前段渲染框架

详细设计文档

|  |  |  |
| --- | --- | --- |
| 文件状态：  [ ] 草稿  [√] 正式发布  [ ] 正在修改 | 文件标识： |  |
| 当前版本： | 1.0 |
| 作 者： | 杨武 |
| 完成日期： | 2022/11 |

目录

[1.功能拆分 3](#_Toc152028919)

[2.流程图 4](#_Toc152028920)

[2.1组件渲染流程 4](#_Toc152028921)

[2.2组件更新逻辑 5](#_Toc152028922)

[2.3. 登录流程图 6](#_Toc152028923)

[3. 组件解析 7](#_Toc152028924)

[3.1.词法解析 7](#_Toc152028925)

[3.2.语法解析 7](#_Toc152028926)

[3.3.自定义指令解析 7](#_Toc152028927)

[4.组件渲染 7](#_Toc152028928)

[4.1.组件渲染 7](#_Toc152028929)

[4.2.组件更新 8](#_Toc152028930)

[4.3.组件卸载 8](#_Toc152028931)

[5.数值代理 8](#_Toc152028932)

[1.proxy代理 8](#_Toc152028933)

[2.Watcher监听 8](#_Toc152028934)

[6.路由管理 9](#_Toc152028935)

[6.1.路由表 9](#_Toc152028936)

[6.2..路由 9](#_Toc152028937)

[7. 页面设计 10](#_Toc152028938)

[7.1.登录页面 10](#_Toc152028939)

[7.2.注册平台 10](#_Toc152028940)

[7.3.音乐平台 11](#_Toc152028941)

[8. 数据库设计 11](#_Toc152028942)

[8.1.用户表 11](#_Toc152028943)

[8.2.歌单表 11](#_Toc152028944)

[8.3.歌曲信息表 11](#_Toc152028945)

[8.3.歌曲表 12](#_Toc152028946)

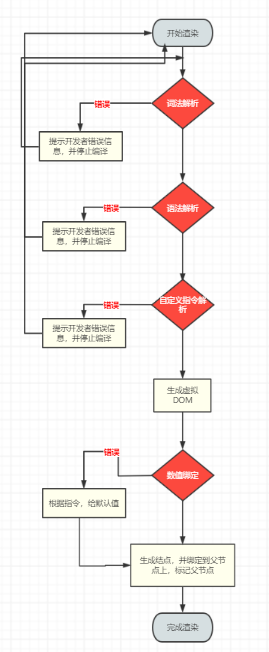
[9. 安全设计 12](#_Toc152028947)

# 1.功能拆分

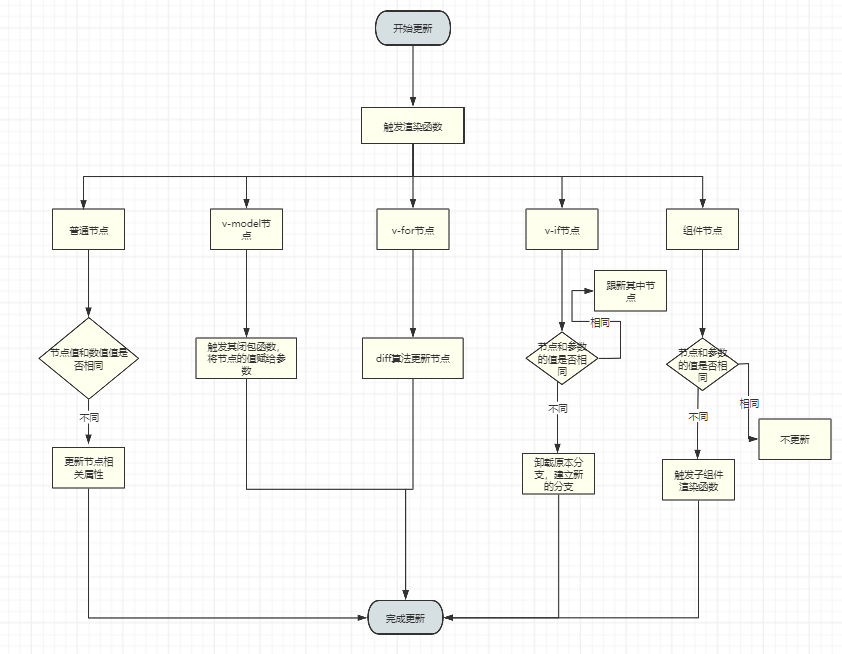
1. 前端动态编译框架功能要求：路由控制，缓存控制，动态编译，响应式更新，组件化开发
   1. 路由控制：开发者通过使用框架所提供函数修改路由，从而跳转页面
   2. 缓存控制：开发者通过使用框架所提供函数即可操作缓存数据
   3. 动态编译：开发者设定命令，编译时根据数据动态的生成节点
   4. 响应式更新：开发者设定响应式数据和组件内页面元素的绑定关系，修改响应式数据会触发组件更新，从而修改页面
   5. 组件化开发：开发者设定组件、组件和其他组件数据和逻辑关系、组件内的数据，编译时组件会读取组件，导入外部数据和逻辑关系，从而生成一个具有单独逻辑单元的一段html代码
2. 应用要求功能：用户和管理员注册登录、创建歌单，修改歌单名称，删除歌单。上传歌曲，下载歌曲，播放歌曲，修改歌曲所属的歌单
   1. 管理员可以查看和删除所有歌单和歌曲，而用户只有使用的功能。
   2. 上传歌曲：会采取转码，切片上传方式，来保证用户收听时的稳定性和快速性。

# 2.流程图

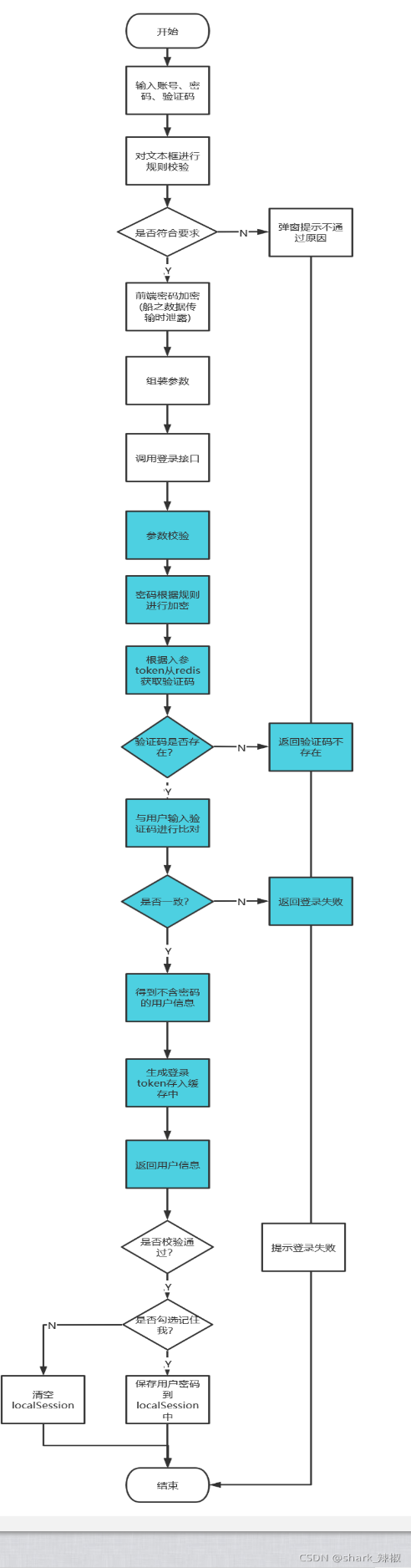
## 2.1组件渲染流程



## 2.2组件更新逻辑



## 2.3. 登录流程图



# 组件解析

## 3.1.词法解析

词法解析中状态机一共有四种状态，根据读取组件模板的字符而改变状态机状态。

状态1：标签开始

状态2：标签简单属性

状态3：标签复杂属性

状态4：标签闭合

词法解析读取组件模板并生成属性数组

## 3.2.语法解析

语法解析负责解析属性数组，并生成语法树。

语法解析便利属性数组，遇到状态1则递归一层，状态四则返回上一层。

状态二和状态三则添加属性至attr属性

## 3.3.自定义指令解析

自定义指令解析器负责解析v-if、v-for、v-model、{{}}四种自定义指令，并生成模板树

自定义指令解析器先序遍历语法树，对于自定义指令有不同的处理逻辑

1. v-for指令

为其附加动态节点标记和v-for标记，新建一个区域结点将其包裹，将其区域结点添加到动态结点库中

1. v-if指令

为其附加动态节点标记和v-if标记，新建一个区域结点将其包裹，将其区域结点添加到动态结点库中

1. v-model指令

为其附加动态节点标记和v-model标记，并将该节点添加到动态节点库中

1. {{}}指令

为其附加动态节点标记，将其区域结点添加到动态结点库中

# 4.组件渲染

## 4.1.组件渲染

组件先序遍历模板树，在其中进行数值绑定，生成虚拟DOM和结点，结点绑定到父节点上，虚拟DOM保存父节点的属性data\_visual\_dom中

组件渲染中面对不同的标签有不同的处理逻辑

**1.普通结点:** 普通结点指的除下列特殊结点外的结点，直接生成结点并返回

**2.v-if结点**：v-if结点根据逻辑判断渲染的分支，生成对应分支结点并返回

3.**v-for结点**：v-for结点遍历数组，并产生分支，生成对应的结点数组并返回

4.**v-mode结点**：v-model使用闭包包裹监听函数，生成结点并返回

5.**{{}}结点**：数值代替占位符，生成结点后返回

6.组件结点：数值代替占位符最为标签属性，再将其保存再props，触发子组件渲染函数

## 4.2.组件更新

绑定数据变化是，数据代理自动触发渲染器，渲染器触发组件更新函数，更新函数从父节点属性data\_visual\_dom获取虚拟dom。将虚拟dom的动态结点库和组件数据对比更新。

组件更新中面对不同的标签有不同的处理逻辑

**1.v-if结点**：v-if结点根据逻辑判断渲染的分支，若分支不变则更新分支。分支改变则卸载原分支，再挂载新分支。分支消失则卸载原分支。

2.**v-for结点**：v-for结点采用快速diff算法对比前后数组变化

3.**v-mode结点**：判断绑定函数时候变化，变化给闭包新监听函数，再赋值。

4.**{{}}结点**：数值代替占位符。

5.组件结点：一次遍历数组属性，判断数值是否变化。变化则修改子组件props数值，再触发子组件更新函数。

## 4.3.组件卸载

组件卸载从底至上卸载子组件，并以此删除父节点data\_visual\_dom属性

# 5.数值代理

## 1.proxy代理

Proxy代理数据，采用set和get函数，可以在绑定元素赋值和被访问时触发。绑定数据被访问时，将对应组件渲染函数保存到WeakMap-Map-Set的三层结构中，写作WeakMap(data, Map(属性名 Set(渲染函数)))，WeakMap保存组件的数值data，Map的键值对key(data属性名)—value(Set(渲染函数))，Set中保留渲染函数。

绑定数据被赋值时，获取该Set，并遍历Set依次触发渲染函数。

## 2.Watcher监听

Watcher参数为数据体（object）, 数据体属性名，绑定函数。会将该绑定函数以闭包的方式保存。Watcher触发时，会先触发绑定函数，再赋值。

# 6.路由管理

系统进入时先生成路由，路由采用单例模式。

## 6.1.路由表

路由表的结构为

{

path: '/',

component: LoginView,

meta: {

title: '欢迎登陆',

},

children: []

}

Path为路径，component为页面组件，meta为附加信息，childrem为子路由。

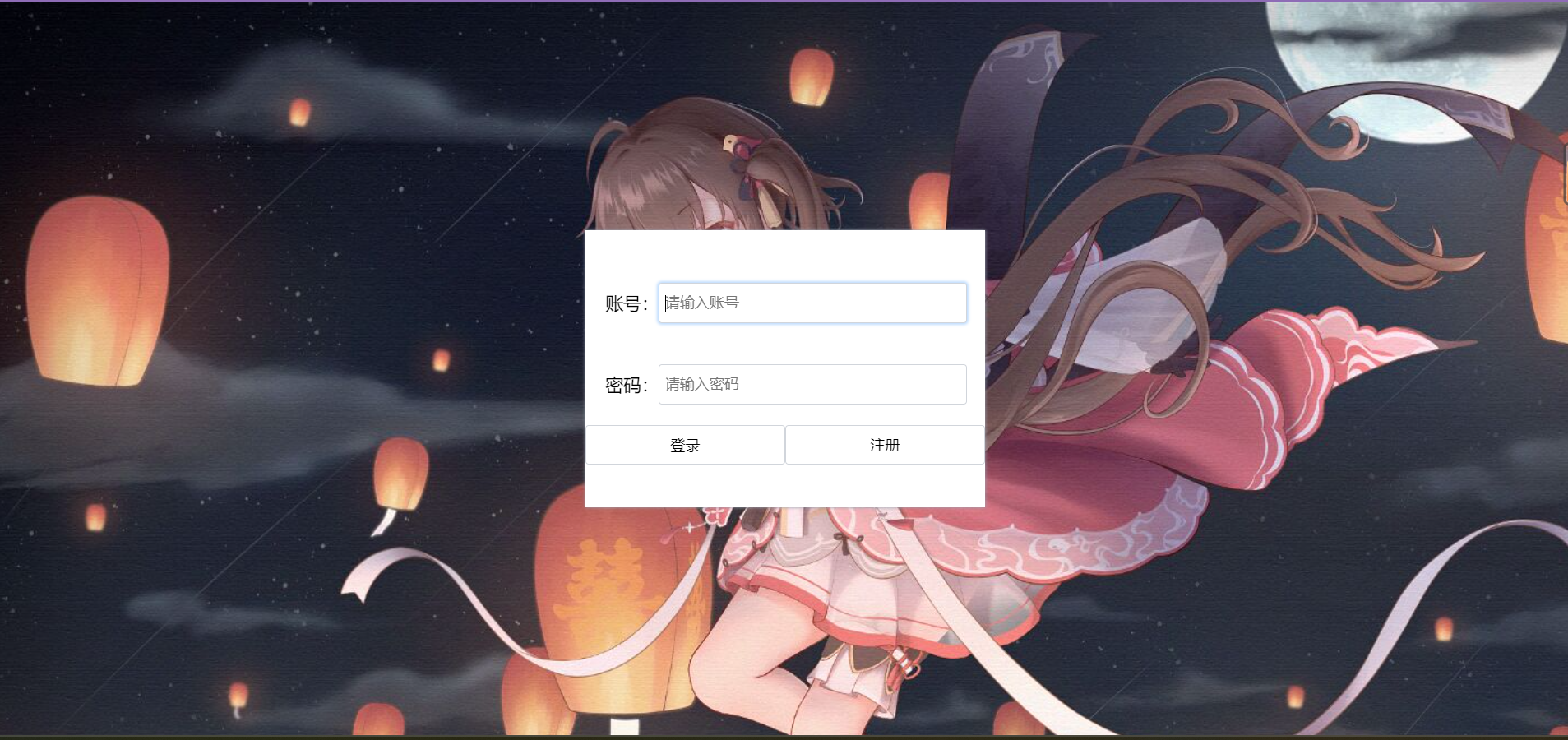
路由表为层级结构

## 6.2..路由

路由读取路由表并生成对象，该对象保留当前路由链接，并更具链接从左到右返回路由表中对应的路由组件。

# 7. 页面设计

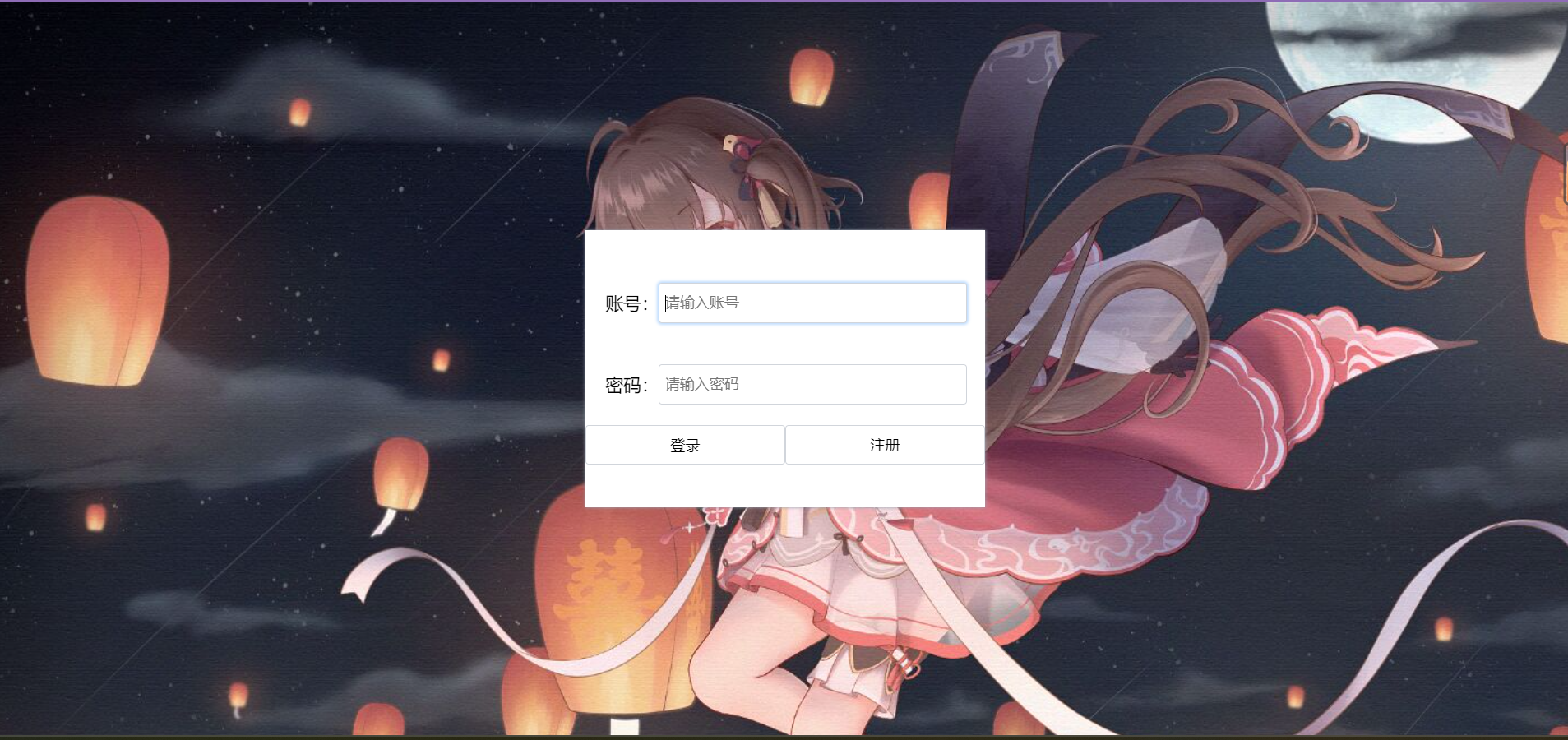
## 7.1.登录页面



整体采用了表单白色居中的设计

登陆时验证进行表单验证，用户的账号和密码需在6位以上

## 7.2.注册平台



整体采用了表单白色居中的设计

注册时验证进行表单验证，用户的账号和密码需在6位以上，并且两次密码需要相同

## 7.3.音乐平台



整体采用半透明居中设计

整体为黑色，字体为白色，按钮为红色和半白色。通过白色透明度创造不同的区域，整体给人一种磨砂的质感

# 8. 数据库设计

## 8.1.用户表

|  |  |  |
| --- | --- | --- |
| 列名 | 类型 | 备注 |
| Id | int | 主键 |
| username | Varchar(255) |  |
| password | Varchar(255) |  |
| salt | Varchar(255) |  |

## 8.2.歌单表

|  |  |  |
| --- | --- | --- |
| 列名 | 类型 | 备注 |
| Id | int | 主键 |
| name | Varchar(255) |  |
| uid | int | 外键，绑定用户表id |

## 8.3.歌曲信息表

|  |  |  |
| --- | --- | --- |
| 列名 | 类型 | 备注 |
| Id | int | 主键 |
| SongName | Varchar(255) |  |
| author | Varchar(255) |  |
| time | int |  |

## 8.3.歌曲表

|  |  |  |
| --- | --- | --- |
| 列名 | 类型 | 备注 |
| sid | int | 外键，绑定歌曲信息id |
| song | longtext |  |

# 9. 安全设计

1.前段路由做了路由守卫，根据用户是否登录和用户类型来判断用户是否能进入

2.后端采用了shiro，用户注册时shiro加密用户

7. 性能设计

后端采用redis缓存，保存用户信息

前端采用浏览器缓存，用户最近的音频文件保留在浏览器中，减少对服务器的访问