UML Assignment3

Software Requirement Specification

同济大学线上运动社区系统设计

组员： 1951593 李航宇 1952728 杨梓浩 1950072 郑柯凡

1952897 胡启云 1951328 曹峰源

指导老师： 刘岩

目录

1 介绍 3
1.1 项目目的 3
1.2 项目应用范围 3
1.3 项目进展 3
1.4 目标读者和阅读建议 4
1.5 平台及框架的应用描述 4
2 系统架构改进 5
2.1 平台相关架构 5
2.1.1 逻辑架构 5
2.1.2 开发架构 6
2.1.3 部署架构 7
2.2 子系统与接口 8
2.2.1 表现层 9
2.2.2 业务层 10
2.2.3 数据层 11
2.2.4 第三方接口 14
2.3 第三方接口规范描述 14
2.3.1 每日饮食摄入信息 API 规范 14
2.3.2 地图信息 API 规范 16
2.4 子系统接口规范 18
2.4.1 用户约球信息管理 19
2.4.2 场地信息管理 20
2.4.3 排名子系统 21
3 设计机制 22
3.1 数据持久化机制 22
3.2 安全机制 24
网页安全模型 24
数据备份与恢复 25
4 用例实现 26
4.1 约球 26
4.2 场地预定 28
5 架构风格与设计决策 29
5.1 架构风格 29
5.2 设计模式 29
5.3 主要设计决策 30
5.3.1 持久化机制的选择 30
5.3.2 读写分离 30
6 原型进展 31
6.1 前端原型 31
6.2 后端原型 35
7 遗留问题 36
8 项目反思 37

9 团队分工 39

1 介绍

1.1 项目目的

随着人们生活水平的逐渐提高，人们对于健康运动的需求越来越大。为满足同济师生对于运动、约球的需求，本次项目目的是设计完成一个同济大学线上运动社区----济动。主要用于实现约球，场地预定，个性化定制，排行榜，信息发布等功能。旨在搭建一个专属于同济的运动社区平台，更贴切地服务在校师生，让校园的运动文化迸发出蓬勃生机。

1.2 项目应用范围

济动----线上社区管理系统是一个基于 Web 的管理系统。其目的是实现约球、个性化定制、运动社交、信息发布的一体化。用户可以使用小程序访问系统。该系统的场景主要包括约球信息的发布、针对用户运动目标的个性化定制、大型体育赛事、场地信息的发布与更新。此外，该在线系统还具有以下特点：在网络环境中运行、对师生学号/工号的验证、拥有集中的数据库等。

1.3 项目进展

在之前的报告中，我们初步完成了系统结构以及分析模型的设计，并且制作了完整的用户使用界面，通过类图展示了不同子系统中类的关系，通过时序图反映了功能实现的时序关系。

在本次报告中，我们绘制了更加具体的逻辑架构、开发架构以及部署架构的模型图，提供了各层子系统中用到的接口及第三方 API 与其规范。我们还给出了分析机制，用例实现。最后，完善了系统原型。

这一次作业中，我们主要的工作有：

1. 确定了济动系统具体的逻辑架构、开发架构与部署架构，并且分析了可能用到的实际技术有哪些。其中逻辑架构采用包图展现。
2. 完善了我们子系统的接口实现与接口规范，并且提供了要用到的第三方 API 的使用细节与使用规范。
3. 采用了数据持久化机制、安全机制等设计机制。数据持久化机制帮助我们的系统服务避免直接对数据库进行操作，提升系统效率与可用性，数据不易丢失；安全机制包含了数据传输安全等一系列安全措施，提升我们系统的安全性。
4. 展示了约球与预约场地的用例实现。
5. 研究了我们系统的架构风格、设计模式以及设计决策的选择。
6. 实现了项目前后端的部分内容，编辑并完善了用户使用界面。

1.4 目标读者和阅读建议

本文档的面向读者为“济动”运动系统的用户以及开发人员。如果您想要对本系统有一个大致的了解，建议您阅读第 1 部分即本项目的总览。若想要了解架构模式，请阅读第 2 部分中的几种架构方式以及接口的设计。第 3 部分，我们给出了设计机制，第 5 部分我们提供了架构风格与设计决策。读者可以在第 3 章与第 5 章了解我们对系统进行优化的想法与思路。要想对我们系统具体的使用有所了解，可以阅读第 4 章两个核心用例：约球与预约场地的用例展示，还可以阅读第 6 部分我们编辑的用户使用界面。最后几个部分我们给出了遗留问题以及团队反思与贡献情况，对近期项目研究的开展进行总结。

1.5 平台及框架的应用描述

在本项目的开发实践中，我们主要还是使用了分层架构的思想，将完整的系统分成几个独立的功能层。同时我们也结合了微服务架构的风格，将系统提供的服务划分成多个功能专一的服务，由各个子系统实现，使得各微服务的实现和部署更加自由，重用性更高，整体的系统也更加稳定。以下将按照自顶向下的顺序对我们整体的技术框架作简要的介绍。

展示层，即系统的前端，根据网页端和小程序端的不同分别使用 vue.js 和 mpVue 框架进行开发，同时使用 Node.js、NPM、Webpack 工具构建完整的前端页面，并使用 AJAX 和 WebSocket 构造实时更新的交互式页面，给予用户更好的浏览体验。同时，所有的 API 基于 Rest 风格规范，数据统一被转换成 JSON 格式发送至接口层。

接口层，为了保证系统的稳定高效，使用 Nginx 集群作为反向代理，保护后台原始资源，同时加速传输。同时，使用 Spring Cloud Gateway 作为网关，提供了过滤数据、统一接口、熔断机制的触发等功能，进一步保障了后台服务的安全性。并且提供了基于 WebJSONToken 标准的 SSO 单点登录机制，一次登录就可以访问所有相互信任的系统。

服务层，先使用 Spring Security 鉴别访问用户的权限，进一步提升安全性，并使用 Spring Cloud Alibaba 提供的 Nacos、Sentinel、Dubbo、Seata 组件构建微服务，并分别提供服务配置、流量控制、分布式系统通信和分布式系统事务处理的功能。阿里云的日志服务框架能够对服务进行记录。而 SkyWalking 在以上服务中均注入了监控探针，调用链路进行监控追踪并提供可视化的通知，能使开发人员清楚得知道系统的历史运行状况，更好地进行系统维护和错误定位。该层组件及上一层中的网关均部署在容器中，并由 Kubernetes 统一管理维护。

数据持久化层主要实现的是数据的存储和访问，选择 Mysql 配合 Hibernate 作为数据库和持久化框架。Mysql 具有轻量化、性能高的特点，且配合 ElasticSearch 可以更方便地实现读写分离，而 Hibernate 在对象的维护和访存方面更具优势。

服务器层主要选用 SpringBoot 作为开发框架，其封装好了一些常用套件，能实现快速开发。

2 系统架构改进

2.1 平台相关架构

2.1.1 逻辑架构

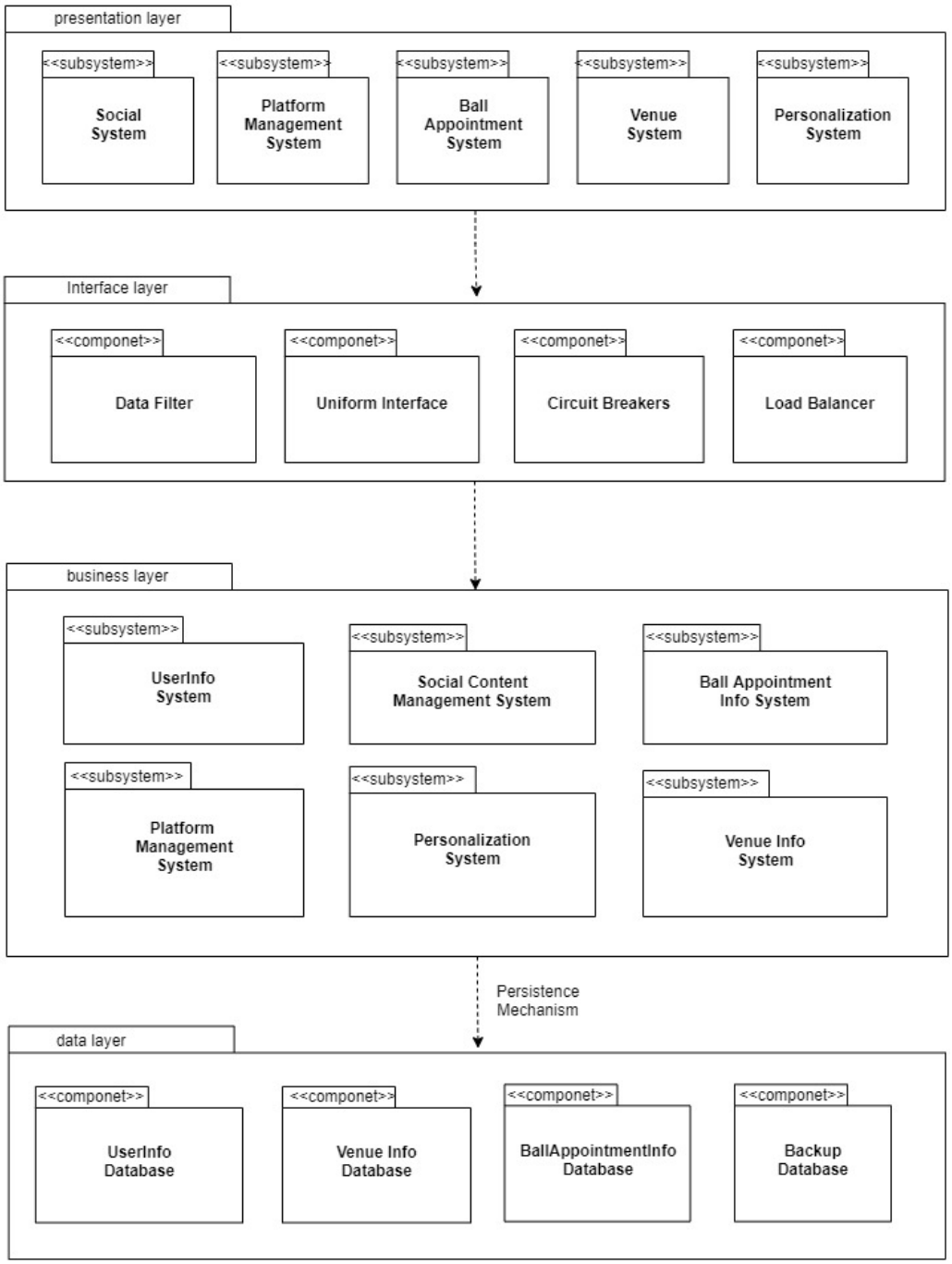


图 1：逻辑架构

2.1.2 开发架构

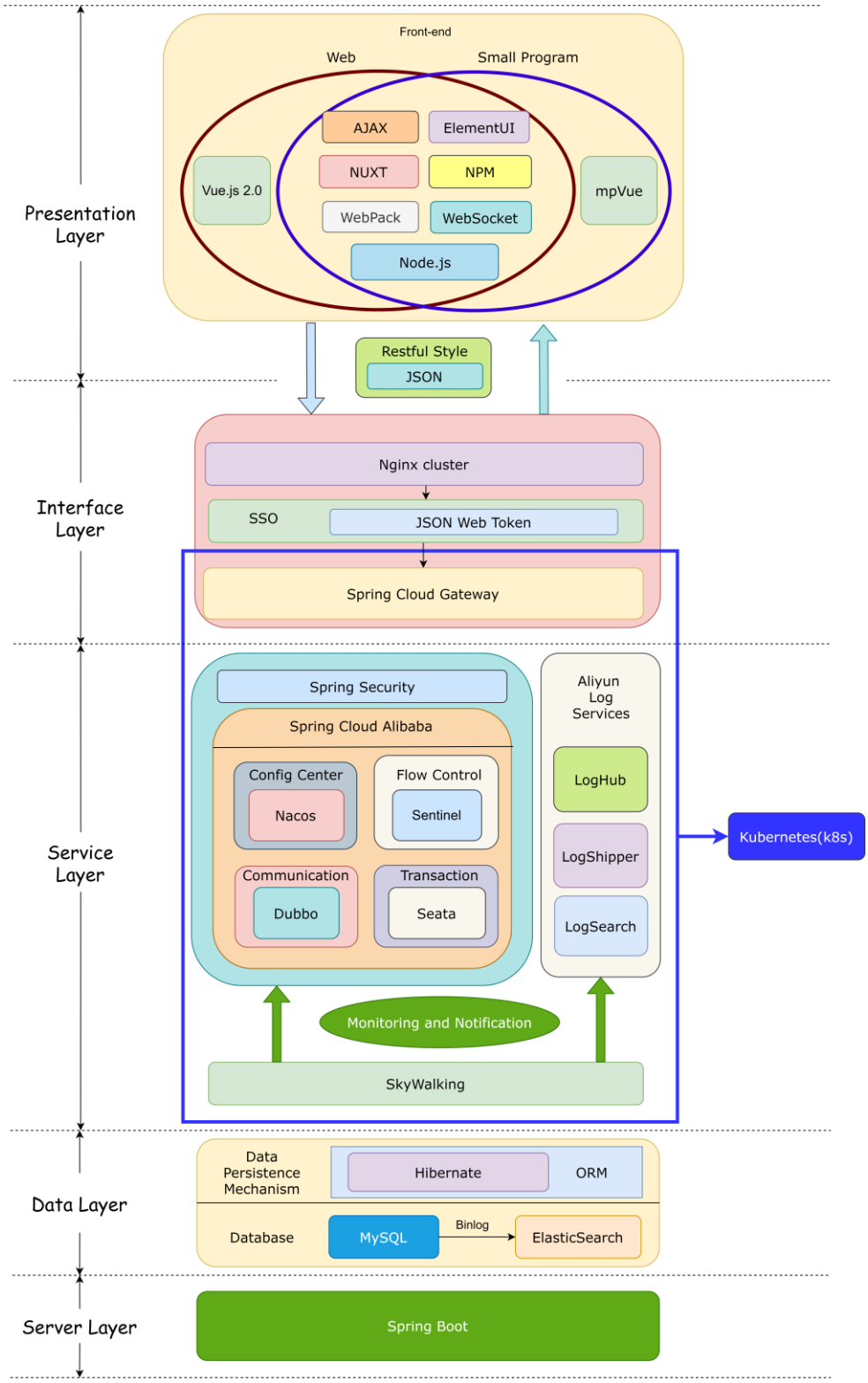


图 2：开发架构

2.1.3 部署架构

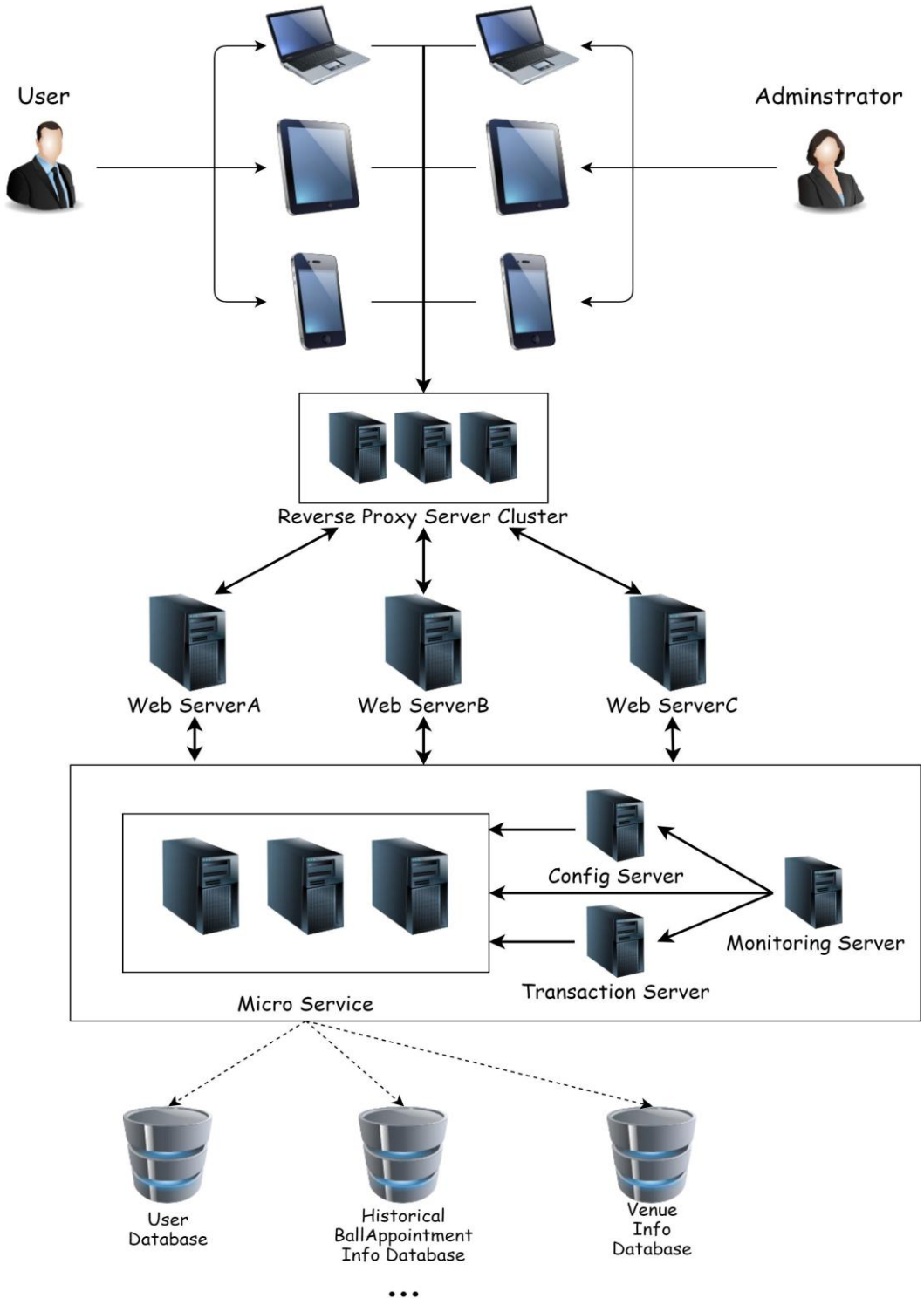


图 3：部署架构

2.2 子系统与接口

表 1： 各层系统划分

business layer	个人信息系统
	社交内容管理系统
	平台信息管理系统
	运动信息管理系统
	排行榜信息管理系统
	场地信息管理系统
presentation layer	社交系统
	平台信息系统
	约球系统
	排行榜信息系统
	评价信用积分系统
	场地信息系统
	个性化定制系统
data layer	约球信息数据库
	用户信息数据库
	场地信息数据库

2.2.1 表现层

社交系统
接口类：ISocialCommunicationSystem
接口函数：
1. EditMessage():void
2. EditMoment():void
3.SendMessage(Content newMessage,string UID):bool　//注:通过审核, 发送成功后返回 true
4.ReleaseMoment(Content newMoment):bool

平台信息系统
接口类：IPlatformInfoSystem
接口函数：
1.EditNotificaiton():void
2.SendNotification(Notification newNotificaiton):void
3.SubmitVenueStatusUpdate(VenueInfo newVenueInfo):void

约球系统
接口类：IBallAppointmentInfoSystem
接口函数：
1. selectAppointmentType(bool appointmentType):void //注：选择约球种类(私发、公共)
2. sendPrivateAppointment(FilteredCondition condition):void
3.selectWantList(PersonInfo List filteredUserList):void //注：在系统返回的过滤名单中选择人员，再私发约球信息
4.sendPublicAppointment():void
5.acceptBallAppointmentInfo(string AppointmentID):bool

排行榜信息系统
接口类：　IRankingInfoSystem
接口函数：
1. sendRankingViewRequest()：RankingList

评价信用积分系统
接口类：ICreditEvaluateSystem
接口函数：
1. sendCreditEvaluationScore(string UID,int evaluateScore)：void

场地信息系统
接口类： IVenueInfoSystem
接口函数：
1. displaySpecificVenueSchedule(string VID):VenueSchedule
2. displayVenueInfo(string venueType):VenueInfo list
3. submitVenueReservation(VenueReservaiton venueReservation):void

个性化定制系统
接口类：IPersonalizationSystem
接口函数：
1. setSportsTarget(string UID,SportTarget target):void
2. viewPlan(string UID):Plan

2.2.2 业务层

个人信息系统
接口类:IUserInfoManageSystem
接口函数：
1.getUserInfo(string UID):userInfo
2.updateUserInfo(string UID,UserInfo newUserInfo):void
3.creditCheck(string UID)： userCreditInfo
4..updateUserCredit(string UID,UserCreditInfo newUserCreditInfo):void
5.getFilteredPersonnelInfo(FilteredCondition condition):PersonInfo List

社交内容管理系统
接口类：ISocialContentManagetSystem
接口函数：
1.auditContent(Content newContent):bool
2.sendMoment(Content newContent):void
3.sendMessage(Content newContent,string UID)

平台信息管理系统
接口类：IPlatformManageSystem
接口函数：
1. PostNotificaiton(Notification newNotificaiton):void
2. PostVenueStatusInfo(VenueInfo newUpdateInformation):void

运动信息管理系统
接口类：IBallAppointmentInfoManageSystem
接口函数：
1. sendAppointmentInfo(string UID,bool appointmentType):void //注：appointmentType 为 false 时代表私发约球信息，为 true 时代表群发约球信息
2. queryMatchInfo(string sportsKind,string sportsTime)：PublicAppointmentInfo List
3. deleteAppointmentInfo(string UID,string appointID):void
4. modifyAppointmentInfo(string UID,string appointID,PublicAppointmentInfo newPublicAppointmnetInfo):void

排行榜信息管理系统
接口类：IRankingListManageSystem
接口函数：
1. updateRankingList(UserInfo userSportTime):void //注：排行榜是按照运动时间进行排名
2. createRankingList(UserInfo List userSportTimeList):RankingList
3. getRankingList():RankingList

场地信息管理系统
接口类： IVenueInfoManageSystem
接口函数：
1. getSpecificVenueInfo(string VID):venueSchedule
2. updateVenueInfo(VenueReservation venueReservationInfo):void

2.2.3 数据层

约球信息数据库
接口类：BallAppointmentInfoDatabaseSystem
接口函数：
1. queryPublicAppointmentInfo():AppointmentInfo
2. storeAppointmentInfo(AppointmentInfo appointmentInfo):void

用户信息数据库
接口类： UserInfoDatabaseSystem
接口函数：
1. queryUserInfo(string UID):UserInfo
2. updateUserInfo(string UID,UserInfo newUserInfo):void
3. queryUserPlanInformation(string UID):Plan
4. updateUerPlanInformation(string UID,Plan newPlan):void

场地信息数据库
接口类:
场地信息系统 VenueInfoDatabaseSystem
接口函数:
1. queryVenueInfo(string VID):VenueSchedule
2. storeVenueReservationInfo(VenueReservation venueReservationInfo):void
3. updateVenueReservationInfo(VenueReservation venueReservationInfo):void

注：1.由于我们系统用到了读写分离的设计，在模型层中的接口函数，如果对数据库进行增删改的操作，目标数据库是 mysql 数据库，如果需要进行查询操作，则用到 elasticsearch。
3. 如下给出各子系统中出现类的具体说明

表 2：类名及其数据成员

类名	数据成员
VenueSchedule	+ venueID:string + venueType:string + venueLocation:string + venueOpeningHours:string + venueUse:bool[ ]
VenueResvation	+ userID:string + userName:string + venueID:string + venueType:string + venueLocation:string + reservationDate:Enum + reservationStartTime:Enum + reservationEndTime:Enum
PublicAppointmentInfo	+ appointmentID:string + userID:string + userName:string + sportKind:string + sportTime:string
AppointmentInfo	+ appointmentID:string + userID:string + userName:string + sportKind:string + sportTime:string
UserInfo	- userID:string - userPassword:string - userName:string - major:string - gender:bool - grade:string - credit:float

	- sportTime:string + userAge:int + userWeight:int + userHerght:int + userContactWay:string
RankingList	+ userID:string + userName:string + sportTime:string + Ranking:int
Content	+ userID:string + imageUrl:string + text:string
Notification	+ managerID:string + notificationID:string + notificationTitle:string + notificationText:string + notifactionTime:string
VenueInfo	+ venueID:string + venueType:string + venueLocation:string + venueOpeningHours:string + venueSchedule:VenueSchedule
FiliteredCondition	+ userMajor:string + userGender:bool + userGrade:Enum
UserCreditInfo	+ userID:string + userRating:int + userEvaluation:string + userAppointmentNumber:int + userDefaultNumber:int
SportTarget	+ sportLevel:Enum + sportTarget:Enum
Plan	+ userID:string + sportPlan:string

2.2.4 第三方接口

每日饮食摄入 API
接口类：IDietIntakeInfo
接口函数：
getDietIntakeInfo(float height,float weight,Target targetInfo):JSON
每日运动安排 API
接口类：ISportInfo
接口函数：
getSportInfo(Plan planInfo):JSON

地图 API
接口类: IMapInfo
接口函数:
getVenueLocation(string venueName):JSON

2.3 第三方接口规范描述

2.3.1 每日饮食摄入信息 API 规范

在查询了与饮食摄入相关信息的 API 资料后，我们设计了一个接口，该接口介于个性化管理系统与外部服务之间，如下图所示。

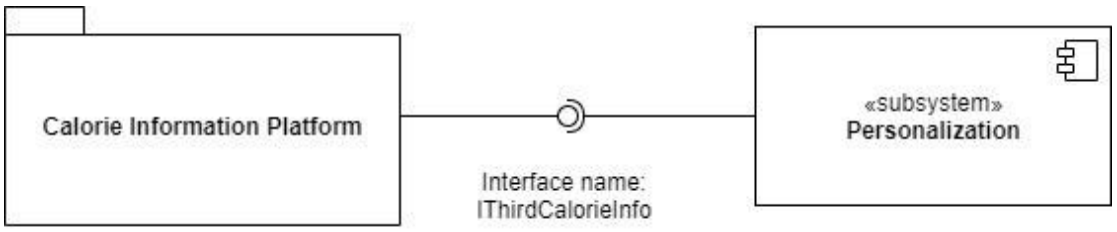


图 4：卡路里摄入信息接口

当有用户调用个性化系统，并上传自己的身高、体重、运动目标时，个性化管理系统就会调用第三方网站提供的接口，同时对数据进行处理，更新数据库。

在本系统的设计中，我们决定采用聚合数据上所提供的接口。根据接口文档，我们需要提供以下的几个参数:

请求参数说明：

名称	必填	类型	说明
key	是	string	在个人中心->我的数据,接口名称上方查看
sex	否	int	性别,1:男 2:女, 默认1
height	是	number	身高(CM), 支持最多1位小数; 如: 178
weight	是	number	体重(KG), 支持最多1位小数; 如: 67.8
age	是	int	年龄
level	是	int	运动量等级: 1：无运动习惯者/久坐族; 2：轻度运动者/每周1至3天运动; 3：中度运动者/每周3至5天运动; 4：激烈运动者/每周6至7天运动; 5：超激烈运动者/体力活工作/每天训练2次

图 5：参数列表

其中的 **key** 可以直接设置为固定值。

sex 和 **age** 直接从我们的个人信息数据库中调取。

而 **height**，**weight** 和 **level** 则需要根据我们个性化管理系统进行输入，获取用户的以上信息。

在发送请求后，该接口会返回一个 JSON 文件，其中就包含了我们需要的每日摄入卡路里值。返回参数如下所示：

返回参数说明：

名称	类型	说明
error_code	int	返回码
reason	string	返回说明
result	string	返回结果集

图 6：返回 JSON

因此我们对 IThirdCalorieInfo 接口设置如下：

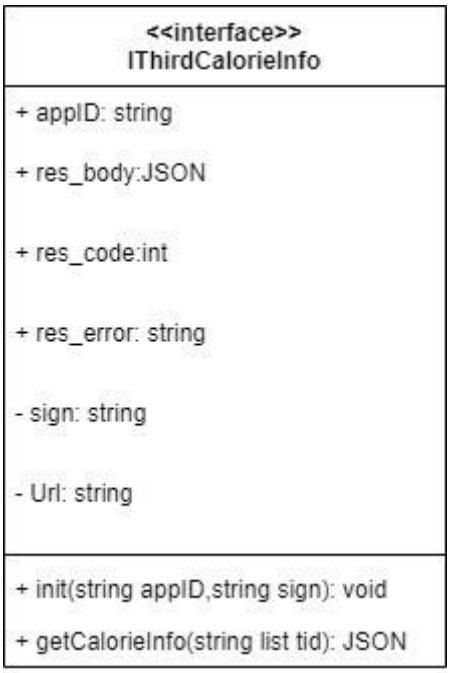


图 7：IThirdCalorieInfo 接口设计

该接口的六个变量，appID,url,sign 为所需要调用的服务 ID 和地址及电子签名，由服务商给出，res_body 为返回的信息体，res_code 和 res_err 为返回代码和错误信息。里面的两个方法，Init 函数用来初始化变量，而 getCalorieInfo 则是根据传入的 tid 数组，从外部调取对应的 JSON 数据。

2.3.2 地图信息 API 规范

在场馆查询时，为了帮助同学更快得定位场馆位置，我们需要从外部获取场馆的地图位置信息。我们选定高德地图作为我们的数据源。该接口位于场地信息管理系统和外部系统之间，如下图所示：

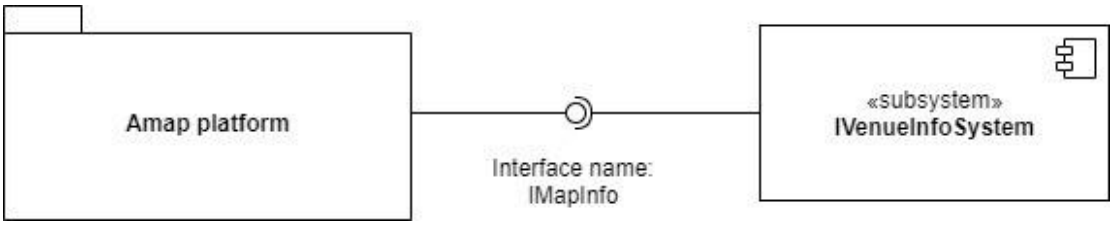


图 8：地图信息接口

当有用户需要进行场馆预约时，则会调取该 API，从而获取相应场馆的照片信息。根据该平台所提供的接口文档，调取接口需要如下的参数：

请求参数　　查看加密签名方式

参数名称	类型	默认值	示例值	必须	描述
ds_id	String			是	数据源
labels	String		朝阳公园,2,...	是	标签。使用规则见labels详细说明（http://lbs.amap.com/api/webservice/guide/api/staticmaps），标签最大数10个
location	String		116.4814...	是	地图中心点坐标。规则：经度和纬度用","分隔 经纬度小数点后不得超过6位。
markers	String		mid,0xFF0...	否	标注。使用规则见markers详细说明（http://lbs.amap.com/api/webservice/guide/api/staticmaps），标注最大数50个
paths	String		10,0x000...	否	折线。使用规则见paths详细说明（http://lbs.amap.com/api/webservice/guide/api/staticmaps），折线和多边形最大数4个
scale	String		1	否	1:返回普通图； 2:调用高清图，图片高度和宽度都增加一倍，zoom也增加一倍（当zoom为最大值时，zoom不再改变）。
size	String		750*300	否	图片宽度*图片高度。最大值为1024*1024
traffic	String		0	否	交通路况标识。底图是否展现实时路况。 可选值： 0，不展现；1，展现。
zoom	String		10	否	地图缩放级别:[1,17]

图 9：参数列表

其中的 **ds_id 和 labels** 我们设置为固定值.
location 根据我们的另一个接口获得地理编码信息。
size 根据我们查看场馆的数量设置相应的值。
最后我们返回的是图片编码值，具体如下所示：

返回参数　　查看公共返回参数

参数名称	类型	示例值	描述
img_base64	String	图片base64编码	图片base64编码

图 10：返回参数

综上，我们对该接口设计如图：

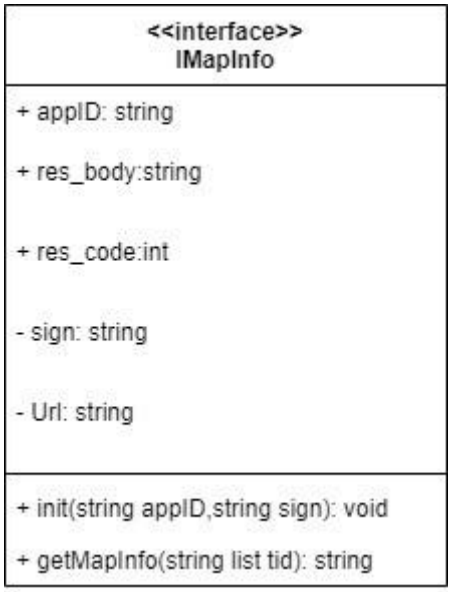


图 11：IMapInfo 接口设计

该接口的五个变量，appID,url,sign 为所需要调用的服务 ID 和地址及电子签名，由服务商给出，res_body 为返回的信息体，res_code 为返回代码。里面的两个方法，Init 函数用来初始化变量，而 getMapInfo 则是根据传入的 tid 数组，从外部调取对应的图片编码数据。

2.4 子系统接口规范

由于本系统设计的子系统与接口较多，就不在此罗列。在此部分中，仅以控制层中的运动信息管理系统为例，展示其接口的具体访问方式、具体方法输入输出以及可能设计的一些数据类的定义。

以下三个接口类中，如无特殊注明，其方法均为同步调用。

2.4.1 用户约球信息管理

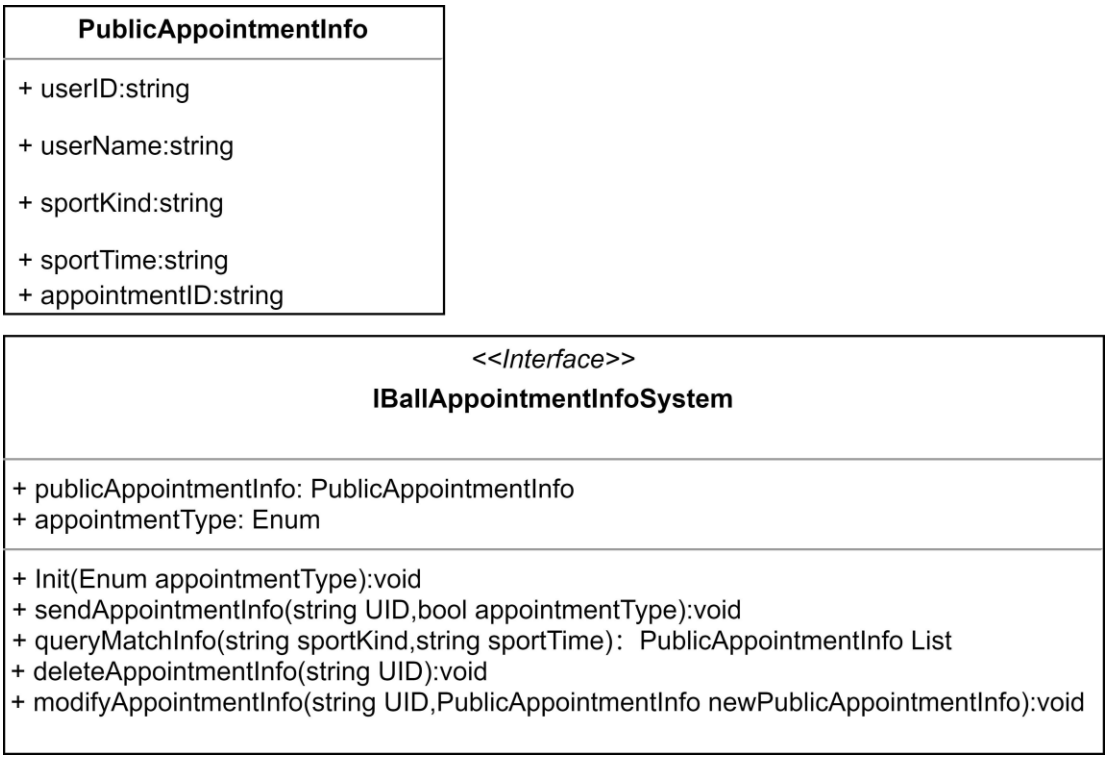


图 12：用户信息管理接口

下面展示的是运动信息管理系统中的几个接口，第一个是 IBallAppointmentInfoSystem 系统，包含数据类有 PublicAppointmentInfo，接口函数操作有初始化、发送、查询、删除和修改，使用方法是由接口层的约球系统调用，实现具体的业务逻辑。

需要特殊说明的是，sendAppointmentInfo()分为两种情况，当参数中 bool 型的 appointmentType 值为 false 的时候，将约球信息发送到公共的大厅界面，为 true 时，选择将约球信息发送给个人用户。

queryMatchInfo()函数为帮助用户在大厅搜索信息的函数，根据运动种类、运动时间来得到相应的约球请求。

deleteAppointmentInfo()是删除已发布的约球信息，modifyAppointmentInfo()是更新个人公开发布的约球信息。

2.4.2 场地信息管理

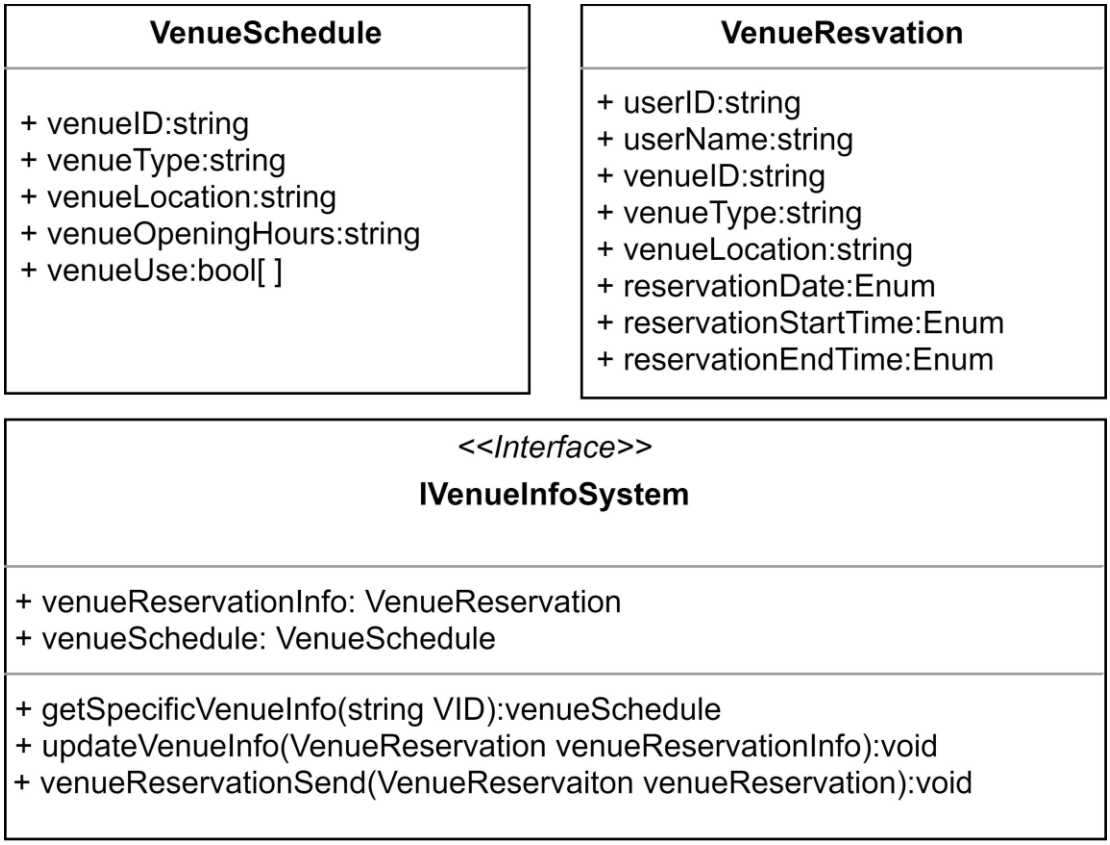


图 13：场地信息管理接口

这个是 IVenueInfoManageSystem 系统，包含 VenueSchedule 和 VenueResvation 数据类，有获取信息，更新信息和场地预约信息发送的接口函数，由视图层中场地预约系统调用

需要特殊说明的是，数据类 VenueSchedule 中，venueUse 是一个 bool 型的数组，数组的每一位代表一段时间场地的使用情况。如果为 false 说明可预约，如果为 true 说明不可预约。数组大小的开辟与时间段划分由具体场地的开放时间确定。

updateVenueInfo()函数是根据：1.用户对场地的预约 2.管理员在某场地有特殊用途时，限制场地使用 两种情况来更新场地的信息。

2.4.3 排名子系统

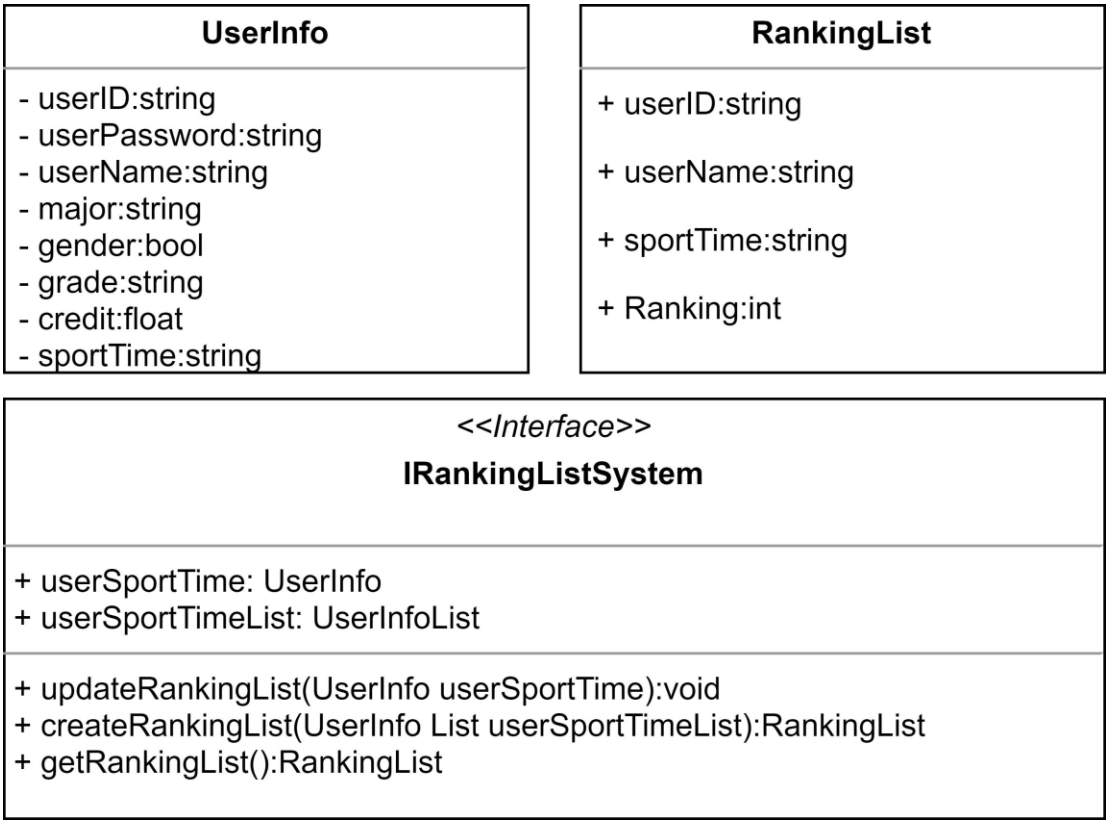


图 14：排名子系统接口

IRankingListSystem 使用的是 UserInfo 和 RankingList 数据类，存放用户信息以及排行榜信息,接口函数有更新，创建以及获取排行榜，由视图层中排行榜系统调用。

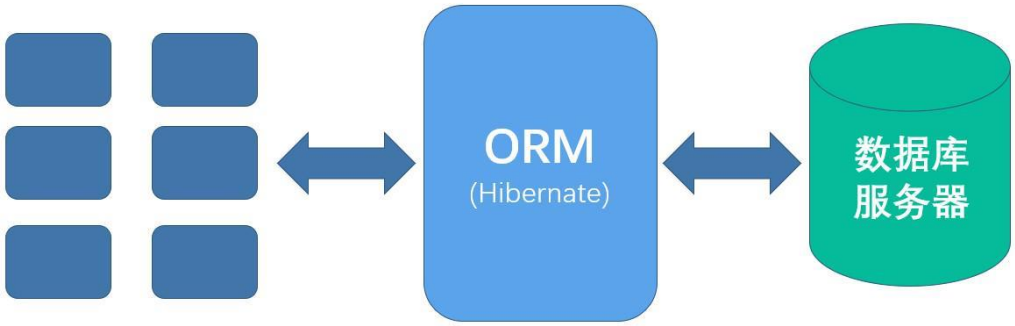
需要说明的是，排名子系统的排行榜功能是通过访问各个用户的场地预约时间来确定排名的，可能存在多个用户时间相同的情况，那么按照学号的先后顺序进行排序，如果当天没有预约场地，则不会进入到排行榜界面之中。

生成排行榜之后，如果有用户产生新的场地使用时间，则会使用 updateRankingList() 函数对排行榜进行更新，保证排行信息具有实时性。

3 设计机制

3.1 数据持久化机制

数据持久化就是将内存中的数据模型转换为存储模型,以及将存储模型转换为内存中的数据模型的统称。Hibernate 就是一个高性能的对象关系型持久化存储和查询的服务。Hibernate 不仅关注于从 Java 类到数据库表的映射，也有 Java 数据类型到 SQL 数据类型的映射，另外也提供了数据查询和检索服务。 Hibernate 作为数据库与界面之间的桥梁，需要面向对象思想操纵对象。



Hibernate 直接提供相关支持，底层驱动可以随意切换数据库，快速简洁。使业务层与具体数据库分开，只针对 Hibernate 进行开发，完成数据和对象的持久化。 数据持久化有如下的几个优点:

1. 程序代码重用性强，即使更换数据库，只需要更改配置文件，不必重写程序代码。
2. 业务逻辑代码可读性强，在代码中不会有大量的 SQL 语言，提高程序的可读性。
3. 持久化技术可以自动优化，以减少对数据库的访问量，提高程序运行效率。

我们最后选用 Hibernate 作为我们的 ORM 框架，具体理由可参照 5.3.1 持久化机制选择。Hibernate 有 6 个核心 API，即 Session,SessionFactory,Transaction,Query,Criteria 和 Configuration。下面我们将依次介绍这些 API 的功能

1.Configuration

Configuration 负责对 Hibernate 对象进行配置。一个 configuration 类的实例代表了一个应用程序从 Java 中的数据类型到 SQL 数据库的完整映射。

2. SessionFactory

SessionFactory 对象由 Configuration 对象创建，用来产生和管理 Session。常情况下每个应用只需要一个 SessionFactory。

3. Session

Session 是应用程序与数据库之间进行交互的一个单线程对象，用于获得与数据库的物理连接。在 Hibernate 中，Session 是轻量级的，对于 Session 实例的创建与销毁的开销都较小。

4. Transaction

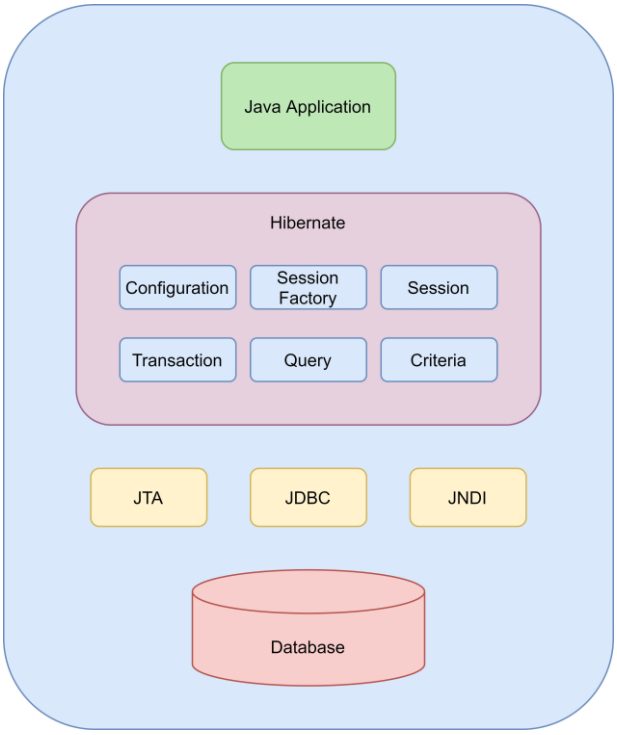
在 Hibernate 中,Transaction 是一个可选的 API，代表数据库中的一个工作单位。

5.Query

一个 Query 实例是通过 Session 创建的。通过 Query，可以使用 SQL 语句或 Hibernate 查询语句 HQL 对于数据库进行查询。

6.Criteria

Criteria 用于创建并执行面向对象的标准化查询。



在我们的业务程序与数据库之间，Hibernate 框架使用 Session 会话，来完成数据的提交、更新、删除、查询等等。

当我们需要向数据库提交数据，例如加入注册用户时，我们会把数据保存到 Session 会话中，然后根据框架的配置文件，自动或手动决定什么时候把这种保存提交到数据库。一般更新排行榜，用户信用分数我们会每日自动统一提交，而新用户注册，约球信息我们会及时提交到数据库。

当我们需要从数据库中查询数据，例如查询约球信息，可用场馆信息时，在查询这些信息之间需要清理缓存（手动清理，或者通过配置文件框架自动清理）使 Session 会话中的数据与数据库中的数据保持一致。这样程序只需要查询 Session 会话中的数据即可。

总之，数据持久化是解决程序与数据库之间交互访问的问题，使程序不直接访问数据库，而是直接访问 Session 会话，然后由 Session 会话与数据库“打交道”。只要确保，程序每次访问 Session 中的数据时，与数据库中的数据保持一致，就不会出现错误。

3.2 安全机制

网页安全模型

网页安全模型主要包括两个方面，首先是网页的安全，包括但不限于网页数据安全传输、跨域访问、用户数据安全等，其次就是浏览器的安全，在网页或者 JavaScript 代码有安全问题时，浏览器也能在运行它们的时候保证自身的安全，不泄露数据。

Same origin Policy 同源策略

当用户访问网页时，某些网页可能暗藏了恶意脚本，浏览器要确保该网页中的数据如 Cookie、用户名和密码等信息不会被其他的恶意网页所获取。因此也就有了“Same origin Policy”同源策略。在理解同源策略之前，我们要首先了解一下域的概念，对于任何一个域，都由三部分构成，分别是协议名称、域名以及端口号。以 https://www.baidu.com 为例，https:// 就是协议名，www.baidu.com 就是域名，80 是 web 的默认端口，所以也就不会显示了。只有当上述三个部分的内容完全一致时，浏览器才会认为信息是同源的，如果返回的数据不是同源的，浏览器就不会去执行文件或数据，也就避免了执行外部恶意 js 脚本导致信息泄露的风险。但是 HTML 里的一些标签是不受同源限制的。

CSP 内容安全策略

因为 HTML 内的一些标签是不受同源限制的，因此也就产生了一种全新的攻击叫做 XSS（跨域脚本攻击），即在同源网站内执行跨域的脚本，而脚本可能存在严重的安全隐患。而 CSP 是一个更高的安全层，其指定了指令可以从何处加载资源，从而防止浏览器从任何其他不安全的位置加载风险资源，能有效地防止跨域脚本攻击，以及其他的一些攻击如数据注入攻击、单击 jacking 等等。

CORS 跨域资源共享

虽然同源策略保障了基本的浏览器数据安全，但这也造成了跨域资源无法共享的问题，大大降低了网页的便捷性。而 CORS 是一种使用 header 来指定哪个外部来源可以访问本地资源的协议，简单的来说，就是服务器可以为允许访问的跨域请求制定白名单。CORS 协议允许服务器向浏览器返回一些 HTTP Headers，通俗地来讲就是白名单，然后浏览器再查看请求源是否在白名单中来选择是否响应请求。

安全传输协议

SSL 协议

对于用户而言，网页的安全还包含一个重要点，就是用户和服务器之间交换数据的安全性。对于一般的网页而言，这些数据的传输都是使用明文方式，也就是说它们对谁都是可见的，这能够满足大多数的使用情况，但是对于隐私的数据，如密码、银行帐号等，如果使用明文来传输就存在风险，为此 Web 引入了安全的数据传输协议 HTTPS，HTTPS 是在 HTTP 协议之上使用 SSL（Secure Socket Layer）技术来对传输的数据进行加密，从而保证了数据的安全性，SSL 协议是构建在 TCP 协议之上、应用层协议 HTTP 之下的，SSL 工作的主要流程是先进行服务器认证，然后是用户认证，但是 SSL 还存在一些问题，例如在涉及多方的电子交易中，SSL 协议并不能协调各方的安全传输和信任关系。

TLS 协议

TLS（Transport Layer Security）是在 SSL3.0 基础上发展起来的，它使用了新的加密算法，因此同 HTTPS 之间不兼容，TLS 用于两个通信应用程序之间，提供保密性和数据完整性，

该协议是由两层子协议组成的，包括 TLS 记录协议（TLS Record）和 TLS 握手协议（TLS Handshake），较低的层为 TLS 记录协议，位于 TCP 协议之上，TLS 握手协议具有三个属性，其一是可以使用非对称的密码术来认证对等方的身份，其二是共享加密密钥的协商是安全的，对偷窃者来说协商加密是难以获得的，此外经过认证的连接不能获取加密，即使是进入连接中间的攻击者也不能，其三是协商是可靠的，如果没有经过通信方成员的检测，任何攻击者都不能修改通信协商。

数据备份与恢复

一个应用的数据库系统总是避免不了故障的发生。安全的数据库系统必须能在系统发生故障后利用已有的数据备份，恢复数据库到原来的状态，并保持数据的完整性和一致性。数据库系统所采用的备份与恢复技术，对系统的安全性与可靠性起着重要作用，也对系统的运行效率有着重大影响。

数据的备份一般采用三种方法：冷备份、热备份和逻辑备份。在我们的平台之中为了让用户的数据不会丢失，我们采用了多种数据备份和恢复技术

由于我们的平台针对的是学生的运动功能，在白天的时间数据会使用的比较频繁，所以在晚上 12 点以后，系统进入维护状态，可以在这个时候对整个数据库的数据进行冷备份，可以更好的保障数据完整性

在平台数据使用频繁的时段，我们采用热备份的方法进行数据备份，虽然这样会让系统效率降低，但是在数据服务器发生宕机的时候，数据可以进行点恢复，避免对用户造成损失。

Oracle 数据备份技术

在我们平台使用 Oracle 进行热备份时的条件是数据库运行在归档模式，Oracle 数据库的 redo 日志记录在数据库上进行的所有活动。LGWR 后台进程以一种循环方式写这些日志文件，从第一个 redo 日志到下一个，直到该组的最后一个，然后由从第一个日志写起。

因为在非归档模式下，当循环写到最后一个日志文件后，就重写第一个日志。因此，非归档模式下唯一的数据库恢复办法就是使用冷备份。

在归档模式下，当 redo 日志满时，一个 ARCH 后台进程就读取全部 redo 日志，然后将其写到归档日志。因此，可以使用热备份和点恢复。在归档日志模式下，如果归档日志目的空间已满，数据库活动将暂时停止，只有释放一些空间后，数据库才能继续运行。通常，background_dump_destination 将产生一个跟踪文件来显示归档方面的问题。

数据恢复

在我们平台中，数据恢复采用的是基于备份的恢复、基于运行时日志的恢复和基于镜像数据库的恢复。

在平台维护期间的冷备份数据可以用来系统的数据恢复，如果系统在非重要时段发生故障，这是没有新出现的数据信息，采用备份恢复的方法最为方便。

由于运行时日志文件是用来记录对数据库每一次更新的文件。所以当系统突然失效而导致事务中断时，可重新装入数据库的副本，把数据库恢复到上一次备份时的状态。然后系统自动正向扫描日志文件，将故障发生前所有提交的事务放到重做队列，将未提交的事务放到撤销队列执行，这样就可把数据库恢复到故障前某一时刻的数据一致性状态，这样可以进行数据的点恢复。

数据库镜像就是在另一个磁盘上复制数据库作为实时副本。当主数据库更新时，DBMS 自动把更新后的数据复制到镜像数据，始终使镜像数据和主数据保持一致性。当主库出现故

障时，可由镜像磁盘继续提供使用，同时 DBMS 自动利用镜像磁盘数据进行数据库恢复。镜像策略可以使数据库的可靠性大为提高，但由于数据镜像通过复制数据实现，频繁的复制会降低系统运行效率，因此一般在对效率要求满足的情况下可以使用。

4 用例实现

4.1 约球

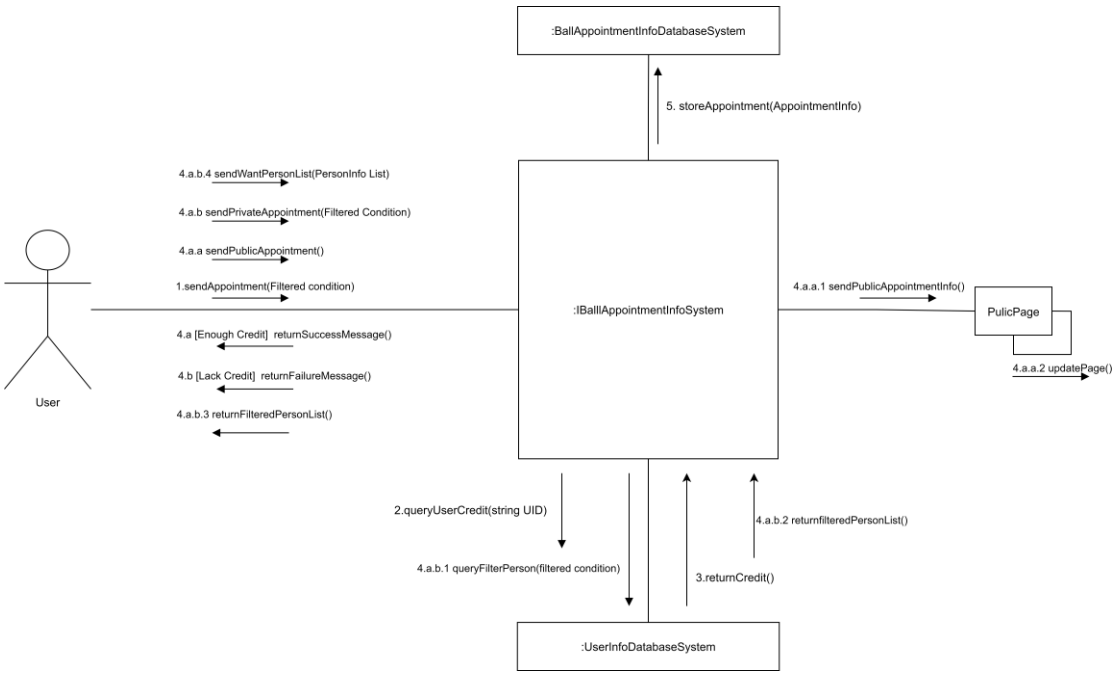


图 15： 约球通信图

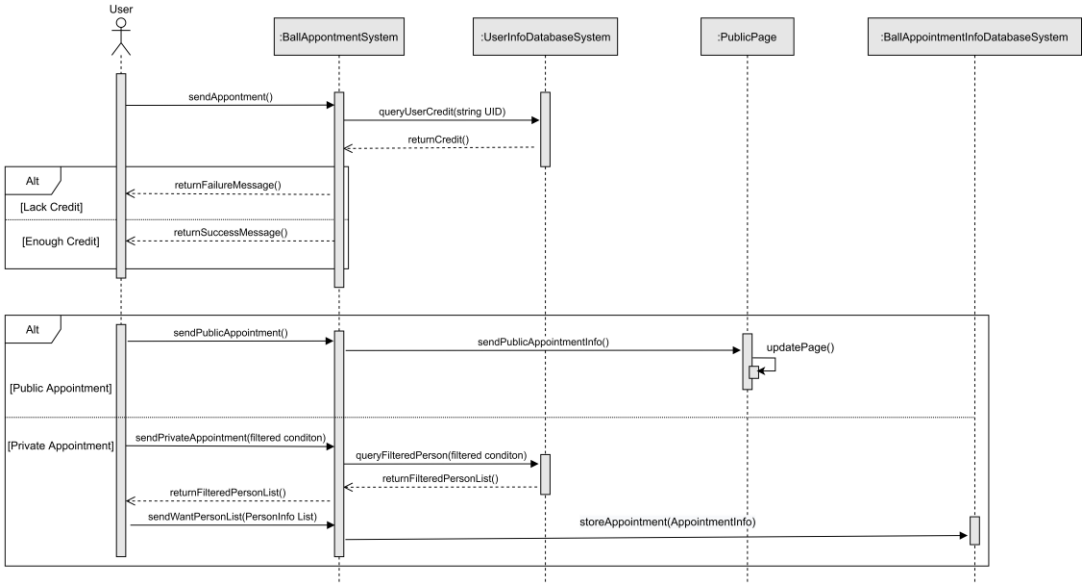


图 16： 约球时序图

用户可以通过约球系统进行约球。首先用户发送一个约球请求，约球系统会在用户信息数据库中查询用户信誉积分。若信誉积分不够，则返回约球失败信息；若约球成功，则用户将选择是在公共大厅进行公开约球，或是私发约球信息。

若用户选择公开约球，则约球系统会将相关信息发送给大厅界面，在大厅界面上进行展示；若用户选择私发约球信息，则用户首先确定筛选条件，约球系统根据此筛选条件，查询用户信息数据库再返回筛选人员名单，用户在筛选名单中进行选择，最终这个约球信息将会被存入约球信息数据库。我们采用数据持久化机制，数据存入数据库之前要经过数据持久化层，再存入数据库。

4.2 场地预定

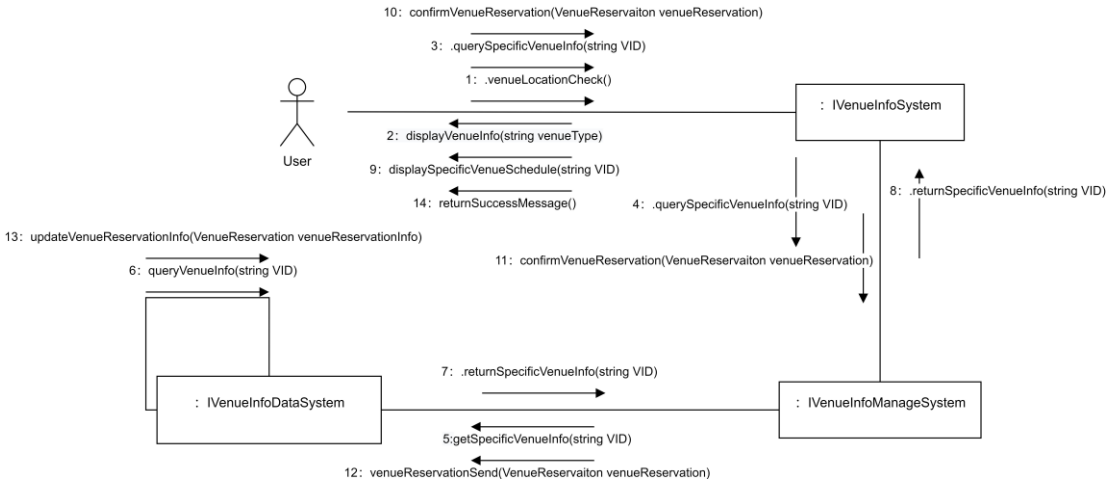


图 17： 场地预定通信图

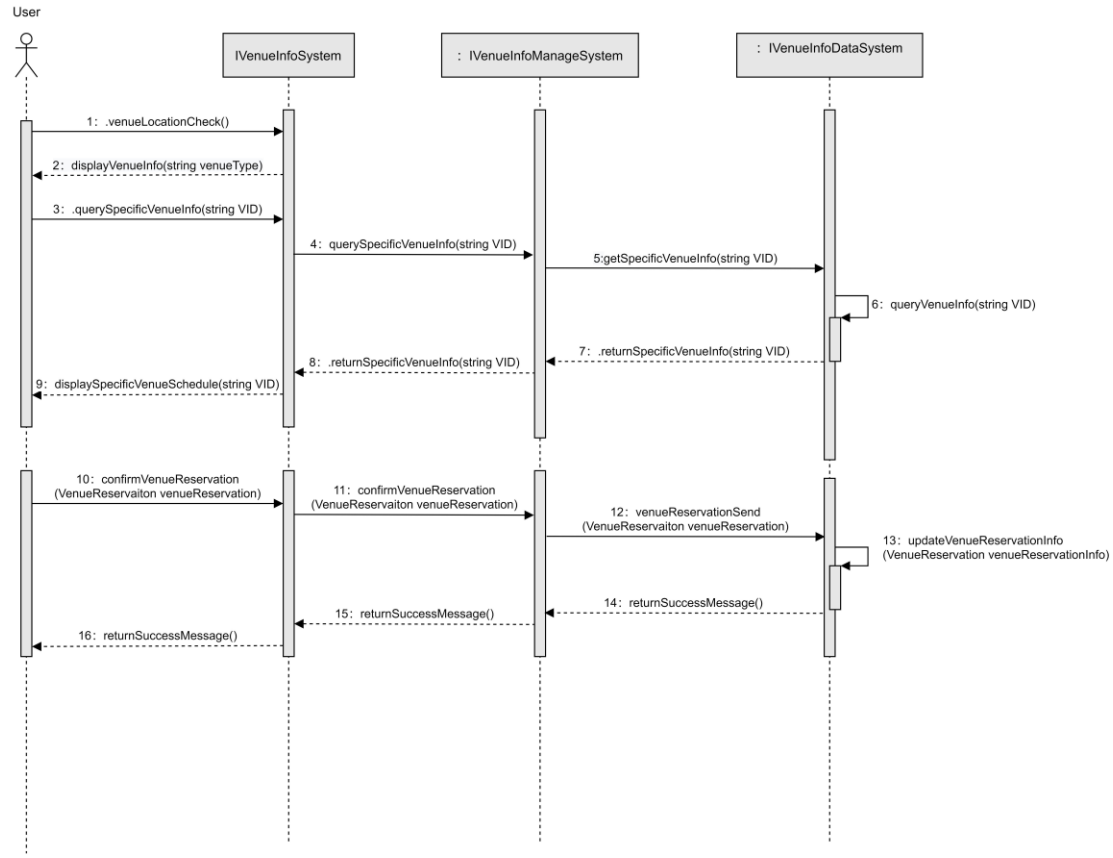


图 18： 场地预定时序图

用户预约场地可以使用场地信息系统进行场地预定，首先用户查询场地信息中，目前还有哪些场地开放，在场地信息界面选择一个具体的场地类型查看相关场地的具体信息。每次

用户查看场地信息的时候，场地信息系统通过控制器从场地信息数据库中查询到最新的场地信息并且返回给用户查看。

用户确定场地提交具体预约信息，在场地信息预约界面选择预约的场地和时间，场地信息管理系统将预约信息发送给场地信息数据库，数据库根据预约信息更改数据库场地状态并且向用户返回一个成功信息。

5 架构风格与设计决策

5.1 架构风格

在架构风格方面，我们结合了面向对象架构风格，分层架构风格和微服务架构风格。面向对象的架构风格建立在数据抽象和面向对象的基础之上，将数据的表示方法及其相应的操作封装在一个抽象的数据类型或者对象中。数据抽象和封装的概念可以在保持外部接口不变的情况下改变内部实现，从而减少甚至避免对外界的干扰。我们改变个性化定制的内部业务逻辑，对外来看并未感受其发生变化，对系统的影响降到最低。继承的概念可以大幅减少冗余的代码，在日后产品迭代时也可以方便地扩展现有代码，提高编码效率，也降低了出错概率，降低软件维护的难度。以对象为中心的设计可以帮助开发人员从静态（属性）和动态（方法）两个方面把握问题，从而更好地实现系统。

分层架构中，每一层都作为下层客户为上层服务，系统结构清晰，因此每一层也仅只与其相邻的层之间有交互，只对相邻的层透明，可以将一个复杂的问题分解为一系列步骤序列。进一步，我们只需要给相邻的层设计和提供约定好的规范接口，能显著地提高每一层的重用性。我们这个系统设想用户需要通过小程序或者网页这个窗口进行数据的输入查看，那么这个数据需要保存到数据库中，因此我们分为三层，分别为表示层、业务层、数据层。表示层是面向用户的界面，数据层则需要通过数据持久化将用户的数据保存起来。以个性化定制为例，表示层需要提供输入控件与用户交互，获取用户输入的个人需求，身高体重，训练目标等等。这些操作本身需要通过业务层来接收，并进行简单逻辑处理，如果合法，将这些数据交给数据访问层处理。

同时，我们也采用了微服务架构风格，将一个应用程序设计构建为一组松耦合的协作服务，每个服务都实现了一部分相关的功能。采用微服务架构风格使每一个微服务部分的代码规模更小，无论是开发还是维护都更加高效。我们将这个线上运动社区根据业务拆开，拆成运动子系统，社交子系统，个性化子系统等等分别进行部署开发，不同的服务部署在不同的 docker，他们通过轻量级的通信协议比如 http 协议进行通信。这样的好处除了解耦之外，在某些业务比如约球，在用户量大的时候需要水平向扩展，进行负载均衡。在传统架构里需要将每个实例都独立运行一份，所有模块都有同样份数，造成资源浪费。微服务架构的话每个模块根据计算复杂量，用户访问量独立进行部署和运行，每个模块都可以独立进行水平向扩展，我们约球的用户访问量大，那就对约球子系统进行独立的负载均衡，充分利用资源。

5.2 设计模式

业务代表模式：

在我们的约球系统中，虽然仅仅针对校园情形下，但是我们想要实现的功能较多，业务逻辑也是较为复杂的，设计专门的接口来支持表现层的局部访问是必要的。所以在我们的系统中，我们采用业务代表模式，用于将表现层与业务层进行解耦。在业务层的子系统中添加业务代表接口，而在使用中，让表现层去使用这些接口与业务层通信，降低了两层之间真实通信的复杂程度，同时降低了系统的耦合度与通信负担。

5.3 主要设计决策

5.3.1 持久化机制的选择

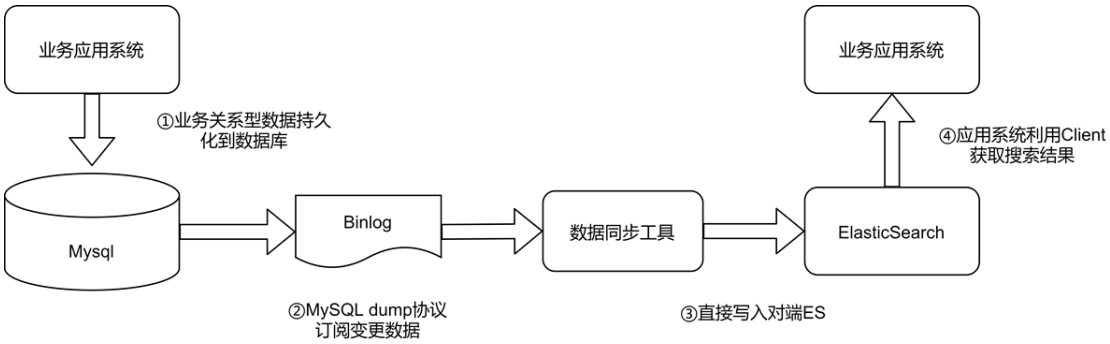
我们的线上运动社区系统的数据库中保存全同济师生的数据，数据量较大。因此数据的安全与可靠性将很大程度上影响我们系统性能的表现。

因此我们采用数据持久化的技术，在我们的系统中，数据的主要传输发生在前端程序和后端数据库的交互中，若处理不好，大量的数据直接传输会导致数据错误和丢失，我们将内存中的数据模型转换为存储模型,以及将存储模型转换为内存中的数据模型，我们需要在业务层和数据库之间加入一个持久层，使程序不会直接与数据库交互。这就需要用到对象关系映射，即 ORM，ORM 可以将 Java 中的对象与关系型数据库中的表建立一种映射关系，是我们能够通过操作 Java 对象就可以完成对数据库表的操作，这样开发人员只需要关注纽带一端映射的对象就可以了。

在选择哪个作为 ORM 的框架的问题上，我们比较了 MyBatis,Hibernate,JOOQ 和 JDBC Template 这几个数据持久化框架，JDBC Template 其实并没有做封装，虽然操作灵活，但只是单纯 SQL 查询，几乎没有做 ORM，十分复杂，所以我们首先排除。JOOQ 作为一款轻量，简单，足够灵活的工具包，有很多优点，但是 JOOQ 相比 Hibernate 和 Mybatis 等主流框架较为小众，而且不是完全免费的，因此我们也没有选择 JOOQ。相较于 Mybatis,Hibernate 在对象的维护和缓存方面更具优势，更加稳定，因此我们最终选择了 Hibernate 作为系统的 ORM 框架实现。

5.3.2 读写分离

因为我们的运动社区系统，用户不需要对上传的数据做到毫秒级的实时反馈，而用户时常需要进行查询操作。数据库的读写操作的使用频率是存在较大的差异的，更新少，查询多。因此我们采用读写分离。MySQL 数据库用来进行写操作，elasticsearch 用来进行读操作。为了保证 MySQL 和 elasticsearch 数据同步，我们使用 Binlog 技术，这样我们就完成了模糊搜索，降低对写的影响，并且用户可以实时看到。



6 原型进展

6.1 前端原型

在我们的系统登录界面，可以使用两种登录方式。可以使用绑定的手机号码进行登录，也可以使用统一身份认证进行登录。忘记密码可以点击忘记密码跳转到找回密码的界面，修改之前的密码进行登录。

在个性化定制功能面板中，首先要获取用户的身高、体重等信息来制定适合用户身体素质的训练计划，用户可以在界面中选择自己的运动目标和想要的运动强度。

私信界面中，用户之间可以通过发送语音、文字等消息形式进行交流。

在用户想约球时，可以选择大厅约球和特定对象约球两种模式，在后一种模式之中，用户可以在界面上筛选想要约球对象的条件，如：专业、性别、年级等。

用户在浏览大厅界面的时候，可以选择自己想要应邀的约球信息完成约球活动，也可以在大厅搜索栏中搜索自己想要约球的运动。

用户在预约场地的时候，既可以在搜索栏中搜索想要预约的场地、也可以在下拉列表中选择一个场地类型进行预约。

用户选择好要预约的场地以后，在具体场地预约界面输入想要预约的位置和想要预约的时间，选择好后，点击确认发送预约信息。

6.2 后端原型

```
events {
    worker_connections  1024;
}


http {
    include       mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  logs/access.log  main;

    sendfile        on;
    tcp_nopush     on;

    keepalive_timeout  0;
    keepalive_timeout  65;

    gzip  on;

    upstream tomcat_proxy{
          server 192.168.3.63:8080 max_fails=3 weight=1 fail_timeout=60s;
          server 192.168.3.64:8080 max_fails=3 weight=1 fail_timeout=60s;
        }
```

关于后端原型，我们进行了负载均衡的配置以及登录功能 JAVA 的简单实现。我们根据我们的开发架构采用 nginx 进行反向代理以及负载均衡。这里是负载均衡的配置，假设有 192.168.3.63 和 192.168.3.64 两个服务器，我们以 1：1 的方式配置，也就是说这两个服务器各负责 50%的任务分配。

```java
@Service
public class LoginServiceImpl implements LoginService {
    @Autowired
    private UserMapper userMapper;

    @Override
    public Result login(LoginDTO loginDTO) {
        if (StringUtils.isEmpty(loginDTO.getLoginName())){
            return new Result(400,"账号不能为空","");
        }
        if (StringUtils.isEmpty(loginDTO.getPassword())){
            return new Result(400,"密码不能为空","");
        }
        QueryWrapper<User> wrapper = new QueryWrapper();
        wrapper.eq("login_name", loginDTO.getLoginName());
        User uer=userMapper.selectOne(wrapper);
        if (uer!=null&&uer.getPassword().equals(loginDTO.getPassword())){
            LoginVO loginVO=new LoginVO();
            loginVO.setId(uer.getId());
            loginVO.setToken(UUID.randomUUID().toString());
            loginVO.setUser(uer);
            return new Result(200,"",loginVO);
        }
        return new Result(401,"登录失败","");
    }
}
```

我们设计了一个较为简单的用于登录的 API，来验证用户的用户名和密码，图中显示了登录的部分代码实现。客户将 JSON 字段发给服务器后，后端返回登陆状态和用户信息，登陆失败则提示失败信息。

7 遗留问题

问题一

本系统的定位是面向同济大学师生的线上运动社区，在实现预约场地等功能时需要调用校内场地的相关信息，但学校相应的接口我们没有查到，一些信息可能无法获取。

问题二

在我们的后端原型中，我们没有能够再进一步的在服务器端完整的实现，而是将负载均衡 nginx 配置好后，用 JAVA 简单实现了登录的功能。后续可以进一步优化。

8 项目反思

李航宇

反思：从 assignment0 到 assignment3，我们在许多方面都作出了修改与改进，开始时我们设想的是一个以约球为主要功能线上运动社区，并附带社团、私教等功能，但这些功能过于宽泛，涉及到的细节过多，后来我们将这些非核心功能去除了一大部分，并对于用户不再进行具体的区分，减小了复杂度，使我们能将精力放在核心功能的挖掘上。

在项目中，我主要负责系统中平台管理子系统与社交子系统的设计与相关说明，也参与了其他部分的相关工作。通过这学期的项目，我对于一个系统的设计与分析有了初步认识，了解到了该从什么角度、以什么方式去分析、设计一个系统；学会了如何通过时序图、交互图等描述一个用例，对于子系统的划分与设计的细节也有了一定了解，对于系统的架构也有了一定认识；此外，通过这学期的项目迭代，我们对于团队合作也有了更多经验，在分配任务与协作上的效率在一次次项目中逐渐提高。

胡启云

在整个项目过程中，我主要专注于个性化定制系统的设计、原型设计和演示和。在与所有团队成员的交流和讨论中，我也学到了很多。选择使用哪种架构和框架，需要我们清楚地了解所有方法的优缺点。

作为一个对系统分析设计没有那么多经验的大二学生，我在网上查了很多资料，参考了很多推荐的书。在完成我自己的部分后，我们其他团队成员之间互相分享了知识，以确保我们始终保持联系。

至于局限性，我认为我们还缺乏实际经验，对架构服务这块的知识了解的还不够充足，以至于我们在实施时没有太多选择。但好在我们通过查找介绍如何设计分析和设计系统的文章，在刘老师对我们系统的评估指点下，在我们团队每个人的齐心协力下，终于完成了同济校园运动社区的整个系统分析与设计

郑柯凡

本学期，在整个项目的分析与设计中，我们遇到了许多难题。从第一次布置任务时的不知所措，对系统分析和设计一无所知，不知道要做什么，到第二次任务时讨论系统的功能、

学习用例图和活动图的画法，再到第三次任务时的架构思考以及类图时序图的思考，我们的项目推进不是一帆风顺，但我们从中学到了很多，也才有了最终的完整系统展示。在整个项目的开发设计中，我主要负责项目架构的设计。在逻辑架构的设计过程中，我们借鉴了部分面向对象和分层架构的风格思想，但在其核心思想和具体的细节实现方面还有待进一步学习。在开发架构和部署架构的设计方面，我们借鉴了一些前沿的技术帮助我们完成预期的系统目标，但对这些技术的了解可能只是停留在表层，对我们整个系统的融合和应用可能不是很合适，这还需要后期深入调研或实际上手之后再做优化修改。

杨梓浩

在这个项目当中，我主要的工作是子系统的一些活动，例如场地预定和约球活动。

在绘制两个用例的时序图的时候，我意识到一个项目中的某个功能完整的行为逻辑要和其他很多的平台联系起来，各个功能系统之间的接口要清晰明了。在做这个系统功能点的时候，我查阅了很多资料，包括时序图和类图的画法，一些活动可能的内在逻辑，不同条件情况下的不同反馈等。这些知识让我设计系统功能逻辑的时候更加得心应手，与队友交流的时候也更加准确快捷。总之，这些系统分析工作让我学到了一个系统的开发逻辑和思考方向，让我收益匪浅。

曹峰源

在本次项目中，我们组设计的是一个适用于校园平台的约球系统。对于我个人来说，我在学习这门课，做这个项目之前，是没有任何开发经验的。这一次虽然没有全部写代码实现，但是确实让我感受到了软件开发是一个怎样的流程，也体会到开发出一个优秀的软件系统是多么复杂与不容易的一件事。

我在项目中主要是与杨梓浩同学负责完成约球功能的实现，从第一次的确定功能到第二次类图、时序图的制作再到最后的接口规范，每一次团队的交流中我都能学到很多东西。最让我印象深刻的还是第三次作业中对于开发架构的认识，真实的框架与技术选择让我们着实犯了难，最后还是通过我们不断地学习与请教解决了这个问题。这也让我体会到了系统分析与设计中团队协作的重要性。

9 团队分工

学号	姓名	主要工作	贡献度
1951593	李航宇	用例实现、子系统与接口	20%
1952728	杨梓浩	用例实现、子系统与接口	20%
1950072	郑柯凡	系统架构、设计机制	20%
1952897	胡启云	主要设计决策、设计机制、文档整理	20%
1951328	曹峰源	子系统与接口、原型进展	20%