**МИНОБРНАУКИ РОССИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕНЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.Шухова»**

**(БГТУ им. В.Г.Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Расчётно-графическое задание

По дисциплине: Технологии Web-программирования

Тема: “Разработка клиент-серверного приложения.”

Выполнил: Руденький А.О.

Проверил: Картамышев С.В.

Белгород 2020

# **Содержание**

[**Содержание** 2](#_Toc59237614)

[**Введение** 4](#_Toc59237615)

[**Глава 1. Серверная часть.** 5](#_Toc59237616)

[**1.1. Краткие теоретические сведения.** 5](#_Toc59237617)

[**1.1.1 Spring Framework** 5](#_Toc59237618)

[**1.1.2. Spring MVC** 5](#_Toc59237619)

[**1.1.3. Maven** 6](#_Toc59237620)

[**1.1.4. JPA** 6](#_Toc59237621)

[**1.1.5. Документирование RestAPI** 8](#_Toc59237622)

[**1.2. Разработка RestApi сервера с использованием Spring Framework.** 9](#_Toc59237623)

[**1.2.1. Проектирование БД.** 9](#_Toc59237624)

[**1.2.2. Создание моделей.** 10](#_Toc59237625)

[**1.2.3. Описание api функций.** 12](#_Toc59237626)

[**1.2.4. Реализация документирования RestApi посредством SpringDoc** 15](#_Toc59237627)

[**Глава 2. Разработка клиентской части** 17](#_Toc59237628)

[**2.1. Краткие теоретические сведения.** 17](#_Toc59237629)

[**2.1.1. ReactJS** 17](#_Toc59237630)

[**2.1.2. Axios** 17](#_Toc59237631)

[**2.1.3. Понятие маршрутизации** 17](#_Toc59237632)

[**2.1.4. Material UI** 18](#_Toc59237633)

[**2.1.5. Bootstrap** 18](#_Toc59237634)

[**2.2. Разработка клиентской части.** 19](#_Toc59237635)

[**2.2.1 Описание логики взаимодействия с RestApi сервера.** 20](#_Toc59237636)

[**2.2.2. Реализация маршрутизации** 23](#_Toc59237637)

[**Глава 3. Докеризация приложения** 24](#_Toc59237638)

[**3.1. Создание контейнеров.** 25](#_Toc59237639)

[**3.2. Группирование контейнеров и развёртывание веб-приложения.** 27](#_Toc59237640)

[**Скриншоты** 30](#_Toc59237641)

[**Заключение** 33](#_Toc59237642)

[**Список литературы** 34](#_Toc59237643)

[**ПРИЛОЖЕНИЯ** 35](#_Toc59237644)

[Приложение А – Код Java Spring сервера. 35](#_Toc59237645)

[Листинг A.1.1.1 – Листинг CloudserverApplication.java 35](#_Toc59237646)

[Листинг A.1.1.2 – Листинг CorsConfiguration.java 35](#_Toc59237647)

[Листинг A.1.1.3 – Листинг OpenAPIConfiguration.java. 35](#_Toc59237648)

[Листинг A.1.1.4 – Листинг application.properties 36](#_Toc59237649)

[Листинг A.1.1.5 – Листинг Client.java 36](#_Toc59237650)

[Листинг A.1.2.1 – Листинг ClientController.java 37](#_Toc59237651)

[Листинг A.1.2.2. – Листинг ClientJPA.java 39](#_Toc59237652)

[Листинг A.1.2.3 – Листинг ClientService.java 39](#_Toc59237653)

[Листинг A.1.3.1 – Листинг SharedFile.java 41](#_Toc59237654)

[Листинг A.1.3.2 – Листинг SharedFileController.java 43](#_Toc59237655)

[Листинг A.1.3.3 – Листинг SharedFileJPA.java 44](#_Toc59237656)

[Листинг A.1.4.1 – Листинг Promo.java 44](#_Toc59237657)

[Листинг A.1.4.2 – Листинг PromoJPA.java 46](#_Toc59237658)

[Листинг A.1.4.3 – Листинг PromoService.java 47](#_Toc59237659)

[Листинг A.1.5.1 – Листинг Session.java 48](#_Toc59237660)

[Листинг A.1.5.2 – Листинг SessionController.java 49](#_Toc59237661)

[Листинг A.1.5.3 – Листинг SessionJPA.java 50](#_Toc59237662)

[Листинг A.1.5.4 – Листинг SessionService.java 50](#_Toc59237663)

[Листинг A.1.6.1 – Листинг FileStorage.java 50](#_Toc59237664)

[Листинг A.1.6.2 – Листинг FileStorageController.java 52](#_Toc59237665)

[Листинг A.1.6.3 – Листинг FileStorageService.java 56](#_Toc59237666)

[Листинг A.1.7.1 – Листинг Minio.java 62](#_Toc59237667)

[Листинг A.1.8.1 – Листинг initController.java 62](#_Toc59237668)

[Приложение Б – Код ReactJS сервера. 64](#_Toc59237669)

[Листинг Б.1.1 – Листинг index.js 64](#_Toc59237670)

[Листинг Б.1.2 – Листинг App.js 65](#_Toc59237671)

[Листинг Б.1.3 – Листинг HomeFrame.js 73](#_Toc59237672)

[Листинг Б.1.4 – Листинг SharedFileFrame.js 79](#_Toc59237673)

[Листинг Б.1.5 – Листинг WelcomeFrame.js 81](#_Toc59237674)

[Листинг Б.2.1 – Листинг FilesManagerTab.js 86](#_Toc59237675)

[Листинг Б.2.2 – Листинг HeaderBar.js 89](#_Toc59237676)

[Листинг Б.2.3 – Листинг ItemFigure.js 91](#_Toc59237677)

[Листинг Б.2.4 – Листинг ModalDialog.js 93](#_Toc59237678)

[Листинг Б.2.5 – Листинг AuthPanel.js 94](#_Toc59237679)

# **Введение**

Сегодня все крупные и успешные предприятия перестают использовать нативные решения различных прикладных задач и активно переходят на использование веб-приложений, которые в последнее время всё чаще реализуются по принципу микросервисной архитектуры. Плюсы и минусы, есть как у нативного ПО, так и у современных веб-приложений, но в любом случае они как правило работают не автономно, а в связке с сервером, который производит большую часть обработки данных на себе и является наиболее важным звеном работы всей платформы.

Реализовать сервер можно на огромном количестве языков программирования, например, сервер можно написать на ЯП Golang, Python, Java, NodeJs, C++ и т.д., но чаще всего не просто с использованием элементарных объектов tcp взаимодействия, а с помощью специальных фрэймворков, которые позволяют избавится от boilerplate кода и бесконечных велосипедов с костылями.

В курсовой работе мы подробно рассмотрим тему разработки веб-приложения, на примере приложения “облачного файлового хранилища”, для этого мы рассмотрим необходимый для этого теоретический материал, создадим сервер на языке программирования java с использованием фрэймворка spring, клиентскую сторону реализуем с использованием ReactJS и в результате работы сделаем вывод о полученных результатах.

# **Глава 1. Серверная часть.**

## **1.1. Краткие теоретические сведения.**

### **1.1.1 Spring Framework**

Spring Framework обеспечивает комплексную модель разработки и конфигурации для современных бизнес-приложений на Java - на любых платформах. Ключевой элемент Spring - поддержка инфраструктуры на уровне приложения: основное внимание уделяется "водопроводу" бизнес-приложений, поэтому разработчики могут сосредоточиться на бизнес-логике без лишних настроек в зависимости от среды исполнения.

### **1.1.2. Spring MVC**

Один из самых главных разделов фреймворка Spring — Spring MVC. Фреймворк Spring Web model-view-controller (MVC) работает вокруг DispatcherServlet, который распределяет запросы по обработчикам. В нём настраивается мэппинг запросов, локали, временные зоны и многое другое. Обработчик по умолчанию строится на аннотациях @Controller и @RequestMapping, которые предоставляют широкий набор гибких методов для обработки запросов. После версии Spring 3.0. механизм @Controller так же позволяет создавать RESTful веб сайты и приложения, используя аннотацию @PathVariable и другие возможности.

В Spring Web MVC есть возможность использовать любой объект в качестве команды или объекта с обратной связью; нет необходимости реализовывать какой-либо специальный интерфейс фреймворка или базовый класс. Связывание данных в Spring является очень гибким: например, оно рассматривает несоответствие типов как ошибки валидации и поэтому это может быть обработано в приложении, а не в качестве системных ошибок.

Таким образом, вам не нужно дублировать свойства бизнес-объектов, в качестве простых нетипизированных строк для ваших объектов форм. Поэтому можно легко обрабатывать неправильные подтверждения или правильно конвертировать их в строки. Вместо этого, желательно связывать такие объекты напрямую с объектами бизнес логики.

### **1.1.3. Maven**

Maven – это фреймворк для автоматизации сборки проектов специфицированных на XML-языке. Сборка происходит по установленным зависимостям в файле pom.xml. Чтобы добавить или исключить какие-либо библиотеки или плагины в сборке проекта, достаточно просто указать зависимость в файле pom.xml и Maven автоматически его подключит и встроит в проект. Также в .xml файле указываются зависимости Spring Boot.

В Spring Boot уже включена Spring-платформа и сторонние библиотеки, для более простого запуска приложений. Пример файла pom.xml из проекта веб-сервера можно увидеть в приложении А.

### **1.1.4. JPA**

JPA – это технология, обеспечивающая объектно-реляционное отображение простых JAVA объектов и предоставляющая API для сохранения, получения и управления такими объектами.

Сам JPA не умеет ни сохранять, ни управлять объектами, JPA только определяет правила: как что-то будет действовать. JPA также определяет интерфейсы, которые должны будут быть реализованы провайдерами. Плюс к этому JPA определяет правила о том, как должны описываться метаданные отображения и о том, как должны работать провайдеры.

Дальше, каждый провайдер, реализуя JPA определяет получение, сохранение и управление объектами. У каждого провайдера реализация разная.

У JPA существуют разные реализации:

– Hibernate

– Oracle TopLink

– Apache OpenJPA

Наиболее высокую популярность среди этих реализаций завоевала Hibernate.

Эта технология является ORM-решением для языка Java. ORM – это отображение объектов какого-либо объектно-ориентированного языка в структуры реляционных баз данных.

Hibernate не только создает связи между Java классами и таблицами баз данных, но также предоставляет средства для автоматического построения запросов и извлечения данных и может значительно уменьшить время разработки.

Hibernate генерирует SQL вызовы и освобождает разработчика от ручной обработки результирующего набора данных и конвертации объектов, сохраняя приложение портируемым во все SQL базы данных.

В самом проекте объекты для JPA помечаются аннотациями. Класс, который будет являться сущностью в базе данных, отмечается аннотацией @Entity. Так как этот класс является сущностью для базы данных, в нем должен быть указан ключ, в нашем случае это ID.

Интерфейс репозитория – @Repository. Этот интерфейс наследуется от интерфейса JpaRepository<T,F>, где T – объект нашей сущности (класс с пометкой @Entity), а F – должен быть того же типа, что и ID из класса-сущности. При правильном наследовании Spring предоставит реализованные методы указанные в интерфейсе. Пример реализации сущности и репозитория представлен в приложении Б.

### **1.1.5. Документирование RestAPI**

Микросервисная архитектура подразумевает набор самостоятельных сервисов, которые общаются друг с другом, а для конечного пользователя выглядит как единая программа. Один из самых популярных протоколов для обмена сообщениями между микросервисами — это REST. Проблема в том, что REST не является само описательным протоколом. Это значит, что клиент должен знать конкретную комбинацию URL, HTTP метода и формата ответа. В некоторых случаях необходимо также знать также формат тела запроса. Обычно реализация REST интерфейса базируется на общих принципах и традициях. В любом случае, REST endpoints всегда должны быть описаны в одном конкретном документе, доступном для всех остальных разработчиков.

Документацию должно быть легко читать, писать, парсить, генерировать, исправлять, обновлять и прочее. Решением является использование Swagger или аналог springdoc, который доступен для java spring и отлично подойдёт для документирования REST API.

## **1.2. Разработка RestApi сервера с использованием Spring Framework.**

### **1.2.1. Проектирование БД.**

В качестве база данных мы выберем свободную объектно-реляционную систему управления базами данных PostgreSQL.

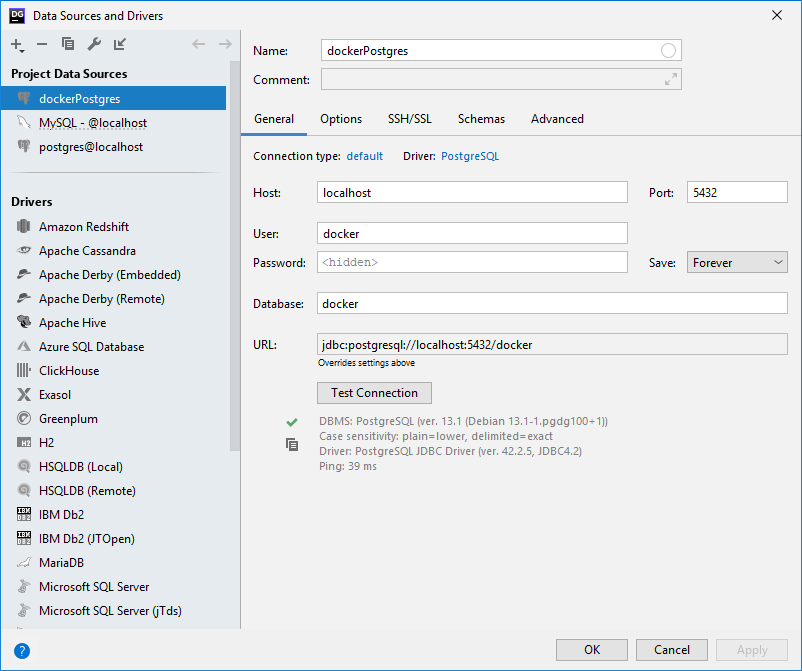


Рис 1.1. Демонстрация проверки связи с базой данных.

Для взаимодействия сервера и БД используем библиотеку Hibernate реализующую ORM, а также паттерн “репозиторий” и Data Transfer Object (DTO).

Для работы нашего облачного хранилища файлов выделим следующие наиболее значимые объекты:

* Клиент
* Файловое хранилище
* Промо-код
* Сессия
* Файл

Определим следующую структуру базы данных на следующей диаграмме:

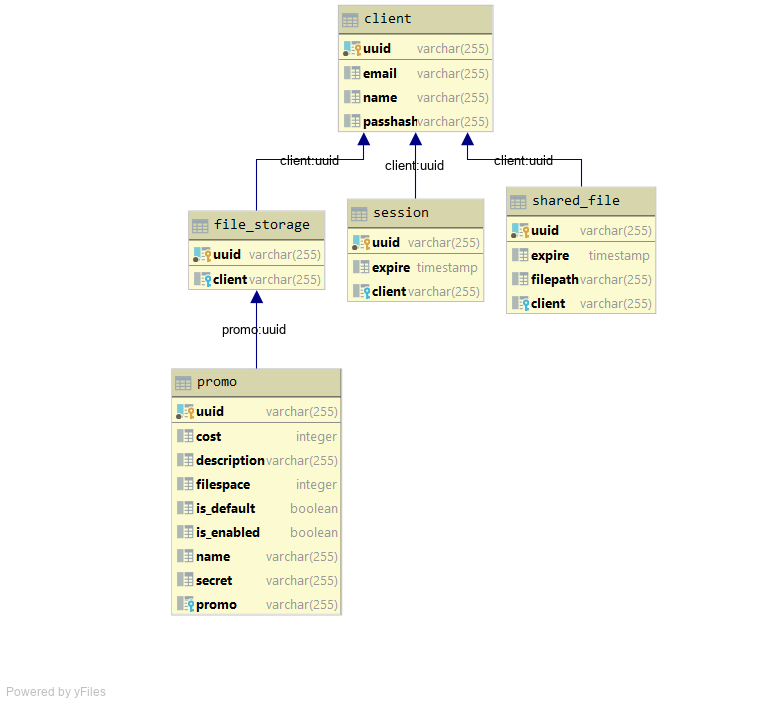


Рис 1.2. Диаграмма структуры базы данных.

### **1.2.2. Создание моделей.**

* Модель сущности “клиент”:

@Entity  
public interface ClientJPA extends JpaRepository<Client, Long>  
{  
 Boolean existsByName(String name);  
 Boolean existsByEmail(String email);  
 Boolean existsByPasshash(String passhash);  
 List<Client> findClientByNameEquals(String name);  
}

* Модель сущности “файловое хранилище”:

@Entity  
@RequiredArgsConstructor  
@NoArgsConstructor  
public class FileStorage implements Serializable {  
  
 @Id  
 @Column(name = "uuid")  
 @GeneratedValue(generator = "system-uuid")  
 @GenericGenerator(name = "system-uuid", strategy = "uuid")  
 @Getter  
 @Setter  
 private String id;  
  
 @OneToOne  
 @JoinColumn(name = "client")  
 @NonNull  
 private Client client;  
  
 @OneToMany  
 @JoinColumn(name = "promo")  
 @NonNull  
 private List<Promo> promo;  
  
}

* Модель сущности “Сессия”:

@Entity  
@RequiredArgsConstructor  
@NoArgsConstructor  
public class Session implements Serializable {  
  
 @Id  
 @Column(name = "uuid")  
 @GeneratedValue(generator = "system-uuid")  
 @GenericGenerator(name = "system-uuid", strategy = "uuid")  
 @Getter  
 @Setter  
 private String token;  
  
  
 @OneToOne  
 @JoinColumn(name = "Client")  
 @Getter  
 @Setter  
 @NonNull  
 private Client client;  
  
 @Column(name = "Expire")  
 @Getter  
 @Setter  
 @NonNull  
 private LocalDateTime expires;  
}

* Модель сущности “Промо-код”:

@Entity  
@RequiredArgsConstructor  
public class Promo implements Serializable  
{  
 @Id  
 @Column(name = "uuid")  
 @GeneratedValue(generator = "system-uuid")  
 @GenericGenerator(name = "system-uuid", strategy = "uuid")  
 @Getter  
 @Setter  
 private String id;  
  
 @Column(name = "name")  
 @Getter  
 @Setter  
 @NonNull  
 private String name;  
  
 @Column(name = "description")  
 @Getter  
 @Setter  
 @NonNull  
 private String description;  
  
 @Column(name = "secret")  
 @Getter  
 @Setter  
 @NonNull  
 private String secret;  
  
 @Column(name = "filespace")  
 @Getter  
 @Setter  
 @NonNull  
 private int filespace;  
  
 @Column(name = "cost")  
 @Getter  
 @Setter  
 @NonNull  
 private int cost;  
  
 @Column(name = "isEnabled")  
 @Getter  
 @Setter  
 @NonNull  
 private Boolean isEnabled;  
  
 @Column(name = "isDefault")  
 @Getter  
 @Setter  
 @NonNull  
 private Boolean isDefault;  
}

* Модель сущности “Файл”:

@Entity  
@RequiredArgsConstructor  
@NoArgsConstructor  
public class File implements Serializable {  
  
 @Id  
 @Column(name = "uuid")  
 @GeneratedValue(generator = "system-uuid")  
 @GenericGenerator(name = "system-uuid", strategy = "uuid")  
 @Getter  
 @Setter  
 private String token;  
  
  
 @OneToOne  
 @JoinColumn(name = "Client")  
 @NonNull  
 private Client client;  
  
 @Column(name = "Expire")  
 @Getter  
 @Setter  
 @NonNull  
 private LocalDateTime expires;  
  
}

### **1.2.3. Описание api функций.**

Выделим следующие api функции которые будут предоставлять java spring сервер:

* **Регистрация пользователя в системе.**

Сигнатура: /api/v1/client/register

Входные данные:

**public class** ReqRegisterDto {  
 **public** String **login**;  
 **public** String **email**;  
 **public** String **password**;  
 **public** String **promo**;  
}

Выходные данные:

Json: статус запроса и данные пользователя (token, name)

Контроллер (Заголовок):

@RequestMapping(value = **"/api/v1/client/register"**)  
**public** @ResponseBody String userReg(@RequestBody ReqRegisterDto data);

* **Авторизация пользователя в системе.**

Сигнатура: /api/v1/client/login

Входные данные:

**public class** ReqLoginDto {  
 **public** String **login**;  
 **public** String **password**;  
 **public** Boolean **remember**;  
}

Выходные данные:

Json: статус запроса и данные пользователя (token, name)

Контроллер (Заголовок):

@RequestMapping(value = **"/api/v1/client/login"**)  
**public** @ResponseBody String userLogin(@RequestBody ReqLoginDto data);

* **Получение содержимого облачной папки пользователя по относительному пути /\*\*.**

Сигнатура: /api/v1/mycloud/\*\*

Входные данные:

**public class** ReqProvideListInfoDto {  
 String **token**;  
}

Выходные данные:

**public class** RespProvideListInfoDto {  
 String **filename**;  
 String **filepath**;  
 Boolean **isDir**;  
}

Контроллер (Заголовок):

@RequestMapping(value=**"/api/v1/mycloud/\*\*"**, method=RequestMethod.***POST***)  
**public** @ResponseBody String provideListInfo(@RequestBody ReqProvideListInfoDto data, HttpServletRequest request);

* **Загрузка файла в облачную папку по относительному пути /\*\***

Сигнатура: /api/v1/upload/mycloud/\*\*

Входные данные:

**public class** ReqHandleFileUploadDto {  
 @Getter  
 @Setter  
 String **token**;  
 @Getter  
 @Setter  
 MultipartFile **file**;  
}

Контроллер (Заголовок):

@RequestMapping(value=**"/api/v1/upload/mycloud/\*\*"**, method=RequestMethod.***POST***)  
**public** @ResponseBody String handleFileUpload (@RequestParam(**"token"**) String token, @RequestParam(**"file"**) MultipartFile file, HttpServletRequest request);

* **Скачивание файла c облачной папки по относительному пути /\*\*.**

Сигнатура: /api/v1/download/mycloud/\*\*

Входные данные:

**public class** ReqGetFileDto {  
 @Getter  
 @Setter  
 String **token**;  
}

Контроллер (Заголовок):

@RequestMapping(value = **"/api/v1/download/mycloud/\*\*"**, method = RequestMethod.***POST***)  
**public void** getFile(  
 @RequestBody ReqGetFileDto data,  
 HttpServletRequest request,  
 HttpServletResponse response);

* **Создание подпапки в облачной папке по относительному пути /\*\*.**

Сигнатура: /api/v1/mkdir/mycloud/\*\*

Входные данные:

**public class** ReqMkDirectoryDto {  
 @Getter  
 @Setter  
 String **token**;  
}

Контроллер (Заголовок):

@RequestMapping(value = **"/api/v1/mkdir/mycloud/\*\*"**, method = RequestMethod.***POST***)  
**public** @ResponseBody String mkDirectory(  
 @RequestBody ReqGetFileDto data,  
 HttpServletRequest request,  
 HttpServletResponse response)**;**

* **Удаление файла/подпапки в облачной папке по относительному пути /\*\*.**

Сигнатура: /api/v1/delete/mycloud/\*\*

Входные данные:

**public class** ReqHandleFileDeleteDto {  
 @Getter  
 @Setter  
 String **token**;  
}

Контроллер (Заголовок):

@RequestMapping(value=**"/api/v1/delete/mycloud/\*\*"**)  
**public** @ResponseBody String handleFileDelete(

@RequestBody ReqHandleFileDeleteDto data, HttpServletRequest request)**;**

* **Проверка актуальности токена.**

Сигнатура: /api/v1/session/checkToken

Входные данные:

@RequestParam(**"token"**) String token

Выходные данные:

Статус запроса и истинна в случае актуальности переданного токена.

Контроллер (Заголовок):

@RequestMapping(value = **"/api/v1/session/checkToken"**, method=RequestMethod.***POST***)  
**public** @ResponseBody String auth(@RequestParam(**"token"**) String token);

### **1.2.4. Реализация документирования RestApi посредством SpringDoc**

В данной реализации сервера, для документации использовали инструмент автодокументации restapi функций SpringDoc, которые мы в процессе описания контроллеров обозначали специальными аннотациями SpringDoc. Следует отметить, что SpringDoc это инструмент, который упрощает создание и обслуживание документов API на основе спецификации OpenAPI 3.

После того как мы запустим сервер, наша документация будет доступна по следующему адресу:

http://localhost:8181/swagger-ui/index.html?configUrl=/v3/api-docs/swagger-config

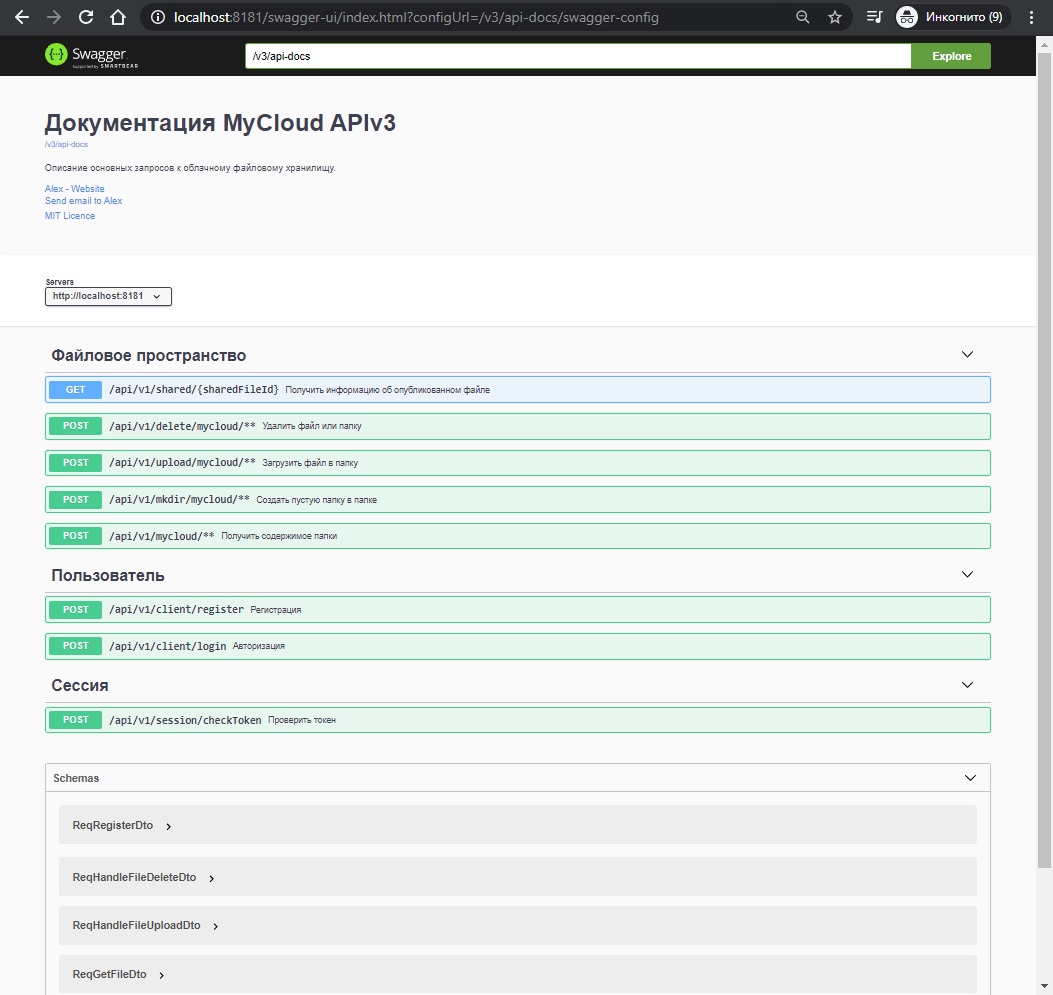
****

Рис 1.3. Документация RestAPI.

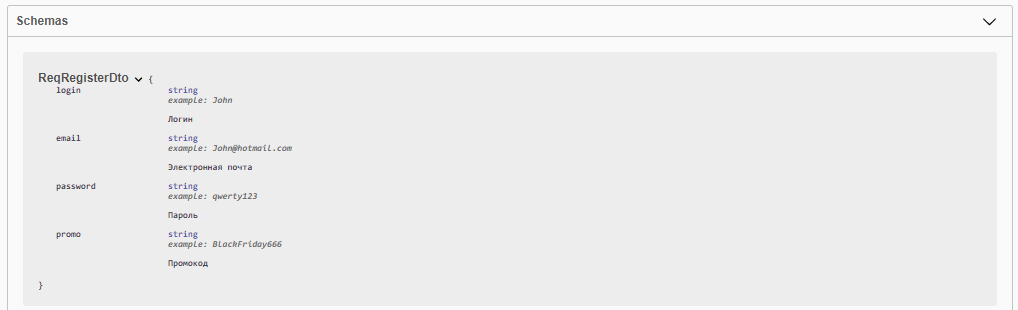
****

Рис 1.4. Пример описания dto структуры.

# **Глава 2. Разработка клиентской части**

## **2.1. Краткие теоретические сведения.**

### **2.1.1. ReactJS**

React (иногда React.js или ReactJS) — JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов.

React разрабатывается и поддерживается Facebook, Instagram и сообществом отдельных разработчиков и корпораций.

React может использоваться для разработки одностраничных и мобильных приложений. Его цель — предоставить высокую скорость, простоту и масштабируемость. В качестве библиотеки для разработки пользовательских интерфейсов React часто используется с другими библиотеками, такими как MobX, Redux и GraphQL.

React был создан Джорданом Валке, разработчиком программного обеспечения из Facebook. На него оказал влияние XHP — компонентный HTML-фреймворк для PHP. Впервые React использовался в новостной ленте Facebook в 2011 году и позже в ленте Instagram в 2012 году. Исходный код React открыт в мае 2013 года на конференции «JSConf US».

### **2.1.2. Axios**

Axios — это JavaScript-библиотека для выполнения либо HTTP-запросов в Node.js, либо XMLHttpRequests в браузере. Она поддерживает промисы — новинку ES6. Одна из особенностей, которая делает её лучше fetch() — автоматические преобразования JSON-данных.

Можно было также использовать классический fetch(), но при использовании fetch() для передачи данных в JSON, необходимо выполнить процесс в два этапа. Сначала сделать фактический запрос, а затем вызвать метод json() для полученных данных с сервера.

### **2.1.3. Понятие маршрутизации**

В любом реальном веб-приложении нужны маршруты, и приложение React не исключение. Пользователь должен видеть, где он находится в приложении в любой момент времени. А видит он свое текущее местоположение в адресной строке браузера. Следовательно, приложение должно уметь сопоставлять определённый URL с соответствующей ему страницей. То есть, если мы введём в адресную строку, например, https://example/appointmens, то приложение должно направить нас на страницу списка приёмов, но не на какую-либо другую.

Также должна работать история. То есть когда пользователь кликает на стрелку "Назад" в браузере, приложение должно направить нас на предыдущую страницу.

Сам по себе React не предоставляет такой возможности, это задача специальных библиотек. Как правило, используя API такой библиотеки мы подключаем компоненты страниц нашего положения, сопоставляя их с определёнными путями. После этого, переходя с одной страницы на другую мы будем видеть в адресной строке, как изменяется текущий путь.

### **2.1.4. Material UI**

Material UI — это набор компонентов React, который реализует Google Material Design (material-ui недавно выпустили v1 библиотеки). Эти компоненты работают изолированно, это означает, что они являются само-поддерживающими и вводят только те стили, которые они должны отображать.

Material UI — это часть Material Design. Material Design — это язык дизайна, впервые представленный Google в 2014 году. Это визуальный язык, который использует макеты на основе сетки, гибкую анимацию и переходы, дополнения и эффекты глубины, такие как освещение и тени. Цель Material Design сводится к трем вещам: Создание, Унификация и Настройка.

### **2.1.5. Bootstrap**

Bootstrap — это открытый и бесплатный HTML, CSS и JS фреймворк, который используется веб-разработчиками для быстрой вёрстки адаптивных дизайнов сайтов и веб-приложений.

Фреймворк Bootstrap используется по всему миру не только независимыми разработчиками, но иногда и целыми компаниями. На Bootstrap создано очень много различных сайтов.

Основная область его применения – это фронтенд разработка сайтов и интерфейсов админок. Среди аналогичных систем (Foundation, UIkit, Semantic UI, InK и др.) фреймворк Bootstrap является самым популярным.

Bootstrap популярен потому, что он позволяет верстать сайты в несколько раз быстрее, чем это можно выполнить на «чистом» CSS и JavaScript. А в нашем мире время – это очень ценный ресурс. Кроме этого, его популярность ещё обусловлена доступностью. Она заключается в том, что на нём даже начинающий разработчик может верстать достаточно качественные макеты, которые трудно было бы выполнить без глубоких знаний веб-технологий и достаточной практики.

Фреймворк Bootstrap представляет собой набор CSS и JavaScript файлов. Чтобы его использовать эти файлы необходимо просто подключить к странице. После этого вам станут доступны инструменты данного фреймворка: колоночная система (сетка Bootstrap), классы и компоненты.

## **2.2. Разработка клиентской части.**

Наш веб-клиент будет реализован на основе ReactJS, который является на сегодняшний самой популярной библиотека JavaScript для разработки пользовательского интерфейса (UI), по сравнению с VueJS и AngularJS. React используя новый метод рендеринга веб-сайтов позволяет добиться максимальной производительности в одностраничных веб-приложениях.

Графической основой приложения будет являться следующий набор инструментов, такой как, Bootstrap и MaterialUI, которые включают в себя HTML- и CSS-шаблоны оформления для веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения. С помощью этих инструментов мы спроектируем сетку нашей веб-страницы (с помощью Bootstrap), а также заполним её графическими элементарными компонентами (с помощью MaterialUI). Недостающие компоненты реализуем сами.

В качестве средства выполнения межсетевых запросов используем инструмент axios, который является JavaScript-библиотекой для выполнения HTTP запросов в браузере без необходимости перезагрузки страницы. Одной из особенностей библиотеки следует отметить возможность поддержки промисов — новинки ES6, а также есть ещё одна отличительная особенность, которая делает её лучше fetch() — автоматические преобразования JSON-данных.

### **2.2.1 Описание логики взаимодействия с RestApi сервера.**

Для реализации межсетевого взаимодействия нашего веб-клиента и сервера, мы создадим специальный hook, который будет описывать все возможные вызовы api функций сервера.

function useApiHook() {

let history = useHistory();

const axios = require('axios');

const rootUrl = "http://localhost:8181/api/v1"

const login = (login, password, remember) => {

axios({

method: 'post',

url: 'http://localhost:8181/api/v1/client/login',

data: {

login: login,

password: password,

remember: remember

},

})

.then(function (response) {

localStorage.setItem('token', response.data.payload.token);

localStorage.setItem('login', response.data.payload.client.name);

if (response.data.status === "ok") {

history.push("/mycloud");

}

})

.catch(function (error) {

console.log(error);

});

}

const register = (nick, email, password, promo) => {

axios({

method: 'post',

url: rootUrl + '/client/register',

data: {

login: nick,

password: password,

email: email,

promo: promo

}

})

.then(function (response) {

console.log(response);

if (response.data.status === "ok") {

login(nick, password, false);

}

})

.catch(function (error) {

console.log(error);

});

}

const logout = () => {

localStorage.clear();

checkToken();

}

const getLocalToken = () => {

return localStorage.getItem('token');

}

const getLocalData = () => {

return localStorage.getItem('login');

}

const listFiles = async (path) => {

return await new Promise((resolve, reject) => {

axios({

method: 'post',

url: rootUrl + path,

data: {

token: getLocalToken()

}

})

.then(function (response) {

console.log(response.data);

if (response.data.status === "ok") {

resolve(response.data.payload);

} else {

reject(response.data);

}

})

.catch(function (error) {

console.log(error);

});

})

}

const uploadFile = (file, path) => {

console.log("Закачиваю", path.slice(-1));

var formData = new FormData();

formData.append("file", file);

formData.append("token", getLocalToken());

return axios.post(rootUrl +"/upload" +path+(path.slice(-1)==="/"?"":"/"), formData, {

headers: {

'Content-Type': 'multipart/form-data'

}

})

}

const downloadFile = (file) => {

const FileDownload = require('js-file-download');

return axios({

url: rootUrl +"/download/mycloud/"+ file.filepath,

method: 'POST',

data: {

token: getLocalToken()

},

responseType: 'blob', // Important

}).then((response) => {

FileDownload(response.data, file.filename);

});

}

const deleteFile = (file) => {

return axios({

url: rootUrl +"/delete/mycloud/"+ file.filepath,

method: 'POST',

data: {

token: getLocalToken()

},

responseType: 'blob',

})

}

const createFolder = (path) => {

return axios({

url: rootUrl +"/mkdir"+ path+(path.slice(-1)==="/"?"":"/"),

method: 'POST',

data: {

token: getLocalToken()

},

})

}

const checkToken = () => {

axios({

method: 'post',

url: 'http://localhost:8181/api/v1/session/checkToken',

params: {

token: getLocalToken()!=null?getLocalToken():"",

},

})

.then(function (response) {

console.log(response);

if (response.data.payload === false) {

history.push("/welcome");

}

})

.catch(function (error) {

console.log(error);

});

}

return [login, register, logout, getLocalToken, listFiles, uploadFile, downloadFile, deleteFile, createFolder, checkToken, getLocalData];

}

Так же продемонстрируем работу взаимодействия клиентской части приложения с REST API сервера на примере авторизации пользователя в системе.

<TabPanel value={panelIndex} index={0} className={"pt-0"}>  
 <CardContent className={"p-0"}>  
 <div style={{marginLeft: "2em", marginRight: "2em"}}>  
  
 <TextField id="outlined-basic" label="Логин"  
 onChange={(e) => setAuthFields({...authFields, ...{"login": e.target.value}})}  
 style={{width: "100%", marginTop: "1em"}}/>  
 <TextField id="outlined-basic" label="Пароль"  
 onChange={(e) => setAuthFields({...authFields, ...{"password": e.target.value}})}  
 type={"password"} style={{width: "100%", marginTop: "1em"}}/>  
 <FormControlLabel  
 control={  
 <Checkbox  
 checked={authFields.remember}  
 onChange={(e) => setAuthFields({...authFields, ...{"remember": e.target.checked}})}  
 color="primary"  
 />  
 }  
 label="Запомнить"  
 />  
  
 </div>  
 </CardContent>  
 <CardActions className="float-left ml-4 mt-3 ">  
 <Button variant="contained" color="primary" size="medium"  
 onClick={() => login(authFields.login, authFields.password, authFields.remember)}>Войти</Button>  
 </CardActions>  
</TabPanel>

Для реализации обращения к самому сервера мы будем использовать axios.

### **2.2.2. Реализация маршрутизации**

На данный момент есть несколько популярных библиотек для маршрутизации: react-router, router5, aviator и пр. Мы будем использовать react-router. Хотя в вашем проекте может быть любая другая - все зависит от бизнес-требований.

Маршрутизацию реализуем в App.js следующим образом:

function App() {  
…  
 return (  
 <div>  
 <Router>  
 <Switch>  
 <Route path="/welcome">  
 <WelcomeFrame useApiHook={useApiHook}/>  
 </Route>  
 <Route strict={false} exact={false} path="/mycloud">  
 <HomeFrame useApiHook={useApiHook}/>  
 </Route>  
 <Route path="/shared/:sharedFileId">  
 <SharedFileFrame useApiHook={useApiHook}/>  
 </Route>  
 <Route path="/">  
 <WelcomeFrame useApiHook={useApiHook}/>  
 </Route>  
 </Switch>  
 </Router>  
 </div>  
 );  
}  
  
export default App;

# **Глава 3. Докеризация приложения**

Docker — программа, позволяющая операционной системе запускать процессы в изолированном окружении на базе специально созданных образов. Несмотря на то, что технологии, лежащие в основе Докера появились до него, именно Докер произвел революцию в том, как сегодня создается инфраструктура проектов, собираются и запускаются сервисы.

Docker позволяет «упаковать» приложение со всем его окружением и зависимостями в контейнер, который может быть перенесён на практически любую систему.

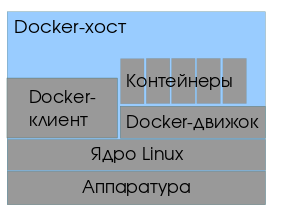


Рис 3.1. Docker на физическом Linux-сервере

Контейнеры чрезвычайно полезны с точки зрения безопасности, воспроизводимости и масштабируемости при разработке ПО и обработке данных. Их использование облегчает жизнь многим разработчикам. Докер позволяет развернуть микросервисную архитектуру запуском одной команды причем с почти 100% гарантией успеха.

Для разработки, упаковки и запуска приложений в контейнере необходимо специальное ПО.

Жизненный цикл (конечный автомат) контейнера можно описать следующей схемой:

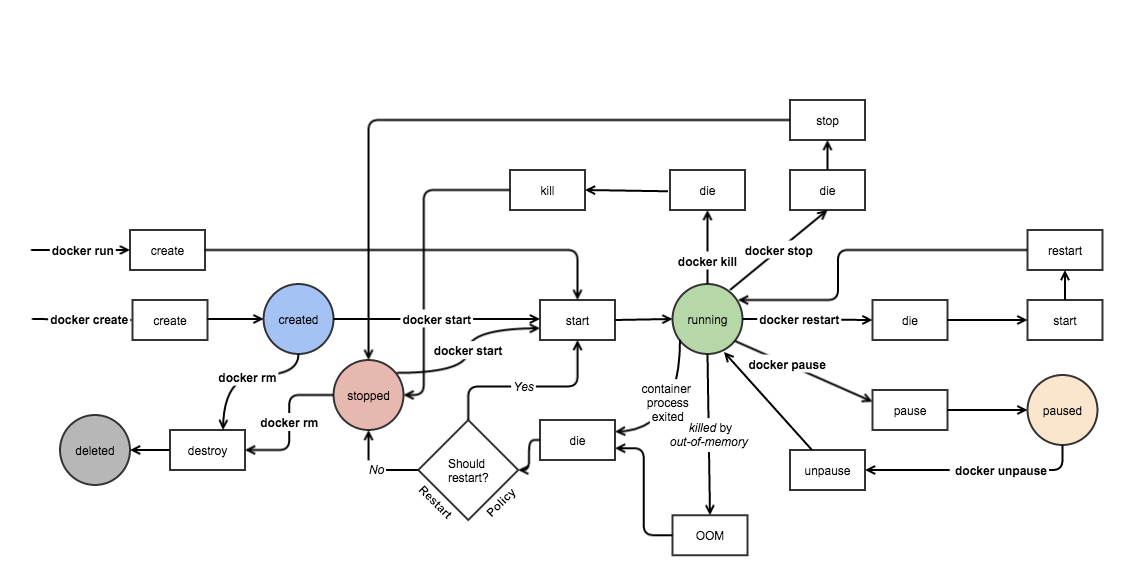


Рис 3.2. Жизненный цикл контейнера.

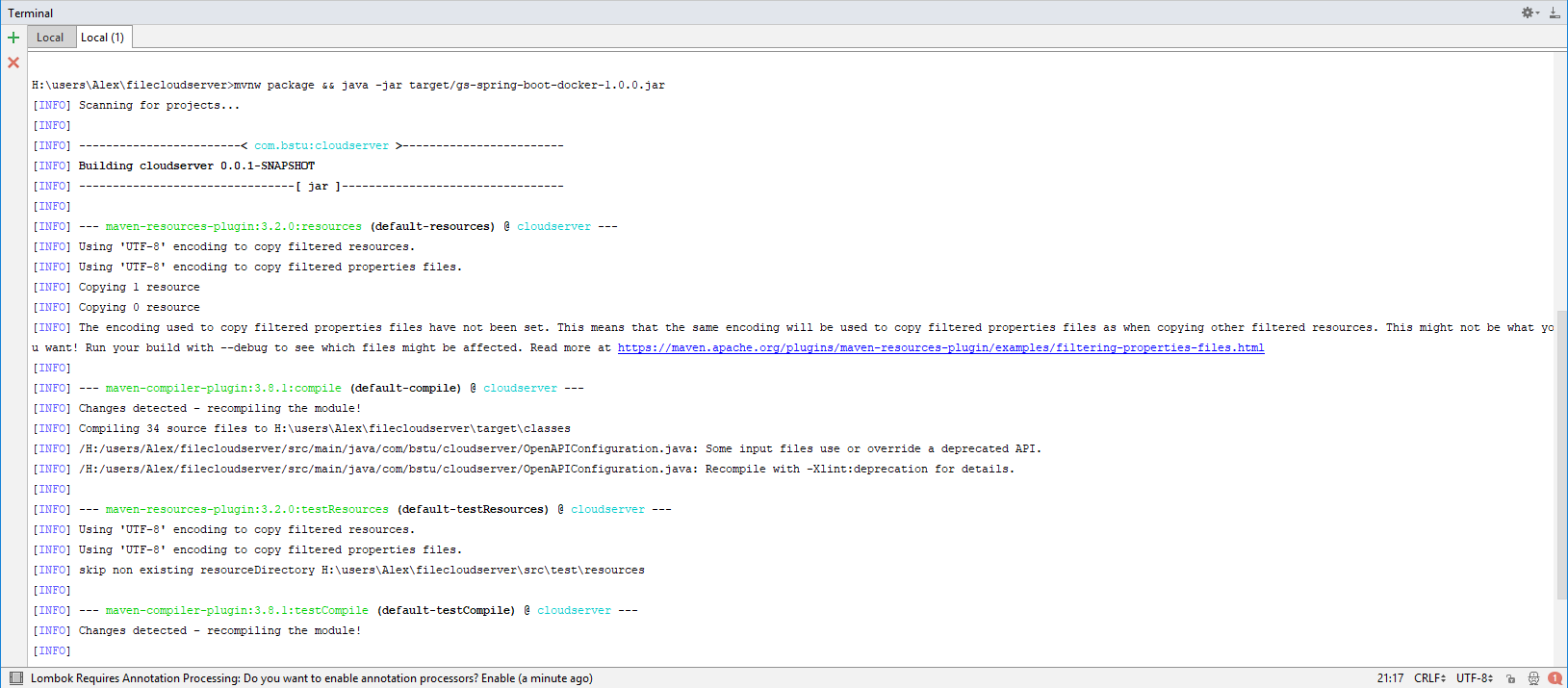
Где кругами изображены состояния, жирным выделены консольные команды, а квадратными показывается то, что в реальности выполняется.

Так же следует отметить, что запускать изолированный контейнер, который живет весь внутри себя — малополезно. Как правило, контейнеру нужно взаимодействовать с внешним миром, принимать входящие запросы на определенный порт, выполнять запросы на другие сервисы, читать общие файлы и писать в них. Все эти возможности дополнительно настраиваются при создании контейнера, если это необходимо.

## **3.1. Создание контейнеров.**

Для докеризации java spring сервера достаточно в консоли IDE выполнить следующую команду:

mvnw package && java -jar target/gs-spring-boot-docker-1.0.0.jar



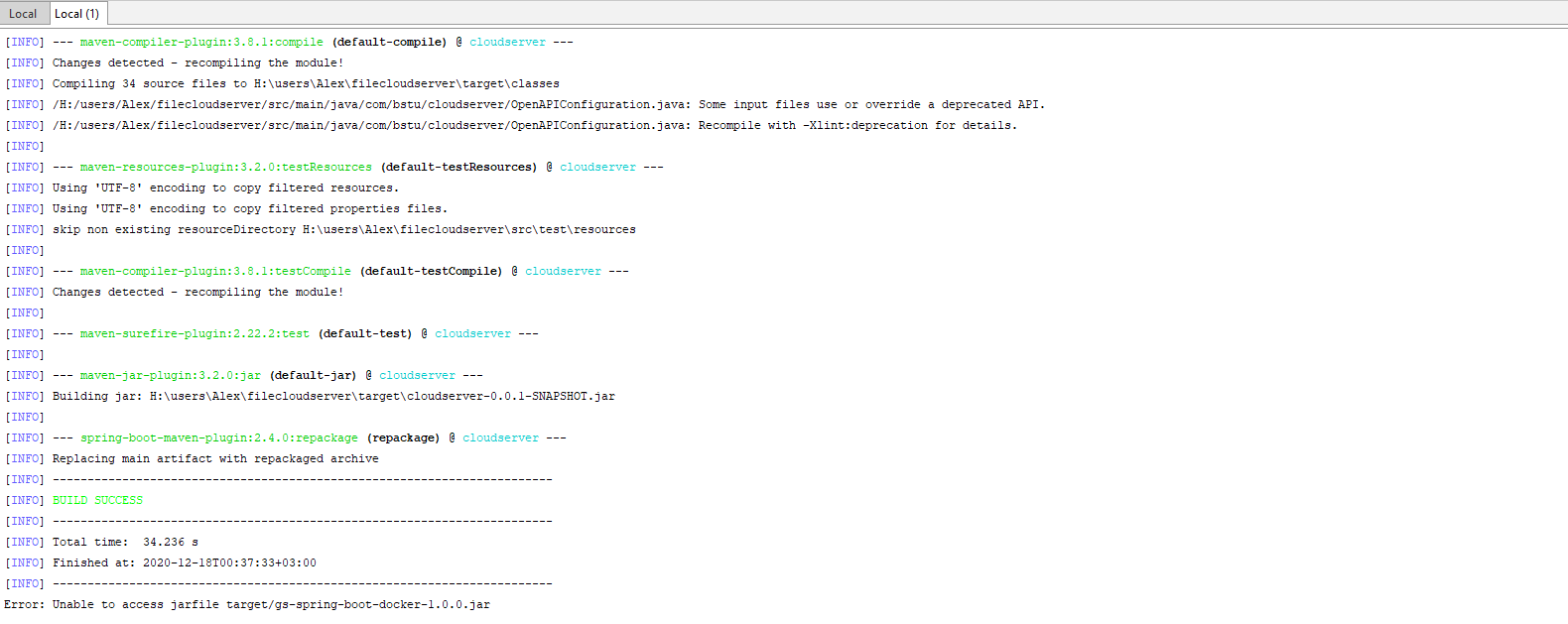


Рис 3.3. Процесс создания контейнера Java Spring сервера

Для докеризации ReactJS приложения, понадобится создать два файла и выполнить одну команду:

Содержимое файла “.dockerignore”:

node\_modules

build

.dockerignore

Dockerfile

Dockerfile.prod

Содержимое файла “Dockerfile”:

FROM node:12-alpine as builder

WORKDIR /app

COPY package.json /app/package.json

RUN npm install --only=prod

COPY . /app

RUN npm run build

Специальная команда сборки контейнера (Команда обязательно должна запускаться в папке с Dockerfile файлом!):

docker build -t sample:dev .

## **3.2. Группирование контейнеров и развёртывание веб-приложения.**

Все необходимые контейнеры мы создали, теперь необходимо собрать все контейнеры “в одну кучу” добавив недостающий postgresql сервер и minio сервер. Эту операцию позволит сделать Docker Compose.

Docker применяется для управления отдельными контейнерами (сервисами), из которых состоит приложение.

Docker Compose используется для одновременного управления несколькими контейнерами, входящими в состав приложения. Этот инструмент предлагает те же возможности, что и Docker, но позволяет работать с более сложными приложениями.

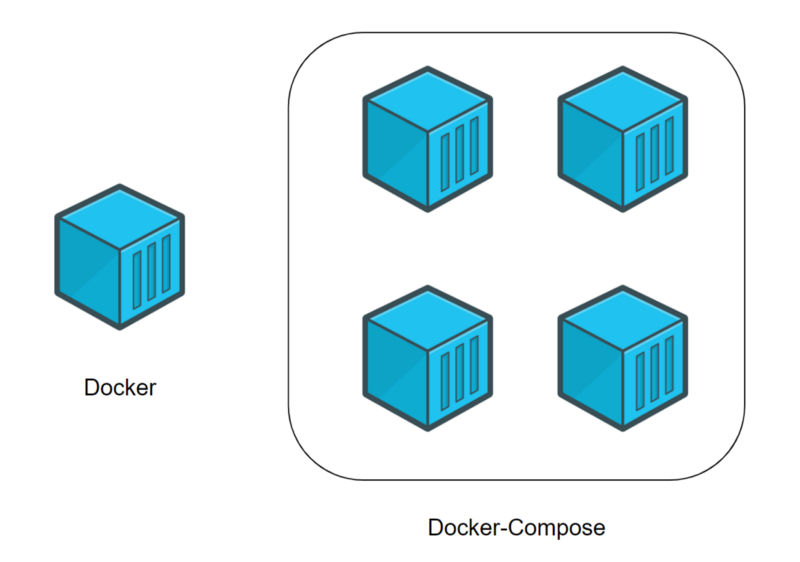


Рис. 3.4. Docker (отдельный контейнер) и Docker Compose (несколько контейнеров).

Для группировки и развёртывания приложения нам необходимо создать файл docker-compose.yml и выполнить специальную команду для запуска всех наших контейнеров.

Содержимое файла “docker-compose.yml”:

version: '3'

services:

db:

image: 'postgres'

ports:

- "${MY\_DOCKER\_IP:-127.0.0.1}:5432:5432"

environment:

POSTGRES\_USER: docker

POSTGRES\_PASSWORD: docker

POSTGRES\_DB: docker

frontserver:

image: 'docker.io/library/sample:dev'

ports:

- "${MY\_DOCKER\_IP:-127.0.0.1}:3000:3000"

tty: true

stdin\_open: true

backserver:

image: 'docker.io/library/cloudserver:0.0.1-SNAPSHOT'

ports:

- "${MY\_DOCKER\_IP:-127.0.0.1}:8181:8181"

depends\_on:

- db

tty: true

stdin\_open: true

environment:

-MINIO\_SERVER\_ACCESS\_KEY="AKIAIOSFODNN7EXAMPLE"

-MINIO\_SERVER\_SECRET\_KEY="wJalrXUtnFEMI/K7MDENG/bPxRfi CYEXAMPLEKEY"

filestorage:

image: 'minio/minio'

command: server /data

ports:

- "${MY\_DOCKER\_IP:-127.0.0.1}:9000:9000"

environment:

MINIO\_ACCESS\_KEY: "AKIAIOSFODNN7EXAMPLE"

MINIO\_SECRET\_KEY: "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"

И выполним команду:

docker-compose up

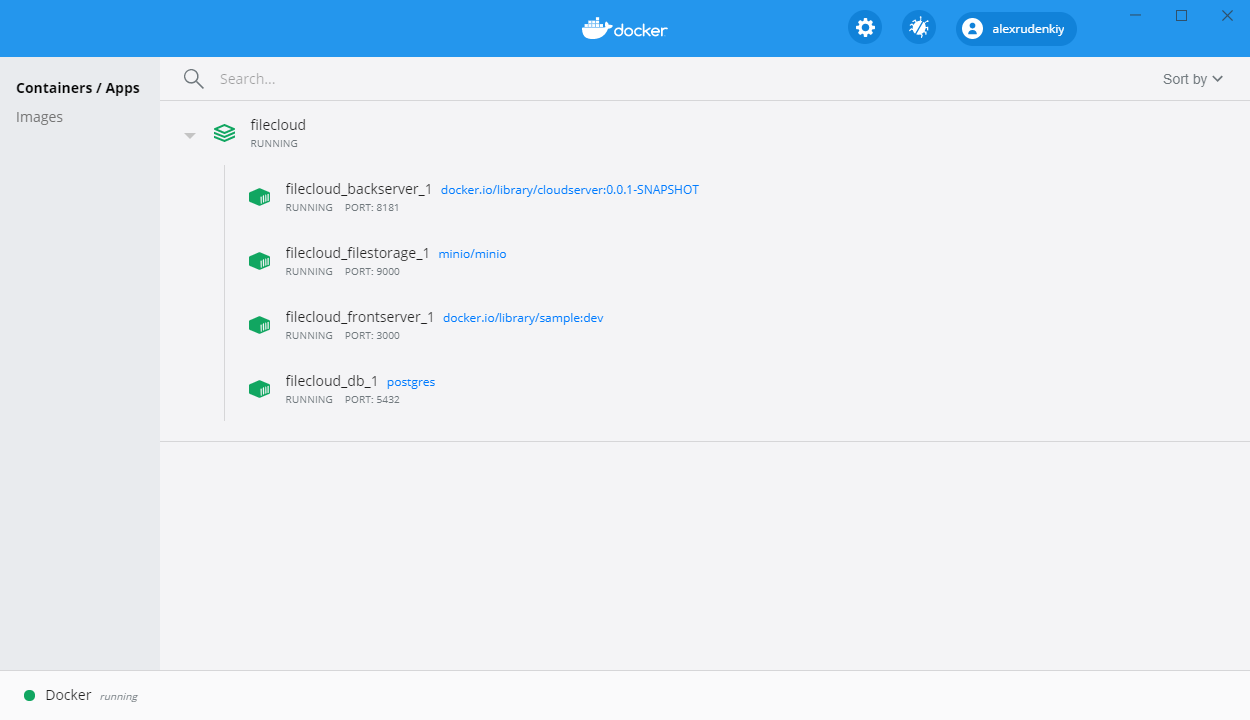


Рис. 3.5. Запущенная группа контейнеров.

После того как все сервера прогрузятся можно будет зайти на наш сайт по адресу http://localhost:3000/.

# **Скриншоты**

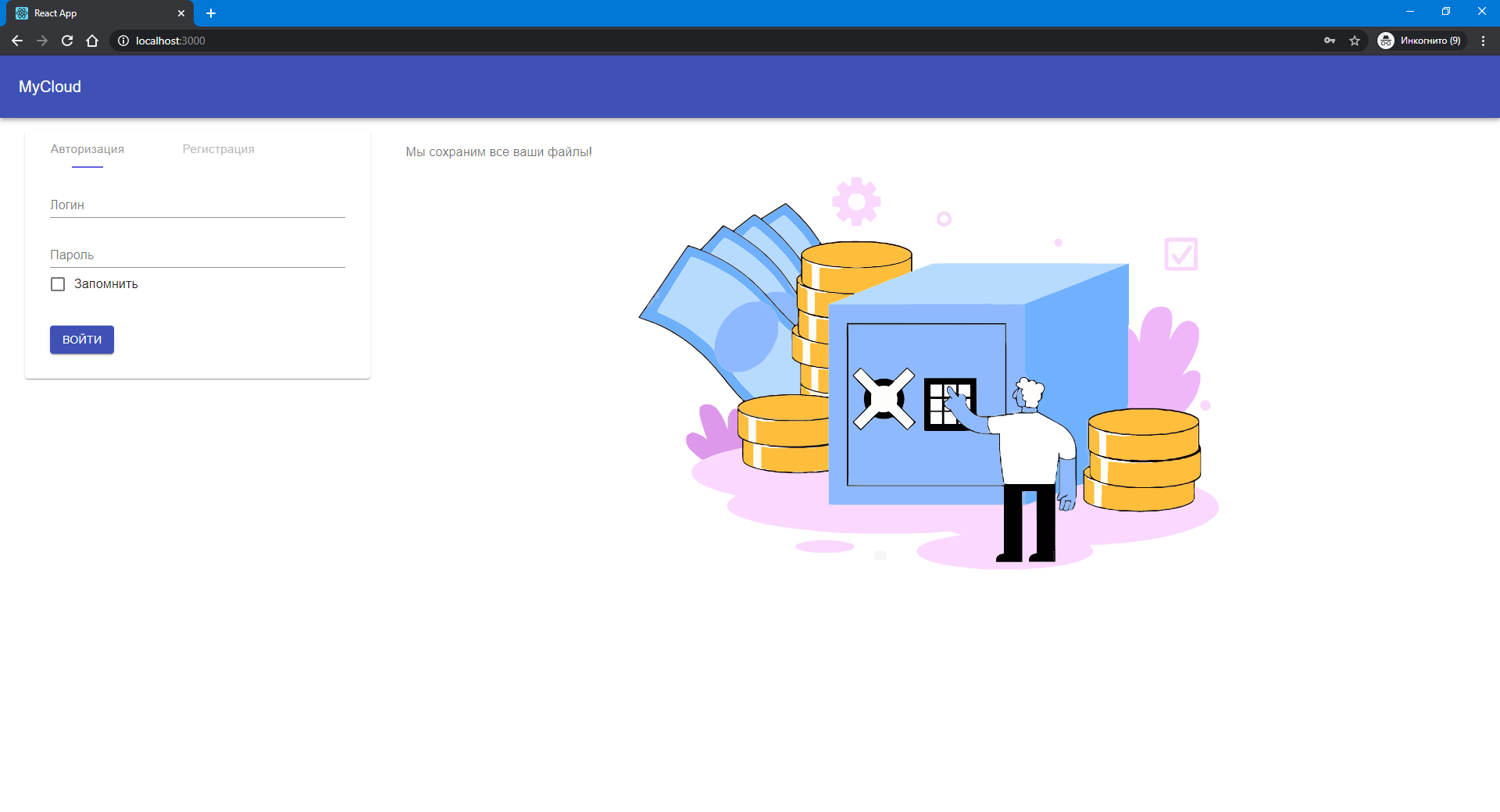


Рис 4.1. Скриншот сайта. Форма авторизации.

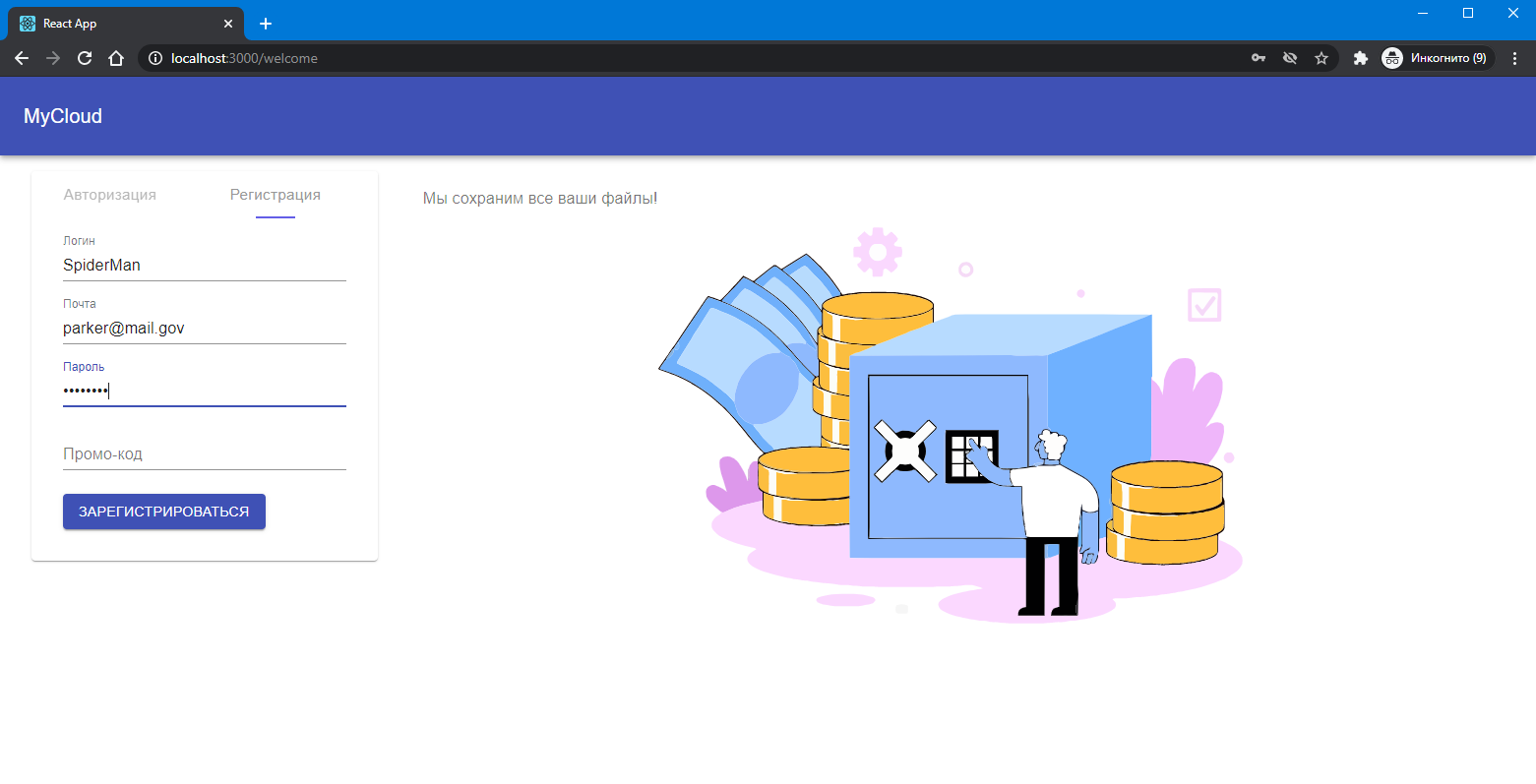


Рис 4.2. Скриншот сайта. Форма регистрации.

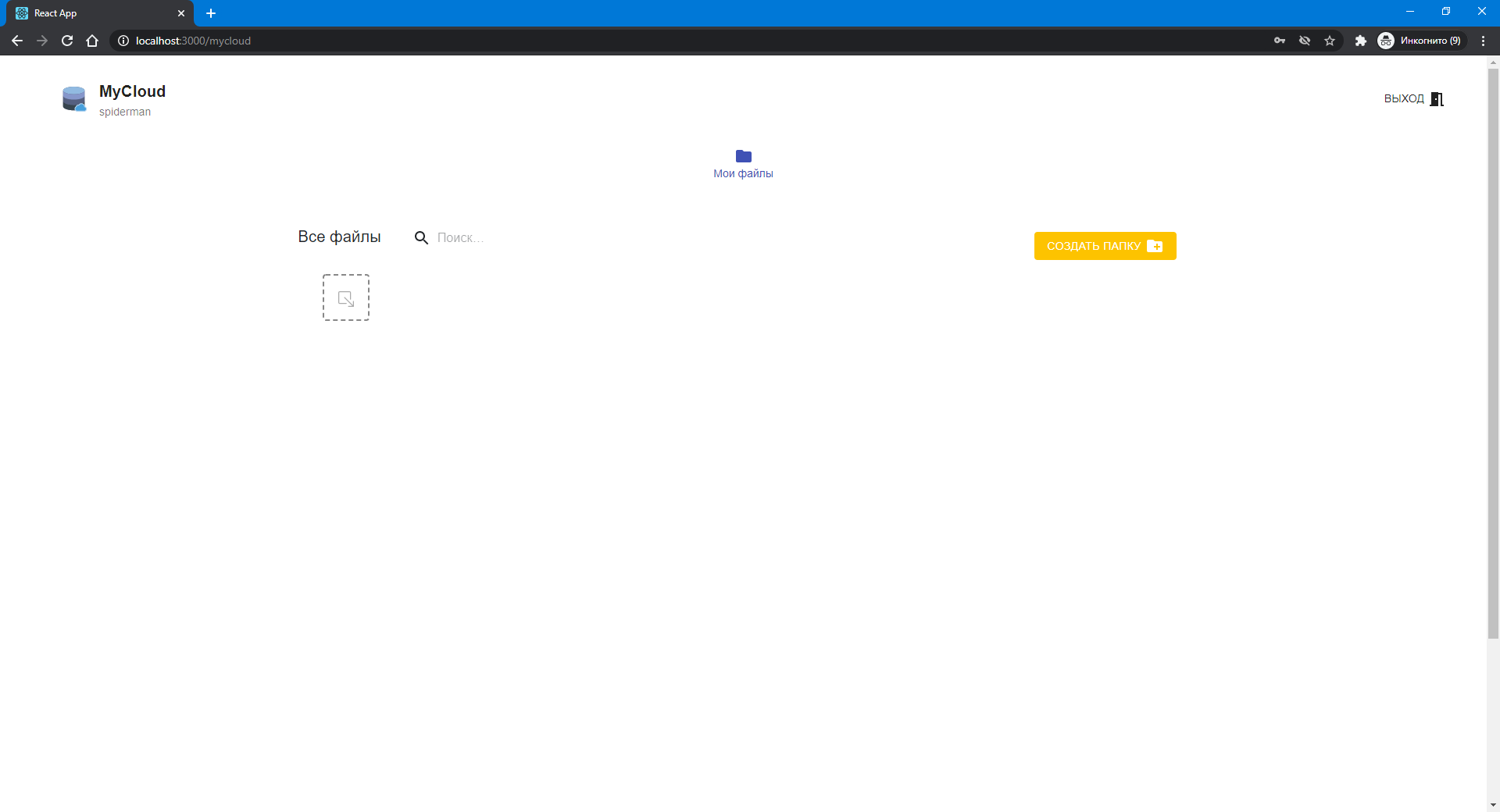


Рис 4.3. Скриншот сайта. Главная форма.

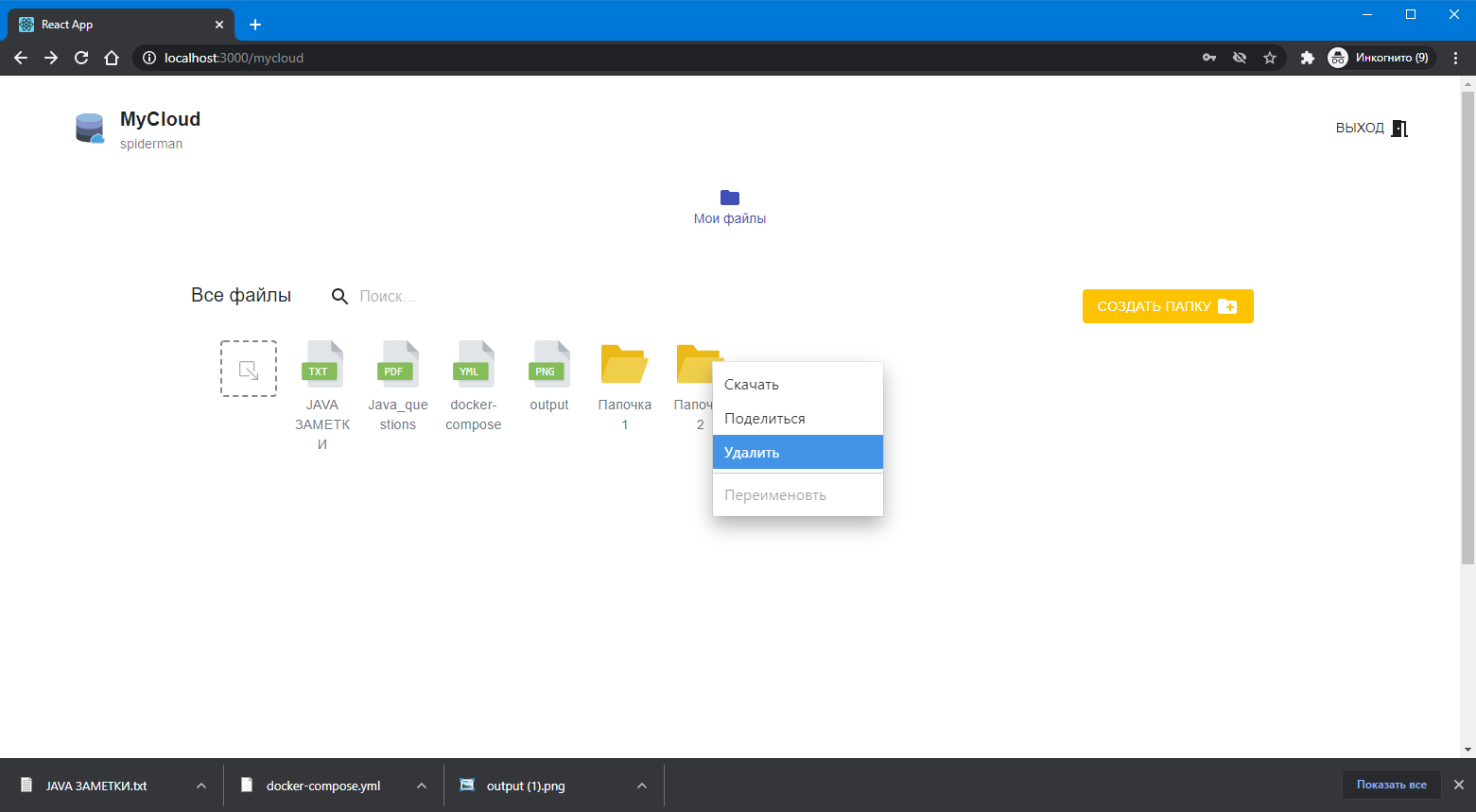


Рис 4.4. Скриншот сайта. Главная форма с загруженными файлами и открытым выпадающим меню.

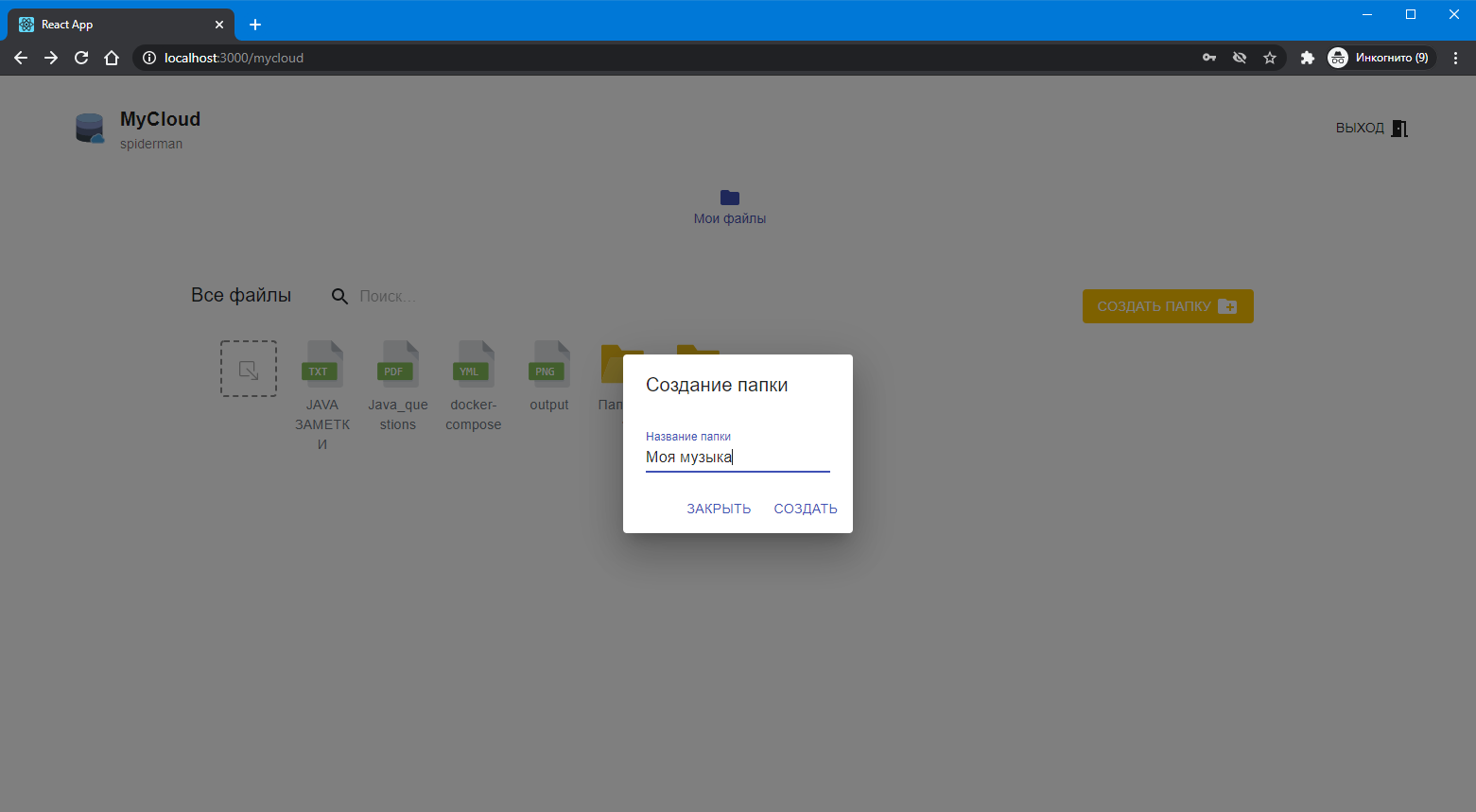


Рис 4.5. Скриншот сайта. Главная форма с модальным окном, отвечающим за создание папки.

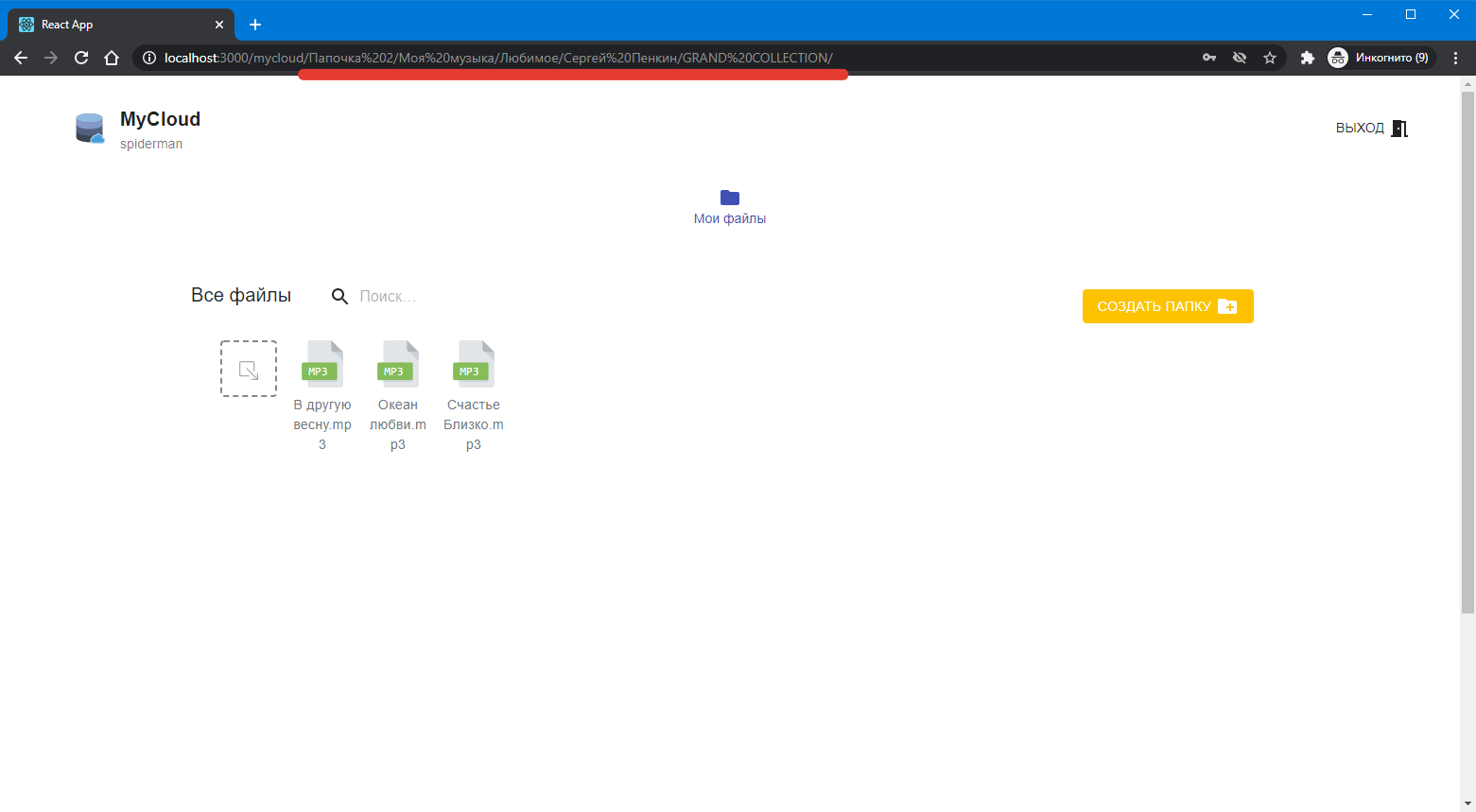


Рис 4.6. Скриншот сайта. Демонстрация маршрута, сформированного при переходе в подпапки.

# **Заключение**

В данной работе мы познакомились с клиент-серверной архитектурой, определили актуальность данной технологии, после чего рассмотрели кратко теоретический материал и разработали веб-приложение, которое было создано по этому принципу. В частности, мы разработали клиентскую часть на ReactJS и серверную на Java Spring, которые взаимодействуют между собой посредством архитектурного стиля взаимодействия компонентов распределённого приложения в сети RestAPI.

Процесс разработки мы начали с описания необходимых спецификаций моделей сущностей, обозначения наиболее необходимых обработчиков запросов. Практическая разработки сервера началась с установки базовых модулей и сопутствующей базовой конфигурацией сервера. Большая часть времени понадобилась на описание контроллеров и реализацией сервисов. Для взаимодействия нашего сервера с БД мы использовали объектно-реляционное отображение, которое реализовано в библиотеке Hibernate. Для большей возможности масштабируемости и независимости от серверной ОС, мы использовали микросервисную технологию, которую прекрасно реализует Docker и соответственно сервис базы данных и файловый сервис семейства Amazon S3 сделали в самостоятельных контейнерах. После завершения процесса разработки, мы приступили к этапу “тестирования”. Тестирование api функций мы производили посредством специального google chrome расширения “Servistate HTTP Editor”, который позволили наиболее быстро и удобно отладить все api функции, а затем выявить, и устранить множество недостатков, связанных с логическими ошибками в коде.

Процесс разработки фронтенд сервера мы начали с реализации вызовов описанных и реализованных на сервере в отдельно выделенном хуке. Затем отладив процесс минимального взаимодействия клиента и сервера, мы стали создавать разметку, параллельно «подгоняя» дизайна сайта под изначально созданный нами упрощённый макет. Оснащая наш макет интерактивными компонентами (кнопками, изображениями и т.д.) мы параллельно реализовывали привязку нашего интерфейса к api вызовам описанным и реализованным в нашем хуке. После того как мы реализовали наш функционал, мы начали следующий этап разработки, который является “тестированием”, после многочисленных тестов было выявлено ряд существенных недостатков, которые мы немедленно исправили и вернулись к этапу “тестирования”.

# **Список литературы**

1. Герберт, Шилдт Java 8. Руководство для начинающих / Шилдт Герберт. - М.: Диалектика / Вильямс, 2015. - 899 c.

2. Джошуа, Блох Java. Эффективное программирование / Блох Джошуа. - М.: ЛОРИ, 2014. - 292 c.

3. Официальная документация Spring. URL: https: // spring. io/docs (Дата обращения: 18.12.2020)

4. Spring IO Platform Reference Guide. URL: http://docs. spring. io/spring/docs/4.3.0. RC2/spring-framework-reference/htmlsingle/ (Дата обращения 14.12.2020)

5. Seth Ladd, Darren Davison, Expert Spring MVC and Web Flows [2006]

6. Кларенс Хо, Роб Харроп Spring 3 для профессионалов [2013]

7. Быстрый старт spring. URL: http://spring-projects.ru/projects/spring-framework/ (Дата обращения: 14.12.2020)

8. Давыдов, Станислав IntelliJ IDEA. Профессиональное программирование на Java / Станислав Давыдов , Алексей Ефимов. - М.: БХВ-Петербург, 2015. - 800 c.

9. Хортон А., Вайс Р. - Разработка веб-приложений в ReactJS - Издательство "ДМК Пресс" - 2016 - 254с. - ISBN: 978-5-94074-819-9 - Текст электронный // ЭБС ЛАНЬ - URL: https://e.lanbook.com/book/97339

10. Алекс, Бэнкс React и Redux. Функциональная веб-разработка. Руководство / Бэнкс Алекс. - М.: Питер, 2018. - 458 c.

11. Ньюмен, Сэм Создание микросервисов / Сэм Ньюмен. - М.: Питер, 2015. - 497 c.

12. Сэм, Ньюмен Создание микросервисов. Руководство / Ньюмен Сэм. - М.: Питер, 2016. - 145 c.

# **ПРИЛОЖЕНИЯ**

## Приложение А – Код Java Spring сервера.

### Листинг A.1.1.1 – Листинг CloudserverApplication.java

package com.bstu.cloudserver;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.EnableAutoConfiguration;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.context.properties.EnableConfigurationProperties;

@SpringBootApplication()

@EnableConfigurationProperties

@EnableAutoConfiguration

public class CloudserverApplication {

public static void main(String[] args) {

SpringApplication.run(CloudserverApplication.class, args);

}

}

### Листинг A.1.1.2 – Листинг CorsConfiguration.java

package com.bstu.cloudserver;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.servlet.config.annotation.CorsRegistry;

import org.springframework.web.servlet.config.annotation.EnableWebMvc;

import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration

@EnableWebMvc

public class CorsConfiguration implements WebMvcConfigurer

{

@Override

public void addCorsMappings(CorsRegistry registry) {

registry.addMapping("/\*\*")

.allowedMethods("GET", "POST","Delete");

}

}

### Листинг A.1.1.3 – Листинг OpenAPIConfiguration.java.

package com.bstu.cloudserver;

import io.swagger.v3.oas.annotations.OpenAPIDefinition;

import io.swagger.v3.oas.annotations.info.Contact;

import io.swagger.v3.oas.annotations.info.Info;

import io.swagger.v3.oas.annotations.info.License;

import io.swagger.v3.oas.annotations.servers.Server;

@OpenAPIDefinition(

info = @Info(

title = "Документация MyCloud APIv3",

description = "" +

"Описание основных запросов к облачному файловому хранилищу.",

contact = @Contact(

name = "Alex",

url = "https://github.com/alex-rudenkiy",

email = "alex-rudenkiy@mail.ru"

),

license = @License(

name = "MIT Licence",

url = "https://opensource.org/licenses/MIT")),

servers = @Server(url = "http://localhost:8181")

)

class OpenAPIConfiguration {

}

### Листинг A.1.1.4 – Листинг application.properties

server.port=8181

spring.jpa.hibernate.ddl-auto=update

spring.jpa.generate-ddl=true

spring.jpa.database-platform=org.hibernate.dialect.PostgreSQL95Dialect

spring.datasource.driverClassName=org.postgresql.Driver

spring.datasource.url= jdbc:postgresql://db:5432/docker?createDatabaseIfNotExist=true

#spring.datasource.url= jdbc:postgresql://localhost:5432/docker?createDatabaseIfNotExist=true

spring.datasource.username=docker

spring.datasource.password=docker

spring.jpa.show-sql=true

spring.session.store-type=none

spring.servlet.multipart.max-file-size=25MB

spring.servlet.multipart.max-request-size=25MB

springdoc.api-docs.enabled=true

springdoc.api-docs.path= /v3/api-docs

springdoc.swagger-ui.disable-swagger-default-url=true

springdoc.swagger-ui.url=/v3/api-docs

springdoc.swagger-ui.configUrl=/v3/api-docs/swagger-config

### Листинг A.1.1.5 – Листинг Client.java

package com.bstu.cloudserver.models.Client;

import lombok.\*;

import org.hibernate.annotations.GenericGenerator;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

import java.io.Serializable;

@Entity

@RequiredArgsConstructor

@NoArgsConstructor

public class Client implements Serializable

{

@Id

@Column(name = "uuid")

@GeneratedValue(generator = "system-uuid")

@GenericGenerator(name = "system-uuid", strategy = "uuid")

@Getter

@Setter

private String id;

@Column(name = "name")

@Getter

@Setter

@NonNull

private String name;

@Column(name = "email")

@Getter

@Setter

@NonNull

private String email;

@Column(name = "passhash")

@Getter

@Setter

@NonNull

private String passhash;

}

### Листинг A.1.2.1 – Листинг ClientController.java

package com.bstu.cloudserver.models.Client;

import com.bstu.cloudserver.Response;

import com.bstu.cloudserver.models.Client.dto.ReqLoginDto;

import com.bstu.cloudserver.models.Client.dto.ReqRegisterDto;

import com.bstu.cloudserver.models.Session.Session;

import com.google.gson.Gson;

import io.minio.errors.\*;

import io.swagger.v3.oas.annotations.Operation;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.ApplicationContext;

import org.springframework.stereotype.Controller;

import org.springframework.transaction.annotation.Transactional;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.ResponseBody;

import java.io.IOException;

import java.security.InvalidKeyException;

import java.security.NoSuchAlgorithmException;

@Controller

@Transactional

public class ClientController {

@Autowired

ApplicationContext context;

@Operation(summary = "Регистрация", description = "", tags = { "Пользователь" })

@RequestMapping(value = "/api/v1/client/register", method = RequestMethod.POST)

public @ResponseBody String userReg(@RequestBody ReqRegisterDto data) throws IOException, InvalidResponseException, InvalidKeyException, NoSuchAlgorithmException, ServerException, ErrorResponseException, XmlParserException, InsufficientDataException, InternalException {

Client v = context.getBean(ClientService.class).register(data);

return new Gson().toJson(new Response(v!=null?"ok":"failed",v));

}

@Operation(summary = "Авторизация", description = "", tags = { "Пользователь" })

@RequestMapping(value = "/api/v1/client/login", method = RequestMethod.POST)

public @ResponseBody String userLogin(@RequestBody ReqLoginDto data) {//, @RequestParam("remember") String remember

Session v = context.getBean(ClientService.class).login(data);

return new Gson().toJson(new Response(v!=null?"ok":"failed",v));

}

}

### Листинг A.1.2.2. – Листинг ClientJPA.java

package com.bstu.cloudserver.models.Client;

import org.hibernate.annotations.Entity;

import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

@Entity

public interface ClientJPA extends JpaRepository<Client, Long>

{

Boolean existsByName(String name);

Boolean existsByEmail(String email);

Boolean existsByPasshash(String passhash);

List<Client> findClientByNameEquals(String name);

}

### Листинг A.1.2.3 – Листинг ClientService.java

package com.bstu.cloudserver.models.Client;

import com.bstu.cloudserver.models.Client.dto.ReqLoginDto;

import com.bstu.cloudserver.models.Client.dto.ReqRegisterDto;

import com.bstu.cloudserver.models.FileStorage.FileStorageService;

import com.bstu.cloudserver.models.Session.Session;

import com.bstu.cloudserver.models.Session.SessionJPA;

import io.minio.MakeBucketArgs;

import org.apache.commons.codec.digest.DigestUtils;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.ApplicationContext;

import org.springframework.stereotype.Service;

import java.time.LocalDateTime;

import static com.bstu.cloudserver.models.FileStorage.Minio.minioClient;

@Service

public class ClientService {

@Autowired

ApplicationContext context;

Client register(ReqRegisterDto data){

Client result = null;

Client client = new Client(data.login, data.email, DigestUtils.md5Hex( data.password ));

if(!context.getBean(ClientJPA.class).existsByName(data.login) && !context.getBean(ClientJPA.class).existsByEmail(data.email)) {

result = context.getBean(ClientJPA.class).save(client);

try {

minioClient.makeBucket(

MakeBucketArgs.builder()

.bucket(context.getBean(FileStorageService.class).getBucketName(result))

.build());

}catch (Exception e){

context.getBean(ClientJPA.class).delete(client);

}

}

return result;

}

Session login(ReqLoginDto data){

Client u = null;

if(context.getBean(ClientJPA.class).existsByName(data.login) && context.getBean(ClientJPA.class).existsByPasshash(DigestUtils.md5Hex(data.password))) {

u = context.getBean(ClientJPA.class).findClientByNameEquals(data.login).get(0);

LocalDateTime actualDateTime = LocalDateTime.now();

if (data.remember) {

actualDateTime = actualDateTime.plusHours(1);

} else {

actualDateTime = actualDateTime.plusYears(1);

}

context.getBean(SessionJPA.class).deleteAllByClientEquals(u);

return context.getBean(SessionJPA.class).save(new Session(u, actualDateTime));

}else {

return null;

}

}

}

### Листинг A.1.3.1 – Листинг SharedFile.java

package com.bstu.cloudserver.models.FileStorage.SharedFile;

import com.bstu.cloudserver.models.Client.Client;

import lombok.\*;

import org.hibernate.annotations.GenericGenerator;

import javax.persistence.\*;

import java.io.Serializable;

import java.time.LocalDateTime;

@Entity

@RequiredArgsConstructor

@NoArgsConstructor

public class SharedFile implements Serializable {

@Id

@Column(name = "uuid")

@GeneratedValue(generator = "system-uuid")

@GenericGenerator(name = "system-uuid", strategy = "uuid")

@Getter

@Setter

private String id;

@OneToOne

@JoinColumn(name = "Client")

@Getter

@Setter

@NonNull

private Client client;

@Column(name = "Expire")

@Getter

@Setter

@NonNull

private LocalDateTime expires;

@Column(name = "Filepath")

@Getter

@Setter

@NonNull

private String filepath;

}

### Листинг A.1.3.2 – Листинг SharedFileController.java

package com.bstu.cloudserver.models.FileStorage.SharedFile;

import com.bstu.cloudserver.Response;

import com.bstu.cloudserver.models.FileStorage.FileStorageService;

import com.google.gson.Gson;

import io.swagger.v3.oas.annotations.Operation;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.ApplicationContext;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.\*;

import javax.servlet.http.HttpServletResponse;

@Controller

public class SharedFileController {

@Autowired

ApplicationContext context;

@Operation(summary = "Получить информацию об опубликованном файле", description = "", tags = { "Файловое пространство" })

@RequestMapping(value = "/api/v1/shared/{sharedFileId}", method = RequestMethod.GET)

public @ResponseBody

String getSharedFileInfo(@PathVariable String sharedFileId) {

SharedFile t = context.getBean(SharedFileJPA.class).getSharedFileByIdEquals(sharedFileId);

return new Gson().toJson(new Response(t!=null?"ok":"failed",t));

}

@Operation(summary = "Скачать опубликованный файл", description = "", tags = { "Файловое пространство" })

@RequestMapping(value = "/api/v1/shared/{sharedFileId}", method = RequestMethod.POST)

public void downloadSharedFile(@PathVariable String sharedFileId, HttpServletResponse response) {

SharedFile t = context.getBean(SharedFileJPA.class).getSharedFileByIdEquals(sharedFileId);

context.getBean(FileStorageService.class).getFile(t.getClient(), t.getFilepath(), response);

}

}

### Листинг A.1.3.3 – Листинг SharedFileJPA.java

package com.bstu.cloudserver.models.FileStorage.SharedFile;

import org.hibernate.annotations.Entity;

import org.springframework.data.jpa.repository.JpaRepository;

@Entity

public interface SharedFileJPA extends JpaRepository<SharedFile, Long>

{

SharedFile getSharedFileByIdEquals(String id);

}

### Листинг A.1.4.1 – Листинг Promo.java

package com.bstu.cloudserver.models.Promo;

import lombok.\*;

import org.hibernate.annotations.GenericGenerator;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

import java.io.Serializable;

@Entity

@RequiredArgsConstructor

public class Promo implements Serializable

{

@Id

@Column(name = "uuid")

@GeneratedValue(generator = "system-uuid")

@GenericGenerator(name = "system-uuid", strategy = "uuid")

@Getter

@Setter

private String id;

@Column(name = "name")

@Getter

@Setter

@NonNull

private String name;

@Column(name = "description")

@Getter

@Setter

@NonNull

private String description;

@Column(name = "secret")

@Getter

@Setter

@NonNull

private String secret;

@Column(name = "filespace")

@Getter

@Setter

@NonNull

private int filespace;

@Column(name = "cost")

@Getter

@Setter

@NonNull

private int cost;

@Column(name = "isEnabled")

@Getter

@Setter

@NonNull

private Boolean isEnabled;

@Column(name = "isDefault")

@Getter

@Setter

@NonNull

private Boolean isDefault;

public void Protected(){

}

}

### Листинг A.1.4.2 – Листинг PromoJPA.java

package com.bstu.cloudserver.models.Promo;

import org.hibernate.annotations.Entity;

import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

@Entity

public interface PromoJPA extends JpaRepository<Promo, Long>

{

List<Promo> findByNameEquals(String name);

List<Promo> findBySecretEquals(String name);

List<Promo> findByIsDefaultEquals(Boolean value);

List<Promo> findByIsEnabledEquals(Boolean value);

}

### Листинг A.1.4.3 – Листинг PromoService.java

package com.bstu.cloudserver.models.Promo;

import com.bstu.cloudserver.models.Session.SessionJPA;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.ApplicationContext;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class PromoService {

@Autowired

ApplicationContext context;

public Boolean getDefault(String token){

return !context.getBean(SessionJPA.class).findSessionByTokenEquals(token).isEmpty();

}

public Promo getByName(String name){

return context.getBean(PromoJPA.class).findByNameEquals(name).get(0);

}

public Promo getBySecret(String secret){

return context.getBean(PromoJPA.class).findBySecretEquals(secret).get(0);

}

public List<Promo> getByDefault(Boolean value){

return context.getBean(PromoJPA.class).findByIsDefaultEquals(value);

}

public List<Promo> getByEnabled(Boolean value){

return context.getBean(PromoJPA.class).findByIsEnabledEquals(value);

}

}

### Листинг A.1.5.1 – Листинг Session.java

package com.bstu.cloudserver.models.Session;

import com.bstu.cloudserver.models.Client.Client;

import lombok.\*;

import org.hibernate.annotations.GenericGenerator;

import javax.persistence.\*;

import java.io.Serializable;

import java.time.LocalDateTime;

@Entity

@RequiredArgsConstructor

@NoArgsConstructor

public class Session implements Serializable {

@Id

@Column(name = "uuid")

@GeneratedValue(generator = "system-uuid")

@GenericGenerator(name = "system-uuid", strategy = "uuid")

@Getter

@Setter

private String token;

@OneToOne

@JoinColumn(name = "Client")

@Getter

@Setter

@NonNull

private Client client;

@Column(name = "Expire")

@Getter

@Setter

@NonNull

private LocalDateTime expires;

}

### Листинг A.1.5.2 – Листинг SessionController.java

package com.bstu.cloudserver.models.Session;

import com.bstu.cloudserver.Response;

import com.google.gson.Gson;

import io.swagger.v3.oas.annotations.Operation;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.ApplicationContext;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.ResponseBody;

@Controller

public class SessionController {

@Autowired

ApplicationContext context;

@Operation(summary = "Проверить токен", description = "", tags = { "Сессия" })

@RequestMapping(value = "/api/v1/session/checkToken", method=RequestMethod.POST)

public @ResponseBody String auth(@RequestParam("token") String token) {

return new Gson().toJson(new Response("ok", context.getBean(SessionService.class).checkToken(token)));

}

}

### Листинг A.1.5.3 – Листинг SessionJPA.java

package com.bstu.cloudserver.models.Session;

import com.bstu.cloudserver.models.Client.Client;

import org.hibernate.annotations.Entity;

import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

@Entity

public interface SessionJPA extends JpaRepository<Session, Long>

{

List<Session> findSessionByTokenEquals(String token);

void deleteAllByClientEquals(Client client);

}

### Листинг A.1.5.4 – Листинг SessionService.java

package com.bstu.cloudserver.models.Session;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.ApplicationContext;

import org.springframework.stereotype.Service;

@Service

public class SessionService {

@Autowired

ApplicationContext context;

public Boolean checkToken(String token){

return !context.getBean(SessionJPA.class).findSessionByTokenEquals(token).isEmpty();

}

}

### Листинг A.1.6.1 – Листинг FileStorage.java

package com.bstu.cloudserver.models.FileStorage;

import com.bstu.cloudserver.models.Client.Client;

import com.bstu.cloudserver.models.Promo.Promo;

import lombok.\*;

import org.hibernate.annotations.GenericGenerator;

import javax.persistence.\*;

import java.io.Serializable;

import java.util.List;

@Entity

@RequiredArgsConstructor

@NoArgsConstructor

public class FileStorage implements Serializable {

@Id

@Column(name = "uuid")

@GeneratedValue(generator = "system-uuid")

@GenericGenerator(name = "system-uuid", strategy = "uuid")

@Getter

@Setter

private String id;

@OneToOne

@JoinColumn(name = "client")

@NonNull

private Client client;

@OneToMany

@JoinColumn(name = "promo")

@NonNull

private List<Promo> promo;

}

### Листинг A.1.6.2 – Листинг FileStorageController.java

package com.bstu.cloudserver.models.FileStorage;

import com.bstu.cloudserver.Response;

import com.bstu.cloudserver.models.Client.Client;

import com.bstu.cloudserver.models.FileStorage.Dto.ReqGetFileDto;

import com.bstu.cloudserver.models.FileStorage.Dto.ReqHandleFileDeleteDto;

import com.bstu.cloudserver.models.FileStorage.Dto.ReqHandleFileUploadDto;

import com.bstu.cloudserver.models.FileStorage.Dto.ReqProvideListInfoDto;

import com.bstu.cloudserver.models.Session.Session;

import com.bstu.cloudserver.models.Session.SessionJPA;

import com.google.gson.Gson;

import io.minio.PutObjectArgs;

import io.minio.errors.\*;

import io.swagger.v3.oas.annotations.Operation;

import io.swagger.v3.oas.annotations.Parameter;

import io.swagger.v3.oas.annotations.media.Schema;

import org.apache.commons.codec.DecoderException;

import org.apache.commons.codec.net.URLCodec;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.ApplicationContext;

import org.springframework.http.MediaType;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.\*;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.ByteArrayInputStream;

import java.io.IOException;

import java.security.InvalidKeyException;

import java.security.NoSuchAlgorithmException;

import java.util.List;

import static com.bstu.cloudserver.models.FileStorage.Minio.minioClient;

@Controller

public class FileStorageController {

@Autowired

ApplicationContext context;

@Operation(summary = "Получить содержимое папки", description = "", tags = { "Файловое пространство" })

@RequestMapping(value="/api/v1/mycloud/\*\*", method=RequestMethod.POST)

public @ResponseBody String provideListInfo(@RequestBody ReqProvideListInfoDto data, HttpServletRequest request) throws DecoderException {

String dirpath = "";

if(request.getRequestURL().toString().split("/mycloud/").length>1) dirpath = new URLCodec().decode(request.getRequestURL().toString().split("/mycloud/")[1]);

return new Gson().toJson(new Response("ok",context.getBean(FileStorageService.class).provideListInfo(data.getToken(), dirpath)));

}

@Operation(summary = "Загрузить файл в папку", description = "", tags = { "Файловое пространство" })

@RequestMapping(value="/api/v1/upload/mycloud/\*\*", method=RequestMethod.POST, consumes = MediaType.MULTIPART\_FORM\_DATA\_VALUE)

public @ResponseBody String handleFileUpload(

@ModelAttribute ReqHandleFileUploadDto data,

HttpServletRequest request) throws DecoderException {

String dirpath = "";

if(request.getRequestURL().toString().split("/mycloud/").length>1) {

String tpath = request.getRequestURL().toString().split("/mycloud/")[1];

dirpath = new URLCodec().decode(tpath);

}

return new Gson().toJson(new Response("ok",context.getBean(FileStorageService.class).handleFileUpload(data.getToken(), data.getFile(), dirpath)));

}

@Operation(summary = "Скачать файл в папку", description = "", tags = { "Файловое пространство" })

@RequestMapping(value = "/api/v1/download/mycloud/\*\*", method = RequestMethod.POST)

public void getFile(

@RequestBody ReqGetFileDto data,

HttpServletRequest request,

HttpServletResponse response) throws DecoderException {

String filePath = "";

if(request.getRequestURL().toString().split("/mycloud/").length>1) filePath = new URLCodec().decode(request.getRequestURL().toString().split("/mycloud/")[1]);

Client owner = null;

List<Session> t = context.getBean(SessionJPA.class).findSessionByTokenEquals(data.getToken());

if(!t.isEmpty()){

owner = t.get(0).getClient();

}

context.getBean(FileStorageService.class).getFile(owner, filePath, response);

}

@Operation(summary = "Создать пустую папку в папке", description = "", tags = { "Файловое пространство" })

@RequestMapping(value = "/api/v1/mkdir/mycloud/\*\*", method = RequestMethod.POST)

public @ResponseBody String mkDirectory(

@RequestBody ReqGetFileDto data,

HttpServletRequest request,

HttpServletResponse response) throws DecoderException {

String dirpath = "";

if(request.getRequestURL().toString().split("/mycloud/").length>1) dirpath = new URLCodec().decode(request.getRequestURL().toString().split("/mycloud/")[1]);

Client c = null;

List<Session> t = context.getBean(SessionJPA.class).findSessionByTokenEquals(data.getToken());

if(!t.isEmpty()){

c = t.get(0).getClient();

}

if (c!=null &&!dirpath.isEmpty()) {

try {

minioClient.putObject(

PutObjectArgs.builder()

.bucket(context.getBean(FileStorageService.class).getBucketName(c))

.object(dirpath)

.stream(

new ByteArrayInputStream(new byte[] {}), 0, -1)

.build());

new Gson().toJson(new Response("ok",""));

} catch (Exception e) {

return null;

}

} else {

return null;

}

return null;

}

@Operation(summary = "Удалить файл или папку", description = "", tags = { "Файловое пространство" })

@RequestMapping(value="/api/v1/delete/mycloud/\*\*", method = RequestMethod.POST)

public @ResponseBody String handleFileDelete(@Parameter(description="UserSessionToken",required=true, schema=@Schema(implementation = ReqHandleFileDeleteDto.class))

@RequestBody ReqHandleFileDeleteDto data,

HttpServletRequest request) throws IOException, InvalidKeyException, InvalidResponseException, InsufficientDataException, NoSuchAlgorithmException, ServerException, InternalException, XmlParserException, ErrorResponseException, DecoderException {

String filepath = "";

if(request.getRequestURL().toString().split("/mycloud/").length>1) filepath = new URLCodec().decode(request.getRequestURL().toString().split("/mycloud/")[1]);

return new Gson().toJson(new Response("ok",context.getBean(FileStorageService.class).handleFileDelete(data.getToken(), filepath)));

}

}

### Листинг A.1.6.3 – Листинг FileStorageService.java

package com.bstu.cloudserver.models.FileStorage;

import com.bstu.cloudserver.models.Client.Client;

import com.bstu.cloudserver.models.FileStorage.Dto.RespProvideListInfoDto;

import com.bstu.cloudserver.models.Session.Session;

import com.bstu.cloudserver.models.Session.SessionJPA;

import com.mpatric.mp3agic.ID3v2;

import com.mpatric.mp3agic.Mp3File;

import io.minio.\*;

import io.minio.errors.\*;

import io.minio.messages.Item;

import org.apache.commons.lang3.RandomStringUtils;

import org.apache.tomcat.util.http.fileupload.IOUtils;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.ApplicationContext;

import org.springframework.stereotype.Service;

import org.springframework.web.multipart.MultipartFile;

import javax.servlet.http.HttpServletResponse;

import java.io.\*;

import java.security.InvalidKeyException;

import java.security.NoSuchAlgorithmException;

import java.util.LinkedList;

import java.util.List;

import static com.bstu.cloudserver.models.FileStorage.Minio.minioClient;

@Service

public class FileStorageService {

@Autowired

ApplicationContext context;

public String getBucketName(Client client) {

System.out.println(client.getName());

return client.getName().replaceAll("[^A-Za-z0-9]", "");

}

public List<RespProvideListInfoDto> provideListInfo(String token, String dirpath) {

Client c = null;

List<RespProvideListInfoDto> result = new LinkedList<>();

List<Session> t = context.getBean(SessionJPA.class).findSessionByTokenEquals(token);

if (!t.isEmpty()) {

c = t.get(0).getClient();

Iterable<Result<Item>> results = minioClient.listObjects(

ListObjectsArgs.builder().bucket(context.getBean(FileStorageService.class).getBucketName(c)).prefix(dirpath).build());

results.forEach(e -> {

try {

result.add(new RespProvideListInfoDto(e.get().objectName(), e.get().objectName(), e.get().isDir()));

} catch (Exception e1) {

e1.printStackTrace();

}

});

}

return result;

}

public Boolean handleFileUpload(String token, MultipartFile file, String dirpath) {

Client c = null;

List<Session> t = context.getBean(SessionJPA.class).findSessionByTokenEquals(token);

if (!t.isEmpty()) {

c = t.get(0).getClient();

}

if (c != null && !file.isEmpty()) {

File randTmpFileName = new File(String.format("tmp\\%s.dat", RandomStringUtils.random(15, true, true)));

try {

byte[] bytes = file.getBytes();

BufferedOutputStream stream =

new BufferedOutputStream(new FileOutputStream(randTmpFileName));

stream.write(bytes);

stream.close();

System.out.println(dirpath + file.getOriginalFilename());

minioClient.uploadObject(

UploadObjectArgs.builder()

.bucket(context.getBean(FileStorageService.class).getBucketName(c))

.object(dirpath + file.getOriginalFilename())

.filename(randTmpFileName.getAbsolutePath())

.build());

if (file.getContentType().equals("audio/mpeg")) {

Mp3File mp3file = new Mp3File(randTmpFileName);

byte[] imageData = null;

ID3v2 id3v2Tag = null;

if (mp3file.hasId3v2Tag()) {

id3v2Tag = mp3file.getId3v2Tag();

imageData = id3v2Tag.getAlbumImage();

}

if (imageData != null) {

String mimeType = id3v2Tag.getAlbumImageMimeType();

String randName = String.format("%s.png", RandomStringUtils.random(15, true, true));

RandomAccessFile tfile = new RandomAccessFile("tmp\\" + randName, "rw");

tfile.write(imageData);

tfile.close();

minioClient.uploadObject(

UploadObjectArgs.builder()

.bucket("musicarts")

.object(randName)

.filename("tmp\\" + randName)

.build());

File dfile = new File(randName);

dfile.delete();

}

}

randTmpFileName.delete();

//return "Вы удачно загрузили " + name + " в " + name + "-uploaded !";

return true;

} catch (Exception e) {

//return "Вам не удалось загрузить " + name + " => " + e.getMessage();

return false;

}

} else {

return false;//return "Вам не удалось загрузить " + name + " потому что файл пустой.";

}

}

public Boolean handleFileDelete(String token, String filepath) throws IOException, InvalidKeyException, InvalidResponseException, InsufficientDataException, NoSuchAlgorithmException, ServerException, InternalException, XmlParserException, ErrorResponseException {

Client c = null;

List<Session> t = context.getBean(SessionJPA.class).findSessionByTokenEquals(token);

if (!t.isEmpty()) {

c = t.get(0).getClient();

}

minioClient.removeObject(

RemoveObjectArgs.builder().bucket(context.getBean(FileStorageService.class).getBucketName(c)).object(filepath).build());

return true;

}

public void getFile(

Client owner,

String filePath,

HttpServletResponse response) {

try {

try {

InputStream is = minioClient.getObject(

GetObjectArgs.builder()

.bucket(context.getBean(FileStorageService.class).getBucketName(owner))

.object(filePath)

.build());

response.setContentType("application/x-download");

response.setHeader("Content-disposition", "attachment; filename=" + ((GetObjectResponse) is).object());

System.out.println(((GetObjectResponse) is).object().toString());

IOUtils.copy(is, response.getOutputStream());

response.flushBuffer();

} catch (Exception e) {

}

} catch (Exception ex) {

throw new RuntimeException("IOError writing file to output stream");

}

}

public Boolean checkToken(String token) {

return false;

}

}

### Листинг A.1.7.1 – Листинг Minio.java

package com.bstu.cloudserver.models.FileStorage;

import io.minio.MinioClient;

public class Minio {

static public MinioClient minioClient =

MinioClient.builder()

.endpoint("http://localhost:9000")

//.endpoint("http://filestorage:9000")

.credentials("AKIAIOSFODNN7EXAMPLE", "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY")

.build();

}

### Листинг A.1.8.1 – Листинг initController.java

package com.bstu.cloudserver.models;

import com.bstu.cloudserver.models.Promo.Promo;

import com.bstu.cloudserver.models.Promo.PromoJPA;

import io.minio.MakeBucketArgs;

import io.minio.errors.\*;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.ApplicationContext;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.servlet.view.RedirectView;

import java.io.IOException;

import java.security.InvalidKeyException;

import java.security.NoSuchAlgorithmException;

import static com.bstu.cloudserver.models.FileStorage.Minio.minioClient;

@Controller

public class initController {

@Autowired

ApplicationContext context;

@RequestMapping(value = "/swagger", method = RequestMethod.GET)

public RedirectView localRedirect() {

RedirectView redirectView = new RedirectView();

redirectView.setUrl("/swagger-ui/index.html?configUrl=/v3/api-docs/swagger-config");

return redirectView;

}

@RequestMapping(value="/api/v1/init", method=RequestMethod.GET)

void init() throws IOException, InvalidKeyException, InvalidResponseException, InsufficientDataException, NoSuchAlgorithmException, ServerException, InternalException, XmlParserException, ErrorResponseException {

context.getBean(PromoJPA.class).save(

new Promo(

"Тариф базовый",

"Доступен абсолютно бесплатно и всем!",

"",

500,

0,

true,

true

)

);

context.getBean(PromoJPA.class).save(

new Promo(

"Тариф профессиональный",

"Доступен не абсолютно бесплатно и всем!",

"",

5000,

5000,

true,

true

)

);

}

}

## Приложение Б – Код ReactJS сервера.

### Листинг Б.1.1 – Листинг index.js

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import App from './App';

import \* as serviceWorker from './serviceWorker';

import {createBrowserHistory} from 'history'

import {SnackbarProvider} from "notistack";

const history = createBrowserHistory();

ReactDOM.render(

<React.StrictMode>

<SnackbarProvider maxSnack={3} anchorOrigin={{

vertical: 'bottom',

horizontal: 'right',

}}>

<App />

</SnackbarProvider>

</React.StrictMode>,

document.getElementById('root')

);

serviceWorker.unregister();

### Листинг Б.1.2 – Листинг App.js

import React, {useEffect} from 'react';

import './App.css';

import 'bootstrap/dist/css/bootstrap.min.css';

import {WelcomeFrame} from "./WelcomeFrame";

import {HomeFrame} from "./HomeFrame";

import { useSnackbar } from 'notistack';

import {

BrowserRouter as Router,

Switch,

Route

} from "react-router-dom";

import {useHistory} from "react-router-dom";

import {SharedFileFrame} from "./SharedFileFrame";

function useApiHook() {

let history = useHistory();

const axios = require('axios');

const rootUrl = "http://localhost:8181/api/v1"

const { enqueueSnackbar } = useSnackbar();

const handleClickVariant = (text, variant) => {

enqueueSnackbar(text, { variant });

};

const login = (login, password, remember) => {

axios({

method: 'post',

url: rootUrl + '/client/login',

data: {

login: login,

password: password,

remember: remember

},

})

.then(function (response) {

if (response.data.status === "ok") {

console.log(response);

localStorage.setItem('token', response.data.payload.token);

localStorage.setItem('login', response.data.payload.client.name);

history.push("/mycloud");

handleClickVariant(`Добро пожаловать ${response.data.payload.client.name} !`,'success');

}else{

handleClickVariant('Неправильный логин или пароль!','warning');

}

})

.catch(function (error) {

console.log(error);

handleClickVariant(error,'error');

});

}

const register = (nick, email, password, promo) => {

if(nick!==undefined && email!==undefined && password!==undefined &&

nick.length!==0 && email.length!==0 && password.length!==0) {

axios({

method: 'post',

url: rootUrl + '/client/register',

data: {

login: nick,

password: password,

email: email,

promo: promo

}

})

.then(function (response) {

console.log(response);

if (response.data.status === "ok") {

login(nick, password, false);

} else {

handleClickVariant('Логин или пароль уже был ранее зарегистрирован!', 'warning');

}

})

.catch(function (error) {

console.log(error);

handleClickVariant(error, 'error');

});

}else {

handleClickVariant('Необходимо заполнить все поля!', 'warning');

}

}

const logout = () => {

localStorage.clear();

checkToken();

}

const getLocalToken = () => {

return localStorage.getItem('token');

}

const getLocalData = () => {

return localStorage.getItem('login');

}

const isOnline = () => {

}

const listFiles = async (path) => {

return await new Promise((resolve, reject) => {

axios({

method: 'post',

url: rootUrl + path,

data: {

token: getLocalToken()

}

})

.then(function (response) {

console.log(response.data);

if (response.data.status === "ok") {

resolve(response.data.payload);

} else {

reject(response.data);

}

})

.catch(function (error) {

console.log(error);

handleClickVariant(error,'error');

});

})

}

const uploadFile = (file, path) => {

console.log("Закачиваю", path.slice(-1));

var formData = new FormData();

formData.append("file", file);

formData.append("token", getLocalToken());

return axios.post(rootUrl +"/upload" +path+(path.slice(-1)==="/"?"":"/"), formData, {

headers: {

'Content-Type': 'multipart/form-data'

}

})

}

const downloadFile = (file) => {

const FileDownload = require('js-file-download');

return axios({

url: rootUrl +"/download/mycloud/"+ file.filepath,

method: 'POST',

data: {

token: getLocalToken()

},

responseType: 'blob', // Important

}).then((response) => {

FileDownload(response.data, file.filename);

});

}

const deleteFile = (file) => {

return axios({

url: rootUrl +"/delete/mycloud/"+ file.filepath,

method: 'POST',

data: {

token: getLocalToken()

},

responseType: 'blob',

})

}

const createFolder = (path) => {

return axios({

url: rootUrl +"/mkdir"+ path+(path.slice(-1)==="/"?"":"/"),

method: 'POST',

data: {

token: getLocalToken()

},

})

}

const getFileInfo = async (sharedfiletoken) => {

return await new Promise((resolve, reject) => {

axios({

method: 'get',

url: rootUrl + "/shared/"+sharedfiletoken,

data: {

token: getLocalToken()

}

})

.then(function (response) {

console.log(response.data);

if (response.data.status === "ok") {

resolve(response.data.payload);

} else {

reject(response.data);

}

})

.catch(function (error) {

console.log(error);

handleClickVariant(error,'error');

});

})

}

const checkToken = () => {

axios({

method: 'post',

url: 'http://localhost:8181/api/v1/session/checkToken',

params: {

token: getLocalToken()!=null?getLocalToken():"",

},

})

.then(function (response) {

console.log(response);

if (response.data.payload === false) {

history.push("/welcome");

}

})

.catch(function (error) {

console.log(error);

handleClickVariant(error,'error');

});

}

return [login, register, logout,

getLocalToken, isOnline, listFiles,

uploadFile, downloadFile, deleteFile,

shareFile, createFolder, checkToken,

getLocalData, getFileInfo];

}

function App() {

return (

<div>

<Router>

<Switch>

<Route path="/welcome">

<WelcomeFrame useApiHook={useApiHook}/>

</Route>

<Route strict={false} exact={false} path="/mycloud">

<HomeFrame useApiHook={useApiHook}/>

</Route>

<Route path="/mygallery">

<HomeFrame useApiHook={useApiHook}/>

</Route>

<Route path="/mymusic">

<HomeFrame useApiHook={useApiHook}/>

</Route>

<Route path="/shared/:sharedFileId">

<SharedFileFrame useApiHook={useApiHook}/>

</Route>

<Route path="/">

<WelcomeFrame useApiHook={useApiHook}/>

</Route>

</Switch>

</Router>

</div>

);

}

export default App;

### Листинг Б.1.3 – Листинг HomeFrame.js

import React, {useState, useCallback, useEffect, useRef} from 'react';

import './App.css';

import 'bootstrap/dist/css/bootstrap.min.css';

import Typography from '@material-ui/core/Typography';

import {createMuiTheme} from '@material-ui/core/styles';

import {Row, Container, Col} from 'react-bootstrap';

import FullWidthTabs from "./TabPanel";

import {FilesManagerTab} from "./components/FilesManagerTab";

import 'react-h5-audio-player/lib/styles.css';

import AudioPlayer from 'react-h5-audio-player';

import {SearchTextBox} from "./components/SearchTextBox";

import SortIcon from '@material-ui/icons/Sort';

import FolderIcon from '@material-ui/icons/Folder';

import ReactDOMServer from 'react-dom/server';

import {motion} from "framer-motion";

import Button from '@material-ui/core/Button';

import {HeaderBar} from "./components/HeaderBar";

import {BrowserRouter as Router, Route, Switch, useHistory, useLocation} from 'react-router-dom'

import {MenuProvider} from "react-contexify";

import {ItemFigure} from "./components/ItemFigure";

import FormDialog from "./components/ModalDialog";

import {useSvgSource} from "./components/UseSvgSource";

import CreateNewFolderIcon from '@material-ui/icons/CreateNewFolder';

const theme = createMuiTheme({

palette: {

primary: {

light: '#757ce8',

main: '#3f50b5',

dark: '#002884',

contrastText: '#fff',

},

secondary: {

light: '#ff7961',

main: '#f44336',

dark: '#ba000d',

contrastText: '#000',

},

},

});

export function HomeFrame(props) {

const [login, register, exit, getLocalToken, isOnline, listFiles, uploadFile, downloadFile, deleteFile, shareFile, createFolder, checkToken] = props.useApiHook();

const location = useLocation();

const history = useHistory();

const [bodyBlurred, setBodyBlurred] = useState(false);

const [track, setTrack] = useState("");

const [showPlayer, setShowPlayer] = useState(false);

const [cloudPath, setCloudPath] = useState(location.pathname);

const [currentPageIndex, setCurrentPageIndex] = React.useState(location.pathname);//props.children[0].props.title

const [files, setFiles] = useState([]);

const [filesRefresh, setFilesRefresh] = useState(false);

const [openModalDialog, setOpenModalDialog] = React.useState(false);

const [newDirName, setNewDirName] = useState("");

useEffect(() => {

checkToken();

setFiles([]);

setCloudPath(location.pathname);

setCurrentPageIndex("/"+location.pathname.split("/")[1]);

const r = listFiles(location.pathname).then((f)=>{

let result = [];

Object.entries(f).forEach(([key, value]) =>

result.push( <MenuProvider id="menu\_id" style={{ display: 'inline-block' }} data={value}><ItemFigure data={value}/></MenuProvider>)

);

setFiles(result);

console.log("files = ",files);

})

setFilesRefresh(false);

}, [filesRefresh, location.pathname]);

useEffect(() => {

history.push(currentPageIndex);

}, [currentPageIndex]);

const playerRef = useRef();

useEffect(() => {

console.log(

ReactDOMServer.renderToString(

<Row>

<Typography className={"mb-4"} variant="h6">

Все файлы

</Typography>

<div><SearchTextBox/></div>

<div style={{

margin: "auto",

marginRight: "1em"

}}><SortIcon/></div>

</Row>

)

);

}, []);

const playFunc = (trackName) => {

setTrack(`${process.env.PUBLIC\_URL}/music/${trackName}`);

playerRef.current.audio.current.play();

showTags(`${process.env.PUBLIC\_URL}/music/${trackName}`);

}

return (

<div>

<header>

<useSvgSource/>

</header>

<div

style={{backgroundImage: "url(https://images.unsplash.com/photo-1508144322886-717c284ab392?ixlib=rb-1.2.1&q=80&fm=jpg&crop=entropy&cs=tinysrgb&w=800&h=533&fit=crop&ixid=eyJhcHBfaWQiOjF9)"}}>

</div>

<body style={{height: "100vh"}}>

<HeaderBar apiHook = {props.useApiHook} />

<Container className="h-100 d-inline-block" fluid className='mt-3'

style={{paddingLeft: "2em", paddingRight: "2em", marginBottom: "5em"}}>

<FullWidthTabs style={{maxHeight: "inherit"}} currentPageIndex={currentPageIndex} setCurrentPageIndex={setCurrentPageIndex}>

<div index={"/mycloud"} title="Мои файлы" icon={<FolderIcon/>} subroute={()=>{setCurrentPageIndex("/mycloud")}}>

<Row>

<Typography className={"mb-4"} variant="h6">

Все файлы

</Typography>

<div><SearchTextBox/></div>

<div style={{

margin: "auto",

marginRight: "1em"

}}>

<Button variant="contained" style={{backgroundColor:"#fdc300", color:"white"}} disableElevation onClick={()=>setOpenModalDialog(true)}>

Создать папку <CreateNewFolderIcon style={{marginLeft:"0.2em"}}/>

</Button>

<FormDialog open={openModalDialog} setOpen={(v)=>setOpenModalDialog(v)} textField={newDirName} setTextField={(v)=>setNewDirName(v)} createFolder={(p)=>{createFolder(`${location.pathname}${((p[0]==="/"||location.pathname[location.pathname.length-1]==="/")?"":"/")}${p}`).then(()=>setFilesRefresh(true))}}/>

</div>

</Row>

<FilesManagerTab files={files} useApiHook={props.useApiHook} setFilesRefresh={()=>setFilesRefresh(true)} cloudPath={cloudPath}/>

</div>

</FullWidthTabs>

</Container>

</body>

</div>

);

}

### Листинг Б.1.4 – Листинг SharedFileFrame.js

import React, {useEffect, useState} from "react";

import Typography from "@material-ui/core/Typography";

import Card from "@material-ui/core/Card";

import CardContent from "@material-ui/core/CardContent";

import CardActions from "@material-ui/core/CardActions";

import Button from "@material-ui/core/Button";

import {useParams} from "react-router";

import GetAppIcon from '@material-ui/icons/GetApp';

import \* as axios from "axios";

export function SharedFileFrame(props) {

let {sharedFileId} = useParams();

const [login, register, logout,

getLocalToken, isOnline, listFiles,

uploadFile, downloadFile, deleteFile,

shareFile, createFolder, checkToken,

getLocalData, getFileInfo] = props.useApiHook();

const downloadSharedFile = (link) => {

const FileDownload = require('js-file-download');

return axios(

{

url: link,

method: 'POST',

data: {

token: getLocalToken()

},

responseType: 'blob', // Important

}

).then(

(response) => {

FileDownload(response.data, fileInfo.filepath);

}

);

}

const [fileInfo, setFileInfo] = useState("");

const [authFields, setAuthFields] = useState({remember: false});

console.log("trig");

useEffect(async () => {

setFileInfo(await getFileInfo(sharedFileId))

}, []);

return (

<div className="container h-100" style={{position: "absolute", right: "0px", left: "0px"}}>

<div className="row align-items-center h-100">

<div className="col-6 mx-auto">

<Card>

<CardContent>

<Typography component="h5" variant="h5">

{fileInfo.filepath}

</Typography>

<Typography variant="subtitle1" color="textSecondary">

Владелец: {fileInfo.client !== undefined ? fileInfo.client.name : ""}

</Typography>

</CardContent>

<CardActions>

<Button size="small" onClick={() => downloadSharedFile("http://localhost:8181/api/v1/shared/"+fileInfo.id)}>Скачать<GetAppIcon

fontSize="small"/></Button>

</CardActions>

</Card>

</div>

</div>

</div>

);

}

### Листинг Б.1.5 – Листинг WelcomeFrame.js

import React, {useState, useCallback, useEffect, useRef} from 'react';

import './App.css';

import 'bootstrap/dist/css/bootstrap.min.css';

import AppBar from '@material-ui/core/AppBar';

import Toolbar from '@material-ui/core/Toolbar';

import Typography from '@material-ui/core/Typography';

import {createMuiTheme} from '@material-ui/core/styles';

import {Row, Container, Col} from 'react-bootstrap';

import {TabPanel} from "./TabPanel";

import 'react-h5-audio-player/lib/styles.css';

import Button from '@material-ui/core/Button';

import Card from "@material-ui/core/Card";

import CardContent from "@material-ui/core/CardContent";

import TextField from "@material-ui/core/TextField";

import CardActions from "@material-ui/core/CardActions";

import {AuthPanel} from "./components/AuthPanel";

import FormControlLabel from "@material-ui/core/FormControlLabel";

import Checkbox from "@material-ui/core/Checkbox";

const theme = createMuiTheme({

palette: {

primary: {

light: '#757ce8',

main: '#3f50b5',

dark: '#002884',

contrastText: '#fff',

},

secondary: {

light: '#ff7961',

main: '#f44336',

dark: '#ba000d',

contrastText: '#000',

},

},

});

export function WelcomeFrame(props) {

const [login, register, exit, getLocalToken, isOnline, listFiles, uploadFile, downloadFile, deleteFile, shareFile] = props.useApiHook();

const [panelIndex, setPanelIndex] = useState(0);

const [authFields, setAuthFields] = useState({remember: false});

return (

<div className="App">

<body>

<AppBar style={{}} position="static">

<Toolbar style={{

marginTop: "0.5em",

marginBottom: "0.5em"

}}>

<Typography variant="h6">

MyCloud

</Typography>

</Toolbar>

</AppBar>

<Container fluid className='mt-3' style={{paddingLeft: "2em", paddingRight: "2em"}}>

<Row>

<Col xs={5} sm={5} md={5} lg={5} xl={3}>

<Card className="pb-4 pt-0">

<AuthPanel panelIndex={panelIndex} setPanelIndex={setPanelIndex}/>

<div>

<TabPanel value={panelIndex} index={0} className={"pt-0"}>

<CardContent className={"p-0"}>

<div style={{marginLeft: "2em", marginRight: "2em"}}>

<TextField id="outlined-basic" label="Логин"

onChange={(e) => setAuthFields({...authFields, ...{"login": e.target.value}})}

style={{width: "100%", marginTop: "1em"}}/>

<TextField id="outlined-basic" label="Пароль"

onChange={(e) => setAuthFields({...authFields, ...{"password": e.target.value}})}

type={"password"} style={{width: "100%", marginTop: "1em"}}/>

<FormControlLabel

control={

<Checkbox

checked={authFields.remember}

onChange={(e) => setAuthFields({...authFields, ...{"remember": e.target.checked}})}

name="checkedB"

color="primary"

/>

}

label="Запомнить"

/>

</div>

</CardContent>

<CardActions className="float-left ml-4 mt-3 ">

<Button variant="contained" color="primary" size="medium"

onClick={() => login(authFields.login, authFields.password, authFields.remember)}>Войти</Button>

</CardActions>

</TabPanel>

</div>

<TabPanel value={panelIndex} index={1} className={"pt-0"}>

<CardContent className={"p-0"}>

<div style={{marginLeft: "2em", marginRight: "2em"}}>

<TextField id="outlined-basic" label="Логин"

onChange={(e) => setAuthFields({...authFields, ...{"login": e.target.value}})}

style={{width: "100%", marginTop: "1em"}}/>

<TextField id="outlined-basic" label="Почта"

onChange={(e) => setAuthFields({...authFields, ...{"email": e.target.value}})}

style={{width: "100%", marginTop: "1em"}}/>

<TextField id="outlined-basic" label="Пароль"

onChange={(e) => setAuthFields({...authFields, ...{"password": e.target.value}})}

type={"password"} style={{width: "100%", marginTop: "1em"}}/>

<TextField id="outlined-basic" label="Промо-код"

onChange={(e) => setAuthFields({...authFields, ...{"promo": e.target.value}})}

style={{width: "100%", marginTop: "1em"}}/>

</div>

</CardContent>

<CardActions className="float-left ml-4 mt-3 ">

<Button variant="contained" color="primary" size="medium"

onClick={() => register(authFields.login, authFields.email, authFields.password, authFields.promo)}>Зарегистрироваться</Button>

</CardActions>

</TabPanel>

</Card>

</Col>

<Col>

<Card elevation={0}>

<CardContent>

<Typography color="textSecondary" gutterBottom>

Мы сохраним все ваши файлы!

</Typography>

<div style={{

textAlign: "center",

width: "57%",

margin: "auto",

minWidth: "300px",

maxWidth: "1000px"

}}>

<img src={`${process.env.PUBLIC\_URL}/images/2020-10-18\_01-00-40.svg`}

className="img-fluid" alt="Responsive image"/>

</div>

</CardContent>

</Card>

</Col>

</Row>

</Container>

</body>

</div>);

}

### Листинг Б.2.1 – Листинг FilesManagerTab.js

import React from 'react';

import 'bootstrap/dist/css/bootstrap.min.css';

import {Row, Container, Col} from 'react-bootstrap';

import Figure from "react-bootstrap/Figure";

import '../assets/css/myStyle.css';

import { StyledDropZone } from 'react-drop-zone'

import { Menu, Item, Separator, Submenu, MenuProvider } from 'react-contexify';

import 'react-contexify/dist/ReactContexify.min.css';

export function FilesManagerTab(props) {

const [login, register, exit, getLocalToken, isOnline, listFiles, uploadFile, downloadFile, deleteFile, shareFile, checkToken] = props.useApiHook();

const onClick = ({ event, props }) => console.log(event,props);

const {files, setFilesRefresh, cloudPath} = props;

const MyAwesomeMenu = () => (

<Menu id='menu\_id'>

<Item onClick={({ event, props })=>{downloadFile(props).then(()=>setFilesRefresh());}}>Скачать</Item>

<Item onClick={onClick}>Поделиться</Item>

<Item onClick={({ event, props })=>{deleteFile(props).then(()=>setFilesRefresh()) }}>Удалить</Item>

<Separator />

<Item disabled>Переименовть</Item>

</Menu>

);

return (

<div>

<MyAwesomeMenu />

<Container fluid className='mt-3' className={"d-flex"} style={{height: "100vh", overflow: "overlay"}}>

<div className={"flex-fill"} style={{ height: "100%" }}>

<Row >

<Figure style={{

paddingBottom: "0px",

paddingTop: "0.5em",

marginBottom: "15px",

marginLeft: "0.5em",

paddingLeft: "0.5em",

paddingRight: "0.5em",

borderRadius: "5px",

}}>

<StyledDropZone

onDrop={(file, text) => uploadFile(file, cloudPath).then(()=>setFilesRefresh())}

style={{

width: "3em",

height: "3em",

zIndex: "12",

padding: "unset"

}}

>

<div class="row" style={{height: "100%"}}>

<div id="col" class="col-md-12 align-self-center">

<svg width="20" height="20" viewBox="0 0 16 16" class="bi bi-box-arrow-down-right" fill="currentColor" xmlns="http://www.w3.org/2000/svg">

<path fill-rule="evenodd" d="M8.636 12.5a.5.5 0 0 1-.5.5H1.5A1.5 1.5 0 0 1 0 11.5v-10A1.5 1.5 0 0 1 1.5 0h10A1.5 1.5 0 0 1 13 1.5v6.636a.5.5 0 0 1-1 0V1.5a.5.5 0 0 0-.5-.5h-10a.5.5 0 0 0-.5.5v10a.5.5 0 0 0 .5.5h6.636a.5.5 0 0 1 .5.5z"/>

<path fill-rule="evenodd" d="M16 15.5a.5.5 0 0 1-.5.5h-5a.5.5 0 0 1 0-1h3.793L6.146 6.854a.5.5 0 1 1 .708-.708L15 14.293V10.5a.5.5 0 0 1 1 0v5z"/>

</svg>

</div>

</div>

</StyledDropZone>

</Figure>

{

files

}

</Row>

</div>

</Container>

</div>

);

}

### Листинг Б.2.2 – Листинг HeaderBar.js

import React from 'react';

import 'bootstrap/dist/css/bootstrap.min.css';

import AppBar from '@material-ui/core/AppBar';

import Toolbar from '@material-ui/core/Toolbar';

import Typography from '@material-ui/core/Typography';

import {Row, Container, Col} from 'react-bootstrap';

import Image from "react-bootstrap/Image";

import '../assets/css/myStyle.css';

import 'react-contexify/dist/ReactContexify.min.css';

import Button from "@material-ui/core/Button/Button";

import MeetingRoomIcon from '@material-ui/icons/MeetingRoom';

import LogoIcon from "../assets/images/database.svg";

export function HeaderBar(props) {

const [login, register, logout, getLocalToken, isOnline, listFiles, uploadFile, downloadFile, deleteFile, shareFile, createFolder, checkToken, getLocalData] = props.apiHook();

return (

<AppBar style={{

backgroundColor: "transparent",

boxShadow: "none", color: "dimgrey", paddingLeft: "3.5em", paddingTop: "1em"

}} position="static">

<Toolbar style={{

marginTop: "0.5em",

marginBottom: "0.5em",

}}>

<Container style={{ margin: "inherit", display: "contents", width:"100%" }}>

<div><Image src={LogoIcon} style={{width: "2em"}} /></div>

<Col>

<Typography color="textPrimary" variant="h6" style={{fontWeight: "600"}}>

MyCloud

</Typography>

<Typography variant="subtitle2" color="textSecondary">

{getLocalData()}

</Typography>

</Col>

<Button disableElevation style={{ padding: "1em",

paddingLeft: "2em",

paddingRight: "2em"

}} onClick={()=>logout()}><Typography variant="subtitle2"style={{ }} >

Выход <MeetingRoomIcon/>

</Typography></Button>

</Container>

</Toolbar>

</AppBar>

);

}

### Листинг Б.2.3 – Листинг ItemFigure.js

import React, {useEffect, useState} from 'react';

import 'bootstrap/dist/css/bootstrap.min.css';

import Figure from "react-bootstrap/Figure";

import {useHistory} from "react-router-dom";

import Axios from "axios";

export function ItemFigure(props) {

const [img, setImg] = useState("");

const [isClicked, setIsClicked] = useState(false);

const [lastTimeClick, setLastTimeClick] = useState(Math.round(new Date() / 1000));

const history = useHistory();

const path = require('path');

const onDoubleClicked = ()=>{

setIsClicked(!isClicked);

console.log(Math.abs(new Date()-lastTimeClick));

if((new Date()-lastTimeClick)<300){

history.push("/mycloud/"+props.data.filename);

}

setLastTimeClick(new Date());

}

useEffect(()=> {

console.log("test");

Axios({

url: props.data.isDir?"/images/folder.svg":"/images/file.svg",

method: 'GET',

}).then((response) => {

setImg("data:image/svg+xml;base64," + btoa(response.data.replace("PNG",props.data.filename.split(".").pop().toUpperCase())));

});

}

,[img]);

return (

<Figure onClick={()=>onDoubleClicked()}

style={{width:"5em",

paddingBottom: "5px",

paddingTop: "0.5em",

marginBottom: "15px",

paddingLeft: "0.5em",

paddingRight: "0.5em",

textAlign: "center",

boxShadow: (isClicked)?"0 0 10px rgba(0,0,0,0.35)":"unset",

borderRadius: "5px"

}}>

<Figure.Image

width={50}

height={50}

src={img}

/>

<Figure.Caption >

<p style={{

textAlign: "center",

overflow: "hidden",

margin: "unset",

wordWrap: "break-word"

}}>{path.basename(props.data.filename).indexOf(".")>0? decodeURI(path.basename(props.data.filename).substring(0,props.data.filename.lastIndexOf("."))):decodeURI(path.basename(props.data.filename))}</p>

</Figure.Caption>

</Figure>

);

}

### Листинг Б.2.4 – Листинг ModalDialog.js

import React, {useState} from 'react';

//import Art from 'public/images/Summertime - Kreayshawn feat. V-Nasty.jpg'

import Button from "@material-ui/core/Button";

import Dialog from "@material-ui/core/Dialog";

import DialogTitle from "@material-ui/core/DialogTitle";

import DialogContent from "@material-ui/core/DialogContent";

import TextField from "@material-ui/core/TextField";

import DialogActions from "@material-ui/core/DialogActions";

export default function FormDialog(props) {

const {open, setOpen, textField, setTextField, createFolder} = props;

return (

<Dialog open={open} onClose={()=>setOpen(false)} aria-labelledby="form-dialog-title">

<DialogTitle id="form-dialog-title">Создание папки</DialogTitle>

<DialogContent>

{/\*<DialogContentText>

Введите название папки

</DialogContentText>\*/}

<TextField

autoFocus

margin="dense"

id="name"

label="Название папки"

value={textField}

type="text"

fullWidth

onChange={(v)=>setTextField(v.target.value)}

/>

</DialogContent>

<DialogActions>

<Button onClick={()=>setOpen(false)} color="primary">

Закрыть

</Button>

<Button onClick={()=>{createFolder(textField); setOpen(false);}} color="primary">

Создать

</Button>

</DialogActions>

</Dialog>

);

}

### Листинг Б.2.5 – Листинг AuthPanel.js

import React from 'react';

import 'bootstrap/dist/css/bootstrap.min.css';

import Typography from '@material-ui/core/Typography';

import '../assets/css/myStyle.css';

import 'react-contexify/dist/ReactContexify.min.css';

import { makeStyles, withStyles } from '@material-ui/core/styles';

import Tabs from '@material-ui/core/Tabs';

import Tab from '@material-ui/core/Tab';

const StyledTabs = withStyles({

indicator: {

display: 'flex',

justifyContent: 'center',

backgroundColor: 'transparent',

'& > span': {

maxWidth: 40,

width: '100%',

backgroundColor: '#635ee7',

},

},

})((props) => <Tabs {...props} TabIndicatorProps={{ children: <span /> }} />);

const StyledTab = withStyles((theme) => ({

root: {

textTransform: 'none',

color: '#979797',

fontWeight: theme.typography.fontWeightRegular,

fontSize: theme.typography.pxToRem(15),

marginRight: theme.spacing(1),

'&:focus': {

opacity: 1,

},

},

}))((props) => <Tab disableRipple {...props} />);

const useStyles = makeStyles((theme) => ({

root: {

flexGrow: 1,

},

padding: {

padding: theme.spacing(0),

},

demo1: {

backgroundColor: theme.palette.background.paper,

},

demo2: {

backgroundColor: '#ffffff',

},

}));

export function AuthPanel(props) {

const {panelIndex,setPanelIndex} = props;

const classes = useStyles();

const handleChange = (event, newValue) => {

setPanelIndex(newValue);

};

return (

<div className={classes.demo2} >

<StyledTabs value={panelIndex} onChange={handleChange} aria-label="styled tabs example" >

<StyledTab label="Авторизация" />

<StyledTab label="Регистрация" />

</StyledTabs>

<Typography className={classes.padding} />

</div>

);

}