****

**软件工程实践（二）**

|  |  |
| --- | --- |
| **学 院** | **计算机与人工智能学院** |
| **专 业** | **软件工程** |
| **班 级** | **软件2002** |
| **组 长** | **周盟** |
| **指导教师** | **唐祖锴** |
| **小组成员** | **周盟、毕列祥、曾俊博、曾梓迪** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2023 | 年 | 06 | 月 | 24 | 日 |

目录

[课程任务 5](#_Toc138534811)

[任务目的 5](#_Toc138534812)

[任务内容 5](#_Toc138534813)

[任务要求 5](#_Toc138534814)

[任务输出 7](#_Toc138534815)

[程序设计与开发 8](#_Toc138534816)

[World of zuul 架构设计 8](#_Toc138534817)

[系统概述： 8](#_Toc138534818)

[架构设计目标： 8](#_Toc138534819)

[技术选型： 8](#_Toc138534820)

[前端架构设计： 8](#_Toc138534821)

[后端架构设计： 8](#_Toc138534822)

[前后端交互： 9](#_Toc138534823)

[数据库设计： 9](#_Toc138534824)

[分工与协作： 9](#_Toc138534825)

[自动化流程： 9](#_Toc138534826)

[开发版本计划与任务分派 10](#_Toc138534827)

[Version 0.1： 10](#_Toc138534828)

[Version 0.2： 10](#_Toc138534829)

[Version 0.3： 10](#_Toc138534830)

[Version 0.4： 11](#_Toc138534831)

[Version 1.0： 11](#_Toc138534832)

[任务分派 11](#_Toc138534833)

[架构设计文档编写 11](#_Toc138534834)

[前端界面编写 - 登录、注册、忘记密码部分 11](#_Toc138534835)

[前端界面编写 - 游戏界面部分 11](#_Toc138534836)

[后端核心游戏逻辑编写 12](#_Toc138534837)

[数据库设计与前后端数据交互格式设计 12](#_Toc138534838)

[测试与问题记录 12](#_Toc138534839)

[开发分支模型与代码合并 13](#_Toc138534840)

[开发规范与测试 14](#_Toc138534841)

[开发规范： 14](#_Toc138534842)

[测试策略： 14](#_Toc138534843)

[集成与版本发布 15](#_Toc138534844)

[小组成员个人任务 15](#_Toc138534845)

# 课程任务

## 任务目的

* 巩固强化软件编程规范
* 提高面向对象软件建模与抽象能力
* 培养小组协同开发能力
* 掌握基于Maven的软件项目管理机制
* 掌握基于Github的小组协同开发工具和平台
* 了解DevOps软件开发流程

## 任务内容

* 创建软件开发小组
* 针对样例代码工程进行小组讨论，确定功能扩充需求点
* 基于Github中的issue管理功能明确工作任务并为组员分配工作任务
* 基于小组商定的分支模型进行软件功能开发，并按开发流程进行代码测试、提交、归并和同步
* 代码提交到远程仓库后，应进行自动化代码格式规范检查和测试以确保功能符合需求设计
* 完成前述各项任务后，可尝试进行代码自动化打包，自动生成可供执行的jar文件

## 任务要求

1. 创建软件开发小组
   * 每个开发小组人数3-5人，推选一人作为组长，负责组织、协调和领导团队开发；
   * 所有小组成员应按操作步骤在github开发平台上加入同一小组，共用同一代码仓库；
2. 开展小组讨论，确定功能扩充点
   * 样例工程“world-of-zuul”具备最基本的程序功能，该项目具有极大的扩展空间，开发小组内可进行沟通讨论，确定系统结构优化需求或功能扩充需求，结构优化或功能扩充项不能少于5项；

可供参考的结构优化或功能扩充项包括但不限于以下内容：

1. 扩展游戏，使得一个房间里可以存放任意数量的物件，每个物件可以有一个描述和一个重量值，玩家进入一个房间后，可以通过“look”命令查看当前房间的信息以及房间内的所有物品信息；
2. 在游戏中实现一个“back”命令，玩家输入该命令后会把玩家带回上一个房间；
3. 在游戏中实现一个更高级的“back”命令，重复使用它就可以逐层回退几个房间，直到把玩家带回到游戏的起点；
4. 在游戏中增加具有传输功能的房间，每当玩家进入这个房间，就会被随机地传输到另一个房间；
5. 在游戏中新建一个独立的Player类用来表示玩家，并实现下列功能需求：
   * 一个玩家对象应该保存玩家的姓名等基本信息，也应该保存玩家当前所在的房间；
   * 玩家可以随身携带任意数量的物件，但随身物品的总重量不能操过某个上限值；
   * 在游戏中增加两个新的命令“take”和“drop”，使得玩家可以拾取房间内的指定物品或丢弃身上携带的某件或全部物品，当拾取新的物件时超过了玩家可携带的重量上限，系统应给出提示；
   * 在游戏中增加一个新的命令“items”, 可以打印出当前房间内所有的物件及总重量，以及玩家随身携带的所有物件及总重量；
   * 在某个或某些房间中随机增加一个magic cookie（魔法饼干）物件，并增加一个“eat cookie”命令，如果玩家找到并吃掉魔法饼干，就可以增长玩家的负重能力；
6. 扩充游戏基本架构，使其支持网络多人游戏模式，具备玩家登陆等功能；
7. 为单机或网络版游戏增加图形化用户界面，用过可以通过图形化界面执行游戏功能；
8. 可以为游戏增加数据库功能，用于保存游戏状态和用户设置；
9. ......
10. 基于Github中的issue管理功能明确工作任务并为组员分配工作任务
    * 将工作任务拆分细化后，明确版本开发计划和里程碑时间节点；
    * 在github平台创建任务issue并为所有组员分配任务；
    * 每位组员可以分别承担不同的开发任务，也可以按照小组角色分别承担开发、测试、集成等工作任务；
    * 工作任务的划分是最终衡量小组成员工作量的重要依据；
11. 基于小组商定的分支模型进行软件功能开发，并按开发流程进行代码测试、提交、归并和同步
    * 小组成员按照小组商定的分支模型在各自的工作分支进行进行开发任务；
    * 工作分支在合并前应同步到远程仓库供教师检查每人的开发工作完成情况；
    * 提交代码时应按照小组约定的规范格式填写代码提交说明，代码提交说明也将作为评分的重要依据；
12. 代码提交到远程仓库后，应进行自动化代码格式规范检查和测试以确保功能符合需求设计；
    * 可以利用github平台的actions功能在代码提交时自动触发代码格式检查，对于不符合规范的代码系统将给出提交失败提示；
    * 可以利用github平台的actions功能在代码提交时自动触发测试用例检查，对于不能通过测试检查的代码系统将给出提交失败提示；
13. 可尝试进行代码自动化打包，自动生成可供执行的jar文件
    * 结合github平台的actions功能和maven编译脚本，在代码通过规范性检查和测试用例后，进行自动化打包，生成可供直接执行的jar文件用于系统发布

## 任务输出

1. 以上所有工作基于github平台完成，以项目小组为单位在项目根目录下创建一个名称为REPORT.md的文件，对项目实训过程进行报告，报告内容应包含各工作内容的设计过程及完成情况的说明；
2. 所有源代码应包含规范化标注；
3. 每位同学以word文档格式提交个人实训报告；

# 程序设计与开发

# World of zuul 架构设计

### 系统概述：

* 本系统是一个基于B/S架构的文字冒险游戏平台，提供给用户一个虚拟世界的探索和互动环境。用户可以通过浏览器访问系统，创建角色、探索房间、收集物品等。

### 架构设计目标：

* 可扩展性：系统能够支持大量的玩家同时在线，并具备横向扩展的能力。
* 可维护性：系统的代码结构清晰、模块化，易于维护和扩展。
* 性能：系统能够快速响应用户请求，保证流畅的游戏体验。
* 安全性：确保用户数据的安全性，包括用户身份验证和数据传输的加密。

### 技术选型：

* 前端技术选型：采用Vue框架和Element-UI组件库，以实现组件化开发和用户友好的界面。
* 后端技术选型：使用Spring Boot作为后端框架，集成MyBatisPlus进行数据库操作，以实现快速开发和易于维护的后端服务。

### 前端架构设计：

* 使用Vue框架进行前端开发，实现组件化开发和响应式设计。
* 采用Element-UI组件库，提供丰富的UI组件和样式，简化前端开发流程。
* 通过Vue Router实现前端路由，实现页面的无刷新加载和导航。
* 使用Vuex进行状态管理，确保数据的一致性和共享。

### 后端架构设计：

* 使用Spring Boot框架，实现快速搭建和开发后端服务。
* 集成MyBatisPlus作为ORM框架，简化数据库操作和提供性能优化。
* 使用Spring Security实现用户认证和权限管理，确保系统的安全性。
* 采用RESTful API风格进行前后端交互，实现数据的传输和交互。

### 前后端交互：

* 前后端通过RESTful API进行数据交互，使用JSON格式进行数据传输。
* 前端发送HTTP请求，后端根据路由进行相应的处理，并返回JSON格式的响应数据。
* 使用Token进行用户认证，前端在请求头中添加Token进行身份验证。

### 数据库设计：

* 使用关系型数据库，如MySQL，设计游戏系统的数据库结构。
* 使用MyBatisPlus进行数据库操作，包括增删改查和事务管理。
* 设计玩家角色表、房间表、物品表等，确保数据的一致性和完整性。

### 分工与协作：

* 创建软件开发小组，每个小组成员负责不同的模块或功能的开发。
* 使用GitHub进行团队协作和版本控制，创建项目仓库和相关分支。
* 通过GitHub的Issue管理功能明确工作任务，并为组员分配任务。
* 按照小组商定的分支模型进行软件功能开发，定期进行代码测试、提交、归并和同步。

### 自动化流程：

* 使用GitHub Actions进行自动化代码格式规范检查和测试用例检查。
* 在代码提交时触发自动化流程，对不符合规范和不能通过测试的代码进行提示。
* 结合Maven编译脚本，实现代码自动化打包，生成可执行的jar文件用于系统发布。

# 开发版本计划与任务分派

### Version 0.1：

基本界面

创建基本UI布局，并实现角色移动功能；

设计简单的游戏场景和游戏角色；

实现基本的副本模式；

基本框架

搭建基本的服务器框架；

创建玩家账号系统和角色信息管理系统；

实现基本数据库功能；

### Version 0.2：

地图和角色展示：

改进游戏场景和游戏角色，提高质量；

支持大规模游戏世界，实现多种地图效果；

添加角色定制功能；

游戏逻辑：

添加游戏中心逻辑处理；

实现简单的游戏规则系统；

添加角色进入游戏和出游戏等操作；

### Version 0.3：

交互系统

添加交互选项，让玩家能够快速地探索游戏世界；

支持玩家交流和动态信息显示；

增加任务管理中心；

数据管理

实现人物和装备等数据管理；

增加在线交互功能；

建立基础架构，可支持大规模游戏操作；

### Version 0.4：

安全和稳定性

增加异常捕获和报告机制；

强化数据安全和游戏安全；

实现玩家数据备份和恢复机制；

### Version 1.0：

正式版本

优化游戏服务器性能；

实现多服务器分布式处理；

提供API接口，便于二次开发和扩展。

## 任务分派

### 架构设计文档编写

- 负责人：周盟

- 描述：根据系统需求和技术方案，编写系统的整体架构设计文档，包括前后端架构、数据交互方式、数据库设计等。

### 前端界面编写 - 登录、注册、忘记密码部分

- 负责人：毕列祥

- 描述：实现游戏的前端界面，包括登录、注册、忘记密码等功能，并实现对表单字段的校验。

### 前端界面编写 - 游戏界面部分

- 负责人：曾俊博

- 描述：实现游戏的前端界面，包括游戏场景展示、角色移动等功能。

后端鉴权认证功能实现

- 负责人：曾梓迪

- 描述：编写后端的鉴权认证功能，包括登录、注册、忘记密码等接口的实现，确保用户的身份验证和安全性。

### 后端核心游戏逻辑编写

- 负责人：周盟

- 描述：根据游戏设计要求，编写后端的核心游戏逻辑，包括游戏进程控制、事件处理等功能。

### 数据库设计与前后端数据交互格式设计

- 负责人：周盟

- 描述：设计游戏所需的数据库结构，确定表的字段和关系，同时定义前后端数据交互的格式，确保数据的一致性和有效性。

### 测试与问题记录

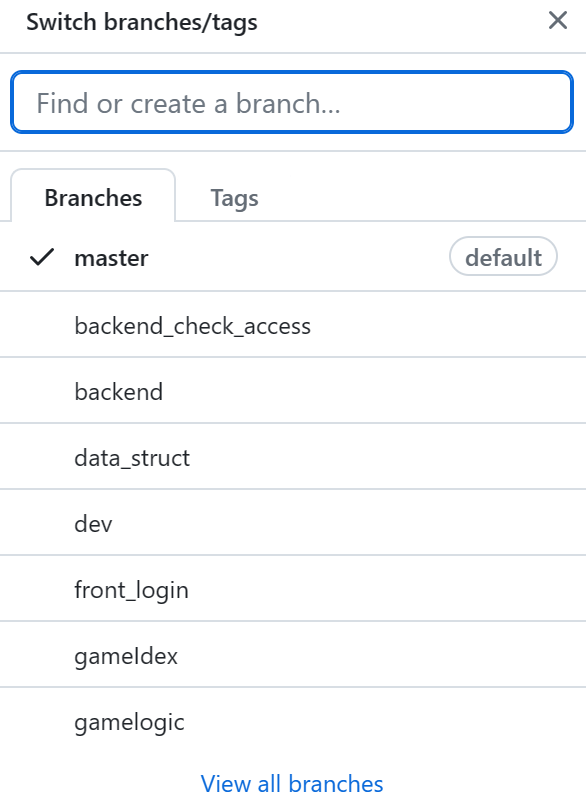
- 负责人：全体成员

- 描述：进行测试并记录系统的功能、性能、安全等方面的问题，及时反馈并解决问题，确保系统的稳定性和质量。

任务分派遵循每个人领取相应的issue并在自己的分支上进行开发的方式，完成阶段性开发任务后提交相应的PR到dev分支，同时每次提交需要简要说明本次提交的内容。

请根据以上计划和任务分派，合理安排自己的时间和工作，确保按时完成任务并达到预期的质量要求。

# 开发分支模型与代码合并



1. 主要分支：

- master: 主分支，用于存放发布版本的稳定代码。

- dev: 开发分支，用于集成各个成员的开发成果，并进行整体功能测试。

2. 功能分支：

- backend\_check\_access: 后端登录注册功能开发分支，由曾梓迪负责。

- gamelogic: 后端游戏逻辑开发分支，由周盟负责。

- frontend\_login: 前端登录注册功能开发分支，由毕列祥负责。

- gameIdex: 前端游戏界面开发分支，由曾俊博负责。

3. 代码合并策略：

- 曾梓迪和毕列祥负责的功能分支（backend\_check\_access和frontend\_login）开发完成后，将代码提交到对应分支，并创建Pull Request（PR）到dev分支。经过代码审查和测试后，由团队成员合并到dev分支。

- 周盟负责的功能分支（gamelogic）开发完成后，同样提交代码到对应分支，并创建PR到dev分支。经过审查和测试后，合并到dev分支。

- 曾俊博负责的功能分支（gameIdex）开发完成后，提交代码到对应分支，并创建PR到dev分支。经过审查和测试后，合并到dev分支。

- dev分支经过一段时间的集成和测试后，当功能达到一定的阶段性成果，进行版本发布，将dev分支合并到master分支，生成新的发布版本v1.0.0。

# 开发规范与测试

## 开发规范：

1.代码规范：

- 遵循统一的命名规范，包括变量、函数、类等命名方式。

- 保持代码的可读性和可维护性，使用合适的缩进、注释和代码结构。

- 遵循团队约定的编码风格，如代码缩进、命名规则等。

2. 分支管理：

- 每个成员负责的任务在独立的分支上开发，确保代码的隔离性和独立性。

- 分支命名规范：按照任务名称或功能进行命名，避免冲突和混淆。

3. 提交规范：

- 每次提交需要简要说明本次提交的内容，以便于其他成员了解代码的改动和目的。

- 使用有意义的提交消息，清晰描述本次提交的目的和影响。

4. 代码审查：

- 团队成员之间进行代码审查，确保代码质量和一致性。

- 审查过程中提供有建设性的反馈和建议，帮助改进代码。

## 测试策略：

1. 单元测试：

- 编写针对各个模块的单元测试用例，验证各个函数和方法的功能和边界条件。

- 执行单元测试并确保测试覆盖率达到预期水平。

2. 集成测试：

- 在dev分支进行集成测试，验证各个模块之间的协作和交互是否正常。

- 模拟真实场景，测试各个功能模块的整体性能和稳定性。

3. 用户验收测试：

- 在开发阶段的某个时点，邀请用户参与测试，收集用户反馈和建议。

- 验证系统是否符合用户需求，修复和改进系统中发现的问题。

4. 性能测试：

- 针对关键功能和系统瓶颈点进行性能测试，评估系统的承载能力和响应时间。

- 优化性能瓶颈，提升系统的性能和用户体验。

# 集成与版本发布

集成：

前后端分离：开发人员可以将前端代码和后端代码分别上传到github版本管理工具，在项目团队内部进行协作；

接口文档：开发人员需要编写接口文档，明确前后端数据交互的具体规范；

联调测试：前后端在自己本地进行单元测试，在完成后进行联调测试；

部署环境：部署至测试环境，完成全面测试后上线。

版本发布：

打标签：在代码上传至版本管理工具后，为每个重要版本打上tag。例如：v0.1, v0.2等等；

版本控制：使用版本控制工具（例如Jenkins）来进行版本管理,每个版本的代码需要在控制工具中单独设置一个编译任务；

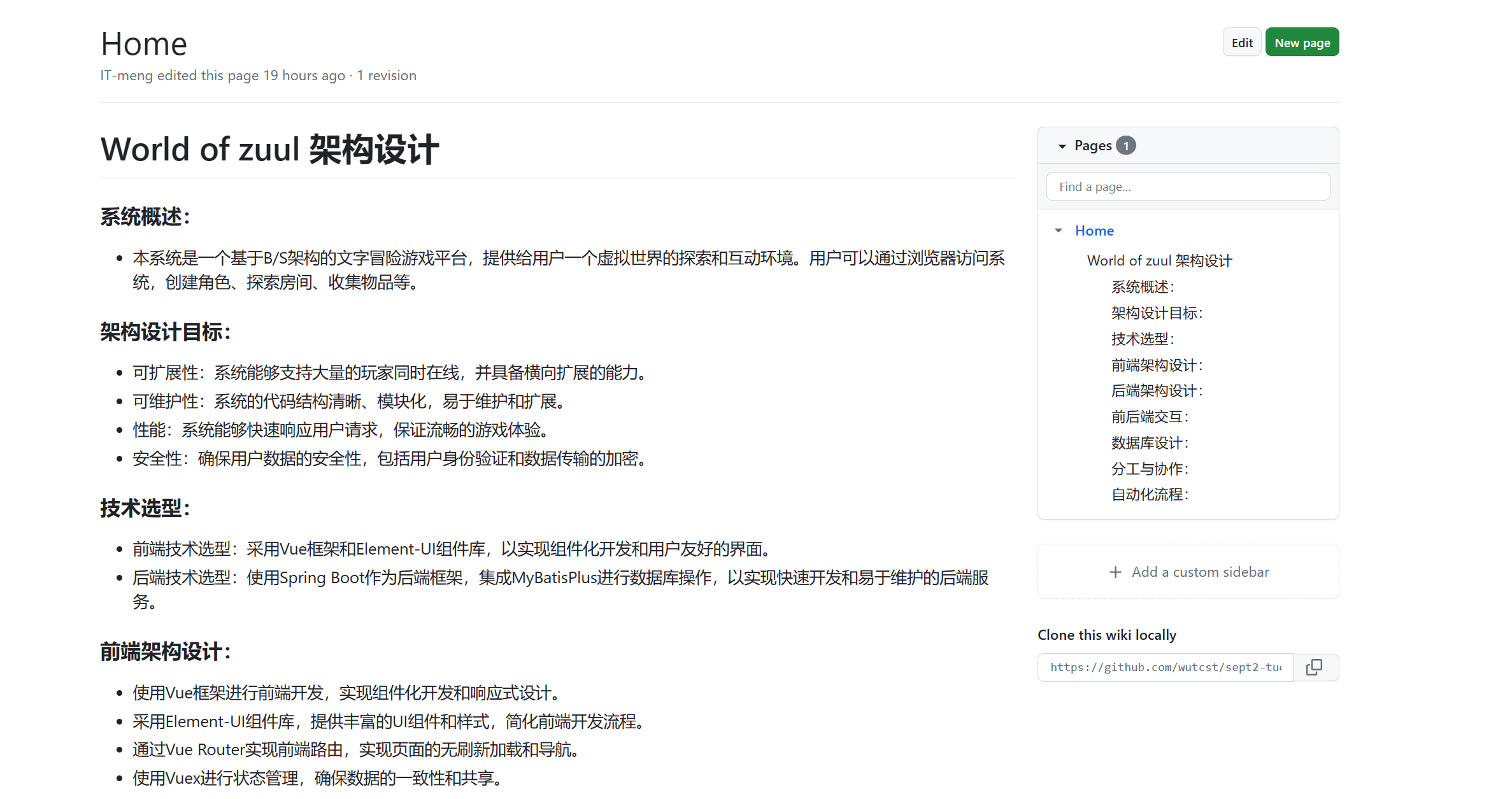
发布环境：在测试环境测试完成后，可以将代码部署至生产环境；

发布通知：发布新版本之后，包括开发人员在内的所有使用人员需要得到相应的通知。

# 小组成员个人任务

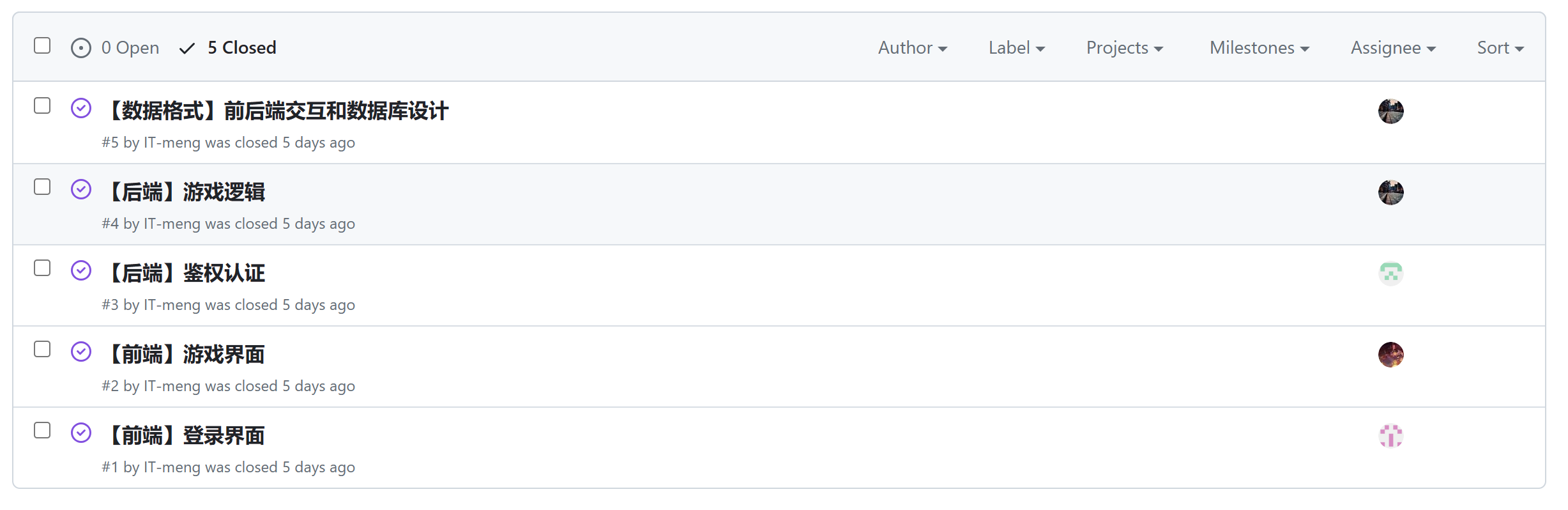
在本次任务中，**周盟**承担了以下几个具体任务：

1. 架构设计文档编写：



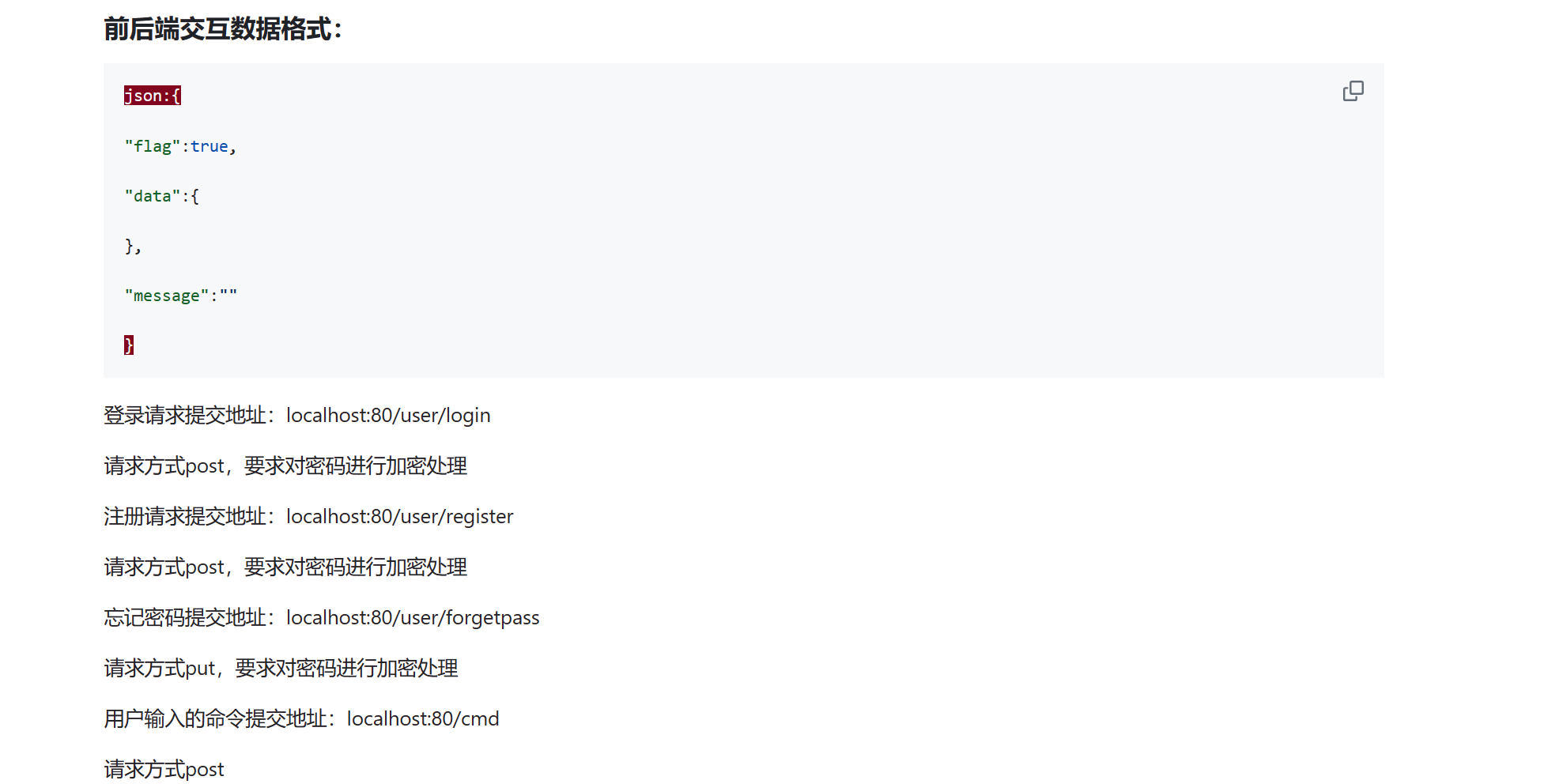
我负责编写系统的架构设计文档。在编写过程中，我详细分析了系统的需求和技术方案，并设计了系统的整体架构，包括前后端的分层结构、数据交互方式以及数据库设计等。我注重了系统的可扩展性和可维护性，确保了系统设计的合理性和稳定性。

2. 相关issue的发布与跟进：



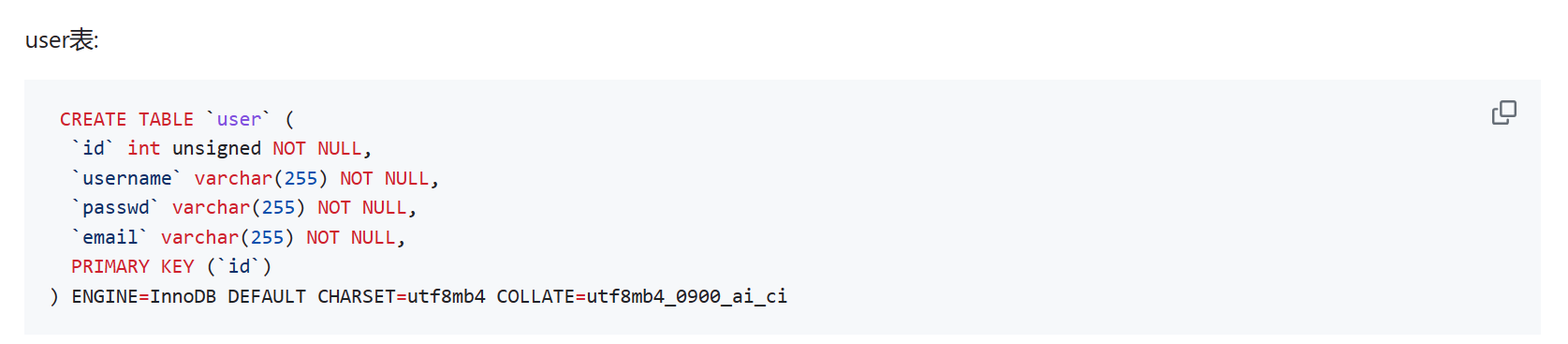
我负责发布与跟进相关的issue。我确保在任务开始时明确了每个成员的任务，并及时发布相关的issue，明确任务的具体内容和要求。我与其他成员保持密切的沟通，跟进任务的进展情况，及时解决遇到的问题，确保任务的顺利进行。

3. 前后端数据交互格式设计：



我负责设计前后端数据交互的格式。在设计过程中，我考虑了数据传输的准确性和效率性，定义了统一的数据格式和接口规范，以确保前后端之间的数据交互的一致性和可靠性。我与前端成员密切合作，进行接口的对接和调试，确保数据的正确传输和处理。

4. 数据库建模：

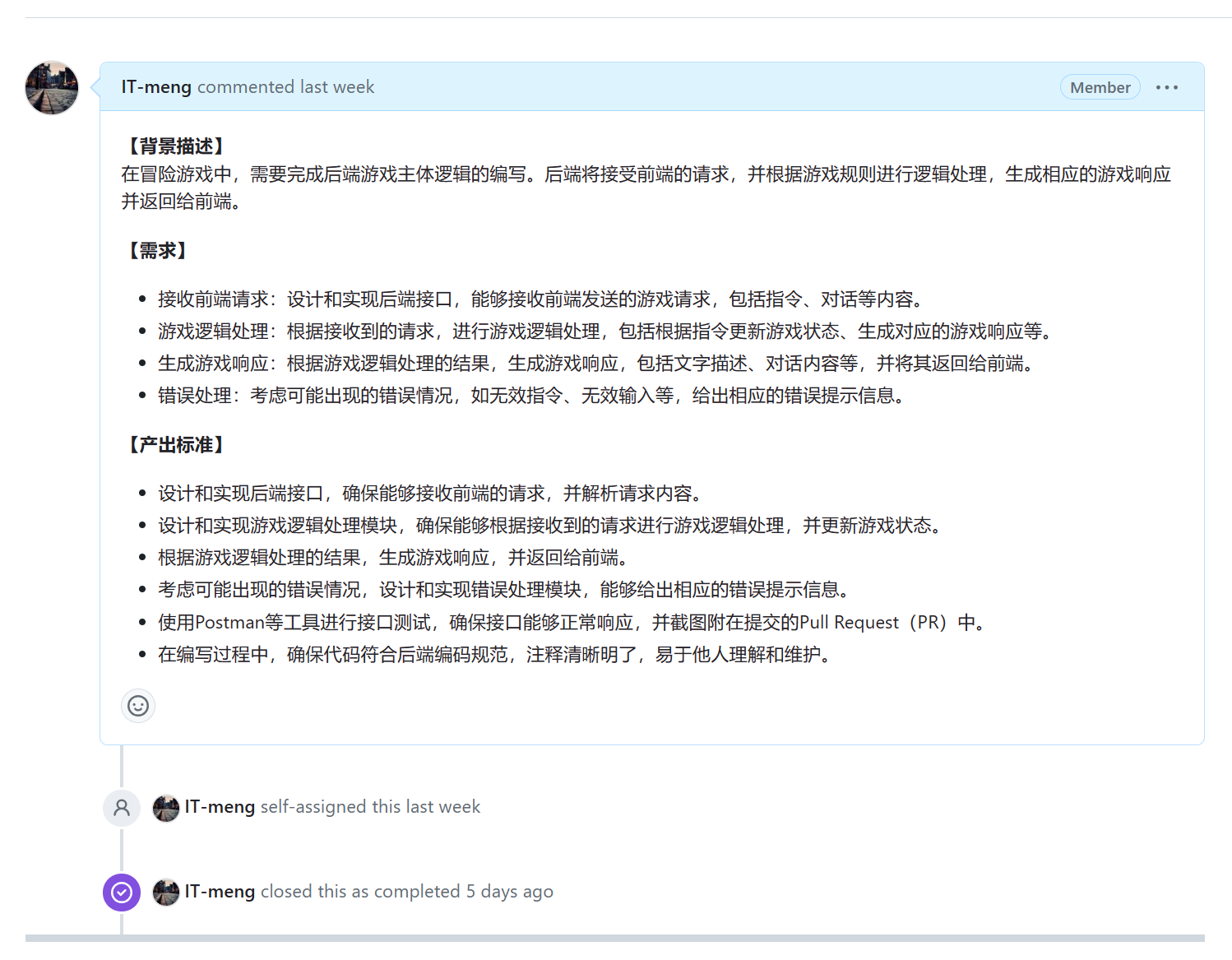


我负责根据系统需求和架构设计进行数据库建模。我仔细分析了系统的数据结构和关系，并设计了合适的数据库表结构，定义了表的字段和关联关系，以满足系统的数据存储和查询需求。我注重数据库的性能和数据一致性，确保数据的有效管理和高效访问。

5. 分支管理和代码合并：

我负责创建自己的开发分支，并协调与其他成员的代码合并工作。我与团队成员密切合作，确保各个分支的代码能够顺利合并，并解决合并过程中可能出现的冲突。我注重代码的质量和可维护性，确保代码的整体一致性和稳定性。

6. 后端核心游戏逻辑的代码编写与测试：



我负责编写后端核心游戏逻辑的代码，并进行相应的单元测试。我根据游戏设计要求，实现了游戏的核心功能和逻辑处理，同时编写了相应的测试用例，确保游戏逻辑的正确性和稳定性。我注重代码的可读性和健壮性，确保代码的质量和可维护性。

7、GitHub Action自动化编译配置：



在本次任务中，**毕列祥**承担了以下几个具体任务：



1、需求收集和分析：小组成员根据项目要求，收集和分析用户需求。这个过程包括与项目经理，UI设计师和后端开发人员进行协调，以确保前端面向用户的需求合理。

2、UI设计：UI设计者负责制定各个页面的美学和功能设计。根据用户需求和游戏主题进行页面UI设计，并制定相应的UI规范和标准。

3、前端编码实现：前端开发人员根据UI设计师提供的UI及需求分析的要求，使用所选的前端框架和其他工具实现整个系统，并确保代码的稳定性和安全性。

4、页面交互逻辑实现：交互设计师负责制定网页的互交模式，以优化用户体验。前端开发人员需要根据交互设计师的要求实现页面的交互逻辑功能。

5、测试和验收：测试工程师对页面进行全面和系统的测试，以确保页面功能满足用户需求，并对市面上同类产品及时进行测试。如果测试成功，项目可以进行交付。

**具体实施情况：**

1. Clone对应代码，在本地构建对应仓库即可开始。
2. 登录界面

<1>设计登录页面的UI。登录页面应该能够吸引用户的注意力，并将重点放在登录表单上。视觉设计应该与游戏主题相一致，以增强用户体验。同时，要考虑响应式设计，以确保页面在不同设备上的表现相同。

<2>构建登录表单。登录表单应该包含用户名和密码字段。应该使用调用API获取用户的认证信息，并通过加密过程保持用户数据的机密性。

<3>添加验证逻辑。在提交表单之前，需建立逻辑来验证表单中输入的数据。例如，可以确保用户名和密码格式正确；可以创建一个验证码来防止攻击者破解密码。

<4>处理登录请求。在用户点击“登录”按钮时，前端代码会通过API将表单中输入的数据发送到后端服务器，以验证用户输入的凭据是否与服务器中保存的凭据匹配。如果验证失败，则需要显示相应错误信息。

<5>显示反馈信息。如果验证成功，则用户应被重定向到游戏主界面。如果验证失败，则需要提供适当的反馈信息，以提醒用户输入的凭据非法。

总之，成功实施World of Zuul登录页的关键在于设计一个用户友好、机能齐全的登录表单，并确保前端代码能与服务器端进行稳定的通讯。此外，代码的质量和性能优化将有助于保证正常的游戏体验。

1. 注册跟修改密码页面

<1>设计注册和修改密码页面的UI。页面设计应该吸引用户的注意力，使用户容易理解如何填写信息。应该使用调用API获取用户的认证信息，并通过加密过程保持用户数据的机密性。

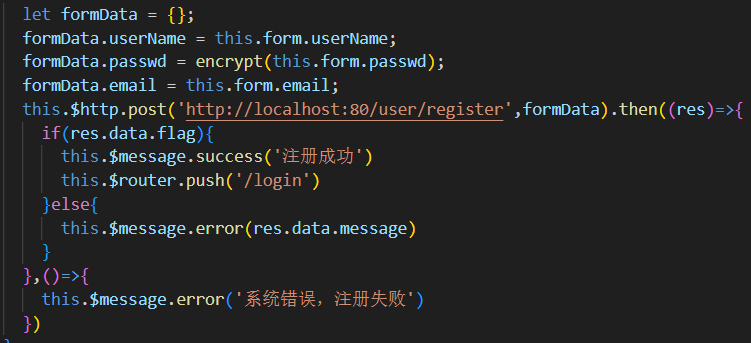
<2>构建表单。注册和修改密码表单应该包含必填字段和可选字段。例如，注册表单需要包含用户名、密码和电子邮件地址等必填字段，并包含其他可选字段，如性别、出生日期等。

<3>添加验证逻辑。在表单提交之前，需要建立逻辑来验证表单中输入的数据。例如，可以确保用户名、密码和电子邮件地址格式正确；可以创建一个验证码来验证表单。確保安全性，例如交叉网站脚本攻击（XSS）和请求伪造攻击（CSRF）。

<4>处理注册和修改密码请求。在用户点击“注册”或“修改密码”按钮时，前端代码会通过API将表单中输入的数据发送到后端服务器，以验证用户输入的凭据是否与服务器中保存的凭据匹配。如果验证失败，则需要显示相应错误信息。

<5>显示反馈信息。如果验证成功，则用户应显示成功信息并返回登陆界面。如果验证失败，则需要提供适当的反馈信息，以提醒用户输入的凭据非法。



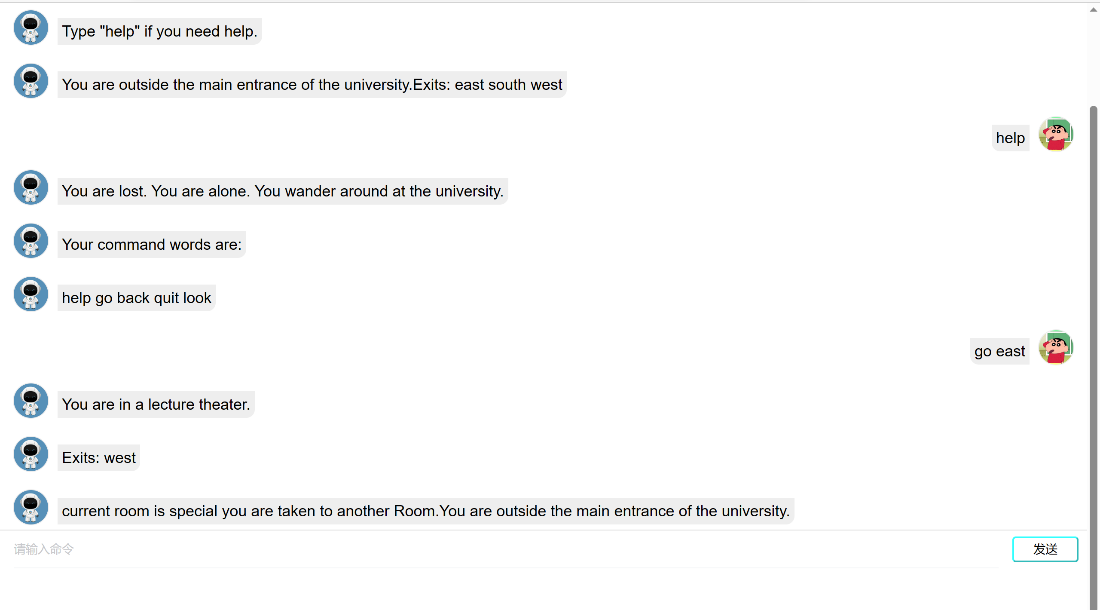


在本次任务中，**曾俊博**承担了以下几个具体任务：

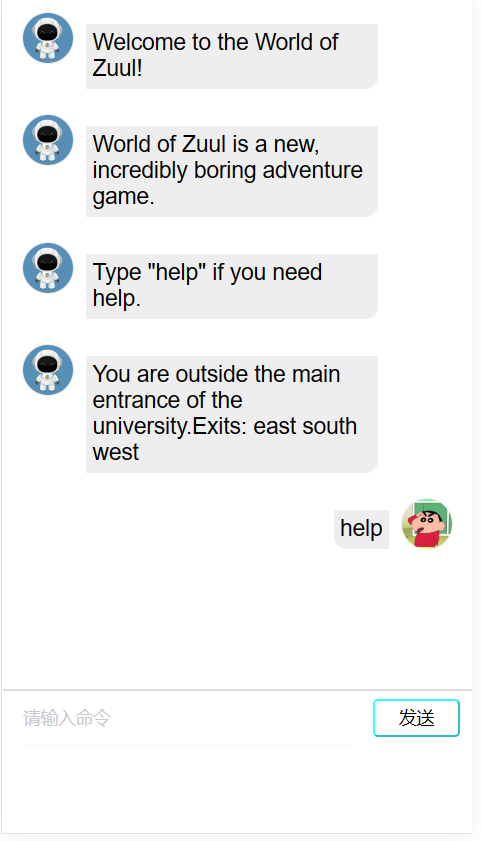
我在github上承接的issue 是编写文字游戏的主页面，对于这个任务，我选择的开发工具是 vue 3 和 vant 等 vue 的一些插件。项目的主要管理方法是在本地建立新的分支进行开发，并使用git管理，在本地测试通过之后上传到远程仓库中。等待另一位前端负责人也提交后，申请合并并处理关键冲突，再将合并的代码pull 到本地进行进一步完善。等待后端人员git之后，使用本地的服务器进行交互测试。

**功能设计方案：**

**前端页面**：采用聊天框的形式，使用vant插件将用户的命令和后端的返回结果显示在同一个页面中，在vue 中定义一个列表来存放当前页面的消息（包括用户输入和后端返回），使用v-for指令将其显示在页面中。以下为效果截图：



**响应式：**使用flex布局方式 实现响应式页面效果，结果如图：



**向后端发送请求：** 使用axios 向后端发送请求，并处理数据，代码和debug截图：





**开发中主要的问题：**

项目中遇到的主要的问题有两个。

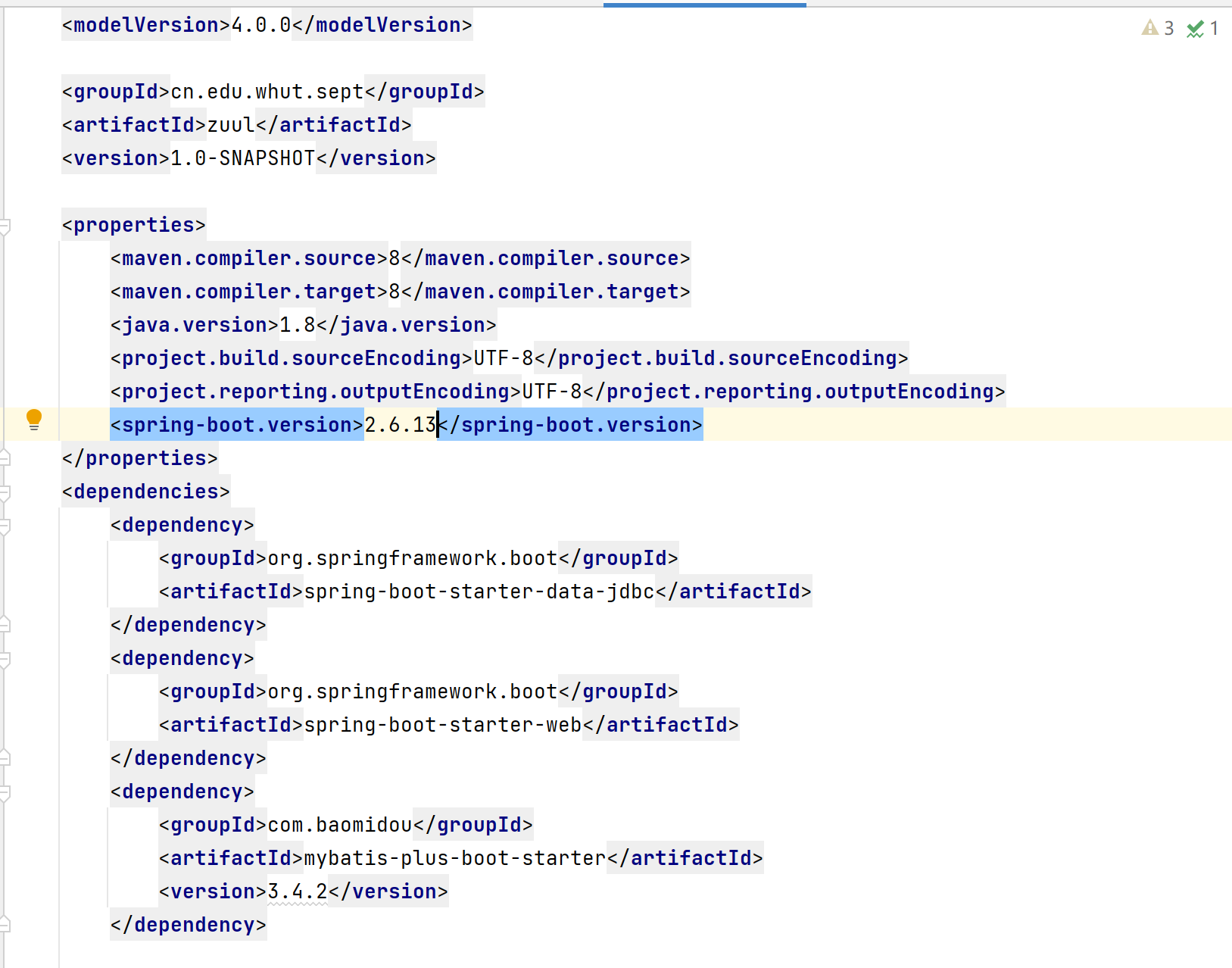
一个是html中使用vue 的{{str}} 显示内容时，str中的\n不会生成回车的问题，在本项目中，我的解决方案是使用 split 将str 按照\n 分成多个字符串组成的数组，使用for循环将数组封装成多个 data 对象，每个data对象对应 msgList 中的一个元素，这样需要换行的消息就分为了多条消息，不会出现文字密集的情况。

另一个问题是前端初步实现之后，进行前后端交互测试时，**后端人员提议在接到消息时能不能像聊天软件一样自动地显示最新的消息。**一开始使用网络上的方法，利用nextTick 进行回调，在Dom元素更新完成后将整体scroll （下滑），但是一直没效果，后来采用了 directive 的方式自定义了一个指令 v-scroll （使用 scrollIntoView 方法实现），在每个消息后加上标记 v-scroll指令的高度为0的div，完成了该功能。

在本次任务中，**曾梓迪**承担了以下几个具体任务：



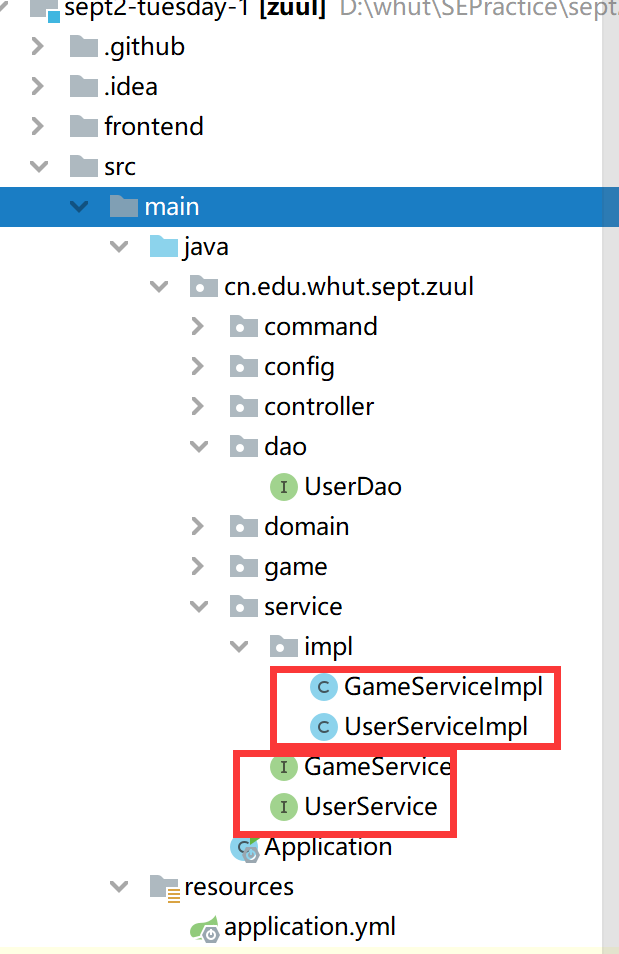
1、进行了Maven依赖导入一些包以及接口用来声明方法，提供抽象性。



2、设计并实现登录接口，包括接收用户的登录请求、验证用户身份信息、生成和返回登录凭证等功能。

3、设计并实现注册接口，包括接收用户的注册请求、验证注册信息有效性、保存用户信息到数据库等功能。

4、设计并实现忘记密码接口，包括接收用户的忘记密码请求、通过验证流程重置用户密码等功能。



5、在接口设计中，考虑安全性和合理的接口返回信息，确保接口的参数校验和异常处理。

6、使用Postman进行接口测试，确保接口能够正常响应，并截图附在提交的Pull Request中。

在编写过程中，确保代码符合后端编码规范，注释清晰明了，易于他人理解和维护。

过程中发现的错误：有时候会发现包没有成功导入，代码就会爆红，分析后得知是自己没有import正确的包，因为有的包名字相同但是完整路径不同，实则是两个不同的包，有时候导错了另一个