软件工程作业管理系统
概要设计

	人员	日期
拟制	刘飞宇 王超逸 曾鹏辉	2017-05-01
评审人	•	•
批准	•	•
签发	•	•

摘　要

随着当今社会的生活多元化，人们娱乐的方式多种多样，诸如 KTV、电影城等成为大众聚会、娱乐、休闲的首选之地，如今社会上的各种 KTV 会所，它们采用的点歌系统，不仅方便了我们挑选歌曲，而且降低了歌曲管理工作的效率，大大降低了相关人员的工作量。

我们在对 KTV 点歌系统进行了详细的需求分析后，设计了 KTV 点歌系统的概要设计文档，为软件的实现提供了一个逻辑框架，方便软件工程师的设计和软件的高效管理和便捷使用。

关键词：KTV　点歌系统　软件设计　概要设计　数据结构　数据库　界面设计　出错处理　安全保密设计　维护设计

目 录

图目录

表目录

第 1 章　引言

1.1　编写目的

在 KTV 点歌系统项目的前一阶段，也就是需求分析阶段，已经将系统用户对本系统的需求做了详细的阐述，这些用户需求已经在上一阶段中对不同用户所提出的不同功能，实现的各种效果做了调研工作，并在需求规格说明书中得到详尽得叙述及阐明。

本阶段已在系统的需求分析的基础上，对 KTV 点歌系统做概要设计。主要解决了实现该系统需求的程序模块设计问题。包括如何把该系统划分成若干个模块、决定各个模块之间的接口、模块之间传递的信息，以及数据结构、模块结构的设计等。在以下的概要设计报告中将对在本阶段中对系统所做的所有概要设计进行详细的说明，在设计过程中起到了提纲挈领的作用。

在下一阶段的详细设计中，程序设计员可参考此概要设计报告，在概要设计 KTV 点歌系统所做的模块结构设计的基础上，对系统进行详细设计。在以后的软件测试以及软件维护阶段也可参考此说明书，以便于了解在概要设计过程中所完成的各模块设计结构，或在修改时找出在本阶段设计的不足或错误。

1.2　项目背景

随着当今社会的生活多元化，人们娱乐的方式多种多样，诸如 KTV、电影城等成为大众聚会、娱乐、休闲的首选之地，如今社会上的各种 KTV 会所，它们采用的点歌系统，不仅方便了我们挑选歌曲，而且降低了歌曲管理工作的效率，大大降低了相关人员的工作量，下面是我们对该软件项目的介绍。

待开发的软件系统的名称：KTV 点歌系统

项目负责人：刘飞宇

开发人员：刘飞宇、王超逸、曾鹏辉

用户：软件开发人员、KTV 点歌系统维护人员、消费者

实现该软件的计算机网络：因特网

第 2 章　任务概述

本系统的目标是实现一个 KTV 点歌系统，包括客户端、服务器端两个部分。

客户端面向消费者，为消费者提供点歌、观影等娱乐服务。

服务器端面向维护人员，提供后台数据库并维护系统正常运行。

2.1　目标

实现 KTV 点歌系统，实现需求规格说明书中所描述的点歌功能、观影功能和娱乐服务等功能，并且保证系统的健壮性和数据安全。

2.2　开发与运行环境

2.2.1　开发环境的配置

表 2.1　开发环境的配置

类别	标准配置	最低配置
计算机硬件	基于 Intelx86 结构的 CPU 主频 >=2.0GHz 内存 >=2G 硬盘 >=200G	基于 Intelx86 结构的 CPU 主频 >=1.6GHz 内存 >=512M 硬盘 >=200G
计算机软件	WindowsXP 系统以上 Visual C++6.0 WinPCap3.1 以上 Tcl84	WindowsXP 系统 Visual C++6.0 WinPCap3.1 以上 Tcl83 以上
网络通信	至少要有一块可用网卡 能运行 IP 协议栈即可	至少要有一块可用网卡 能运行 IP 协议栈即可
其他	采用 MySQL 数据库	采用 MySQL 数据库

表 2.2　测试环境的配置

类别	标准配置	最低配置
计算机硬件	基于 Intelx86 结构的 CPU 主频 >=2.0GHz 内存 >=2G 硬盘 >=200G	基于 Intelx86 结构的 CPU 主频 >=1.6GHz 内存 >=512M 硬盘 >=200G
计算机软件	WindowsXP 系统以上 Visual C++6.0 WinPCap3.1 以上 Tcl84	WindowsXP 系统 Visual C++6.0 WinPCap3.1 以上 Tcl83 以上
网络通信	至少要有一块可用网卡 能运行 IP 协议栈即可	至少要有一块可用网卡 能运行 IP 协议栈即可
其他	采用 MySQL 数据库	采用 MySQL 数据库

2.2.2　测试环境的配置

2.2.3　运行环境的配置

2.3　需求概述

为了方便消费者在 KTV 中娱乐，我们开发具有丰富功能的 KTV 点歌系统，既方便用户快速挑选自己想要唱的歌曲，还能在唱歌同时进行一系列操作；除了唱歌之外，还可以选择电影库中已有的电影进行观看；同时在唱歌观影的同时还提供了服务功能，可以挑选饮料小吃等服务，各种服务易于操作，有效、快速。

为了方便维护人员维护 KTV 点歌系统，本系统在设计中逻辑严密，层次结构清晰，可快速定位故障位置，大大提高了维护人员的工作效率。

2.4　条件与限制

本系统只限在 KTV 中使用

表 2.3　运行环境的配置

类别	标准配置	最低配置
计算机硬件	基于 Intelx86 结构的 CPU 主频 >=2.0GHz 内存 >=2G 硬盘 >=200G	基于 Intelx86 结构的 CPU 主频 >=1.6GHz 内存 >=512M 硬盘 >=200G
计算机软件	WindowsXP 系统以上 Visual C++6.0 WinPCap3.1 以上 Tcl84	WindowsXP 系统 Visual C++6.0 WinPCap3.1 以上 Tcl83 以上
网络通信	至少要有一块可用网卡 能运行 IP 协议栈即可	至少要有一块可用网卡 能运行 IP 协议栈即可
其他	采用 MySQL 数据库	采用 MySQL 数据库

第 3 章　总体设计

3.1　软件描述

主要功能如下：

1. 用户可以按以下方式挑选歌曲。

按歌手名字: 可在本功能中按照歌手名字选择相应的歌曲等。

按歌曲类别: 可在本功能中按照歌曲类别进行选择, 包括流行歌曲, 民族风格, 情歌对唱, 经典名歌, 影视金曲等, 用户可以根据自己的需要按照不同的分类进行查询, 点播想要的歌曲。

按歌曲语种: 可在本功能中按照歌曲语种进行选择, 包括华语歌, 粤语歌, 英文歌, 韩文歌, 日文歌等, 用户可以根据语种来挑选歌曲。

按推荐歌单: 可在本功能中按照推荐歌单所列的歌曲进行选择。

2. 用户可以按以下方式查询歌曲信息。

拼音点歌: 可在本功能的支持下, 通过输入歌曲的首字母来选择歌曲。

手写点歌: 可在本功能的支持下, 通过书写歌曲的名字来选择歌曲。

3. 用户可以对自己已挑选的歌曲进行删除。

4. 用户可以对歌曲按歌曲名、所属歌手、语种、歌曲名字数方法对歌曲进行分类。

5. 用户可以上传增加新的歌曲。

6. 用户可以对自己添加的歌曲补充信息。

7. 用户补充新歌曲信息后, 该歌曲可以自动加入正确分类中。

8. 用户可以设置歌曲的相对音量控制和声道选择。

9. 具有录音功能, 可以把自己唱的歌录下来。

10. 具有电影库, 用户可挑选电影来看。

11. 用户可以暂停音乐的播放、切换到下一首音乐、重新开始、切换伴唱、原唱等操作。

12. 用户可以使用服务功能，呼叫服务员、查看剩余时间、购买饮料、小吃等。

13. 管理员可以对后台数据库进行录入、删除、更新操作。

3.2 处理流程

3.2.1 总体流程

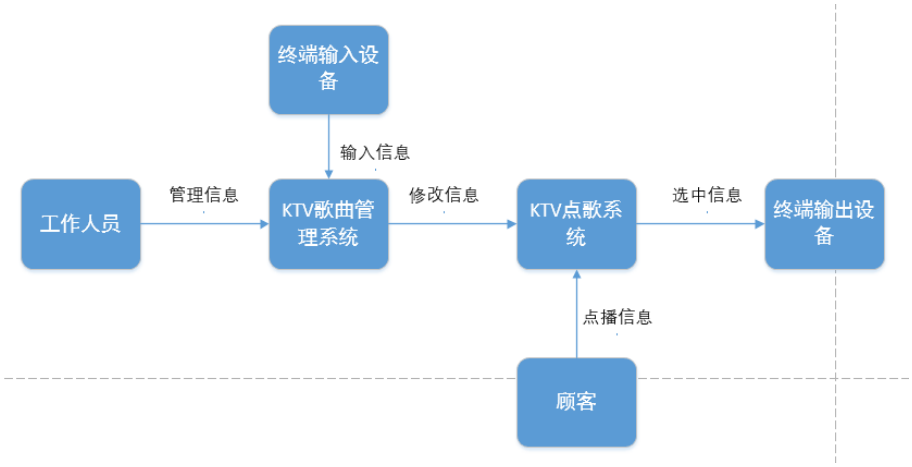

图 3.1 总体流程

3.2.2 系统基本流程

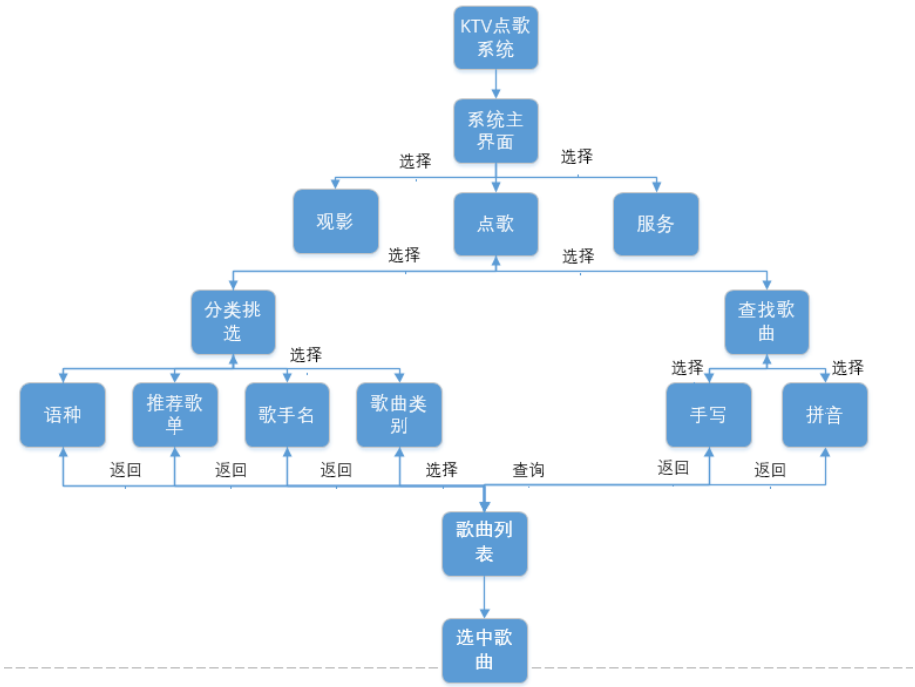

图 3.2 系统基本流程

3.2.3 客户端基本流程

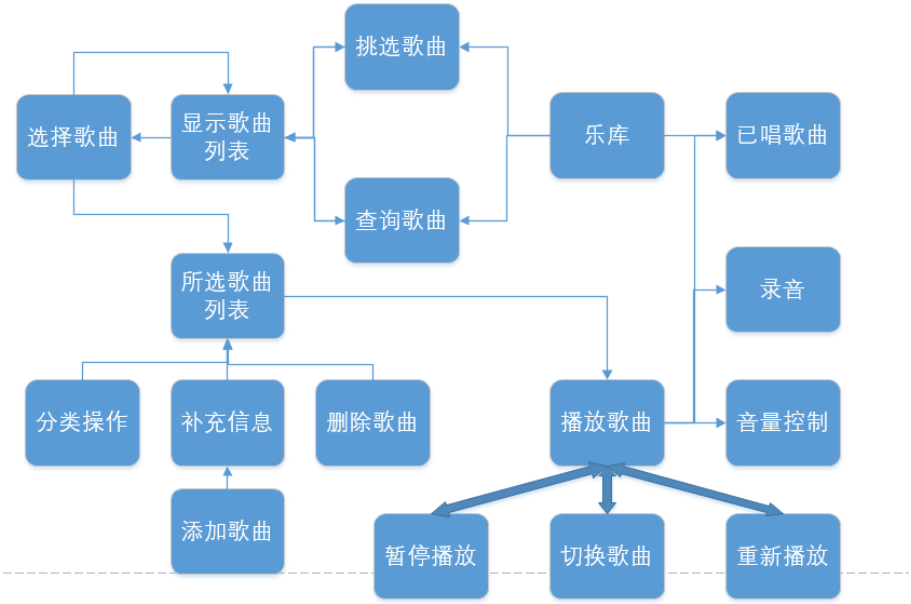

图 3.3 客户端基本流程

3.3 功能结构设计

3.3.1 管理员功能

图 3.4 管理员功能逻辑结构顶层图

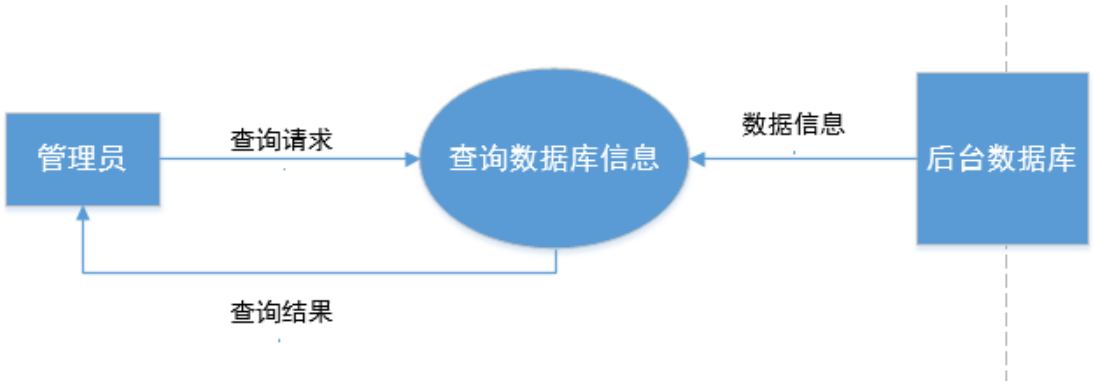

图 3.5　管理员功能逻辑结构 0 层图 1

图 3.6　管理员功能逻辑结构 0 层图 2

图 3.7　管理员功能逻辑结构 0 层图 3

图 3.8　管理员功能逻辑结构 0 层图 4

3.3.2　服务管理功能

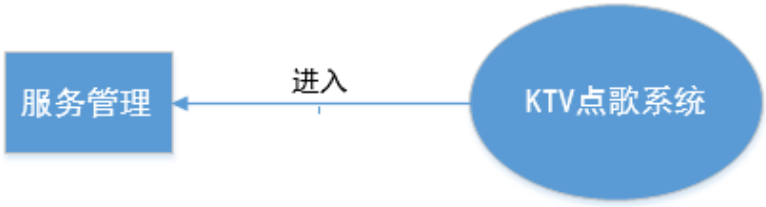

图 3.9　服务管理功能逻辑结构顶层图

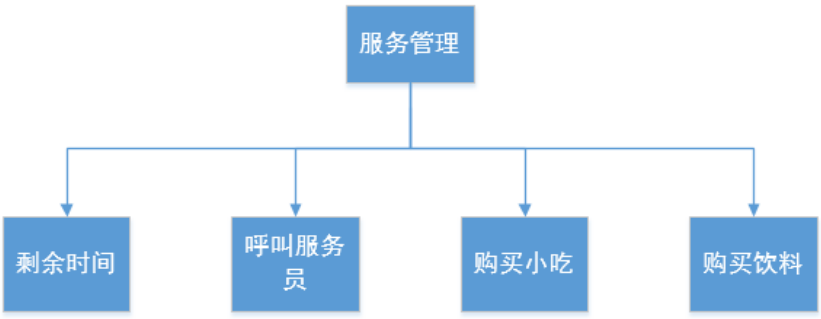

图 3.10　服务管理功能逻辑结构 0 层图

图 3.11　服务管理功能逻辑结构 1 层图 1

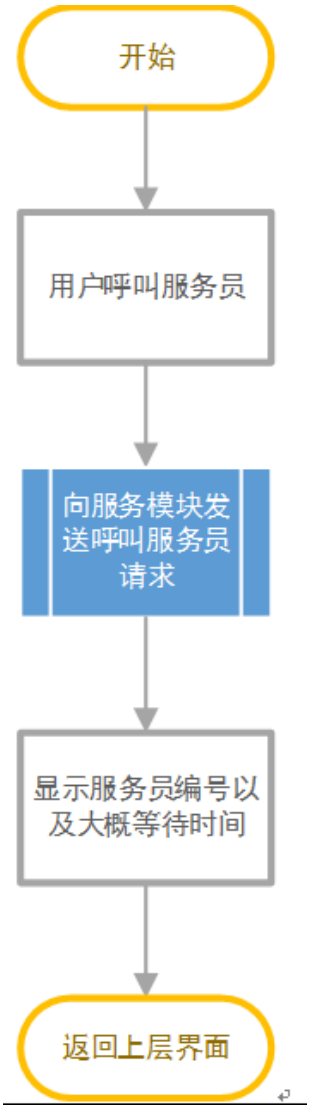

图 3.12　服务管理功能逻辑结构 1 层图 2

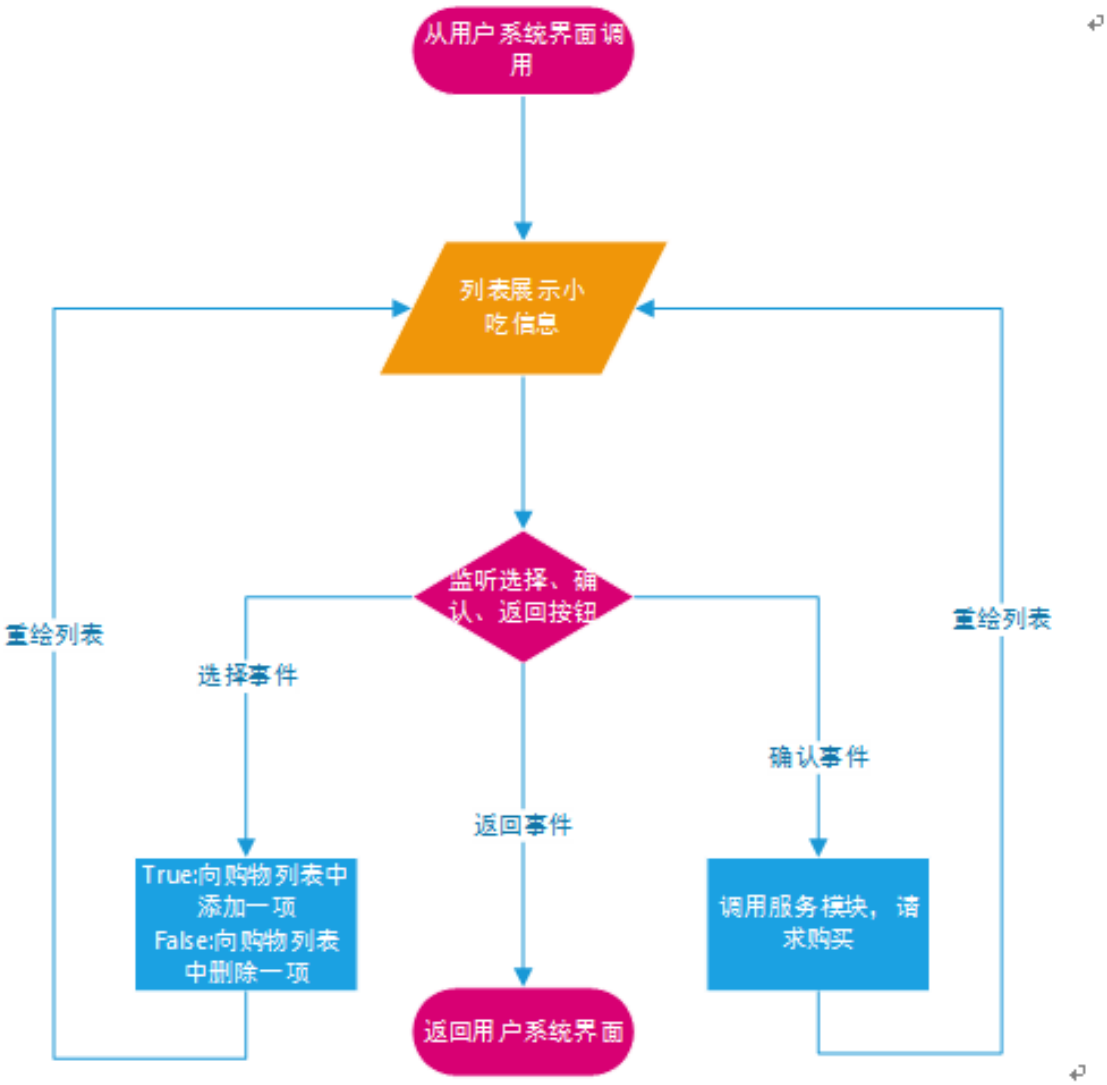

图 3.13　服务管理功能逻辑结构 1 层图 3

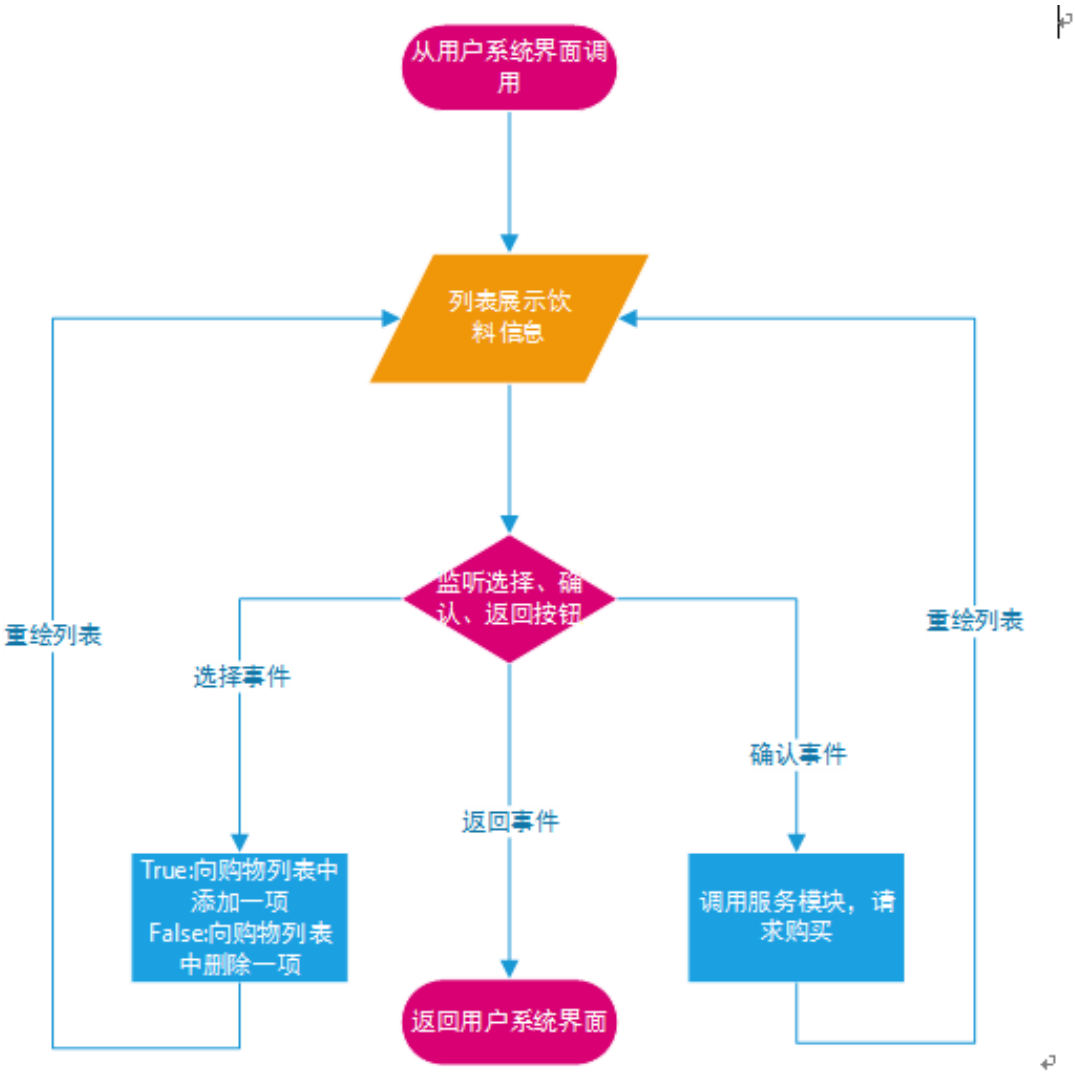

图 3.14　服务管理功能逻辑结构 1 层图 4

3.3.3　点歌观影功能

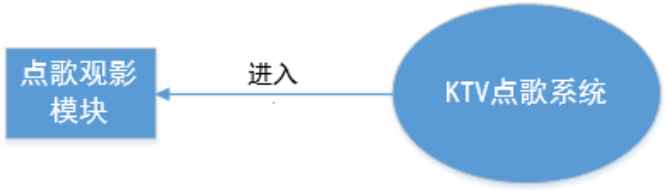

图 3.15　点歌观影功能逻辑结构顶层图

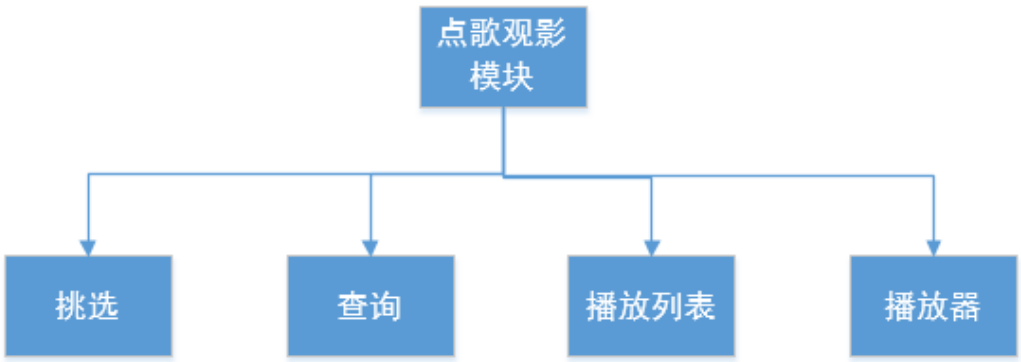

图 3.16　点歌观影功能逻辑结构 0 层图

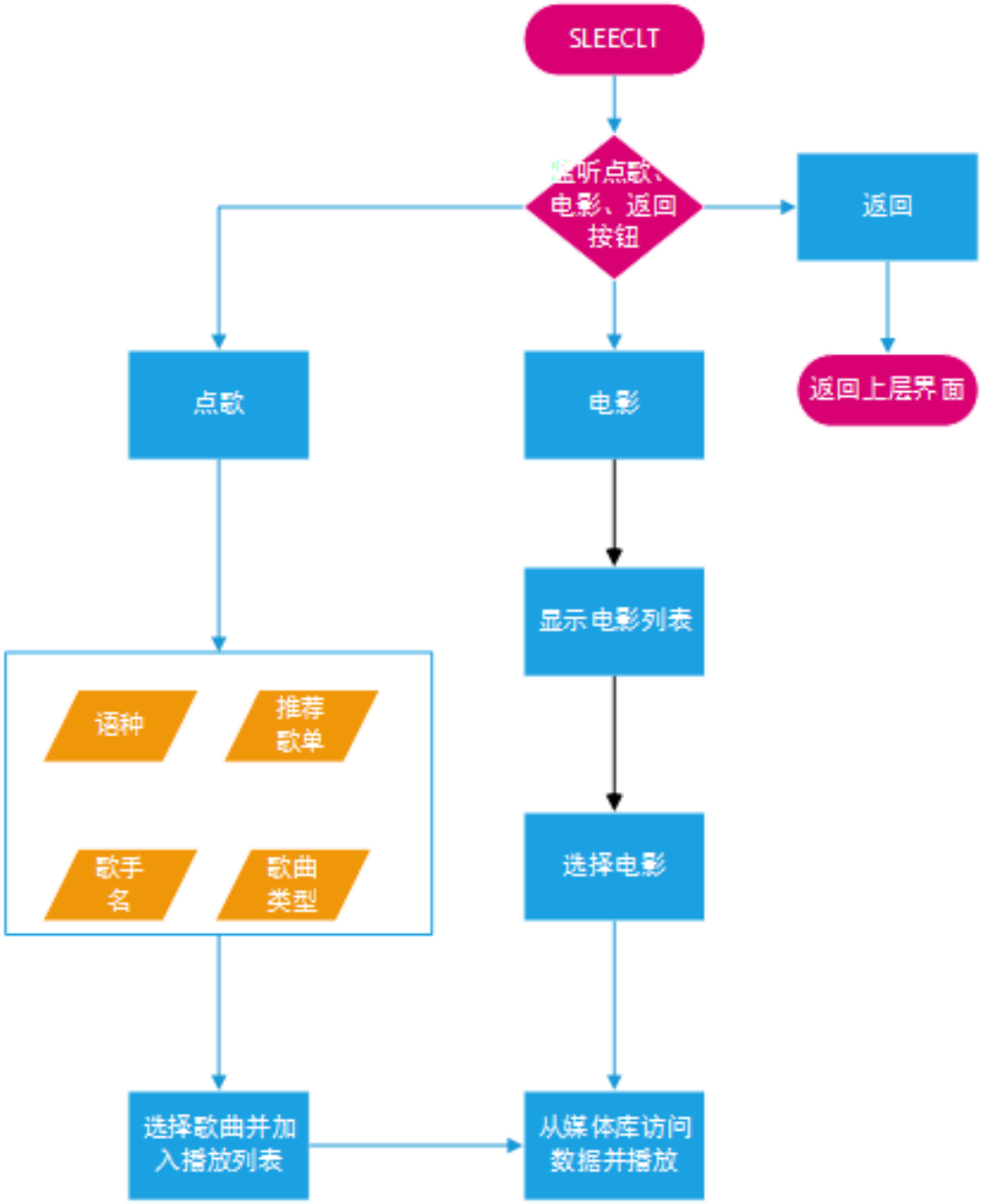

图 3.17　点歌观影功能逻辑结构 1 层图 1

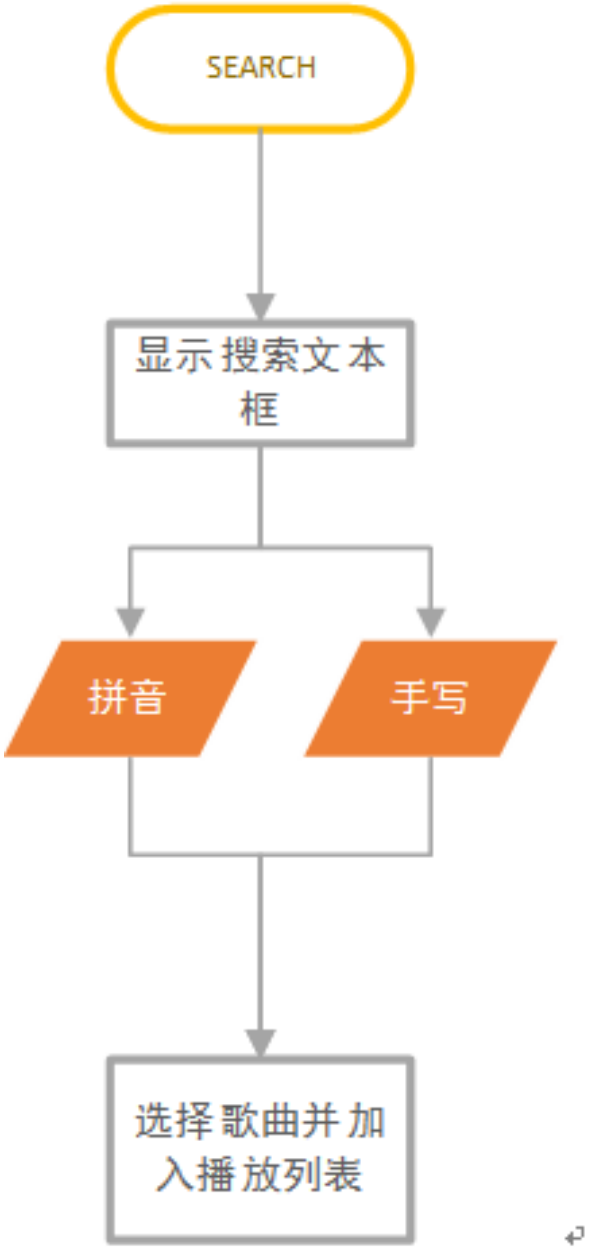

图 3.18　点歌观影功能逻辑结构 1 层图 2

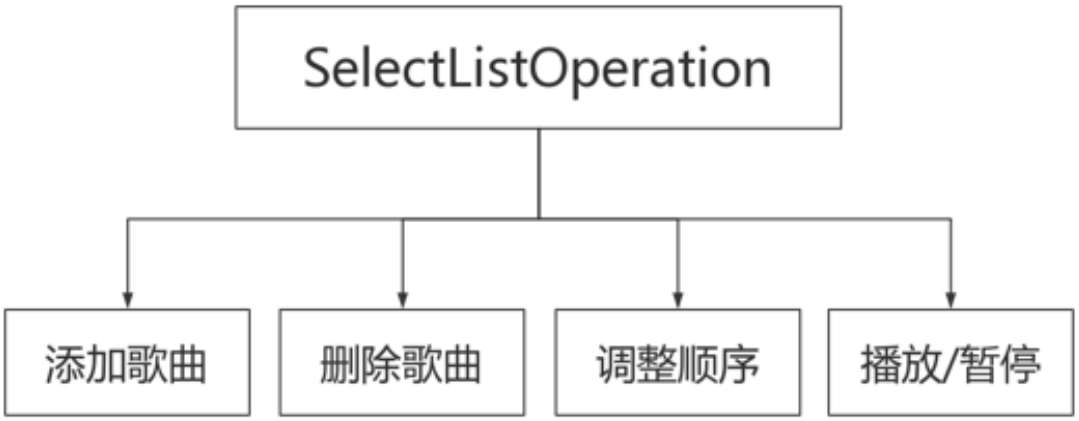

图 3.19　点歌观影功能逻辑结构 1 层图 3

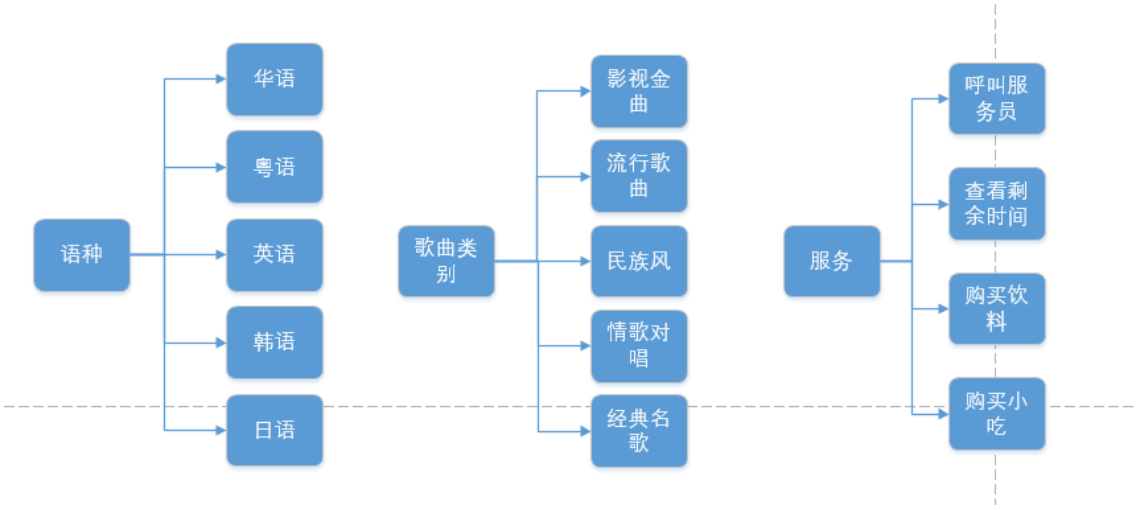

图 3.20　点歌观影功能逻辑结构 2 层图

3.3.4 播放器功能

图 3.21 播放器功能逻辑结构顶层图

图 3.22 播放器功能逻辑结构 0 层图

3.4 功能需求与程序代码的关系

表 3.1 功能需求与程序代码的关系表

数据结构	管理员功能	服务管理功能	点歌观影功能	播放器功能
Song	Y	•	Y	Y
Singer	Y	•	Y	Y
Movie	Y	•	Y	•
Snack	Y	Y	•	•
Drink	Y	Y	•	•
Administrator	Y	•	•	•

第 4 章　接口设计

4.1　外部接口

4.1.1　用户界面

1) 通过手写或拼音点歌，搜索相应歌曲并可以选择

2) 可以按类别挑选歌曲

3) 可以选择观影

4) 可以选择服务

5) 已选中歌曲列表的每首歌曲有删除按钮

6) 已选中歌曲列表中有添加按钮，点击后可以通过 USB 线导入手机中的歌曲

7) 已选中歌曲列表中有分类按钮

8) 唱歌界面有暂停、切换、重新开始按钮、伴唱、原唱按钮

9) 唱歌界面有录音按钮、音量调整、声道选择按钮

10) 管理员可对后台数据库数据进行录入、删除、更新操作

4.1.2　软件接口

调用 Potplayer 播放器播放歌曲 mv 和电影。

4.1.3　硬件接口

USB plug 插口

电源插口

音响

话筒

4.2　内部接口

4.2.1　选歌管理和列表管理内部接口

调用接口为 SendFromSelectToList(Song)

参数 Song:

类型定义：

```
public Song(){
   int songID;
   String songName;
   String songLanguage;
   String singer;
   String songType;
   String songPath;
   String songNameLetter;
   String songNumber;
   String songRemark;
}
```

含义：将选择的歌曲加入歌曲列表中。

4.2.2 列表管理和播放器内部接口

调用接口为 SendFromListToPlay(Song)

参数定义：

类型定义：

```
public Song(){
   int songID;
   String songName;
   String songLanguage;
   String singer;
   String songType;
   String songPath;
   String songNameLetter;
   String songNumber;
   String songRemark;
}
```

含义：列表管理将当前播放的歌曲信息送给播放器。

4.2.3 播放器和电影列表接口

调用接口为：PlayFilm(Film)

参数定义：

类型定义：

```
public Film(){
   int filmID;
   String filmName;
   String actor;
   String data;
   String time;
   String introduction;
}
```

含义：列表管理将当前播放的电影信息送给播放器。

4.2.4 管理员和后台歌曲数据库接口

调用接口为：ChangeSong(Song)

参数定义：

类型定义：

```
public Song(){
   int songID;
   String songName;
   String songLanguage;
   Singer singer;
   String songType;
   String songPath;
   String songNameLetter;
   String songNumber;
   String songRemark;
}
```

含义：管理员对后台歌曲数据库进行录入、删除、更新操作。

4.2.5 管理员和后台影视数据库接口

调用接口为：ChangeFilm(Film)

参数定义：

类型定义：

```
public Film(){
   int filmID;
   String filmName;
   String actor;
   String data;
   String time;
   String introduction;
}
```

含义：管理员对后台影视数据库进行录入、删除、更新操作。

4.2.6 管理员和小吃接口

调用接口为：ChangeSnack(Snack)

参数定义：

类型定义：

```
public Snack(){
   int snackID;
   String snackName;
   int price；
}
```

含义：管理员对小吃进行增删改操作。

4.2.7 管理员和饮料接口

调用接口为：ChangeDrink(Drink)

参数定义：

类型定义：

```
public Drink(){
   int drinkID;
   String drinkName;
```

int price；

}

含义：管理员对饮料进行增删改操作。

第 5 章　数据结构设计

5.1　逻辑结构设计

5.1.1　数据结构 1：Song

```
import java.io.Serializable;
public class Song implements Serializable {
    int songId;
    String songName;
    String songLanguage;
    Singer singer;
    String songType;
    String songPath;
    String songNameLetter;
    String songNumber;
    String songRemark;
    public String getSongName() {
        return songName;
    }
    public void setSongName(String singerName) {
        this.songName = singerName;
    }
    public int getSongId() {
        return songId;
    }
    public void setSongId(int songId) {
        this.songId = songId;
    }
    public String getSongLanguage() {
        return songLanguage;
```

```
}
public void setSongLanguage(String songLanguage)
    this.songLanguage = songLanguage;
}
public String getSongNameLetter() {
    return songNameLetter;
}
public void setSongNameLetter(String songNameLetter)
    this.songNameLetter = songNameLetter;
}
public String getSongNumber() {
    return songNumber;
}
public void setSongNumber(String songNumber)
    this.songNumber = songNumber;
}
public String getSongPath() {
 return songPath;
}
public void setSongPath(String songPath) {
    this.songPath = songPath;
}
public String getSongRemark() {
    return songRemark;
}
public void setSongRemark(String songRemark) {
    this.songRemark = songRemark;
}
public String getSongType() {
    return songType;
}
public void setSongType(String songType) {
```

```
        this.songType = songType;
    }
    public Singer getSinger() {
        return singer;
    }
    public void setSinger(Singer singer) {
        this.singer = singer;
    }
}
```

5.1.2 数据结构 2：Singer

```
import java.util.HashSet;
import java.util.Set;
public class Singer {
    private int singerId;
    private String singerName;
    private String singerImage;
    private String singerNameFirst;
    private String singerType;
    private String singerRemark;
    private Set song = new HashSet(0);
    public Set getSong()
        return song;
    }
    public void setSong(Set song)
        this.song = song;
    }
    public int getSingerId()
        return singerId;
    }
    public void setSingerId(int singerId) {
        this.singerId = singerId;
```

```
    }
    public String getSingerImage()
        return singerImage;
    }
    public void setSingerImage(String singerImage) {
        this.singerImage = singerImage;
    }
    public String getSingerName() {
        return singerName;
    }
    public void setSingerName(String singerName) {
        this.singerName = singerName;
    }
    public String getSingerNameFirst()
        return singerNameFirst;
    }
    public void setSingerNameFirst(String singerNameFirst) {
        this.singerNameFirst = singerNameFirst;
    }
    public String getSingerRemark() {
        return singerRemark;
    }
    public void setSingerRemark(String singerRemark) {
        this.singerRemark = singerRemark;
    }
    public String getSingerType() {
        return singerType;
    }
    public void setSingerType(String singerType)
        this.singerType = singerType;
    }
}
```

5.1.3 数据结构 3：Movie

```
import java.io.Serializable;
public class Movie implements Serializable {
    int movieId;
    String movieName;
    String movieLanguage;
    String actor;
    String movieType;
    String moviePath;
    public String getMovieName() {
        return movieName;
     }
     public void setMovieName(String actorName)
        this.movieName = actorName;
    }
     public int getMovieId() {
        return movieId;
    }
     public void setMovieId(int movieId) {
        this.movieId = movieId;
     }
     public String getMovieLanguage() {
        return movieLanguage;
     }
     public void setMovieLanguage(String movieLanguage) {
        this.movieLanguage = movieLanguage;
     }
     public String getMoviePath() {
        return moviePath;
     }
     public void setMoviePath(String songPath) {
```

```
        this.moviePath = moviePath;
    }
    public String getMovieType() {
        return movieType;
    }
    public void setMovieType(String songType) {
        this.movieType = movieType;
    }
    public void setActor(String actor) {
        this.actor = actor;
    }
}
```

5.1.4 数据结构 4：Snack

```
public class Snack {
    private int price;
    private String snackName;
    public String getSnackName(){
        return snackName;
    }
    public void setSnackName(String snackName) {
        this.snackName = snackName;
    }
    public String getPrice() {
        return price;
    }
    public void setPrice(String price) {
        this.price = price;
    }
}
```

5.1.5 数据结构 5：Drink

```
public class Drink {
    private int price;
    private String drinkName;
    public String getDrinkName() {
        return drinkName;
    }
    public void setDrinkName(String drinkName) {
        this.drinkName = drinkName;
    }
    public String getPrice() {
        return price;
    }
    public void setPrice(String price) {
        this.price = price;
    }
}
```

5.1.6 数据结构 6：Administrator

```
public class Administrator {
    private int ID;
    private int password;
    public Boolean login() {
    return Y/n;
    }
    public void insertSong(Song song) {
}
    public void deleteSong(Song song) {
    }
    public void updateSong(Song song) {
    }
```

```
public void insertMovie(Movie movie) {
}
public void deleteMovie(Movie movie) {
}
public void updateMovie(Movie movie) {

public void insertDrink(Drink drink) {
}
public void deleteDrink(Drink drink) {
}
public void updateDrink(Drink drink) {
}
public void insertSnack(Snack snack) {
}
public void deleteSnack(Snack snack) {
}
public void updateSnack(Snack snack) {
}
public String getID() {
 return ID;
}
public void setPassword(int password) {
 this.password = password;
}
```

5.2 物理结构设计

各数据结构无特殊物理结构要求。

5.3 数据结构与程序模块的关系

表 5.1　数据结构与程序代码的关系表

数据结构	管理员功能	服务管理功能	点歌观影功能	播放器功能
Song	Y	•	Y	Y
Singer	Y	•	Y	Y
Movie	Y	•	Y	•
Snack	Y	Y	•	•
Drink	Y	Y	•	•
Administrator	Y	•	•	•

第 6 章　数据库设计

6.1　数据库环境说明

本系统的数据系统采用 MySQL 数据库系统。MySQL 是一种快速易用的 RDBMS，很多企业 (不分规模大小) 都在使用它来构建自己的数据库。MySQL 由一家瑞典公司 MySQL AB 开发、运营并予以支持。它之所以非常流行，原因在于具备以下这些优点:

- 基于开源许可发布，无需付费即可使用。

- 自身的功能非常强大，足以匹敌绝大多数功能强大但却价格昂贵的数据库软件。

- 使用业内所熟悉的标准 SQL 数据库语言。

- 可运行于多个操作系统，支持多种语言，包括 PHP、PERL、C、C++ 及 Java 等语言。

- 支持大型数据库，最高可在一个表中容纳 5 千多万行。每张表的默认文件大小限制为 4GB，不过如果操作系统支持，你可以将其理论限制增加到 800 万 TB。

- 可以自定义。开源 GPL 许可保证了程序员可以自由修改 MySQL，以便适应各自特殊的开发环境。

6.2　数据库的命名规则

6.2.1　数据库命名规范

1. 尽量简洁明义，能够一眼看出来这个数据库是用来做什么的;
2. 使用名词作为数据库名称，并且只用英文，不用中文拼音;
3. 使用英文字母，全部小写，控制在 3-7 个字母以内;
4. 如果有多个单词，则使用下划线隔开，不建议驼峰命名;

6.2.2　表命名规范

1. 表名使用英文小写单词，如果有多个单词则使用下划线隔开;
2. 表名简介，使用常见单词，避免使用长单词和生僻词;

3. 表引擎取决于实际应用场景及当前数据库中的已经存在的存储引擎; 日志及报表类表建议用 myisam，与搜索相关的表建议用 innodb 引擎。总体来讲数据库默认 innodb;

4. 数据表必须有主键，且建议均使用该类型的 id 作为主键, 并为其创建索引;

5. 默认使用 utf8 字符集 (由于数据库定义使用了默认，数据表可以不再定义，但为保险起见，建议都写上);

6. 所有的表都必须有备注，写明白这个表中存放的数据内容;

7. 预估表数据量，如果数据量较大 (超过 500w) 则需要考虑分表策略。

8. 职责相近的表，命名规则应该相同; 表名是单数还是复数，关联表如何命名，字符数限制等。

6.2.3 字段命名规范

1. 数据库字段命名与表名命名类似;

2. 使用小写英文单词，如果有多个单词使用下划线隔开;

3. 使用简单单词，避免生僻词;

4. 字段应当有注释，描述该字段的用途及可能存储的内容，如枚举值则建议将该字段中使用的内容都定义出来;

5. 是别的表的外键均使用 xxx_id 的方式来表明;

6. 表的主键一般都约定成为 id，自增类型;

7. 所有字段，均为非空，最好显示指定默认值;

8. 有些驱动对 tinyint 支持不够好，通常建议按容量来选择字段;

9.text 字段尽量少用，或是拆到冗余表中;

6.3 逻辑设计

6.3.1 关系模式要求

主属性: 包含键的所有属性关系模式要求达到 4NF(减少冗余，消除操作异常)。

第一范式 (1NF): 若关系模式 R 的每一个分量是不可分的数据项，则关系模式属于第一范式。即每个属性都是不可拆分的。

第二范式 (2NF):R 属于 1NF, 且每一个非主属性完全依赖于键 (没有部分依赖)，则 R 属于 2NF。

第三范式 (3NF):R 属于 2NF, 且每个非主属性即不部分依赖于码，也不传递依赖于码。

例如: 歌曲信息 (歌曲 ID，歌曲名，语种，歌手名，类型，排行)

该关系的主键是: 歌曲 ID

歌曲 ID → 歌曲名，歌曲 ID → 歌曲名，歌曲 ID → 语种，歌曲 ID → 歌手名，歌曲 ID → 类型，歌曲 ID → 排行; 即所有属性都唯一直接依赖于歌曲 ID

6.3.2 实体 ER 图

6.3.2.1 歌曲信息

	歌曲 ID	歌曲名	语种	歌手名	类型	排行

图 6.1 歌曲信息

满足 3NF，歌曲 ID 为主码。

6.3.2.2 歌手信息

	歌手 ID	歌手名	歌手类型

图 6.2 歌手信息

满足 3NF，歌手 ID 为主码。

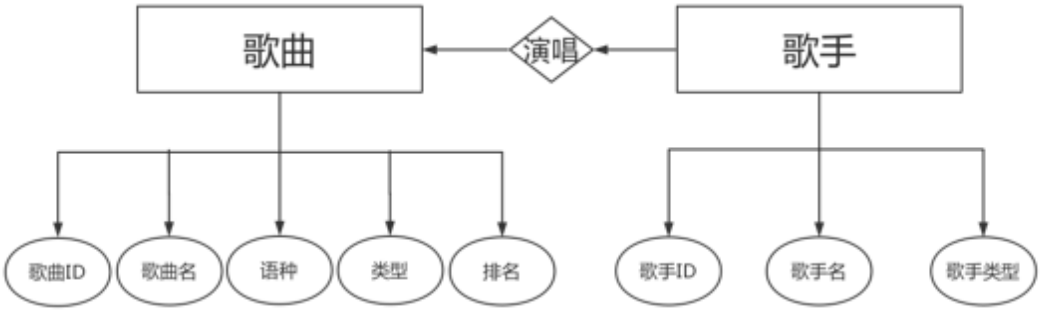

图 6.3　歌手歌曲关系图

6.3.2.3　饮料信息

	饮料名	价格

图 6.4　饮料信息

满足 3NF，饮料名为主码。

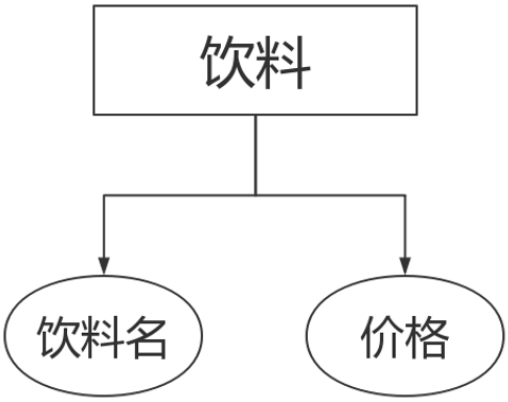

图 6.5　饮料信息结构图

6.3.2.4 小吃信息

	小吃名	价格

图 6.6 小吃信息

满足 3NF，小吃名为主码。

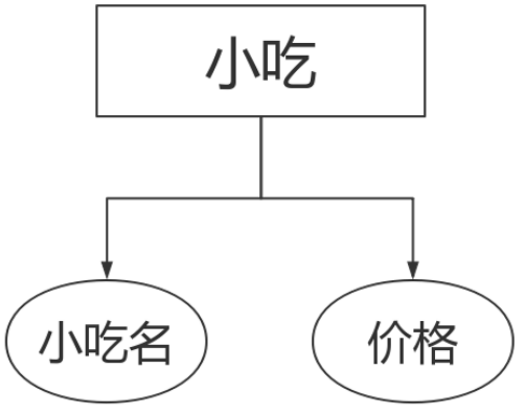

图 6.7 小吃信息结构图

6.3.2.5 电影信息

	电影 ID	影片名	主演	上映日期	时长	简介

图 6.8 电影信息

满足 3NF，电影 ID 为主码。

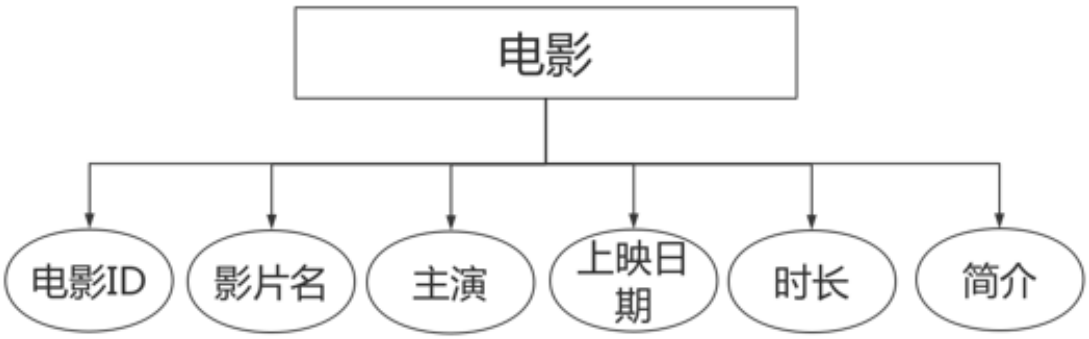

图 6.9　电影信息结构图

6.4　物理设计

6.4.1　数据库产品

使用 Oracle 公司的 MySQL 数据库，采用集中式。

6.4.2　实体属性、类型、精度

顺序	歌曲名	语种	歌手	类型	时长
int	String	String	String	String	int

图 6.10　歌曲列表设计图

6.5　安全性设计

1. 避免从互联网访问 MySQL 数据库，确保特定主机才拥有访问特权
2. 定期备份数据库
3. 禁用或限制远程访问
4. 设置 root 用户的口令并改变其登录名
5. 移除测试 (test) 数据库
6. 禁用 LOCAL INFILE
7. 移除匿名账户和废弃的账户
8. 降低系统特权
9. 降低用户的数据库特权

10. 移除和禁用.mysql_history 文件

11. 安全补丁

12. 启用日志

13. 改变 root 目录

14. 禁用 LOCAL INFILE 命令

6.6 数据库管理与维护说明

数据的管理和维护基于 MySQL 系统自身的功能进行，使用企业管理器的导入数据，导出数据，备份数据库和还原数据库等功能对数据库进行管理和维护。

第 7 章　界面设计

7.1　界面的关系图和工作流程图

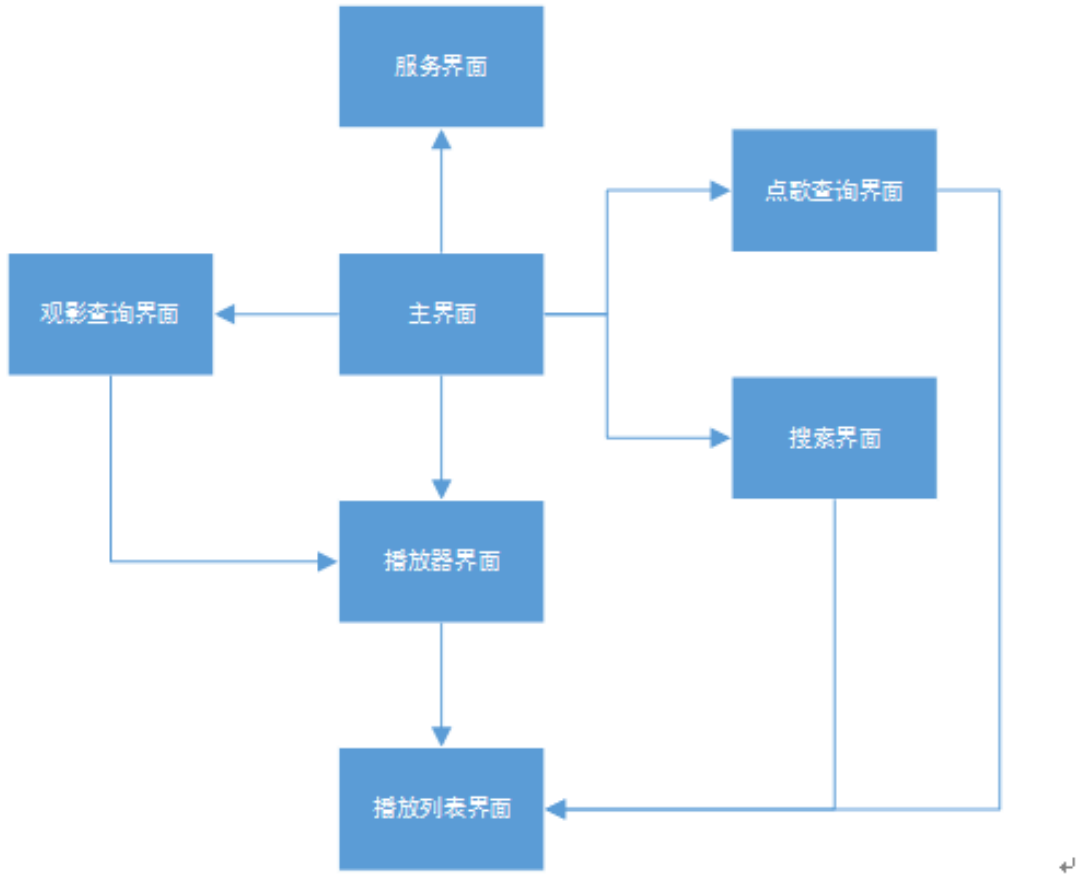

图 7.1　界面关系图

注释：每个下层界面都带有返回键返回上层界面

7.2　主界面设计

主界面由搜索、点歌、观影、播放器、服务等控件组成。

点击搜索框或搜索按钮进入搜索界面，点击点歌进入选择歌曲界面, 点击观影进入选择电影界面，点击播放器进入播放器界面，点击服务进入服务界面。右下角有一个关机按钮。

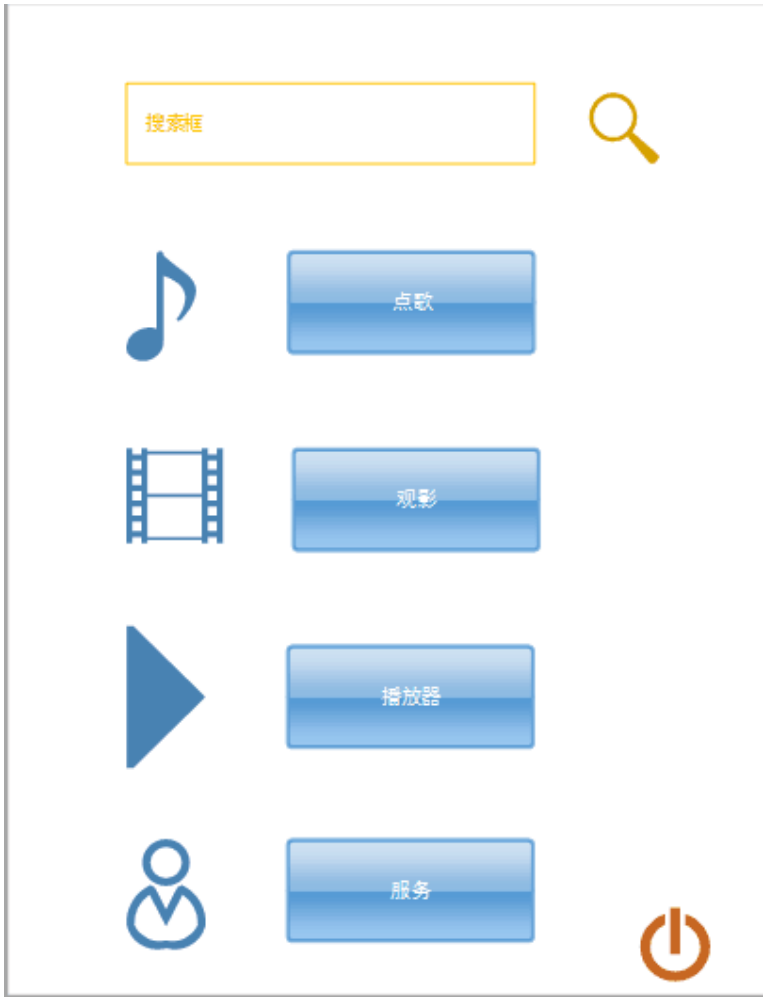

图 7.2　主界面

7.3　搜索界面设计

Search 通过手写或者拼音方式对数据库进行 select 操作，并将数据库查找结果显示在屏幕上，选取想要的歌曲加入歌曲列表。

主界面的构成：将屏幕分为两部分，左为手写，右为拼音。点击手写后在屏幕上写入歌曲或歌手名。点击拼音后通过屏幕上的软键盘输入歌曲或歌手名的首字母拼音。

图 7.3　搜索界面

7.4　点歌界面设计

Select 界面通过语种，推荐歌单，歌手名和歌曲类别四种分类方式进行点歌，将选取的歌曲添加到歌曲列表。

点击点歌进入点歌界面，屏幕分为四部分，左上为推荐歌曲，右上为歌手名，左下为语种，右下为歌曲类型。

其中语种分为华语，英语，粤语，日语和韩语，歌曲类型分为影视金曲，流行歌曲，民族风，情歌对唱和经典名歌。点击任一种后进入相应的数据库，显示数据库列表中的歌曲信息，点击加入列表即可完成点歌。每个层次界面左上都有 back 按钮，点击可返回上一级界面。

图 7.4　点歌界面

7.5　观影界面设计

点击观影进入观影界面，会将数据库中的电影信息全部显示在屏幕上，点击想观看的电影即可播放。

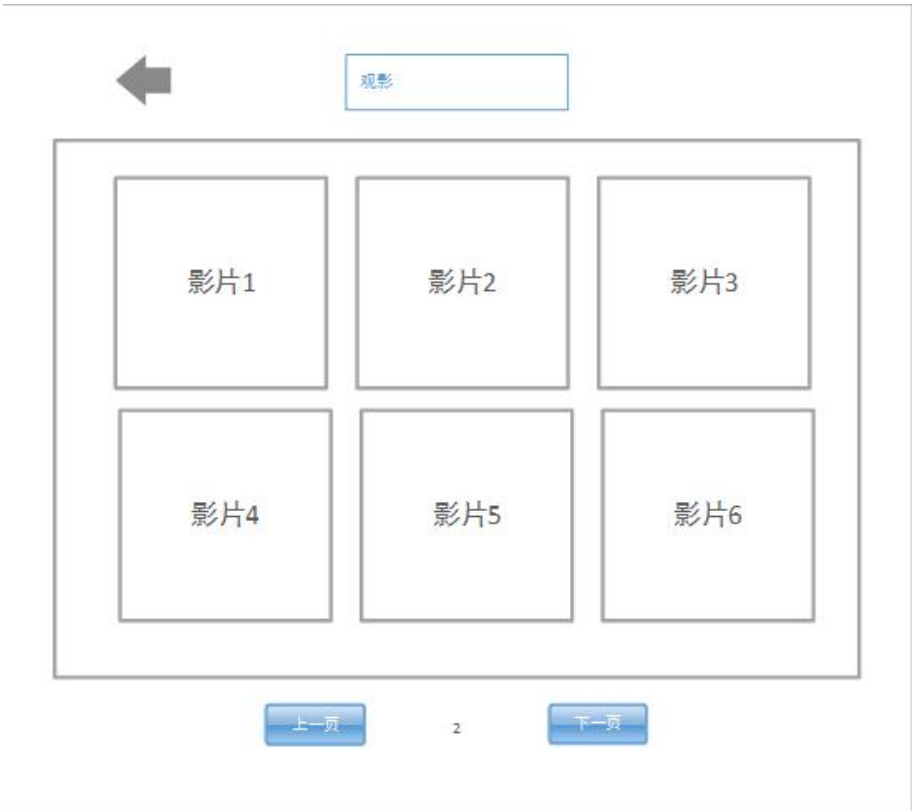

图 7.5　观影界面

7.6　播放器界面设计

主界面的构成：全屏显示 mv 或视频，歌词下面有返回、上一首、重播、暂停/播放，下一首、音量、播放列表和录音按钮。

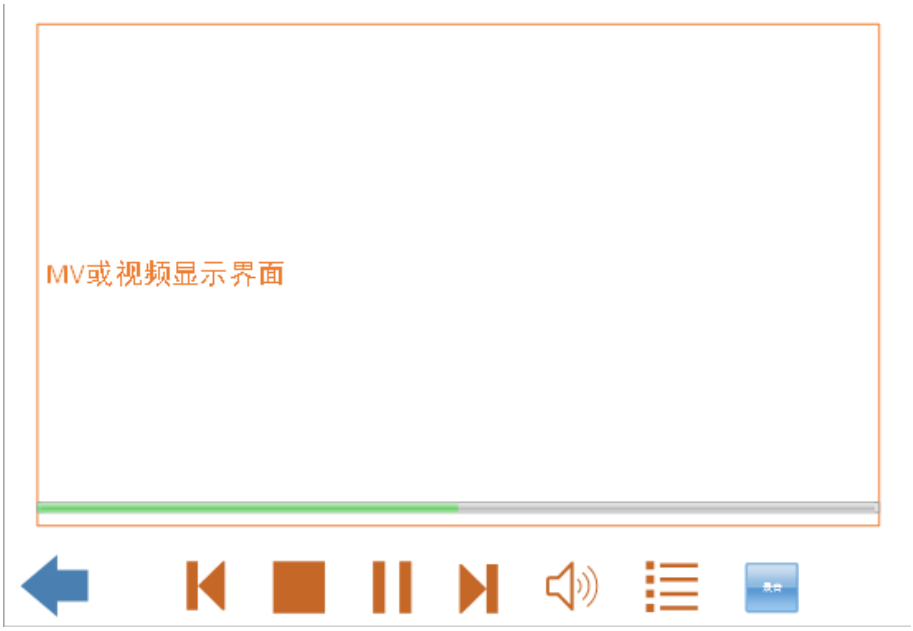

图 7.6　播放器界面

7.7　服务界面设计

Service 实现了呼叫服务员，查看剩余时间，购买饮料喝小吃的功能。

主界面的构成：屏幕上方显示剩余时间，右侧为呼叫服务员按钮，下方为购买时长，购买饮料，购买小吃的选项卡。点击时长、小吃、饮料按钮会切换选项卡，并显示相关商品的所有信息，点击复选框即可加入购买列表。左上角为返回按钮，点击可以返回上一界面；左下角为购物车按钮，点击可查看已购买物品列表，并确认购买。

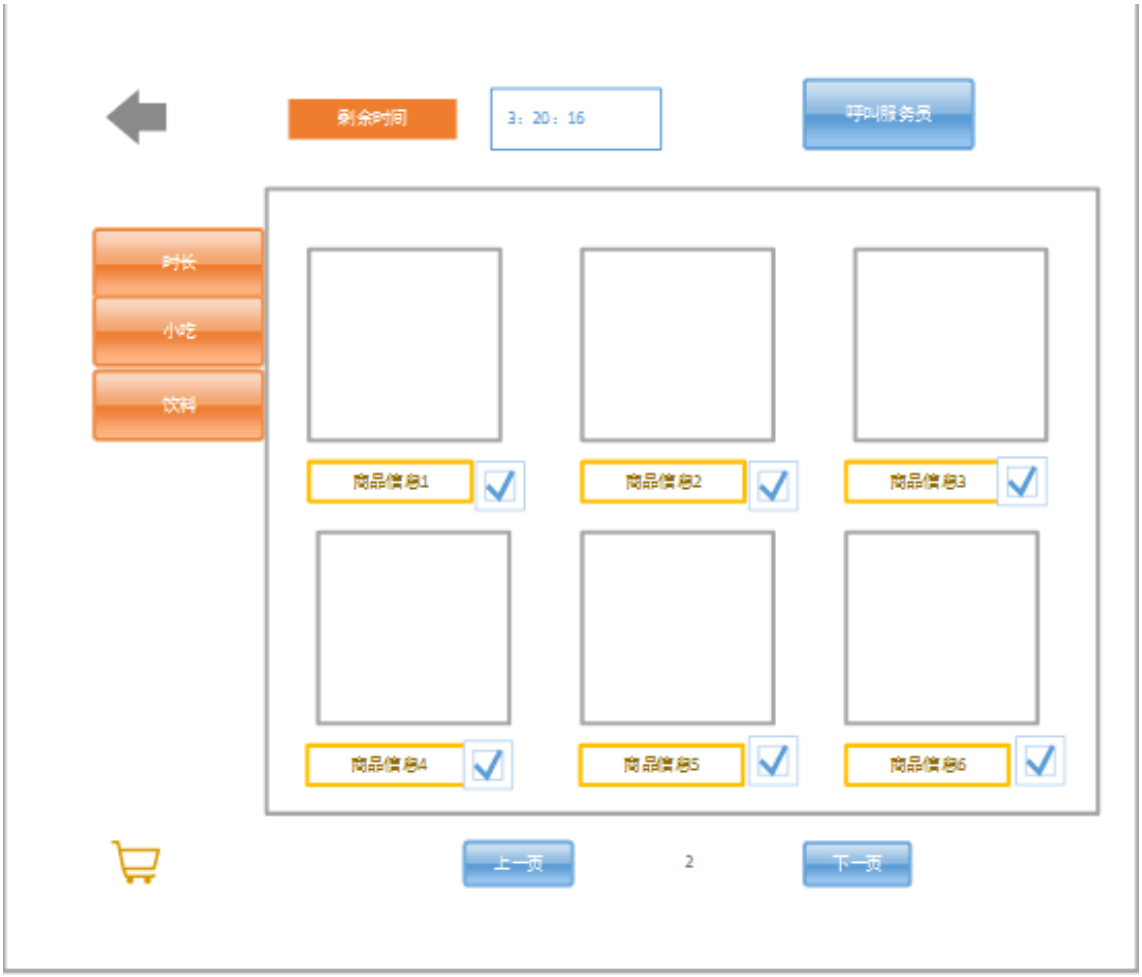

图 7.7　服务界面

7.8　列表界面设计

多个模块需要展现列表，现通过歌曲列表描绘列表布局。

SelectListOperation 通过对已选歌曲操作，实现改变歌曲播放顺序，删除和添加操作。主界面的构成：点击屏幕上的已选歌曲，歌曲列表将从右侧滑出，占屏幕三分之一，从上到下为播放歌曲的顺序，第一个为现在正在播放的歌曲，用高亮标识歌曲名，点击该歌名可以实现播放/暂停操作，可以通过触屏拖拽改变歌曲顺序，将歌曲左滑后出现删除选项，点击即可删除。

图 7.8　列表界面

第 8 章　出错处理设计

8.1　数据库出错处理

对数据库进行备份，当数据库出现错误后，优先使用备份数据库，系统继续正常使用，并通知主机数据库的错误信息，便于后期维护。

8.2　某模块失效处理

当模块失效时，立即暂停剩余时间并上传到前台服务器，停止所有服务，并通知前台服务人员。

保存相应的错误信息，若系统重启后可继续使用，则可继续服务，当系统结束服务后，由维护人员对错误进行维护。

若系统重启仍无法使用，则停止服务，可换至另一台机器继续进行服务。

第 9 章　安全保密设计

使用 PBKDF2 算法进行加密。该算法大致过程为在 HASH 算法基础上增加随机盐，并进行多次 HASH 运算。HASH 算法一般选用 sha256，随机盐的长度一般不能少于 8 字节，HASH 次数至少也要 100 次。

第 10 章　维护设计

确保网络及操作系统安全。网络系统是数据库应用的外部环境和基础，网络系统安全是数据库安全的第一道屏障。从技术角度讲，网络系统层次的安全防范技术有很多种，大致可以分为防火墙、数字签名与认证、入侵检测等。操作系统是数据库系统的运行平台，能够为数据库系统提供一定程度的安全保护。操作系统的安全控制方法主要是采用隔离控制、访问控制、信息加密和审计跟踪。主要安全技术有操作系统安全策略、安全管理策略等。

加强管理员身份验证。管理员身份验证是数据库系统的重要防线。利用窗体身份验证数据库程序的漏洞，进而获取存储在数据库中的用户身份验证密码，这是目前对网络数据库攻击最常见的方式。使用带有 salt 值的单向密码哈希值，以避免用户密码在数据库中以明文形式存储，减轻字典攻击带来的威胁。

对重要数据加密。数据加密交换又称密码学，是计算机系统对信息进行保护的一种最可靠的办法。它利用密码技术对信息进行交换，实现信息隐蔽，从而有效保护信息的安全不受侵犯。数据库加密要求加解密的粒度是每个记录的字段数据。采用库外口加密的方式，对密钥的管理较为简单，只需借用文件加密的密钥管理方法，将加密后的数据块纳入数据库，在算法或数据库系统中做些必要的改动就行。这样有利于公共数据字典的使用和维护系统的完整性。

做好数据库备份与恢复。数据备份是备份数据库某个时刻的数据状态，当系统出现意外时用来恢复系统。采用数据库的 UNDO//REDO 日志和 CheckPoint 功能来恢复数据库。

做好数据库的并发控制。为解决数据库在多个用户同时访问统一数据的读写冲突问题，必须维护数据的一致性和正确性。在这个卡拉 OK 系统中，有多个用户共享数据库，由于产生了并发操作，有可能影响数据的一致性。所以，要用“锁”应采用多粒度锁协议等办法来控制并发调度，保证数据库的一致性

具体方案：

每日流程

1. 一致性检查: 检查数据库的完整性.

2. 更新索引的统计数据;

3. 整理数据库碎片;

4. 重建索引;

5. 备份;

6. 检查文件的大小, 并释放磁盘空间;

每周流程

1. 备份系统数据库;

2. 数据库大小/增长情况/磁盘自由空间的情况;

3. 批处理作业是否正确执行;

4.DBCC 作业是否正确执行;

5.SQL 日志的错误;

6. 复制日志代理的运行情况;

7. 复制分发清除作业是否正确执行;

8.mySQL 上一次重新启动.