基于 AI 的智能斗地主平台

需求(分析)说明书(规约)

修订历史记录

编写日期	SEPG	版本	说明	作者
2019.11	GF	1.0	初稿	崔垚，杨雨奇，蒋伟博，吴国栋
2019.12	GF	2.0	终稿	崔垚，杨雨奇，蒋伟博，吴国栋

目录

1. 引言 4

1.1. 背景 4

1.2. 参考资料 4

1.3. 用户的特点 5

2. 功能需求 5

2.1. 系统范围 5

2.2. 系统体系结构 6

2.3. 系统总体流程 8

2.4. 需求分析 10

3. 非功能需求 30

3.1. 性能要求 30

3.2. 数据管理能力要求 31

3.3. 安全及保密性要求 31

3.4. 其他专门要求 31

4. 运行环境规定 32

4.1. 设备 32

4.2. 支持软件 33

4.3. 需求跟踪 ..... 错误!未定义书签。

1. 引言

1.1. 背景

a．待开发的软件的名称：欢乐斗地主 --基于 AI 的 WEB 多人游戏平台

b．项目的任务提出者、开发者：项目组成员

用户：任何已在本游戏平台注册的互联网用户

运行环境：

MySQL5.5.3 及以上版本；

JDK1.8 以上版本；

IDE 使用 IDEA 编译运行。

c．该软件系统同其他系统或其他机构的基本的相互来往关系：无

1.2. 参考资料

[1] IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.

[2] Software H. Example Use Case Specification[R/OL]. [2013-11-15]. http://www.hippo- software.co.uk/downloads/Example%20Use%20Case%20Specification.pdf.

[3]软件工程：实践者的研究方法（第八版）Roger S. Pressman 著

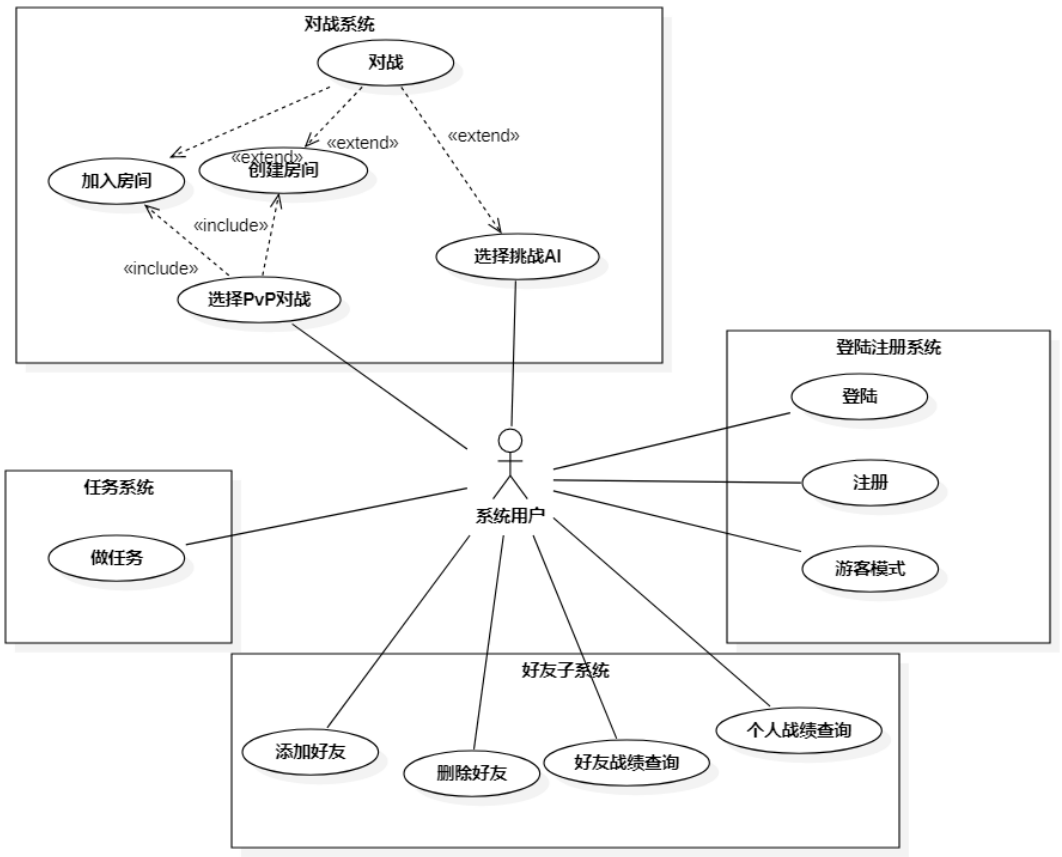
1.3. 用户的特点

年龄，职业，教育水平涉及广泛，爱好斗地主游戏或想要通过 AI 提升技术水平

预期使用频度：一天多次

2. 功能需求

2.1. 系统范围



整个系统主要包括对战系统，好友系统，登录注册系统，任务系统。3 个子系统。对应所包括的功能点如上。

2.2. 系统体系结构

groupId	artifactId
org.springframework.boot	spring-boot-starter-web
org.springframework.boot	spring-boot-starter-websocket
org.mybatis.spring.boot	mybatis-spring-boot-starter
mysql	mysql-connector-java
com.alibaba	fastjson
org.springframework.boot	spring-boot-devtools

要实现的架构是用源/架构编写的。让我们看看项目中包含的依赖项。

1. Spring Boot Starter 网站。 它允许我们创建一个基于 Spring Boot 的 MVC Web 程序。

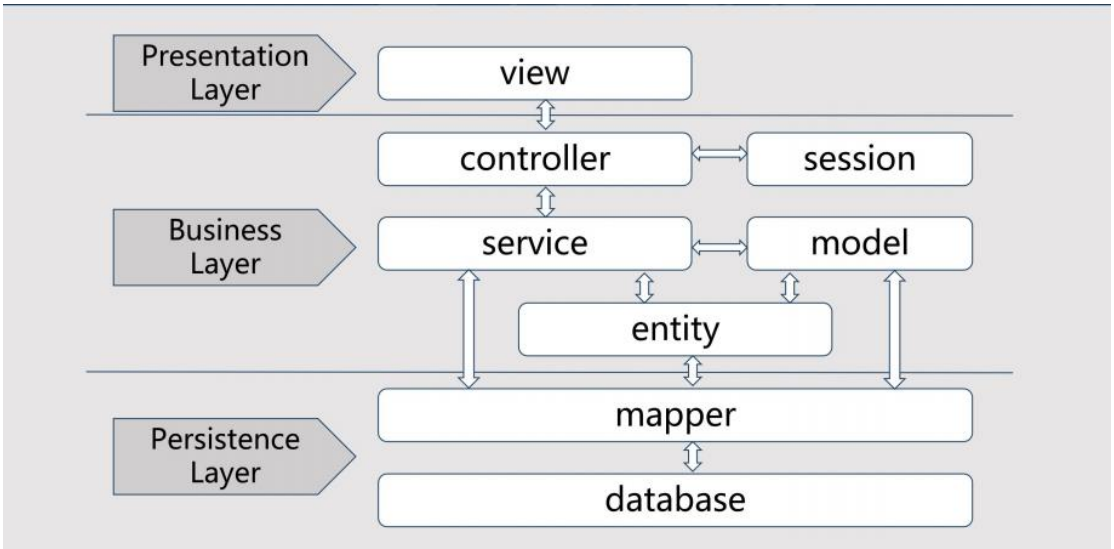
2. 网络套接字启动器。 它允许会话进行全双工连接，这意味着浏览器和服务器并发送消息种子主动消息，只要保持会话连接活。

3. mybatis。 这是一个广泛使用的持久层框架，用户包括阿里巴巴等主流公司。

4.数据库我们使用 MySql 作为存储库，使用 jdbc 进行 Java 连接。

5.我们以 json 格式来回发送消息。 因此，我们还导入了 json 依赖项。

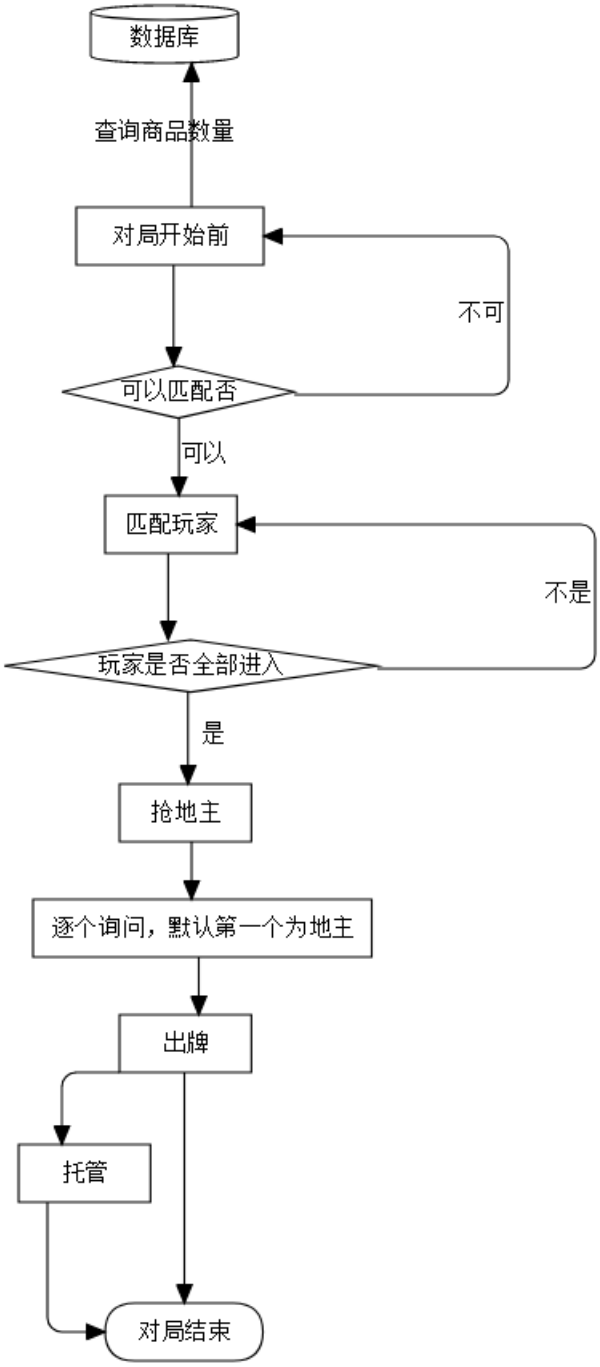
6. Dev-tools 用于热部署，编译器可以在其中重新构建并重新运行项目时间很短。



上图是我们的主要架构。我们采用了经典的 3 层 Web 框架。

当消息从表示层到达业务层时，第一个模块是控制者。首先负责取消请求并处理消息开始。它还用于控制网络连接和响应发送。例如，在 Web 套接字，控制器需要记住会话，以便以后使用消息交换。然后，请求进入服务层。它用作操作逻辑处理。最简单的在不涉及特定数据的情况下，它可以计算然后返回结果而无需持久层。但是，这种情况越来越少，因为我们可以考虑到 BS，几乎可以使用 JavaScript 在表示层完成所有这些操作框架项目。当涉及数据库连接时，使用映射器进行数据库访问管理，有时不同线程中的服务可能需要数据库同时。关于发送和接收对象，除了普通对象（如 int 或字符串，一个实体对象适用于此功能。对于一些不需要的临时对象存储在数据库中，将它们存储到模型中，它们就可以进行数据自我管理。另外，模型可以与持久层交互。

2.3. 系统总体流程



对局开始前

后端查询，仓库中的商品有多少，并返回给前端。如果一个商品没有，则显示购买，否则显示使用。对于一天记牌器等，如果已经使用了，那么使用的按钮不可点击。点击使用后可以撤销。直到点击开始游戏之后，才向后台发送使用请求。

匹配玩家

牌局选项一共有 3 个按钮，一共可以组成 2^3=8 个牌局选项，因此，我们需要在后台构建 8 个池子。一旦玩家进入

抢地主

向第一个玩家询问是否需要抢地主。等待时间 10 秒。如果玩家没有按时选择，则默认不抢地主。其他人显示该人剩余的秒数。可以把数秒当成一个迭代点。如果三个人都没有选择地主，则默认第一个人为地主。后续：可以重新发牌，让大家重新选。如果三轮都没有人选择，默认是第一个人为地主。

出牌

如果第一次出牌，或者一轮刚结束，则只显示出牌按钮。否则在牌局进行中，显示 不出 提示 出牌 三个按钮。后续：出牌的人显示数秒，时限 10s。其他用户也可以看到数秒。在如果第一次出牌，或者一轮刚结束时，默认按电脑提示的出牌。如果在牌局进行中，默认不出。

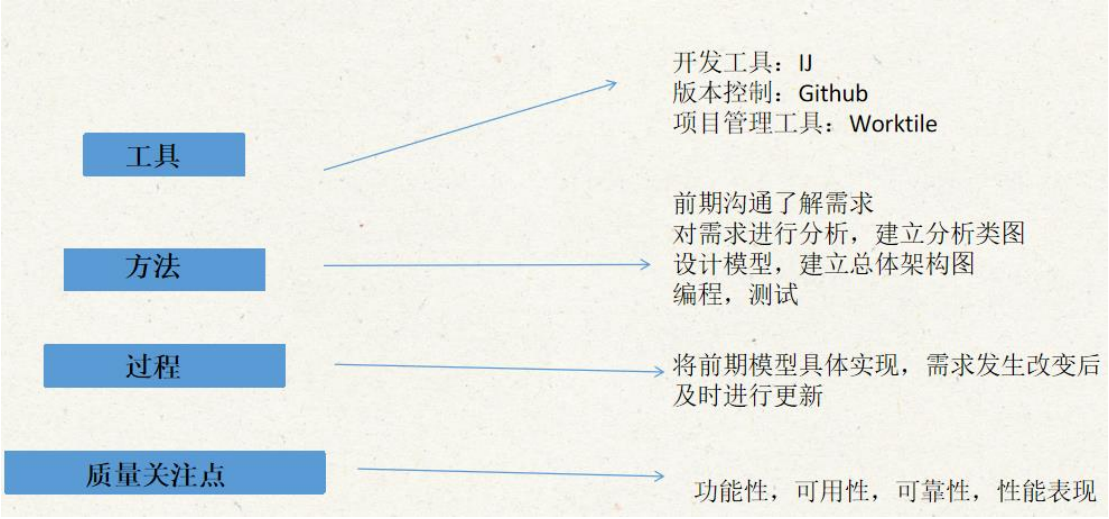
托管

前端变为”退出托管“。程序会在后台记录该操作。轮到它时，前端不显示操作栏（数字 不出 提示 出牌），直接由后端计算结果，并返回给各个前端。需要在每局对战结束后清除前后端托管状态

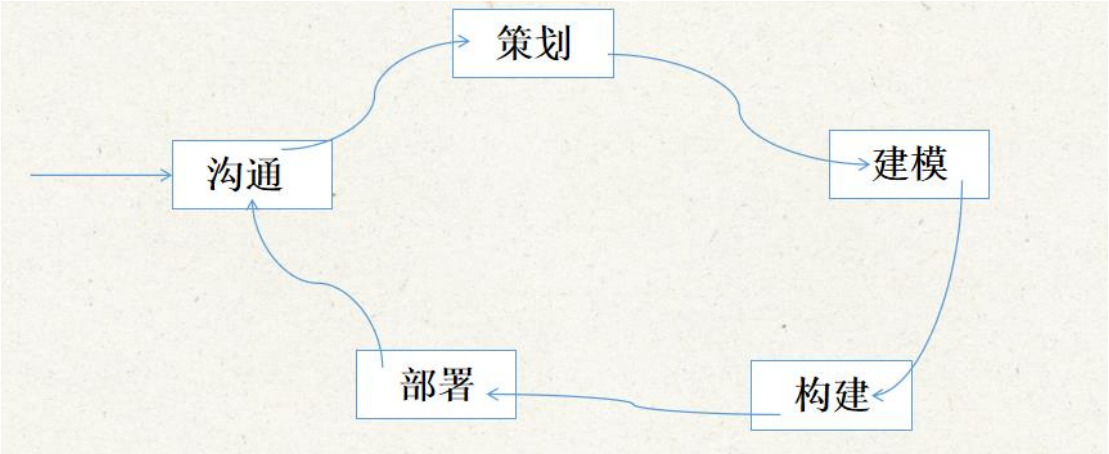
对局结束

2.4. 需求分析

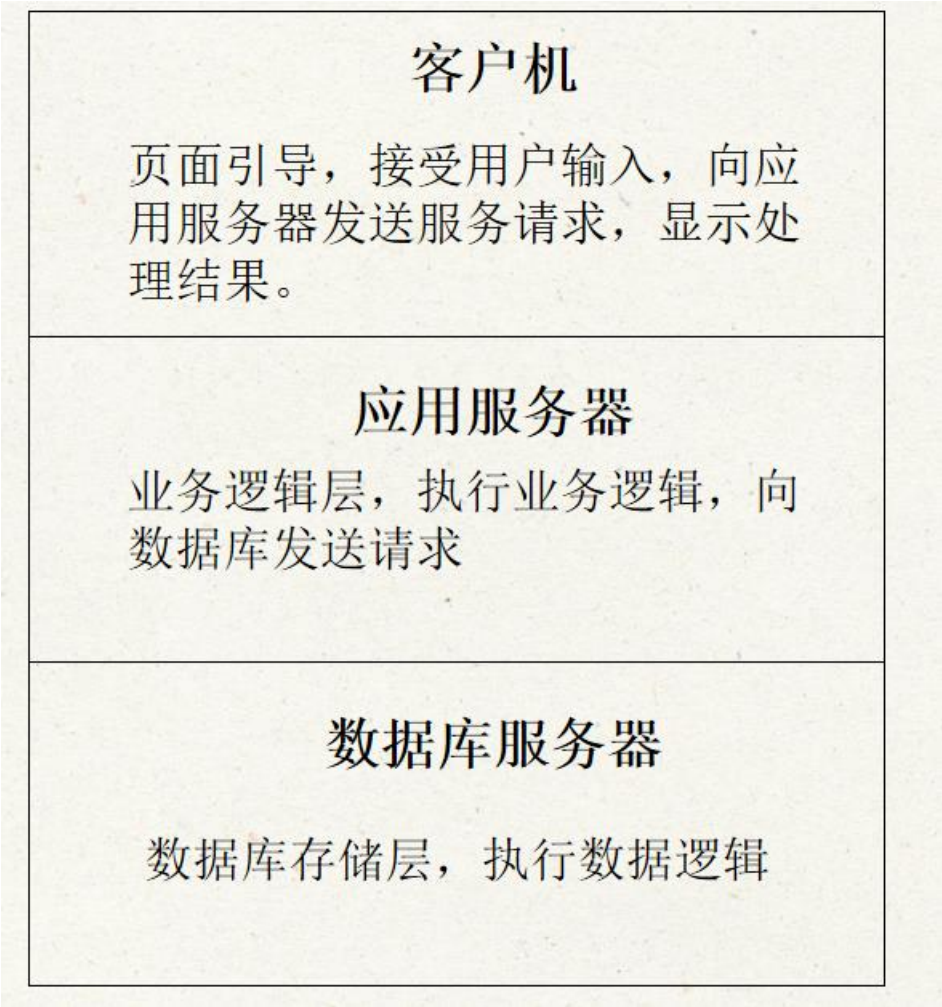
基本层次图



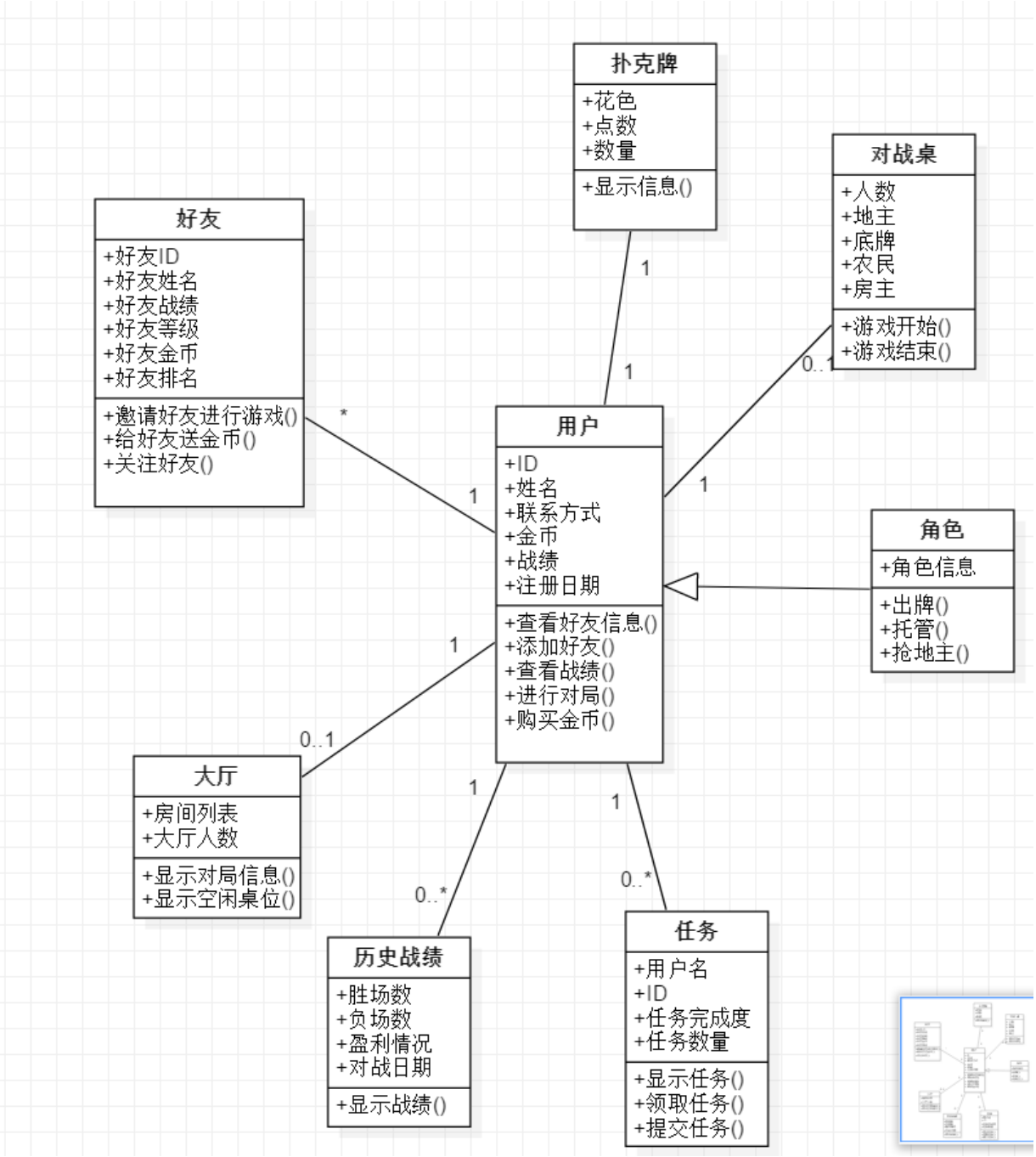
过程流



三层 B/S 架构


客户机
页面引导，接受用户输入，向应用服务器发送服务请求，显示处理结果。
应用服务器
业务逻辑层，执行业务逻辑，向数据库发送请求
数据库服务器
数据库存储层，执行数据逻辑

分析类图

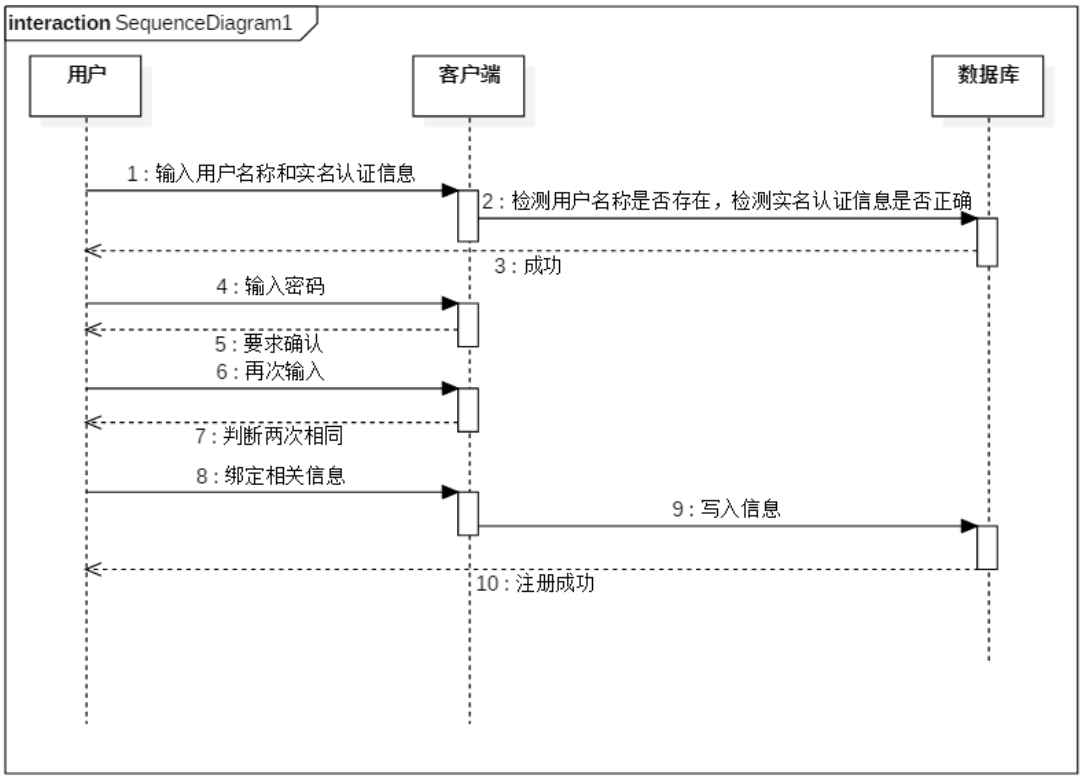


2.4.1. 功能建模 （活动图，时序图）

2.3.1 登录注册模块

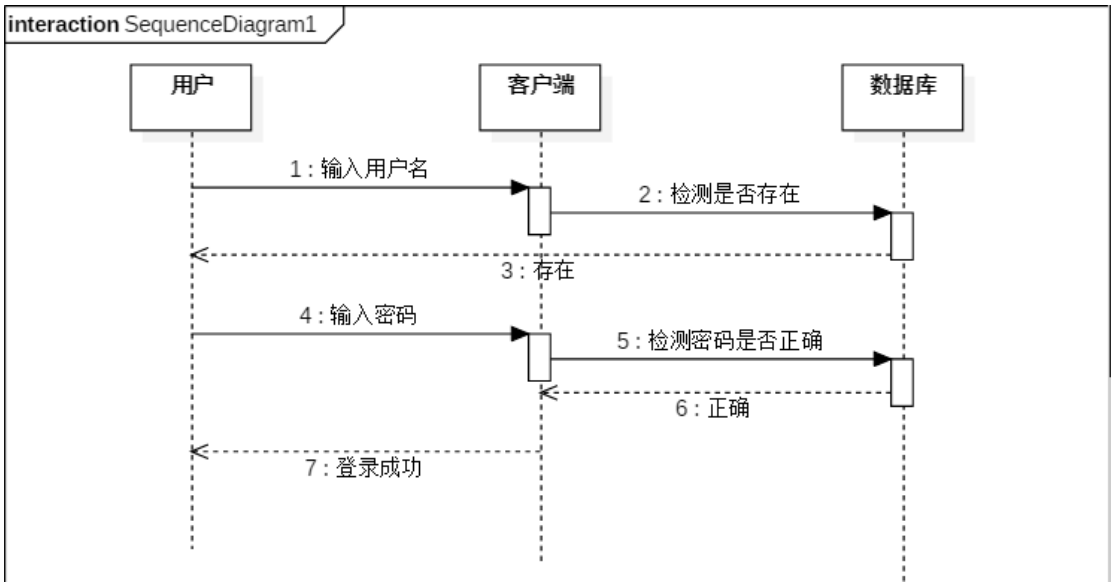
用例编号	01
用例名称	注册
文字描述	用户输入账号，系统验证账号是否可用，可用则继续，否则需要重新输入；而后系统根据用户输入的账号发送验证码，用户填入收到的验证码，系统检查验证码是否正确，正确则继续，否则需要重新输入；然后用户输入密码、重复密码，系统检查密码格式、重复密码是否相同等，是则通过，否则重新输入；最后用户点击注册按钮，注册成功，跳转至登录页面。
被包含用例	无
被扩展的用例	无
频度	第一次登陆时
主要角色	一般用户
前置条件	系统处于工作状态 用户进入注册界面
后置条件	获得唯一标识的账号
主成功场景	1.用户输入用户名称和实名认证信息 2.用户输入密码 3.用户再次输入密码

	4.用户绑定相关账号 5.系统创建账号，注册完成
可选操作流程	1a.用户名和实名信息不符，提示重输 2a.用户输入密码不符合格式要求，提示重输 3a.两次密码输入不同，提示重输



用例编号	02
用例名称	登录

用例描述	用户输入账号，系统检查账号格式是否正确，是则进行下一步，否则重新输入；而后用户输入密码，系统检查密码是否正确，是则进行下一步，否则重新输入；最后点击登录按钮，进入网站主页。
被包含用例	无
被扩展用例	无
主要角色	一般用户
前置条件	系统处于工作状态 用户进入注册界面
后置条件	成功登陆系统
主成功场景	1.用户输入用户名 2.用户输入密码 3.登录成功
可选操作流程	1a.用户名不存在提示重输 2a.密码错误，提示重输



用例编号	03
用例名称	游客模式
用例描述	在该种情况下，用户可以在未注册的情况下体验游戏，但客户端并不会保存相关数据，每次以一个空用户的形式登录。
被包含用例	无
被扩展用例	无
主要角色	一般用户
前置条件	系统处于工作状态 用户进入初始界面
后置条件	成功以未注册状态登录系统

主成功场景	1.用户进入主界面 2.选择游客模式 3.登录成功
可选操作流程	无

2.3.2 对战模块

用例编号	04
用例名称	选择 PvP 对战
用例描述	用户在登录状态下选择与其他真实玩家进行对战，需要在线匹配等待
被包含的用例	创建房间，加入房间
被扩展的用例	无
主要角色	一般用户
前置条件	系统处于工作状态 用户成功登陆
后置条件	返回一组房间列表给用户

主成功场景	1.用户成功登陆 2.选择 PvP 对战 3.进入 PvP 大厅
可选操作流程	1a.用户未登录，提示先登录

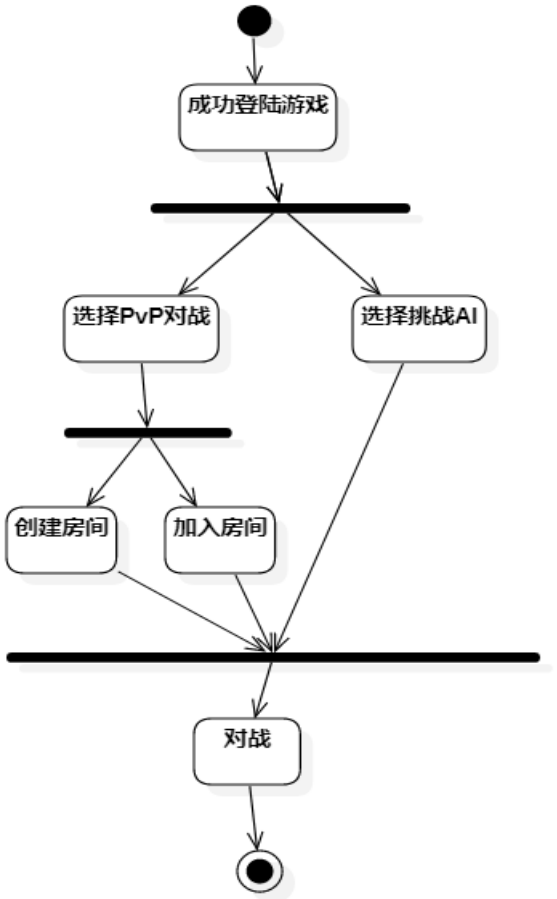
用例编号	05
用例名称	选择挑战 AI
用例描述	用户可以在登录状态下选择与服务器的两个 AI 进行对战游戏
主要角色	一般用户
被包含用例	无
被扩展用例	对战
前置条件	系统处于工作状态 用户成功登陆
后置条件	进入房间与 AI 对战
主成功场景	1.用户成功登陆

	2.选择挑战 AI 3.进入房间开始对战
可选操作流程	1a.用户未登录，提示先登录

用例编号	06
用例名称	加入房间
用例描述	用户在选择 PvP 对战后，选择加入已存在的房间进行匹配对战
主要角色	一般用户
前置条件	系统处于工作状态 用户选择 PvP 对战 存在可用房间
后置条件	进入房间准备对战
被包含的用例	无
被扩展的原理	对战
主成功场景	1.进入 PvP 大厅界面 2.选择房间加入

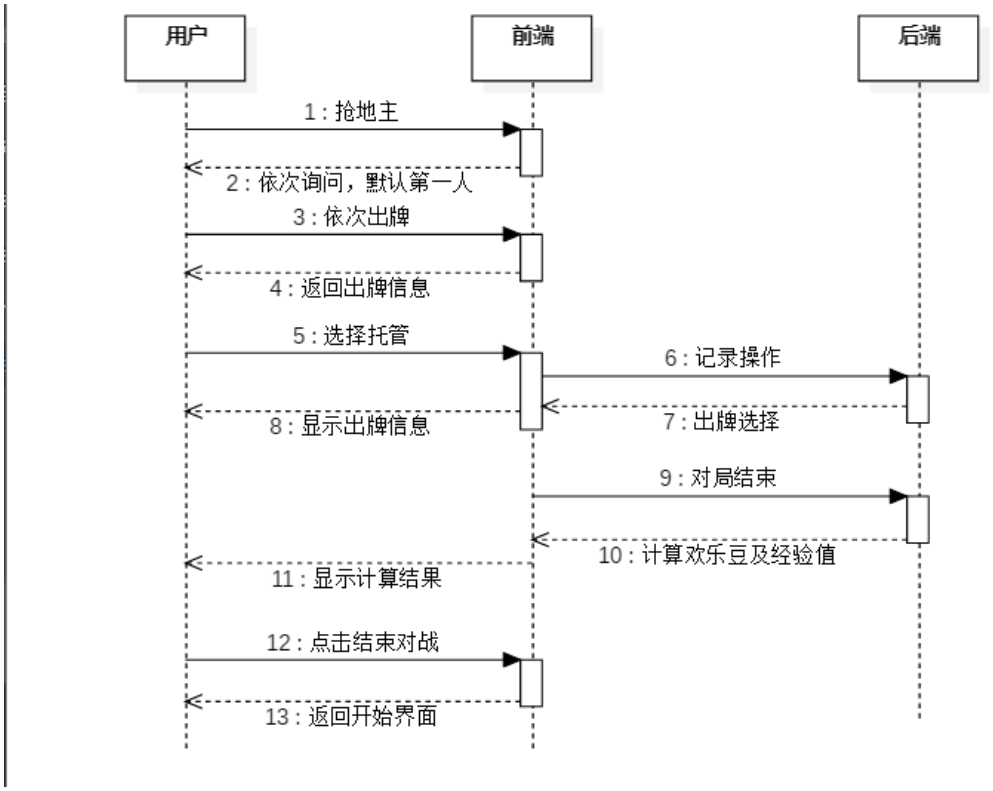
	3.准备开始游戏
可选操作流程	2a.若不存在房间，则无法加入只能创建房间

用例编号	07
用例名称	创建房间
用例描述	用户在选择 PvP 对战后，希望自主创建房间开始游戏对战
被包含的用例	无
被扩展的用例	对战
主要角色	一般用户
前置条件	系统处于工作状态 用户选择 PvP 对战
后置条件	生成新的房间并加入
主成功场景	1.进入 PvP 大厅界面 2.创建房间 3.进入房间准备开始游戏
可选操作流程	无



用例编号	08
用例名称	对战
用例描述	用户在登录后可选用两种方式开始对战
主要角色	一般用户
被包含用例	无
被扩展用例	无
前置条件	系统处于工作状态 对局开始
后置条件	用户成功完成一局对战

主成功场景	1.抢地主，依次询问 2.出牌（可选择托管） 3.对局结束，显示信息 4.返回开始界面
可选操作流程	1a.选择时间超时，默认为不抢，询问下一个人 1b.每个人都选择抢地主，默认第一个人为地主 2b.出牌规则不符合要求，不允许出牌



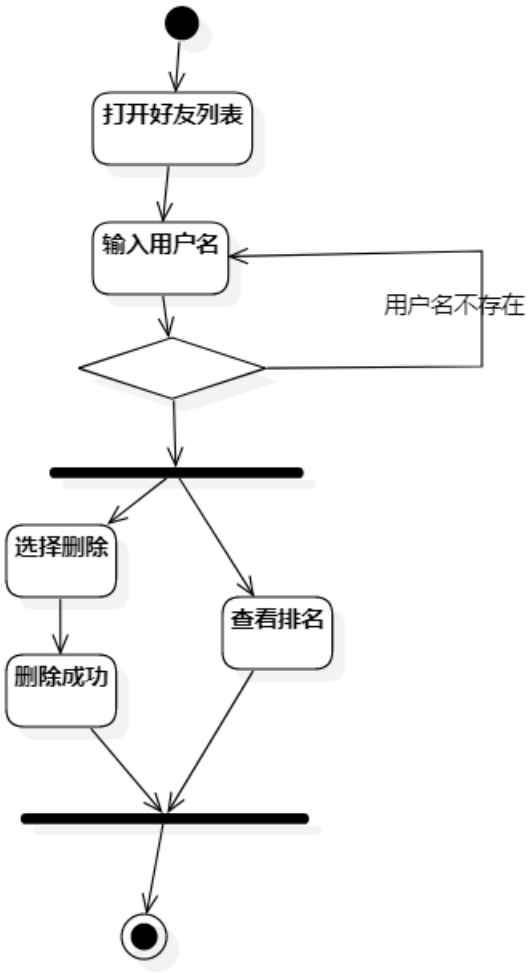
2.3.3 好友模块

用例编号	09
用例名称	好友删除
原理描述	用户希望对自己的好友列表进行管理，去删除某特定用户
主要角色	一般用户
被包含用例	无
被扩展用例	无
前提条件	系统处于工作状态 用户已登陆
主成功场景	1.打开好友列表 2.输入用户名 3.查找好友，选择删除 4.删除成功
可选操作流程	2a.输入的用户名不存在，提示重输 3a.输入的用户名不是好友，提示重输

用例编号	10
用例名称	好友添加
用例描述	用户希望对自己的好友列表进行管理，去添加某特定用户
主要角色	一般用户
被包含用例	无
被扩展用例	无
前提条件	系统处于工作状态 用户已登陆
主成功场景	1.打开好友列表 2.输入用户名 3.查找好友，选择添加 4.添加成功
可选操作流程	2a.输入用户名不存在，提示重输 3a.该用户已经是好友，提示重输

用例编号	11
用例名称	查看好友排名

用例描述	用户在拥有好友的情况下，想了解指定好友的成绩排名
主要角色	一般用户
被包含用例	无
被扩展用例	无
前提条件	系统处于工作状态 用户已登陆且有好友
主成功场景	1.打开好友列表 2.输入用户名 3.显示排名
可选操作流程	2a.用户名不存在，提示重输 3a.该好友未进行对战，显示无

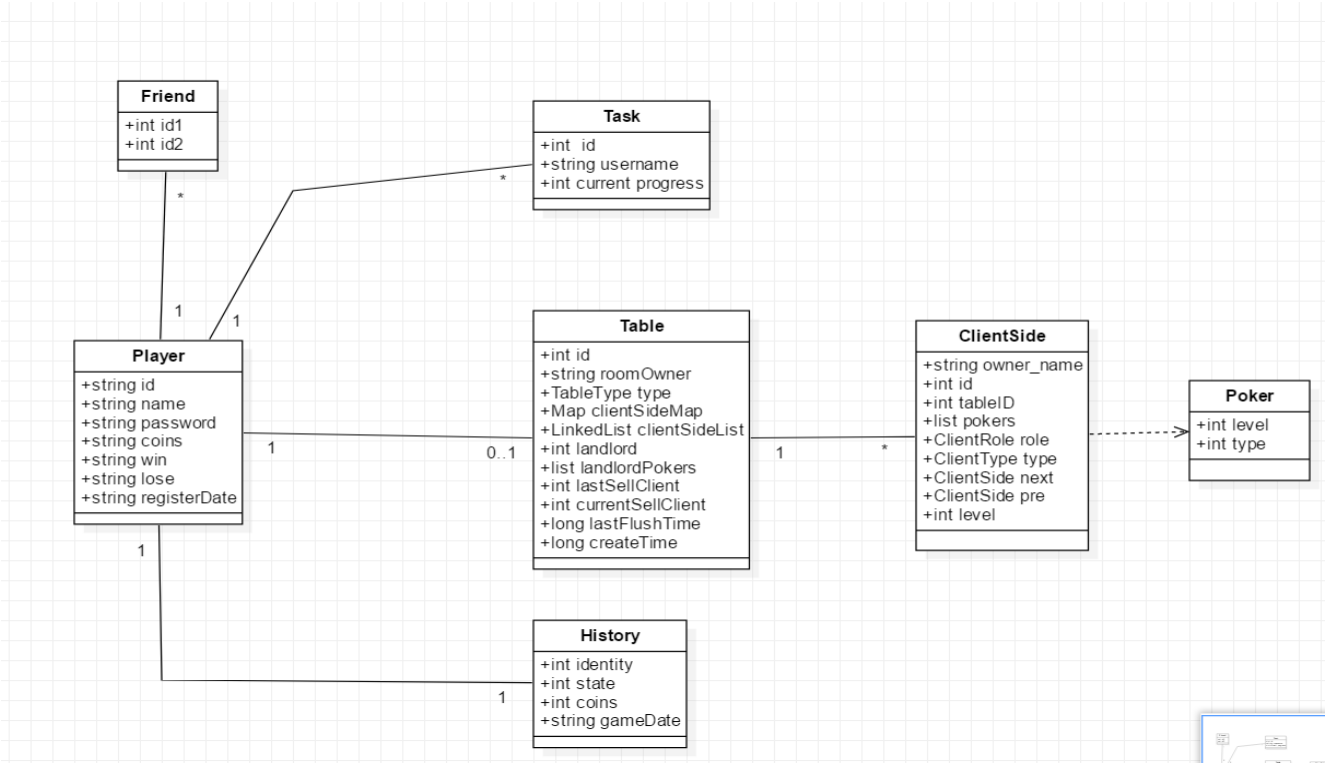


用例编号	12
用例名称	个人战绩查询
用例描述	用户希望对自己个人战绩进行查询
主要角色	一般用户
被包含用例	无
被扩展用例	无
前提条件	系统处于工作状态 用户已登陆

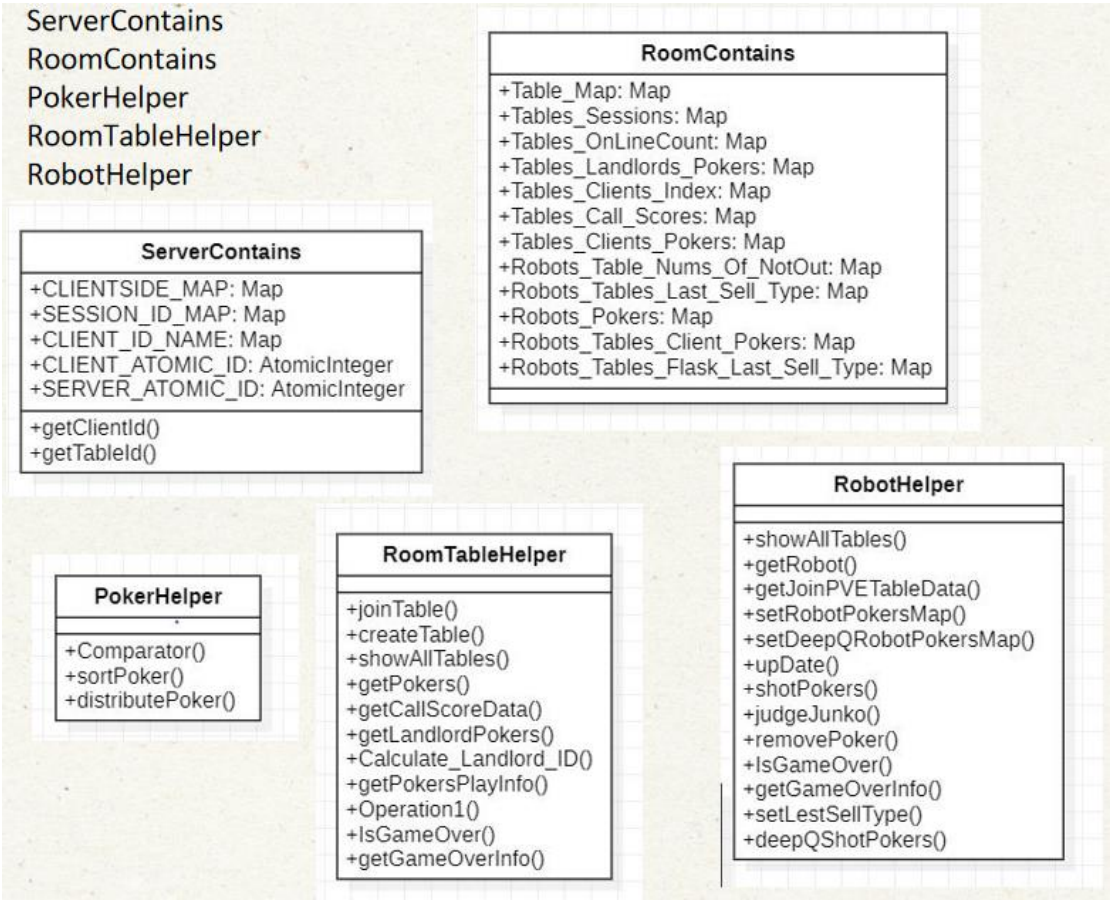
主成功场景	1.打开个人信息页 2.查看战绩
可选操作流程	无

2.4.2.数据建模

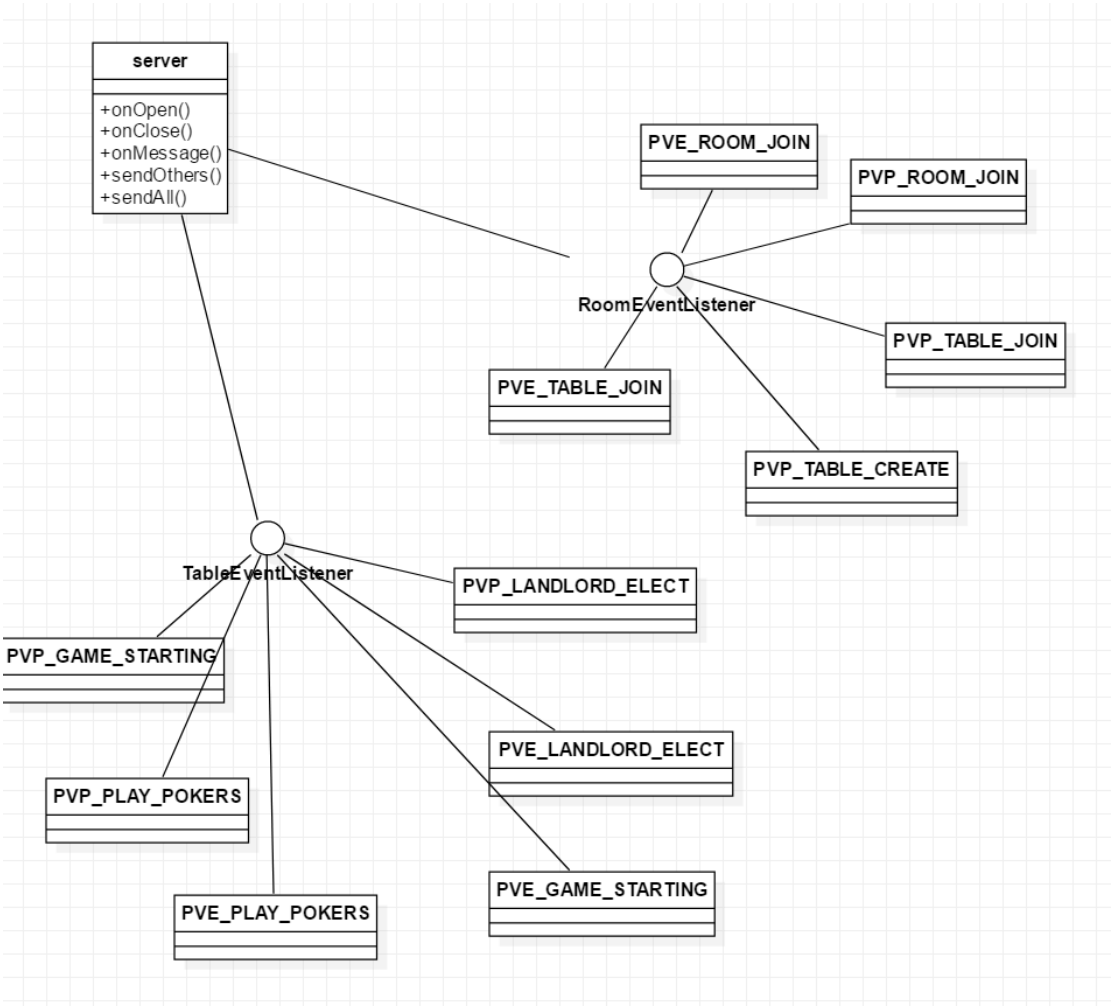
类图



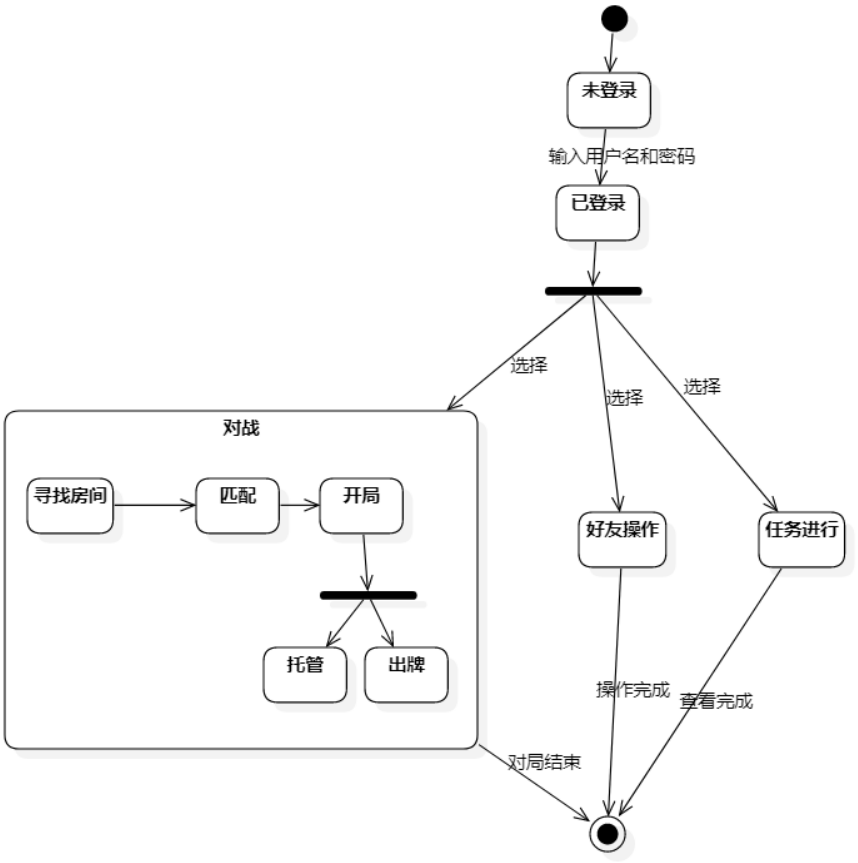
辅助类图


ServerContains
RoomContains
PokerHelper
RoomTableHelper
RobotHelper
ServerContains
+CLIENTSIDE_MAP: Map
+SESSION_ID_MAP: Map
+CLIENT_ID_NAME: Map
+CLIENT_ATOMIC_ID: AtomicInteger
+SERVER_ATOMIC_ID: AtomicInteger
+getClientId()
+getTableId()
RoomContains
+Table_Map: Map
+Tables_Sessions: Map
+Tables_OnLineCount: Map
+Tables_Landlords_Pokers: Map
+Tables_Clients_Index: Map
+Tables_Call_Scores: Map
+Tables_Clients_Pokers: Map
+Robots_Table_Nums_Of_NotOut: Map
+Robots_Tables_Last_Sell_Type: Map
+Robots_Pokers: Map
+Robots_Tables_Client_Pokers: Map
+Robots_Tables_Flask_Last_Sell_Type: Map
RobotHelper
+showAllTables()
+getRobot()
+getJoinPVETableData()
+setRobotPokersMap()
+setDeepQRobotPokersMap()
+upDate()
+shotPokers()
+judgeJunko()
+removePoker()
+IsGameOver()
+getGameOverInfo()
+setLestSellType()
+deepQShotPokers()
RoomTableHelper
+joinTable()
+createTable()
+showAllTables()
+getPokers()
+getCallScoreData()
+getLandlordPokers()
+Calculate_Landlord_ID()
+getPokersPlayInfo()
+Operation1()
+IsGameOver()
+getGameOverInfo()
PokerHelper
+Comparator()
+sortPoker()
+distributePoker()

接口设计


server
+onOpen()
+onClose()
+onMessage()
+sendOthers()
+sendAll()
PVE_ROOM_JOIN
PVP_ROOM_JOIN
RoomEventListener
PVP_TABLE_JOIN
PVE_TABLE_JOIN
PVP_TABLE_CREATE
TableEventListener
PVP_LANDLORD_ELECT
PVP_GAME_STARTING
PVE_LANDLORD_ELECT
PVP_PLAY_POKERS
PVE_GAME_STARTING
PVE_PLAY_POKERS

2.4.3. 行为建模（状态图）



3. 非功能需求

3.1. 性能要求

3.1.1. 精度

欢乐豆，经验值，胜场数等数据的记录保留整数

时间精确到秒

3.1.2. 时间特性要求

（1）响应时间

加载页面的响应时间应控制在1～2s用户对按钮的操作的响应时间应控制在小于1s网络状态良好的情况下，对信息的加载时间应小于1秒提交信息后，信息的更新时间应小于2秒

（2）更新处理时间

网络传播速度正常的情况下，数据库传输时间应小于1秒采用消息中间件，异步更新，不影响主进程 UI 响应。

3.1.3. 输人输出要求

注册时输入的用户名是长度小于10的string，密码是数字字母混合的长度小于10的string输出的格式均符合上述精度的要求

3.2. 数据管理能力要求

数据存储要求：本项目采用oracle数据库进行数据存储。Oracle在低档软硬件平台可以使用更少的资源支持更多的用户，而在高档平台上可支持成百上千个用户，满足了安全性要求，提供了角色分工的安全保密管理。且Oracle在数据库管理功能、完整性检查、安全性、一致性方面都有良好的表现。

3.3. 安全及保密性要求

软件必须严格按照设定的安全机制运行，并有效防止非授权用户进入本系统。软件必须提供对系统中各种码表的维护、补充操作。软件必须按照需求规定记录各种日志。软件对用户的所有错误操作或不合法操作进行检查，并给出提示信息。用户必须对系统中的材料成本信息进行维护。

3.4. 其他专门要求

1. 可维护性

可维护性是指在不影响系统其他部分的情况下修改现有系统功能中问题或缺陷的能力。 身为开发人员，我们在创建和设计系统架构时，为了提高系统的可维护性，必须考虑以下 几个方面的要素:低耦合、高内聚合系统文档记录。本系统将采用严格的软件工程的规范进行开发，并采用良 好的设计模式保证系统各模块之间的低耦合及模块之间的高内聚。 本系统的所有代码将会被详细注释，对于系统所有代码，我们会生成详尽的技术文档。 对于系统开发过程可能出现的报错，我们将以文档的方式详细罗列报错码及对应的报错信 息。

2. 可扩展性

系统建成后，应在现行系统上不需要做大的改动或不影响整个系统结构，就可以增加 功能模块,这就必须在系统设计时留有接又,使其具有可扩展性和维护性，这样就方便在 后期的维护过程中根据用户的需求添加相应的功能，同时也不会影响系统其他功能模块的正常运行。除此之外,还应当保证所有文档说明的详细、完整度，使其他接收者或新加入开发者能够快速地了解网页的情况，并且正确地在原有基础上进行更新和加强。

4. 运行环境规定

4.1. 设备

商用笔记本电脑或台式机

4.2. 支持软件

操作系统：WIN10

数据库：Mysql