Отчет о проверке № 7999364

Дата выгрузки: 2023-05-11 00:49:34
Пользователь: dmitry.vert@mail.ru, ID: 7999364

Отчет предоставлен сервисом «Антиплагиат»
на сайте antiplagius.ru/

Информация о документе

№ документа: 7999364
Имя исходного файла: ИКБО-24-20_РКСП_Верт.docx
Размер файла: 6.69 МБ
Размер текста: 29695
Слов в тексте: 4045
Число предложений: 232

Информация об отчете

Дата: 2023-05-11 00:49:34 - Последний готовый отчет
Оценка оригинальности: 100%
Заимствования: 0%

100.00% 0%

Источники:

Доля в тексте	Ссылка

Информация о документе:

МИНОБРНАУКИ РОССИИФедеральное государственное бюджетное образовательное учреждение высшего образования "МИРЭА - Российский технологический университет" РТУ МИРЭА Институт информационных технологий (ИТ) Кафедра инструментального и прикладного программного обеспечения (ИиППО) КУРСОВАЯ РАБОТА по дисциплине: Разработка серверных частей интернет-ресурсов по профилю: Разработка программных продуктов и проектирование информационных систем направления профессиональной подготовки: 09.03.04 "Программная инженерия" Тема: "Разработка клиент-серверного фуллстек-приложения для размещения объявлений с использованием Spring Boot и React" Студент: Верт Дмитрий Андреевич Группа: ИКБО-24-20 Работа представлена к защите _____/Верт Д.А./ (подпись и ф.и.о. студента) Руководитель: Куликов Александр Анатольевич, к.т.н., доцент Работа допущена к защите _____/Куликов А.А./ (подпись и ф.и.о. рук-ля) Оценка по итогам защиты: _____ _____ / _____ / _____ / _____ / (подписи, дата, ф.и.о., должность, звание, уч. степень двух преподавателей, принявших защиту) М. РТУ МИРЭА. 2023 АННОТАЦИЯ Отчёт содержит 36 страниц, 22 иллюстрации, 8 листингов (большая часть из которых расположены в приложении А в конце отчёта), 15 литературных источников. Целью данной курсовой работы являлось создание фуллстек-приложения по размещению объявлений. Данный программный продукт был разработан с использованием технологий Java 17, Spring Boot, React, PostgreSQL и Docker Compose. В приложении предусмотрена система авторизации пользователей, сводка созданных объявлений и зарегистрированных пользователей, список объявлений с адресом, картинкой и описанием, форма создания новых объявлений, а также панель администратора с возможностью блокировки/активации пользователей и удаления объявлений. В качестве стека технологий для реализации серверной части использовался Spring Boot, включающий Spring Data JPA для управления объектно-реляционной моделью и взаимодействия с базой данных PostgreSQL, а также Spring Security для конфигурирования политик безопасности приложения. Веб-интерфейс был реализован с использованием ReactJS. Приложение было развернуто в Docker-контейнерах с помощью Docker Compose на выделенном персональном сервере и доступно для использования всеми желающими по соответствующему URL. Исходный код приложения доступен для просмотра и скачивания по соответствующей ссылке в сети Интернет. СОДЕРЖАНИЕ ПЕРЕЧЕНЬ СОКРАЩЕНИЙ 5 ВВЕДЕНИЕ 7 1 ОБЩИЕ СВЕДЕНИЯ 8 1.1 Обоснование выбора средств ведения разработки 8 2 ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ 9 3 ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ 10 3.1 Анализ предметной области 10 3.2 Проектирование архитектуры приложения 13 3.3 Проектирование базы данных 14 3.4 Разработка серверной части веб-приложения 15 3.5 Разработка клиентской части веб-приложения 21 3.6 Развертывание приложения с помощью Docker Compose 23 3.7 Обзор разработанного программного продукта

25 3.8 Подготовка к защите курсовой работы 28 ЗАКЛЮЧЕНИЕ 30 СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ 31 ПРИЛОЖЕНИЕ А 33 ПЕРЕЧЕНЬ СОКРАЩЕНИЙ БД-База данныхСУБД-Система управления базами данныхAPI-Application Programming Interface (программный интерфейс)CRUDCreate, Read, Update, Delete (перечень базовых операций - создания, чтения, обновления, удаления)CSS-Cascading Style Sheets (формальный язык описания внешнего вида документа, написанного с использованием языка разметки)DDL-Data Definition Language (группа операторов определения данных в SQL)DML-Data Manipulation Language (группа операторов управления данными в SQL)DTO-Data Transfer Object (объект, представляющий программную сущность и хранящий соответствующие данные из полей БД)HTML-HyperText Markup Language (стандартизированный язык разметки документов для просмотра веб-страниц в браузере)IDE-Integrated Development Environment (интегрированная среда разрабоотки - комплекс программных средств, используемый программистами для разработки программного обеспечения)JDK-Java Development Kit (бесплатно распространяемый компанией Oracle Corporation комплект разработчика приложений на языке Java)JPA-Java Persistence API (спецификация API Java EE, предоставляет возможность сохранять в удобном виде Java-объекты в базе данных)JSON-JavaScript Object Notation (текстовый формат обмена данными, основанный на JavaScript)MVP-Minimum Viable Product (минимально жизнеспособный продукт - продукт, обладающий минимальными, но достаточными для удовлетворения первых потребителей функциями)REST-Representational State Transfer (архитектурный стиль взаимодействия компонентов распределённого приложения в сети)SSH-Secure Shell (сетевой протокол прикладного уровня, позволяющий производить удалённое управление операционной системой)UI-User Interface (пользовательский интерфейс)URL-Uniform Resource Locator (унифицированный указатель ресурса) ВВЕДЕНИЕ На данный момент информационные технологии проникли во все сферы государства, общества и бизнеса. Трудно представить организацию любого процесса без задействия современных технологий, поскольку они значительно его упрощают, систематизируют и оптимизируют. В частности, каждое уважающее себя учебное учреждение имеет свой интернет-портал, а любой сколько-нибудь значительный бизнес обязан иметь собственный сайт или приложение для ведения предпринимательской деятельности через Интернет. Более того, некоторые бизнес-услуги могли и вовсе быть немыслимы до широкого распространения веб-приложений. Целью данной курсовой работы является разработка клиент-серверного фуллстек-приложения для размещения объявлений с использованием Spring Boot и React [1]. Для систематизации работы и упрощения восприятия отчета процесс разработки был поделен на семь основных частей: 1. Обоснование выбора средств ведения разработки. 2. Анализ предметной области. 3. Проектирование базы данных. 4. Разработка серверной части веб-приложения. 5. Разработка клиенткой части веб-приложения. 6. Развертывание приложения на сервере. 7. Обзор разработанного программного продукта. Приложение, созданное в результате выполнения курсовой работы должно представлять MVP (Minimum Viable Product) сервиса объявлений и иметь базово необходимый функционал для размещения и просмотра объявлений, иметь графический веб-интерфейс пользователя, производить аутентификацию и авторизацию пользователей, быть спроектировано в клиент-серверной архитектуре и развернуто на сервере, в готовом к работе состоянии для всех пользователяй сети Интернет. 1 ОБЩИЕ СВЕДЕНИЯ 1.1 Обоснование выбора средств ведения разработки Первым этапом любой разработки является выбор и установка соответсвующего программного обеспечения. Так в качестве среды разработки была выбрана IntelliJ IDEA Ultimate. Преимуществами IDEA является статус де факто стандарта при выборе IDE (Integrated Development Environment) в разработке на Java. Версия JDK (Java Development Kit) выбрана 17, так как она самая современная из долго поддерживаемых, включает в себя все самые важные и необходимые функции языка Java [2]. А где Java, там почти всегда и Spring [3] - стандарт при выборе фреймворка для разработки enterprise-приложений на Java. Для организации выполнения запросов к базе данных из среды Java был применён фреймворк Spring Data - интегрированный в экосистему Spring компонент. Развертываться приложение будет на Spring Boot, поскольку он обеспечивает простейшую "из коробки" интеграцию с остальным Java-приложением на Spring. В роли СУБД была выбрана PostgreSQL [4], так как она бесплатна, широко распространена, имеет продвинутый функционал, стабильна и масштабируема, в отличие от NoSQL решений. В качестве системы сборки используется Maven [5], как наиболее популярный и гибко кастомизируемый инструмент. При разработке UI (User Interface) был использован React [6], который позволяет разрабатывать сложные интерфейсы и обеспечивает удобную и быструю разработку фронтенда. Приложение развертывается в Docker контейнерах [7] на выделенном персональном сервере. Для управления контейнерами использовался Docker Compose. 2 ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ Разработанное приложение предназначено для создания и управления платформой онлайн объявлений. Оно предназначено для использования как полноценный продукт для тех, кто хочет размещать объявления в сети, а также для тех, кто хочет создавать собственные проекты, связанные с онлайн объявлениями. Приложение предоставляет функции для создания, редактирования, просмотра и удаления объявлений, а также для поиска объявлений по различным критериям. Целевой аудиторией данного программного продукта могут быть как частные лица, которые ищут товары и услуги, так и компании, которые хотят разместить свои объявления в интернете для привлечения новых клиентов. Также, это может быть интересно для начинающих разработчиков, которые хотят понимать, как работает полноценный веб-сервис и как можно разрабатывать приложения, используя технологии, применяемые в данном проекте. . 3 ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ 3.1 Анализ предметной области Сбор информации был начат с изучения уже существующих на рынке аналогов. Для этого в качестве лидера в нише был выбран и изучен популярный сайт "Avito", который предоставляет широкий перечень услуг, в том числе размещение объявлений. На рисунке 1 представлена главная страница сервиса, на которой можно наблюдать разбиение объявлений по различным категориям и общий поиск. Так же обратим внимание на яркую кнопку "Разместить объявление" в правом верхнем углу, а так же переход на страницу входа и регистрации рядом. Рисунок 1 - Главная страница сервиса "Avito" Спустившись ниже, можно наблюдать сами объявления, рекомендованные системой (рисунок 2). Объявление имеет главную фотографию, заголовок, цену, и дату публикации, а так же адрес. На одной странице собраны несколько объявлений, так, чтобы каждое не занимало много места и предоставляя только главную информацию. Рисунок 2 - Вид объявлений на "Avito" Вторым сервисом для анализа выступит "Юла" (рисунок 3). Здесь, так же как и у "Avito", на главной странице располагается поиск по объявлениям, выбор

категорий, авторизация, и яркая зеленая кнопка "Разместить объявление". Рисунок 3 - Главная страница сервиса "Юла" Сами объявления на "Юле" выглядят несколько иначе (рисунок 4) - на них представлено меньше информации, лишь цена, заголовок и фото. Компоновка объявлений же идентична. Рисунок 4 - Вид объявлений на "Юле" Проанализировав возможности данных сервисов можно сделать вывод, что для конкурентноспособности разрабатываемого приложения MVP должен обладать перечисленными выше качествами, а именно: 1. Регистрация, авторизация. 2. Просмотр списка объявлений, у каждого из которых есть фотография, и дополнительная информация, включая заголовок, адрес, цену, дату и описание. 3. Возможность размещения собственного объявления. 4. Поиск по объявлениям. 5. Панель администратора для удобства возможной модерации сайта. 3.2 Проектирование архитектуры приложения Данное приложение разрабатывается в клиент-серверной архитектуре, где клиентскую часть представляет фронтенд, написанный на JavaScript библиотеке React, а серверную часть - бэкенд, написанный на Java с использованием фреймворка Spring. Фронтенд взаимодействует с сервером посредством RESTful API, предоставляемого бэкендом [8]. В рамках данной архитектуры, все запросы клиента обрабатываются на сервере, где они принимаются соответствующими методами контроллеров. Контроллеры, в свою очередь, вызывают необходимые сервисы, которые взаимодействуют с базой данных и возвращают результаты запросов в виде JSON-объектов. Таким образом, в данном приложении фронтенд и бэкенд представляют отдельные компоненты, которые взаимодействуют друг с другом посредством RESTful API по сети. Схематичная диаграмма спроектированной архитектуры сервиса представлена на рисунке 5. Рисунок 5 - Архитектура сервиса размещения объявлений База данных так же запущена независимо от бэкенда, что обеспечивает общую низкую связанность этих трех компонентов и позволяет в будущем разнести их на три физически отдельных сервера, что гарантирует хороший потенциал для масштабирования такой распределенной системы. Однако, на данный момент все три компонента будут развернуты на одном хосте. 3.3 Проектирование базы данных После проведения анализа предметной области и выделения функциональных требований к проекту, следующим этапом необходимо провести описание главных сущностей в будущей системе. Так, на рисунке 6 представлена разработанная схема базы данных, включающая в себя такие сущности как: пользователь, объявление, адрес и изображение, а так же связи между ними. Рисунок 6 - Схема спроектированной базы данных База данных была создана в PostgreSQL с помощью инициализирующего скрипта, запускаемого Spring при старте приложения и подтверждения соединения с базой данных. Данный скрипт создает структуру таблиц с помощью DDL-команд [9]. Сама база данных запущена внутри Docker контейнера для удобства последующего развертывания внутри единого контейнеризированного окружения разрабатываемого приложения [10]. На листинге 1 можно ознакомиться с содержимым этого скрипта. 3.4 Разработка серверной части веб-приложения 3.4.1 Разработка слоя доменных сущностей На данном этапе для сущностей в SQL-таблицах были разработаны отображение в виде Java объектов, а так же и DTO (Data Transfer Object), которые будут передаваться на клиент (рисунок 7). Spring Data JPA в свою очередь берёт на себя управление запросами к БД из Java-окружения и обработку ответов от PostgreSQL. Рисунок 7 - Диаграмма классов слоя домена Все операции над сущностями на стороне приложения происходят с помощью посредника - интерфейса CrudRepository. От него наследуются репозитории для работы с сущностями текущего приложения (рисунок 8), затем каждому из таких интерфейсов сопоставляется имплементация, которая является бином в контексте Spring, и которая сама выполняет отправку необходимых SQL-запросов. Разработчик со своей стороны лишь описывает контракт тех или иных запросов. Репозитории в данном приложении за неимением сложной логики используются для выполнения базовых CRUD операций, поэтому предопределенная Spring имплементация полностью удовлетворяет требованиям проекта. Рисунок 8 - Диаграмма классов репозиториев 3.4.2 Разработка слоя сервисов Центральным местом всего приложения, реализующим основную бизнес-логику, является слой сервисов. Здесь самым сложным функционалом выступило сохранение изображений на локальной машине. Для этого, как можно наблюдать на листинге 2, в конфигурационном файле задается локальный путь директории для хранения изображений, если такового нет - то он создается, и уже в нем для каждого объявления создается индивидуальная папка для фотографий, которые относятся конкретно к данному объявлению. Полный путь до изображения при этом сохраняется в поле объекта класса Image и записывается в базу данных. Аналогично на листинге 3 представлен процесс считывания изображения из файловой системы для отправки на клиент по запросу. Во время всех этих манипуляций изображение проходит нетривиальный процессы сериализации и десериализации, которые заранее реализованы средствами используемых для этих целей библиотек. С остальными сервисами приложения, а так же их методами можно ознакомиться на диаграмме классов, представленной на рисунке 9. Рисунок 9 - Диаграмма классов слоя сервисов В соответствии с одним из требований, стоящих перед данной работой, на слое сервисов также реализован компонент, выполняющий инициализацию базы данных тестовыми данными. Всего при старте приложения, в случае если БД пуста, в систему добавляются два пользователя с ролями USER и ADMIN, а так же два полноценных объявления. С отрывком программы, отвечающим за инициализацию базы данных можно ознакомиться на листинге 4. Листинг 4 - Инициализация базы тестовыми данными @Override public void run(String... args) { if (userService.getByEmailOptional(ADMIN_EMAIL).isPresent() && userService.getByEmailOptional(USER_EMAIL).isPresent()) { return; } getUsers().forEach(userService::create); log.info("Database init: users created"); getAdverts().forEach(advertService::create); log.info("Database init: adverts created"); loadImages(); log.info("Database init: images created"); } Поскольку были затронуты роли USER и ADMIN, то так же стоит упомянуть о системе авторизации пользователей в системе. Для этих целей был применён компонент Spring Security. Механизм авторизации реализован самый элементарный - Basic Auth [11], при котором на сервер приходят логин и пароль, и в случае их корректности, возвращается статус 200 OK. В дальнейшем, клиент должен сохранить параметры входа в кэше, и добавлять их в заголовок каждого запроса [12]. Когда сервер получает любой такой запрос, то он проходит через фильтр безопасности, в которой происходит авторизация пользователя по переданным в заголовке запроса реквизитам. Когда они считываются - то происходит аутентификация пользователя, а затем сопоставление его роли с ограничениями ресурса, к которому он запрашивает доступ. На листинге 5 представлен отрывок из класса конфигурации безопасности приложения. 3.4.3 Разработка слоя контроллеров Контроллеры - компоненты системы, помеченные аннотацией

@RestController и выполняющие связывающую функцию между эндпоинтами API и методами сервисов. В данном приложение представлены пять контроллеров (рисунок 10). Рисунок 10 - Диаграмма классов слоя контроллеров Поскольку серверная часть представляет RESTful API и не отдает HTML-страницы непосредственно (что в свою очередь и позволяет построить многоуровневую клиент-серверную систему и уменьшить зацепление между компонентами), то зона ответственности бэкенда заканчивается на предоставлении ответов на запросы в формате данных JSON, которые получает клиент [13]. Ответ в JSON формате формируется на основе DTO, которые с помощью соответствующих библиотек проходят процесс конвертации из/в Java-объектов (для этого в данном приложении и используются мапперы на диаграмме выше). Со списком разработанных DTO можно было ознакомиться ранее в пункте, посвященном доменным сущностям. Все эндпоинты проектировались в соответствие с архитектурным стилем REST. Список всех разработанных эндпоинтов приложения представлен на рисунке 11. Рисунок 11 - Все эндпоинты серверной части приложения Для удобства последующего развертывания бэкенд на Spring был контейнеризирован с помощью Docker, для этого в верхнеуровневую директорию добавлен Dockerfile с содержимым, которое представлено на листинге 6. В нем происходит копирование исходного кода внутрь файловой системы контейнера, затем внутри контейнера происходит сборка приложения в jar-архив, который впоследствии и запускается внутри контейнером с предоставлением внешнего порта 8080 из контейнера. Листинг 6 - Контейнеризация серверной части приложения FROM maven:3.8.3-openjdk-17 AS build COPY src /home/advertapp-api/src COPY pom.xml /home/advertapp-api WORKDIR /home/advertapp-api RUN mvn clean install -DskipTests=true FROM openjdk:17-alpine RUN apk --no-cache add curl EXPOSE 8080 COPY --from=build /home/advertapp-api/target/advertapp-api-1.0.0.jar /usr/local/lib/advertapp-api.jar ENTRYPOINT ["java", "-jar", "/usr/local/lib/advertapp-api.jar"] 3.5 Разработка клиентской части веб-приложения Фронтенд приложения был реализован на библиотеке React для JavaScript, и включает в себя около 15-ти компонентов, необходимых для отображения главной страницы, страниц регистрации