<SJTU>

天天爱 K 歌
软件架构文档

版本 **1.1**

修订历史记录

日期	版本	说明	作者
4/5/2021	1.0	初稿	曹沅欣、江雨泽、郭志东
4/5/2021	1.1	根据项目的实际情况修正	江雨泽

目录

软件架构文档

1. 简介

1.1 目的

本文档将从构架方面对系统进行综合概述，其中会使用多种不同的构架视图来描述系统的各个方面。它用于记录并表述已对系统的构架方面作出的重要决策。系统的开发者可以通过本文档快速了解系统的主要架构，从而加快开发进度、提高开发质量。

1.2 参考资料

无参考资料

2. 用例视图

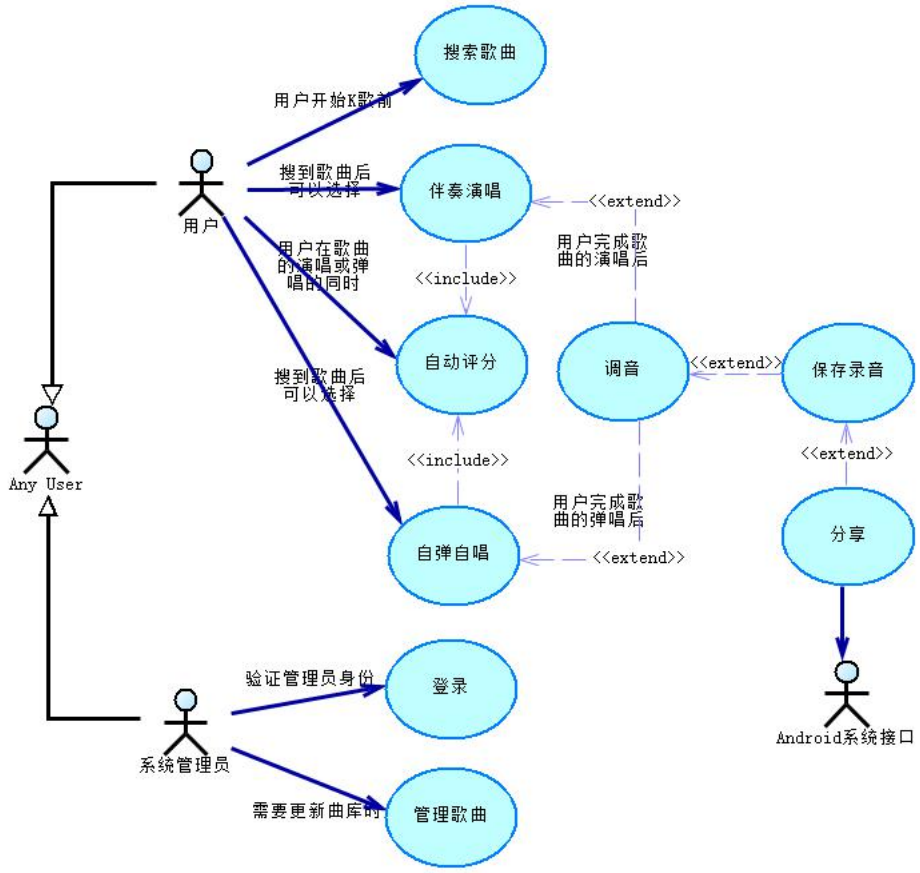

图 1 用例视图

本项目的参与者有 3 个——系统管理员、用户与 Android 系统接口。系统管理员可以登录本项目的管理员系统，并对歌曲信息、文件进行维护。普通用户想要开始 K 歌，既可以直接在歌曲列表中选择歌曲，也可以使用“搜索歌曲”功能选择自己想唱的歌曲。K 歌分为两种模式：伴奏演唱与自弹自唱，无论在哪种模式下，系统都会自动对用户的演唱（弹奏）进行自动评分。在用户演唱（弹奏）结束后，用户可以选择是否通过系统接口保存并分享作品。

3. 逻辑视图

3.1 概述

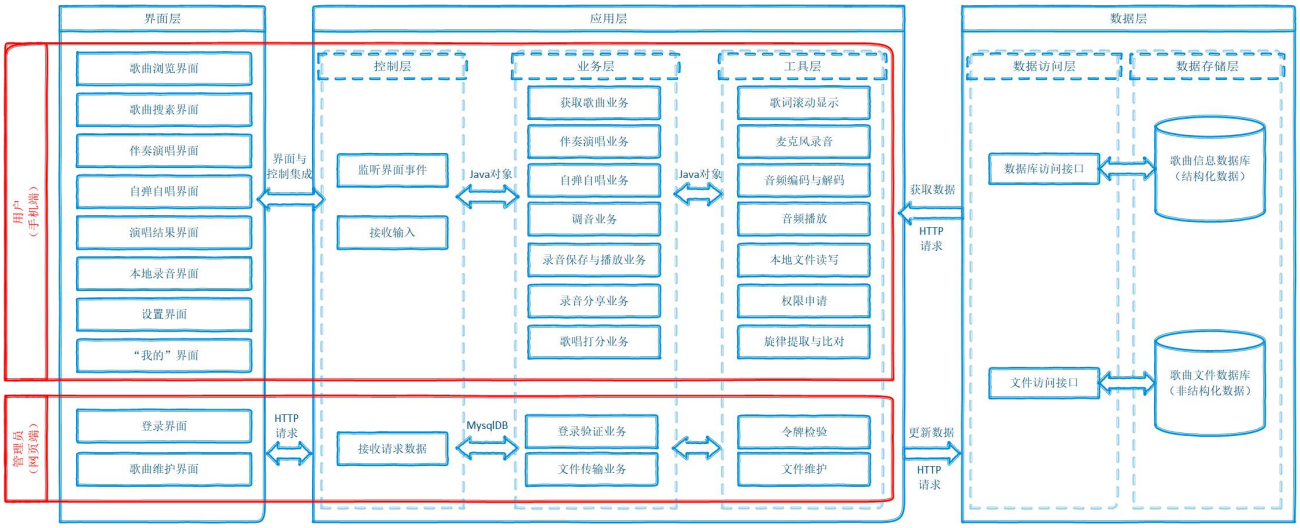

图 2 逻辑视图

本项目主要的逻辑架构为 3 Tiers 架构。界面层和应用层又分为用户端和管理员端，用户端用 Android Studio+Java 进行开发，部署在用户的手机上；管理员端使用 Vue+Flask 开发，运行在网页浏览器上。各层的功能如图所示。

应用层内部的业务层和工具层采用分层架构，根据抽象程度进行分层，将具体业务与可复用的服务进行分离，增加项目的可维护性。

3.2 在构架方面具有重要意义的设计包

本项目中最重要的包即应用层中的工具层。该包中实现了核心业务所要求的歌词滚动显示、音频编码与解码、旋律提取与比对等复杂功能。用户端 APP 的多个界面都会调用工具层中的方法。

4. 部署视图

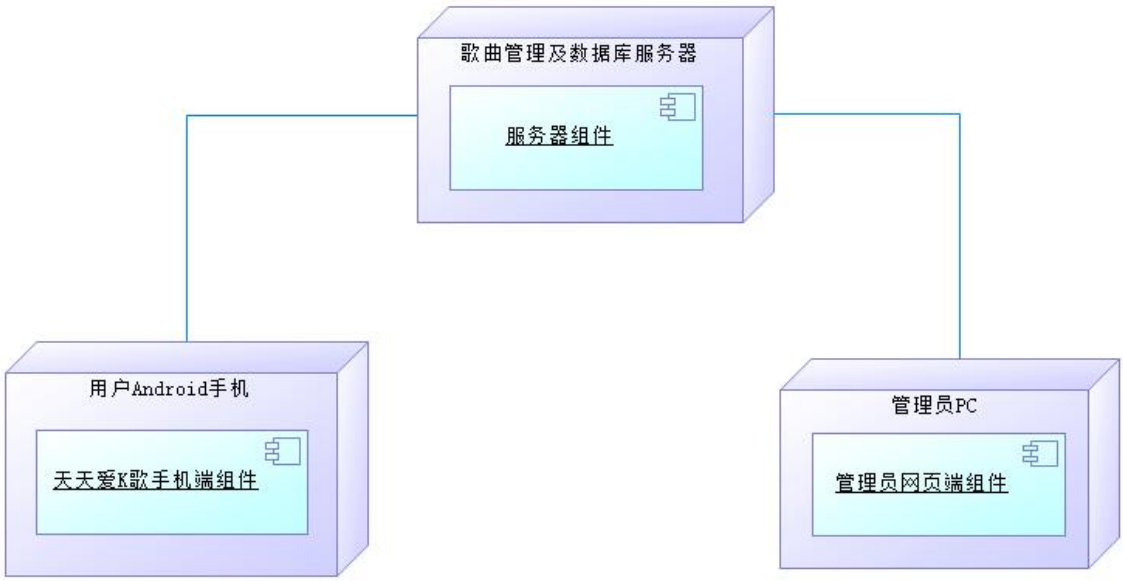

图 3 部署视图

本项目的物理架构为 CS、BS 混合架构。用户通过手机端“天天爱 K 歌”APP 使用 K 歌相关功能，歌曲信息从部署在歌曲管理及数据库服务器上的歌曲管理系统后端获取；管理员通过 PC 上的网页浏览器登录管理员系统，并通过管理员系统的网页端组件与服务器交互，维护歌曲信息。

5. 进程视图

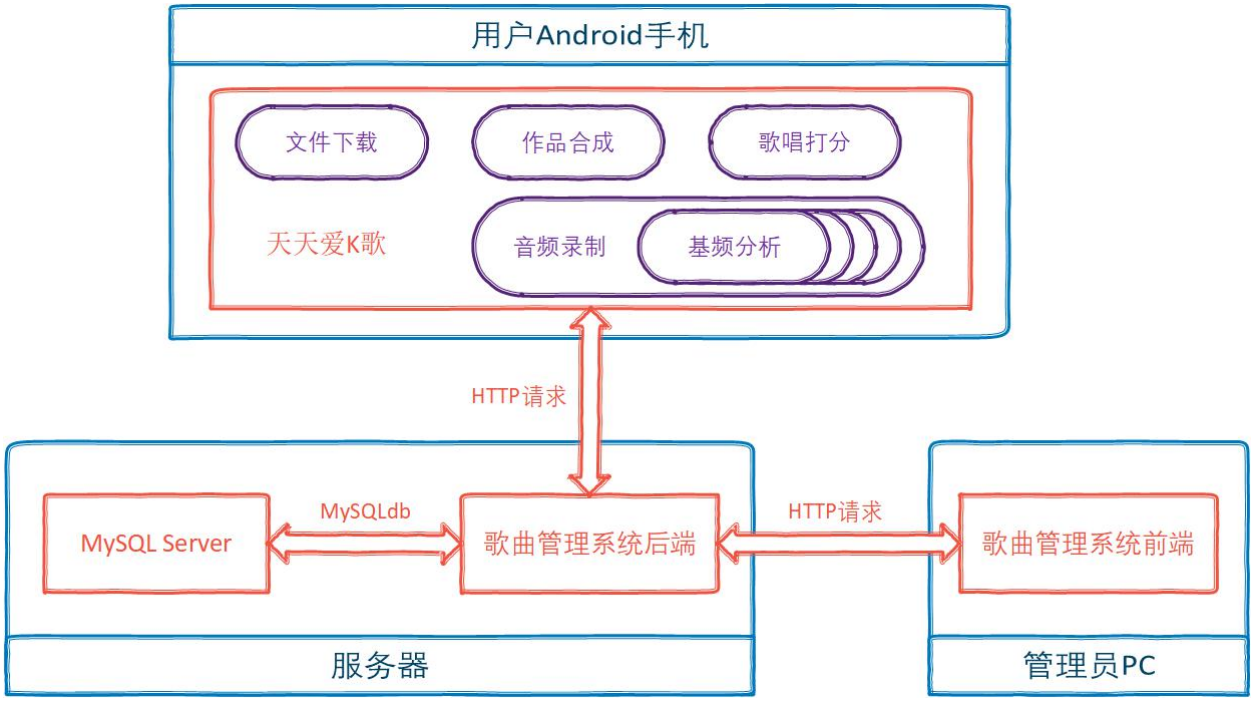

图 4 进程视图

本项目最多有 4 个进程（红色框）同时运行在 3 个不同的节点（蓝色框）上。

本项目的主要进程——“天天爱 K 歌”手机 APP 运行在用户的 Android 手机上，该进程包含了多个线程（紫色框），每个线程的职责如图所示。在“音频录制”线程中，又包括了多个“基频分析”线程，因为打分算法需要将用户录音切分成多段时长较短的音频并对其进行基频分析，而基频分析计算量大，需要为每一次基频分析都分配一个新的线程，才能尽快完成一整句话的基频分析，降低打分算法的延迟。该进程通过 HTTP 请求与服务器上的管理员系统后端通信。

管理员系统的前端界面运行在管理员的计算机上，采用单线程设计，同样通过 HTTP 请求与服务器上的管理员系统后端通信。

歌曲管理系统后端和 MySQL server 必须在服务器上保持运行状态，否则手机 APP 无法通过 HTTP 请求正常获取歌曲信息。歌曲管理系统后端采用单线程设计，在接收到 HTTP 请求后通过 MySQLdb 与 MySQL server 通信，获取数据后以 HTTP Response 的形式将数据返回。

6. 实现视图

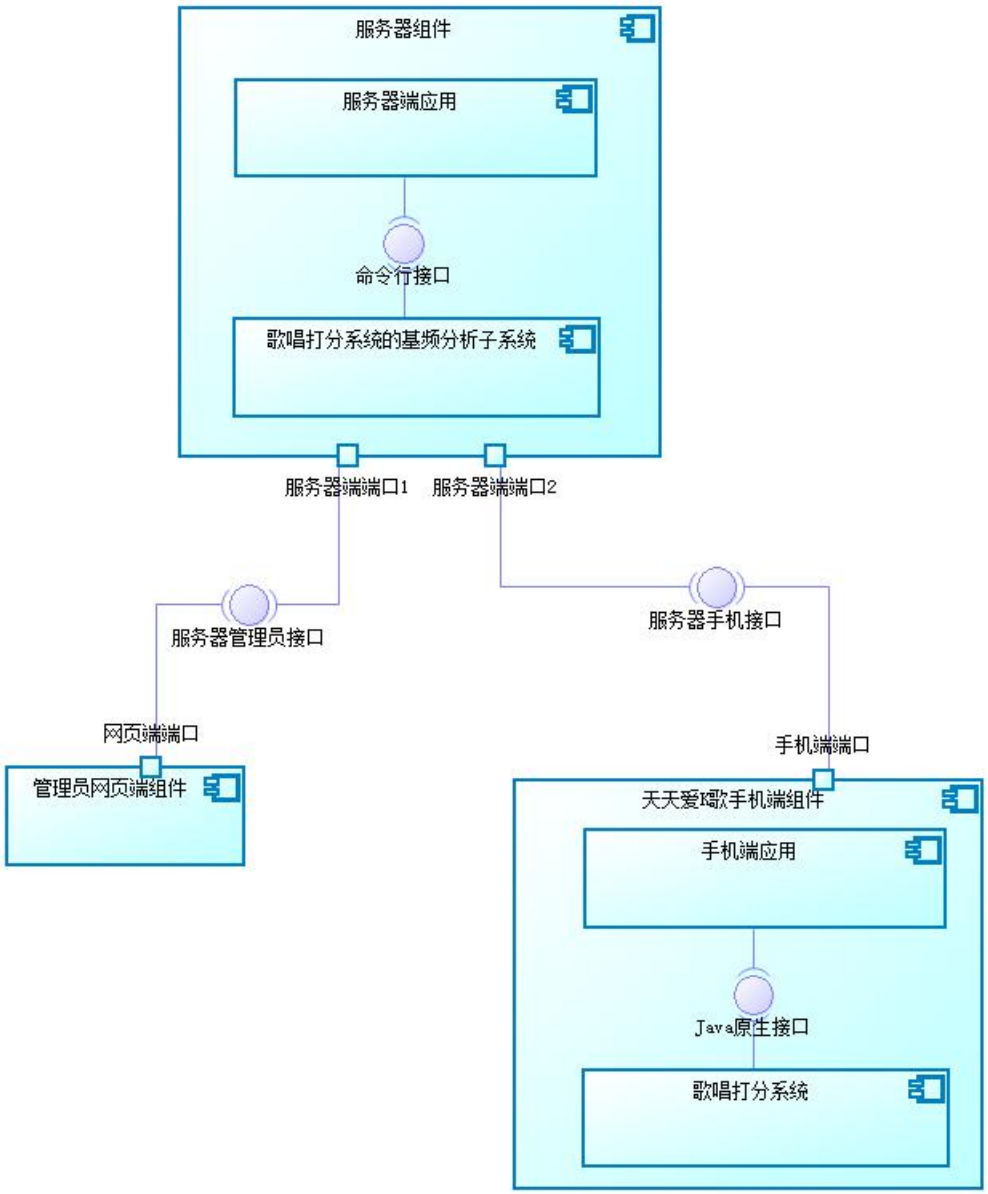

图 5 实现视图

本项目的实现分为服务器端，手机端和管理员网页端。手机端是普通用户使用天天爱 K 歌的主要程序，其实现分为手机端应用和歌唱打分系统两部分进行，通过 Java 原生接口进行通信。服务器端是承担响应系统服务请求、承担服务、保障服务的主要载体，其实现分为服务器端应用和歌唱打分系统的基频分析子系统两部分进行，通过命令行接口进行通信。管理员网页端实现系统管理员管理在线曲库的功能。管理员网页端与服务器端通过服务器管理员接口进行通信，手机端与服务器端通过服务器手机接口进行通信。

7. 技术视图

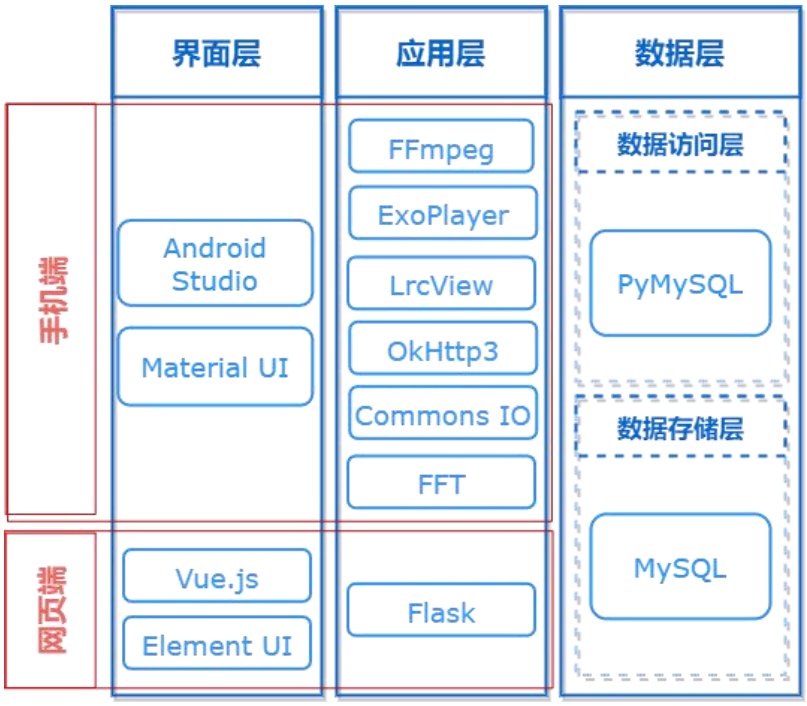

图 6 技术视图

技术选型	用途
Android Studio	“天天爱 K 歌”手机端 APP 开发平台
Material UI	“天天爱 K 歌”手机端 APP 组件库
FFmpeg	音频混合、截取工具
ExoPlayer	手机端音频、视频播放器
LrcView	歌词高亮显示组件
OkHttp3	手机端 Ajax 库
Commons IO	手机端文件操作库
FFT	快速傅里叶变换数学库
Vue.js	网页端管理员系统前端框架
Element UI	网页端管理员系统前端组件库
Flask	管理员系统后端框架，从管理员系统前端接收文件并储存至服务器，并向手机端 APP 发送文件
PyMySQL	提供 Flask 与 MySQL 数据库的接口，用于从数据库中存取数据
MySQL	关系型数据库，在本项目中用于存储歌曲信息

8. 核心算法设计

下面我们将通过伪代码对该系统中歌曲演唱实时打分子系统中针对音准的打分算法进行设计。

```
for (枚举录音延迟)//消除录音延迟对评分的影响
{
    从硬盘中载入用户演唱的基频分析结果文件到userF0;
    从硬盘中载入原唱的基频分析结果文件到originF0;
    取出userF0中的第一个元素记为LowerF0;
    取出userF0中的第一个元素记为UpperF0;
    for (遍历originF0, 记当前元素为F0Data)
    {
        while (userLowerF0.time + 人声平滑模糊偏移 < F0Data.time)//利用two-Pointer算法优化时间复杂度
        {
            在userF0中取出LowerF0的下一个元素到LowerF0;
        }
        while (userUpperF0.time - 人声平滑模糊偏移 < F0Data.time)//利用two-Pointer算法优化时间复杂度
        {
            在userF0中取出UpperF0的下一个元素到UpperF0;
        }
        在userF0中找出在lowerF0到upperF0间音高的最小值minPitch和最大值maxPitch;
        if (F0Data.Pitch在minPitch和maxPitch之间) 演唱正确次数++;
        else
        {
            if (F0Data.Pitch升高或降低一个半音阶后在minPitch和maxPitch之间) 轻微跑掉次数++;
            else    严重跑掉次数++;
        }
    }
    根据演唱正确次数，轻微跑掉次数和严重跑掉次数来计算得分;
    利用当前录音延迟下的得分更新最高得分maxScore;
}
返回maxScore作为当前乐句打分;
```

9. 质量属性的设计

本项目采用 3 Tiers 架构，将界面层、应用层与数据层分离，提高了系统整体性能，有利于前后端的可扩展性与可移植性。

本项目使用多线程对于录音进行基频分析，很大程度上提高了歌唱打分速度，增加了 K 歌系统的易用性。

本项目将歌唱打分、歌词高亮、粒子特效等功能分离为独立的模块，采用多种设计模式，提高了系统的可扩展性与可移植性，同时提高了系统的可靠性，使其不至于因某个特定模块的问题而整体崩溃。