Липецкий государственный технический университет

Факультет автоматизации и информатики
Кафедра прикладной математики

ОТЧЕТ
о производственной практике
в ООО "МедСофт"

Руководитель от предприятия	_____ подпись	_____ дата	Акулов С. В. фамилия, инициалы
Студент	_____ подпись	_____ дата	Воротников П. А. фамилия, инициалы

Группа ПМ-20-1

Руководитель от кафедры ПМ	_____ подпись	_____ дата	Орешина М. Н. фамилия, инициалы
	_____ подпись	_____ дата	Акулова С. А. фамилия, инициалы

Липецк 2023 г.

Содержание

1. Общая характеристика предприятия

- **Полное наименование:** Общество с ограниченной ответственностью «МедСофт»;
- **Сокращенное наименование:** ООО «МедСофт»;
- **Директор:** Карноза Владимир Викторович (действует на основании Устава);
- **Юридический адрес:** Россия, 398017, г. Липецк, ул. 9 мая, владение 27, помещение 2, офис 301;
- **Дата создания:** 2006 год;
- **Продукты компании:**Квазар.ИПРА, Квазар.Клиника, Квазар.КМИС, Квазар.Фарм, Квазар.РИР;
- **Род деятельности:** Разработка программного обеспечения для платных клиник и иных лечебных учреждений.

2. Задание

Разработать микросервис (аналог dropmefiles), который будет реализовывать функционал добавления, хранения и получения файлов по сети. В процессе разработки использовать систему контроля версий Git.

3. Ход работы

1. Изучение среды разработки IntellijIdea;
2. Создание проекта с использованием системы сборки Gradle;
3. Создание простейшего микросервисного приложения;
4. Проектирование архитектуры, разделение приложения на слои;
   (a) Создание контроллера;
   (b) Выделение абстракции DocumentRepository;
   (c) Реализация DocumentRepository с сохранением в файловую систему;
   (d) Проектирование БД и DBDocumentrepository;
5. Рефакторинг кода.

4. Основная часть

4.1 Изучение среды разработки IntellijIdea

IntelliJ IDEA —- это IDE, интегрированная среда разработки (комплекс программных средств, который используется для написания, исполнения, отладки и оптимизации кода) для Java, JavaScript, Python и других языков программирования от компании JetBrains. Отличается обширным набором инструментов для рефакторинга (перепроектирования) и оптимизации кода.
В данной работе она будет использоваться для написания кода на языке программирования Java.

4.2 Создание проекта с использованием системы сборки Gradle

Для создания приложения на Spring Boot необходимо создать базовый проект в Intellij Idea, а затем добавить в зависимости Gradle следующую строчку *implementation 'org.springframework.boot:spring-boot-starter-web'*.

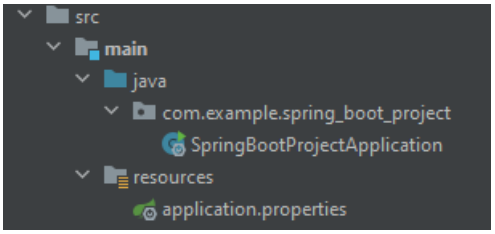

Рис. 1 – Файлы проекта

4.3 Создание простейшего микросервисного приложения

Приложение должно уметь получать файл по сети, сохранять его в хранилище и создавать в базе данных строку, которая будет со-

держать имя файла, email пользователя, короткую ссылку на файл и путь к файлу. Помимо этого, приложение будет после добавления файла отправлять на почту пользователя ссылку на созданный файл.

4.4 Проектирование архитектуры/разделение на слои

Запросы на добавление/получение файла будут обрабатываться контроллером. Контроллер взаимодействует с сущностью DocumentRepository, которая предоставляет интерфейс для сохранения и извлечения данных. В свою очередь, DocumumentRepository использует интерфейс, который предоставляет ему сущность DBManager, задача которой – работать с БД.

4.5 Создание сущности хранимых экземпляров

Сначала надо определить, какие данные будут определять сущность хранимого файла. Для этого создадим класс поля которого будут определять имя, путь, email, короткую ссылку файла.

```
public class FileEntity {
    public FileEntity(){

    }
    private String name;
    private String originalName;
    private String email;
    private String url;

    public String getUrl() {
        return url;
    }
```

```
public void setUrl(String url) {
    this.url = url;
}

public String getOriginalName() {
    return originalName;
}

public void setOriginalName(String originalName) {
    this.originalName = originalName;
}

private byte[] file;

public void setName(String name) {
    this.name = name;
}

public void setEmail(String email) {
    this.email = email;
}



public String getName() {
    return name;
}
public String getEmail() {
    return email;
}
```

```
    public byte[] getFile() {
        return file;
    }

    public void setFile(byte[] file) {
        this.file = file;
    }
}
```

4.6 Создание контроллера

Основные методы в контроллере следующие: метод получения файла из хранилища и метод сохранения файла.

Метод получения файла:

```
@RequestMapping(value = "add/file", method = RequestMethod.POST,
consumes = {MediaType.MULTIPART_FORM_DATA_VALUE})
    public String saveFile(@RequestParam String name,
        @RequestParam MultipartFile document,
        @NotNull @Email @RequestParam String email)
        throws IOException, OperationNotSupportedException,
        SQLException {

        FileEntity fileEntity = new FileEntity();
        fileEntity.setFile(document.getBytes());
        fileEntity.setName(name);
        fileEntity.setOriginalName(document.getOriginalFilename());
        fileEntity.setEmail(email);

        String url =  documentRepository.add(fileEntity);
        service.sendMessage(url, fileEntity.getEmail());
```

```
        return url;
    }
```

Метод получения файла по короткой ссылке:

```
@GetMapping(value = "/{url}")
    public FileEntity GetFile(@PathVariable String url)
    throws IOException, SQLException {
        return documentRepository.get(url);
    }
```

4.7 Выделение абстракции DocumentRepository

Интерфейс DocumentRepository будет определять, какие методы можно использовать для операции с данными. Методы, которые используются контроллером для сохранения и извлечения данных, называются add и get.

Реализация интерфейса, которой пользуется контроллер, называется DBDocumentRepository. Реализует он два вышеперечисленных метода следующим образом.

Метод add:

```
@Override
    public String add(FileEntity file)
    throws OperationNotSupportedException,
    SQLException, IOException {
        if(Files.exists(
        Paths.get(directoryProperties.path(), file.getName()))){
            throw new FileIsAlreadyExistException
            ("Файл с таким именем уже существует");
        }

        Files.write(Paths.get
        (directoryProperties.path(), file.getName()),
        file.getFile());

        int id = dbManager.insert
        ("INSERT INTO file (name_file, email) VALUES (?, ?);",
        new Object[]{file.getName(), file.getEmail()});
        dbManager.update
        ("UPDATE file SET short_url=? WHERE id=?;",
        new Object[]{URLGenerator.encode(id), id});

        return URLGenerator.encode(id);
    }
```

Метод get:

```
@Override
    public FileEntity get(String url) {
        try{
            List<FileEntity> result =
            dbManager.query(this::map,
```

```
                "SELECT * FROM file WHERE short_url = ?;",
                new Object[]{url});

        if(result == null ||
        result.isEmpty()){
            throw new
            FileNotFoundException("Файл не найден");
        }

        return result.get(0);
    }
    catch (SQLException e){
        throw new RuntimeException(e);
    }
}
```

4.8 Проектирование БД и DBManager

База данных, используемая для хранения данных о файлах, содержит одну таблицу file, которая состоит из полей id (универсальный идентификатор), name_file (имя файла), email (почта пользователя, добавившего файл), short_url (короткая ссылка, которая определяется значением определённой функции с аргументом, равным id файла).

Необходимо создать класс, который будет отвечать за подключение и работу с БД (PostgreSQL).

```
private PreparedStatement prepare(PreparedStatement statement,
Object[] params) throws SQLException {
    for(int i = 0; i < params.length; i++){
        statement.setObject(i + 1, params[i]);
    }
```

```
    return statement;
}


public <T> T query(Function<ResultSet, T> mapper, String sql,
Object[] params) throws SQLException {
    try(Connection connection = dataSource.getConnection();
        PreparedStatement preparedStatement =
        connection.prepareStatement(sql);){
        prepare(preparedStatement, params);
        var exec = preparedStatement.executeQuery();
        return mapper.apply(exec);
    }
}

public int insert(String sql, Object[] params)
throws SQLException {
    try(Connection connection = dataSource.getConnection();
        PreparedStatement preparedStatement =
        connection.prepareStatement(sql,
        Statement.RETURN_GENERATED_KEYS);){
        prepare(preparedStatement, params);
        preparedStatement.executeUpdate();
        var exec = preparedStatement.getGeneratedKeys();
        if(exec.next()){
            return exec.getInt("id");
        }
        return -1;
    }
}
```

4.9 Генерация коротких ссылок

Для генерации коротких ссылок будем использовать класс, которых содержит две функции: одна из них преобразует id в короткую ссылку, другая преобразует короткую ссылку в id.

```
public static String encode(long id){
    StringBuilder stringBuilder = new StringBuilder();
    if(id == 0){
        stringBuilder.append(allowedCharacters[0]);
    }
    while(id > 0){
        stringBuilder.append(allowedCharacters[(int)(id % base)]);
        id /= base;
    }
    return stringBuilder.toString();
}

public static long decode(String string){
    char[] array = string.toCharArray();
    int id = 0;
    for(int i = 0; i < array.length; i++){
        id += allowedString.indexOf(array[i]) * Math.pow(base, i);
    }
    return id;
}
```

4.10 Использование Nginx

Для того, чтобы ссылки на файлы приобрели разумный вид, будем перенаправлять запросы, направленные на url dropmefiles.com, на наш сервис с помощью службы Nginx.

Чтобы добиться этого, в файле hosts надо прописать строку *127.0.0.1 dropmefiles.com*, которая будет перенаправлять соответствующие запросы на сервер, на котором работает сервис.

Потом нужно добавить в конфигурации Nginx следующие строки:

```
location / {
    proxy_pass http://127.0.0.1:9002;
    proxy_http_version 1.1;
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
}
```

После этого nginx будет перенаправлять запросы на порт 9002, на котором работает приложение.

4.11 Отправление ссылки на электронную почту

Для того, чтобы приложение умело работать с электронной почтой, необходимо в зависимости проекта добавить *implementation 'org.springframework.boot:spring-boot-starter-mail'*.

Вышеупомянутый класс контроллера при добавлении файла вызывает функцию отправки сообщения на почту. Для реализации этой функции в проекте создан класс MyEmailService. Ниже приведён код этого класса.

```
@Service
public class MyEmailService {
    @Autowired
    private JavaMailSender sender;
    public void sendMessage(String text, String email){
```

```
        SimpleMailMessage message = new SimpleMailMessage();

        message.setTo(email);
        message.setText(text);

        sender.send(message);
    }
}
```

4.12 Git

В ходе работы использовалась система контроля версий Git. Дерево коммитов приведено на скриншоте.

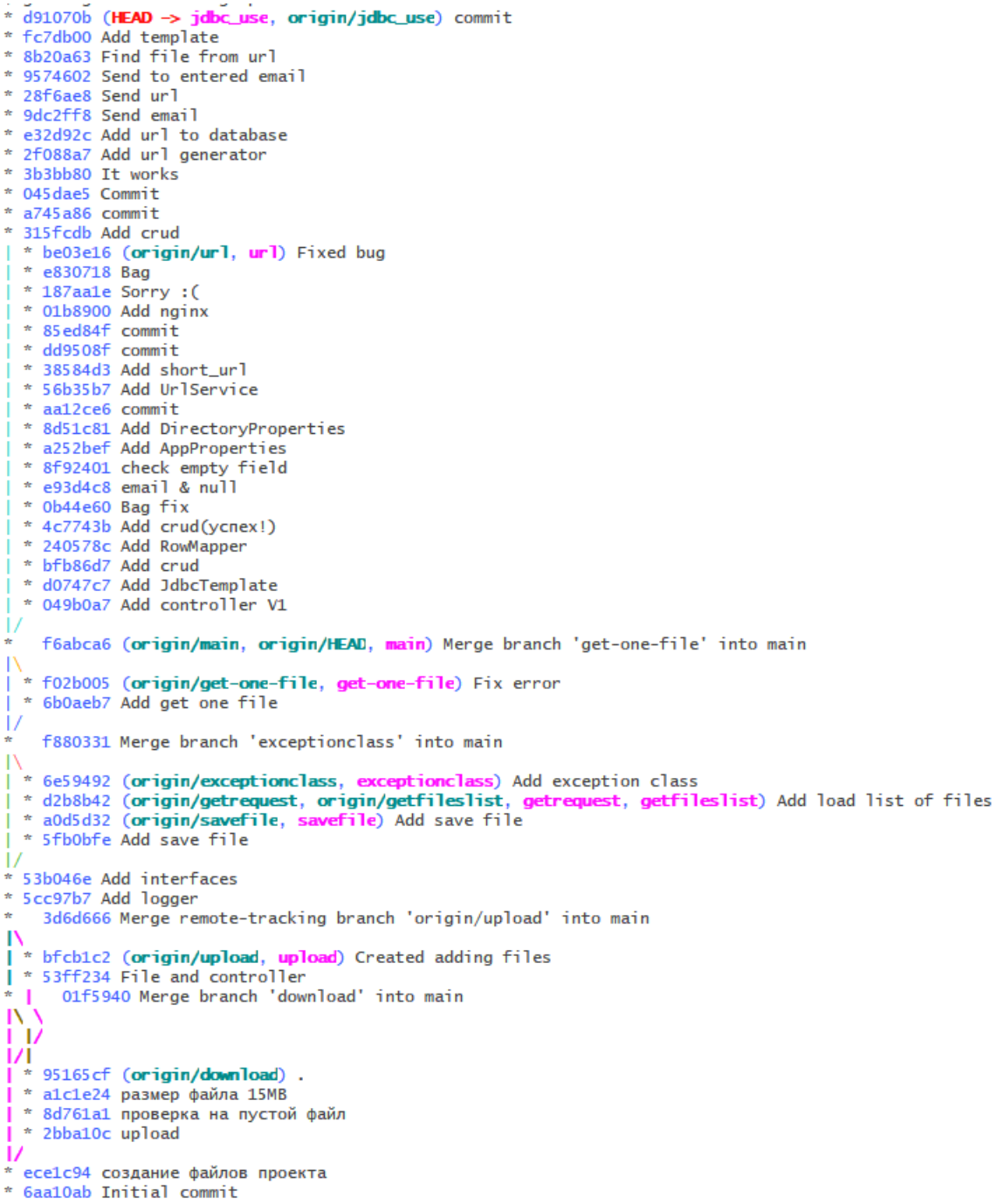

Рис. 2 – Дерево коммитов

5. Вывод.

В ходе проделанной работы был спроектирован микросервис с несколькими слоями: контроллер, репозиторий, модель. Контроллер отвечает за работу с запросами, репозиторий работает с сохранением и получением данных. Модель работает непосредственно с БД.

Был изучен паттерн проектирования Repository, среда разработки Intellij Idea, система сборки проекта Gradle, сервис Nginx для перенаправления запросов, микросервисный фреймворк Spring Boot и система контроля версий Git.