IMusic 在线音乐播放系统说明文档

一、项目背景及意义

1.1 项目背景

随着互联网技术的快速发展和移动设备的普及，在线音乐服务已成为人们日常生活中不可或缺的一部分。传统的本地音乐播放器已无法满足用户随时随地听歌的需求，而现有的商业音乐平台往往存在版权限制、广告干扰等问题。

IMusic 项目旨在构建一个轻量级、易部署的个人音乐播放平台，让用户能够管理和播放自己的音乐收藏，同时提供现代化的 Web 界面和流畅的用户体验。

1.2 项目意义

1. **个人音乐管理**：为用户提供一个统一的平台来管理个人音乐收藏
2. **跨设备访问**：基于 Web 技术，用户可以在任何设备上通过浏览器访问音乐库
3. **技术学习价值**：项目采用微服务架构，涵盖现代 Web 开发的核心技术栈
4. **隐私保护**：用户可以自行部署，音乐数据完全由自己掌控

二、项目功能介绍

2.1 核心功能

功能模块	功能描述
用户管理	用户注册、登录、会话管理
歌曲搜索	按歌曲名称或艺术家搜索音乐
在线播放	支持音乐在线播放、暂停、进度控制、音量调节
收藏管理	添加/移除收藏歌曲
歌单管理	创建和管理个人歌单

2.2 功能特点

- **实时搜索**：快速搜索音乐库中的歌曲
- **播放控制**：支持上一曲/下一曲切换、进度拖拽、音量调节
- **播放动画**：专辑封面随播放状态旋转
- **持久登录**：支持记住登录状态，自动恢复会话
- **响应式设计**：适配不同屏幕尺寸的设备

三、技术栈及架构

3.1 技术栈概览

层级	技术
前端	React 19 + Vite 7
后端	Spring Boot 3.2 + Spring Cloud Gateway
数据库	MySQL 8.0
ORM	Spring Data JPA + Hibernate

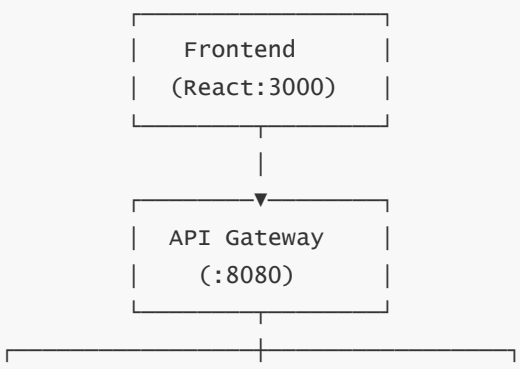
3.2 前端技术栈

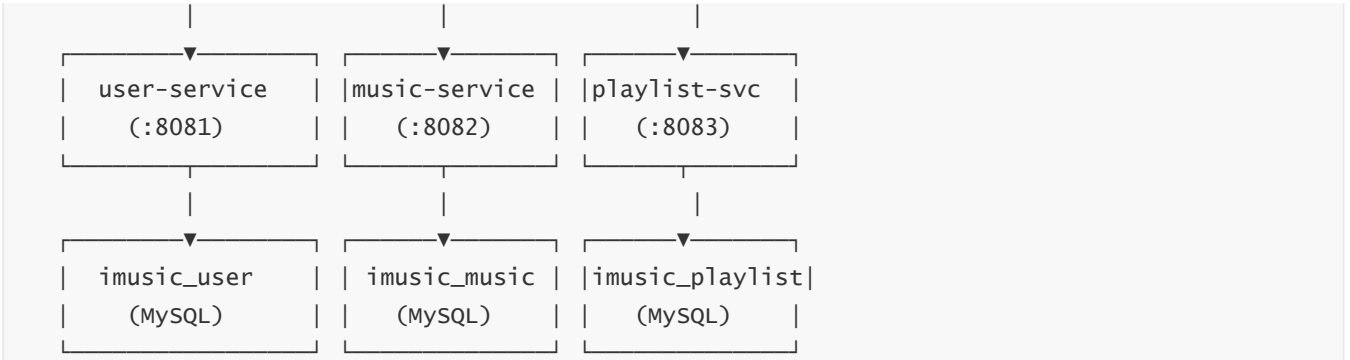
技术	版本	用途
React	19.2.0	前端 UI 框架，构建用户界面
Vite	7.1.10	前端构建工具，提供快速的开发体验
CSS3	-	样式设计，实现响应式布局

3.3 后端技术栈

技术	版本	用途
Java	17	后端开发语言
Spring Boot	3.2	后端应用框架
Spring Cloud Gateway	-	API 网关
Spring Data JPA	-	数据持久层框架
MySQL	8.0	关系型数据库
Maven	-	项目构建和依赖管理

3.4 微服务架构图





3.5 微服务说明

服务	端口	数据库	功能
gateway-service	8080	-	API 网关、路由转发、CORS
user-service	8081	imusic_user	用户注册、登录、验证
music-service	8082	imusic_music	歌曲搜索、播放、资源服务
playlist-service	8083	imusic_playlist	歌单管理、收藏功能

3.6 请求处理流程

```
HTTP Request → Gateway → 路由转发 → 对应微服务 → Service → Repository → MySQL
                                                    ↓
                                                ApiResponse
                                                    ↓
HTTP Response ← Gateway ← JSON 序列化 ←——┘
```

四、数据库设计

4.1 数据库选型

项目采用 **MySQL 8.0** 作为数据存储方案，每个微服务独立数据库：

- **imusic_user**：用户服务数据库
- **imusic_music**：音乐服务数据库
- **imusic_playlist**：歌单服务数据库

4.2 数据表设计

4.2.1 用户表 (users) - imusic_user

字段名	数据类型	约束	说明
id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	用户ID

字段名	数据类型	约束	说明
username	VARCHAR	NOT NULL, UNIQUE	用户名
password	VARCHAR	NOT NULL	密码

4.2.2 歌曲表 (songs) - imusic_music

字段名	数据类型	约束	说明
id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	歌曲ID
file_path	VARCHAR	NOT NULL	音频文件路径
cover_file_path	VARCHAR	NOT NULL	封面图片路径
title	VARCHAR	NOT NULL	歌曲标题
artist	VARCHAR	NOT NULL	歌手
album	VARCHAR	NOT NULL	专辑名称
album_artist	VARCHAR	NOT NULL	专辑艺术家
year	VARCHAR	NOT NULL	发行年份
genre	VARCHAR	NOT NULL	音乐流派
track_number	VARCHAR	NOT NULL	曲目编号
bitrate	VARCHAR	NOT NULL	比特率
duration	VARCHAR	NOT NULL	时长

4.2.3 歌单表 (song_list) - imusic_playlist

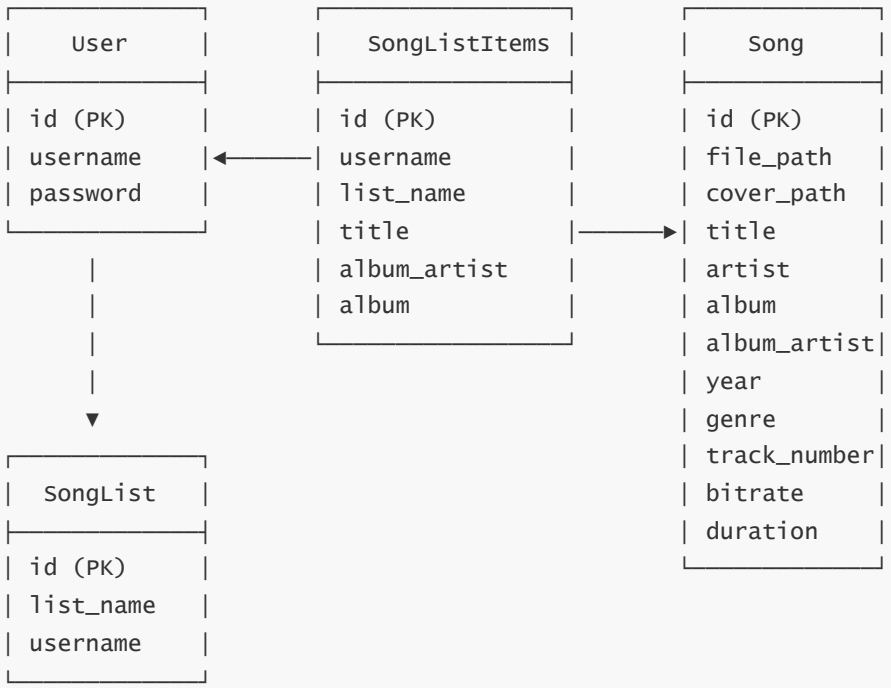
字段名	数据类型	约束	说明
id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	歌单ID
list_name	VARCHAR	NOT NULL	歌单名称
username	VARCHAR	NOT NULL	创建用户

4.2.4 歌单项目表 (song_list_items) - imusic_playlist

字段名	数据类型	约束	说明
id	BIGINT	PRIMARY KEY, AUTO_INCREMENT	记录ID
title	VARCHAR	NOT NULL	歌曲标题
album_artist	VARCHAR	NOT NULL	专辑艺术家

字段名	数据类型	约束	说明
album	VARCHAR	NOT NULL	专辑名称
username	VARCHAR	NOT NULL	所属用户
list_name	VARCHAR	NOT NULL	所属歌单

4.3 ER 图



五、关键技术及解决方案

5.1 微服务架构

技术难点：服务拆分与服务间通信

解决方案：

- 使用 Spring Cloud Gateway 作为 API 网关
- 各服务独立数据库，通过 REST API 通信
- playlist-service 通过 HTTP Client 调用 user-service 和 music-service

```java
// MusicServiceClient.java - 服务间调用
@Component
public class MusicServiceClient {
    private final RestTemplate restTemplate;

    public SongResponse getSongByInfo(String title, String albumArtist, String album) {
        String url = "http://localhost:8082/api/song/get?title={title}&albumArtist=
{albumArtist}&album={album}";
        return restTemplate.getForObject(url, SongResponse.class, title, albumArtist,
album);
    }
}
```

5.2 API 网关配置

技术难点：统一路由和跨域处理

解决方案：

- Gateway 统一处理 CORS
- 根据路径前缀路由到不同服务

```yaml
# application.yml - Gateway 路由配置
spring:
  cloud:
    gateway:
      routes:
        - id: user-service
          uri: http://localhost:8081
          predicates:
            - Path=/api/user/**
        - id: music-service
          uri: http://localhost:8082
          predicates:
            - Path=/api/song/search/all, /api/song/play, /cover/**, /mp3/**
        - id: playlist-service
          uri: http://localhost:8083
          predicates:
            - Path=/api/song/**, /api/user/songLists/**
```

5.3 前端代理配置

技术难点：开发环境跨域问题

解决方案：

- Vite 配置代理，将 API 请求转发到网关

```
// vite.config.js
export default defineConfig({
  server: {
    proxy: {
      '/api': { target: 'http://localhost:8080', changeOrigin: true },
      '/cover': { target: 'http://localhost:8080', changeOrigin: true },
      '/mp3': { target: 'http://localhost:8080', changeOrigin: true }
    }
  }
})
```

5.4 用户认证

技术难点：微服务架构下的用户身份传递

解决方案：

- 前端存储用户凭证到 LocalStorage
- 请求时通过 Header 传递用户信息
- playlist-service 调用 user-service 验证用户

```
// 前端请求携带用户信息
headers: {
  'X-Username': username,
  'X-Password': password
}
```

5.5 播放动画效果

技术难点：播放状态与动画同步

解决方案：

- CSS 动画配合 React 状态控制
- 播放时添加 `playing` 类名启动旋转

```
.player-cover {
  animation: rotate 20s linear infinite;
  animation-play-state: paused;
}
.player-cover.playing {
  animation-play-state: running;
}
```

六、技术点及开发工作

6.1 前端开发工作

组件/模块	功能描述	技术要点
App.jsx	主应用组件	状态管理、路由控制
SearchBar.jsx	搜索栏组件	用户输入处理
SongList.jsx	歌曲列表组件	列表渲染、事件处理
Player.jsx	播放器组件	音频控制、进度条、旋转动画
Favorites.jsx	收藏列表组件	收藏管理
Login.jsx	登录组件	表单验证、API调用
Toast.jsx	提示组件	消息通知
LoadingOverlay.jsx	加载遮罩	加载状态展示

6.2 后端微服务开发工作

gateway-service（API 网关）

类名	功能描述
GatewayServiceApplication	网关启动类
application.yml	路由配置、CORS 配置

user-service（用户服务）

模块	类名	功能描述
Controller	UserController	用户登录、注册、验证
Service	UserService	用户业务逻辑
Repository	UserRepository	用户数据访问
Entity	User	用户实体类
DTO	LoginRequest	登录请求
DTO	ApiResponse	统一响应格式
Config	WebConfig	CORS 配置

music-service（音乐服务）

模块	类名	功能描述
Controller	SongController	歌曲搜索、播放
Service	SongService	歌曲业务逻辑
Repository	SongRepository	歌曲数据访问
Entity	Song	歌曲实体类
DTO	SongRequest	歌曲请求
DTO	SongResponse	歌曲响应
Config	WebConfig	静态资源配置

playlist-service（歌单服务）

模块	类名	功能描述
Controller	PlaylistController	歌单管理、收藏管理
Service	SongListService	歌单业务逻辑
Service	SongListItemsService	歌单项目业务逻辑
Repository	SongListRepository	歌单数据访问
Repository	SongListItemsRepository	歌单项目数据访问
Entity	SongList	歌单实体类
Entity	SongListItems	歌单项目实体类
Client	MusicServiceClient	音乐服务调用
Client	UserServiceClient	用户服务调用

6.3 API 接口列表

用户服务 (user-service)

接口路径	方法	功能描述
/api/user/register	POST	用户注册
/api/user/login/password	POST	用户登录
/api/user/validate	POST	验证用户

音乐服务 (music-service)

接口路径	方法	功能描述
/api/song/search/all	POST	搜索歌曲
/api/song/play	POST	播放歌曲
/api/song/get	GET	获取歌曲详情
/cover/**	GET	封面图片资源
/mp3/**	GET	音频文件资源

歌单服务 (playlist-service)

接口路径	方法	功能描述
/api/user/songLists/show	POST	获取用户歌单
/api/user/songLists/add	POST	创建歌单
/api/song/search/insonglist	POST	查询歌单中的歌曲
/api/song/add/tosonglist	POST	添加歌曲到歌单
/api/song/delete/fromsonglist	POST	从歌单删除歌曲

七、项目目录结构

```
imusic/
├── gateway-service/                    # API 网关服务
|   ├── pom.xml
|   └── src/main/
|       ├── java/.../gateway/
|       |   └── GatewayServiceApplication.java
|       └── resources/
|           └── application.yml         # 路由配置
|
├── user-service/                       # 用户服务
|   ├── pom.xml
|   └── src/main/java/.../user/
|       ├── UserServiceApplication.java
|       ├── controller/UserController.java
|       ├── service/UserService.java
|       ├── repository/UserRepository.java
|       ├── entity/User.java
|       ├── dto/
|       └── config/WebConfig.java
|
├── music-service/                      # 音乐服务
|   ├── pom.xml
```

```
|   └── src/main/
|       ├── java/.../music/
|       |   ├── MusicServiceApplication.java
|       |   ├── controller/SongController.java
|       |   ├── service/SongService.java
|       |   ├── repository/SongRepository.java
|       |   ├── entity/Song.java
|       |   ├── dto/
|       |   └── config/WebConfig.java
|       └── resources/
|           ├── music_info.json        # 歌曲元数据
|           ├── cover/                 # 封面图片
|           └── mp3/                   # 音频文件
|
├── playlist-service/                  # 歌单服务
|   ├── pom.xml
|   └── src/main/java/.../playlist/
|       ├── PlaylistServiceApplication.java
|       ├── controller/PlaylistController.java
|       ├── service/
|       |   ├── SongListService.java
|       |   └── SongListItemsService.java
|       ├── repository/
|       ├── entity/
|       ├── dto/
|       ├── client/                    # 服务间调用
|       |   ├── MusicServiceClient.java
|       |   └── UserServiceClient.java
|       └── config/WebConfig.java
|
├── imusic-frontend/                   # 前端项目
|   ├── package.json
|   ├── vite.config.js                 # Vite 配置（含代理）
|   ├── index.html
|   └── src/
|       ├── main.jsx
|       ├── App.jsx
|       ├── config.js
|       └── components/
|
├── logs/                              # 服务日志目录
├── start.sh                           # 一键启动脚本
├── stop.sh                            # 停止脚本
└── README.md
```

八、部署说明

8.1 环境要求

- **后端**：JDK 17+、Maven 3.6+、MySQL 8.0+
- **前端**：Node.js 18+、npm

8.2 一键启动

```
# 启动所有服务（需要 sudo 密码启动 MySQL）
./start.sh

# 停止所有服务
./stop.sh
```

启动脚本会自动：

1. 检查并启动 MySQL 服务
2. 创建数据库和用户
3. 启动 4 个后端微服务
4. 启动前端开发服务器

8.3 手动启动

```
# 1. 启动后端微服务（按顺序）
cd user-service && mvn spring-boot:run &
cd music-service && mvn spring-boot:run &
cd playlist-service && mvn spring-boot:run &
cd gateway-service && mvn spring-boot:run &

# 2. 启动前端
cd imusic-frontend
npm install
npm run dev
```

8.4 访问地址

- 前端：http://localhost:3000
- API 网关：http://localhost:8080

8.5 数据库配置

MySQL 用户：`imusic` / `imusic123`

```
-- 数据库
imusic_user       -- 用户数据
imusic_music      -- 歌曲数据
imusic_playlist   -- 歌单数据
```

8.6 日志查看

```
tail -f logs/user-service.log
tail -f logs/music-service.log
tail -f logs/playlist-service.log
tail -f logs/gateway-service.log
tail -f logs/frontend.log
```

九、源代码

GitHub 仓库地址：https://github.com/MagicalFlames/i_music

十、总结

IMusic 是一个完整的微服务架构的在线音乐播放系统，具有以下特点：

1. **微服务架构**：采用 Spring Cloud Gateway + 多服务拆分，职责清晰
2. **技术先进**：使用 React 19 + Spring Boot 3 + MySQL 8 等最新技术
3. **功能完善**：涵盖用户管理、歌曲搜索、在线播放、收藏管理等核心功能
4. **服务独立**：各服务独立数据库，可独立部署和扩展
5. **一键部署**：提供启动脚本，自动初始化数据库和启动服务

签名