《ToDo心理咨询项目系统》

详 细 设 计

v1.0

xxxx系统有限公司

**版 本 历 史**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本/状态 | 作者 | 参与者 | 日期 | 备注 |
| 1.0 | 刘骥飞、陈毅飞 | 华央恒、肖一帆、何正邦 | 2024.7.6 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目录

[第一部分 引言 7](#_Toc171150652)

[一、编写目的 7](#_Toc171150653)

[二、项目背景 7](#_Toc171150654)

[三、定义 7](#_Toc171150655)

[1、技术类 7](#_Toc171150656)

[2、业务类 7](#_Toc171150657)

[四、参考资料 7](#_Toc171150658)

[第二部分 项目概述 7](#_Toc171150659)

[第三部分 总体设计 7](#_Toc171150660)

[一、技术架构设计 7](#_Toc171150661)

[二、核心控制流程 7](#_Toc171150662)

[1、核心控制流程图 7](#_Toc171150663)

[2、核心控制流程说明 7](#_Toc171150664)

[第四部分 界面设计 7](#_Toc171150665)

[一、界面框架设计 8](#_Toc171150666)

[二、PC界面设计 8](#_Toc171150667)

[1、登录界面设计 8](#_Toc171150668)

[2、主体界面设计 8](#_Toc171150669)

[3、功能界面设计 8](#_Toc171150670)

[3.1资讯分享 8](#_Toc171150671)

[3.2专家列表 8](#_Toc171150672)

[3.3专家咨询 8](#_Toc171150673)

[第五部分 单元模块设计 9](#_Toc171150674)

[一、数据访问层设计 9](#_Toc171150675)

[1、类图设计 9](#_Toc171150676)

[2、类的详细设计描述 9](#_Toc171150677)

[2.1 接口设计 9](#_Toc171150678)

[2.2 接口实现类汇总 9](#_Toc171150679)

[2.2.1 a接口 9](#_Toc171150680)

[2.2.2 b接口 9](#_Toc171150681)

[二、业务逻辑层设计 9](#_Toc171150682)

[1、类图设计 9](#_Toc171150683)

[2、类的详细设计描述 9](#_Toc171150684)

[2.1接口设计 9](#_Toc171150685)

[2.2 业务接口实现类汇总 9](#_Toc171150686)

[2.2.1 a接口 9](#_Toc171150687)

[2.2.2 b接口 9](#_Toc171150688)

[三、系统管理 10](#_Toc171150689)

[第六部分 数据库设计 10](#_Toc171150690)

[一、数据库整体结构图 10](#_Toc171150691)

[二、系统管理 10](#_Toc171150692)

[1、WMS\_T\_U\_G表结构 10](#_Toc171150693)

[2、系统管理外键清单 10](#_Toc171150694)

[第七部分 补充设计和说明 10](#_Toc171150695)

[一、编译运行环境设计 10](#_Toc171150696)

[二、包路径与WEB目录结构设计 10](#_Toc171150697)

[1、包路径设计 10](#_Toc171150698)

[2、WEB目录结构设计 10](#_Toc171150699)

# 第一部分 引言

## 一、编写目的

本文档是心理咨询系统开发过程中的关键文档，详细记录了系统开发过程中的各个细节，为开发团队提供具体的设计指导，包括系统架构、用户界面、功能模块、数据流和接口等，确保所有参与者对项目有共同的理解。本文档主要由项目团队开发成员编写，读者主要包括项目团队成员、开发人员、测试人员以及其他与项目相关的利益相关者。

本文档内容包括引言部分，概述项目，总体设计，界面设计，单元模块设计，数据库设计和补充说明设计。具体包括但不限于项目开发背景，给出技术类和业务类的定义，设计技术架构，绘制核心控制流程图，设计界面，单元模块，数据库，并定义编译运行环境设计，设计保路径和WEB目录结构设计。为项目留下详细的记录，便于未来的维护、升级或复审。

## 二、项目背景

随着社会压力的增加和心理健康意识的普及，人们对心理健康服务的需求日益增长。根据“心理健康蓝皮书”《中国国民心理健康发展报告(2021~2022)》显示，国民心理健康情况仍然不容乐观，抑郁风险检出率为10.6%，焦虑风险检出率为15.8%。仍然有相当一部分的群体饱受心理健康的困扰。目前有99.2%的咨询师支持视频咨询。在简单心理近5年中发生的心理咨询中，视频咨询的比例也在逐年升高，2023年65.7%的心理咨询通过视频方式进行。用户渴望与心理咨询专家面对面沟通的需求正在日益上升。

《2023-2029年中国心理咨询行业市场深度分析及未来趋势预测报告》显示中国心理咨询行业市场规模约为638.1亿元，人均花费约为5944.6元。在用户群体如此庞大，心理咨询市场如此广阔的今天，我们遗憾的发现，相关的心理咨询工作者从专业水平和数量上都远远满足不了当下的需求，根据《2022年心理咨询工作者职业状况与心理健康状况调查报告》显示，心理咨询工作者的心理健康知识水平达标率仅为55.6%。不少工作者的年接案量超过3000小时，心理健康服务需求增加，人才数量紧缺，解决人们的心理健康问题，刻不容缓。

基于此，我们设计开发一款ToDo心理咨询系统，之于人工，ToDo心理咨询系统具备快速定位、强大存储、安全保密、维护成本低、使用期限长等，不受时间、区域、地点等优势。心理咨询师也能借助ToDo系统分析数据，及时沟通疏导，让心理咨询真正走进千家万户。

## 三、定义

### 1、技术类

### 2、业务类

## 参考资料

[1] 杨惠烽,徐莉.基于Android手机局部定位的心理咨询系统的设计与开发[J].晋中学院学报,2024,41(03):15-19.

# 项目概述

ToDo心理咨询系统是一个全面且综合的心理健康管理平台，旨在为用户提供一个集心理专家咨询、心理测试、解压工具和资讯分享于一体的在线环境。通过本系统，不同角色能够获得定制化的体验和服务。

根据不同身份的角色，系统支持多样化的需求，客户端分为用户端，专家端和管理端。不同客户端的功能概述如下：

## 2.1用户功能概述

用户可根据自身需求向相关心理专家资讯，业务流程包括选择对应的心理专家，用户可以选择查看专家预设好的问题与答案，如不符合自身需求，可付费与专家实时对话咨询，查看专家空闲时间段并进行预约。

用户可选择回答系统预设好的题库，在回答完全部问题后，后台会基于用户选择的答案给出对应的心理健康指数，分析用户心理状态和心理倾向，并给出适宜的建议和推荐相关研究方向的专家。

解压工具准备了包括但不限于解压小游戏，心理放松引导等多种途径，用户可以按照自己的喜好选择相对应的解压途径。

用户可查看每日的专家推荐的一些心理咨询小知识，获取更多的使用心理小妙招。

用户可管理自己的个人信息，例如添加自己的心理问题标签，更有针对性的推荐相对应的资讯和专家。

## 2.2专家功能概述

专家可以查看自己的用户预约业务，并根据自己的需求管理预约咨询的日程和记录，解决用户的心理问题。

专家将每次咨询过后的用户基本信息和咨询详情记录成日志，上传到平台内，以便下次用户咨询时专家调用记录，快速获取用户状态。

专家每日可推荐一些常见的心理咨询小知识，帮助用户走出常见的心理误区。

## 2.3管理员功能概述

系统支持管理员发布和管理资讯、测试、解压工具等内容。

管理员需要管理平台账号的基本信息，对不同身份的平台账号进行权限划分。

系统应收集用户反馈和使用数据，以便管理员进行系统优化。

管理员需要对专家进行专业认证和等级认证，以便帮助用户按需预约。

管理员需要监控与把握系统整体运行状态，及时更新、发布更新公告等，以便维护和改进用户体验。

管理员需要根据用户和专家的反馈，定制化系统功能和服务。

# 第三部分 总体设计

## 一、技术架构设计

开发工具：Intellij IDEA

依赖管理：Maven

数据库：MySQL

后端框架：Spring Boot+Mybatis

通信协议：WebScoket

第三方jar包：Java-WebScoekt

前后端数据交互格式：json

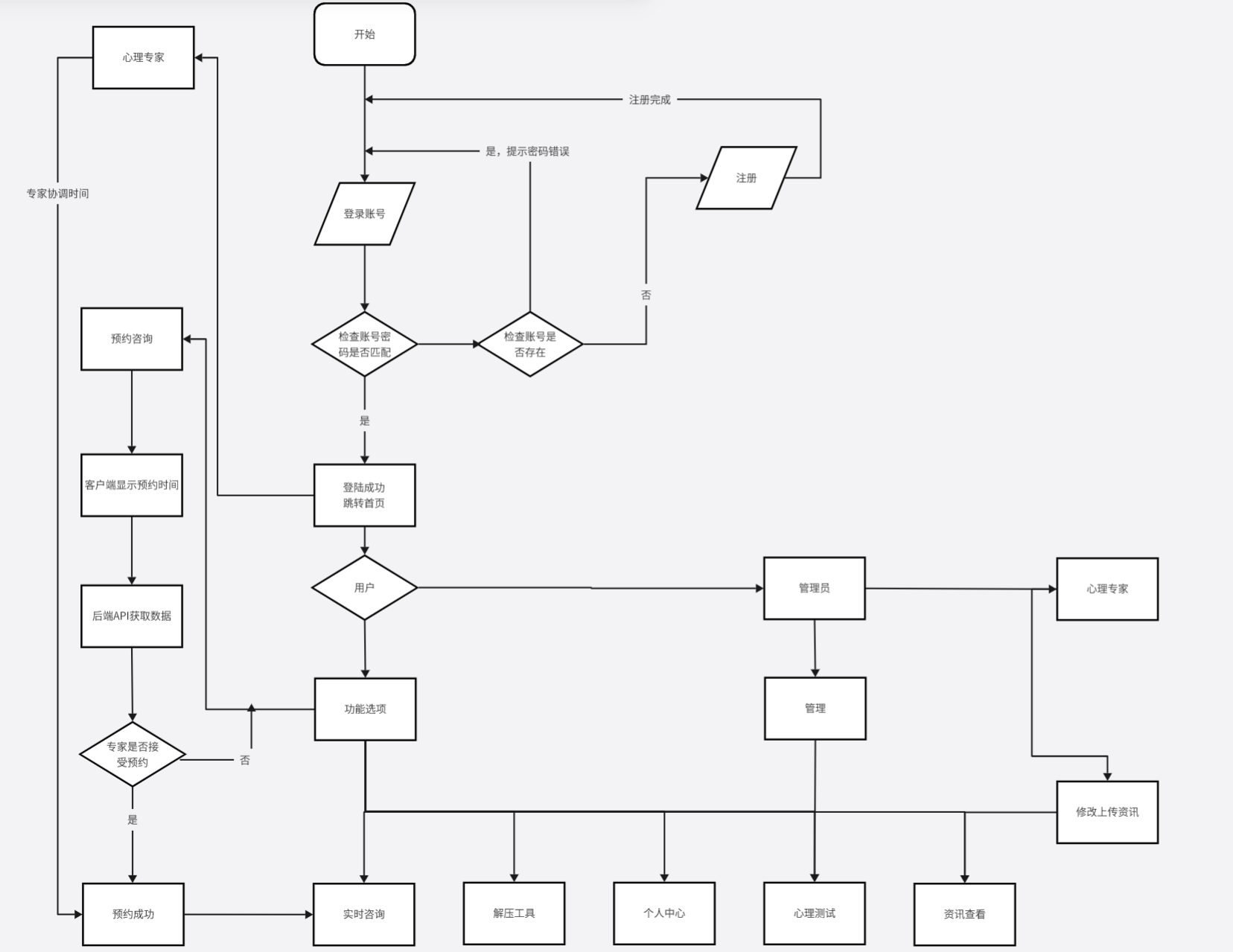
前端框架和工具：vue+Bootstrap

|  |  |
| --- | --- |
| 名称 | 描述 |
| Vue2.0 | 前端开发框架 |
| Uni-app、uni-ui | 可视化组件库 |
| Jdk8 | Java环境 |
| SpringBoot | 后端开发框架 |
| shiro+jwt | 用户鉴权框架 |
| Mybatis、Mybatis-plus | 数据持久层框架 |
| Redis | 高速缓存数据结构服务器 |
| MySQL 8.3.0 | 数据库 |
| Flask | 后端开发框架 |
| Nginx | 反向代理服务器 |
| Docker | 容器部署 |
| Docker Swarm | 集群管理工具 |

docker swarm进行集群部署，弹性收缩，分布式部署。Nginx负责反向代理。

## 二、核心控制流程

### 1、核心控制流程图



### 核心控制流程说明

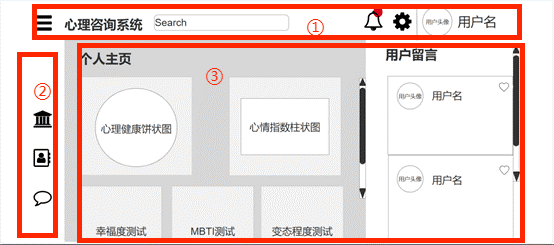
基于核心控制图，根据用户主功能设计。

# 第四部分 界面设计

本项目的界面设计通过清晰的布局和直观的交互设计，提供了极佳的用户体验。界面的淡色风格视觉效果良好，同时保持了一致的设计风格，使得用户能够快速上手并流畅地操作。同时浅色调通常给人一种清新、舒缓的感觉，有助于减轻用户的心理负担和压力，这对于心理咨询平台尤其重要。

## 一、界面框架设计

界面框架的设计遵循了一致性、简洁性、响应性、可用性。为系统奠定良好的基础，提供清晰、简洁、高效的用户体验，使用户在使用过程中感到愉快和满意。



①部分为标题和页头，显示当前页面的名称信息，帮助用户了解他们在系统中的位置；②部分为导航栏/侧边栏，提供主要的导航选项，帮助用户在系统的各个部分之间快速切换；③部分为内容，展示用户需要访问和操作的核心内容。

## 二、PC界面设计

### 1、登录界面设计

#### 1.1用户Pc端登录

WEB登录界面：



WEB注册界面：



WEB忘记密码界面：



#### 1.2专家登录

专家登录界面：



### 2、主体界面设计

#### 2.1个人中心

主界面（个人主页页面）：



个人信息页面：



个人预约记录页面：



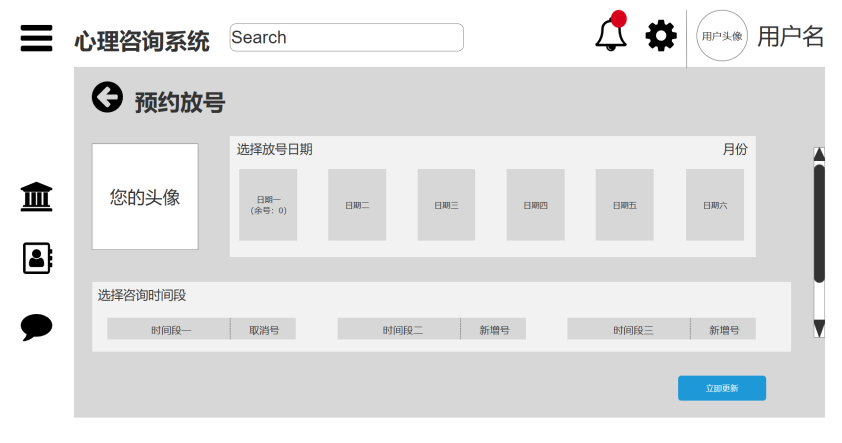
#### 2.2专家个人中心

个人主页，也是登录系统后进入的主界面：

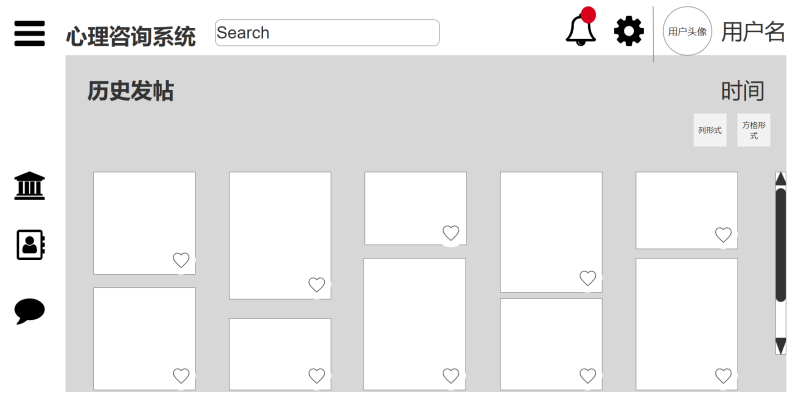


咨询广场与专家列表与用户端一致，此处不展示。接下来展示从主界面可进入的四个分界面：预约放号、您的发帖、发布咨询和历史订单。

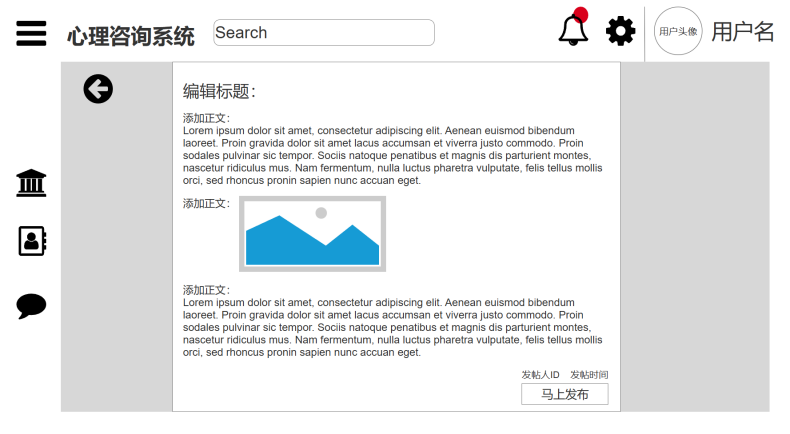
预约放号界面，获得专家号后可选择日期放号，进行心理咨询



您的发帖界面，展示了发布的所有心理资讯帖子



发布资讯界面，可再次发布心理相关的资讯



历史预约界面，可查看心理咨询预约记录



个人界面，可修改信息，展示专家个人风采：



#### 2.3管理员个人中心

管理员主界面：



从管理员端可进入以下界面：申请管理、系统公告、流量管理和内容管理。系统公告和专家端的发布咨询页面基本一致，不再赘述。

以下是申请管理界面：



以下为点开查看详细申请的界面:

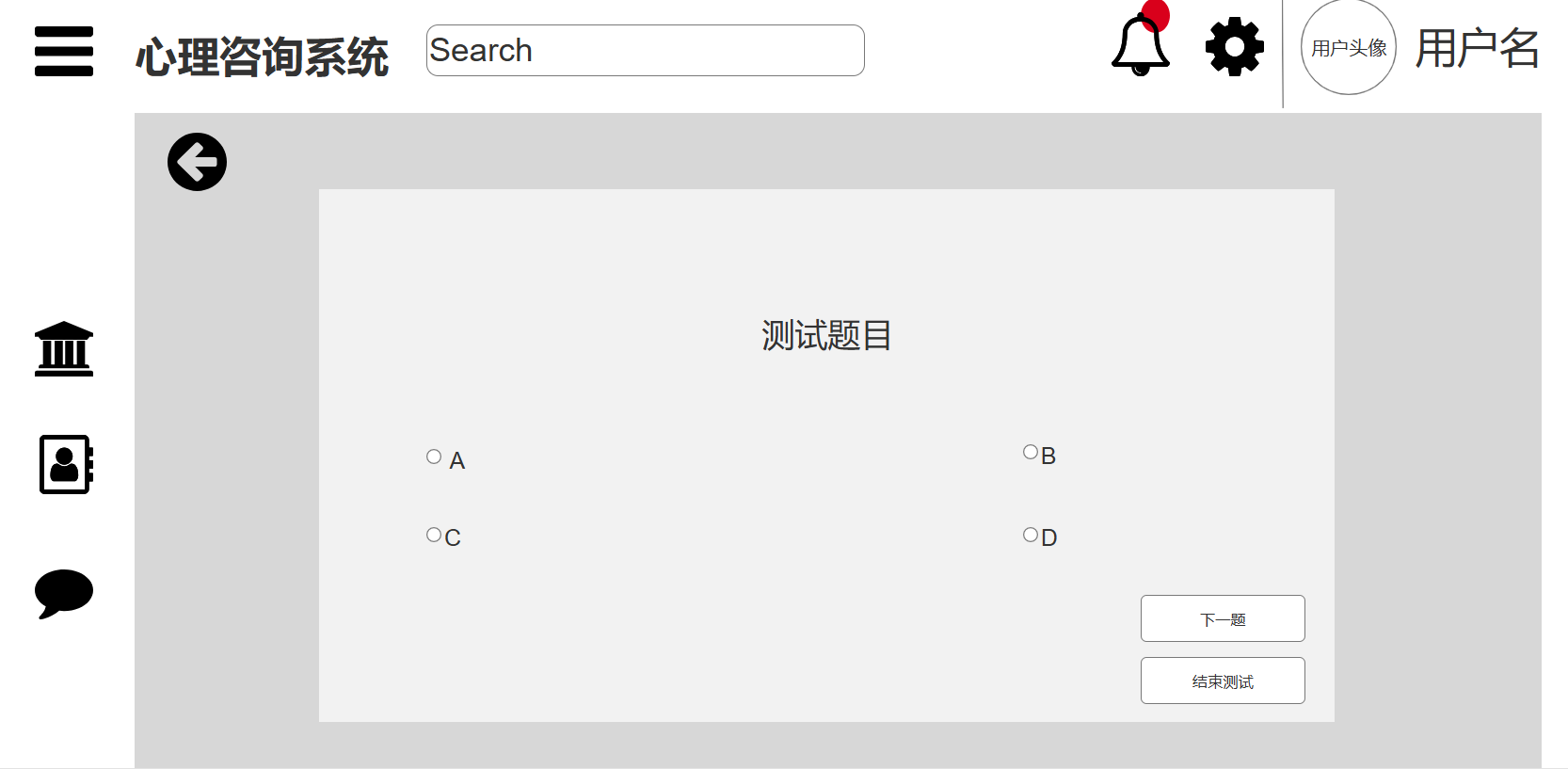


### 3、功能界面设计

#### 3.1心理测试

心理测试主界面：



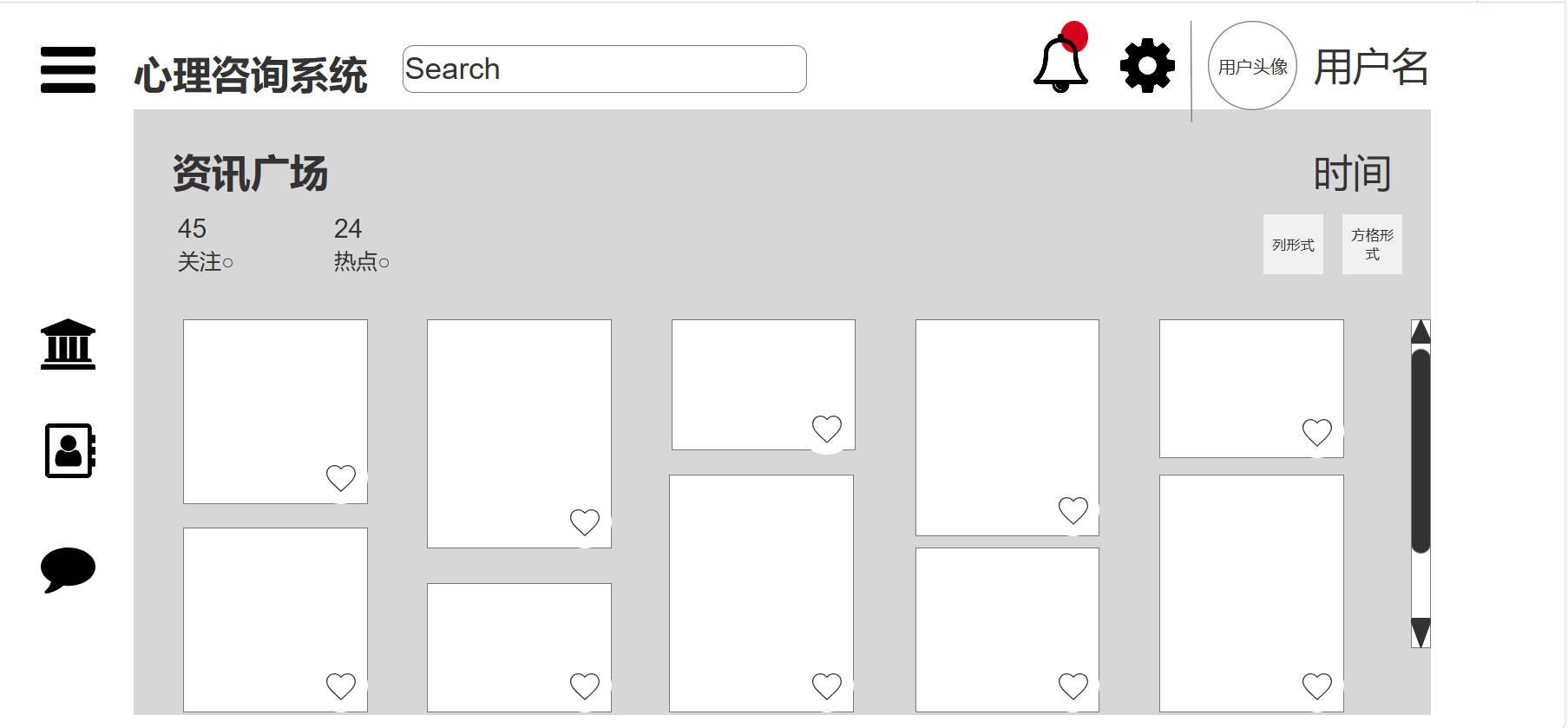
心理测试答题界面：

心理测试结果界面：



#### 3.2资讯广场

资讯广场界面：

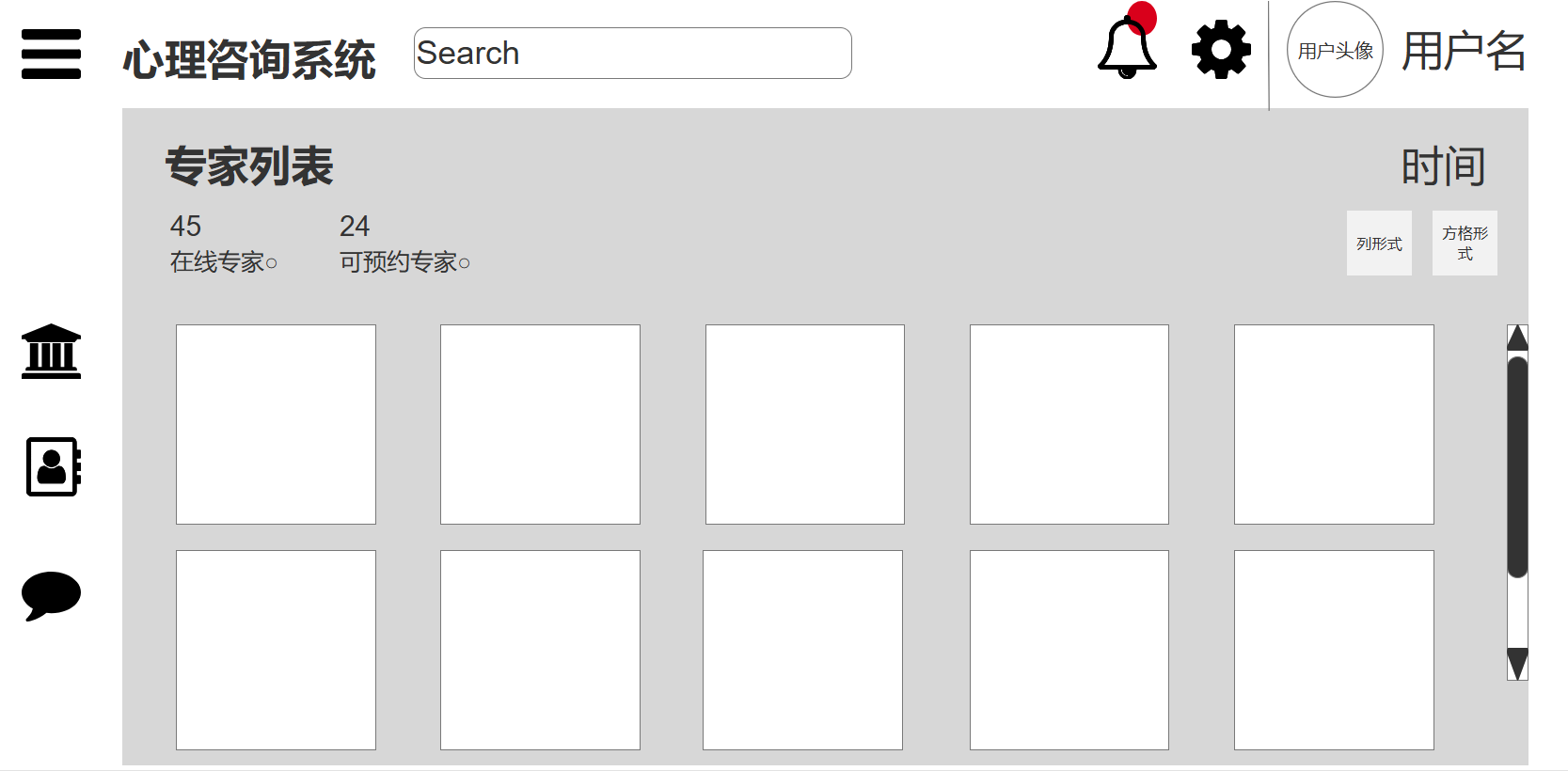


咨询内容界面：



#### 3.3专家咨询

专家咨询主页面（专家列表页面）：



专家详情页面：



#### 3.4心理预约

专家预约界面：



专家在线咨询界面：



#### 3.5解压工具

解压工具主界面：



解压内容界面：



解压结果界面：

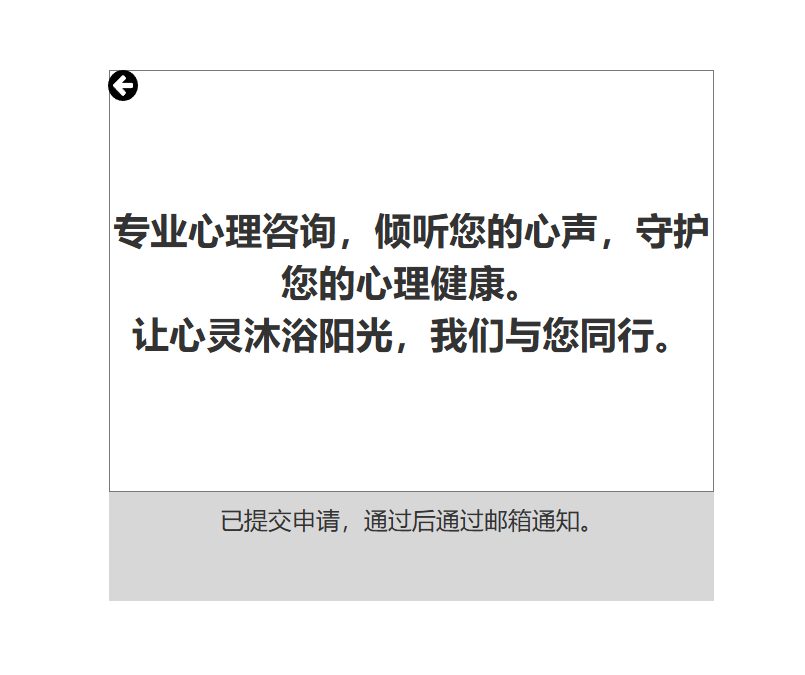


#### 3.6专家申请专家号

请专家号界面：



申请状态中界面：



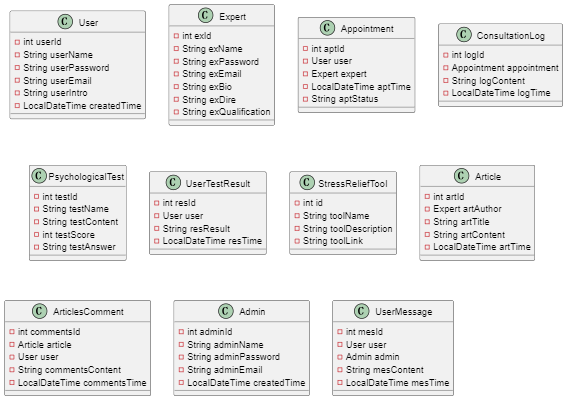
修改放号日期后的提示页面：



# 第五部分 单元模块设计

## 一、Entity层设计

### 1、类图设计

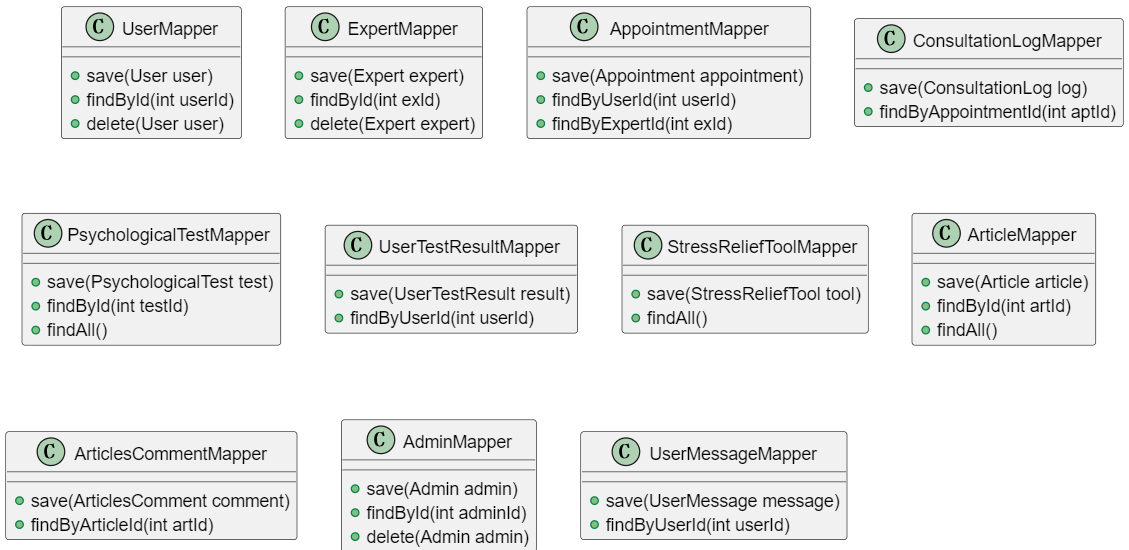


### 2、类的具体设计描述

|  |  |
| --- | --- |
| 实体类名称 | 实体类详细信息 |
| 用户实体类 | 用户实体类包含用户的基本信息，如用户ID、用户名、用户密码、用户邮箱、用户简介和创建时间。 |
| 专家实体类 | 专家实体类包含专家的基本信息，如专家ID、专家名称、专家密码、专家邮箱、专家简介、方向和资格。 |
| 预约实体类 | 预约实体类包含预约的基本信息，如预约ID、用户ID、专家ID、预约时间和预约状态。 |
| 咨询日志实体类 | 咨询日志实体类包含咨询日志的基本信息，如日志ID、预约ID、日志内容和日志时间。 |
| 心理测试实体类 | 心理测试实体类包含心理测试的基本信息，如测试ID、测试名称、测试内容、测试分数和测试答案 |
| 用户测试结果实体类 | 用户测试结果实体类包含用户的测试结果信息，如结果ID、用户ID、结果内容和结果时间。 |
| 解压工具实体类 | 解压工具实体类包含解压工具的基本信息，如工具ID、工具名称、工具描述和工具链接。 |
| 文章实体类 | 文章实体类包含文章的基本信息，如文章ID、作者ID、文章标题、文章内容和文章发布时间。 |
| 文章评论实体类 | 文章评论实体类包含文章评论的基本信息，如评论ID、文章ID、用户ID、评论内容和评论时间 |
| 管理员实体类 | 管理员实体类包含管理员的基本信息，如管理员ID、管理员名称、管理员密码、管理员邮箱和创建时间。 |
| 用户消息实体类 | 用户消息实体类包含用户消息的基本信息，如消息ID、用户ID、管理员ID、消息内容和消息时间。 |

## 二、Mapper层设计

#### 1、类图设计

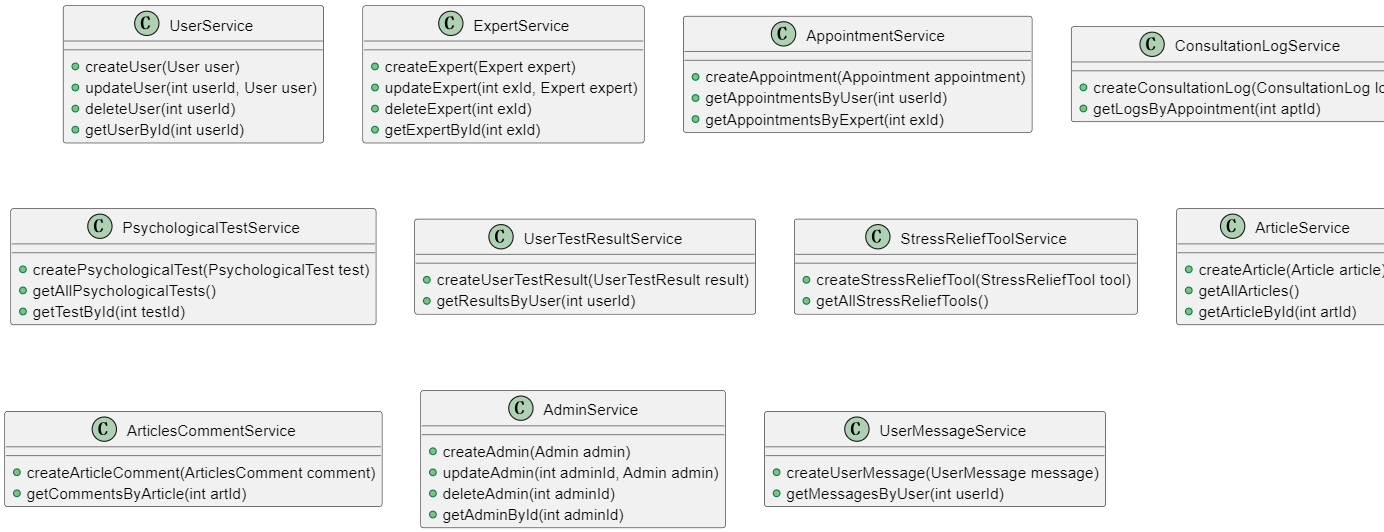


#### 2、类的接口设计描述

|  |  |
| --- | --- |
| mapper接口名称 | mapper接口详细信息 |
| 用户mapper接口 | 用户mapper接口定义了与用户相关的数据库操作，包括创建用户、更新用户、删除用户和根据ID查找用户。 |
| 专家mapper接口 | 专家mapper接口定义了与专家相关的数据库操作，包括创建专家、更新专家、删除专家和根据ID获取专家。 |
| 预约mapper接口 | 预约mapper接口定义了与预约相关的数据库操作，包括创建预约、根据用户ID获取预约和根据专家ID获取预约。 |
| 咨询日志mapper接口 | 咨询日志mapper接口定义了与咨询日志相关的数据库操作，包括创建咨询日志和根据预约ID获取咨询日志。 |
| 心理测试mapper接口 | 心理测试mapper接口定义了与心理测试相关的数据库操作，包括创建心理测试、获取所有心理测试和根据ID获取心理测试。 |
| 用户测试结果mapper接口 | 用户测试结果mapper接口定义了与用户测试结果相关的数据库操作，包括创建用户测试结果和根据用户ID获取测试结果。 |
| 解压工具mapper接口 | 解压工具mapper接口定义了与解压工具相关的数据库操作，包括创建解压工具和获取所有解压工具。 |
| 文章mapper接口 | 文章mapper接口定义了与文章相关的数据库操作，包括创建文章、获取所有文章和根据ID获取文章。 |
| 文章评论mapper接口 | 文章评论mapper接口定义了与文章评论相关的数据库操作，包括创建文章评论和根据文章ID获取评论。 |
| 管理员mapper接口 | 管理员mapper接口定义了与管理员相关的数据库操作，包括创建管理员、更新管理员、删除管理员和根据ID获取管理员。 |
| 用户消息mapper接口 | 用户消息mapper接口定义了与用户消息相关的数据库操作，包括创建用户消息和根据用户ID获取消息。 |

## 三、Service层设计

#### 1、类图设计



#### 2、类的详细设计描述

##### 2.1接口设计

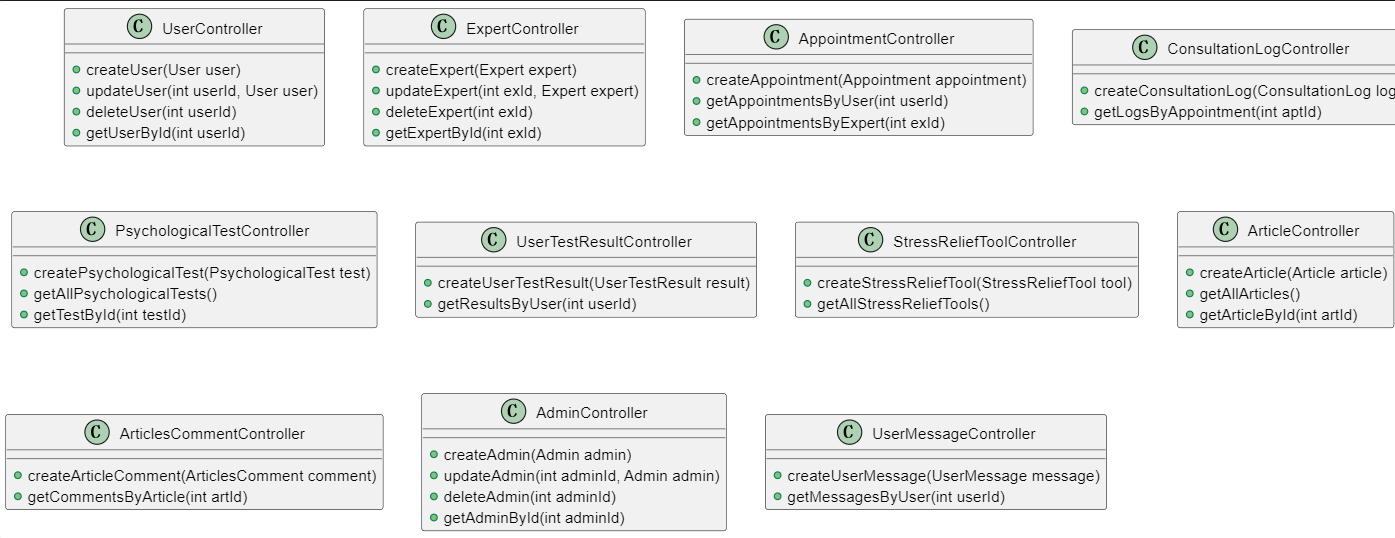
|  |  |
| --- | --- |
| 接口名称 | 接口详细信息 |
| 用户服务接口 | 用户服务接口定义了与用户相关的业务逻辑，包括创建用户、更新用户、删除用户和根据ID获取用户。 |
| 专家服务接口 | 专家服务接口定义了与专家相关的业务逻辑，包括创建专家、更新专家、删除专家和根据ID获取专家。 |
| 预约服务接口 | 预约服务接口定义了与预约相关的业务逻辑，包括创建预约、根据用户ID获取预约和根据专家ID获取预约。 |
| 咨询日志服务接口 | 咨询日志服务接口定义了与咨询日志相关的业务逻辑，包括创建咨询日志和根据预约ID获取咨询日志。 |
| 心理测试服务接口 | 心理测试服务接口定义了与心理测试相关的业务逻辑，包括创建心理测试、获取所有心理测试和根据ID获取心理测试。 |
| 用户测试结果服务接口 | 用户测试结果服务接口定义了与用户测试结果相关的业务逻辑，包括创建用户测试结果和根据用户ID获取测试结果。 |
| 解压工具服务接口 | 解压工具服务接口定义了与解压工具相关的业务逻辑，包括创建解压工具和获取所有解压工具。 |
| 文章服务接口 | 文章服务接口定义了与文章相关的业务逻辑，包括创建文章、获取所有文章和根据ID获取文章。 |
| 文章评论服务接口 | 文章评论服务接口定义了与文章评论相关的业务逻辑，包括创建文章评论和根据文章ID获取评论。 |
| 管理员服务接口 | 管理员服务接口定义了与管理员相关的业务逻辑，包括创建管理员、更新管理员、删除管理员和根据ID获取管理员。 |
| 用户消息服务接口 | 用户消息服务接口定义了与用户消息相关的业务逻辑，包括创建用户消息和根据用户ID获取消息。 |

##### 2.2接口实现类汇总

|  |  |
| --- | --- |
| 实现类名称 | 实现类详细信息 |
| 用户服务实现类 | 用户服务实现类实现了用户服务接口，包含用户的创建、更新、删除和查询的具体业务逻辑。 |
| 专家服务实现类 | 专家服务实现类实现了专家服务接口，包含专家的创建、更新、删除和查询的具体业务逻辑。 |
| 预约服务实现类 | 预约服务实现类实现了预约服务接口，包含预约的创建和查询的具体业务逻辑。 |
| 咨询日志服务实现类 | 咨询日志服务实现类实现了咨询日志服务接口，包含咨询日志的创建和查询的具体业务逻辑。 |
| 心理测试服务实现类 | 心理测试服务实现类实现了心理测试服务接口，包含心理测试的创建和查询的具体业务逻辑。 |
| 用户测试结果服务实现类 | 用户测试结果服务实现类实现了用户测试结果服务接口，包含用户测试结果的创建和查询的具体业务逻辑。 |
| 解压工具服务实现类 | 解压工具服务实现类实现了解压工具服务接口，包含解压工具的创建和查询的具体业务逻辑。 |
| 文章服务实现类 | 文章服务实现类实现了文章服务接口，包含文章的创建和查询的具体业务逻辑。 |
| 文章评论服务实现类 | 文章评论服务实现类实现了文章评论服务接口，包含文章评论的创建和查询的具体业务逻辑。 |
| 管理员服务实现类 | 管理员服务实现类实现了管理员服务接口，包含管理员的创建、更新、删除和查询的具体业务逻辑。 |
| 用户消息服务实现类 | 用户消息服务实现类实现了用户消息服务接口，包含用户消息的创建和查询的具体业务逻辑。 |

## 四、Controller层设计

#### 1、类图设计

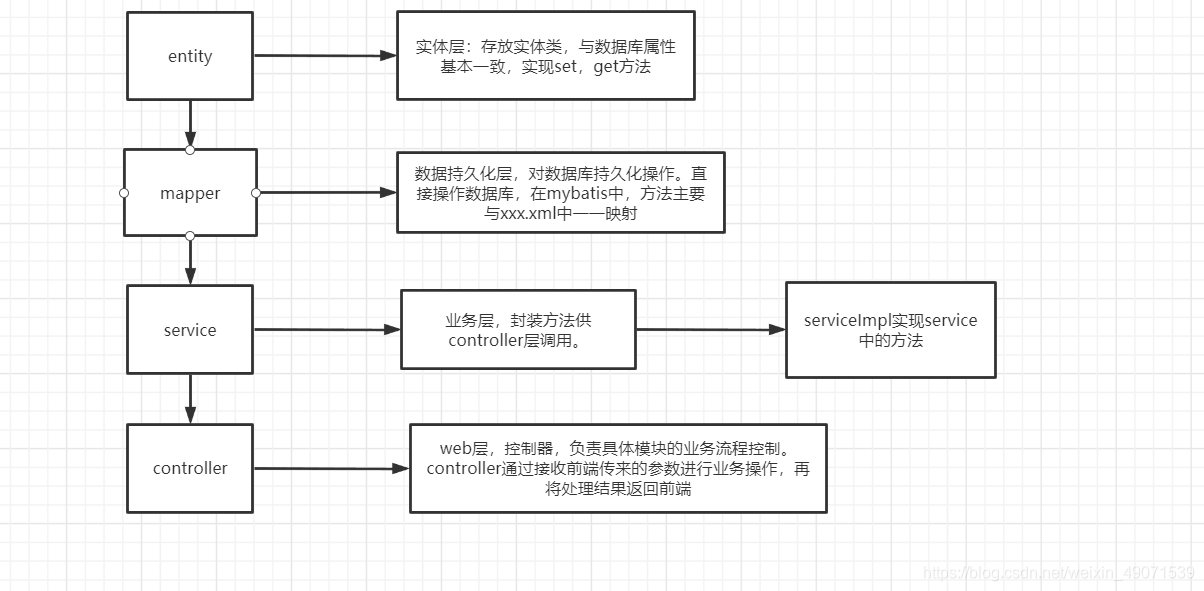


#### 2、类的接口设计描述

|  |  |
| --- | --- |
| controller类名称 | controller类详细信息 |
| 用户controller类 | 用户controller类定义了与用户相关的HTTP请求，包括创建用户、更新用户、删除用户和根据ID查找用户。 |
| 专家controller类 | 专家controller类定义了与专家相关的HTTP请求，包括创建专家、更新专家、删除专家和根据ID获取专家。 |
| 预约controller类 | 预约controller类定义了与预约相关的HTTP请求，包括创建预约、根据用户ID获取预约和根据专家ID获取预约。 |
| 咨询日志controller类 | 咨询日志controller类定义了与咨询日志相关的HTTP请求，包括创建咨询日志和根据预约ID获取咨询日志。 |
| 心理测试controller类 | 心理测试controller类定义了与心理测试相关的HTTP请求，包括创建心理测试、获取所有心理测试和根据ID获取心理测试。 |
| 用户测试结果controller类 | 用户测试结果controller类定义了与用户测试结果相关的HTTP请求，包括创建用户测试结果和根据用户ID获取测试结果。 |
| 解压工具controller类 | 解压工具controller类定义了与解压工具相关的HTTP请求，包括创建解压工具和获取所有解压工具。 |
| 文章controller类 | 文章controller类定义了与文章相关的HTTP请求，包括创建文章、获取所有文章和根据ID获取文章。 |
| 文章评论controller类 | 文章评论controller类定义了与文章评论相关的HTTP请求，包括创建文章评论和根据文章ID获取评论。 |
| 管理员controller类 | 管理员controller类定义了与管理员相关的HTTP请求，包括创建管理员、更新管理员、删除管理员和根据ID获取管理员。 |
| 用户消息controller类 | 用户消息controller类定义了与用户消息相关的HTTP请求，包括创建用户消息和根据用户ID获取消息。 |

## 五、系统管理

在一个典型的Java Spring应用程序中，Entity代表数据库中的表，通过ORM框架将Java对象映射到数据库；Mapper用于将数据库查询结果映射到实体类，通常使用MyBatis；Service封装业务逻辑，处理数据并调用Mapper接口；Controller处理客户端请求，调用Service层来执行具体的业务逻辑，最终将结果返回给客户端。这四层通过各自的职责分离，形成一个清晰的分层结构，便于维护和扩展。



在这个分层架构中，Entity、Mapper、Service和Controller通过各自的职责分工，形成了一个清晰的调用链：

Mapper接口使用Entity类来定义方法参数和返回值。Mapper的主要职责是将数据库中的数据转换为Entity对象，或将Entity对象存储到数据库中。这种关系使得数据访问层与业务逻辑层解耦，Mapper只关注数据的持久化和查询，而不涉及业务逻辑。

Service层调用Mapper接口来执行数据库操作。Service层封装了对Mapper的调用，并增加了业务逻辑、事务管理和数据处理。通过这种关系，Service层可以集中处理业务逻辑，而将数据操作的细节交给Mapper层处理。

Controller层调用Service层来处理客户端请求。Controller负责接收请求参数，并将这些参数传递给Service层进行业务处理，最后将结果返回给客户端。Service层的存在使得Controller层可以保持简洁，专注于请求处理和响应生成，而不需要处理复杂的业务逻辑。

Controller与前端交互，接收HTTP请求并返回HTTP响应。Controller将前端传递的参数转换为内部使用的数据格式，并调用Service层进行处理。处理完成后，Controller将结果转换为前端需要的格式（如JSON或HTML视图）并返回给客户端。Controller层的设计确保了前端和后端的通信简洁和高效。

通过这种分层结构，每个层次的职责变得更加明确，从而使代码更容易维护和扩展。各层之间通过依赖注入和接口调用实现松耦合，这不仅提高了代码的重用性，还使得各层可以独立测试和开发。这种架构模式广泛应用于企业级应用开发中，提供了良好的扩展性和可维护性。

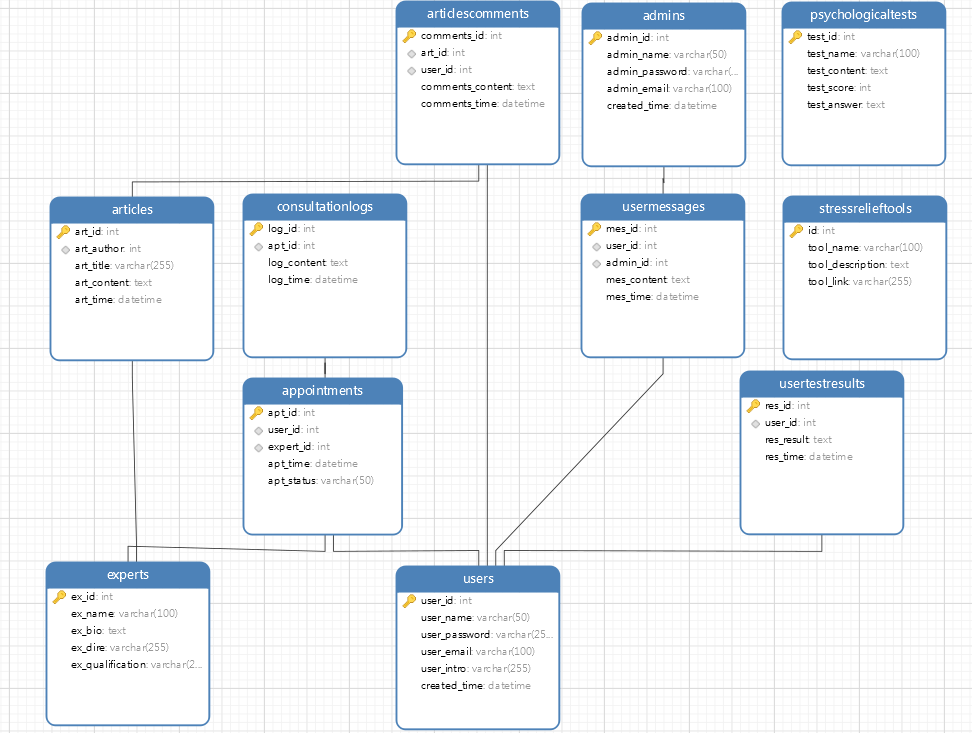
# 第六部分 数据库设计

## 一、数据库整体结构图

数据库由11张表构成，分别为用户表(Users)、心理专家表(Experts)、预约表(Appointments)、咨询日志表(ConsultationLogs)、心理测试题库表(PsychologicalTests)、心理测试结果表(UserTestResults)、解压工具表(StressReliefTools)、专家资讯表(Articles)、资讯留言表(ArticlesComments)、管理员表(Admins)和用户消息中心表(UserMessages)。每张表都有主键，大部分表还与其他表有外键关联，用以保证数据的完整性和一致性。每张表的结构清晰，字段定义明确，主要表之间通过外键建立了关联，确保了数据的联动性和一致性。

第三范式 (3NF) 是数据库设计中的一种标准，用于确保数据的结构化和减少数据冗余。在这些表中非主键字段都直接依赖于主键，没有冗余的依赖关系存在，满足第三范式要求。这样的设计大大减少了数据冗余，避免了数据重复存储带来的浪费。同时也确保了数据的一致性和完整性，因为每个非主键字段都直接依赖于主键，减少了数据异常的可能性。结构清晰、规范的数据库更易于理解和修改，简洁的表结构意味着更快的数据检索和更新操作，这也提升了数据库的可维护性和查询效率。此外，第三范式的设计有助于避免更新和删除操作中的异常情况，从而增强了数据库的可靠性和稳定性。

数据库的整体结构如下：



## 二、数据库详细设计

### 1、详细表结构设计

#### 1.1 table1 Users 用户表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **属性名** | **字段名** | **数据类型** | **长度** | **主键/外键** |
| 用户ID | user\_id | INT | 11 | 主键 |
| 用户名 | user\_name | VARCHAR | 50 |  |
| 密码 | user\_password | VARCHAR | 255 |  |
| 邮箱 | user\_email | VARCHAR | 100 |  |
| 病情简介 | user\_intro | VARCHAR | 255 |  |
| 注册时间 | created\_time | DATETIME |  |  |

#### 1.2 table2 Experts 心理专家表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **属性名** | **字段名** | **数据类型** | **长度** | **主键/外键** |
| 专家ID | ex\_id | INT | 11 | 主键 |
| 姓名 | ex\_name | VARCHAR | 100 |  |
| 简介 | ex\_bio | TEXT |  |  |
| 主治方向 | ex\_dire | VARCHAR | 255 |  |
| 资历 | ex\_qualification | VARCHAR | 255 |  |

#### 1.3 table3 Appointments 预约表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **属性名** | **字段名** | **数据类型** | **长度** | **主键/外键** |
| 预约ID | apt\_id | INT | 11 | 主键 |
| 用户ID | user\_id | INT | 11 | 外键 (References Users.user\_id) |
| 专家ID | expert\_id | INT | 11 | 外键 (References Experts.ex\_id) |
| 预约时间 | apt\_time | DATETIME |  |  |
| 预约状态 | apt\_status | VARCHAR | 50 |  |

#### 1.4 table4 ConsultationLogs 咨询日志表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **属性名** | **字段名** | **数据类型** | **长度** | **主键/外键** |
| 咨询日志ID | log\_id | INT | 11 | 主键 |
| 预约ID | apt\_id | INT | 11 | 外键 (References Appointments.apt\_id) |
| 日志内容 | log\_content | TEXT |  |  |
| 创建时间 | log\_time | DATETIME |  |  |

#### 1.5 table5 PsychologicalTests 心理测试题库表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **属性名** | **字段名** | **数据类型** | **长度** | **主键/外键** |
| 测试题目ID | test\_id | INT | 11 | 主键 |
| 题目名称 | test\_name | VARCHAR | 100 |  |
| 题目内容 | test\_content | TEXT |  |  |
| 题目分值 | test\_score | INT | 11 |  |
| 题目答案 | test\_answer | TEXT |  |  |

#### 1.6 table6 UserTestResults 心理测试结果表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **属性名** | **字段名** | **数据类型** | **长度** | **主键/外键** |
| 测试结果ID | res\_id | INT | 11 | 主键 |
| 用户ID | user\_id | INT | 11 | 外键 (References Users.id) |
| 测试结果 | res\_result | TEXT |  |  |
| 测试时间 | res\_time | DATETIME |  |  |

#### 1.7 table7 StressReliefTools 解压工具表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **属性名** | **字段名** | **数据类型** | **长度** | **主键/外键** |
| 解压工具ID | id | INT | 11 | 主键 |
| 工具名称 | tool\_name | VARCHAR | 100 |  |
| 工具描述 | tool\_description | TEXT |  |  |
| 工具链接 | tool\_link | VARCHAR | 255 |  |

#### 1.8 table8 Articles专家资讯表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **属性名** | **字段名** | **数据类型** | **长度** | **主键/外键** |
| 咨询ID | art\_id | INT | 11 | 主键 |
| 发布人 | art\_author | INT | 11 | 外键 (References Experts.ex\_id) |
| 标题 | art\_title | VARCHAR | 255 |  |
| 内容 | art\_content | TEXT |  |  |
| 创建时间 | art\_time | DATETIME |  |  |

#### 1.9 table9 Articlescomments资讯留言表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **属性名** | **字段名** | **数据类型** | **长度** | **主键/外键** |
| 留言ID | comments\_id | INT | 11 |  |
| 咨询ID | art\_id | INT | 11 | 外键 (References Articles.art\_id) |
| 发布人ID | user\_id | INT | 11 | 外键 (References Users.user\_id) |
| 内容 | comments\_content | TEXT |  |  |
| 创建时间 | comments\_time | DATETIME |  |  |

#### 1.10 table10 Admins 管理员

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **属性名** | **字段名** | **数据类型** | **长度** | **主键/外键** |
| 管理员ID | admin\_id | INT | 11 | 主键 |
| 用户名 | admin\_name | VARCHAR | 50 |  |
| 密码 | admin\_password | VARCHAR | 255 |  |
| 邮箱 | admin\_email | VARCHAR | 100 |  |
| 创建时间 | created\_time | DATETIME |  |  |

#### 1.11 table11 UserMessages用户消息中心

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **属性名** | **字段名** | **数据类型** | **长度** | **主键/外键** |
| 消息ID | mes\_id | INT | 11 |  |
| 用户ID | user\_id | INT | 11 | 外键 (References Users.user\_id) |
| 发布管理员ID | admin\_id | INT | 11 | 外键 (References Admins.admin\_id) |
| 内容 | mes\_content | TEXT |  |  |
| 创建时间 | mes\_time | DATETIME |  |  |

### 2、数据库外键清单

|  |  |  |  |
| --- | --- | --- | --- |
| 表名 | 字段 | 被引用的表（父） | 被引用的字段 |
| appointments | user\_id | users | user\_id |
| appointments | ex\_id | experts | ex\_id |
| articles | art\_author | experts | ex\_id |
| articlescomments | art\_id | artticles | art\_id |
| articlescomments | user\_id | users | user\_id |
| consultationlogs | apt\_id | appointments | apt\_id |
| usermessages | user\_id | users | user\_id |
| usermessages | admin\_id | admins | admin\_id |
| usertestresults | user\_id | users | user\_id |

# 第七部分 补充设计和说明

## 一、编译运行环境设计

编译运行环境的设计对项目的成功至关重要。一个精心设计的编译运行环境能够确保开发团队在统一的标准下进行开发，从而避免由于环境差异引起的兼容性问题。通过提供详细的环境配置和依赖管理，可以在不同的机器上重复构建和运行项目，从而提高可重复性。此外，通过严格的依赖管理和环境配置，编译运行环境能够有效减少安全漏洞，确保系统的安全性和可靠性。这对于需要处理敏感数据的项目尤为重要，因为它能够确保数据的安全性。

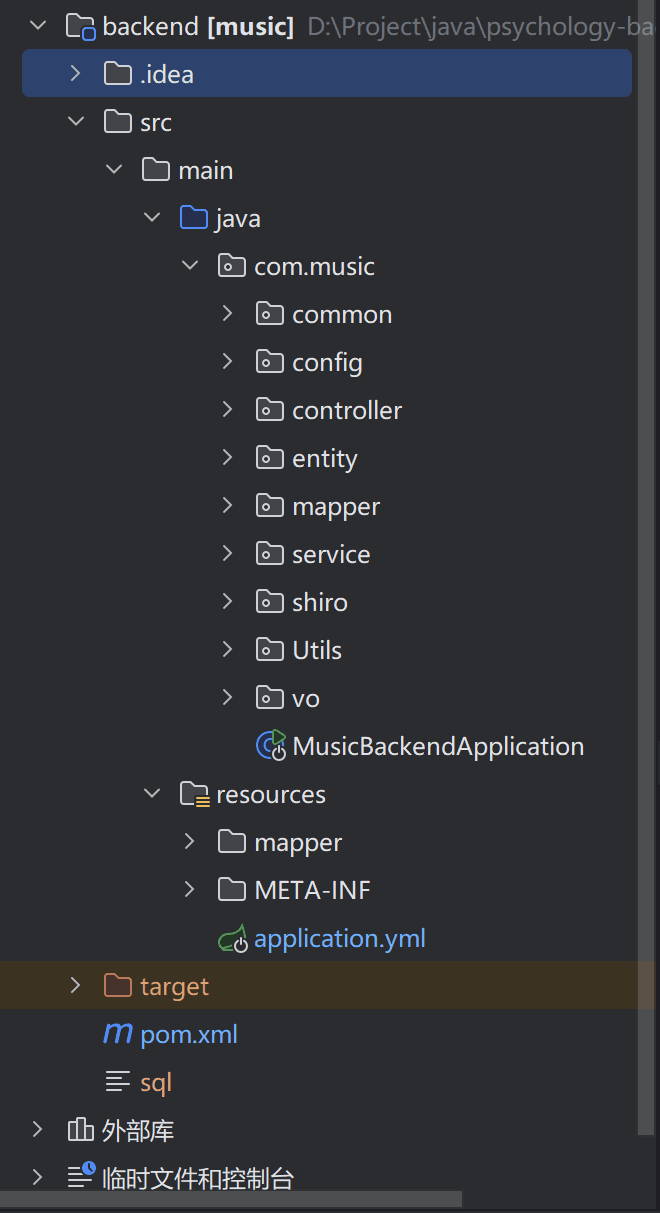
本系统的具体编译运行环境如下：

|  |  |
| --- | --- |
| 项目名称 | 具体描述 |
| 操作系统 | Windows11 |
| 编程语言 | java 8、vue 2.0 |
| IDE | Intellij IDEA |
| 数据库 | MySQL |
| 后端框架 | Spring Boot+Mybatis |
| 前端框架和工具 | vue+Bootstrap |
| 前后端数据交互格式 | json |
| 通信协议 | WebScoket |
| 依赖管理 | Maven |
| 构建工具 | Maven |
| 版本控制系统 | gitee |
| 依赖库 |  |
| 运行环境 | 阿里云服务器 ECS 2核4G 香港服务器 带宽5Mbps  公网ip地址 8.217.178.86  操作系统版本 Ubuntu20.04 x64 |
| 部署工具 | Docker 26.1.2 |

## 二、包路径与WEB目录结构设计

### 1、后端包路径设计

系统的后端包路径设计结构清晰，采用模块化的层次结构，将控制层、服务层、数据访问层、模型层和配置层分离，命名规范且语义明确。这样的设计有助于维护和扩展，依赖关系明晰，能够有效减少耦合度，对性能影响较小。详细后端包路径如下：



### 2、前端WEB目录结构设计

前端WEB目录结构设计清晰合理，将不同功能模块分离，层次分明，便于理解和导航。目录和文件命名规范，语义明确，使得项目易于维护和扩展。这样的设计有助于提高开发效率和代码的可读性，且新功能的添加不会破坏现有结构，具有良好的灵活性和可维护性。具体前端WEB目录结构设计如下：