一、 数据库设计

我们对网页业务进行了需求分析，通过分析拟定了数据库的表名和字段，其中表包含 users 用户表、friends 好友表、music_upload 上传音乐表、music_like 点赞的音乐、music_fav 收藏的音乐，comment 评论表，bloc 博客表，具体每个表的设计和建表命令如下：

1.users 表存放数据库信息：id（主键,自增）account（不可重复） password（密码，加密）phone email age sex birthday create_date introduction(50 字符以内) show(展示范围) avator（头像）

```
CREATE TABLE `users` (
`id` BIGINT(0) NOT NULL AUTO_INCREMENT,
`account` VARCHAR(64) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL COMMENT '账号',
`nickname` VARCHAR(64) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL COMMENT '昵称',
`password` VARCHAR(64) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL COMMENT '密码',
`create_date` DATE NULL DEFAULT NULL COMMENT '注册日期',
`birthday` DATE NULL DEFAULT NULL COMMENT '生日',
`email` VARCHAR(128) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '邮箱',
`phone` VARCHAR(20) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '手机号',
`age` INT NULL DEFAULT NULL COMMENT '年龄',
`sex` VARCHAR(6) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT NULL COMMENT '性别',
`introduction` VARCHAR(256) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL COMMENT '简介',
`show` INT NULL DEFAULT 0 COMMENT '展示范围',
`avator` VARCHAR(256) CHARACTER SET utf8 COLLATE utf8_general_ci NULL COMMENT '头像',
PRIMARY KEY (`id`) USING BTREE
) ENGINE = INNODB AUTO_INCREMENT = 16 CHARACTER SET = utf8 COLLATE = utf8_general_ci
```

2.friends 存放好友: user1 user2 status

```
CREATE TABLE `friends` (
  `user1_id` BIGINT(0) NOT NULL,
  `user2_id` BIGINT(0) NOT NULL,
  `status` INT NULL DEFAULT NULL COMMENT '状态',
  PRIMARY KEY (`user1_id`,`user2_id`)
  ) ENGINE = INNODB AUTO_INCREMENT = 16 CHARACTER SET = utf8 COLLATE = utf8_general_ci
```

3**. music_upload 我上传的音乐: id（主键） up_user_id(空代表官方上传) name save_path photo_path type(短或长) duration description up_time permission(是否公开)**

```
  CREATE TABLE `music_upload` (
  `id` BIGINT(0) NOT NULL AUTO_INCREMENT,
  `up_user_id` BIGINT(0) NULL COMMENT '上传用户 id',
  `name` VARCHAR(64) CHARACTER SET utf8 COLLATE utf8_general_ci NULL COMMENT '音频名称',
  `up_time` DATE NULL DEFAULT NULL COMMENT '上传时间',
  `save_path` VARCHAR(256) CHARACTER SET utf8 COLLATE utf8_general_ci NULL COMMENT '保存路径',
  `photo_path` VARCHAR(256) CHARACTER SET utf8 COLLATE utf8_general_ci NULL COMMENT '对应图片路径',
  `description` VARCHAR(256) CHARACTER SET utf8 COLLATE utf8_general_ci NULL COMMENT '描述',
  `type` INT NOT NULL COMMENT '音频类型',
  `duration` INT NOT NULL COMMENT '秒数',
  `permission` INT NOT NULL DEFAULT 0 COMMENT '查看权限',
  PRIMARY KEY (`id`) USING BTREE
  ) ENGINE = INNODB AUTO_INCREMENT = 16 CHARACTER SET = utf8 COLLATE = utf8_general_ci
```

4. music__favorite 我收藏的音乐: user_id music_id fav_time type(短或长)

```
  CREATE TABLE `music_favorite` (
  `user_id` BIGINT(0) NOT NULL COMMENT '用户 id',
  `music_id` BIGINT(0) NOT NULL COMMENT '音频 id',
  `fav_time` DATE NULL DEFAULT NULL COMMENT '收藏时间',
  `type` INT NOT NULL COMMENT '音频类型',
  PRIMARY KEY (`user_id`,`music_id`)
  ) ENGINE = INNODB AUTO_INCREMENT = 16 CHARACTER SET = utf8 COLLATE = utf8_general_ci
```

5.music__like 我点赞的音乐: user_id music_id like_time type(短或长)

```
  CREATE TABLE `music_favorite` (
  `user_id` BIGINT(0) NOT NULL COMMENT '用户 id',
  `music_id` BIGINT(0) NOT NULL COMMENT '音频 id',
  `fav_time` DATE NULL DEFAULT NULL COMMENT '收藏时间',
  `type` INT NOT NULL COMMENT '音频类型',
  PRIMARY KEY (`user_id`,`music_id`)
  ) ENGINE = INNODB AUTO_INCREMENT = 16 CHARACTER SET = utf8 COLLATE = utf8_general_ci
```

//长音频的 id 和短音频的映射，包括开始时间，音轨，文件路径等

6.评论：id content comment_time **music_id user_id parent_id to_user_id(二层评论所评论的评论的 user) level（1/2 这里只支持二层评论）**

```
CREATE TABLE `music_comment` (
`id` BIGINT(0) NOT NULL AUTO_INCREMENT COMMENT '上传用户 id',
`content` VARCHAR(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci NOT NULL COMMENT '内容',
`comment_time` DATE NOT NULL COMMENT '评论时间',
`music_id` BIGINT(0) NOT NULL COMMENT '音乐 id',
`user_id` BIGINT(0) NOT NULL COMMENT '评论用户 id',
`parent_id` BIGINT(0) NOT NULL COMMENT '二层评论所评论的评论 id',
`to_user_id` BIGINT(0) NOT NULL COMMENT '二层评论所评论的用户 id',
`level` INT NOT NULL DEFAULT 1 COMMENT '层级',
PRIMARY KEY (`id`) USING BTREE,
INDEX `music_id`(`music_id`) USING BTREE
) ENGINE = INNODB AUTO_INCREMENT = 1 CHARACTER SET = utf8 COLLATE = utf8_general_ci
```

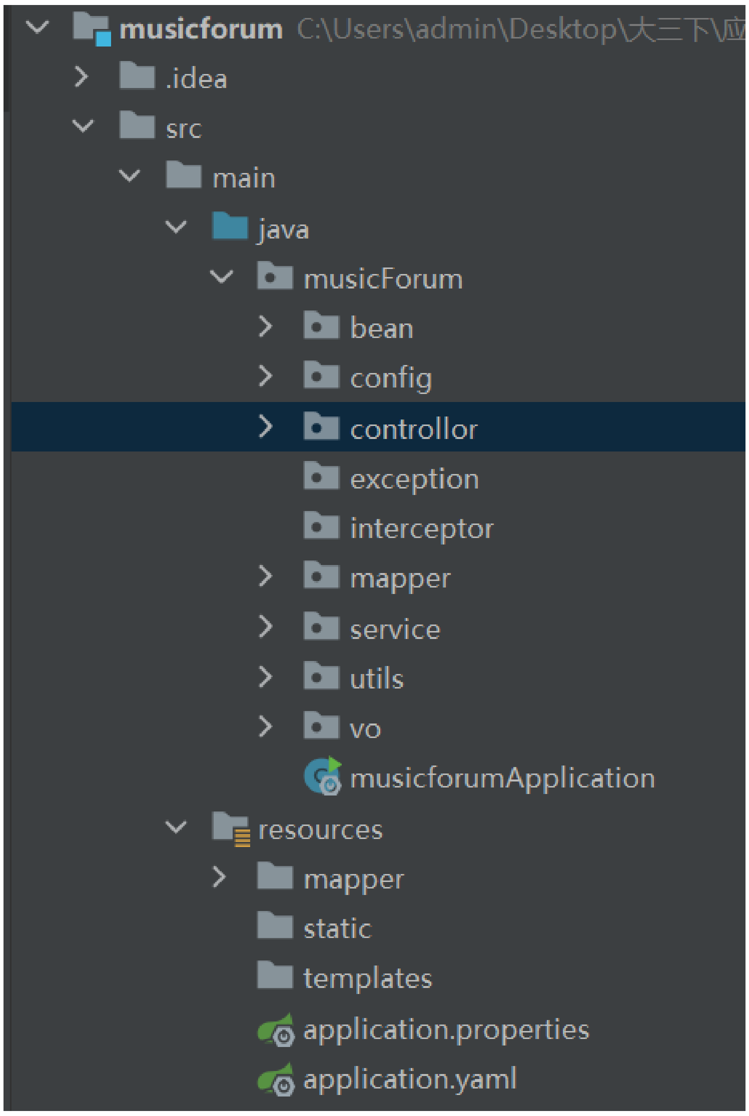
7.bloc 博客: id user_id text up_time music_id like_num fav_num permission

```
CREATE TABLE `bloc` (
`id` BIGINT(0) NOT NULL AUTO_INCREMENT,
`user_id` BIGINT(0) NULL COMMENT '上传用户 id',
`up_time` DATE NULL DEFAULT NULL COMMENT '上传时间',
`text` VARCHAR(512) CHARACTER SET utf8 COLLATE utf8_general_ci NULL COMMENT '文本',
`music_id` BIGINT(0) NULL COMMENT '附件音乐 id',
`like_num` INT NOT NULL DEFAULT 0 COMMENT '点赞数',
`fav_num` INT NOT NULL DEFAULT 0 COMMENT '收藏数',
`permission` INT NOT NULL DEFAULT 0 COMMENT '查看权限',
PRIMARY KEY (`id`) USING BTREE
) ENGINE = INNODB AUTO_INCREMENT = 16 CHARACTER SET = utf8 COLLATE = utf8_general_ci
```

二、 后端设计

后端选择 java 作为设计语言，IDEA 作为编程 IDE，maven 作为构建工具，使用 springboot+mybatisplus 作为开发框架。其中 springboot 作为 java 开发 web 程序的常用框架，mybatis 作为 java 操作数据库的框架而 mybatisplus 是对 mybatis 的增强减少了查询 xml 的书写工作量，这两个框架是当下 java 开发网页的流行框架，体现了后端的技术先进性。

后端代码架构如下：



```
musicforum C:\Users\admin\Desktop\大三下\应
  .idea
  src
    main
      java
        musicForum
          bean
          config
          controllor
          exception
          interceptor
          mapper
          service
          utils
          vo
          musicforumApplication
      resources
        mapper
        static
        templates
        application.properties
        application.yaml
```

后端采用了多层架构融合了 MVC 框架，bean 代表数据库接口层，其

中一个表对应一个类，config 类表示 springboot 中的配置类，包括 mybatisplus 配置、分页插件配置等等，controllor 类代表与网页的接口，拦截网页请求并调用 service 层服务返回给前端，service 层即服务层调用 mapper 层操作数据库完成各种服务，mapper 层代表 mybatisplus 操作数据库的接口层，exception 代表异常类、interceptor 代表拦截器，这两个包暂时没有实现，vo 包里是用于前后端数据交互的类，utils 包里是工具类，包括状态码类、result 返回结果类、操作七牛云服务器的类，利用 ffmpeg 进行音乐合成等的类。

1. Bean 包设计

Bean 里包含每个表的类，其中各个类都类似，这里只展示 users 表的类，其中@Data 等注解用于自动生成 get、set 方法和构造函数，@TableId 表示主键，mybatisplus 的主键默认使用雪花算法做分布式 id 而不是简单的递增，可以使得数据量大分表时也可以应对，@TableField 用于解决 mybatisplus 自动生成查询语句时的关键字冲突问题，数据字段采用 private 提高安全性，代码如下：

```java
@Data
@AllArgsConstructor
@NoArgsConstructor
public class users {
    @TableId
    private Long id;
```

```java
    //解决关键字冲突

    @TableField("`account`")
    private String account;
    private String nickname;
    @TableField("`password`")
    private String password;
    private String phone;
    private String email;
    private Integer age;
    private String sex;
    private Date createTime;
    private String birthday;
    private String introduction;
    @TableField("`show`")
    private Integer show;
```

```
    //头像，有一个默认头像,写在配置文件中
    private String avatar;
}
```

2. controllor 包设计

controller 层主要用于拦截浏览器的请求并提供服务返回数据，包括查询个人信息，查询音频信息，设置个人信息，上传音频，合成音频等等…

cotrollor 层基本上调用一个或几个 service 层的服务，因此类似，不过文件存储服务器的部分会在 controllor 层之间完成，下面展示上传音频的代码，postMapper 代表接受 post 请求，后面是请求路径，传入音频的各种信息，将音频的图片和音频本身存入服务器，其中名字使用 randomUUID 生成唯一的分布式 ID，调用了 qiniu 云的 utils 传入 qiniu 云，字节流直传传入本机，再调用 service 层将信息存入数据库，代码如下：

```
@PostMapping("uploadMusic")
    public Result uploadMusic(@RequestBody Long id,
@RequestBody String name,
                           @RequestBody String
description, @RequestBody Integer type,
                           @RequestBody Integer
permission, @RequestBody MultipartFile photo,
                           @RequestBody MultipartFile
music, @RequestBody String generatePath) {
```

```
       //记得更新我的点赞状态，不要重复点赞
```

```
       //原始文件名称
       String originalFilename =
photo.getOriginalFilename();
        String photoPath =
UUID.randomUUID().toString() + "."
                +
StringUtils.substringAfter(originalFilename, ".");
```

```
        //上传到七牛云服务器
        boolean upload = qiniuUtils.upload(photo,
musicImgParentPath+photoPath);
        if (!upload){
            return
Result.fail(ErrorCode.UPLOAD_MUSIC_PHOTO_ERROR.getCod
e(),

ErrorCode.UPLOAD_MUSIC_PHOTO_ERROR.getMsg());
```

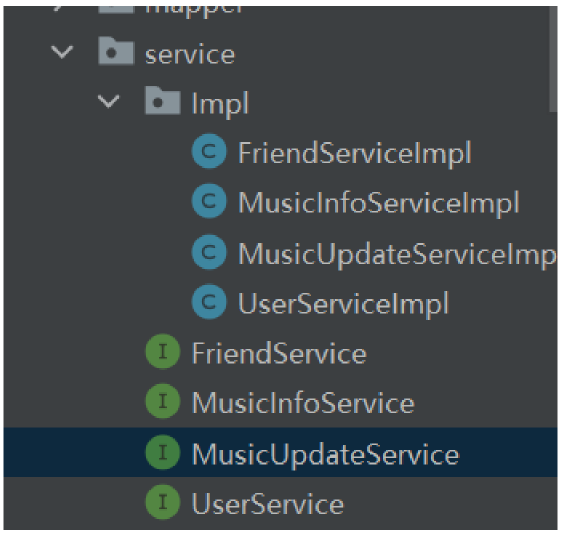
```
        //如果传了 generatePath 则直接调用上传数据库
        if(generatePath!=null){
            return
musicUpdateService.uploadMusic(id,name,description,ty
pe,

permission,photoPath,StringUtils.substringAfter(gener
atePath, "/"));
        }

        String originalFilename2 =
music.getOriginalFilename();
        String savePath = UUID.randomUUID().toString()
+ "."
                +
StringUtils.substringAfter(originalFilename2, ".");
```

```
        //上传到七牛云服务器
        upload = qiniuUtils.upload(music,
musicRemoteStorageParentPath+savePath);
        if (!upload){
            return
Result.fail(ErrorCode.UPLOAD_MUSIC_ERROR.getCode(),

ErrorCode.UPLOAD_MUSIC_ERROR.getMsg());
        }
```

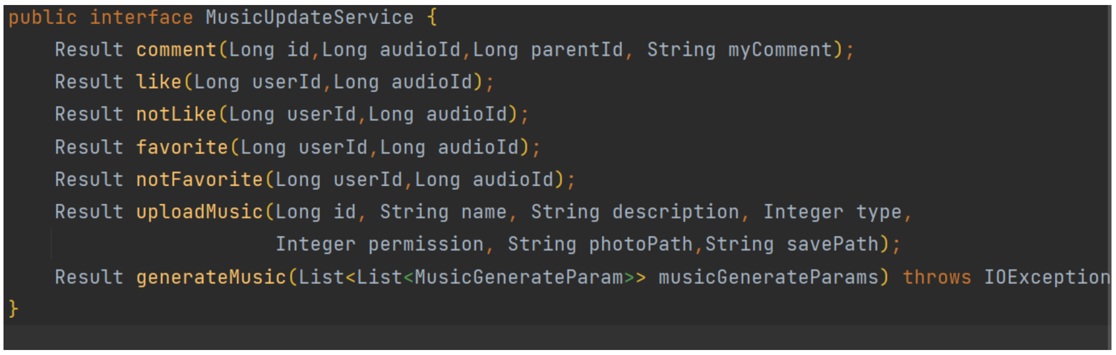
```
        //如果是短视频，上传到阿里云服务器
        FileOutputStream bos = null;
        try {
            bos = new
FileOutputStream(musicLocalStorageParentPath+savePath
);
            bos.write(music.getBytes());
            bos.flush();
            bos.close();
        } catch (IOException e) {
            e.printStackTrace();
            return
Result.fail(ErrorCode.UPLOAD_MUSIC_ERROR.getCode(),

ErrorCode.UPLOAD_MUSIC_ERROR.getMsg());
        }
        return
musicUpdateService.uploadMusic(id,name,description,ty
pe,
                permission,photoPath,savePath);
    }
```

3. service 层设计

service 层负责具体服务的实现，负责调用 mapper 层操作数据库并为 controllor 层提供服务，其中 service 层每个类都有接口类和实现类，更符合设计的解耦性：



musicupdateService 接口如下：



```java
public interface MusicUpdateService {
    Result comment(Long id,Long audioId,Long parentId, String myComment);
    Result like(Long userId,Long audioId);
    Result notLike(Long userId,Long audioId);
    Result favorite(Long userId,Long audioId);
    Result notFavorite(Long userId,Long audioId);
    Result uploadMusic(Long id, String name, String description, Integer type,
                       Integer permission, String photoPath,String savePath);
    Result generateMusic(List<List<MusicGenerateParam>> musicGenerateParams) throws IOException
}
```

其中 service 服务大多是根据参数查询后设置返回字段，或者根据参数保存数据库，查询和保存使用的是 mapper 层操作数据库，其中上传音频的服务如下：

```java
    public Result uploadMusic(Long id, String name, String description, Integer type,
                    Integer permission, String photoPath,String savePath){
    musicUpload musicUpload = new musicUpload();
    musicUpload.setDescription(description);
    musicUpload.setUpUserId(id);
    musicUpload.setName(name);
    musicUpload.setPermission(permission);
    musicUpload.setPhotoPath(photoPath);
    musicUpload.setSavePath(savePath);
    musicUpload.setType(type);
    musicUpload.setUpTime(new Date());

    //获取音频时间

musicUpload.setDuration(MusicUtils.getMp3Duration(musicLocalStorageParentPath+savePath));
    musicUploadMapper.insert(musicUpload);

    MusicSavaReturn musicSavaReturn = new MusicSavaReturn();

musicSavaReturn.setDuration(musicUpload.getDuration());
    musicSavaReturn.setAudioId(musicUpload.getId());

musicSavaReturn.setAudioTime(musicUpload.getUpTime());
musicSavaReturn.setAudioImg(musicImgParentPath+photoPath);

musicSavaReturn.setAudioUrl(qiniuUtils.url+musicRemoteStorageParentPath+savePath);
    //return
    return Result.success(musicSavaReturn);
}
```

其中生成音频服务较为复杂，要调用音频处理工具类进行音频合成，中间涉及到各种临时音频文件的生成，用完之后删除临时文件，再把生成的音频存入服务器，代码如下：

```
    public Result
generateMusic(List<List<MusicGenerateParam>>
musicGenerateParamLists) throws IOException {
```

```
    //先得到最长音频时间

    Double maxTime = 0d;
    for (List<MusicGenerateParam>
musicGenerateParams : musicGenerateParamLists) {
        MusicGenerateParam musicGenerateParam =
musicGenerateParams.get(musicGenerateParams.size() -
1);
        double time =
musicGenerateParam.getStartTime() +
musicGenerateParam.getDuration();
        if(time>maxTime) maxTime=time;
    }
    String concatTemPath =
musicLocalStorageParentPath+UUID.randomUUID();
    Path path = Paths.get(concatTemPath);
    Files.createDirectories(path);

    Path path2 = Paths.get(concatTemPath+"/tem");
    Files.createDirectories(path2);

    ArrayList<String> temfilePaths = new
ArrayList<>();
```

```
    //再对每个音频做 concat,对中间结果缓存

    int i=0;
    for (List<MusicGenerateParam>
```

```java
musicGenerateParamList : musicGenerateParamLists) {
        ArrayList<String> filePaths = new
ArrayList<>();
        ArrayList<Double> startTime = new
ArrayList<>();
        ArrayList<Double> duration = new
ArrayList<>();
        for (MusicGenerateParam musicGenerateParam :
musicGenerateParamList) {

filePaths.add(musicGenerateParam.getSavePath());

startTime.add(musicGenerateParam.getStartTime());

duration.add(musicGenerateParam.getDuration());
        }
        String outFilePath = concatTemPath+"/" + i +
".mp3";
        temfilePaths.add(outFilePath);
        String temFilePath = concatTemPath + "/tem/";

MusicUtils.concatMp3ListWithNull(filePaths,startTime,
duration,maxTime,
                outFilePath,temFilePath);
        i++;
```

```java
        //删除 tem 文件夹下的临时文件

        File file = new File(concatTemPath+"/tem");
        String[] files = file.list();
        for (String f:files) {
            new File(concatTemPath+"/tem",f).delete();
        }
    }
```

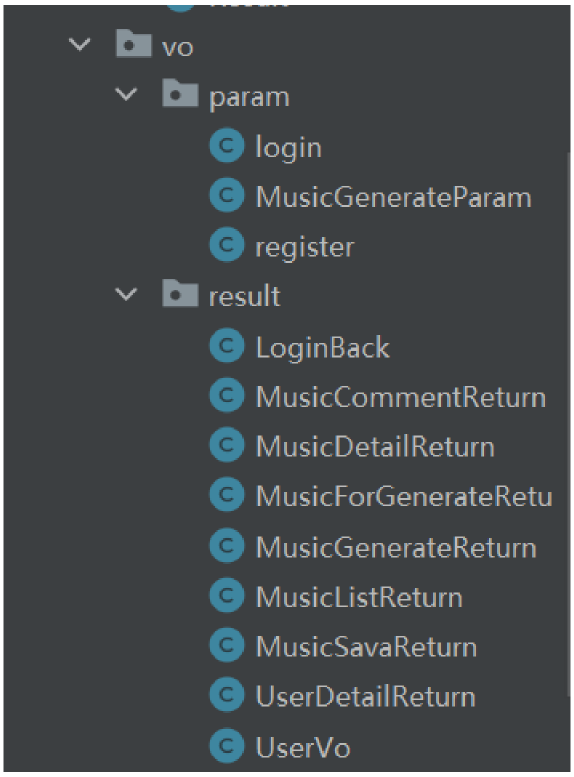
```java
    //删除 tem 文件夹

    new File(concatTemPath+"/tem").delete();

    String savePath = UUID.randomUUID()+".mp3";
```

```java
    //再做混合

MusicUtils.mixMp3List(temfilePaths,musicLocalStorageP
arentPath+savePath);
```

```java
    //中间文件删除,删除 concatpath 的子文件和文件夹

    File file = new File(concatTemPath);
    String[] files = file.list();
    if(files.length>0){
        for (String f:files) {
            new File(concatTemPath,f).delete();
        }
        //在删除全部文件后，将上层目录删除
        new File(file.getPath()).delete();
    }else{
        //若文件夹为空，则只删除上层文件夹
        new File(file.getPath()).delete();
    }
    //把音频存进七牛云
    boolean b =
qiniuUtils.uploadLocal(musicLocalStorageParentPath +
savePath,
            musicRemoteStorageParentPath + savePath);
    //如果失败返回失败
    if(!b){
        return
Result.fail(ErrorCode.UPLOAD_MUSIC_ERROR.getCode(),

ErrorCode.UPLOAD_MUSIC_PHOTO_ERROR.getMsg());
    }
    //暂时不存数据库
    //返回基本信息
    MusicGenerateReturn musicGenerateReturn = new
MusicGenerateReturn();

musicGenerateReturn.setUrl(qiniuUtils.url+musicRemote
StorageParentPath+savePath);
    musicGenerateReturn.setDuration(maxTime);
    return Result.success(musicGenerateReturn);
}
```

4. vo 层设计

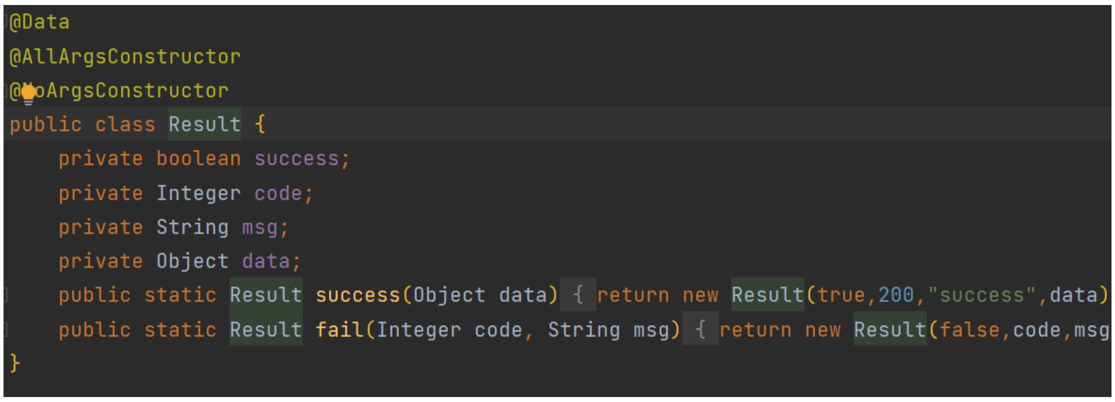
vo 层主要用于前后端交互的数据，较为简单，架构如下：



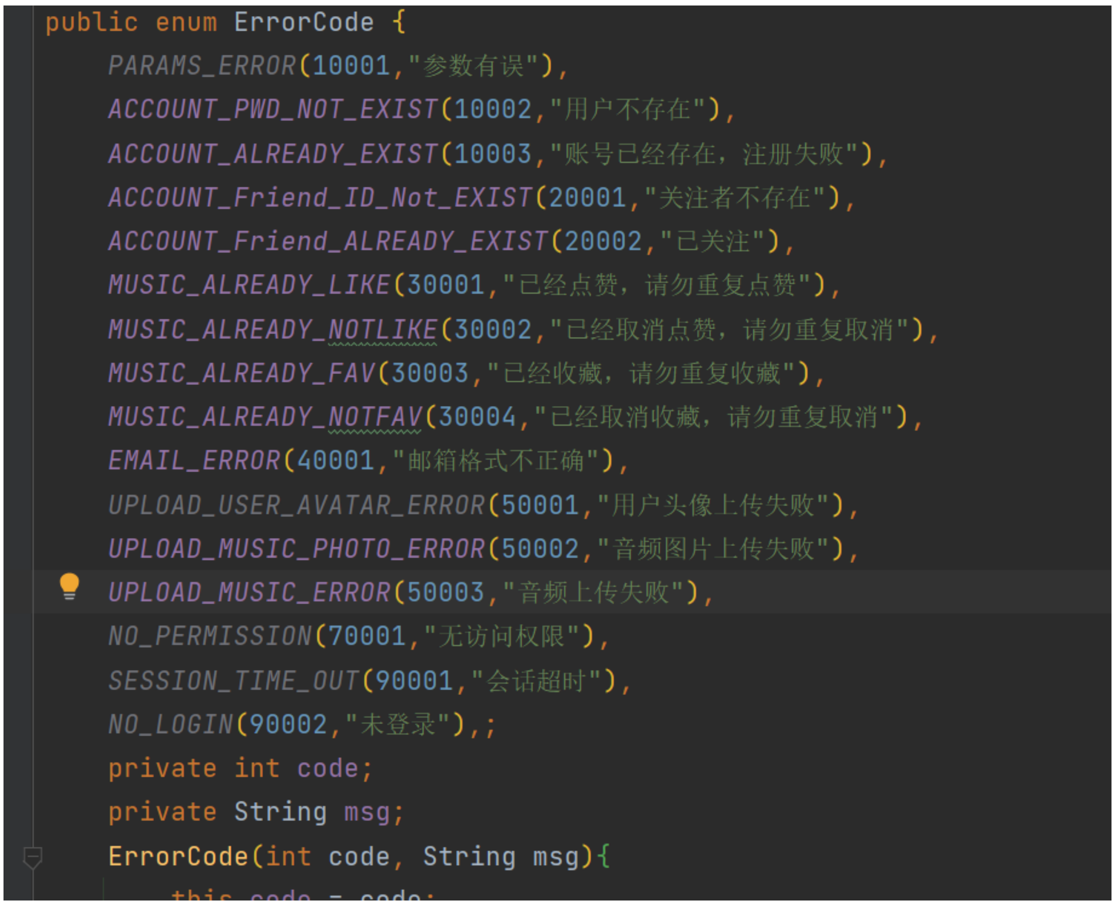
```
vo
    param
        login
        MusicGenerateParam
        register
    result
        LoginBack
        MusicCommentReturn
        MusicDetailReturn
        MusicForGenerateRetu
        MusicGenerateReturn
        MusicListReturn
        MusicSavaReturn
        UserDetailReturn
        UserVo
```

5. utils 包

Uitils 包包含了各种工具类，其中 qiniu 云工具类是官方 copy 得到不做介绍，result 类代表了统一的返回格式包括状态码、message、data，errorcode 代表了各种错误码和 message 的对应关系，如下：

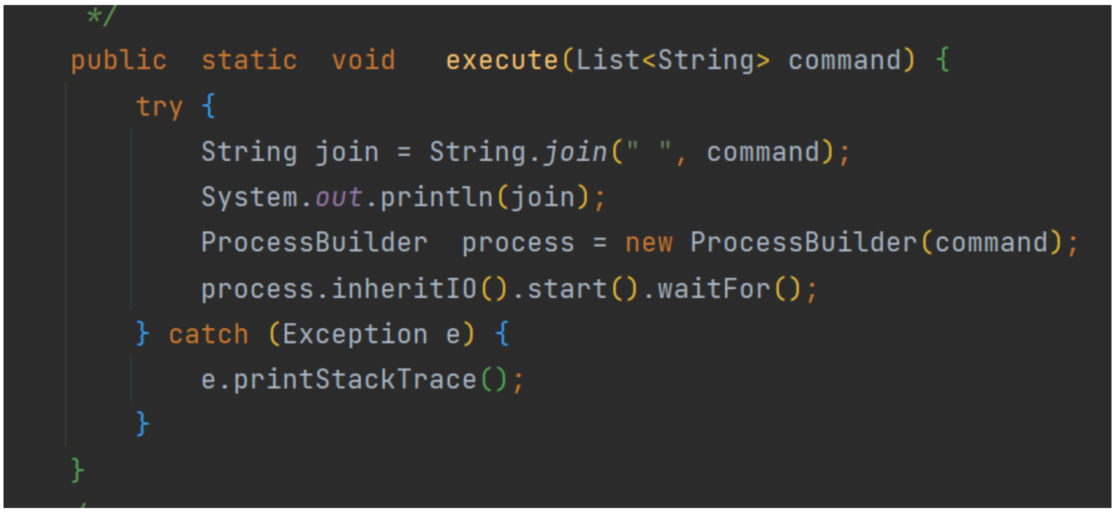


```java
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Result {
    private boolean success;
    private Integer code;
    private String msg;
    private Object data;
    public static Result success(Object data) { return new Result(true,200,"success",data)
    public static Result fail(Integer code, String msg) { return new Result(false,code,msg
}
```



```java
public enum ErrorCode {
    PARAMS_ERROR(10001,"参数有误"),
    ACCOUNT_PWD_NOT_EXIST(10002,"用户不存在"),
    ACCOUNT_ALREADY_EXIST(10003,"账号已经存在，注册失败"),
    ACCOUNT_Friend_ID_Not_EXIST(20001,"关注者不存在"),
    ACCOUNT_Friend_ALREADY_EXIST(20002,"已关注"),
    MUSIC_ALREADY_LIKE(30001,"已经点赞，请勿重复点赞"),
    MUSIC_ALREADY_NOTLIKE(30002,"已经取消点赞，请勿重复取消"),
    MUSIC_ALREADY_FAV(30003,"已经收藏，请勿重复收藏"),
    MUSIC_ALREADY_NOTFAV(30004,"已经取消收藏，请勿重复取消"),
    EMAIL_ERROR(40001,"邮箱格式不正确"),
    UPLOAD_USER_AVATAR_ERROR(50001,"用户头像上传失败"),
    UPLOAD_MUSIC_PHOTO_ERROR(50002,"音频图片上传失败"),
    UPLOAD_MUSIC_ERROR(50003,"音频上传失败"),
    NO_PERMISSION(70001,"无访问权限"),
    SESSION_TIME_OUT(90001,"会话超时"),
    NO_LOGIN(90002,"未登录"),;
    private int code;
    private String msg;
    ErrorCode(int code, String msg){
        this.code = code;
```

而音频处理工具类较为复杂，主要是利用 ffmpeg 处理音频，excuate 执行函数将传入参数合并成 command 后开新进程执行 ffmpeg，如下：



```java
 */
public  static  void   execute(List<String> command) {
    try {
        String join = String.join(" ", command);
        System.out.println(join);
        ProcessBuilder  process = new ProcessBuilder(command);
        process.inheritIO().start().waitFor();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

因此其他函数主要就是得到这个命令的 list 调用 execuate 即可，但获取音频信息的函数还涉及到输出重定向和正则表达式匹配，利用输出重定向到变量中，利用正则表达式找到其中代表时间的部分再将时间字符串转换成秒，函数如

下:

```
    public static Double getMp3Duration(String
filePath) {
    List<String> command = new ArrayList<>();
    //获取 JavaCV 中的 ffmpeg 本地库的调用路径
    String ffmpeg =
Loader.load(org.bytedeco.ffmpeg.ffmpeg.class);
    command.add(ffmpeg);
    command.add("-i");
    command.add(filePath);
    //执行并获得命令行输出
    String join = String.join(" ", command);
    System.out.println(join);
    ProcessBuilder  process = new
ProcessBuilder(command);
    //输出重定向
    process.redirectErrorStream(true);
    try {
        Process p= process.start();
        BufferedReader buf = null;
        String line = null;
        buf = new BufferedReader(new
InputStreamReader(p.getInputStream()));
        StringBuilder sb= new StringBuilder();
        while ((line = buf.readLine()) != null) {
            System.out.println(line);
            sb.append(line);
        }
        p.waitFor();
        String out = sb.toString();
        //System.out.println("输出: "+sb);
        //正则匹配找到时间
        Pattern pattern =
Pattern.compile("(?i)Duration:\\s*(\\d+\\s*):(\\d+\\s
*):(\\d+\\s*.\\d+\\s*)");
        Matcher matcher = pattern.matcher(out);
        if(matcher.find()){
            //System.out.println("匹配字符串:
```

```
"+matcher.group(0));
            String hour = matcher.group(1);
            String minute = matcher.group(2);
            String second = matcher.group(3);
            return
(Double.parseDouble(hour)*60+Double.parseDouble(minut
e))*60
                    + Double.parseDouble(second);
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return -1.1;
}
```

6. 部署

后端设计部分如上所述，部署部分首先使用 maven 打包项目，将 jar 包传入服务器，利用 java -jar xxx.jar 命令运行 jar 包即可启动后端服务，注意服务器要安装 java11 和 ffmpeg