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

Кафедра информатики и программирования

РАЗРАБОТКА WEB-ПРИЛОЖЕНИЯ ДЛЯ ФОРМИРОВАНИЯ ЗАКАЗОВ ПРОКАТНОЙ КОМПАНИИ С ИСПОЛЬЗОВАНИЕМ SPRING FRAMEWORK И REACT.JS

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 441 группы
направления 02.03.03 Математическое обеспечение и администрирование информационных систем
факультета компьютерных наук и информационных технологий
Шмыгина Семена Сергеевича

Научный руководитель,
проф. кафедры информатики и
программирования, д.т.н. _____ А.С. Фалькович

подпись, дата

Зав. кафедрой ИиП,
к.ф.-м.н., доцент _____ М.В. Огнева

подпись, дата

Саратов 2022

ВВЕДЕНИЕ

Особенностью работы прокатных компаний является то, что прокатное оборудование предоставляется клиентам на определенное время, поэтому возникает необходимость контролировать процесс возврата оборудования и сроки возврата. Рассматриваемая прокатная компания занимается сдачей в аренду различного рода оборудования для мероприятий, такие как концерты, корпоративные мероприятия, свадьбы и т.д.

Подобные мероприятия нуждаются в техническом оснащении музыкальным и световым оборудованием. Заказчик может связаться с менеджером прокатной компании, обсудить количество необходимого оборудования и оформить заказ.

В рамках данной бакалаврской работы будет разработано web-приложение для учета оборудования на складе и формирования заказов для одной из таких прокатных компаний. Приложение будет состоять из серверной и клиентской части. Серверная часть будет разработана на языке Java с использованием Spring Framework. Клиентская часть, в свою очередь, будет разработана на языке JavaScript с использованием библиотеки React.js.

Актуальность данной бакалаврской работы состоит в том, что в прокатной компании, для которой будет реализовано приложение, отсутствует корпоративное программное обеспечение для формирования заказов и учета оборудования на складе. Наличие подобного приложения облегчит проведение бизнес-процессов.

Целью данной бакалаврской работы является разработка web-приложения для учета оборудования и формирования заказов, которое будет сопровождаться аутентификацией пользователей данной прокатной компании.

Поставленная цель определила **следующие задачи**:

1. Изучение возможных архитектур проектирования приложений и выбор подходящей архитектуры для реализации приложения;
2. Изучение Spring Framework и реализация серверной части приложения с его помощью;
3. Изучение библиотеки React.js и реализация клиентской части с ее помощью;
4. Организация взаимодействия серверной и клиентской частей приложения.

Методологические основы разработки web-приложений с использованием Spring Framework и React.js представлены в работах Б. Эккеля, У. Крейга, Д. Валке.

Практическая значимость бакалаврской работы состоит в том, что в прокатной компании, для которой было разработано приложение, присутствует система учета оборудования на складе и сформированных заказах, но пользование ей недостаточно удобно. Таким образом, было решено разработать web-приложение, обладающее подобным функционалом, которое было бы удобнее использовать.

Структура и объём работы. Бакалаврская работа состоит из введения, четырех разделов, заключения, списка использованных источников и одиннадцати приложений. Общий объем работы – 71 страница, из них 46 страниц – основное содержание, включая 18 рисунков, список использованных источников информации – 20 наименований.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

Первый раздел «Существующий способ хранения данных о заказах и оборудовании» посвящен описанию того, каким образом хранятся данные об оборудовании на складе и сформированных заказах на данный момент, приведен пример того, почему данный способ хранения данных является недостаточно удобным в использовании.

Также определен основной функционал разрабатываемого web-приложения, а именно:

1. организация хранения данных об оборудовании и заказах в базе данных;
2. разработка серверного приложения для манипуляции данными в базе;
3. разработка клиентского приложения, которое будет взаимодействовать с серверным приложением;
4. реализация возможности управления учетными записями пользователей.

В роли пользователей выступают сотрудники прокатной компании.

Итоги. В данном разделе удалось в полной мере понять актуальность данной бакалаврской работы, а также определить главные требования к разрабатываемому приложению.

Второй раздел «Теоретические сведения о разработке web-приложений» посвящен описанию существующих архитектур архитектур web-приложений.

В пункте 2.1 рассмотрено что такое архитектура приложения и ее назначение.

Архитектура приложения описывает шаблоны и методы, используемые для разработки. Архитектура дает разработчикам план и рекомендации, которым нужно следовать, чтобы в итоге они получили хорошо структурированное приложение.

В пункте 2.1.1 приведено определение монолитной архитектуры, ее достоинства и недостатки.

Монолитная архитектура — тип архитектуры, который представляет собой единый стек приложений, который содержит все функции в рамках одного приложения. Это тесно связано как во взаимодействии между услугами, так и в том, как они разрабатываются и предоставляются.

В пункте 2.1.2 приведено определение микросервисной архитектуры, ее достоинства и недостатки.

Микросервисная архитектура — это и архитектура, и подход к написанию программного обеспечения. В микросервисах приложения разбиваются на мельчайшие компоненты, независимые друг от друга. Каждый из этих компонентов или процессов является микрослужбой.

В пункте 2.2 рассмотрено, что представляет собой Spring Framework и приведены примеры компонентов, из которых он состоит.

Spring Framework — это платформа Java, обеспечивающая комплексную поддержку инфраструктуры для разработки приложений Java. Spring управляет инфраструктурой, поэтому при разработке приложения можно сосредоточиться именно на деталях реализации

В пункте 2.2.1 рассмотрен механизм внедрения зависимостей и инверсии управления, приведены его преимущества и недостатки.

Инверсия управления (IoC или Inversion of Control) — это принцип разработки программного обеспечения, который передает управление объектами или частями программы контейнеру или фреймворку. Чаще всего мы используем его в контексте объектно-ориентированного программирования.

Внедрение зависимостей (Dependency Injection или DI) — это шаблон, который использется для реализации IoC, где инвертируемый элемент управления устанавливает зависимости объекта.

В пункте 2.3 рассмотрена библиотека React.js, ее преимущества, которые объясняют популярность данной библиотеки.

React.js — это библиотека для разработки пользовательского интерфейса на основе JavaScript.

В пункте 2.3.1 рассмотрено расширение языка Java Script JSX, его назначение и пример использования.

В пункте 2.3.2 представлено описание объектной модели документа (DOM) и принцип его действия.

DOM — это представление пользовательского интерфейса (user interface, UI) в приложении.

В пункте 2.3.3 приведено описание того, что представляют собой компоненты в библиотеке React.js, и основные функции компонентов, а также приведен пример того, как выглядит пользовательский интерфейс, разбитый на компоненты.

React разделяет пользовательский интерфейс на множество **компонентов**, что делает отладку более доступной, и каждый компонент имеет собственный набор свойств и функций.

В пункте 2.4 рассмотрен протокол HTTP для обмена данными. Данный протокол используется в разработанном продукте для взаимодействия серверной и клиентской частей приложения.

Протокол передачи гипертекста (HTTP) — это метод кодирования и передачи информации между клиентом (например, веб-браузером) и веб-сервером. HTTP является основным протоколом для передачи информации через Интернет.

Итоги. В данном разделе были изучены основные технологии, которые использовались при разработке серверной и клиентской частей приложения.

Третий раздел «Разработка серверной части приложения» посвящен реализации серверной части приложения.

В пункте 3.1 приведена спроектированная база данных с использованием компонента Spring Framework — Spring Data JPA, который предлагает разработчикам организовать таблицы в базе данных на основе Java-классов.

В пункте 3.2 реализовано взаимодействие серверной части приложения с базой данных, сконфигурирован доступ сервера к базе данных. Также

приведено разделение серверного приложения на слои, посредством которых производится доступ к данным (слой репозиториев), их обработка (слой сервисов) и передача запрашивающей стороне (слой контроллеров).

В пункте 3.3 приведена конфигурация компонента Spring Security для ограничения доступа к серверным ресурсам, а также реализована аутентификация пользователей приложения с использованием JWT-токена, которым, после его получения в ходе аутентификации, будет сопровождаться каждый последующий запрос к серверу в специальном заголовке запроса Authorization.

Итоги. В данном разделе была разработана серверная часть приложения, а именно: спроектирована база данных, организовано взаимодействие сервера с базой данных и аутентификация пользователей с использованием JWT-токена.

Четвертый раздел «Разработка клиентской части приложения» посвящен реализации клиентской части приложения, посредством которого пользователь будет взаимодействовать с серверным приложением.

В пункте 4.1 представлена структура интерфейса, компонент плашки навигации, с помощью которой пользователь может переходить на следующие страницы:

1. Страница списка пользователей;
2. Страница списка оборудования;
3. Страница списка заказов.

Перейдя на каждую из приведенных страниц пользователь может добавить или удалить другого пользователя, оборудование или заказ соответственно.

Кроме того, в данном пункте показан внешний вид страницы входа в систему и страница профиля пользователя.

В пункте 4.2 реализована конфигурация Redux.js, которая позволяет хранить различного рода состояния в локальном хранилище и обеспечивает удобный способ работы с хранимыми данными.

В пункте 4.3 реализована страница списка пользователей, к которой имеют доступ только пользователи, наделенные правами администратора. Также приведены примеры того, как выглядят страница списка пользователей и страница добавления нового пользователя.

В пункте 4.4 реализована страница списка оборудования и приведены примеры того, как выглядит страницы списка оборудования, хранящегося на складе и добавления нового оборудования.

В пункте 4.5 реализована страница списка сформированных заказов, находясь на которой пользователь может добавить или удалить новое оборудование.

В пункте 4.6 подведены итоги разработки web-приложения и приведен отзыв менеджера прокатной компании, которому было показан получившийся продукт.

Итоги. В данном разделе была построена клиентская часть приложения, с которой будет взаимодействовать пользователь. Были разработаны основные страницы пользовательского интерфейса:

1. Страница входа в систему;
2. Страница профиля текущего пользователя;
3. Страница списка пользователей и добавления нового пользователя;
4. Страница списка оборудования и добавления нового оборудования;
5. Страница сформированных заказов и формирования нового заказа.

ЗАКЛЮЧЕНИЕ

В рамках данной бакалаврской работы было разработано приложение для прокатной компании, занимающейся арендой различного рода светового и звукового оборудования, а также техническим оснащением мероприятий, такие как концерты, корпоративные мероприятия, свадьбы и спектакли.

Были выполнены все поставленные задачи, а именно:

- Изучение возможных архитектур проектирования приложений и выбор подходящей архитектуры для реализации приложения;
- Изучение Spring Framework и реализация серверной части приложения с его помощью;
- Изучение библиотеки React.js и реализация клиентской части с ее помощью;
- Организация взаимодействия серверной и клиентской частей приложения.

Данное приложение было разработано с целью облегчения ведения бизнес-процессов. По сравнению с предыдущим способом хранения информации об оборудовании и заказов, данное приложение оказалось действительно более удобным инструментом для ведения бизнес-процессов.

Основные источники информации

1. What is an application architecture? [Электронный ресурс]. – URL: www.redhat.com/en/topics/cloud-native-apps/what-is-an-application-architecture (Дата обращения: 19.02.2022)
2. What is Monolithic Architecture? [Электронный ресурс]. – URL: www.integrate.io/glossary/what-is-monolithic-architecture/ (Дата обращения: 19.02.2022)
3. Advantages and Disadvantages of Microservices Architecture [Электронный ресурс]. – URL: cloudacademy.com/blog/microservices-architecture-challenge-advantage-drawback/ (Дата обращения: 19.02.2022)
4. Introduction to Spring Framework [Электронный ресурс]. — URL: https://docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/overview.html (дата обращения: 14.04.2022)
5. What Is React [Электронный ресурс]. – URL: https://www.simplilearn.com/tutorials/reactjs-tutorial/what-is-reactjs (Дата обращения: 21.03.2022)
6. What Is HTTP? [Электронный ресурс]. – URL: https://www.nginx.com/resources/glossary/http/ (Дата обращения: 19.02.2022)
7. What is a REST API? [Электронный ресурс]. — URL: https://www.redhat.com/en/topics/api/what-is-a-rest-api (дата обращения: 29.03.2022)
8. Уоллс К. Spring в действии. – М.: ДМК Пресс, 2013. – 752 с.: ил.