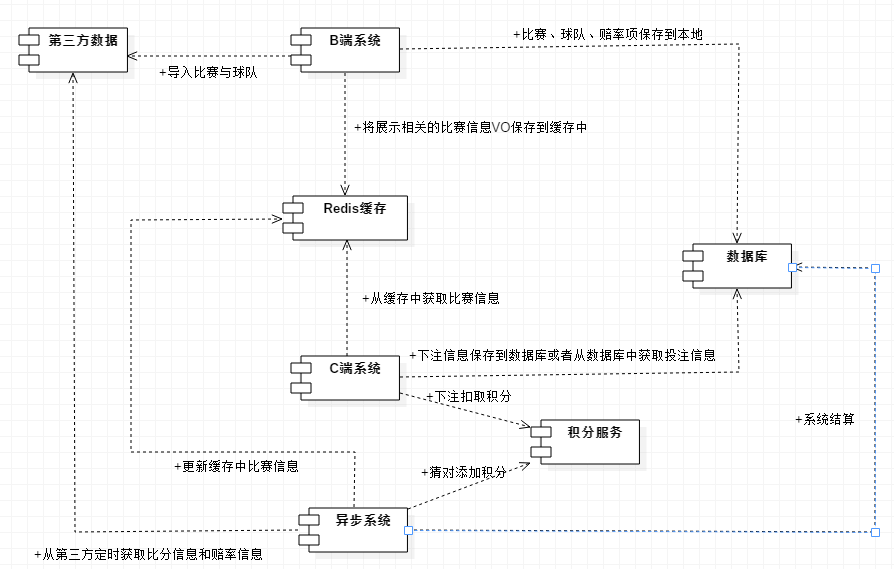
**系统设计：**



B端系统：导入球队信息、比赛信息，同时组装与比赛展示相关的VO保存到缓存中，key为该比赛本地的id，value为MatchVo对象。

C端系统：从缓存中获取信息来查看比赛列表和比赛详情；做下注操作将下注信息保存到数据库中，同时扣取积分。在下注的时候还会更新缓存中存放用户本场比赛的投注信息以及最近10条投注信息。

异步系统：定时的调用第三方API去刷新缓存中的比分、赔率和状态；如果三个值不会再变化的时候会去将数据库中的比赛、赔率项中的相关值更新；比赛结束之后会去做结算操作更新到数据库和积分服务。

**概要设计：**

**运营部分：**

1. **查看第三方所提供的比赛：**

**设计概要：**需要与第三方数据进行集成，调用其获取比赛列表的API来获取到一段时间内的所有比赛。

**API:** List<Match> getAllMatchesFromThridParty(String country)

1. **保存比赛：**

**设计概要：**首先通过第三方那边主队id和客队id从本地的guess\_team表中获取id作为guess\_match表中home\_team\_id和guest\_team\_id，将第三方那边获取到的比赛id做为guess\_match表中的external\_id，将第三方那边的比赛开始时间作为guess\_match表中的beginTime，将第三方那边获取到的比赛状态进行装换对应guess\_match表中的status（比赛状态：0代表未开始，1代表进行中，2代表已结束），将从第三方那边获取到的比分分别作为guess\_match表中的home\_score和guest\_score，deleteFalg为取默认值0.其他字段取默认值，之后将现有的胜负平三个投注项作为name，其他字段取默认值存入到guess\_bet\_item表中.将比赛信息、赔率信息、球队信息组成VO对象，以数据库id为key，VO为值存储在缓存中。

**API:** void batchSaveMatches(List<Match match>)

1. **查看比赛：**

**设计概要：**从本地数据库中获取到所指定时间的所有比赛对象。

**API:** Match getMatch(matchId)

1. **删除比赛：**

**设计概要：**从本地缓存和数据库中所指定的比赛批量删除。

**API:** void batchDeleteMatch(matchId)

1. **导入队伍：**

**设计概要：**调用第三方服务将队伍id作为external\_id，队名作为name的值，队徽作为logo的值将队伍信息保存到guess\_team表中。

**API:** void batchAddTeam(List<Team> team)

1. **查看队伍：**

**设计概要：**从本地数据库中获取到所有队伍对象。

**API:** Team getTeam(teamId)

1. **编辑队伍：**

**设计概要：**将用户新修改的对象先保存到缓存，之后再保存到数据库。更新缓存中数据该球队所参加的比赛的值。

**API:** void updateTeam(Team)

**客户部分：**

1. **查看比赛列表：**

**设计概要：**根据传入的beginDate，从redis缓存中获取到比赛日期为指定日期的所有比赛，如果该天没有比赛则取下一天有比赛的那天的数据。所有比赛按照比赛开始时间顺序排列。

**API：**List<MatchVo>listMatches(beginDate)

1. **查看比赛详情：**

**设计概要：**从缓存中获取这场比赛的详情；从缓存中获取用户的投注记录，如果没有则从数据库中获取，同时更新缓存；从缓存中获取这场比赛的最近10条投注记录，如果缓存中没有则从数据库中获取，同时更新缓存。

**API:** MatchVo getMatch(matchId)

1. **下注：**

**设计概要：**将用户下注信息保存到guess\_bet表中，更新用户本场比赛投注的缓存和用户最近10条投注记录的缓存，同时调用积分API扣除该用户的积分。

**API:** void doBet(Bet)

**异步任务：**

1. **刷新比赛状态和比分：**

**设计概要：**异步的任务会定时地去调用第三方的API获取比赛的状态以及当前的比分，并更新到redis中，如果比赛已经结束的话，会将比赛状态和比分更新到本地数据库中。当比赛状态为结束的时候，会触发系统结算功能。

**API:** void refreshMatch()

1. **刷新赔率：**

**设计概要：**异步的任务会定时地去调用第三方的API获取比赛的赔率，并存入到redis中，如果比赛已经开始的话，则会将最终的赔率更新到本地数据库中。

**API:** void refreshBet()

1. **系统结算：**

**设计概要：**如果没中奖，则奖金为0；如果中奖，则奖金为注数\*赔率并取整更新下注表，同时给用户添加上所中积分。

**API:** void calculate();

**缓存设计：**

**比赛缓存：**

**概述：**在运营导入比赛的时候将加入到本地数据库的时候会将该比赛放入到缓存中，在此有两个缓存，一个是String类型，key为id，value为MatchVo；另一个缓存是zset类型，key为beginDate，score为beginTime+id，value为id。详情页面的时候直接从String类型中的缓存中获取，列表页的时候传入beginDate，获取到该beginDate中的所有排好序的元素。另外，当用户删除比赛的时候会将这场比赛从这两个缓存中删除掉。

**Redis数据类型：**（1）String，key为id，value为MatchVo。（2）zset, key为beginDate，score为beginTime+id，value为id。

**有效期：**这场比赛开始时间之后的两个月。

**用户本场比赛投注缓存：**

**概述：**在用户投注的时候会将其本次投注信息追加到缓存中，再存入数据库。在取的时候，直接从缓存中取出来。

**Redis数据类型：**list，key为userId+matchId，元素为Bet对象。

**有效期：**这场比赛开始时间之后的两个月。

**最近10条投注记录的缓存：**

**概述：**在用户投注的时候会将其投注信息push到缓存中，如果缓存中元素个数超过10个则pop最后那一个。在取的时候，直接从缓存中取出来。

**Redis数据类型：**list，key为lastestTenBet\_matchId，元素为LatestRecord。

**有效期：**这场比赛开始时间之后的两个月。

**API设计**

**获取比赛列表：**

**协议：**HTTP/GET

**URL：**待定

**输入：**date, minSize, direction

**输出：**List<MatchVo> 获取当前日期的所有比赛，如果当前日期没有比赛则向前或者向后获取最近一天有比赛的该天所有比赛。

**获取比赛详情：**

**协议：**HTTP/GET

**URL：**待定

**输入：**matchId

**输出：**MatchDetailVo

**下注：**

**协议：**HTTP/POST

**header：**X-Auth-Token

**URL：**待定

**输入：**Bet

**输出：**无

**查看我的下注记录:**

**协议：**HTTP/GET

**URL：**待定

**输入：**无

**输出：**List<MyBetRecord>

**附：API中结构解析**

**MatchVo的结构：**

{

homeTeamName:”恒大”,

homeTeamLogo:”http://abc.tupian.com/hengda.png”,

guestTeamName:”鲁能”,

guestTeamLogo:”http://abc.tupian.com/hengda.png”,

beginTime:” 2016-01-09 17:00:00”,

homeScore:”0”,

guestScore:”0”;

homeOdd:”1.25”,

homeDraw:”1.9”,

guestOdd:”2.10”

status:”0”

}

**MatchDetailVo结构：**

{

matchVo：MatchVo，

betMatchRecord：BetMatchRecord

latestRecord：LatestRecord

}

**BetMatchRecord结构：**

{

betItemName：“主胜”，

odds：“2.13”，

number：“100”，

status：“未开奖”，

time：“2016-01-09 17:00:00”

}

**LatestRecord结构：**

{

phoneNumber：“1512\*\*\*\*341”，

betItemName：“主胜”，

number：“5000”，

time：“2016-01-09 17:00:00”

**}**

**Bet结构：**

{

matchId：“23”，

betItemId：“132”，

betPoints：“150”，

}

**MyBetRecord结构：**待定

**服务调用:**

