# 1、系统概述

**1.1、系统简介**

本系统目的是满足不具备专业音乐知识的人的音乐创作需求，本系统使用HMM（隐马尔可夫模型）在用户给出的音频文件和音律需求的基础上，自动预测生成合适的配乐文件。

术语说明：

* 依赖注入（dependency injection，缩写为 DI）是一种软件设计模式，也是实现控制反转的其中一种技术。此模式确保了任何想要使用给定服务的物件不需要知道如何建立这些服务。该设计的目的是为了分离关注点，分离接收方和依赖，从而提供松耦合以及代码重用性。
* 控制反转（Inversion of Control，缩写为IoC），是面向对象编程中的一种设计原则，可以用来减低计算机代码之间的耦合度。其中最常见的方式叫做依赖注入（Dependency Injection，简称DI），还有一种方式叫“依赖查找”（Dependency Lookup），本项目中使用DI。

**1.2、系统目标**

* 用户可以通过简单的操作即可为自己的音频文件生成合适的伴奏
* 用户可以方便的使用服务，包括网站和客户端等
* 用户可以安全的使用服务，包括数据传输的加密等
* 用户可以登陆，实现历史记录的记忆等
* 可以对高并发请求快速处理
* 后端实现完善的异常处理与日志记录

**1.3、系统运行环境**

包括对硬件平台、操作系统、数据库系统、编程平台、网络协议等的描述。

硬件平台：

前端：任何支持现代浏览器的硬件平台或运行主流操作系统的平台

后端：Linux/Windows10

操作系统：

客户端：理论上可以支持Linux，macOS，Windows等系统

服务端：支持Windows、Linux

数据库系统：

Mysql-8.0.30-0ubuntu0.20.04.2

编程框架：

前端：Vue.js、jQuery、layui

后端：Flask，Springboot

应用端：Electron

网络协议：

HTTP、HTTPS

**1.4、开发环境**

开发语言：

* Java：17.0.5 LTS
* Python：3.10.9
* JavaScript：> ES6
* Node.js：18.14.2

工程工具：

开发工具：

* Pycharm
* IntelliJ IDEA
* Visual Studio Code

构建工具：

1. Maven：3.8.1
2. Webpack

测试工具：

1. Junit：5.9.2
2. HTTP Client

代码管理工具：

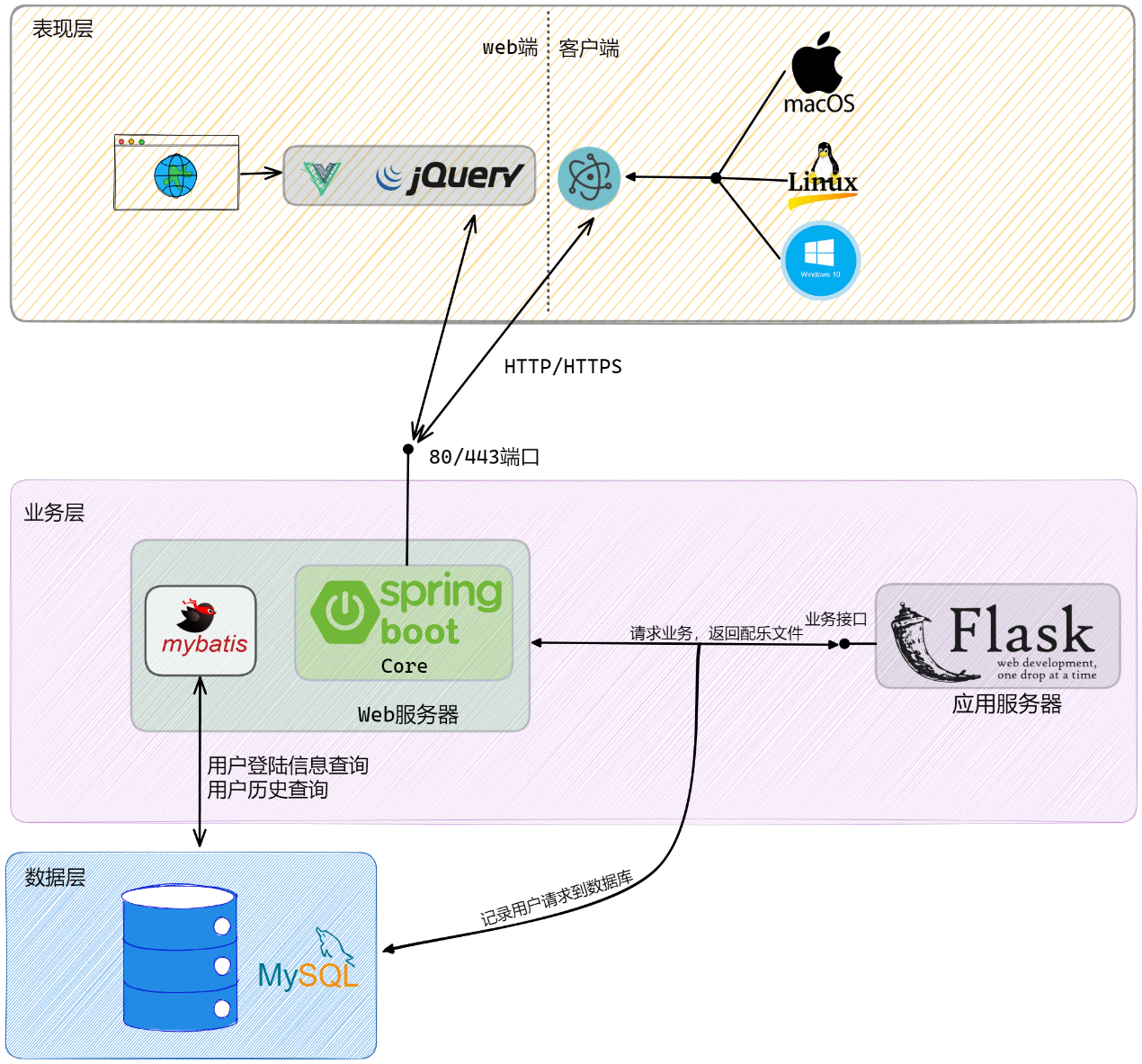
1. Git

文档生成工具：

1. Swagger3

# 2、总体结构设计

**2.1、软件结构**



图中各个模块之间均使用接口来传输数据和信息，模块之间耦合度低，后端使用Flask部署ai模型，其余全部使用Springboot完成，利用Springboot高解耦的特点，几乎所有配置均可在文件中修改，且使用DI、IoC技术来实现解耦。前端通过访问后端提供的端口获取数据，耦合度低，方便后端和前端的升级换代。

**2.2、设计思想**

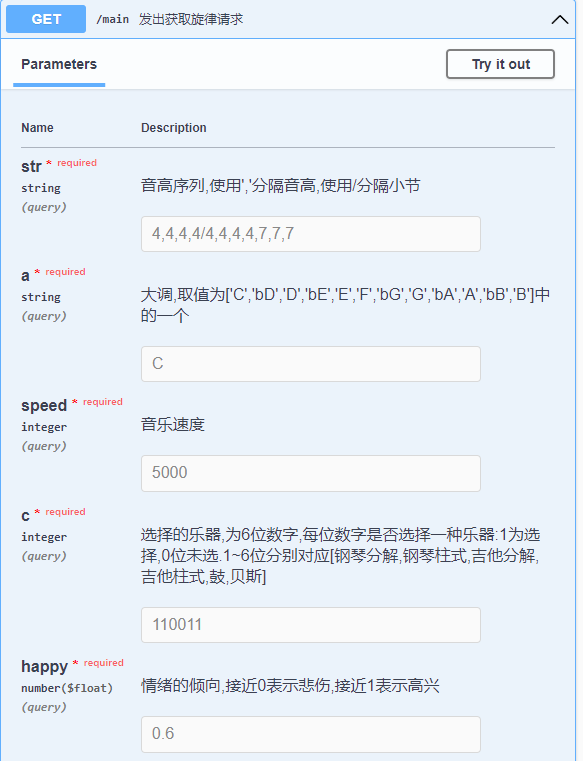
使用目前的主流后端架构，分为数据层，业务层，表现层等，各层之间使用接口进行数据交互，降低代码耦合度，应用服务器和数据库均部署在内网，用户仅能直接访问Web服务器，保证安全性。

# 3、模块设计

1. **旋律生成模块**
2. 功能描述

使用Python编写，使用Flask部署了AI模型，近提供一个业务接口，可根据输入的音高序列和其他音律设定生成配乐，并返回二进制数据。

1. 接口描述（OpenAPI3.0规范）





1. 数据结构描述

* HMM

内部实现viterbi或其他算法，可以从文件中读取权值，可根据已有音高生成预测旋律

* chord

一个用于生成三和弦的类，根据传入的和弦根音符、速度类型和和弦类型，生成一个三和弦，并存储三和弦的音符和速度。速度根据不同的速度类型生成，音符根据不同的和弦类型生成。

* track

可以读取chrod中的数据到音轨中，可根据音符序列，速度，乐器等生成MIDI文件。

1. 实现描述

隐马尔可夫模型、维特比算法：使用python根据数学理论实现。

MIDI文件生成：使用python的mido库实现。

MIDI转MP3：使用开源工具Timidity++将MIDI文件转为MP3，音色库来自互联网。

1. 其他

本模块使用Flask部署了一个AI模型，且只开放一个接口用于业务处理，本服务器应当部署在内网不对外开放，而是由Web服务器请求业务，以避免可能的安全风险。

1. **表现层模块**
2. 功能描述

用户直接接触到的模块，提供了人机交互界面，利用后端提供的接口实现了业务。预计提供网页端和客户端两种方式。

1. 实现描述

网页端：

* UI：使用layui和ElemenUI库
* 接口请求：使用jQuery和axios
* 网页框架、数据绑定与事件处理：Vue.js

应用端：

使用Electron技术在网页端基础上修改完善。

1. 其他

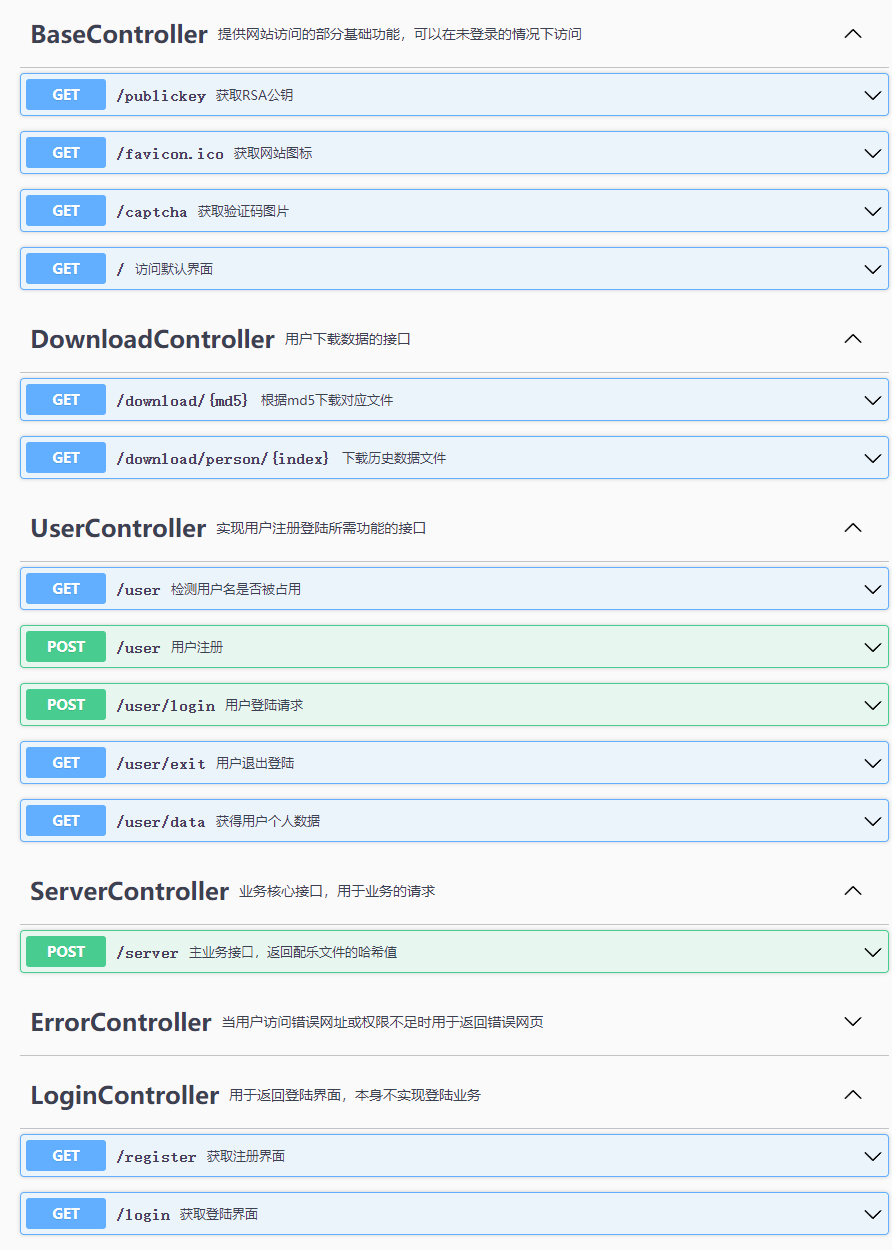
借助于Electron强大的跨平台能力，使得Electron可以同时构建支持Windows，Linux，macOS等平台的桌面端。从而减少工作量。

1. **Web服务器模块**
2. 功能描述

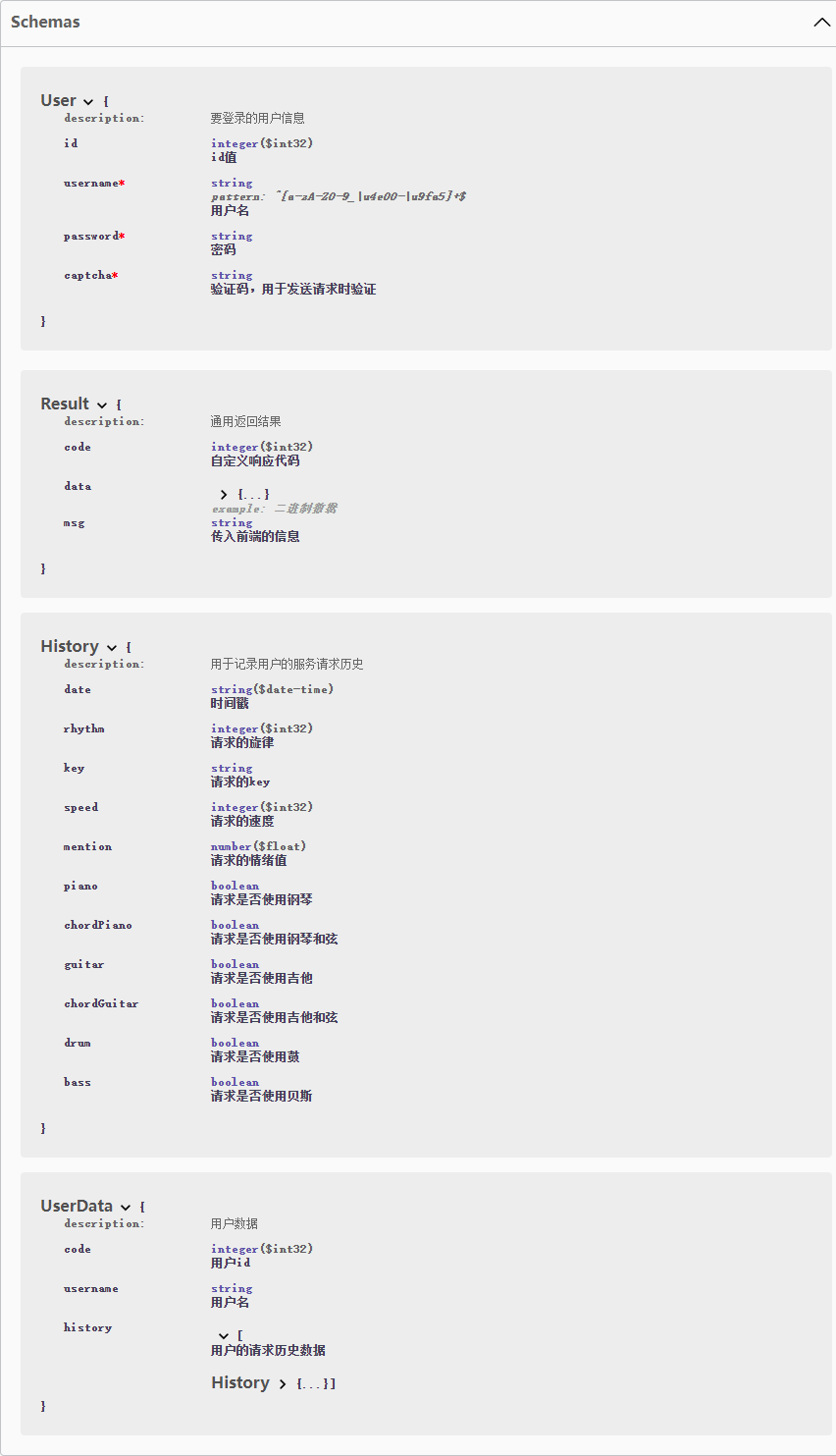
后端的核心模块，使用Java编写，使用springboot3框架，实现了后端数据校验，用户鉴权，注册登陆，数据库连接，用户业务处理，日志记录，加密传输，全局异常处理，页面返回等功能，编写并通过了测试用例

1. 接口描述（OpenAPI3.0规范）

鉴于完全展开将占据大量空间，只展示部分API的粗略介绍。



1. 数据结构描述



1. 实现描述

* 验证码：基于java.awt实现绘制字母与数字并进行旋转扭曲、添加干扰图案、导出图片等功能。后端实现验证码校验
* 数据加密：基于java.security包实现生成、保存、加载RSA密钥，以及数据加密解密等功能
* 数据库交互：使用Mybatis+Druid+jdbc进行数据交互
* 登陆注册、历史记录查询：从MySQL数据库中查询并处理数据
* 日志管理：slf4j+logback框架
* 提供访问接口、异常处理、DI、IoC、鉴权：Spring框架
* 网页部分的见表现层模块说明

1. 其他

本模块作为Web服务器运行，对公网开放，并负责处理用户的所有请求，内部使用Mybatis进行数据库查询，提升了安全性和性能。

# 4、数据库与数据结构设计

**4.1、 数据库及数据表**

使用的数据库为：Mysql-8.0.30

数据表设计：

tbl\_user表：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra | COMMENT |
| id | int | NO | PRI | null | auto\_increment | 用户id，唯一值 |
| username | varchar(50) | NO | UNI | null |  | 用户名，唯一值 |
| password | char(32) | NO |  | null |  | 用户密码，储存哈希值 |
| salt | char(32) | NO |  | null |  | 计算密码哈希时随机生成的盐 |

tbl\_user\_history表：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Field | Type | Null | Key | Default | Extra | Comment |
| history\_id | int | NO | PRI | null | auto\_increment | 历史记录id |
| applicant\_id | int | NO | MUL | null |  | 服务请求者id |
| date | datetime | NO |  | null |  | 服务请求时间 |
| rhythm | int | NO |  | null |  | 音乐节奏 |
| key | char(2) | NO |  | null |  | key |
| speed | int | NO |  | null |  | 速度 |
| mention | int | NO |  | null |  | 情绪,0-100 |
| piano | bit(1) | NO |  | b'0' |  | 钢琴分解 |
| chord\_piano | bit(1) | NO |  | b'0' |  | 钢琴柱式 |
| guitar | bit(1) | NO |  | b'0' |  | 吉他分解 |
| chord\_guitar | bit(1) | NO |  | b'0' |  | 吉他柱式 |
| drum | bit(1) | NO |  | b'0' |  | 鼓 |
| bass | bit(1) | NO |  | b'0' |  | 贝斯 |
| file\_name | varchar(100) | NO |  | null |  | 输入文件缓存路径 |
| file\_name\_md5 | varchar(32) | NO | MUL | null |  | 输入文件md5计算值 |

**4.2、 数据结构设计**

所有数据结构的详细信息均在上表中说明，tbl\_user表中每一条数据都对应一位用户的信息，tbl\_user\_history中的每一条数据都对应用户的一次请求。对于表格之间的关系，以主表tbl\_user的id为主键，从表tbl\_user\_history的applicant\_id为外键建立了外键约束，一个tbl\_user中的数据可以对应tbl\_user\_history中的多条数据。

**4.3、 数据存储设计**

所有数据及建立的索引的设计均在表中说明，读取时通过Mybatis读入所有数据到Java类中（密码仅在登陆校验时读取）并在服务器中处理。

对于tbl\_user表，为了保证安全性，在用户注册时会随机生成盐并进行多轮哈希以避免彩虹表攻击，数据库仅保存哈希后的值以避免泄露明文密码。对于tbl\_user\_history表，在applicant\_id上建立外键约束以避免删除tbl\_user中数据导致tbl\_user\_history表中出现废弃数据。

# 5、接口设计

**4.1、外部接口**

应用端将访问后端提供的接口，Web服务器开放80和443端口，并启用强制HTTPS。前端通过访问后端提供各类服务的接口以进行工作，接口的进一步信息请参见[3.3.2](#Web服务器模块的接口描述)节

**4.2、内部接口**

内部接口有数据库提供的连接接口与应用服务器提供的接口，前者用于数据交换，后者用于实现核心业务，接口的进一步信息请参见[3.1.2](#旋律生成模块的接口描述)节

# 6、其他设计

* 并行处理：使用springboot框架和flask框架自带的多线程功能
* 安全设计：敏感数据均隐藏在内网，对外服务器使用Shiro安全框架进行鉴权等。
* 内存管理：使用Druid数据库连接池，避免频繁建立释放数据库连接造成的内存开销，建立线程池，减少频繁建立释放线程的内存开销。
* 实时监控：使用springboot提供的Actuator功能，监控服务器运行状况。