****

**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** | **林日奋** |
| **小组成员** | **林日奋、王煜泽、王金宇、王宇轩** |
| **指导教师** | **唐祖锴** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 2023 | 年 | 6 | 月 | 24 | 日 |

**目录**

[1 课程任务 1](#_Toc91495513)

[2 开发版本计划 2](#_Toc91495514)

[3 任务分配 2](#_Toc91495515)

[4 开发分支模型 5](#_Toc91495516)

[5 程序设计与开发 6](#_Toc91495515)

[6 小组成员个人任务 9](#_Toc91495517)

[7 代码合并 23](#_Toc91495518)

[8 开发规范与测试 24](#_Toc91495519)

[9 集成与版本发布 28](#_Toc91495520)

# 1 课程任务

1.1 任务目的

（1）巩固强化软件编程规范

（2）提高面向对象软件建模与抽象能力

（3）培养小组协同开发能力

（4）掌握基于Maven的软件项目管理机制

（5）掌握基于Github的小组协同开发工具和平台

（6）了解DevOps软件开发流程

1.2 任务内容

（1）创建开发小组

每个开发小组人数2-4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，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；扩充后的软件应包含用户交互界面及功能服务模块，以提供便捷的功能操作。

（3）为小组成员分配开发任务

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；

（4）按分支模型执行开发工作

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，进行代码测试、提交、归并和同步；

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；

（5）自动化代码格式规范检查与测试

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；

（6）尝试进行自动化打包与发布

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

1.3任务输出

所有源代码应包含规范化标注；

编写课程实践报告，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 开发版本计划

World of Zuul文字冒险游戏

版本：v1.0.0

**（1）需求分析和规划阶段：2023.06.10-2023.06.12**

确定项目的基本功能需求，包括用户注册、登录和聊天功能。

制定用户界面设计和交互流程。

完成人员分工文档、提交规范文档、接口规范文档、分支管理规范文档等的编写。

**（2）前期准备阶段：2023.06.12-2023.06.13**

搭建前端开发环境：安装Vue CLI、配置开发工具和依赖项。

搭建后端开发环境：安装Java、Spring Boot、配置开发工具和依赖项。

创建项目代码仓库，建立前端和后端项目结构。

**（3）前端开发阶段：2023.06.14-2023.06.18**

实现用户注册页面和功能。

实现用户登录页面和功能。

实现聊天页面和基本聊天功能。

**（4）后端开发阶段：2023.06.15-2023.06.22**

创建用户管理模块，包括用户注册和登录的逻辑。

创建聊天模块，处理聊天相关的请求和逻辑。

**（5）测试和修复阶段：2023.06.22-2023.06.23**

进行前端和后端的单元测试。

发现和修复可能存在的问题和缺陷。

**（6）发布和部署阶段：2023.06.23-2023.06.24**

部署前端应用到合适的Web服务器。

部署后端应用到合适的应用服务器。

配置服务器和域名，确保应用可以通过浏览器访问。

# 3 任务分配

本团队wuwu共四人，按照小组角色分别承担前端开发、后端开发、测试、集成等工作任务。

3.1 前端测试与开发

（1）登录界面开发

* 创建登录页面的Vue组件。
* 使用Vue Router管理路由，并设置登录页面的路由。
* 使用Element UI组件库设计和构建登录表单。
* 实现登录表单的输入验证逻辑。
* 发起与后端的登录请求，并处理登录结果。
* 根据登录结果进行页面跳转或显示错误提示信息。

（2）登录界面开发

* 创建注册页面的Vue组件。
* 使用Vue Router管理路由，并设置注册页面的路由。
* 使用Element UI组件库设计和构建注册表单。
* 实现注册表单的输入验证逻辑。
* 发起与后端的注册请求，并处理注册结果。
* 根据注册结果进行页面跳转或显示错误提示信息。

（3）登录界面开发

* 创建聊天页面的Vue组件。
* 使用Vue Router管理路由，并设置聊天页面的路由。
* 使用Element UI组件库设计和构建聊天界面的布局。
* 实现与后端的数据交互，包括发送和接收聊天消息。
* 显示聊天消息的历史记录和实时消息。
* 实现发送消息的功能，并处理发送消息的逻辑。
* 添加样式和动画效果，以提升用户体验。

3.2 后端设计与开发1

（1）实现"go"功能

* 设计并实现与移动命令相关的API接口，接收玩家的移动指令并进行相应的处理。
* 根据玩家的当前位置和目标位置，更新玩家的位置信息，并返回相应的结果。

（2）实现"look"功能

* 设计并实现与查看命令相关的API接口，接收玩家的查看指令并进行相应的处理。
* 根据玩家的当前位置，返回当前房间的描述信息、可见物品等相关信息。

（3）实现"items"功能

* 设计并实现与查看物品命令相关的API接口，接收玩家的查看物品指令并进行相应的处理。
* 根据玩家的当前位置，返回当前房间中的可见物品列表。

（4）实现"take"功能

* 设计并实现与捡起命令相关的API接口，接收玩家的捡起指令并进行相应的处理。
* 根据玩家的当前位置和指定的物品，更新玩家的物品列表和房间的物品列表。

（5）实现"go"功能

* 设计并实现与放下命令相关的API接口，接收玩家的放下指令并进行相应的处理。
* 根据玩家的当前位置和指定的物品，更新玩家的物品列表和房间的物品列表。

3.3 后端设计与开发2

（1）实现"help"功能

* 设计并实现与帮助命令相关的API接口，接收玩家的帮助指令并进行相应的处理。
* 返回游戏的帮助信息，包括命令列表、操作说明等。

（2）实现"quit"功能

* 设计并实现与退出命令相关的API接口，接收玩家的退出指令并进行相应的处理。
* 处理玩家的退出逻辑，保存游戏进度并关闭游戏。

（3）实现"eatCookie"功能

* 设计并实现与食用命令相关的API接口，接收玩家的食用指令并进行相应的处理。
* 根据玩家的当前位置和指定的食物，更新玩家的状态和房间的状态。

（4）实现"back"功能

* 设计并实现与回退命令相关的API接口，接收玩家的回退指令并进行相应的处理。
* 实现玩家回退到上一个房间的功能。

（5）实现高级"back"功能

* 设计并实现与高级回退命令相关的API接口，接收玩家的高级回退指令并进行相应的处理。
* 实现玩家回退到指定房间的功能。

3.4 初步测试和前后端联调

* 前端测试：使用Yapi进行前端接口测试，验证前端与后端API的交互是否正常。模拟各种场景，包括正常操作、异常操作等，确保前端功能的稳定性和一致性。检查前端界面的布局、样式和交互，确保用户体验良好。
* 后端测试：使用Postman进行后端接口测试，验证后端API的功能是否正确。模拟各种请求，包括正确请求、错误请求等，确保后端的接口逻辑和数据处理的准确性。检查后端返回的数据是否符合预期，确保数据传输的准确性和完整性。
* 联调和问题修复：协调前端和后端开发人员进行接口联调，解决接口调用的问题和错误。跟踪和解决前后端交互过程中的数据传输问题。收集反馈和bug报告，并与开发人员协作解决问题。
* 完成controller层设计和实现

LoginController

* 设计并实现LoginController，处理登录相关的请求。
* 创建与登录功能相关的API接口，接收用户的登录请求并进行相应的处理。
* 验证用户提交的登录信息，包括用户名和密码的合法性。
* 调用相应的服务层方法，验证用户身份信息，并返回登录结果。
* 处理登录成功或失败的情况，返回相应的响应结果给前端。

RegisterController

* 设计并实现RegisterController，处理注册相关的请求。
* 创建与注册功能相关的API接口，接收用户的注册请求并进行相应的处理。
* 验证用户提交的注册信息，包括用户名、密码和其他必填字段的合法性。
* 调用相应的服务层方法，创建新用户并保存注册信息。
* 处理注册成功或失败的情况，返回相应的响应结果给前端。

ChatController

* 设计并实现ChatController，处理聊天相关的请求。
* 创建与聊天功能相关的API接口，接收用户的聊天请求并进行相应的处理。
* 处理用户发送的聊天消息，包括保存消息、发送给其他用户等操作。
* 根据需要实现消息的广播、群聊、私聊等功能。
* 处理聊天消息的接收和展示，包括获取历

# 4 开发分支模型

4.1 分支列表



4.2 分支说明

* dev分支由组长创建，dev分支由main分支切出，组长将dev分支clone到本地，添加阅读分支管理规范文档、提交规范文档、前后端接口规范文档、任务分工文档、issue说明后提交。
* 每个人名字的为每个人自己的开发分支，四个人在自己分支下完成代码开发，提交若干次commit，-m提交规范见《提交规范文档》处理用户发送的聊天消息，包括保存消息、发送给其他用户等操作。
* 若感觉自己需要新建分支，可新建每个人名字后加序号的分支，注意从各自的个人分支切出即可，如从linrifen分支可切出linrifen1,linrifen2，同时联系组长确认无误。
* 通过提交pr的方式合并分支，解决合并冲突，提交pr请联系组长。
* 待所有人完成代码后，四个个人分支会先合并到dev分支，此时dev分支中应含有整个项目代码。
* 四个人均可将dev分支pull到本地，在本次尝试运行项目，若四个人都成功运行，可将dev分支合并到main分支，至此完成任务。
* 若dev分支的全项目代码四个人pull到本地后均不能成功运行，利用debug分支进行前端、后端、数据库联调
* debug分支在后期综合测试时由组长创建，开发阶段不涉及到。

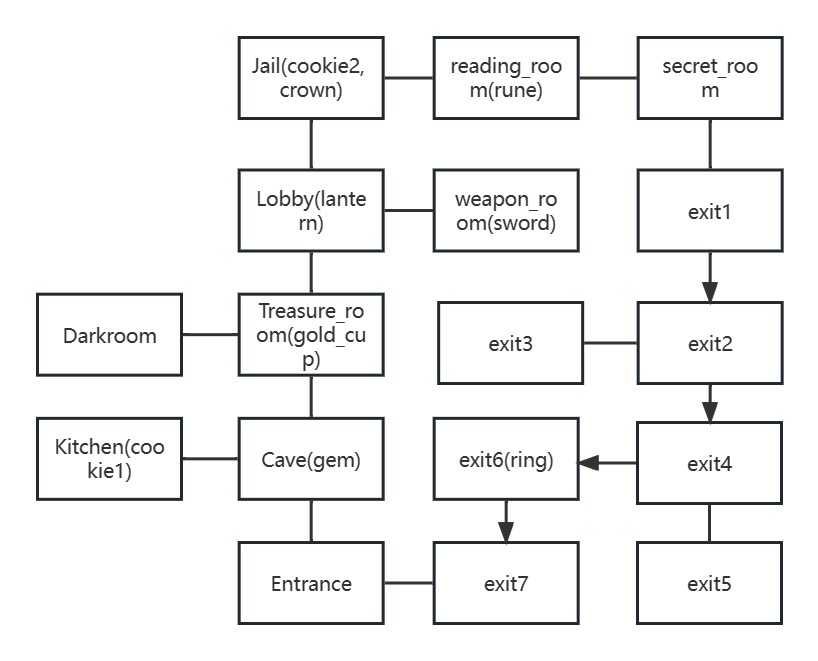
# 5 程序设计与开发

5.1 最终代码仓库

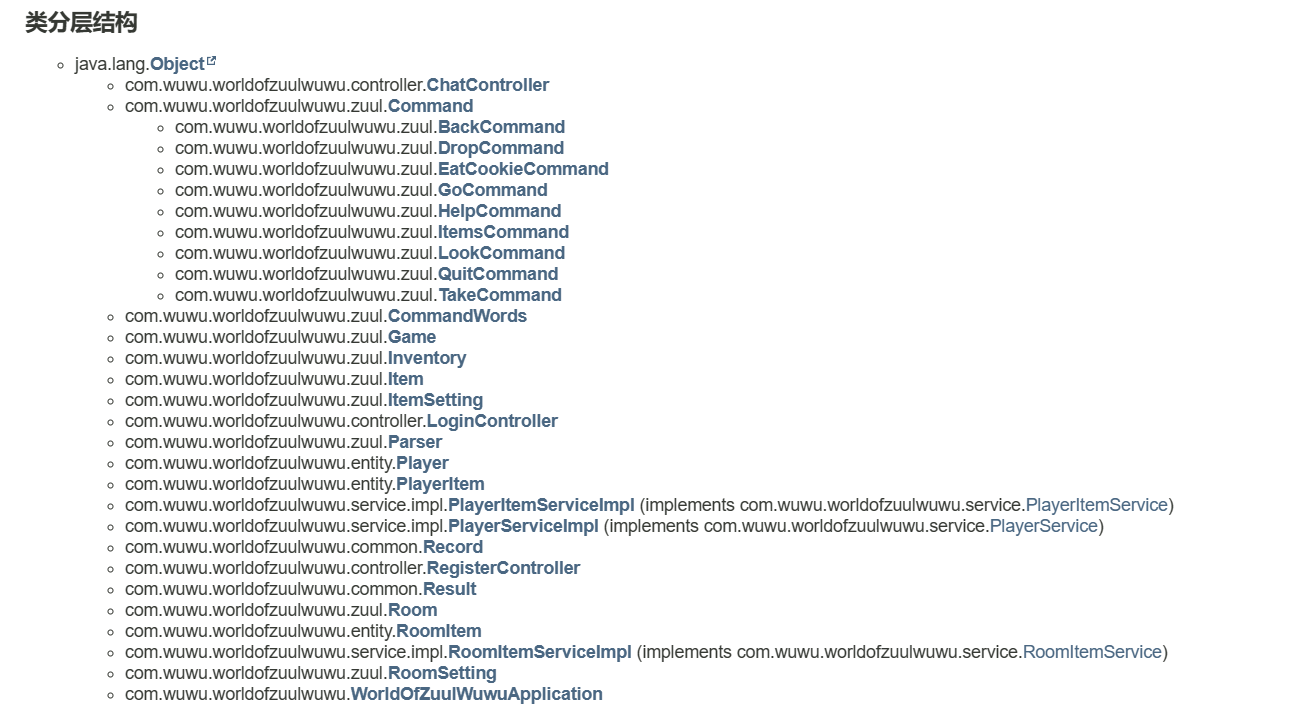


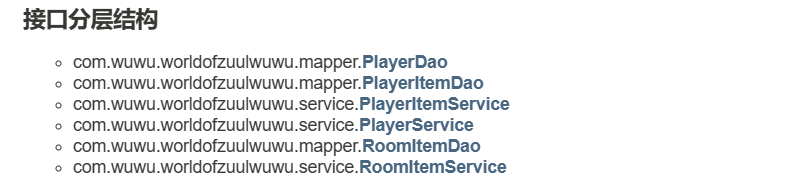
5.2 游戏地图设计

World of zuul问题冒险游戏的地图设计通常是由一系列房间和连接它们的门组成的。每个房间代表着游戏中的一个位置，玩家可以在这些位置之间移动。

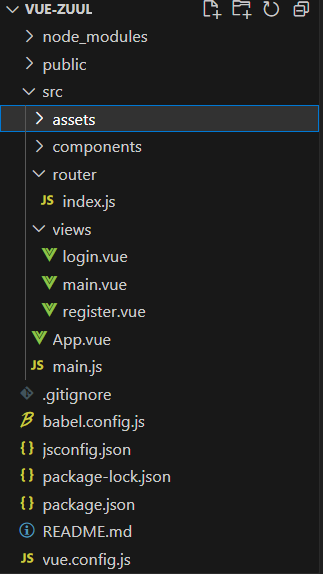


5.2 后端项目结构：

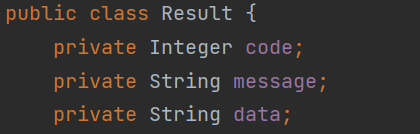




5.3 前端项目结构



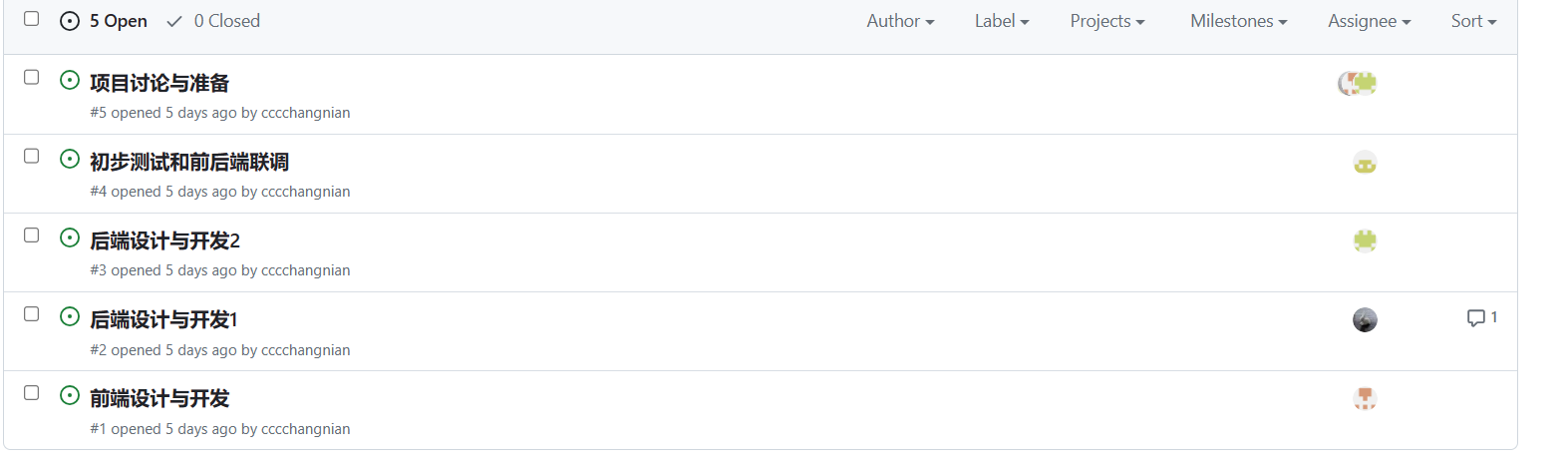
5.4 前后端交互对象Result



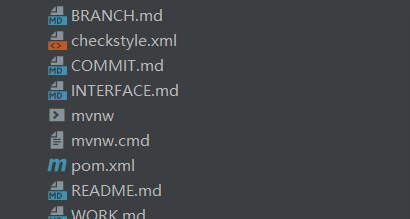
# 6 小组成员个人任务

6.1 前端设计与开发-林日奋

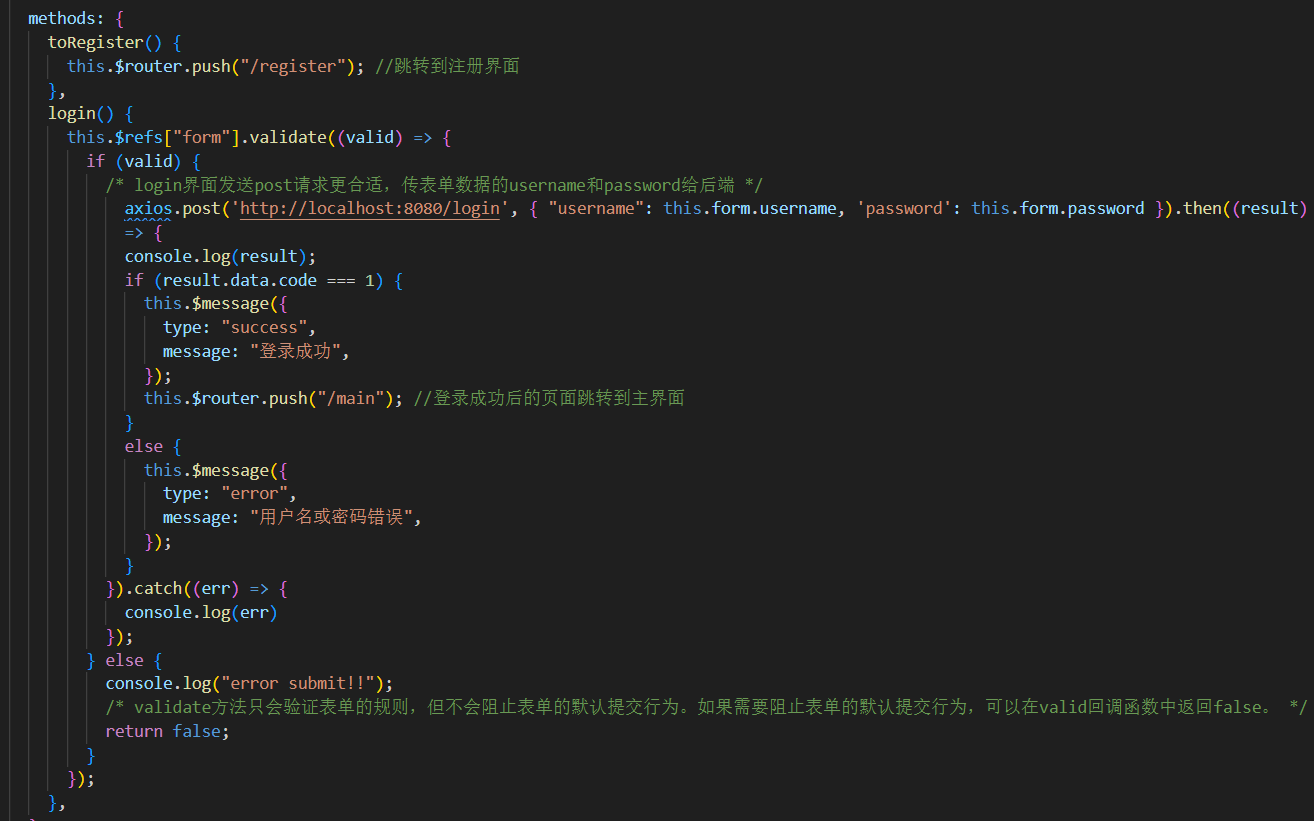
（1）开发前期发布ISSUE并分配ISSUE给合适的成员。

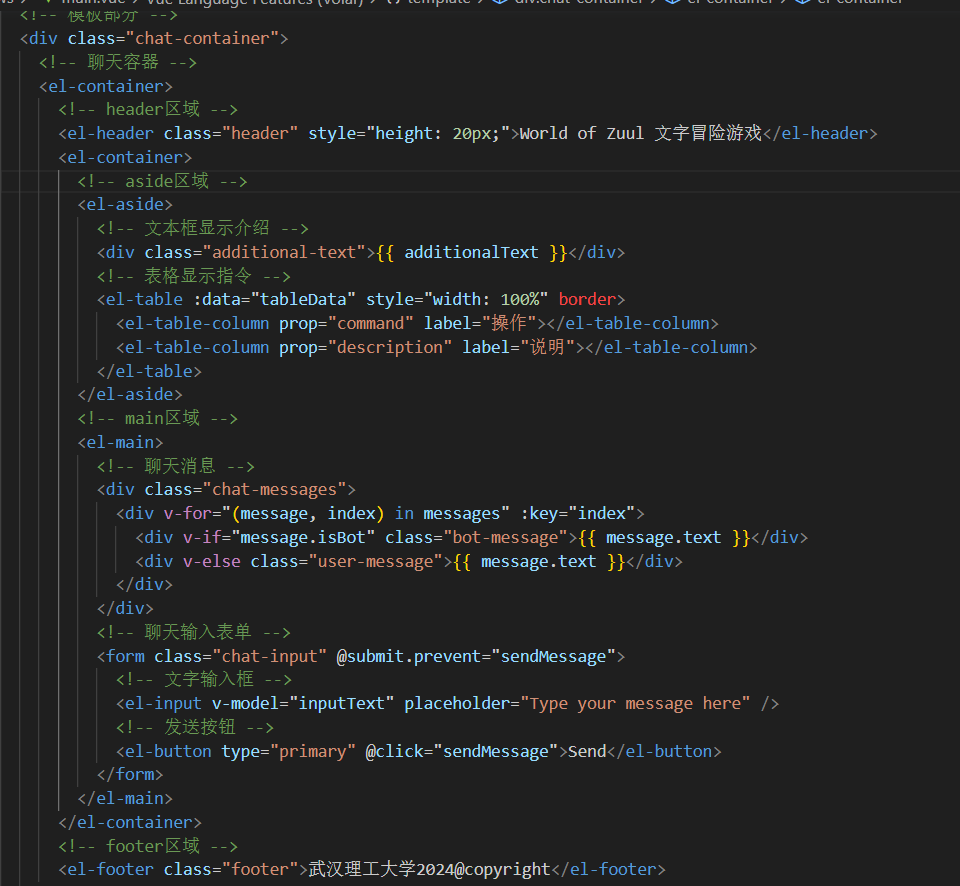


（2）结合组内成员的讨论结果确定并编写提交规范文档COMMIT.md,分工文档WORK.md，分支规范文档BRANCH.md，接口规范文档INTERFACE.md。



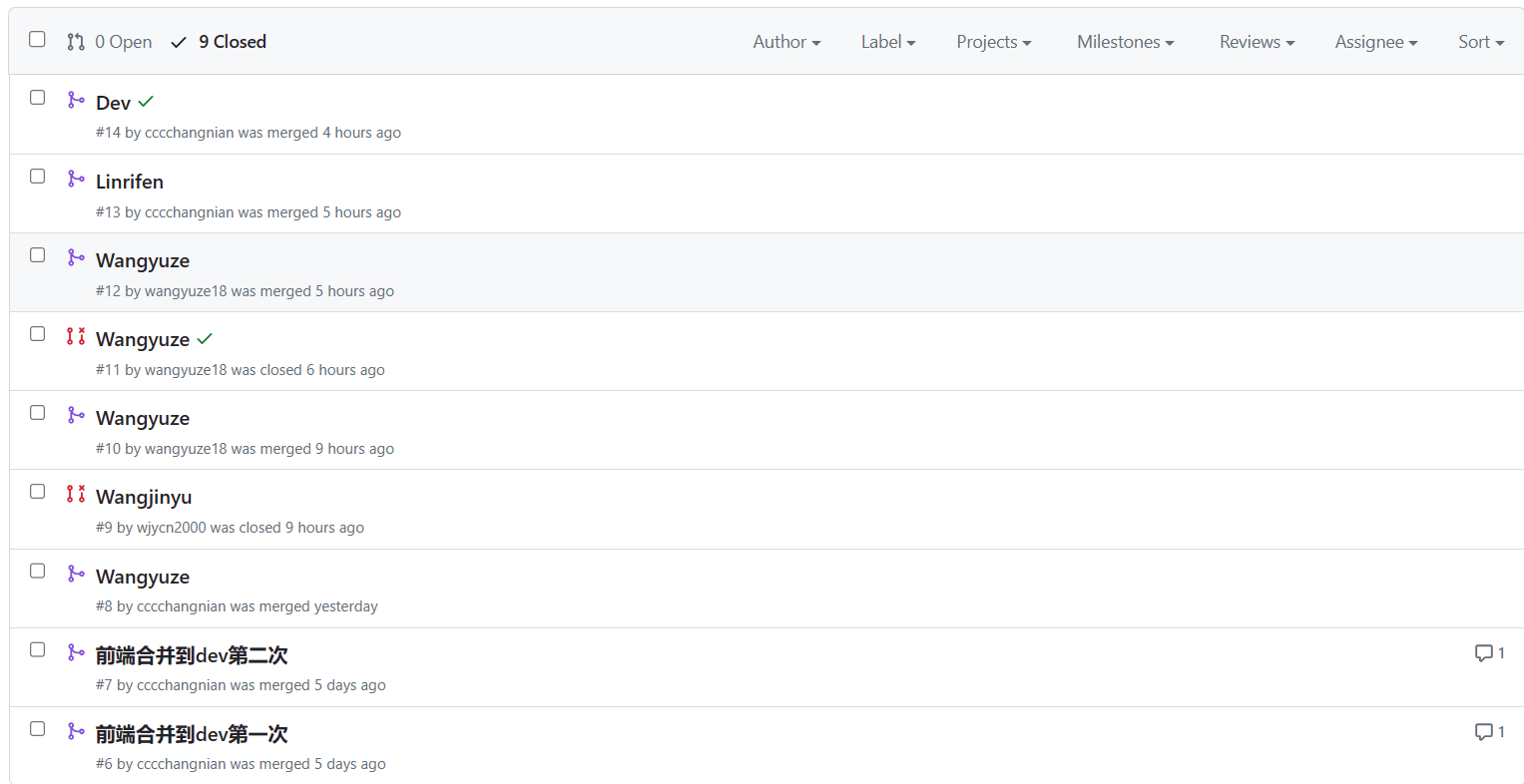
（3）负责完成前端开发，主要是结合vue2和element-ui实现完整的login.vue，register.vue，main.vue的布局及逻辑。设置前端路由信息，确保页面间跳转正常。



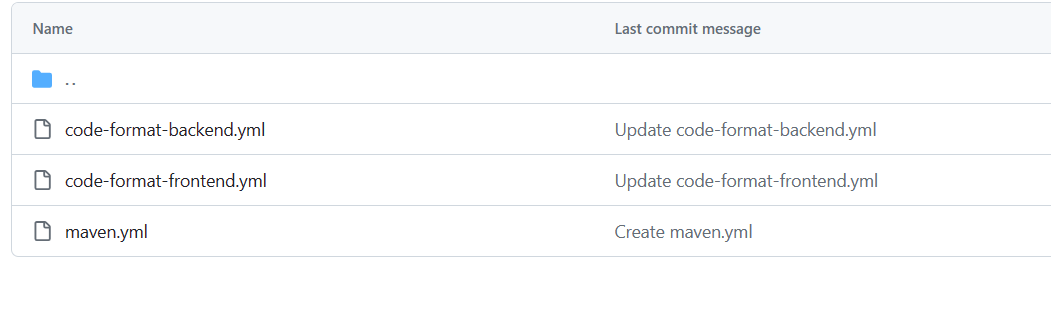


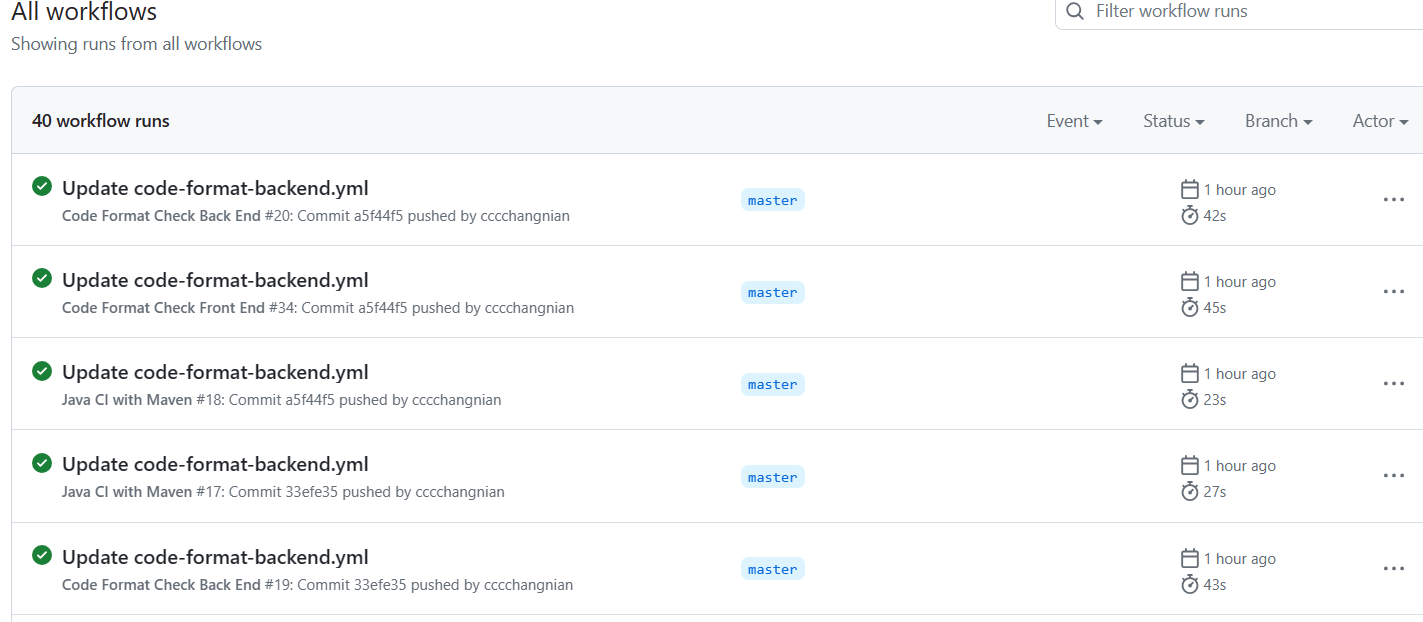


（4）跟进组内成员开发进度，管理分支提交pull\_request，处理冲突并合并pr请求。



（5）完成工作六种yml文件的编写，保证actions成功运行。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。

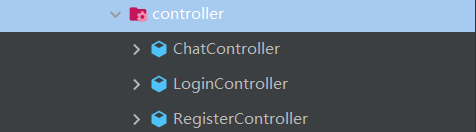




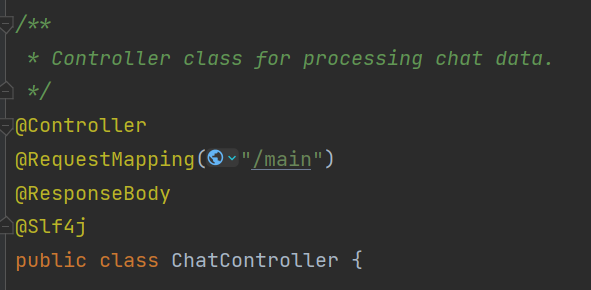
6.2后端设计与开发-王煜泽、王金宇

由于二人合作较为紧密，协同实现了结构及各层的实现，这里进行结合叙述。

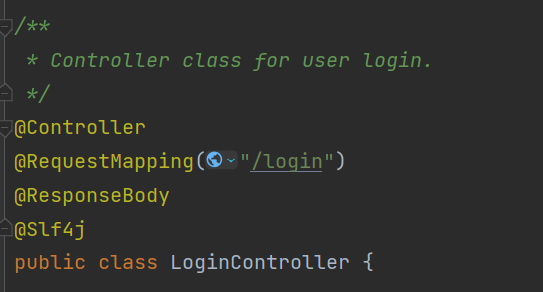
（1）完成Controller层设计和实现



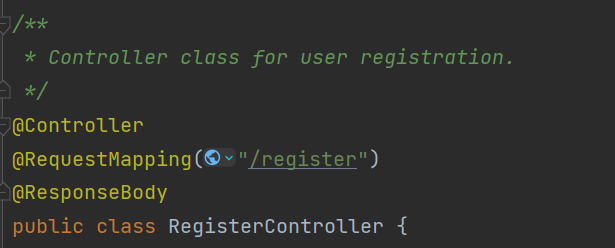
处理聊天消息请求



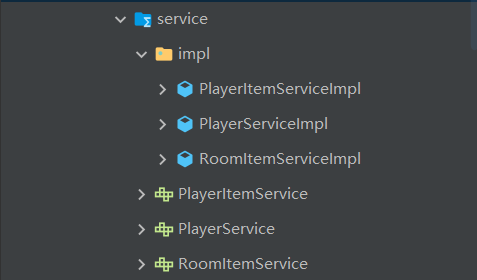
处理用户登录请求



处理用户注册请求

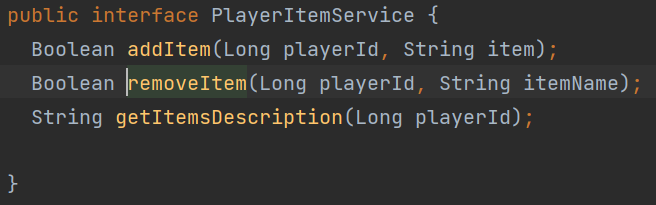


（2）完成Service层设计和实现



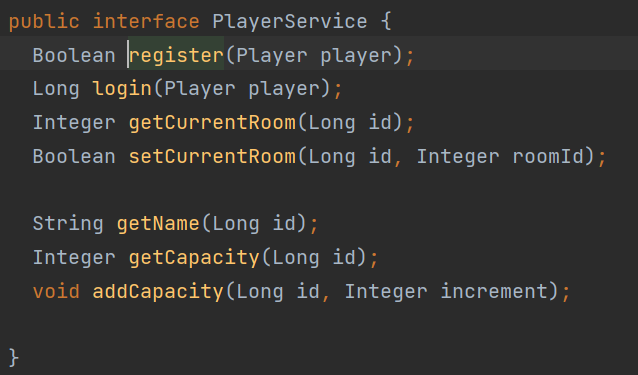
PlayerItemService

处理和PlayerItem相关的业务逻辑，通过DAO与数据库进行交互，根据玩家id，进行添加物品，移除物品，得到物品介绍的功能。



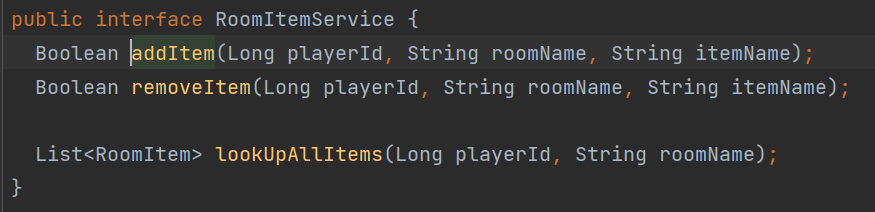
PlayerService

处理和Player相关的业务逻辑，通过DAO与数据库进行交互，实现了登录，注册功能，根据玩家id，进行获取玩家房间，设置玩家房间，获取玩家名称，获取玩家容量，改变玩家容量的功能。



RoomItemService

处理和RoomItem相关的业务逻辑，通过DAO与数据库进行交互，根据玩家id，进行添加物品，移除物品，查看房间物品的功能。

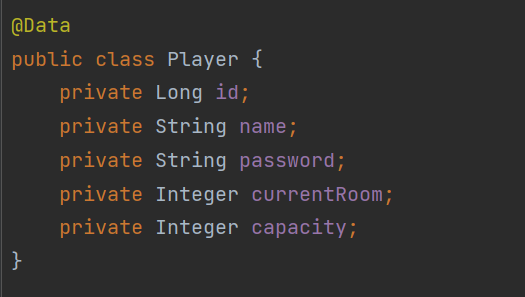


（3）完成DAO层设计和实现

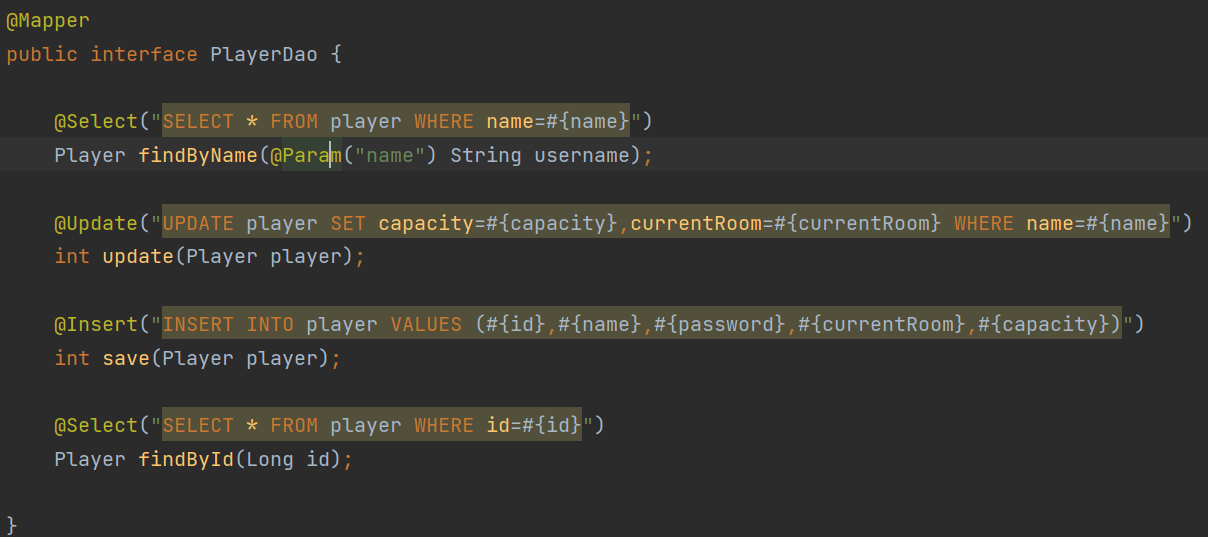


PlayerDao

实体类player

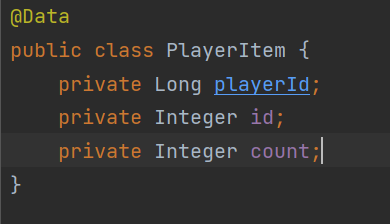


用作"Player"实体的数据访问对象（DAO）,定义了与数据库交互的接口，用于对"Player"实体执行CRUD操作



PlayerItemDao

实体类PlayerItem

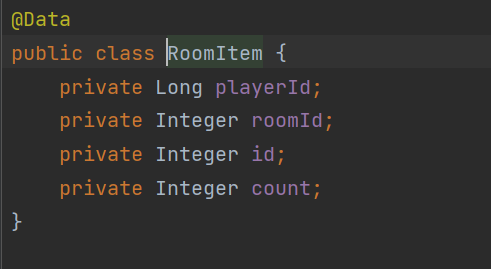


用作"PlayerItem"实体的数据访问对象（DAO）,定义了与数据库交互的接口，用于对"PlayerItem"实体执行CRUD操作



RoomItemDao

实体类RoomItem

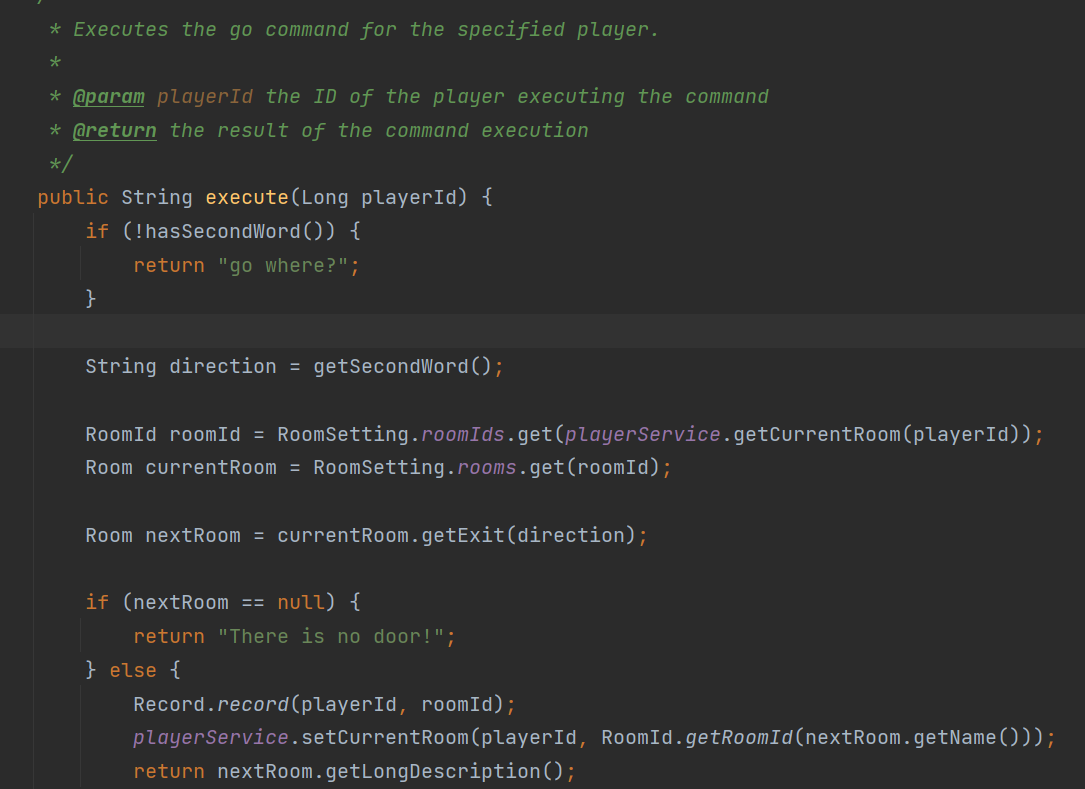


用作"RoomItem"实体的数据访问对象（DAO）,定义了与数据库交互的接口，用于对"RoomItem"实体执行CRUD操作

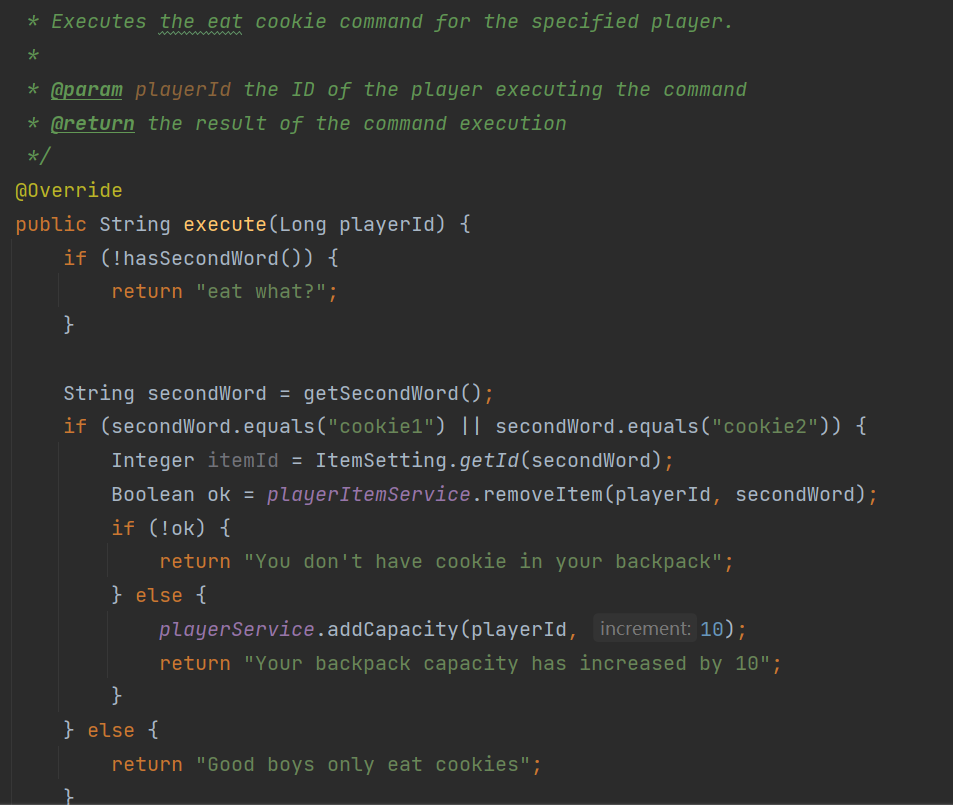


（4）完成Command实现，Command使用了命令模式，实现了接口中的execute功能。

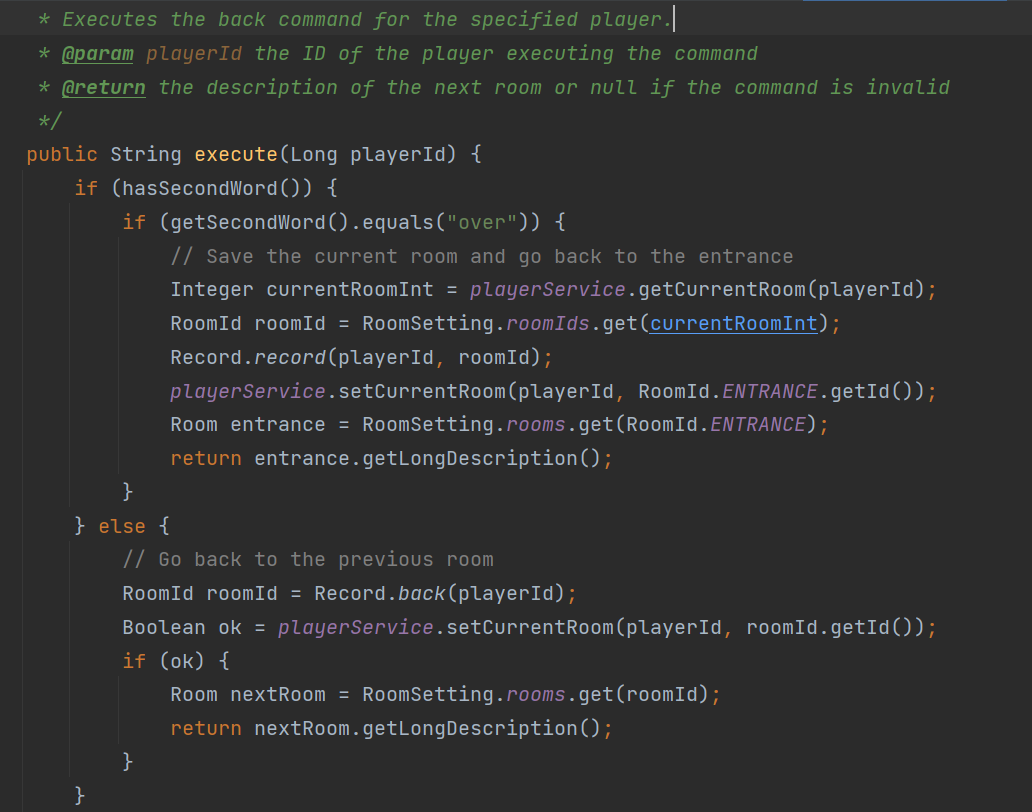
go command



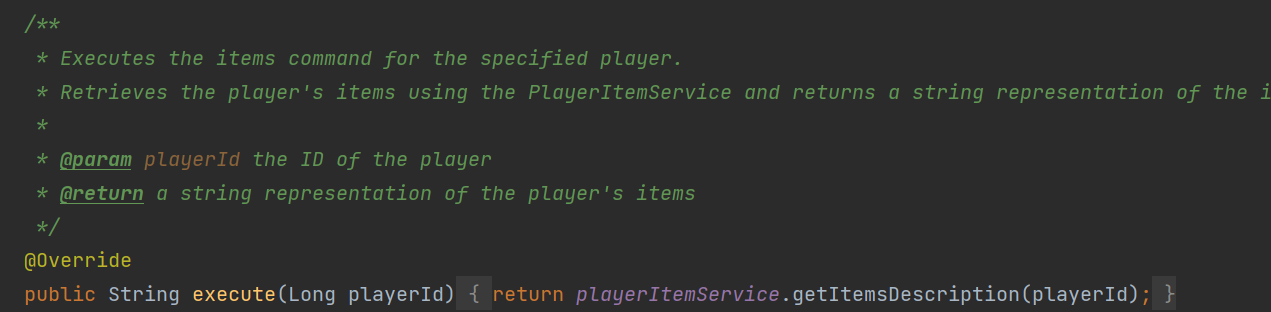
eat command



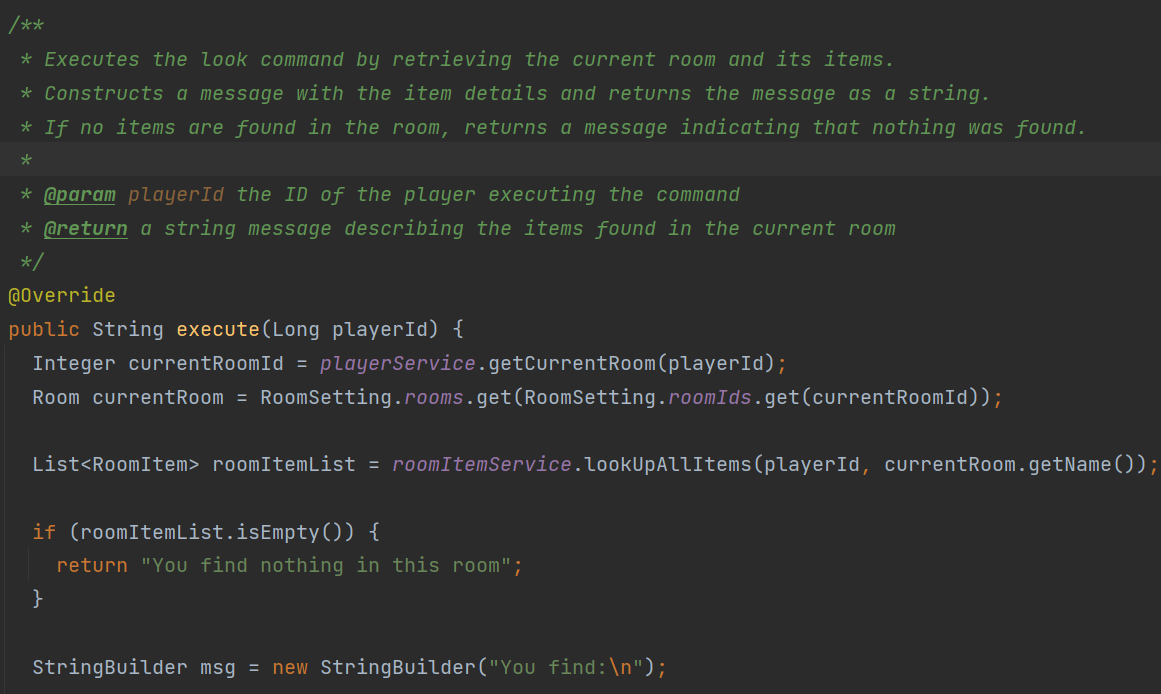
back command

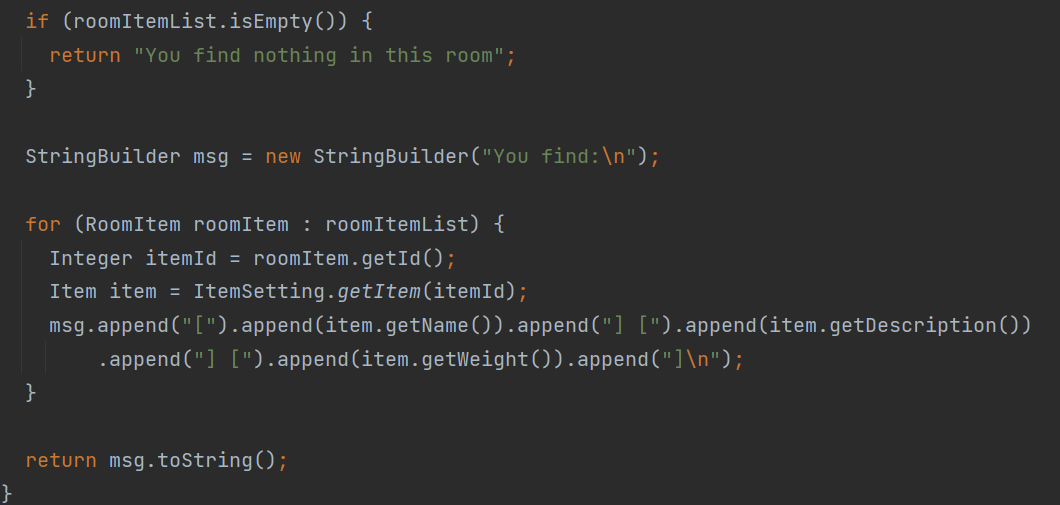


items command

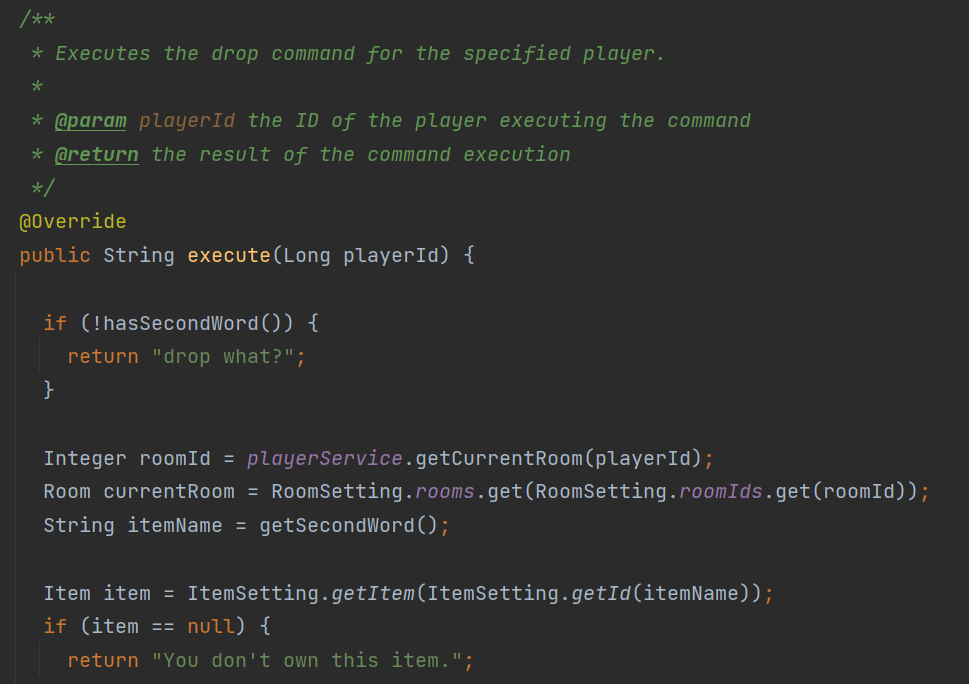


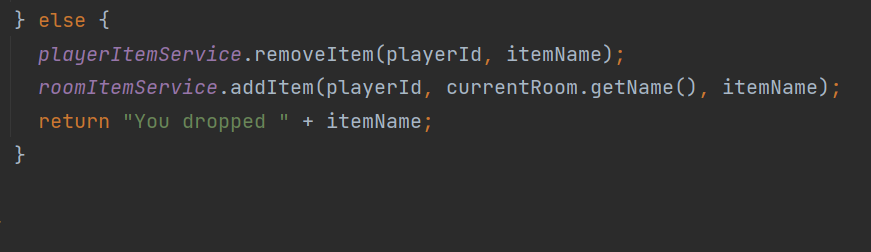
look command





drop command





take command



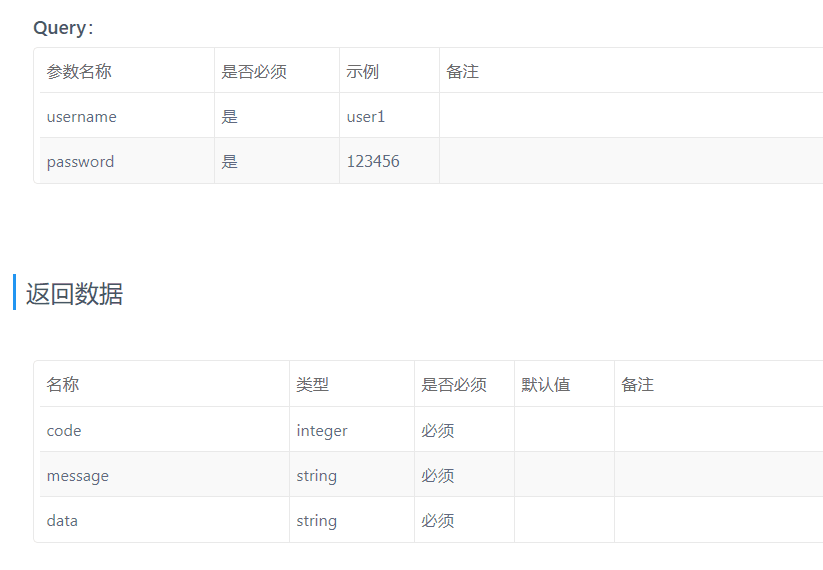
6.4前后端测试及联调人员-王宇轩

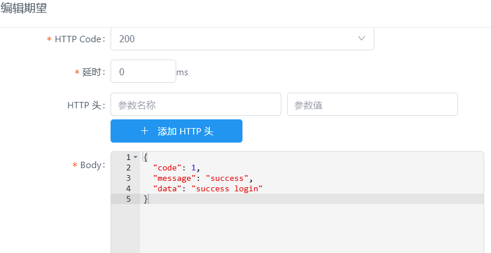
（1）前端测试

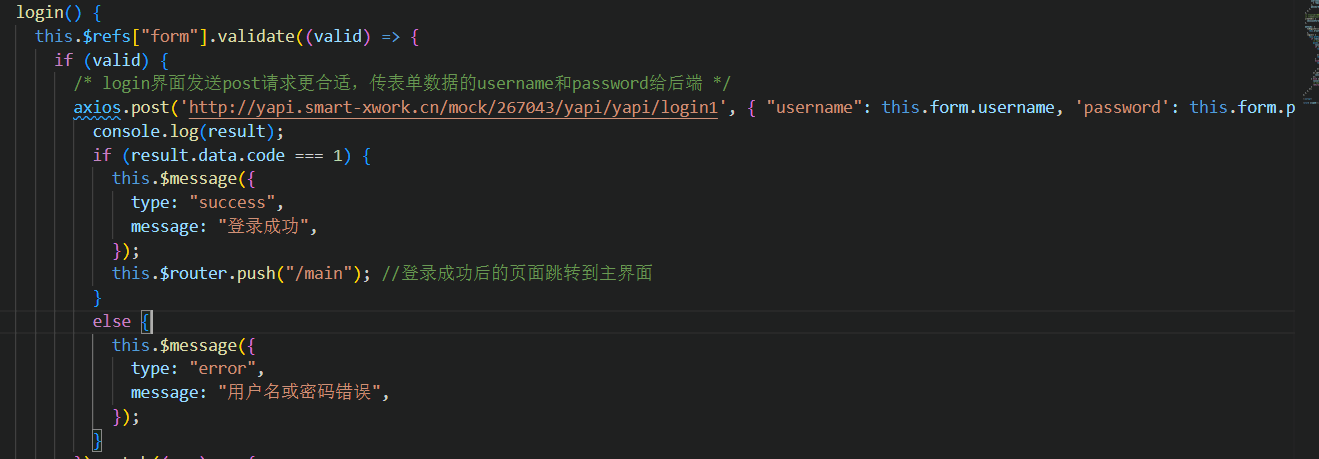
Yapi是一款基于Node.js的开源接口管理平台，主要用于接口管理和接口自动化测试。它提供了接口管理、接口文档、Mock数据、自动化测试等功能，可以帮助开发人员更加高效地管理和测试接口。使用Yapi进行前端接口测试，验证前端与后端API的交互是否正常。

对于登录界面，注册界面，主界面模拟各种场景，包括正常操作、异常操作等，确保前端功能的稳定性和一致性。检查前端界面的布局、样式和交互，确保用户体验良好。

部分测试截图如下：

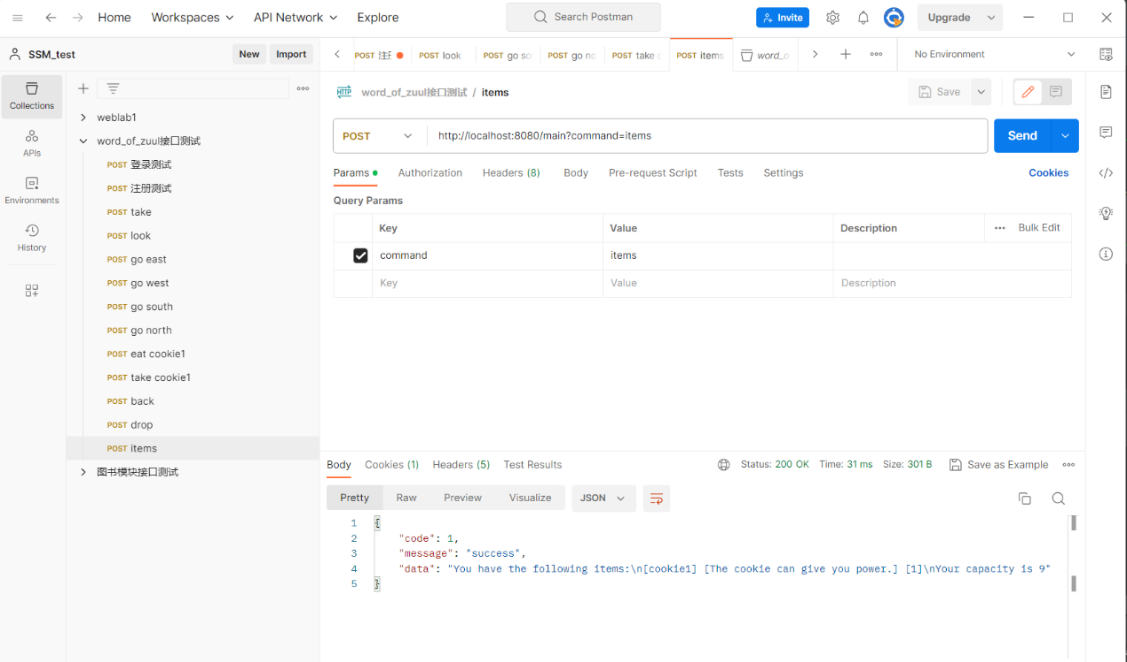
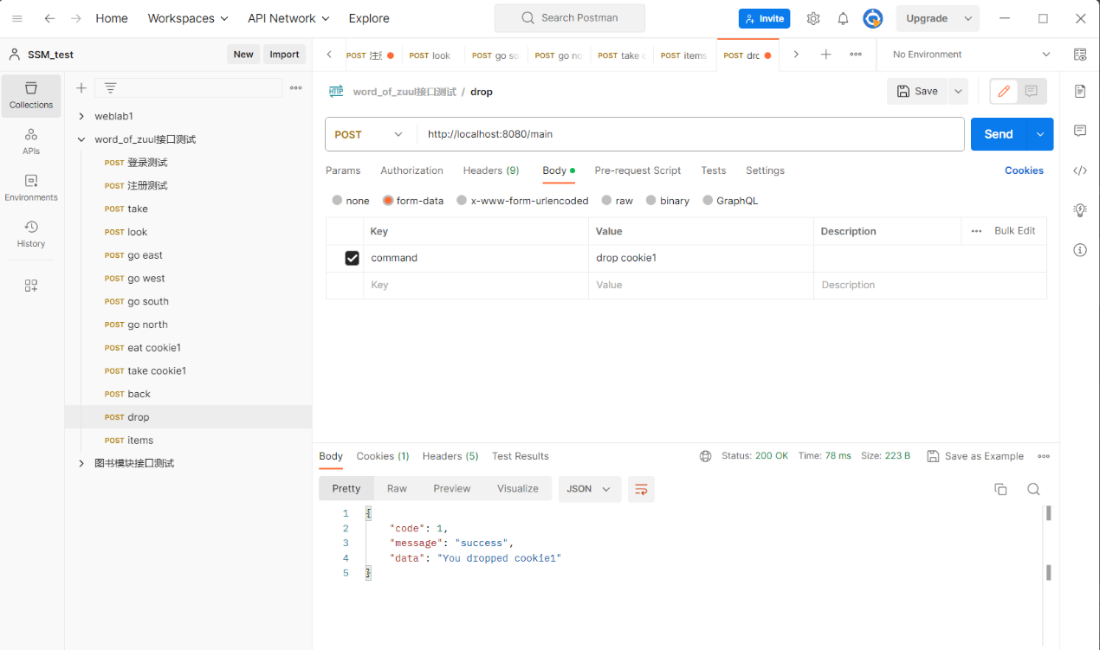
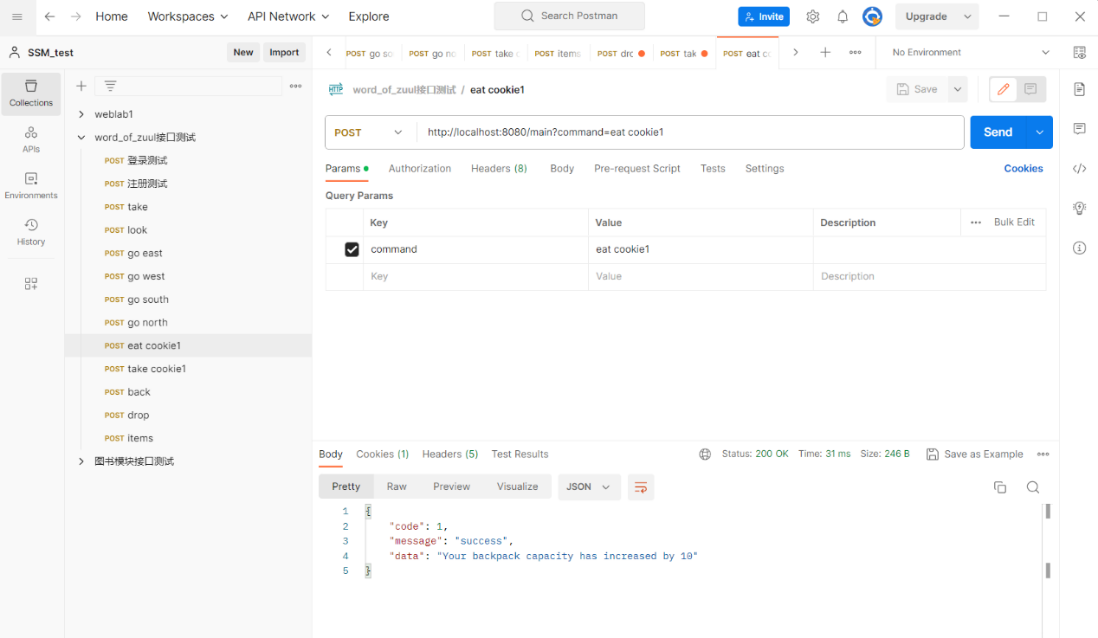
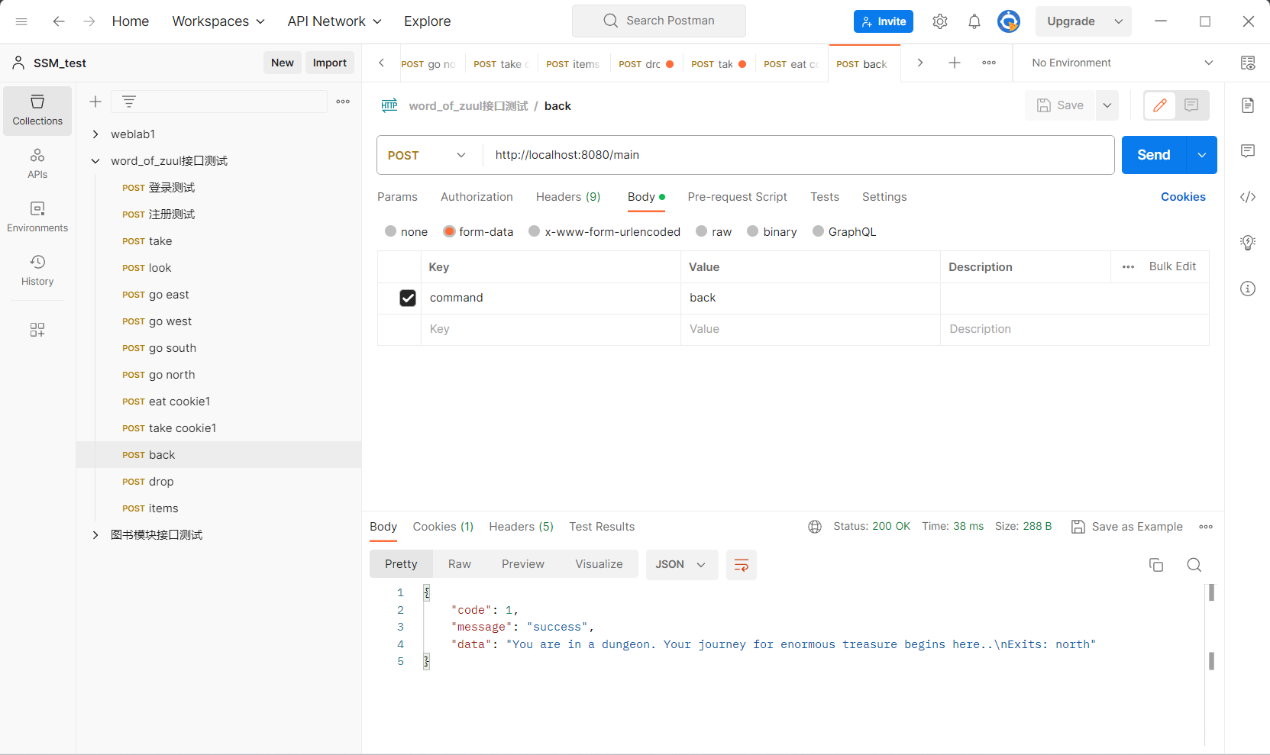
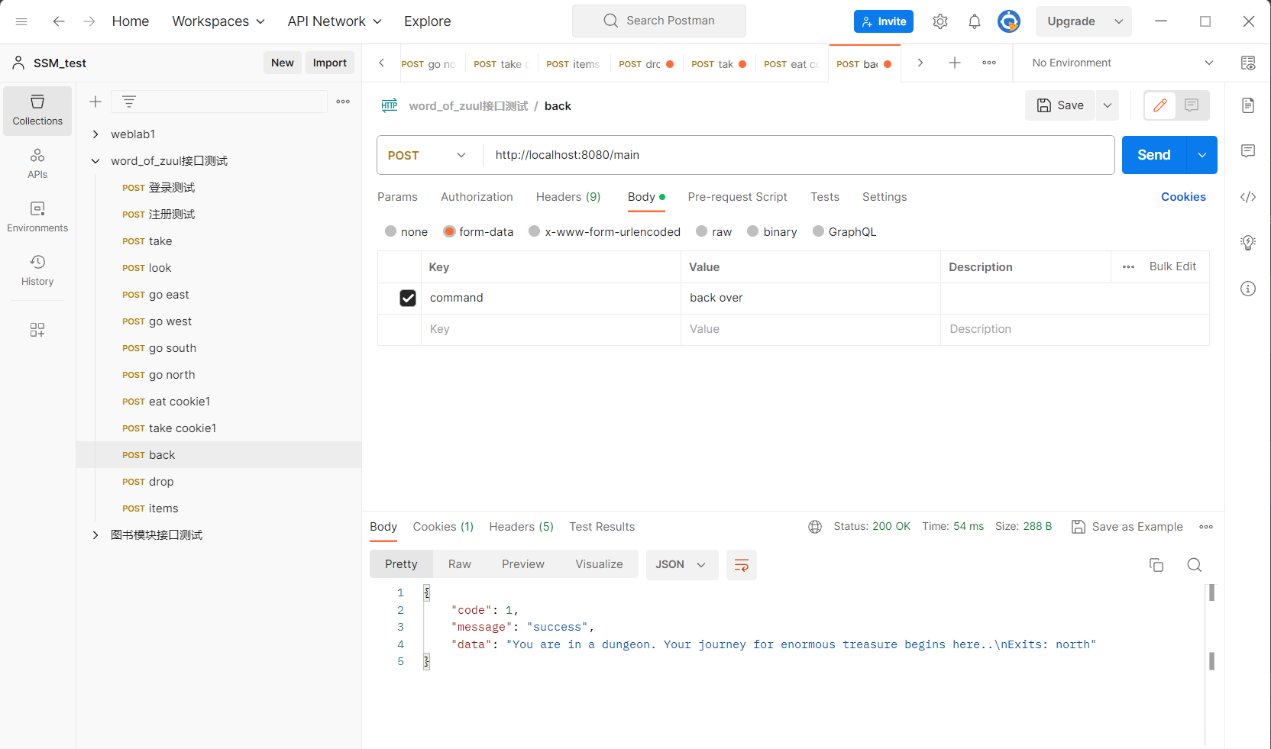






（2）后端测试

Postman 是一款流行的 API 开发和测试工具，旨在简化 API 测试和开发的过程。它提供了一个用户友好的界面，使开发人员能够轻松地创建、发送和调试 HTTP 请求，并查看服务器返回的响应。Postman 支持多种请求方法（如 GET、POST、PUT、DELETE 等），并允许设置请求参数、请求头、身份验证、请求体等。它还提供了丰富的功能，如集合管理、环境变量、测试脚本、数据驱动测试等，帮助开发人员更高效地进行 API 测试和开发工作。



# 7 代码合并

在项目开发过程中，我们充分利用了 GitHub 的 Pull Request 功能进行代码提交和团队协作。我们主要通过以下步骤：

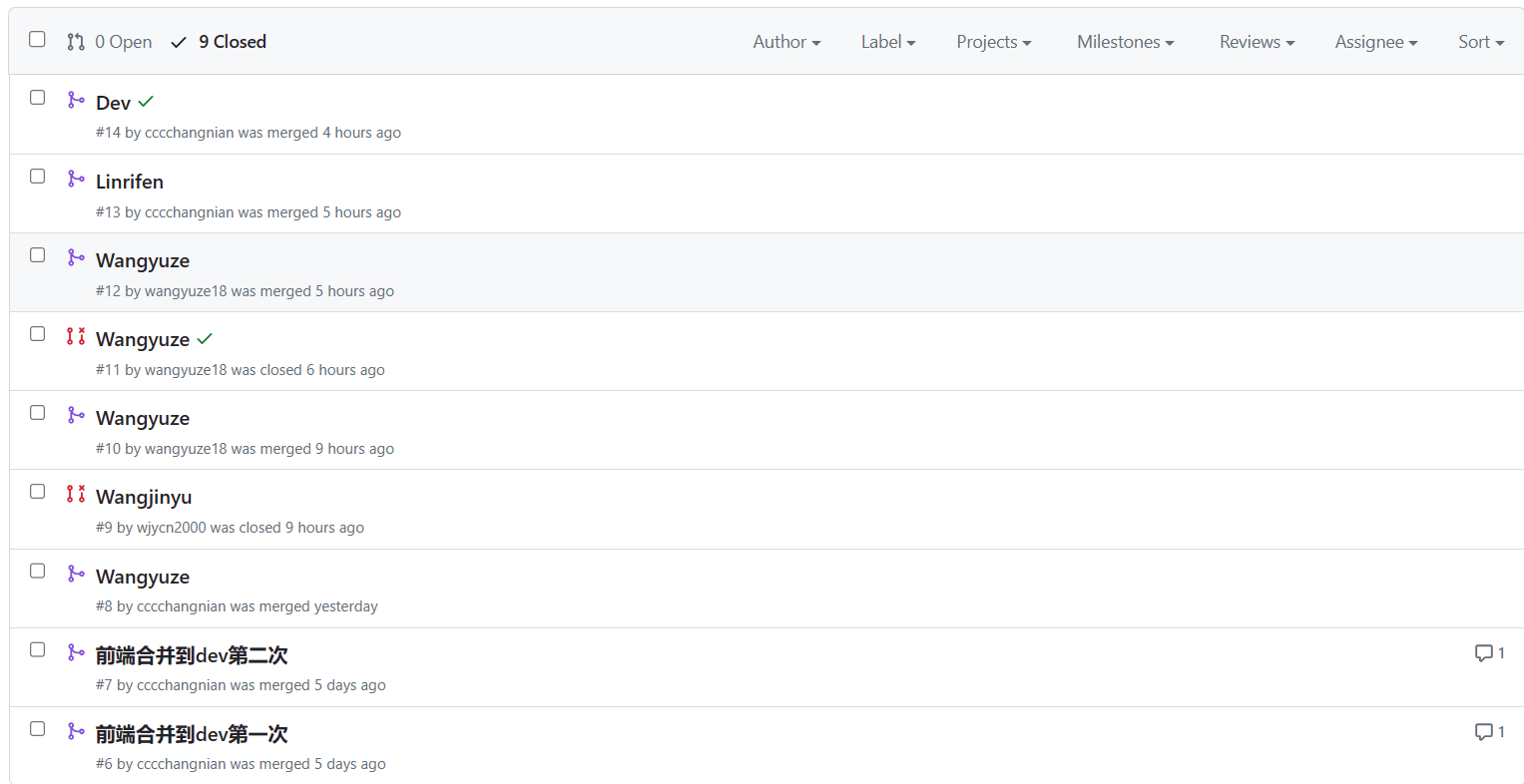
（1）创建 Pull Request：在 GitHub 界面上，我们导航到我们的远程仓库，切换到特性分支，并点击 "New Pull Request" 按钮。我们选择将特性分支与目标分支（通常是 dev 或 master）进行比较。

（2）撰写说明：在创建 Pull Request 时，我们填写了相应的标题和描述。我们详细说明了所做的更改、实现的功能和任何其他相关的信息。我们还与团队成员进行讨论和代码审查，以确保代码质量和一致性。

（3）代码审查和讨论：其他团队成员对 Pull Request 中的代码进行审查，并提供反馈和建议。我们使用 GitHub 的评论功能进行讨论和交流，解决可能存在的问题和改进代码。

（4）完成并合并：在经过团队成员的审查、讨论和必要的修改后，我们确认 Pull Request 中的代码符合要求并可以安全合并到目标分支。我们点击 "Merge" 按钮，将特性分支的代码合并到目标分支中。

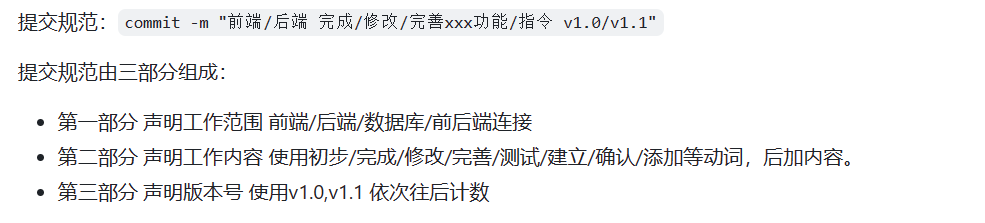
（5）完成 Pull Request：在将代码合并到目标分支后，我们关闭 Pull Request，并记录相应的注释和说明。



# 8 开发规范与测试

8.1 提交规范

在项目开发前期准备工作，团队成员进行讨论，共同确定提交标准及规范，完成提交规范文档COMMIT.md。





8.2接口规范

在项目开发前期准备工作，团队成员进行讨论，共同确定前后端接口及规范，完成接口规范文档INTERFACE.md。



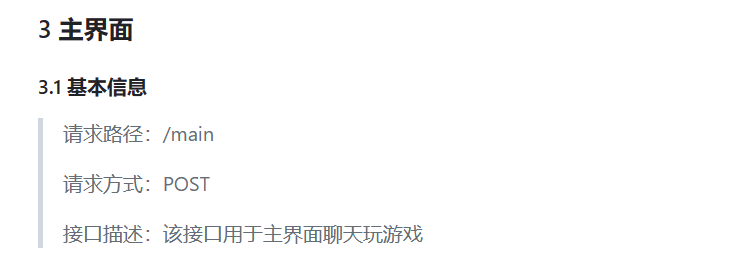














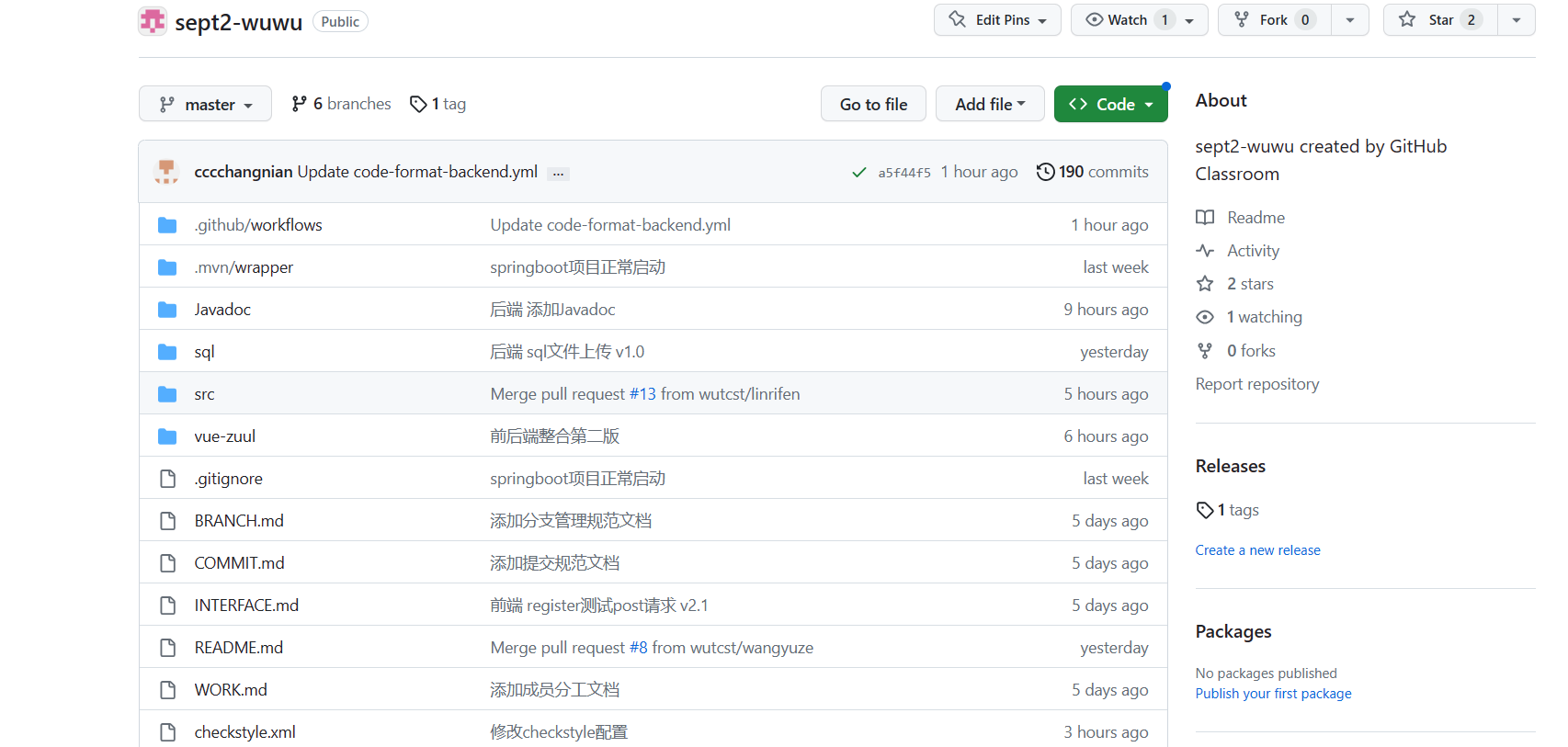


# 9 集成与版本发布

9.1 集成

项目集成是指将不同的组件、模块或子系统整合在一起，形成一个完整的系统或应用程序。在软件开发中，项目集成通常涉及将不同的代码库、服务、第三方库、组件和其他资源整合在一起，以创建一个统一的解决方案。

代码开发完毕后，代码库、其他资源等集成到master分支。



9.2 版本发布

添加标签（tag）并发布版本是在版本控制系统中为特定的代码状态打上标记，并将其作为一个特定版本进行发布。这有助于团队和用户明确了解代码的状态和版本，并可以方便地进行代码回滚或对特定版本进行部署。

为完成后的项目添加tag并提交版本v1.0.0。

