вставлять данные из контроллера непосредственно в HTML-шаблоны.
10. **Контроллер (Controller)**:
    * Контроллер обрабатывает запросы от пользователя и взаимодействует с моделью и представлением.
    * В JavaSpring контроллеры представлены классами, аннотированными **@Controller** или **@RestController**.
    * Контроллеры содержат методы для обработки различных типов запросов (GET, POST, PUT, DELETE) и вызывают соответствующие сервисы для выполнения бизнес-логики.
11. **Репозиторий (Repository)**:
    * Репозиторий отвечает за доступ к данным из базы данных.
    * В JavaSpring репозитории обычно реализуются с помощью интерфейсов, расширяющих **JpaRepository**, которые предоставляют базовые методы для работы с данными.
    * Репозитории используются сервисами для получения, сохранения, обновления и удаления данных.
12. **Сервис (Service)**:
    * Сервис содержит бизнес-логику приложения.
    * Он обычно используется для выполнения операций с данными, координирования действий между различными компонентами приложения и обеспечения единого доступа к функциональности приложения.
13. **Сервер (Server)**:
    * Сервер представляет собой окружение, в котором работает приложение.
    * В JavaSpring приложение обычно запускается на встроенном сервере, который обрабатывает HTTP-запросы и управляет жизненным циклом приложения.

Каждый из этих компонентов играет важную роль в построении архитектуры приложения на JavaSpring, обеспечивая его модульность, гибкость и масштабируемость. Они позволяют разделить различные аспекты приложения и обеспечить их независимую разработку и тестирование.

|  |
| --- |
| **Глава 2. Проектная стадия** |
| **2.1. Проектирование общей архитектуры приложения** |

Первое, с чего начиналась работа — это класс с инициализацией полей класса Product. Класс Product представляет собой запись о товаре. Каждая запись содержит информацию о цене (price), изображении (image), описания (description).

DefaultController является компонентом веб-приложения, отвечающим   
за взаимодействие с пользователем и обработку запросов, связанных   
с управлением записями дневника чувств и справочника. Контроллер предоставляет различные методы для обработки запросов пользователей и взаимодействия с соответствующими сервисами для выполнения операций   
с данными.

Методы контроллера позволяют просматривать, добавлять, редактировать записи в справочнике товаров. Кроме того, контроллер обеспечивает перенаправление пользователя на соответствующие страницы для отображения результатов операций.

Для обеспечения безопасности приложения контроллер также взаимодействует с модулем безопасности (например, SpringSecurity), обеспечивая аутентификацию и авторизацию пользователей при доступе к различным функциям приложения.

Методы контроллера обрабатывают различные HTTP-запросы, например, GET и POST, для отображения страниц, получения и обновления данных.

Кроме того, контроллер содержит методы для удаления типов продукта (productTypeListDelete) и редактирование существующих (productTypeListEdit). Эти методы позволяют пользователям управлять своими данными и вносить изменения в уже существующие.

В целом, DefaultController является важной частью приложения, обеспечивающей удобный интерфейс для работы с данными.

Класс ProductType представляет собой запись в справочнике. Каждая запись содержит уникальный идентификатор (id), тип товара (name).

Класс MyUserDetails реализует интерфейс UserDetails, который используется SpringSecurity для представления информации о пользователе, необходимой для аутентификации и авторизации. Вот его функции:

1. privateUseruser; - поле, хранящее объект User, который представляет собой модель пользователя вашего приложения.
2. publicMyUserDetails(Useruser){...} - конструктор класса, принимающий объект User и сохраняющий его в поле user.
3. getAuthorities() - возвращает коллекцию ролей (GrantedAuthority), которые имеет пользователь. В данном случае, роли берутся из поля role объекта User и преобразуются в GrantedAuthority.
4. getPassword() и getUsername() - возвращают пароль и имя пользователя соответственно, из объекта User.
5. isAccountNonExpired(), isAccountNonLocked(), isCredentialsNonExpired(), isEnabled() - методы, возвращающие булево значение, указывающее, истек ли срок действия учетной записи, заблокирована ли учетная запись, устарели ли учетные данные и включена ли учетная запись. В данном случае, все они всегда возвращают true, что означает, что учетная запись всегда действительна, не заблокирована, учетные данные не устарели и учетная запись всегда включена.

Этот класс используется вместе с вашим UserDetailsService, чтобы предоставить

**2.2. Проектирование WEB – интерфейса приложения**

Для проектирования WEB – интерфейса приложения был использован HTML. После прохождения авторизации пользователь попадает на страницу home.html.

Данный HTML-код представляет собой шаблон для веб-страницы, использующей Thymeleaf в качестве шаблонизатора. Thymeleaf позволяет создавать динамические веб-страницы, вставляя данные из модели Java в HTML-шаблон. В данном случае, предполагается, что на сервере настроены контроллеры для обработки запросов, указанных в атрибуте action форм.

Давайте разберем основные элементы данного шаблона:

<!DOCTYPE html>: определяет тип документа и его версию.

<html lang="en" xmlns:th="http://www.thymeleaf.org">: указывает   
на использование Thymeleaf в качестве шаблонизатора и определяет пространство имен для Thymeleaf.

<head>: содержит метаданные страницы, такие как заголовок (<title>).

<style>: секция стилей для определения внешнего вида элементов страницы.

<body>: Основная часть страницы, содержащая контент, видимый пользователю.

Формы (<form>): используются для создания кнопок, которые выполняют определенные действия при нажатии. Атрибут action указывает на адрес, куда будет отправлен запрос при отправке формы, а method определяет HTTP-метод (GET или POST).

Кнопки (<button>): cодержатся внутри форм и используются для выполнения определенных действий на сервере при их нажатии.

**2.3. Проектирование базы данных**

В качестве базы данных для этого проекта был выбран MySQL. В файле application.yaml указаны следующие настройки.

В данном примере конфигурации SpringBoot для подключения к базе данных PostgreSQL через JPA и Hibernate используется следующая конфигурация:

1. datasource: cодержит настройки для подключения к базе данных. driver-class-name указывает класс драйвера MySQL. url содержит URL-адрес для подключения к базе данных. username и password содержат учетные данные пользователя базы данных.
2. jpa: cодержит настройки JPA и Hibernate. hibernate.ddl-auto определяет стратегию автоматического создания таблиц в базе данных. Значение update указывает Hibernate на автоматическое обновление схемы базы данных при изменении сущностей JPA.

….