# 软件架构设计文档

团队名称：TBD

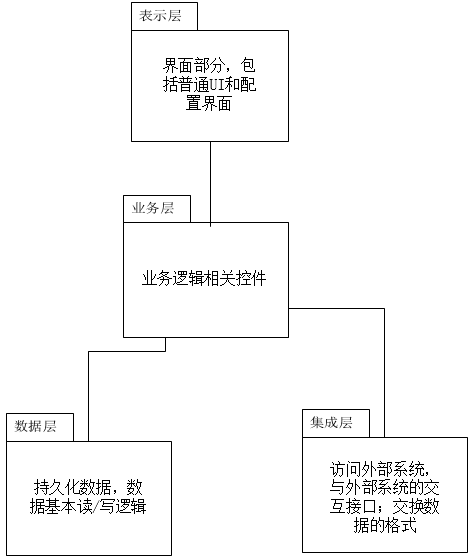
## 文档更新记录：

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **作者** | **版本描述（应该说明做了哪些修改及修改的标识等）** | **日期** |
| V1.0 | 陈元亮 | 新建文档，发布到服务器SVN上 | 2015/03/10 |
| V2.0 | 李嘉麒 | 迭代二更新 | 2015/04/13 |

## 引言：

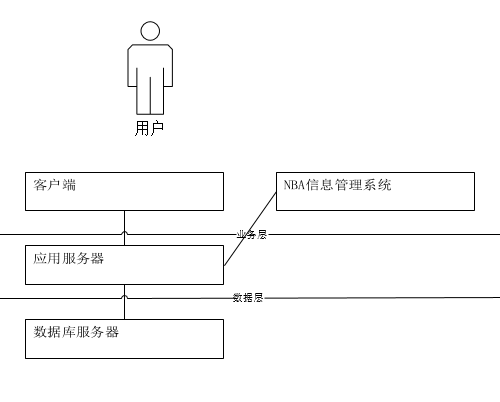
|  |  |
| --- | --- |
| 内容和说明 |  |
| 1.编写目的 | 本文档提供NBA数据分析系统的软件架构概览，采用若干架构视图描述系统的不同方面，以便表示构造系统所需要的重要架构决策。 |
| 2.对象和范围 | 本文档的读者是( TBD )团队内部的开发和管理人员，参考了RUP的《软件架构文档模板》，用于指导下一循环的代码开发和测试工作。 |
| 3.参考文献 | 《软件需求规格说明书》  《软件架构文档模板》 |
| 4.名词和术语 | UI BL Data |

## 系统的分层架构:



系统划分为以下4个逻辑层次。

1. 表示层：用于前台界面展示和配置的层次。
2. 业务层：包含业务控制和逻辑的层次。
3. 数据层：定义和存储系统相关数据的层次。
4. 集成层：定义和集成与外部系统交互接口的层次。



系统可以部署在下面3个物理层次。

1. 访问层：用于用户访问系统的层次
2. 业务层：部署业务控制和逻辑的层次
3. 数据层：部署和存储系统中的相关数据的层次

## 系统架构设计如下：

系统架构中的对象分为5类：

1. UI对象，负责处理处理系统数据的展现与用户之间的人机交互
2. BLService对象：负责提供UI服务的抽象接口，获取从数据端组装好的数据。
3. BLController对象：负责实现BLservice所申明的接口，分配任务给其他controller
4. DataService对象:负责提供给BL层服务的抽象接口，从数据库中读取数据
5. DataController对象：负责实现DataService所申明的接口

系统中的组件和组件接口：

表示层

UI

业务层

BL

BLService

数据层

DataService

Data

## 5. 接口规范

### 5.1 BL模块接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| **BLController.getALLPlayers()** | 语法 | Public ArrayList<PlayerVo>  **getALLPlayers** (); | |
| 前置条件 | 已创建一个BLController对象并完成init() | |
| 后置条件 | 调用BLController的**getALLPlayers**方法，并返回所有的Players全部信息 | |
| **BLController.getALLTeams()** | 语法 | Public ArrayList<TeamVo>  **getALLTeams** (); | |
| 前置条件 | | 已创建一个BLController对象并完成init() |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 后置条件 | | | 调用BLController的**getALLTeams**方法，并返回所有的Teams全部信息 |
| **BLController.getALLMatches()** | | 语法 | public ArrayList<MatchVo> **getALLMatches** (); | |
| 前置条件 | 已创建一个BLController对象并完成init() | |
| 后置条件 | 调用BLController的**getALLMatches**方法，并返回所有的Matches全部信息 | |
| **BLController.getProgress()** | | 语法 | Public int **getProgress();** | |
| 前置条件 | 已创建一个BLController对象并init() | |
| 后置条件 | 在init后，查看已加载的进度，从0~9 | |
| **BLController.init()** | | 语法 | Public void **init();** | |
| 前置条件 | 已创建一个BLController对象 | |
| 后置条件 | 计算所有的数据并且开启线程读图，同时更改progress，知道其为9 | |
| **BLController.isMatchChanged()** | | 语法 | Public boolean **isMatchChanged()** | |
| 前置条件 | 已创建一个BLController对象并完成init() | |
| 后置条件 | 调用data层的**isMatchChanged()** | |
| **BLController.getResult()** | | 语法 | Public ArrayList<Object> getResult(BLParameter parameter) | |
| 前置条件 | 已创建一个BLController对象 | |
| 后置条件 | 返回根据命令行参数所要求的答案 | |
| **BLController.getAllianceAverageData()** | | 语法 | Public double[]**getAllianceAverageData();** | |
| 前置条件 | 已创建一个BLController对象并完成init() | |
| 后置条件 | 得到联盟(30个球队加起来)的平均数据（场均得分，场均篮板，场均助攻，罚球%，三分%） | |
| **BLController.getMostImprovedPlayer()** | | 语法 | **Public ArrayList<Player> getMostImprovedPlayer(String field,int num)** | |
| 前置条件 | 已创建一个BLController对象并完成init() | |
| 后置条件 | 根据field的数据得到前5个进步最快的球员 | |
| **BLController.getTodayKingPlayer** | | 语法 | Public ArrayList<todayPlayer> **getTodayKingPlayer(String date,String field,int num)** | |
| 前置条件 | 已创建一个BLController对象并完成init() | |
| 后置条件 | 根据field得到当天的最佳球员 | |
| **BLController.getSeasonKingPlayer** | | 语法 | Public ArrayList<Player> **getSeasonKingPlayer(String field,int num)** | |
| 前置条件 | 已创建一个BLController对象并完成init() | |
| 后置条件 | 根据field得到当赛季的最佳球员 | |
| **BLController.getTodayMatches** | | 语法 | Public ArrayList<Match> **getTodayMatches(String date)** | |
| 前置条件 | 已创建一个BLController对象并完成init() | |
| 后置条件 | 返回当天的比赛 | |
| **BLController.getSeasonKingTeam** | | 语法 | Public ArrayList<Team> **getSeasonKingTeam(String field,int num)** | |
| 前置条件 | 已创建一个BLController对象并完成init() | |
| 后置条件 | 根据field得到当赛季最佳球队 | |
| **BLController.getTeamsByMatch** | | 语法 | Public Team[] **getTeamsByMatch(Match match)** | |
| 前置条件 | 已创建一个BLController对象并完成init() | |
| 后置条件 | 得到某场比赛的主客球队 | |
| **BLController.getTeamByName** | | 语法 | Public Team **getTeamByName(String teamName)** | |
| 前置条件 | 已创建一个BLController对象并完成init() | |
| 后置条件 | 根据球队名字得到球队引用 | |

|  |
| --- |
| **需要的服务（需接口）** |

|  |  |
| --- | --- |
| **DataService. getALLPlayers** | 从数据端中得到所有的球员的部分信息 |
| **DataService. getALLTeams** | 从数据端中得到所有的球队部分信息 |
| **DataService. getALLMatches** | 从数据端中得到所有的比赛部分信息 |

### 5.2 Data模块接口规范

|  |  |  |  |
| --- | --- | --- | --- |
| **提供的服务（供接口）** | | | |
| **DataService.getALLPlayers()** | 语法 | Public ArrayList<PlayerVo>  **getALLPlayers** (); | |
| 前置条件 | 已创建一个DataController对象 | |
| 后置条件 | 调用DataController的**getALLPlayers**方法，并返回全部的Players部分信息 | |
| **DataService.getALLTeams()** | 语法 | Public ArrayList<TeamVo>  **getALLTeams** (); | |
| 前置条件 | | 已创建一个DataController对象 |
| 后置条件 | | 调用DataController的**getALLTeams**方法，并返回全部的Teams部分信息 |
| **DataService.getALLMatches()** | 语法 | | public ArrayList<MatchVo> **getALLMatches** (); |
| 前置条件 | | 已创建一个DataController对象 |
| 后置条件 | | 调用DataController的**getALLMatches**方法，并返回全部的Matches部分信息 |

|  |  |  |
| --- | --- | --- |
| **DataService.isMatchChanged()** | 语法 | Public boolean isMatchChanged(); |
| 前置条件 | 已创建一个DataController对象 |
| 后置条件 | 返回是否有match数据的更新，如果有更新，则自动更新相应数据再返回true |
| **DataService.readAllImages()** | 语法 | Public void readAllImages(); |
| 前置条件 | 已创建一个DataController对象 |
| 后置条件 | 开启线程对图片进行读取，同时更改BLController.progress值 |