MMORPG 设计文档

刘思尧 & 苏宇皓
BUPT

Contents

1. 项目介绍

1.1. MMORPG

MMORPG ，是英文 Massive （或 Massively ）Multiplayer Online Role-Playing Game 的缩写。至今尚未有 MMORPG 的正式中文译名，而在中国比较常见的译法则是“大型多人在线角色扮演游戏”，是网络游戏的一种。在所有角色扮演游戏中，玩家都要扮演一个虚构角色，并控制该角色的许多活动。

1.2. 运行机制

MMORPG 分为客户端和服务器两部分。我们使用浏览器作为客户端。玩家从客户端通过互联网连接，登陆服务器端后才能进行游戏。
玩家的资料保存在服务器端。游戏的过程，是玩家扮演的角色和其他玩家控制的角色在网络虚拟空间中实时互动。而非玩家扮演的角色（即 NPC）则往往是在游戏中提供特殊服务的人物，如销售虚拟物品，提供任务等。
我们提供服务器供玩家游戏，负责管理游戏中的虚拟世界，并不断为游戏做出更新，以便留住玩家以及吸引其他人来玩这款游戏。

2. 游戏需求

2.1. 游戏流程

每一个**用户**创建自己的**角色**。系统内置了**怪物**，角色可以**组队**进入**副本**攻击**怪物**获得各种奖励。角色还可以在地图上的**商店**购物，相互添加**好友**。**队伍**之间可以发起对战。

2.2. 名词解释

角色: 角色是玩家进行游戏的载体，玩家操控角色在游戏中行动。角色拥有一系列属性，包括昵称、生命值 (HP)、攻击力、防御力、经验值、金钱。
好友: 好友是两个角色之间的对等关系，互为好友的两个角色可以更快的找到对方。
仓库: 一个角色拥有多个仓库，仓库中可以存储角色的物品（包括装备和道具）。角色在攻打副本时不可以访问仓库。
背包: 一个角色有且只有一个背包，用于在攻打副本时携带所需物品。
道具: 道具是能够在一次攻打副本期间对玩家某项数值进行增强的物品。
装备: 装备分为剑、盾、铠甲三种，可以提升角色的部分属性值。角色不能装备一件以上的同种装备。
宠物: 角色可以饲养宠物 (从商店购买)，宠物会跟随在玩家身后，但只有装饰作用。
队伍: 一支队伍由一个或更多玩家组成，并有一个队长。队伍可以攻打副本。
副本: 副本是一张地图。副本中有怪物，角色可以在副本中攻击怪物并得到奖励。
主世界: 主世界是一张地图。角色可以在主世界访问商店、仓库或进入副本。
地图: 地图是一个二维矩阵，每一格被分配了地形。角色可以在地图中移动并触发事件。
金币: 游戏中的虚拟货币。

2.3. 游戏规则

2.3.1. 用户与角色

用户登录后，可以选择创建角色，如果已有角色，可以销毁角色或使用该角色进入游戏。
角色有三个装备槽——剑、盾、铠甲。每个槽只能装备一件装备。
角色可以申请与其他角色结为好友，若对方同意，则将对方加入双方的好友列表。

2.3.2. 主世界

所有在线玩家位于主地图或副本地图。角色进入游戏后，出生于主地图随机位置。
角色可以通过商店购买/出售可交易物品，可交易物品包括所有装备、道具、宠物。
角色可以通过副本入口点进入对应的副本。
道具可以在仓库与背包之间转移。
用户可以通过界面内菜单进行人民币充值，1 元 =100 金币。

2.3.3. 副本

怪物在且仅在副本中出现。角色尝试进入怪物所在格子时视为发动攻击，参照战斗规则。如果怪兽在战斗中死亡，会在自己周围 3*3 的方格内掉落物品，包括装备与道具。
战斗规则：

1. 发起战斗的一方先进行攻击，令攻击方为 A，防御方为 B。
2. 每次攻击，A 对 B 的 HP 造成 A 攻击 - B 防御点伤害。
3. 一次攻击结束后，攻击与防御方交换，进行下一轮攻击。
4. 有一方的 HP 小于等于 0 时，战斗结束。

角色可以在非战斗时刻使用道具。

3. 功能设计

下面是根据上文给出的需求分析得到的软件功能要求。

3.1. 用户登陆、注册与创建角色

3.1.1. 用户登录注册

登录注册页面前端部分内容，主要是用户账号和密码在 js 代码里做对应的正则匹配，这是验证的第一步，保证用户输入格式的正确性同时也从一方面减少用户向后台发送没必要的错误请求。前端向后端请求的方式使用 POST。在后台接收前端传送的信息同样要经过验证正则的规则匹配步骤，因为有可能一些人并不是通过合法的方式进行访问，可以减少没必要的数据库查询。

3.1.2. 用户密码安全性

由于很多人习惯使用相同的密码，为了降低撞库的风险，不应当使用明文存储密码。密码的保存使用“密钥 + 不可逆加密算”，使用如 sha1，sha256 等不可逆的加密算法进行加密后再存入数据库。

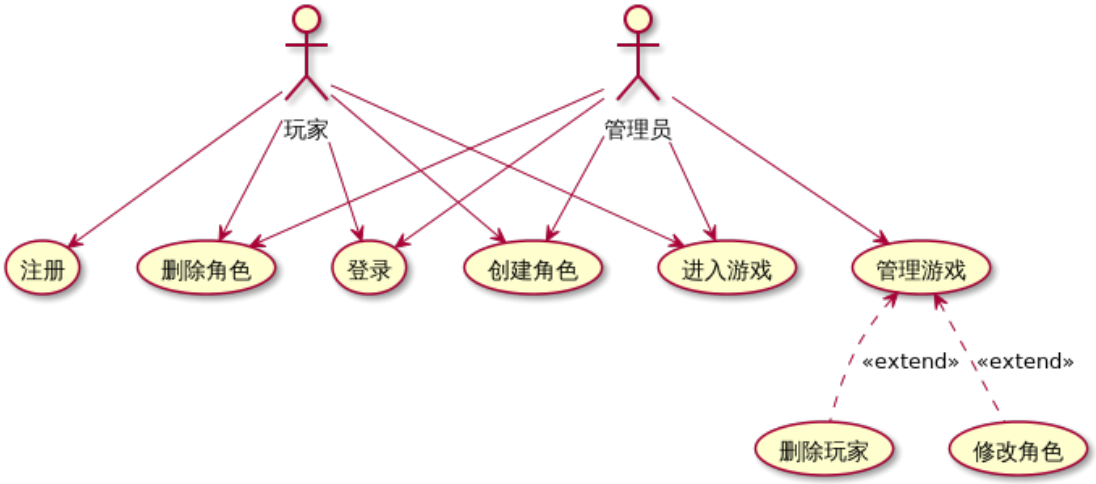

Figure 1. 用例图

3.1.3. 登录状态的保存

由于 http 协议是无状态的，所以用户的登录状态就要靠后台相应的数据来记录，用户登录成功后在 seesion 中保存登录用户，然后将该状态通过 cookie 返回到客户端，之后通过比对 cookie 和 seesion 信息来验证用户是否已登录。

3.1.4. 密码找回功能

因为后台密码是加密的，无法提供明文密码的找回。但是可以通过手机短信验证或者邮箱验证，验证成功后直接设置新密码。安全起见，向重置密码的 url 地址加入时间戳和唯一随机数来保证这个链接只在某个时间内有效，如果在密码修改成功或者事件过期，就把这个唯一随机数给删除，保存密码方式同上。

3.1.5. 角色的创建

一个用户只能创建一名角色，而且没有角色的用户无法进行游戏。因此可以让用户在创建账号后立即进行角色的创建。在前端让用户挑选自己喜欢的皮肤与特性等角色属性，将数据传给后端存入数据库。角色创建后直接进入游戏。一般不希望用户经常修改角色，因此将删除角色的功能放在进入游戏后的设置界面内。

3.2. 角色交互

3.2.1. 导航与事件触发

秉承简单的原则，基本只使用上下左右四个按钮。玩家通过上下左右在地图上移动，如果玩家尝试进入某个特殊的格子（例如怪兽所在的格子、商店所在的格子），就触发事件。可以触发的事件包括：

- 与怪兽战斗
- 打开商店界面
- 打开仓库界面

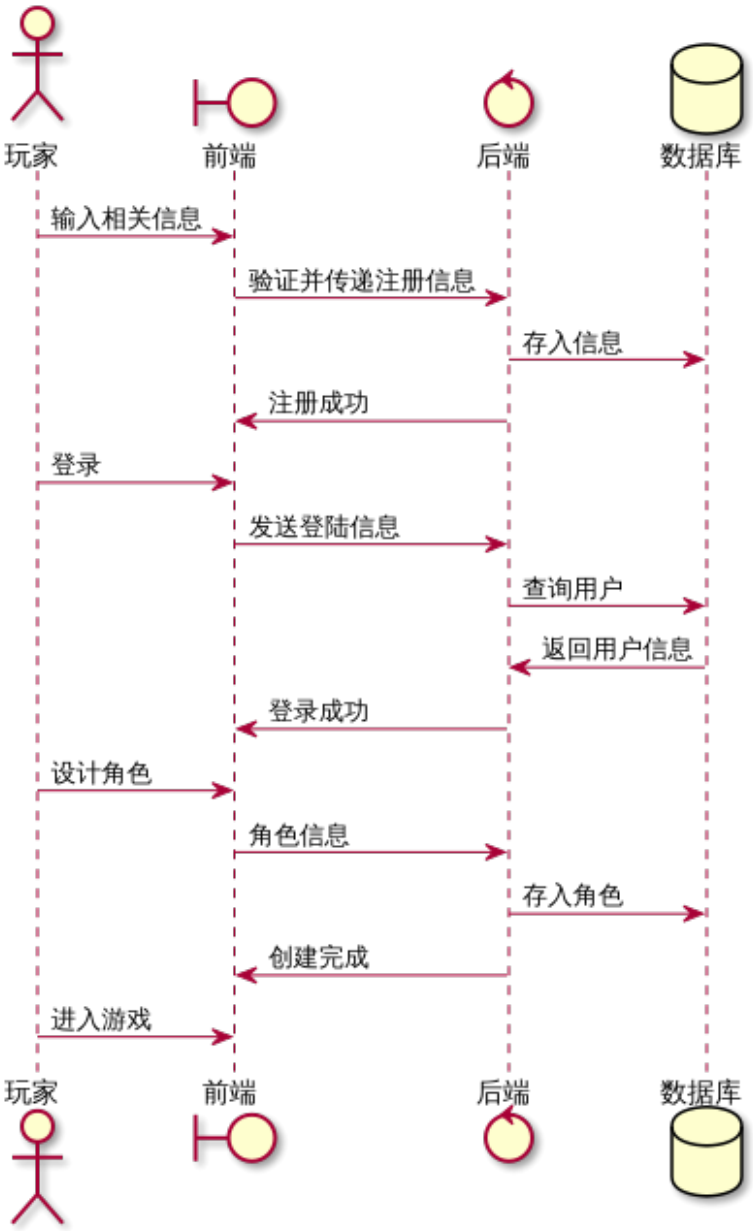

Figure 2. 主流程时序图

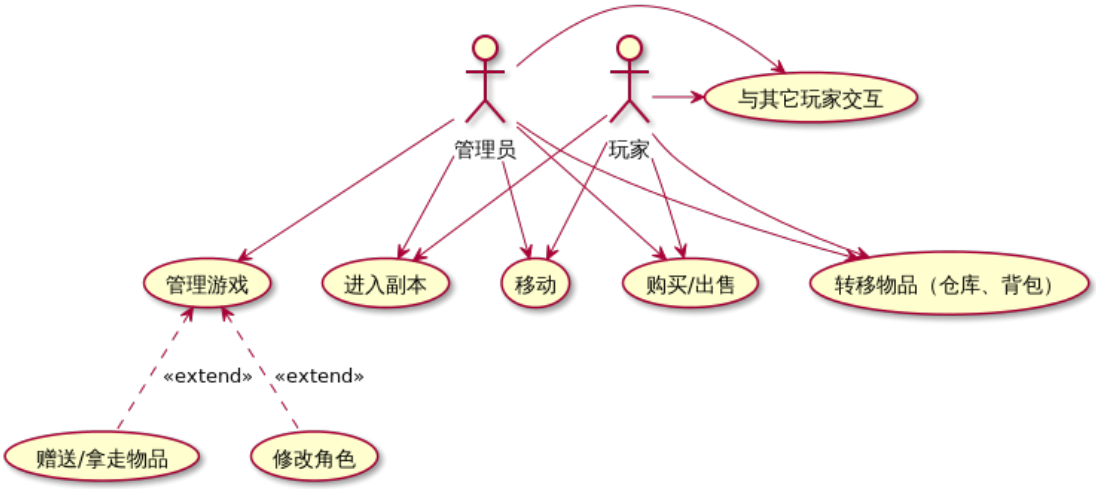

Figure 3. 用例图

- 进入副本
- 拾取物品
- 离开副本

3.2.2. 其他操作

出去导航时出发的时间，角色还能进行许多其他操作，通过前端 UI 引导，使用鼠标完成。这些操作主要与其他玩家交互、道具与装备的使用与转移、在商店购买有关。这些操作会在后面详细说明。

3.3. 购买与充值

3.3.1. 商店购物

游戏中的虚拟货币称为金币。玩家通过导航进入商店后，会弹出商店菜单。游戏设定为在商店中可以无限量购买或出售所有的物品，包括宠物、装备与道具。因此不必存储商店中的库存余量。前端需要按照分类展示所有的物品，玩家根据需要将要购买的物品加入购物清单，清单在加入第一个物品时自动生成。清单显示在一个单独的窗口中，并实时计算当前所有物品的金币数。在玩家选择完毕后，可以进行结算，从玩家余额中扣除相应的金币，并向玩家对应的仓库中添加所购买的物品，这是前端与后端进行数据传递的一步。

3.3.2. 人民币充值

玩家获取金币有两种途径，一是攻打副本时怪物掉落以及副本奖励，一是通过人民币充值。由于涉及到金钱交易，处理需要十分谨慎，在后端采用数据库事务模式。与各种支付平台对接，使得用户可以通过多种方式支付。

3.4. 道具与装备

设计道具与装备是为了丰富游戏内容，增强玩家的体验。

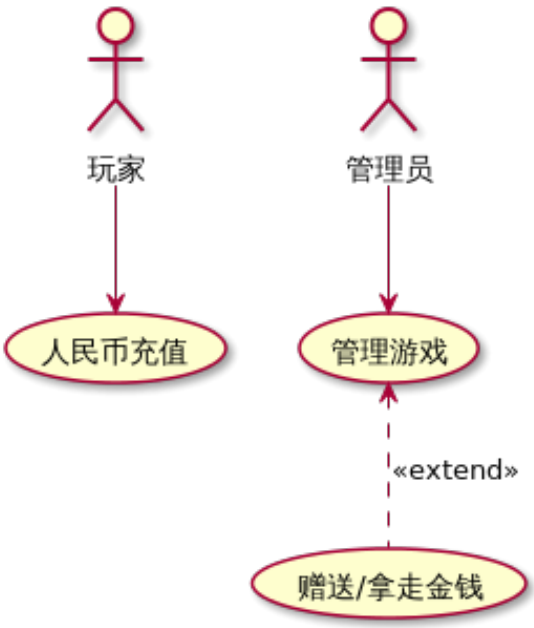

Figure 4. 用例图

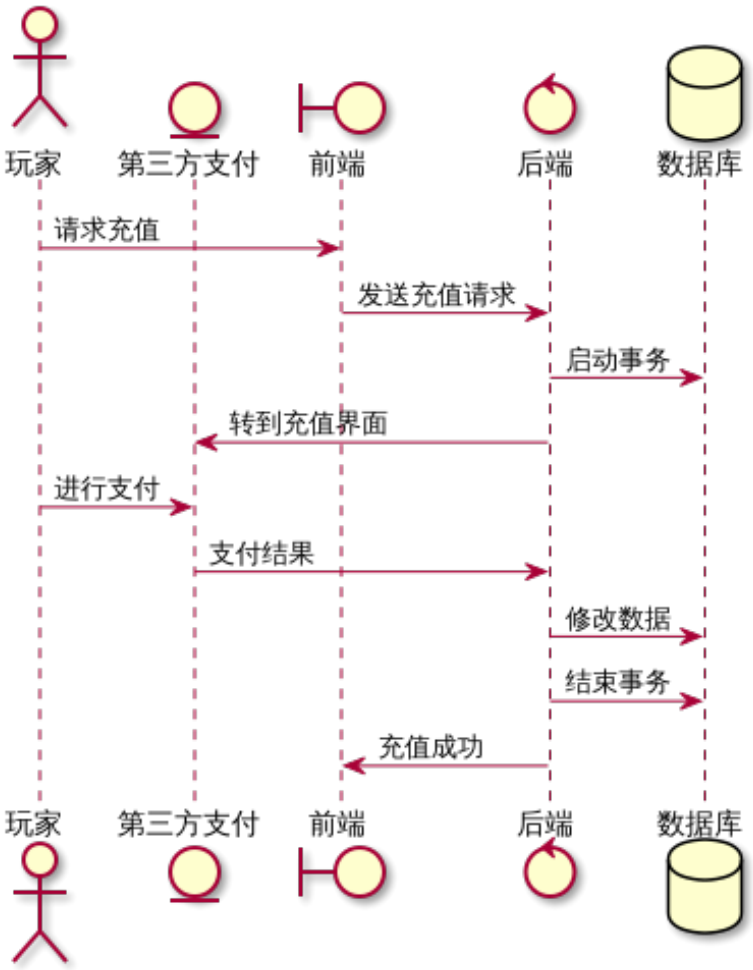

Figure 5. 充值时序图

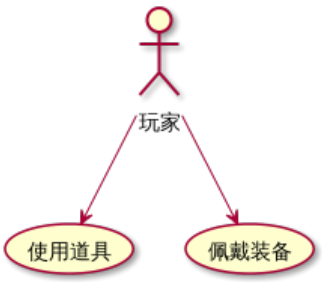

Figure 6. 用例图

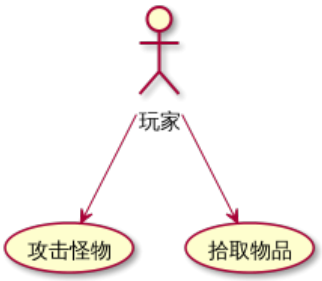

Figure 7. 用例图

道具用于在副本中使用，增加玩家的生命值或攻击力。玩家有两个容器可以存储道具：仓库与背包。玩家的所有道具都可以存储在仓库中，但是在副本中无法访问仓库，要将道具放入背包（可容纳的数量有限）带入副本才可以使用。因此要设计两个展示界面：仓库与背包，仓库在主界面通过导航事件访问，背包可以在游戏界面下方的状态栏打开。玩家在仓库界面时可以浏览自己拥有的所有道具，也可以在仓库和背包之间转移物品。在攻打副本时，可以打开背包并使用其中的道具。同时，在攻打副本过程中获得的战利品也会被放入背包。
装备可以提升玩家的属性，分为剑、盾、铠甲三类。玩家同一时间只能装备一把剑、一面盾和一身铠甲。装备只能放置在仓库中，如果在副本中获得了装备会被传送到仓库。玩家可以在浏览仓库时选择某件装备并穿上。
仓库分为装备仓库和道具仓库两种，玩家可以自行创建新的仓库（为了方便对物品进行分类）。

3.5. 副本与奖励

玩家可以攻打副本并获得奖励，这也是用户玩游戏的主要动机。怪兽只出现在副本中，玩家可以在主世界内与其它玩家组队并攻打副本，通过杀死副本内的怪兽来得到奖励。按照设定，副本指某一个特定的含有怪物的地图。副本对每一个队伍是独立的，也就是说一支队伍攻打副本并不会对队伍以外的玩家造成任何影响。副本和主世界一样，都是一张地图，因此采用与主世界相同的显示模块，但是增加了触发战斗的导航事件。
队伍进入副本所在位置便可以攻打副本，这时从数据库查询对应副本的地图并返回给前端，前端通过后端同步其它在线玩家的位置信息，并在拾取战利品时修改数据库。

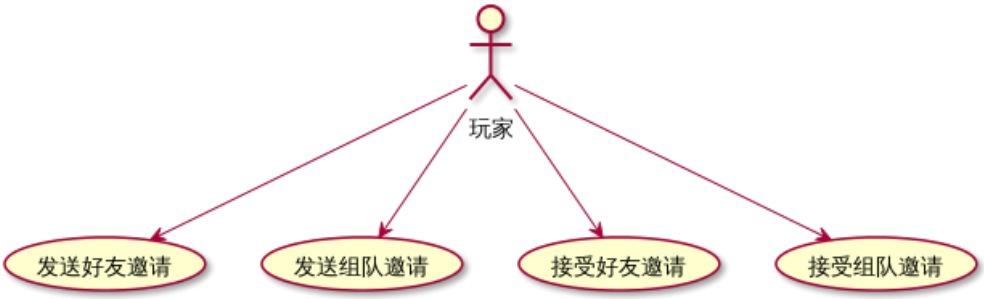

Figure 8. 用例图

3.6. 多人联机

进行多人联机需要获取所有其他玩家的状态（位置、朝向、动作等），这样的同步通过在后端记录在线玩家的状态完成。这些状态并不需要被存入数据库，因为没有持久化的必要。前端每隔一定时间就与后端进行同步，来更新所有玩家的状态。
除了在地图上显示所有在线玩家，还有与其他玩家互动的需求。为此，设置一个消息箱，玩家 A 希望与玩家 B 组队或结为好友时，可以发送消息，玩家 B 看到消息后即可做出应答。

4. 数据库设计

4.1. 实体-关系模型

4.1.1. ER 模型说明

用户-角色：一个用户能且仅能创建一个角色，用户用作登陆和账号管理，角色再游戏中使用，他们是一一对应、双方全参与关系。
角色-充值清单：一个角色可以没有账单或多个账单，但每一个账单必须要有一个角色与之对应，角色-账单为一对多关系，账单全参与。
角色-队伍：一个角色属于且仅属于一支队伍，一个队伍至少有一个角色，因此角色、队伍均为全参与，角色-队伍为多对一。
队伍-PVE 副本：一个队伍可以攻打多个副本，一个副本可以由多个队伍攻打，均为部分参与，队伍-副本为一对多。
PVE 副本-怪物等级：一个副本中有 1 个到多个怪物，每个怪物可以在 0 个或多个副本中，副本全参与，多对多关系。
怪物等级-怪物：怪物等级为弱实体集合，表示不同等级的同种怪物，怪物 id 和其等级共同缺点一只怪物。双方全参与，为多对一关系。
角色-背包：一个角色有且仅有一个背包，一个背包也只由一个人拥有，双方全参与，为一对一关系。
背包-道具：一个背包可以装配有 0 个或多个道具，一种道具可以在 0 个或多个背包中。都是部分参与，为多对多关系。
角色-仓库：一个角色拥有若干仓库（仓库分为装备仓库和道具仓库），一个仓库只能由一个角色拥有，双方全参与，为一对一多关系。

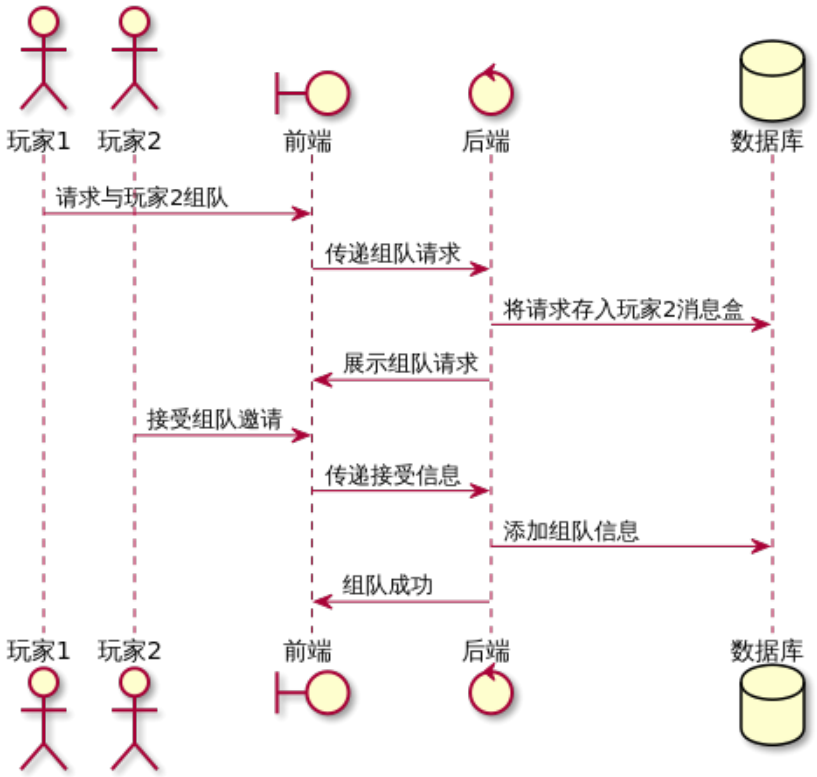

Figure 9. 组队时序图

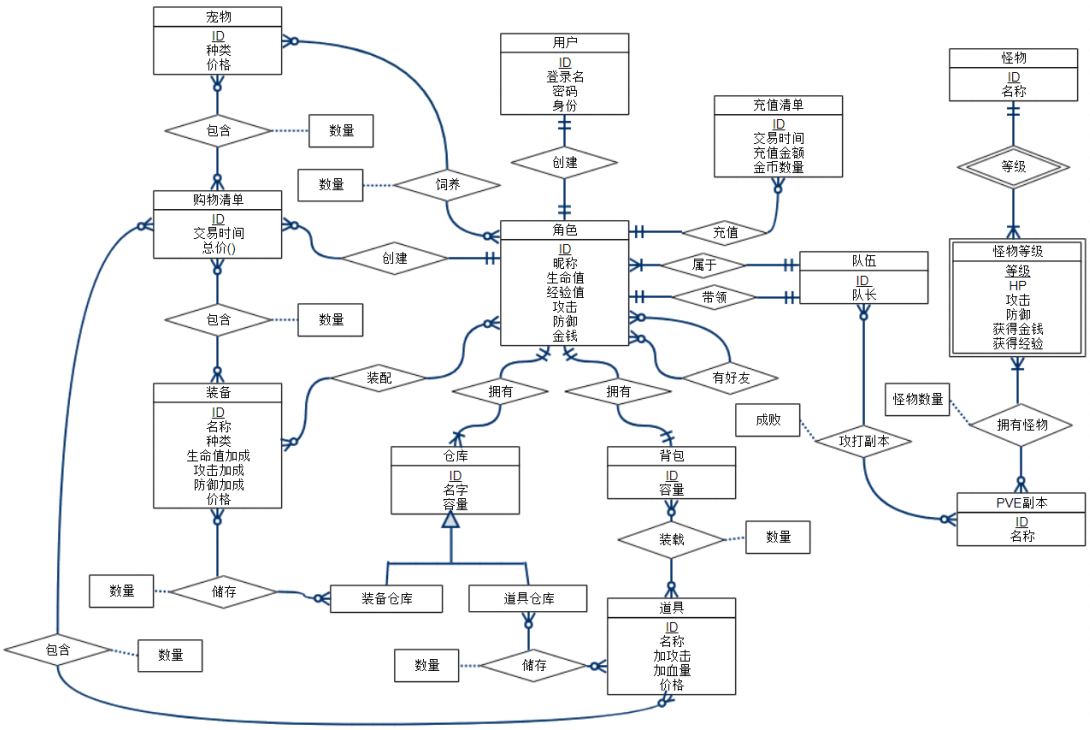

Figure 10. E-R 图

道具仓库-道具：一个道具仓库可以储存有 0 个或多个道具，一种道具可以在 0 个或多个道具仓库中。都是部分参与，为多对多关系。
装备仓库-装备：一个装备仓库可以储存有 0 个或多个装备，一种装备可以在 0 个或多个装备仓库中。都是部分参与，为多对多关系。
角色-装备：一个角色可以装配 0 个或多个装备，一种装备可以在 0 个或多个装备仓库中。都是部分参与，为多对多关系。
角色-购物清单：一个角色可以有 0 个或多个购物清单，一个购物清单可以只能一个角色拥有，一个购物清单必须对应一个角色。角色部分参与，购物清单全参与，为一对多关系。
角色-宠物：一个角色可以饲养若干个宠物，每种宠物可以有若干个饲养者。双方都是部分参与，为多对多关系。
购物清单-宠物/装备/道具：一个宠物清单中可以有多个各种宠物/装备/道具，一种宠物/装备/道具可以在多个清单中。

4.2. 数据库表结构

用户:

```
User(user_id, name, password, identity, phone_num,
     email, verify_code, expire_time);`
```

由于**用户**和**角色**有一一对应关系，用户为主表，在角色中创建外键 user_id。因为一一对应，外键为 **unique**；因为全参与，外键为 **not null**。
一个角色至少在一个**队伍**里，一个队伍可以有多个人，在角色中创建外键 team_id，外键为 **unique**；因为全参与，外键为 **not null**。
角色:

```
Role(role_id, nickname, hp, exp, atk, def,
     money, user_id, team_id);
```

仓库和角色为多对一，在仓库中创建外键 role_id。因为全参与，外键为 **not null**:

```
EquipRepository(rep_id, name, volume, role_id);
ItemRepository(rep_id, name, volume, role_id);
```

背包和角色为多对一，在背包中创建外键 role_id。因为全参与，外键为 **not null**:

```
Backpack(backpack_id, name, volume, role_id);
```

充值清单和角色为多对一，在账单中创建外键 role_id。因为全参与，外键为 **not null**:

```
Bill(bill_id, time, deposit, money, role_id);
```

一个**队伍**有一个队长，建立外键 leader。因为一一对应，外键为 **unique**；因为全参与，外键为 **not null**:

```
Team(team_id, leader);
```

副本: `Dungeons(dungeons_id, name);`

副本和**队伍**是多对多关系，因此创建一张新表，主键为 team_id 和 role_id:

`Fights(team_id, dungeons_id, result);`

怪物: `Monster(monster_id,name);`

怪物的等级是弱实体，需要怪物加上其等级才能作为一个实体，因此 monster_id 和 level 共同作为主键:

`MonsterLevel(monster_id, level, hp, atk, def, money, exp);`

副本和怪物实体是多对多关系，因此创建一张新表:

`HaveMonster(dungeon_id,monster_id,max_num);`

道具: `Item(item_id, name, kind, hp, atk, def, price);`

装备: `Equipment(equip_id, name, hp, atk, price);`

装备和仓库是多对多关系，因此创建一张新表:

`StoreEquipment(rep_id, eqiup_id, number);`

道具和仓库是多对多关系，因此创建一张新表:

`StoreItem(rep_id, item_id, number);`

背包和仓库是多对多关系，因此创建一张新表:

`BackpackItem(backpack_id, item_id, number);`

装备和角色是多对多关系，因此创建一张新表:

`DressEquip(role_id, equip_id, number);`

宠物: `Pet(pet_id, kind, price);`

宠物和角色是多对多关系，因此创建一张新表:

`AdoptPet(role_id, pet_id, number);`

一个用户有多个**购买清单表**，在清单中创建外键 role_id。因为购物清单全参与，外键为 **not null**:

`ShopList(shopping_id, time, total_price, role_id);`

宠物和购买清单为多对多关系，创建一张新表:

`ShopPet(pet_id, shopping_id, number);`

装备和购买清单为多对多关系，创建一张新表:

`ShopEquip(equip_id, shopping_id, number);`

道具和购买清单为多对多关系，创建一张新表:
`ShopItem(item_id, shopping_id, number);`

每个角色有若干**朋友**，为多对多关系，创建一张新表:
`Friend(role_id_1, role_id_2);`

4.3. 数据库事务

部分敏感的操作（如充值、购买）应保证其 ACID 性质，避免造成错误和损失。

4.3.1. 充值

```
START TRANSACTION;
INSERT INTO BILL VALUES(...)                                    # 插入充值账单
第三方扣款
UPDATE User SET Money = Money + 充值金额 WHERE user_id =... # 增加账户余额
COMMIT;
```

4.3.2. 购买装备/道具/宠物

```
START TRANSACTION;
INSERT INTO StoreEquipment/StoreItem/AdoptPet VALUES(...); # 插入购买的物品
INSERT INTO ShopList VALUES(...);                           # 插入购买账单
INSERT INTO ShopEquip/ShopItem/ShopPet VALUES(...);         # 插入账单表项
UPDATE Role SET money = ... WHERE role_id = ...   ;         # 更新用户金钱
COMMIT;
```

4.3.3. 迁移道具/装备

```
START TRANSACTION;
从来源位置删除一个物品（例如 StoreEquipment）;
插入这个物品到目的位置（例如 DressEquip）     ;
COMMIT;
```

5. 功能模块设计

5.1. 登陆模块

玩家登陆，创建和删除角色。
主要函数：

- `signin(user_id, password)` 在数据库中验证参数是否对应
- `signup(user_id, password)` 在数据库中插入新的用户
- `signout(user_id, password)` 在后端删除相应的类和 session

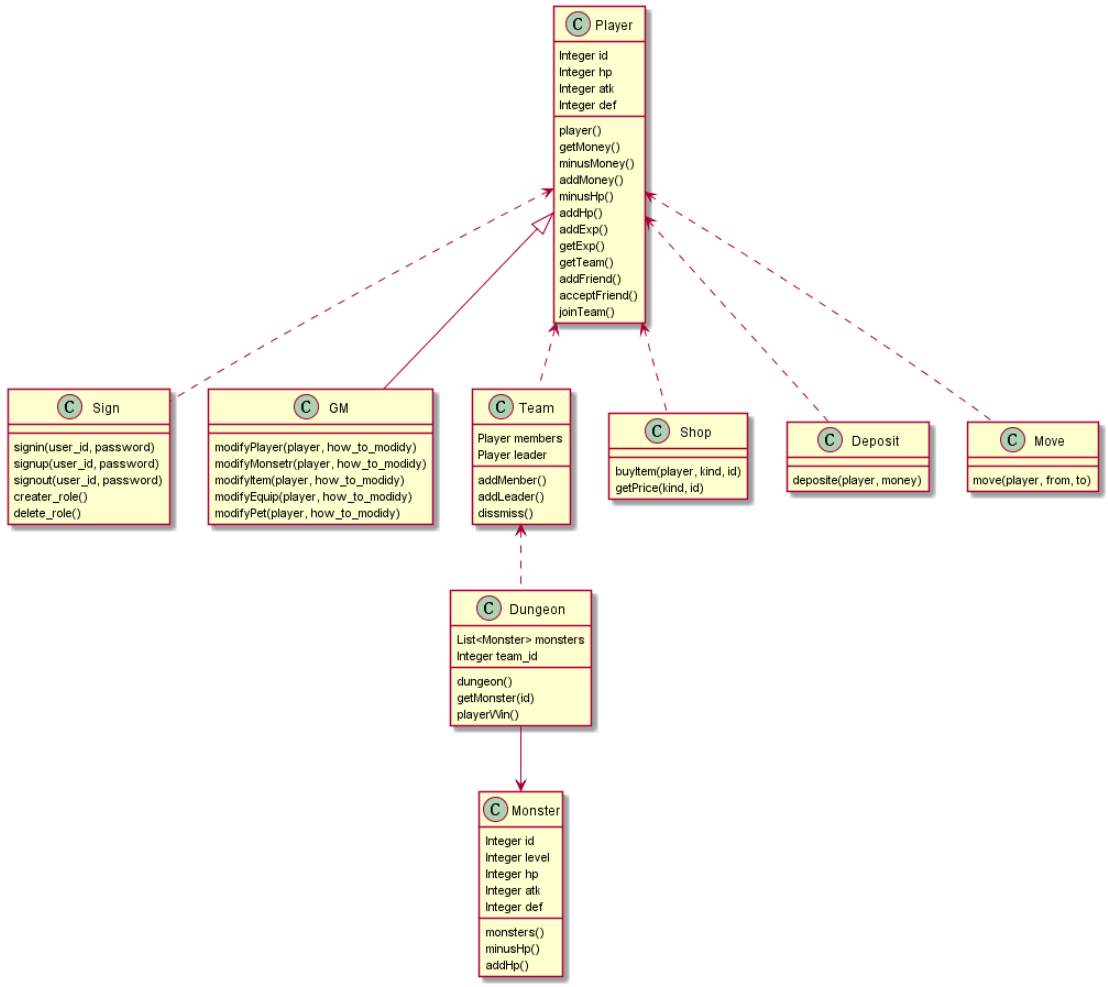

Figure 11. 模块类图

- `creater_role()` 从数据库中创建角色。
- `delete_role()` 从数据库中删除角色。

5.2. 玩家模块

实现登陆的玩家管理，在攻打副本，购买等操作。
主要函数：

- `player()` 构造函数，初始化登陆的玩家，从数据库中读出玩家基本信息。
- `getMoney()` 从数据库中读出玩家的金钱。
- `minusMoney()` 从数据库中减去一定数量的金钱。
- `addMoney()` 在数据库中增加一定数量的金钱。
- `minusHp()` 减少生命值。
- `addHp()` 增加生命值。
- `addExp()` 在数据库中增加一定数量的经验。
- `getExp()` 从数据库中查询玩家的经验。
- `getTeam()` 从数据库读取玩家在哪个小队。
- `addFriend()` 发出一个好友请求。
- `acceptFriend()` 接受好友请求，写入数据库。
- `joinTeam()` 加入一个小队，写记录到数据库。

主要成员变量：

- `id` 玩家 id。
- `hp` 玩家当前生命值。
- `atk` 玩家当前攻击力。
- `def` 玩家当前防御力。

5.3. 管理员模块

管理员继承了玩家的所有属性。可以参加游戏，可以管理游戏。
主要函数：

- `modifyPlayer(player, how_to_modidy)` 修改玩家相关的数据。
- `modifyMonsetr(player, how_to_modidy)` 修改怪物相关的数据。
- `modifyItem(player, how_to_modidy)` 修改道具相关的数据。
- `modifyEquip(player, how_to_modidy)` 修改道具相关的数据。
- `modifyPet(player, how_to_modidy)` 修改宠物相关的数据。

5.4. 队伍模块

管理组队
主要函数：

- `addMenber()` 添加成员记录到数据库和成员变量。
- `addLeader()` 添加队长记录到数据库和成员变量。
- `dissmiss()` 队长解散组队。

主要成员变量：

- members 队员
- leader 队长

5.5. 购物模块

用于玩家购买道具/宠物/装备。
主要函数：

- buyItem(player, kind, id) 购买道具/宠物/装备，执行购买事务。
- getPrice(kind, id) 获取道具/宠物/装备的价格。

5.6. 怪物模块

用于实现游戏中攻打副本时的怪物管理。
主要函数：

- monsters() 构造函数，初始化副本产生的怪物，从数据库中读出怪物基本信息。
- minusHp() 减少生命值。
- addHp() 增加生命值。

主要成员变量：

- id 怪物 id。
- level 怪物等级。
- hp 怪物当前生命值。
- atk 怪物当前攻击力。
- def 怪物当前防御力。

5.7. 副本模块

副本模块管理一次队伍攻打副本的全部内容，包括怪物和队伍。在攻打副本时初始化成员变量。
主要函数：

- dungeon() 初始化所有的怪物和他的位置，分配 id；生成对战记录写入数据库。
- getMonster(id) 获取某一个怪物。
- playerWin() 玩家胜利，写记录

主要成员变量：

- monsters 副本中的怪物。
- team_id 对战的队伍。

5.8. 充值模块

此模块本因有第三方接口，在作业中简化。
主要函数：
deposite(player, money) 执行充值事务。

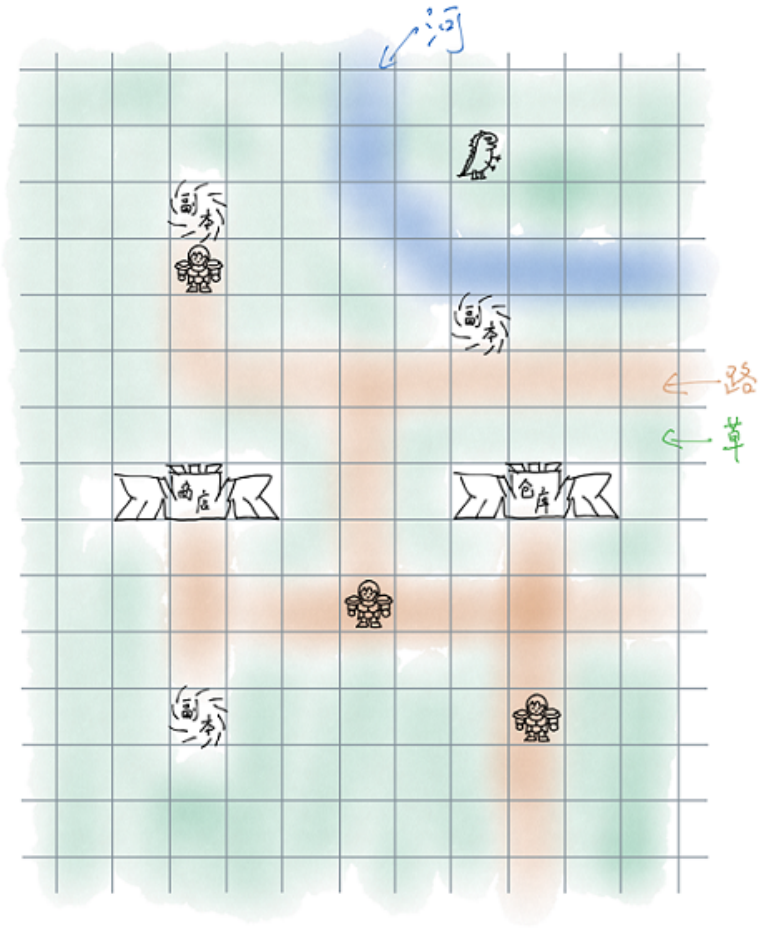

Figure 12. 用户界面草图

5.9. 物品迁移模块

用于玩家的物品（道具，装备）在仓库、背包等容器中相互迁移。
主要函数：
`move(player, from, to)` 执行迁移事务。

5.10. 视图模块

视图模块负责与用户界面及交互有关的操作。

- `navigate(direction)` 向 `dir` 方向移动角色一格
- `open_menu(menu_name)` 打开某个菜单

主要成员变量：

- `stat` 视图当前所处状态（主世界、副本、菜单等）
- `position` 当前角色所在位置

6. 技术和环境分析

6.1. 开发技术

- 前端
  - HTML
  - CSS
  - JS (Vue.js)
- 后端
  - Web 服务器：uWSGI
  - Web 应用：Django 2.1.4

6.2. 开发环境

Windows 10

6.3. 生产环境

- 阿里云服务器 CentOS 7.4
- CPU ：1 核 Xeon E5
- 硬盘：40G SSD
- 内存：2G