

软件工程设计文档

项目名称：模拟商业竞拍交易大赛系统

组名：proj9_rua

组长：寇明阳 2015011318

组员：姚超 2015011319

黄权伟 2015011328

李忠明 2015011326

目录

软件工程设计文档	1
1 概述	3
1.1 系统概述.....	3
1.2 文档概述.....	3
2 整体框架	4
2.1 系统框架图.....	4
2.2 整体设计思路.....	4
3 系统体系结构设计	5
3.1 运行环境.....	5
3.2 服务器与后端.....	5
3.2.1 服务器端功能简介	5
3.2.2 服务器端与数据库、web 端、微信小程序端交互流程	5
3.2.3 类和方法功能描述	9
3.2.4 单元测试结果	14
3.3 数据库设计.....	15
3.4 Web 端（管理员端）	18
3.4.1 Web 端功能简介	18
3.4.2 Web 端设计总览	19
3.4.3 Web 端设计详细介绍	19
3.5 微信小程序（用户端）	25
3.5.1 用户端功能简介	25
3.5.2 用户端设计总览	25
3.5.3 用户端设计详细介绍	26
4 接口设计	32
4.1 Web 端与后端	32
4.2 微信小程序端与后端.....	36
5 服务器配置	42
5.1 部署方式.....	42
5.2 基本架构.....	42
5.3 详细部署流程.....	42
5.3.1 django 后端部署	42
5.3.2 Nginx 配置	43
5.3.3 管理员前端部署	44
6 小组人员	45

1 概述

1.1 系统概述

ASDAN，是一个非盈利性教育组织，得到英国教育部课程局（QCA）和英国高等院校招生服务中心（UCAS）授权，为全球学生提供素质教育认证和领导力课程开发。

ASDAN 举办的模拟商业竞赛是由学生组成的多支团队在线下展开的商业竞赛，完全仿真还原市场和公司运营的真实环境与决策过程，包含运营、竞拍交易和路演三大部分，赢取最多利润的公司成为冠军。其中的 竞拍交易 部分需要一个独立的在线系统，供多支参赛团队实时参与竞赛，通过 竞拍、生产、交易流程获取最多利润，打败竞争对手赢得比赛。

模拟商业竞拍交易大赛系统是用来模拟 ASDAN 举办的模拟商业竞赛的竞拍交易部分的系统。管理员可以通过系统的 web 端进行创建一场比赛，查看比赛信息和队伍信息，发起拍卖，设置倒计时，查看排名，更改比赛阶段等操作。用户可以通过微信小程序端进行登录，查看队伍信息，查看拍卖信息，生产交易，查看排名等操作。

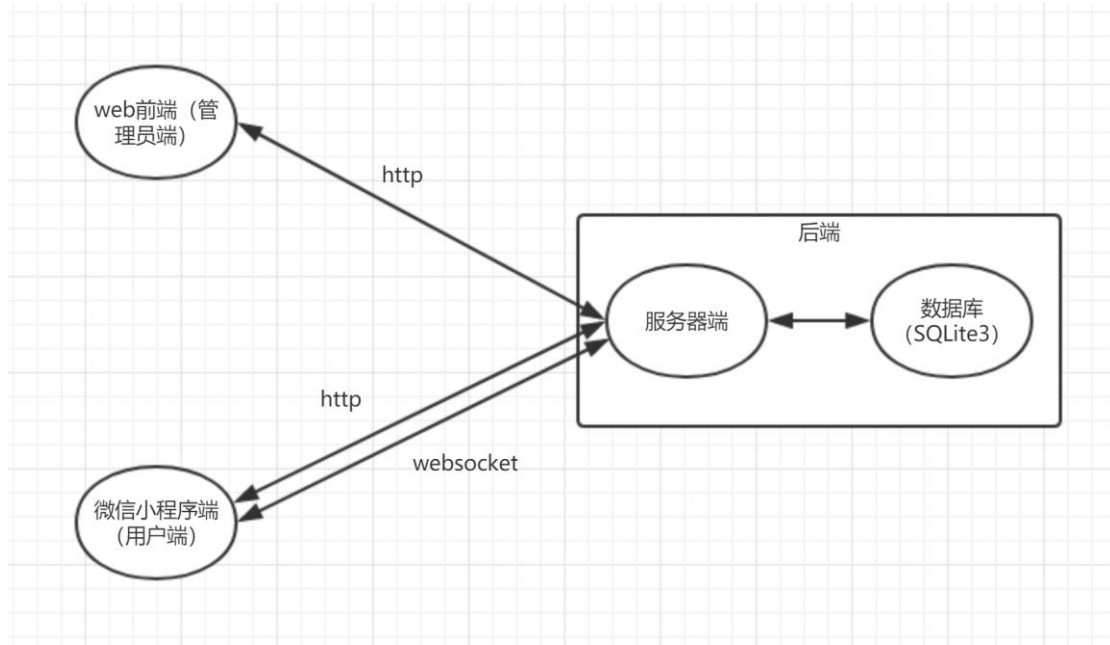
1.2 文档概述

本文档将整个系统分为后端、数据库、web 端、微信小程序端四个模块。先介绍整体框架，再依次介绍各个模块的内部详细实现方法，以及各个模块之间的关系和传递数据的接口。

2 整体框架

2.1 系统框架图

系统的各个模块的交互关系如下图所示：



2.2 整体设计思路

整个系统分为 web 端、微信小程序端、数据库、服务器端四个模块，数据库与服务器端共同组成后端。采用前后端分离的方式，web 端、微信小程序端、后端之间都是解耦合关系，不依赖于其他模块，可以单独存在。数据库与服务器端作为一个整体，与 web 端、微信小程序端在 gitlab 上分为三个独立的项目，互不影响。Web 端、微信小程序端分别和服务端交互，交互有两种方式，分别是基于 http 协议的方式和收发 websocket 的方式。数据的传输都是 JSON 格式。

3 系统体系结构设计

3.1 运行环境

- 网页端（管理员端）使用 chrome、firefox、360 等主流浏览器访问。
- 微信小程序开发者可以通过微信开发者工具编辑及使用，其他游客可通过管理员生成的二维码进行使用。
- 后端部署在腾讯云服务器上，也可在本机上使用 python2 或 python3 运行服务器（支持 Windows、Linux、MacOS 系统）

3.2 服务器端

3.2.1 服务器端功能简介

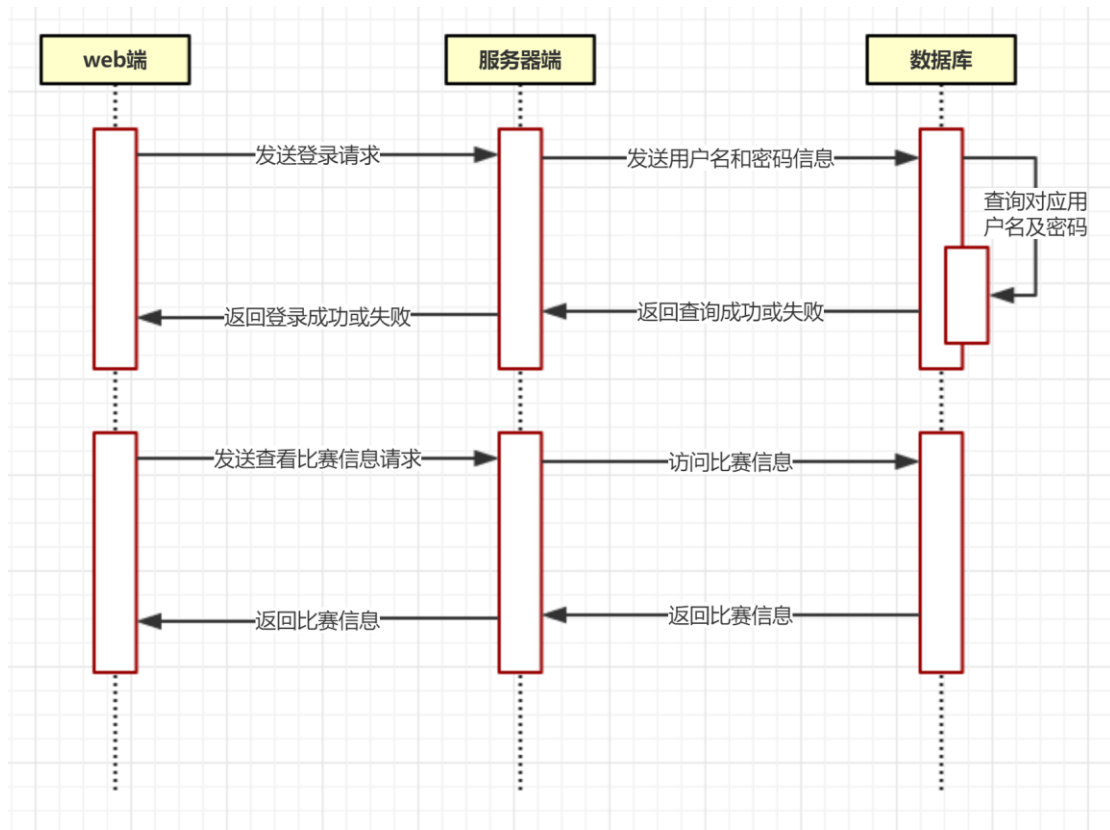
服务器端的主要功能是响应来自 web 端管理员发来的请求（包括创建一场比赛，查看比赛信息和队伍信息，发起拍卖，设置倒计时，查看排名，更改比赛阶段等），微信小程序端发来的请求（登录，查看队伍信息，查看拍卖信息，生产交易，查看排名等）并更新数据库内的相关数据。

3.2.2 服务器端与数据库、web 端、微信小程序端交互流程

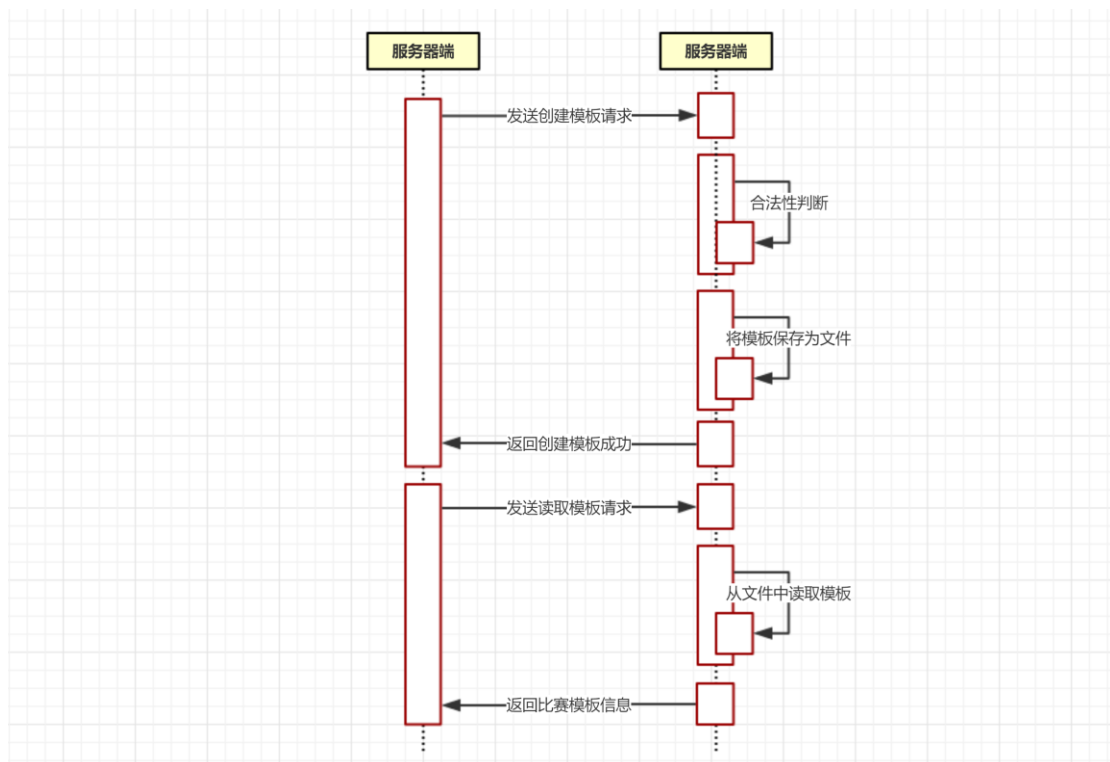
服务器端与数据库、web 端、微信小程序端交互的主要流程分为七个阶段：

- （1）初始化阶段（管理员登录、查看比赛信息）
 - （2）管理员创建或读取模板阶段（可选）
 - （3）比赛初始化阶段（管理员创建比赛、用户登录、管理员读取准备信息、管理员开始比赛）
 - （4）拍卖阶段（管理员记录拍卖结果、用户查看拍卖结果）
 - （5）生产交易阶段（用户发起生产请求、发起交易请求、响应交易请求、管理员查看交易订单）
 - （6）结算阶段（用户及管理员查看排名信息）
 - （7）用户及管理员查看各队伍详细资产，在比赛的任意阶段都可以发起这一请求
- 各阶段主要流程的 UML 序列图如下：

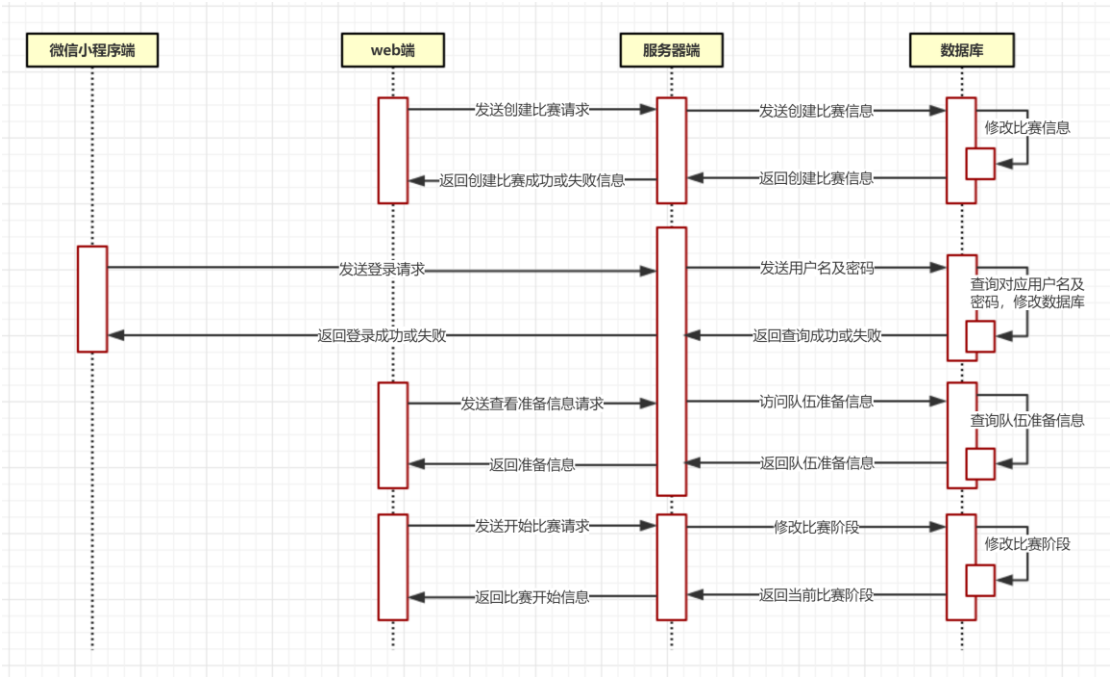
3.2.2.1 初始化阶段



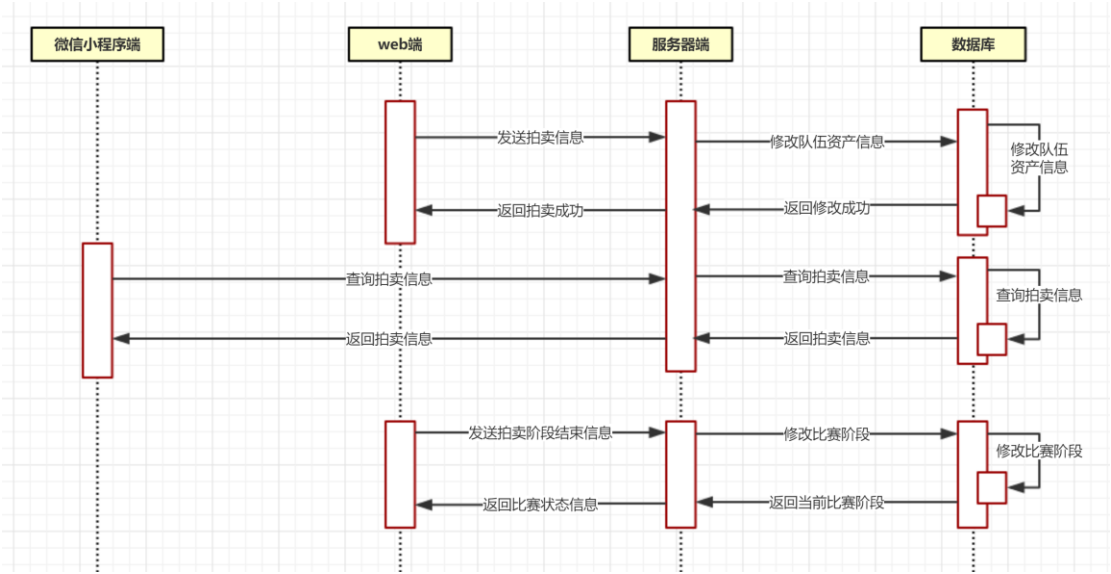
3.2.2.2 管理员创建或读取模板阶段（可选）



3.2.2.3 比赛初始化阶段

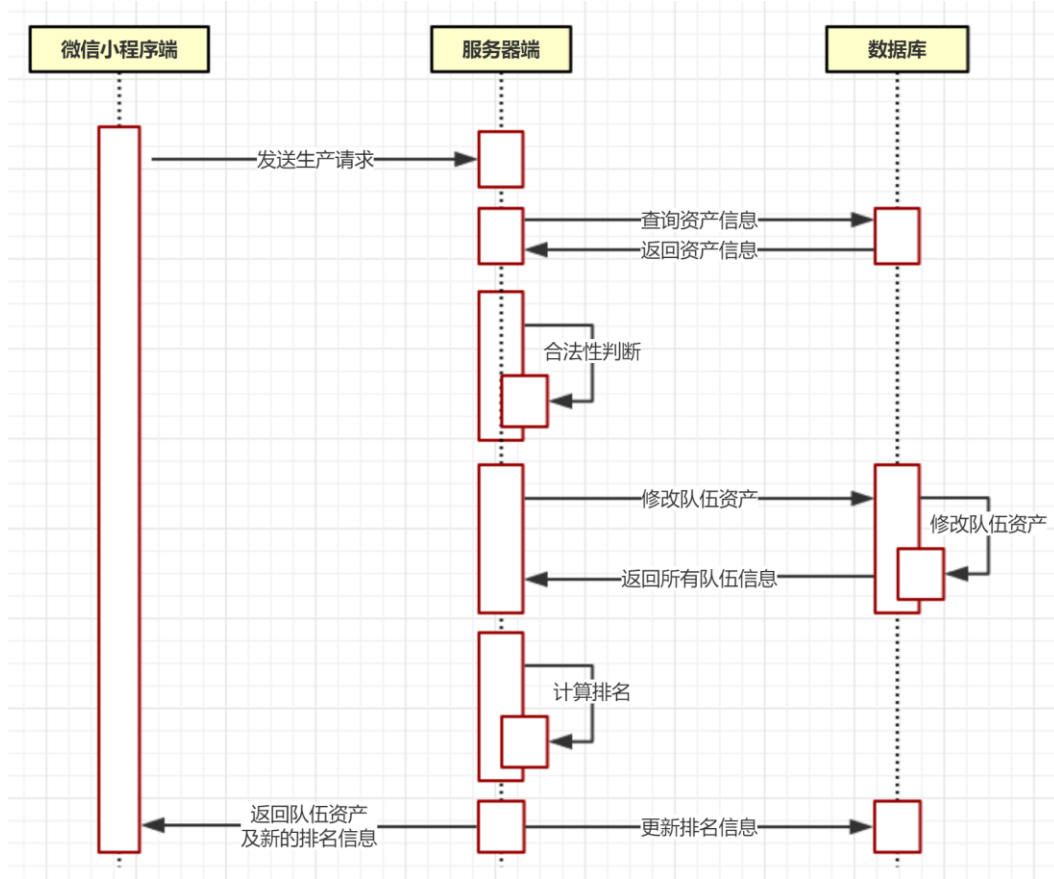


3.2.2.4 拍卖阶段

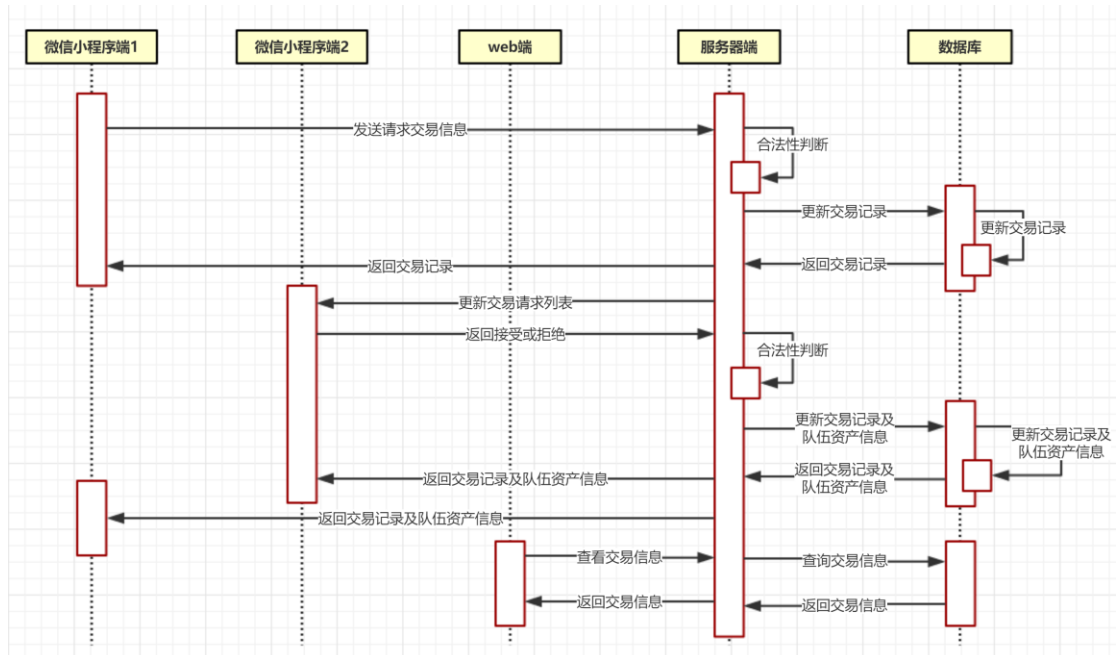


3.2.2.5 生产交易阶段

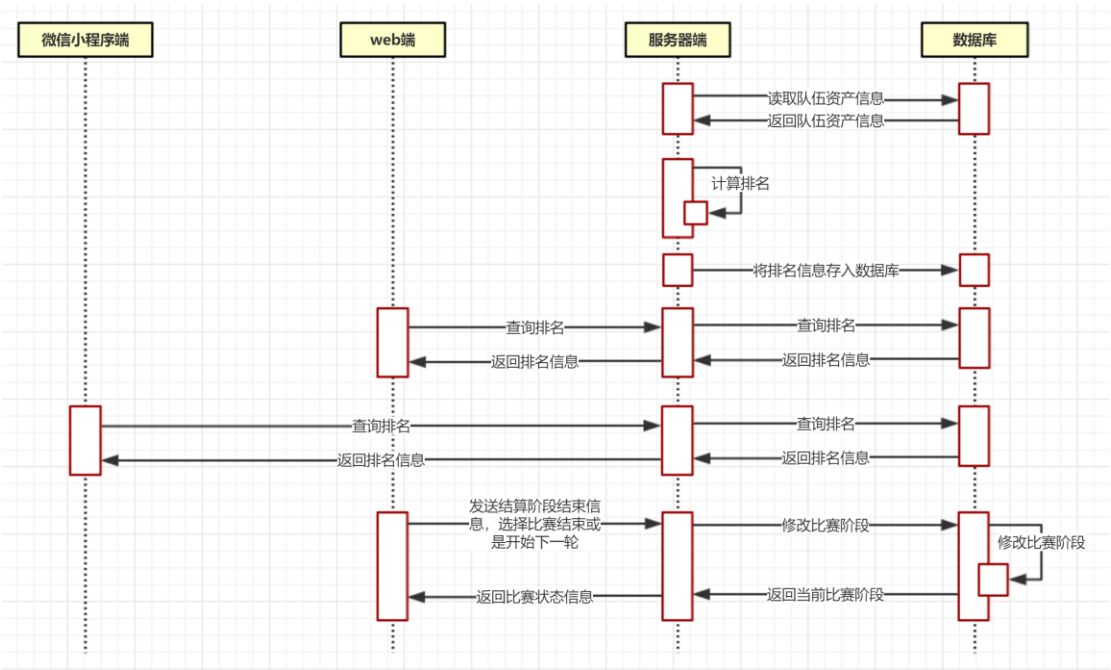
3.2.2.5.1 生产



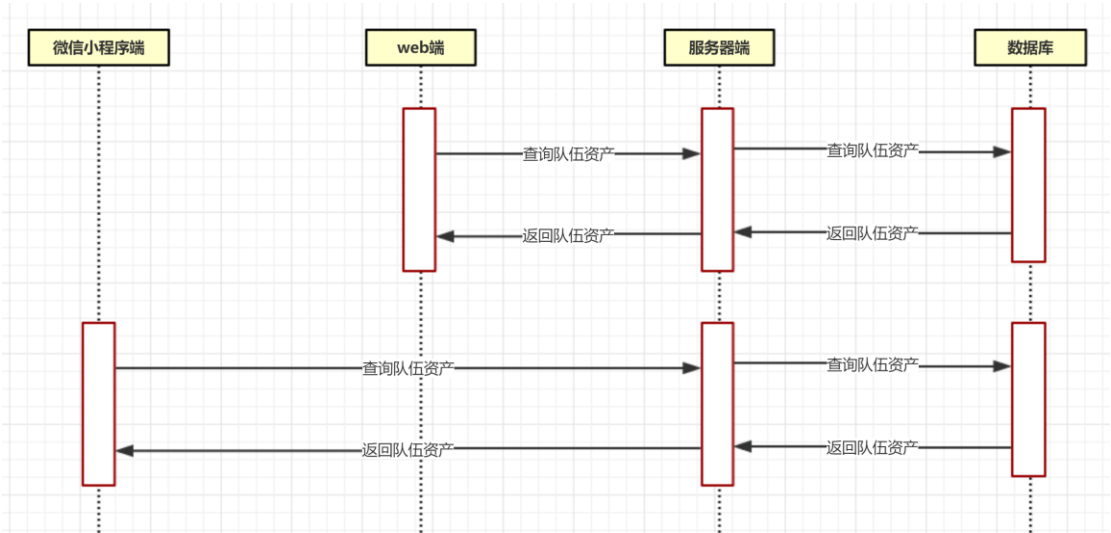
3.2.2.5.2 发起及响应交易请求，管理员查看交易信息



3.2.2.6 结算阶段



3.2.2.7 用户及管理员查看队伍资产



3.2.3 类和方法功能描述

3.2.3.1 概述

后端的文件结构如下：

```
E:.\
.coverage
build.sh
db.sqlite3
manage.py
README.md

api
  admin.py
  adminlogin.py
  apps.py
  auction.py
  build.py
  competition.py
  const.py
  consumers.py
  models.py
  produce.py
  routing.py
  team.py
  template.py
  tests.py
  trade.py
  try.py
  urls.py
  views.py
  __init__.py

backend2
  settings.py
  urls.py
  wsgi.py
  __init__.py
```

其中/api 文件夹为工程的主文件夹，其中各主要文件及所含的主要函数将在下一部分详细叙述。

3.2.3.2 后端主要文件、类及方法

下面将详细叙述每个主要文件中所含的主要类及方法

3.2.3.2.1、adminlogin.py

1、adminLogin(response)

传入管理员登录时的用户名和用户信息，返回含登录成功或失败信息的 json 对象。

3.2.3.2.2、auction.py

1、auctionRecord(compNumber, teamNumber, material, dMoney, description)

记录某一次拍卖信息的方法，传入比赛 id，队伍 id，材料种类，所花的钱和拍卖信息，返回 bool 变量表示成功或失败。

2、getAuctionInfoJson(infoList)

传入从数据库中获取的拍卖信息列表，返回可以发送给用户或管理员的 json 格式拍卖信息列表。

3.2.3.2.3、build.py

1、getTeamInformation(succeed, accModel, teamModel, compModel)

传入数据是否合法、从数据库中获取的用户、队伍、比赛模型，返回可以发送给用户或管理员的 json 格式队伍信息。

2、getOtherTeamList(teamModel, compModel)

传入从数据库中获取的队伍、比赛模型，返回可以发送给用户或管理员的 json 格式的所有其他队伍信息。

3、accountBuild(userid, teamid, gameid)

传入用户 id、队伍 id、比赛 id，返回 bool 变量表示是否合法、从数据库中获取的用户、队伍、比赛模型。限制用户必须是当前队伍中的。

4、otherTeamBuild(userid, teamid, gameid)

传入用户 id、队伍 id、比赛 id，返回 bool 变量表示是否合法、从数据库中获取的用户、队伍、比赛模型。限制用户必须不是当前队伍中的。

3.2.3.2.5、competition.py

1、buildCompetition(settings)

传入比赛配置，创建一场比赛，返回在数据库中新建立的当前比赛模型。

3.2.3.2.6、const.py

包含了一些常数的定义。

3.2.3.2.7、consumers.py

传输 websocket 和处理 websocket 请求的文件，包含了建立 websocket 连接、接收 websocket、将某些需要实时更新的信息（比赛阶段、倒计时、队伍资产信息、排名信息、交易信息）整理成 websocket 并发送的方法。

3.2.3.2.8、models.py

数据库模型的声明，包含统计比赛数、队伍数、用户数、机器数、管理员账号数的模型及机器、用户、队伍、比赛、记录的信息的模型。数据库模型的具体信息将在 3.3 部分进行详细说明。

3.2.3.2.8、produce.py

1、produceBuild(userid, teamid, gameid, machineid, totalMaterial, totalMoney)

进行某一次生产的方法，传入用户 id，队伍 id，比赛 id，所用机器 id，生产材料数，所花的钱，返回可以发送给用户或管理员的 json 格式对象代表生产是否成功。

2、getProduceInfoJson(infoList)

传入从数据库中获取的生产信息列表，返回可以发送给用户或管理员的 json 格式生产信息列表。

3.2.3.2.9、routing.py

针对 websocket 收发的地址定位文件，内含每个 websocket 接口需要调用哪个方法的信息。

3.2.3.2.10、team.py

1、buildTeam(teamName, teamCount, initialMoney, compNumber, memberNumber, initialPasswordLength)

建立队伍的方法，传入队伍名、队伍 id、初始金钱、比赛 id、成员数、初始密码长度，建立一个队伍和队伍中的成员并对每个成员生成用户名和初始密码。

2、buildAccount(username, password, teamNumber)

建立用户账号的方法，传入用户名、密码、队伍 id，建立一个用户账号。

3、getTotalResult(teamModels)

传入队伍列表，返回可以发送给用户或管理员的 json 格式队伍排名和资产详细信息列表。

4、teamRankTotal(teamModels)

传入队伍列表，返回可以发送给用户或管理员的 json 格式队伍排名和资产简略信息列表。

5、cmp_to_key(mycmp)

排序比较函数，用于比较两队之间资产

6、teamRankCalculate(teamModels, mRed, mGreen, mBlue)

队伍排名计算函数，传入队伍列表、每种材料各几个可以组成一个房子，计算每个队伍可以拥有几个房子和队伍排名

7、teamCalculate(teamModel, mRed, mGreen, mBlue)

队伍资产计算函数，传入当前队伍模型、每种材料各几个可以组成一个房子，计算当前队伍可以拥有几个房子

3.2.3.2.11、template.py

1、getTemplate()

从文件中读取所有比赛模板并返回

2、addTemplate(response)

传入一个请求，解析模板信息并将新的模板信息写入文件。返回一个 json 对象代表是否添加成功。

3.2.3.2.12、tests.py

单元测试文件，包含对各个方法进行单元测试和对整个比赛流程进行测试的函数。

3.2.3.2.13、trade.py

1、getTradeMessageJson(info, state)

整理交易信息，将传入的从数据库中读取的交易信息整理成 json 格式并返回。

2、message(response)

获取某一队伍的所有交易信息的方法。

3、sell(response)

发起一次卖出交易的方法。传入一个请求，返回 json 对象，代表是否成功发送。

4、cancel(response)

拒绝或取消一次交易的方法。传入一个请求，返回 json 对象，代表是否成功拒绝或取消。

5、accept(response)

接受一次交易的方法。传入一个请求，返回 json 对象，代表是否交易成功。

6、adminInfo(infoList)

管理员获取交易信息的方法。传入一个请求，返回 json 对象，代表交易信息列表。

3.2.3.2.14、urls.py

针对 http 收发的地址定位文件，内含每个 http 接口需要调用哪个方法的信息。

3.2.3.2.15、views.py

传输 http 信息和处理 http 请求的文件，包含了建立 http 连接、接收 http 信息、处理收到的 http 请求，进行相应的操作并将结果返回给用户或管理员的方法。

其中具体请求如下：

管理员（web 端）使用：

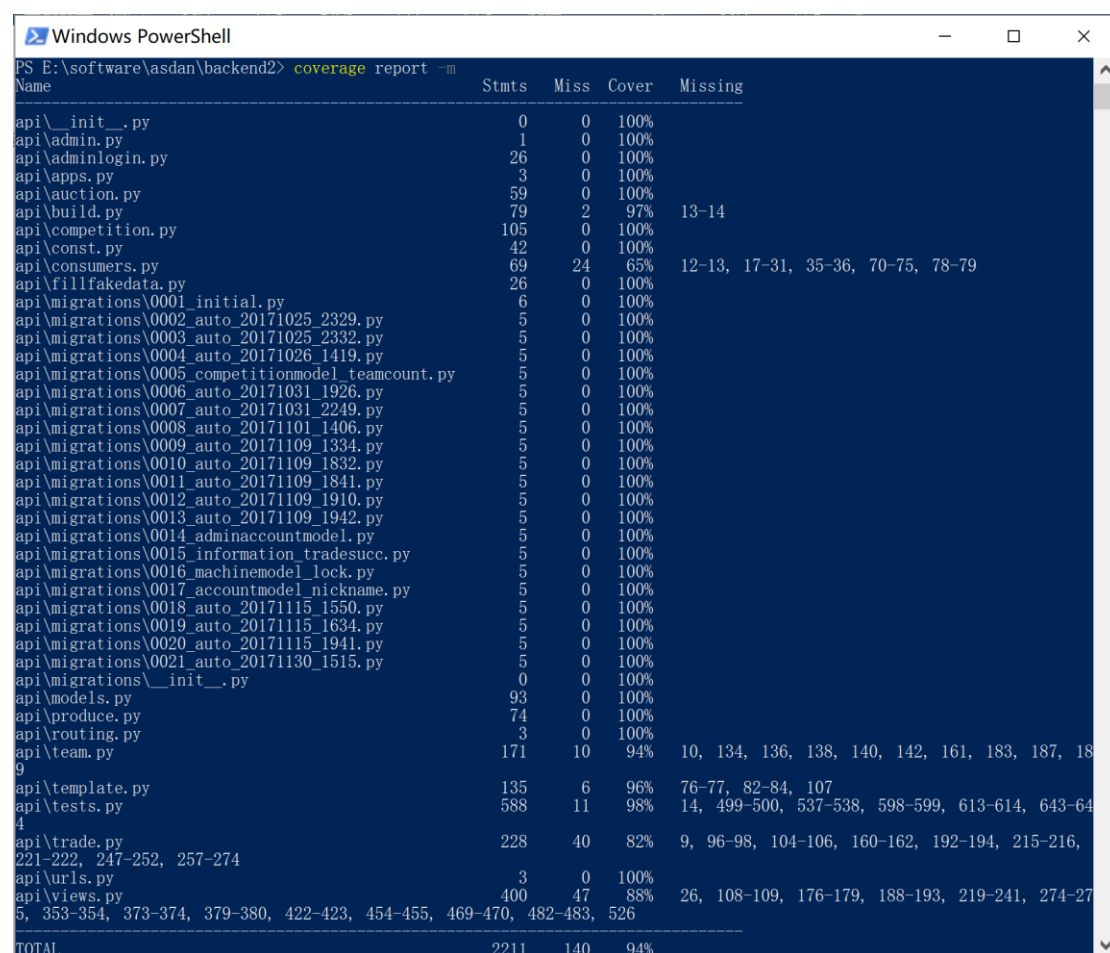
- 1、build(req): 创建比赛，创建模板或读取模板
- 2、overview(req): 比赛总览
- 3、main(req): 比赛阶段更改
- 4、ready(req): 查看队伍是否准备
- 5、sale(req): 记录拍卖信息
- 6、tradeApi(req): 设置生产交易环节的倒计时，获取交易信息列表
- 7、result(req): 获取队伍排名和资产信息
- 8、adminlogin(req): 管理员登录
- 9、deletgame(req): 删除一场比赛
- 10、account(req): 主动查询比赛阶段

用户（微信小程序端）使用：

- 11、login(req): 登录

- 12、userSetting(req): 设置用户昵称和密码
- 13、check(req): 主动查询队伍信息
- 14、next(req): 主动查询比赛阶段
- 15、deallnit(req): 查询生产交易阶段倒计时
- 16、dealMessage(req): 获取本队交易信息
- 17、retDeal(req): 接受或拒绝交易
- 18、auctionApi(req): 查看本队详细资产信息
- 19、ranking(req): 查询排名信息
- 20、myAuctionList(req): 查询本队拍卖信息
- 21、AuctionList(req): 查询整体拍卖信息
- 22、myProduceList(req): 查询本队生产信息
- 23、produceMaterial(req): 进行生产

3.2.4 单元测试结果



Name	StmtS	Miss	Cover	Missing
api_init_.py	0	0	100%	
api\admin.py	1	0	100%	
api\adminlogin.py	26	0	100%	
api\apps.py	3	0	100%	
api\auction.py	59	0	100%	
api\build.py	79	2	97%	13-14
api\competition.py	105	0	100%	
api\const.py	42	0	100%	
api\consumers.py	69	24	65%	12-13, 17-31, 35-36, 70-75, 78-79
api\fillfakedata.py	26	0	100%	
api\migrations\0001_initial.py	6	0	100%	
api\migrations\0002_auto_20171025_2329.py	5	0	100%	
api\migrations\0003_auto_20171025_2332.py	5	0	100%	
api\migrations\0004_auto_20171026_1419.py	5	0	100%	
api\migrations\0005_competitionmodel_teamcount.py	5	0	100%	
api\migrations\0006_auto_20171031_1926.py	5	0	100%	
api\migrations\0007_auto_20171031_2249.py	5	0	100%	
api\migrations\0008_auto_20171101_1406.py	5	0	100%	
api\migrations\0009_auto_20171109_1334.py	5	0	100%	
api\migrations\0010_auto_20171109_1832.py	5	0	100%	
api\migrations\0011_auto_20171109_1841.py	5	0	100%	
api\migrations\0012_auto_20171109_1910.py	5	0	100%	
api\migrations\0013_auto_20171109_1942.py	5	0	100%	
api\migrations\0014_adminaccountmodel.py	5	0	100%	
api\migrations\0015_information_tradesucc.py	5	0	100%	
api\migrations\0016_machinemodel_lock.py	5	0	100%	
api\migrations\0017_accountmodel_nickname.py	5	0	100%	
api\migrations\0018_auto_20171115_1550.py	5	0	100%	
api\migrations\0019_auto_20171115_1634.py	5	0	100%	
api\migrations\0020_auto_20171115_1941.py	5	0	100%	
api\migrations\0021_auto_20171130_1515.py	5	0	100%	
api\migrations_init_.py	0	0	100%	
api\models.py	93	0	100%	
api\produce.py	74	0	100%	
api\routing.py	3	0	100%	
api\team.py	171	10	94%	10, 134, 136, 138, 140, 142, 161, 183, 187, 189
api\template.py	135	6	96%	76-77, 82-84, 107
api\tests.py	588	11	98%	14, 499-500, 537-538, 598-599, 613-614, 643-644
api\trade.py	228	40	82%	9, 96-98, 104-106, 160-162, 192-194, 215-216, 221-222, 247-252, 257-274
api\urls.py	3	0	100%	
api\views.py	400	47	88%	26, 108-109, 176-179, 188-193, 219-241, 274-275, 353-354, 373-374, 379-380, 422-423, 454-455, 469-470, 482-483, 526
TOTAL	2211	140	94%	

单元测试结果如下，可以看到覆盖率达到了 94%。而未覆盖的部分是因为 websocket 架构基于 django 的 channels 库，需要与微信小程序联动，在 test 程序中难以测试，因此无法运行到某几行。

3.3 数据库设计

数据库的主要功能是存储并提供相关数据，包括各场比赛、各个队伍、各个用户及每个机器、每条消息的相关数据。采用 `django` 框架自带的数据库接口（`models`），进行与 `SQLite3` 数据库的交互。

数据表的设计如下

3.3.1、CompetitionCount 表

列名	属性	说明
name	CharField(30)	一个比赛计数器的名称
count	IntegerField()	一个比赛计数器的统计结果

比赛计数器，统计当前比赛数

3.3.2、TeamCount 表

列名	属性	说明
name	CharField(30)	一个队伍计数器的名称
count	IntegerField()	一个队伍计数器的统计结果

队伍计数器，统计当前队伍数

3.3.3、AccountCount 表

列名	属性	说明
name	CharField(30)	一个用户计数器的名称
count	IntegerField()	一个用户计数器的统计结果

用户计数器，统计当前用户数

3.3.4、MachineCount 表

列名	属性	说明
name	CharField(30)	一个机器计数器的名称
count	IntegerField()	一个机器计数器的统计结果

机器计数器，统计当前机器数

3.3.5、InformationCount 表

列名	属性	说明
name	CharField(30)	一个消息计数器的名称

count	IntegerField()	一个消息计数器的统计结果
-------	----------------	--------------

消息计数器，统计当前消息数

3.3.6、AdminAccountModel 表

列名	属性	说明
username	CharField(30)	管理员用户名
password	CharField(30)	管理员密码

管理员账户模型，保存用户名及密码

3.3.7、MachineModel 表

列名	属性	说明
number	IntegerField()	机器编号
material	IntegerField()	机器可以生产的材料
rest	IntegerField()	机器剩余生产次数
lock	IntegerField()	机器是否被锁定（锁定后不能使用）
teamNumber	IntegerField()	机器属于的队伍 id

机器模型，保存一个机器的信息

3.3.8、AccountModel 表

列名	属性	说明
username	CharField(30)	用户名
password	CharField(30)	密码
nickname	CharField(30)	用户昵称
number	IntegerField()	用户 id
teamNumber	IntegerField()	用户属于的队伍 id
ready	IntegerField()	用户是否准备就绪

用户模型，保存一个用户的信息

3.3.8、TeamModel 表

列名	属性	说明
name	CharField(30)	队名
teamNumber	IntegerField()	队伍在当前比赛的编号
number	IntegerField()	队伍 id（不会与其他比赛中的队伍重复）
peopleCount	IntegerField()	队伍人数
materialRed	IntegerField()	队伍拥有的红色材料（砖头）数
materialGreen	IntegerField()	队伍拥有的绿色材料（木材）数

materialBlue	IntegerField()	队伍拥有的蓝色材料（金属）数
money	IntegerField()	队伍拥有的金钱
rank	IntegerField()	队伍排名
houses	IntegerField()	队伍拥有的房子数
ready	IntegerField()	队伍是否准备就绪
UPmessage	IntegerField()	是否需要更新信息

队伍模型，保存一个队伍的信息

3.3.9、CompetitionModel 表

列名	属性	说明
name	CharField(30)	比赛名称
date	CharField(30)	比赛日期
number	IntegerField()	比赛 id
peopleCount	IntegerField()	比赛人数
peopleCreated	IntegerField()	已创建的账号数
machineCount	IntegerField()	已拍卖的机器数
teamCount	IntegerField()	队伍数
money	IntegerField()	每个队伍的初始金钱
competitionState	IntegerField()	比赛状态（处于哪个阶段）
turn	IntegerField()	比赛处于第几轮
materialNeedRed	IntegerField()	建造一个房子需要的红色材料（砖头）数
materialNeedGreen	IntegerField()	建造一个房子需要的绿色材料（木材）数
materialNeedBlue	IntegerField()	建造一个房子需要的蓝色材料（金属）数
moneyNeedRed	IntegerField()	生产一个红色材料需要的金钱数
moneyNeedGreen	IntegerField()	生产一个绿色材料需要的金钱数
moneyNeedBlue	IntegerField()	生产一个蓝色材料需要的金钱数
machinePriceRed	IntegerField()	红色机器（生产砖头）的起拍价
machinePriceGreen	IntegerField()	绿色机器（生产木材）的起拍价
machinePriceBlue	IntegerField()	蓝色机器（生产金属）的起拍价
UPtime	IntegerField()	倒计时是否更新
time	CharField(30)	倒计时
desc	CharField(30)	比赛介绍
initialPasswordLength	IntegerField()	每队初始密码长度
teams	ManyToManyField (TeamModel)	比赛中的每个队伍

比赛模型，保存一场比赛的信息

3.3.10、Information 表

列名	属性	说明
----	----	----

category	CharField(30)	信息类型
description	CharField(30)	信息详细描述
time	CharField(30)	信息时间
compnum	IntegerField()	信息所属比赛 id
team1num	IntegerField()	与信息有关的队伍 1 的 id
team2num	IntegerField()	与信息有关的队伍 2 的 id（可能为空）
dmoney	IntegerField()	信息对应的事件的金钱变化量
material	IntegerField()	信息对应的事件的变化材料种类
dmaterial	IntegerField()	信息对应的事件的材料变化量
restmaterial1	IntegerField()	队伍 1 当时剩余的材料
restmaterial2	IntegerField()	队伍 2 当时剩余的材料
restmoney1	IntegerField()	队伍 1 当时剩余的金钱
restmoney2	IntegerField()	队伍 2 当时剩余的金钱
machineid	IntegerField()	信息对应的事件的机器 id
machinerest	IntegerField()	信息对应的事件的机器剩余量
trademachine	ManyToManyField (MachineModel)	该信息为交易信息时，交易的机器
materialred	IntegerField()	该信息为交易信息时，交易的红色材料数
materialgreen	IntegerField()	该信息为交易信息时，交易的绿色材料数
materialblue	IntegerField()	该信息为交易信息时，交易的蓝色材料数
tradetype	IntegerField()	交易类型（正在进行，成功，失败）
number	IntegerField()	信息 id

信息模型，保存一条信息（生产信息、交易信息、拍卖信息）

3.4 Web 端（管理员端）

3.4.1 Web 端功能简介

Web 端为赛事的管理员使用，主要功能为管理所有赛事、控制一场比赛的进行以及比赛数据的统计和记录。

初始化：赛事管理员设置参赛人数、起始资金等比赛信息，然后创建一场新的比赛，给每个参赛团队准备登录账号和密码。

拍卖：赛事管理员负责记录现场的拍卖结果并将数据发送给各队伍。

生产：进入生产/交易环节后赛事管理员发起倒计时并可浏览各订单情况。

结算：管理员端显示各队伍的信息。

可视化：比赛过程中提供单独的界面实时展示各个团队的资产信息以及生产/交易环节的倒计时。提供各种表格用于比赛的数据统计。

整个前端设计结构完整，功能齐全，易用性强，符合目前主流的网页设计要求。

3.4.2 Web 端设计总览

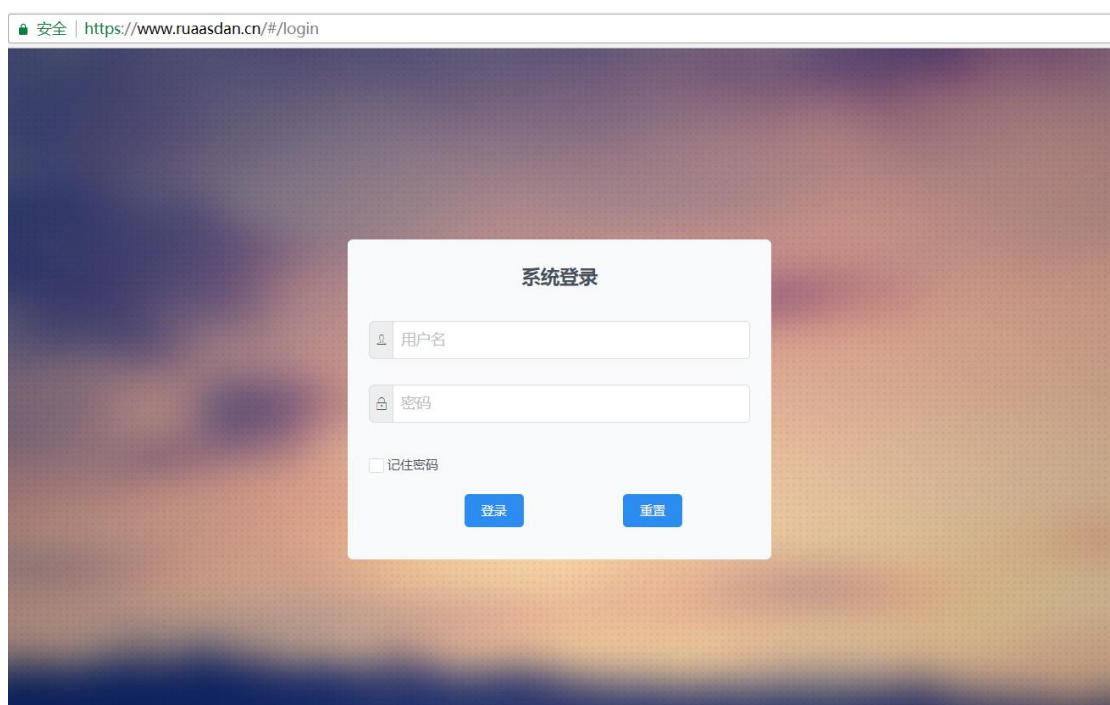
Web 端共有 12 个页面，分别为：

- 登录界面
- 比赛总览界面
- 创建比赛界面
- 比赛流程界面
- 队伍准备界面
- 拍卖环节界面
- 生产/交易环节界面
- 结算环节界面
- 屏幕展示入口界面
- 屏幕管理界面
- 大屏幕单独展示界面
- 表格管理总览界面

各页面间的逻辑为：首先展示登录界面，登录成功后进入比赛总览，此时可以进入创建比赛界面或进入一场比赛，进入一场比赛会进入该比赛的队伍准备界面来开始比赛。在一场比赛中通过比赛流程界面来管理整场比赛的进行，具体为拍卖环节界面→生产/交易环节界面→结算环节界面。在比赛进行过程中随时可以通过左边侧栏进入屏幕展示入口界面、屏幕管理界面或表格管理总览界面，从屏幕展示入口界面可以进入大屏幕单独展示界面。

3.4.3 Web 端设计详细介绍

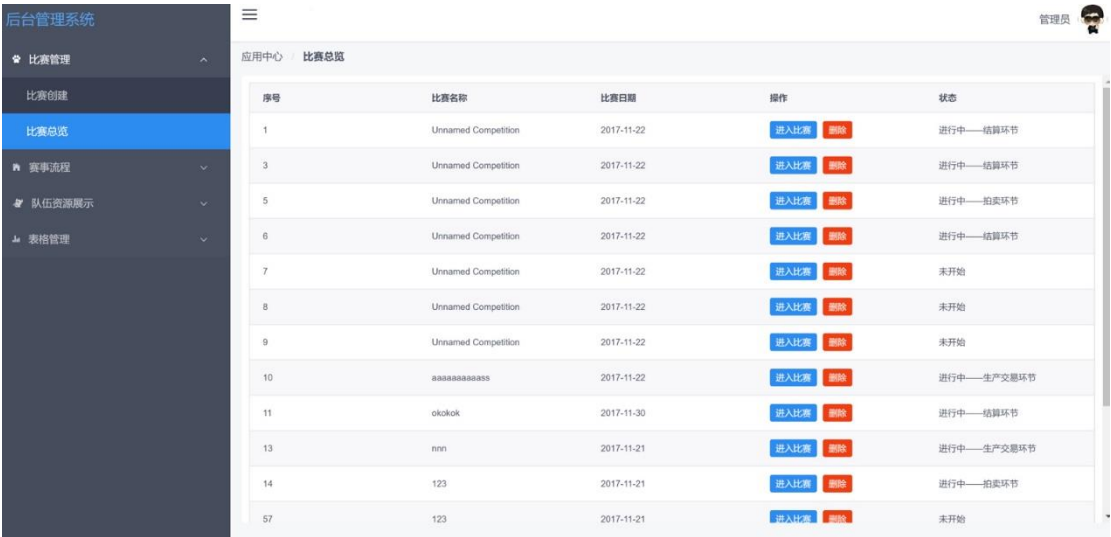
3.4.3.1 登录界面



只有管理员账号可以登录，为开设注册功能，给管理员提供了默认账号 **admin1** 和密码 **asdfghjkl**，防止所有人员都可以登录管理员端。

登录界面的设计比较简洁，采用 **admin-vue** 组件，填入默认账号密码点击登录即可。

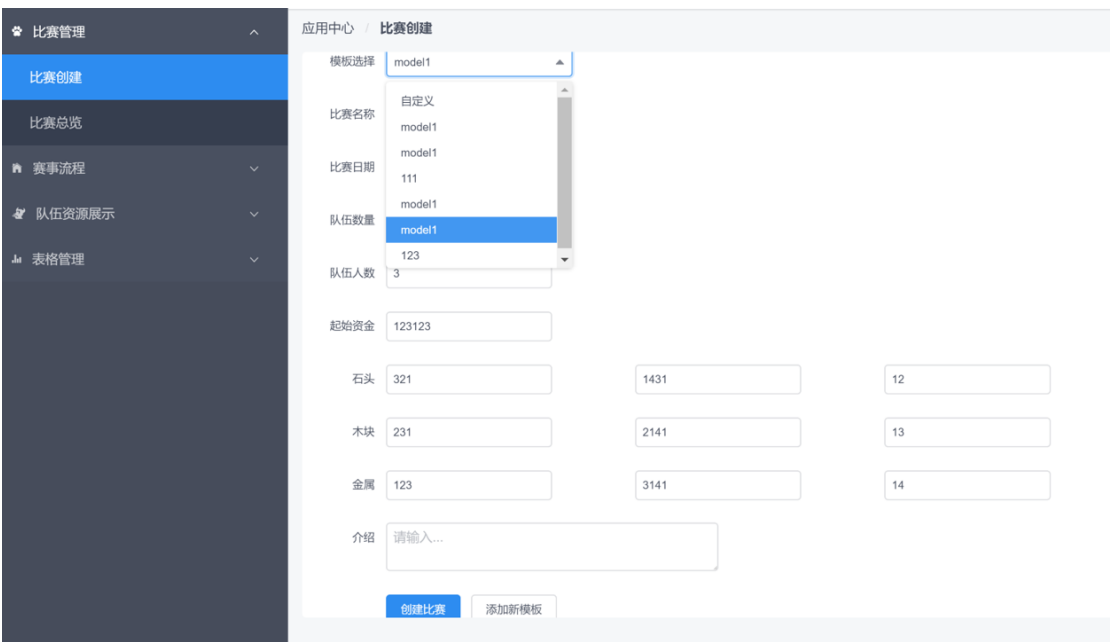
3.4.3.2 比赛总览界面



页面左侧为侧栏，总览情况下可点击比赛创建。页面主体显示所有比赛的情况，设计“进入比赛”和“删除”按钮，点击“进入比赛”按钮可以进入该场比赛，实现对该场比赛的管理；点击“删除”按钮可以删除已经结束或者未开始的比赛。

页面采用 **vue-admin** 经典的界面架构，清晰明了，用户很好使用。

3.4.3.3 比赛创建界面

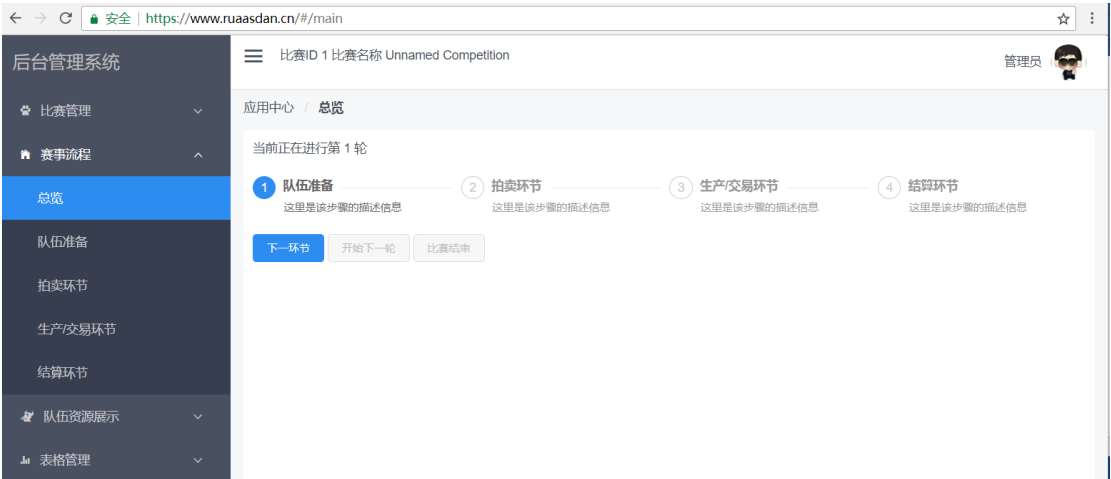


比赛创建页面中模板选择可以通过下拉菜单选择对应模板导入，选择自定义时可以将这

次设计的比赛情况通过最下方的“添加新模板”按钮来添加为新的模板，新的模板会生成文件存在本地，点击最下方的“创建比赛”按钮会以上面的参数来创建比赛。

表单采用了 `iview` 的 `form` 组件，数据封装成一个 `json` 格式，提交时发送给后台，页面加载时向后台请求模板的列表和各模板的信息。

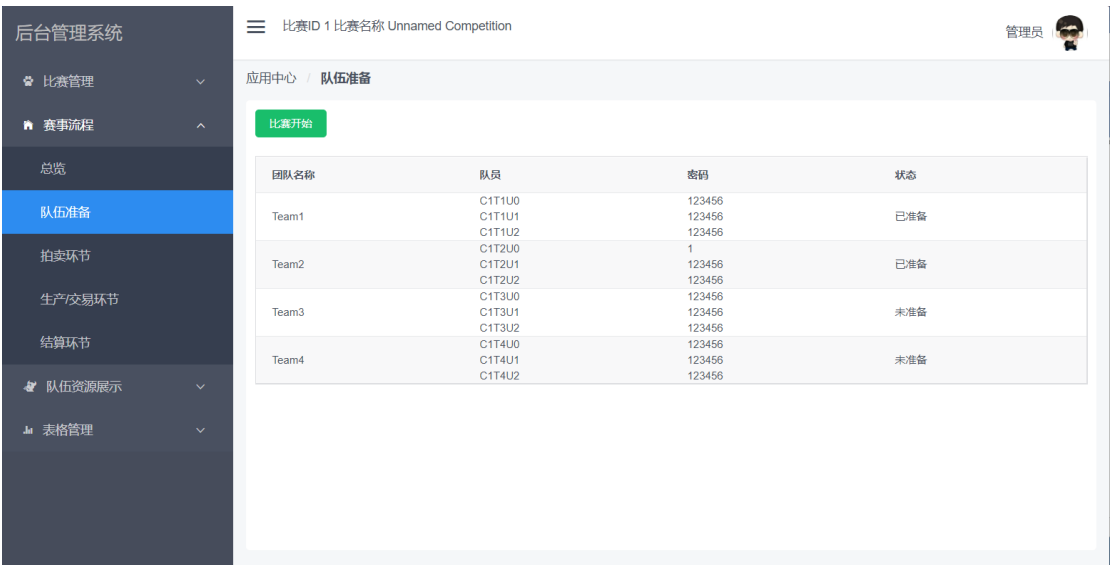
3.4.3.4 比赛流程界面



页面的左边侧栏赛事流程展开，展现整个赛事的所有流程。页面主体的最上方展示当年管理的比赛 ID 和名称。下面有三个按钮，“下一环节”按钮使环节的显示跳转到下一环节，并控制页面的跳转。“开始下一轮”和“比赛结束”按钮在比赛进行到结算环节时可以点击，此时“下一环节”按钮为不可点击状态，此时可以选择进行下一轮比赛或者本次比赛结束。

与后端的数据交互中每次点击按钮的时候通过发送一个 `json` 封装的数据，变量包括比赛 id，进行轮数和当前环节情况。

3.4.3.5 队伍准备界面



主体和赛事流程界面差距为用表格来实时显示各队伍的准备情况，管理员可实时看到各

队伍的准备情况来决定比赛的开始——点击“比赛开始”按钮，此时页面跳转至流程总览。
表格的实现为 `iview` 的 `table` 组件，将后台获取的 `json` 数据直接可以导入到表格展示。

3.4.3.6 拍卖环节界面

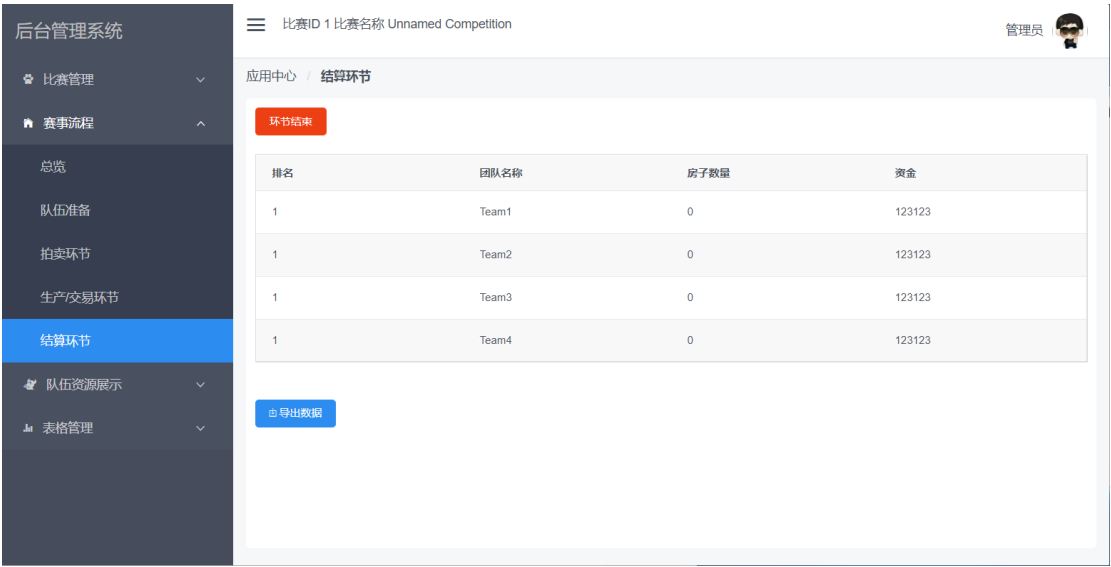
通过下拉菜单选择拍得的队伍和机器，输入拍卖的金额，点击提交向队伍发送拍卖信息，点击“环节结束”按钮可以跳转到赛事流程界面。

表单采用的是 `iview` 的 `form` 组件，与 `json` 数据绑定，发送给后端的是封装好的 `json` 数据。

3.4.3.7 生产/交易环节界面

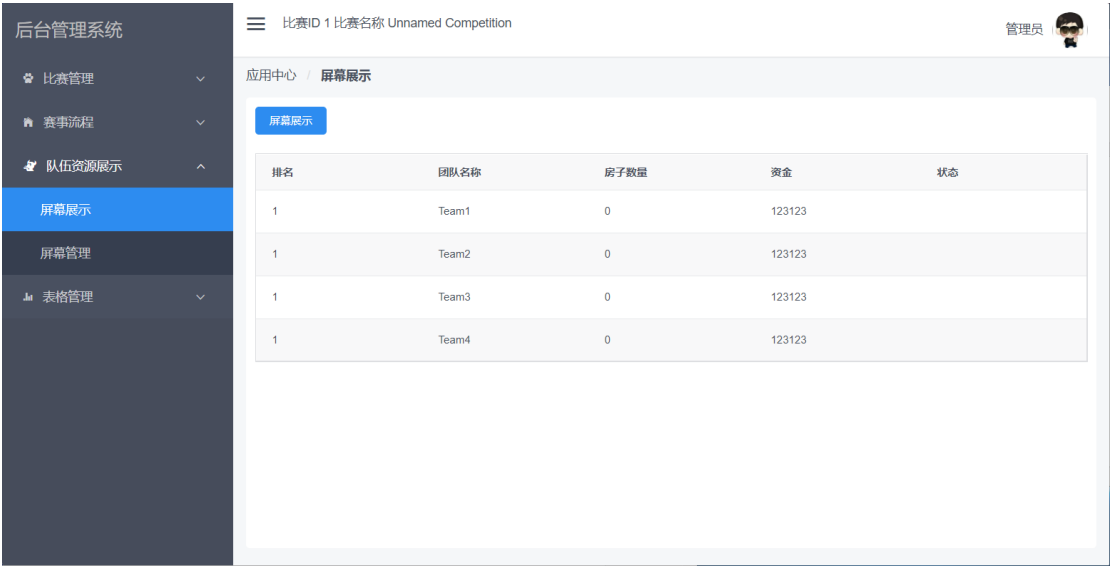
倒计时可以自行设计时长，点击“倒计时开始”按钮开始计时，倒计时时长可以任意通过时长设置更改。倒计时显示用的一个开源库做的样式，写成一个单独的组件，在大屏幕展示中也使用了该组件。下方是一个订单的表格，同样使用 `iview` 的 `table` 组件，通过点击“导出数据”按钮可以将表格数据导出为 `excel`。

3.4.3.8 结算环节界面



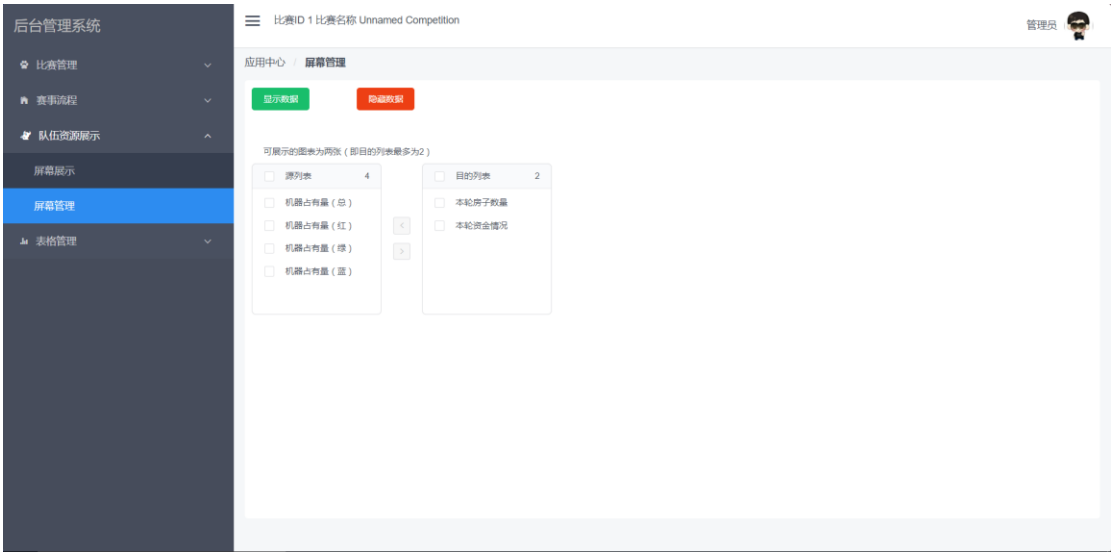
结算环节展示各队伍该轮的排名和具体信息，数据可以通过点击“导出数据”按钮生成 excel 文件，通过点击“环节结束”按钮跳转到赛事流程界面。

3.4.3.9 屏幕展示入口界面



通过点击“屏幕展示”按钮弹出一个新的网页作为大屏幕展示，下面表格为大屏幕展示的排名部分。

3.4.3.10 屏幕管理界面



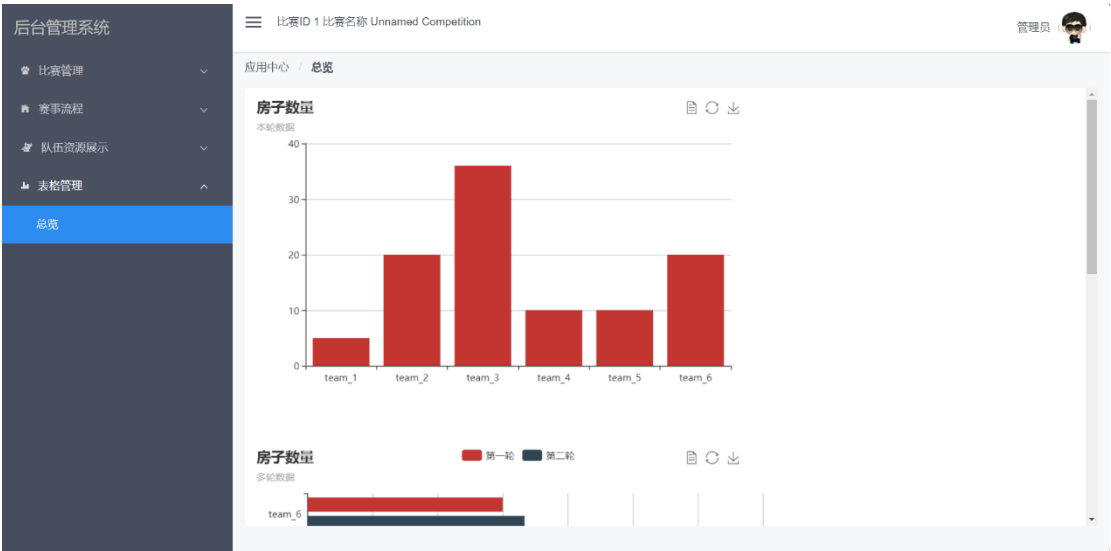
“显示数据”和“隐藏数据”按钮控制大屏幕排名等数据是否显示，为实现环节结束前 5 分钟数据隐藏来保持神秘感。下面使用 `iview` 的 `transfer` 组件实现的穿梭框，右侧最多为两个数据，为展现在大屏幕上的表格。在该界面设置好后，大屏幕刷新后则显示对应的表格。

3.4.3.11 大屏幕单独展示界面



最上方为生产/交易环节的倒计时同生产/交易界面的倒计时同步。下方的左侧为队伍的排名及相关信息，右侧为数据统计的表格展示，二者均可在屏幕管理页面被修改。其中表格使用了 `echarts` 开源库。

3.4.3.12 表格管理总览界面



展示各种表格的数据统计，柱状图、条形图和饼状图。该表格均可点击右上角最右按钮下载为图片保存。最左边按钮为显示数据，中间按钮为刷新表格，该表格采用了 echarts 的开源库。

3.5 微信小程序（用户端）

3.5.1 用户端功能简介

微信小程序为赛事参赛队员使用，通过赛事方线下分配的密码登录，可以在管理员与赛事队员的共同操作，完成一场比赛，具体操作如下：

登录：通过线下派发的账号及密码登录。

修改密码及用户名。

拍卖：可实时更新拍卖信息，支持本队拍卖纪录查看及其他所有队伍的信息查看。

生产交易：队员可以进行物资生产及交易相关操作，同时支持信息查看。本环节有时间限制，有倒计时功能，最后 5 分钟不允许信息查看。

结算：统计排名。

上诉所有功能界面的跳转皆有管理员操作，队员不可返回至上一环节。

3.5.2 用户端设计总览

微信小程序共有 7 大页面，分别为：

- 登录界面
- 修改密码及用户名界面
- 等待界面
- 拍卖界面

- 拍卖信息界面
- 信息查看界面
- 生产/交易界面
 - 实时排名界面
 - 交易界面
 - 生产界面
 - 信息查看界面
- 最终排名界面
- 队伍信息界面

界面之间的关系如下：用户通过登录界面登录某一拍卖游戏，然后进入修改密码及用户名界面，用户可选择进入跳过此环节。然后进入等待界面，等待管理员开始，此界面会自动跳转，进入拍卖界面。在拍卖界面，用户在拍卖信息界面可以查看最新的拍卖信息，然后在信息查看界面可以查看其它队伍信息，即进入队伍信息界面。拍卖环节结束后，管理员进入下一环节，用户端自动跳转生产/交易界面，此环节中，用户可以在实时排名界面，实时得到当前最新排名，然后交易界面可以进行交易，生产界面可以生产，同时可以进行信息查看。最后是最终排名界面。

3.5.3 用户端设计详细介绍

3.5.3.1 登录界面



赛事队员只有在赛事刚开始时，根据赛事方线下分发的账号密码才可以登录。一场比赛结束后账号密码作废，用户不可再次登录。

该界面同时提供了网络错误及账号或用户名错误等登录提示。

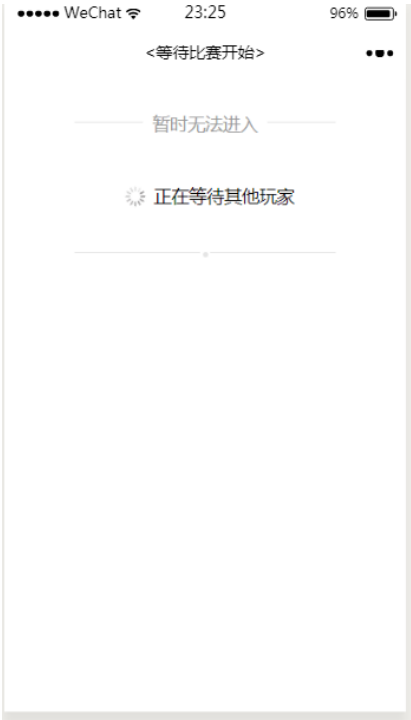
设计上采用了微信官方提供的 WeUI 样式，符合微信小程序设计规范。

3.5.3.2 修改密码和用户名界面



在此界面，参赛队员可以设置新的密码及一个个性化的昵称，然后点击“确认修改”，系统会返回修改成功与否的结果。当然，用户也可以直接跳过此环节，点击“跳过”按钮即可，此时用户昵称默认为用户账号名。

3.5.3.3 等待界面



用户进入此界面后，即等待游戏开始，此时用户端不能进行任何操作，同时若某位用户进入该界面后，管理员可以实时得知，当所有队员全部准备完毕后，管理员点击进入下一环节（拍卖环节）后，用户界面会自动跳转。

此界面主要使用 **websocket** 实时监听服务器端的消息，便于界面跳转。

3.5.3.4 拍卖信息界面



在拍卖信息界面，用户可以实时得到最新的拍卖信息，并且当一个新的拍卖信息出现时，界面有实时提醒功能，便于用户手动刷新最新的拍卖信息。

为了便于用户查看，本队的拍卖纪录的背景色为淡绿色，利于和其他队伍拍卖信息区分。

3.5.3.5 信息查看界面

3.5.3.6 实时排名界面



本界面提供了实时排名信息，环节倒计时功能。在环节开始、比赛只有五分钟以及环节结束都有相关提醒功能。排名界面中，前三名分别以不同的字体和背景色区分，同时本队也以不同的背景色区分，便于赛事队员观看，同时，当倒计时大于五分钟时，点击某支队伍可以直接进入其队伍信息界面进行详细的信息查看。

本环节采用了 **websocket** 对排名信息进行实时的更新，用户不需要对排名更新做任何操作。

3.5.3.7 交易界面



此界面为交易界面，用户可以发起一个主动的交易申请（卖出），该界面支持选定交易对象，收取金币数，卖出的材料及机器。此环节服务器端会有交易非法判断，用户端点击“发送交易申请”后，系统会反馈给用户交易结果。同时该界面还有一个交易信息子界面，在该界面，有所有的本队相关的交易信息，用户可以在此进行交易相关的操作，如同意交易、拒绝其他队伍发起的交易以及撤回自己的交易。

交易信息子界面采用了 **websocket** 进行实时的信息提醒。

3.5.3.8 生产界面



在生产界面，用户可以对自己拥有的机器进行生产操作。

3.5.3.9 最终排名界面



一轮比赛结束后，界面会跳转到最终排名界面，显示比赛结束后最终的拍卖信息，同时用户可以点击某支队伍，进入到该队伍的队伍信息界面进行查看。

3.5.3.10 队伍信息界面



在队伍信息界面，用户查看一支队伍所拥有的所以资产情况以及所含队员。

4 接口设计

4.1 Web 端与后端

4.1.1 接口：<https://www.ruaasdan.cn/#/build>

```
数据{  
    name: "",  
    templetname:"",  
    date: "",  
    teamnumber:"",  
    membernumber:"",  
    money: "",  
    desc: "",  
    redprice:"",  
    rmachineprice:"",  
    rpercent:"",  
    greenprice:"",  
    gmachineprice:"",  
    gpercent:"",  
    blueprice:"",  
    bmachineprice:"",  
    bpercent:"",  
}
```

为一场比赛构成的基本信息，为方便下面表述，将其中所有参数记为 **GAME** 集合。

4.1.1.1 页面初始化时向后台请求模板信息

发出参数：

Type:'requiremodel'

返回 json:

```
Data1:[{  
    GAME1, GAME2, ..., GAMEn  
}]
```

4.1.1.2 向后台发出创建比赛所需的信息

发出参数：

Type:'game',

GAME

返回 json:


```
{
  "ok": yes/no,
  "gameid":
}
```

4.1.1.3 向后台发出添加新模板的信息

发出参数:

```
Type:'addmodel',
GAME
```

返回 json:

```
{
  "ok": yes/no,
  "gameid":
}
```

4.1.2 接口: <https://www.ruaasdan.cn/#/overview>

向后台请求所有比赛的情况数据

方法: get

发出参数: 无

返回 json:

```
data1: [
  {
    number:1,
    name: 'GAME1',
    time: '2017-10-11',
    operate:"",
    status: '进行中'
  }
]
```

4.1.3 接口: <https://www.ruaasdan.cn/#/main>

比赛流程界面时向后台发送比赛信息

发出参数:

```
{
  Gameid:3,
  Turn:1,
  Unit:2
}
```

```
}
```

返回 json: 无

4.1.4 接口: <https://www.ruaasdan.cn/#/ready>

一场比赛中各队伍准备情况

发出参数:

```
{  
    Gameid: 3  
}
```

返回 json:

```
data1: [  
    {  
        number:1,  
        name: 'GAME1',  
        password:'',  
        member: 'a1 a2 a3',  
        status: '准备中'  
    }  
]
```

4.1.5 接口: <https://www.ruaasdan.cn/#/trade>

交易环节不断向后端请求订单信息并发送倒计时信息

发出参数:

```
{  
    Gameid:  
    Time:  
}
```

返回 json:

```
data1: [  
    {  
        number:"",  
        partA:"",  
        partB:"",  
        money:"",  
        operate:""  
    }  
]
```

4.1.6 接口: <https://www.ruaasdan.cn/#/sale>

拍卖环节向后台发送拍卖信息

发出参数:

```
{
    Gameid:
    Teamid:
    Machine:
    Money:
    Desc:
}
```

返回 json: 无

4.1.7 接口: <https://www.ruaasdan.cn/#/result>

向后台请求各队伍信息

发送参数:

```
{
    Gameid: ''
}
```

返回 json:

```
data1: [{
    rank:",
    teamid:",
    name:",
    home:",
    money:",
    allmachine:",
    redmachine:",
    greenmachine:",
    bluemachine:"
}]
```

4.1.8 接口: <https://www.ruaasdan.cn/#/adminin>

登录信息验证

发送参数:

```
{
    Username: ''
    Password:"
```

```
}  
返回 json:  
{  
    "ok":yes/no  
}
```

4.2 微信小程序与后端

4.2.1 接口 <https://www.ruaasdan.cn/api/login>

作用：用于用户登录

方法：POST

发送数据：

```
data:{  
    username:"",  
    password:""  
}
```

返回参数 “go” 若为 ‘no’，表示登录失败，否则会返回数据：

```
{  
    go:"",  
    gameid:"",  
    teamid:"",  
    userid:"",  
    state:"",  
    round:"",  
    nickname:"",  
    money:"",  
    teamMates:"",  
    material:"",  
    machine:"",  
    otherteam:""  
}
```

4.2.2 接口 <https://www.ruaasdan.cn/api/userSetting>

作用：用户用户修改密码及用户名

方法：POST

发送数据：

```
data:{  
    userid:"",  
    teamid:""
```

```
        gameid:"",
        nickname:"",
        newPassWord:""
    }

```

返回数据:

```
    data:{
        go:""
    }

```

若“go”为‘yes’则表示修改成功，否则失败。

4.2.3 接口 <https://www.ruaasdan.cn/api/auction>

作用：用户进入拍卖界面后获得赛事初始信息

方法：POST

发送数据:

```
    data:{
        userid:"",
        teamid:"",
        gameid:""
    }

```

返回数据:

```
    data:{
        message:"",
        round:"",
        teamname:"",
        money:"",
        teammate:"",
        material:"",
        machine:"",
        otherTeam:"",
        nickname:""
    }

```

若“message”为‘succeed’ ,表示获取初始信息成功，否则失败。

4.2.4 接口 <https://www.ruaasdan.cn/api/AuctionList>

作用：获取所有拍卖纪录

方法：POST

发送数据

```
    data:{
        gameid:""
    }

```

返回数据，一个包含所有拍卖纪录的列表

4.2.5 接口 <https://www.ruaasdan.cn/api/myAuctionList>

作用：获得本队的拍卖纪录

方法：POST

发送数据：

```
data:{
    userid:"",
    teamid:"",
    gamid:""
}
```

返回数据：一个包含本队所有拍卖纪录的列表

4.2.6 接口 <https://www.ruaasdan.cn/api/check>

作用：获取其他队伍的队伍信息

方法：POST

发送数据：

```
data:{
    userid:"",
    teamid:"",
    gameid:""
}
```

返回数据：

```
data:{
    money:"",
    teamMates:"",
    machine:"",
    material:"",
    house:""
}
```

4.2.7 接口 <https://www.ruaasdan.cn/api/dealInit>

作用：用户生产交易环节的初始化

方法：POST

发送数据：

```
data:{
    gameid:"",
    teamid:""
}
```

返回数据：

```
data:{
```

```
        price:"",
    Uptime:"",
}
```

4.2.8 接口 <https://www.ruaasdan.cn/api/ranking>

作用：获取排名信息

方法：POST

发送数据：

```
data:{
    gameid:"",
    teamid:""
}
```

返回数据：一个包含全部排名信息的列表。

4.2.9 接口 <https://www.ruaasdan.cn/api/dealOut>

作用：发送一个交易请求

方法：POST

发送数据：

```
data:{
    teamid:"",
    gameid:"",
    userid:"",
    obj:"",
    money:"",
    material:"",
    machine:""
}
```

返回数据：“message”为‘failed’表示交易发送失败，否则成功。

4.2.10 接口 <https://www.ruaasdan.cn/api/produce>

作用：向服务器发送生产请求

方法：POST

发送数据：

```
data:{
    userid:"",
    teamid:"",
    gameid:"",
    id:"",
    total:"",
}
```

```
        totalMoney:""
    }
}
```

返回数据：无，若成功直接表示生产成功，否则失败。

4.2.11 接口 <https://www.ruaasdan.cn/api/myProduceList>

作用：获取本队的生产记录

方法：POST

发送数据：

```
data:{
    teamid:"",
    gameid:"",
    userid:""
}
```

返回数据：一个包含本队全部生产记录的列表。

4.2.12 接口 <https://www.ruaasdan.cn/api/dealMessage>

作用：获取本队的交易记录

方法：POST

发送数据：

```
data:{
    teamid:"",
    gameid:""
}
```

返回数据：一个包含本队全部交易记录的列表。

4.2.13 接口 <https://www.ruaasdan.cn/api/retDeal>

作用：用于对交易信息进行回应

方法：POST

发送数据：

```
data:{
    Type:"",
    Number:"",
    gameid:"",
    teamid:"",
    userid:""
}
```

返回数据：“go”为‘yes’表示操作成功，否则表示失败。

4.2.14 接口 `wss://www.ruaasdan.cn/api/webSocketLogin`

作用：用于服务器通知最新消息

发送数据：`gamid + teamid + userid`

返回数据：`state, token`

其中 `state` 为 2 表示更新拍卖，为 3 表示有新的交易信息，为 4 表示更新生产交易环节结束时间，为 5 表示更新本队财富。`State` 为 3 表示进入下一环节。

5 服务器配置

5.1 部署方式

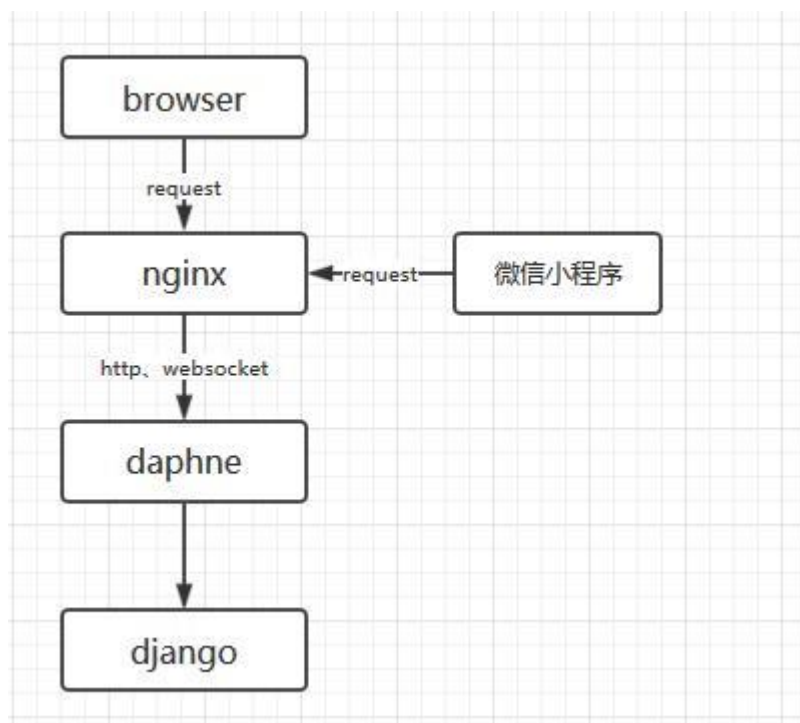
nginx 作为反向代理服务器

daphne 作为 channels 的 interface server （接口服务器）

redis 作为 channels 暂存 message 的数据库

5.2 基本架构

基本架构如图所示：



5.3 详细部署流程

5.3.1 django 后端部署

5.3.1.1 配置 settings.py

让 redis 作为 channel 的后端存储

```
CHANNEL_LAYERS = {
    'default': {
        'BACKEND': 'asgi_redis.RedisChannelLayer',
        'CONFIG': {
            'hosts': [('localhost', 6379)],
        },
        'ROUTING': 'api.routing.channel_routing',
    }
}
```

5.3.1.2 启动 Daphne

daphne 命令在你安装 channels 库的时候已经依赖安装到系统中，但是要增加一个 asgi.py 文件如下

```
import os
from channels.asgi import get_channel_layer

os.environ.setdefault("DJANGO_SETTINGS_MODULE", "backend2.settings")

channel_layer = get_channel_layer()
```

通过指令 `daphne -b 0.0.0.0 -p 8089 backend2.asgi:channel_layer` 启动 daphne

5.3.1.3 运行 worker 进程

`python manage.py runworker --threads 100`

5.3.2 Nginx 配置

```
location /api {

    proxy_pass http://0.0.0.0:8089/api;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";

    proxy_redirect      off;
    proxy_set_header    Host $host;
    proxy_set_header    X-Real-IP $remote_addr;
    proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header    X-Forwarded-Host $server_name;
}
```

Nginx 把请求反向代理到 daphne

5.3.3 管理员前端部署

使用 webpack 工具把 vue 前端文件打包成一个 dist2 文件夹,文件夹下会生成一个 index.html 文件,可以通过 nginx 直接部署。具体配置如下:

```
location / {  
    root /home/ubuntu/dist2;  
    index index.html index.html;  
}
```

6 小组人员

寇明阳负责后端（服务器端及数据库）的设计和测试

姚超负责 web 端（管理员端）及大屏展示的设计

黄权伟负责微信小程序端（用户端）的设计

李忠明负责服务器的部署