南京大学软件学院

SPORTLIFE 系统设计文档

张云飞 141250197

2016-11-2

目录

1 引言

1.1 编制目的

本报告详细完成对 sportlife 网站的概要设计，达到指导网站开发的目的，同时实现和测试人员及用户的沟通。

本报告面向开发人员、测试人员及最终用户而编写，是了解系统的导航。

1.2 参考资料

Sportlife 网站需求规格说明文档。

2 产品概述

参考需求规格说明文档中对产品的概述。

3 逻辑视角

Sportlife 网站系统中，选择了分层体系结构风格，将系统分为 3 层（展示层、逻辑层、数据层）能够很好地示意整个高层抽象。展示层包括 html 网站页面的实现，逻辑层包含业务逻辑处理的实现，数据层负责数据的持久化和访问。分层体系结构的逻辑视角和逻辑设计方案如图 1 和图 2 所示

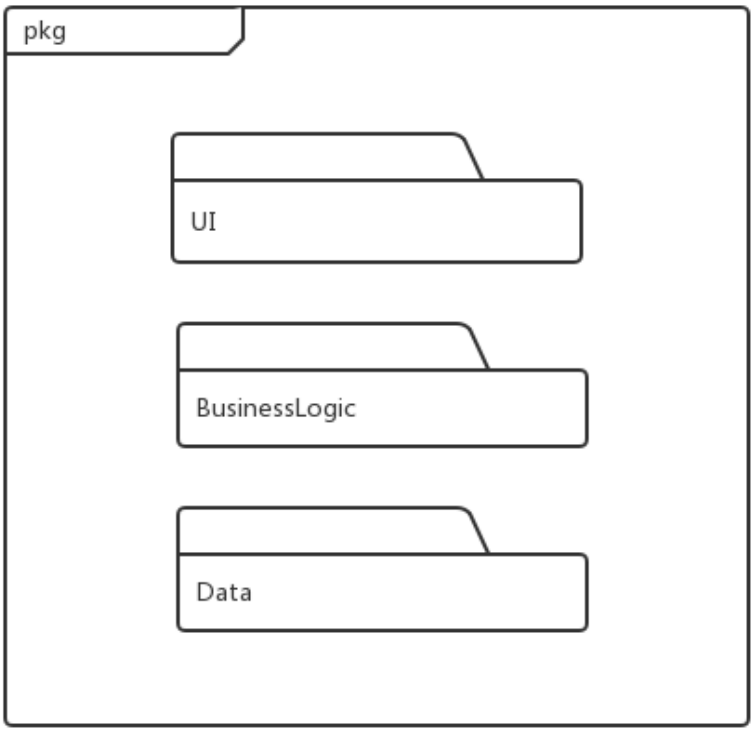

图 1 分层结构逻辑视角

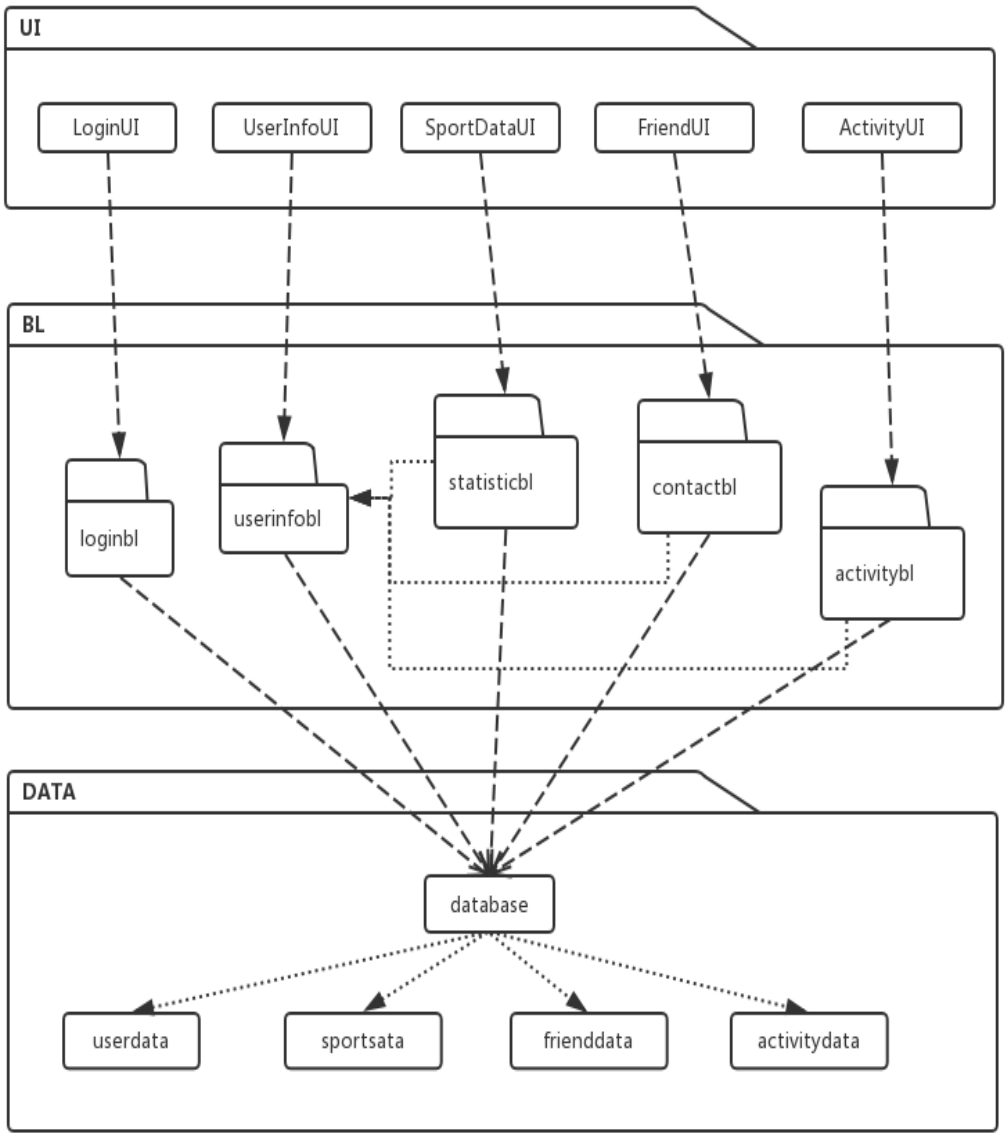

图 2 软件体系结构逻辑设计方案

4 组合视角

4.1 开发包图

开发包	依赖的其他开发包
html	Js,css,businesslogic
js	,html,css
css	html

loginbl	database
userinfobl	database
statisticbl	userinfobl,database
contactbl	Userinfobl,database
activitybl	Userinfobl,database
database	sqlite

4.2 运行时进程

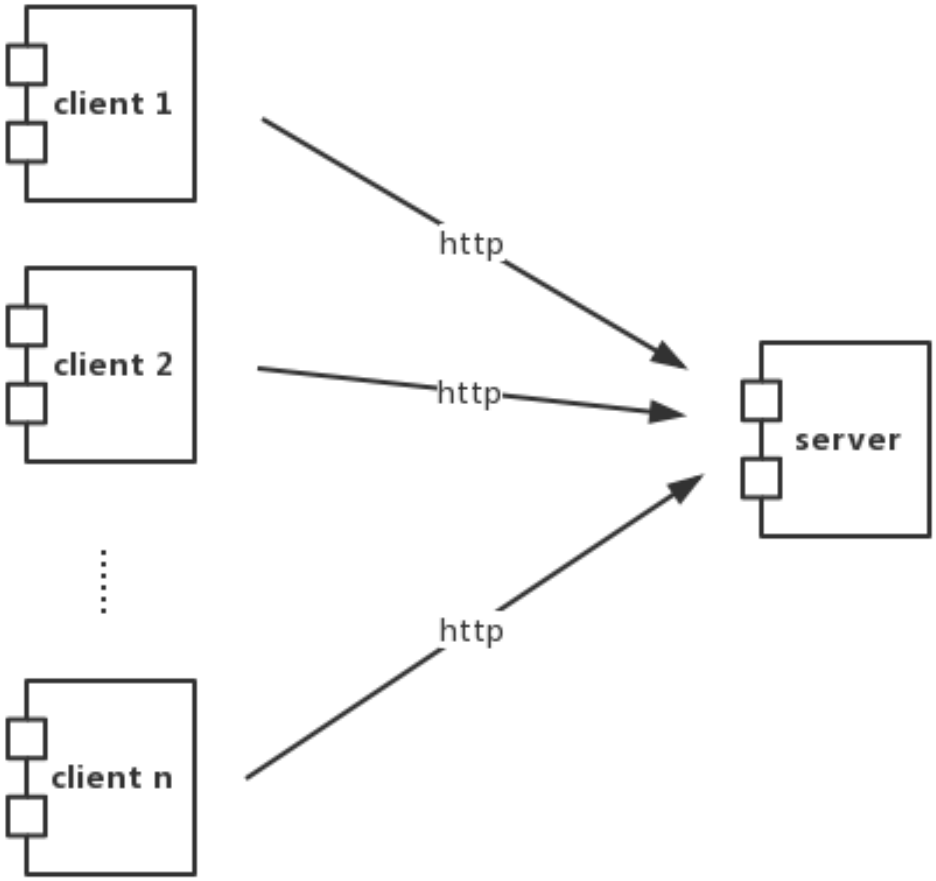

图 3　进程图

4.3 物理部署

Sportlife 系统在客户端通过使用浏览器进行访问，服务端放在服务端的机器上。

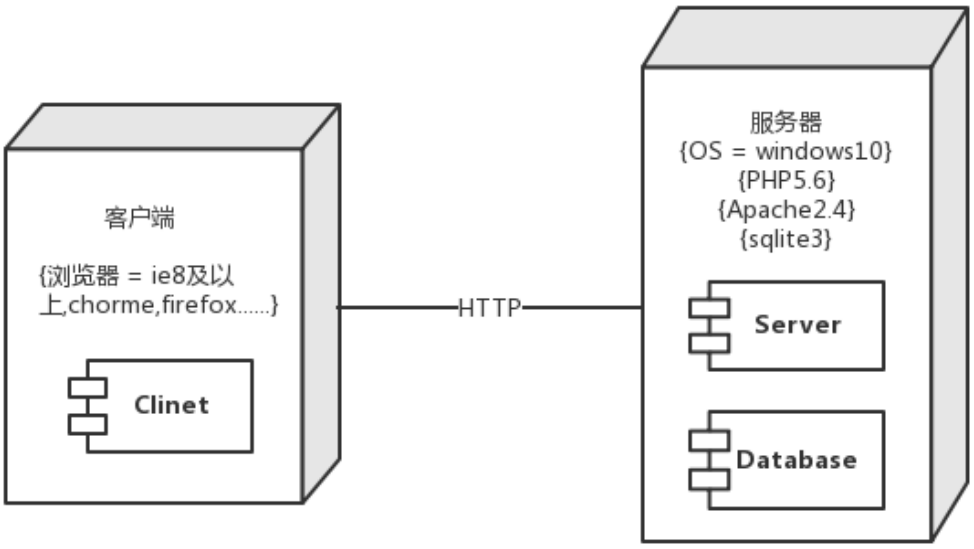

图 4 部署图

5 接口视角

5.1 界面层的分解

根据需求，系统存在如下界面：注册登录界面，主界面，用户运动统计界面，好友界面，活动界面，用户信息编辑界面。界面跳转如图 5 所示：

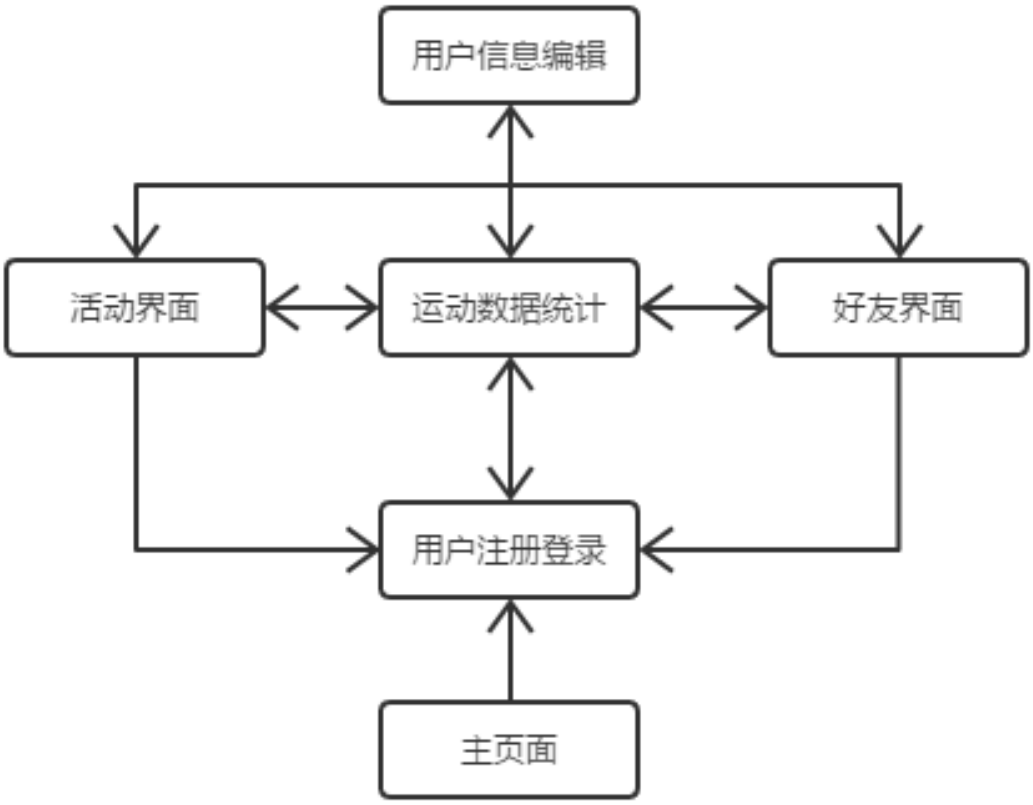

图 5 界面跳转图

用户界面层模块负责用户界面的显示，利用 html5,css 和 JavaScript 来实现。

5.2 逻辑层的分解

5.2.1 逻辑层模块的职责

逻辑层模块的职责如下表所示：

模块	职责
Userbl	负责实现用户的注册登录以及个人信息编辑的功能
Statisticbl	负责实现对用户的运动数据进行统计所需要的服务
Contactbl	负责实现好友管理所需要的功能

Activitybl	负责实现活动管理所需要的功能

5.2.2 逻辑层模块的接口规范

逻辑层的接口规范分别如下：

Userbl 模块的接口规范		
UserHandle.getUser	语法	function getUser($userid)
	前置条件	数据库中存在请求的 Userid
	后置条件	得到用户的基本信息
UserHandle.saveUser	语法	function saveUser($userInfo)
	前置条件	UserInfo 中的数据有效
	后置条件	保存用户信息

statisticbl 模块的接口规范		
statisticHandle. getStatisticsToday	语法	function getStatisticsToday($userid)
	前置条件	数据库中存在请求的 Userid
	后置条件	得到用户的当天运动数据统计信息
statisticHandle. GetStatisticsWeek	语法	function getStatisticsWeek($userid)
	前置条件	数据库中存在请求的 Userid
	后置条件	得到用户的一周内运动数据统计信息
statisticHandle. GetStatisticsAll	语法	function getStatisticsAll($userid)
	前置条件	数据库中存在请求的 Userid
	后置条件	得到用户的历史运动数据统计信息
statisticHandle. saveSportData	语法	function saveSportData($sportData)
	前置条件	传入的数据格式正确
	后置条件	系统保存用户的运动数据

contactbl 模块的接口规范		
contactHandle. getContactList	语法	Function getContactList($userid)
	前置条件	数据库中存在请求的 Userid
	后置条件	得到用户的好友列表
ContactHandle. getFriendInfo	语法	function getFriendInfo($userid, $friendid)
	前置条件	数据库中存在请求的 Userid 和 friendid
	后置条件	得到用户的好友信息
contactHandle. AddFriend	语法	function addFriend($userid, $friendid)
	前置条件	数据库中存在请求的 Userid 和 friendid
	后置条件	将该好友加到用户的好友中
contactHandle. delFriend	语法	function delFriend($userid, $friendid)
	前置条件	数据库中存在请求的 Userid 和 friendid
	后置条件	将该好友从用户的好友中删除

activitybl 模块的接口规范		
activityHandle. getActivityList	语法	Function getActivityList ()
	前置条件	无
	后置条件	得到活动平台的活动列表
activityHandle. getUserActivitys	语法	function getUserActivitys($userid)
	前置条件	数据库中存在请求的 Userid

	后置条件	得到用户相关的活动列表
activityHandle. GetActivityInfo	语法	function GetActivityInfo ($activityid)
	前置条件	数据库中存在请求的activityid
	后置条件	获得该活动的详细信息
activityHandle. AddActivity	语法	function addActivity ($activityid)
	前置条件	无
	后置条件	将该活动添加到数据库中
activityHandle. updateActivity	语法	function updateActivity ($activityid)
	前置条件	数据库中已存在该活动
	后置条件	将修改后的活动添加到数据库中
activityHandle. delActivity	语法	Function delActivity ($activityid)
	前置条件	数据库中已存在该活动
	后置条件	将该活动从数据库中删除

5.3 数据层的分解

数据层主要给逻辑层提供数据访问服务，通过 sqlite 数据库实现对数据的存储。数据层模块的描述具体如下图所示。

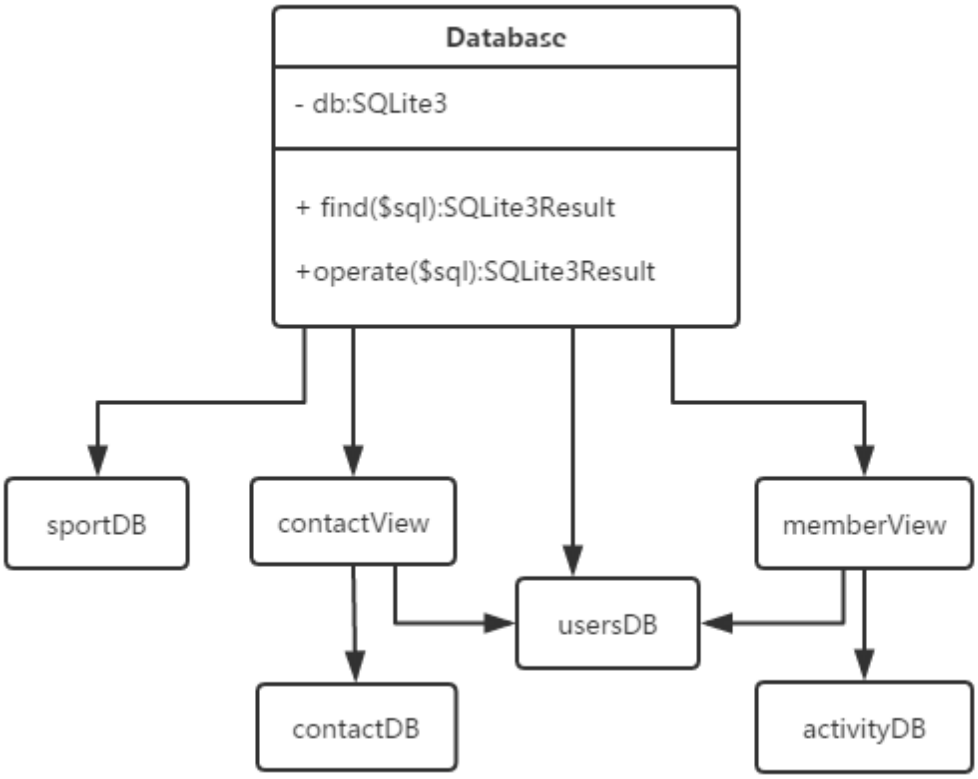

图 6 数据层模块

5.3.1 数据层模块的接口规范

数据层的接口规范如下：

data 模块的接口规范		
DB.find	语法	function find($sql)
	前置条件	Sql 语句合法
	后置条件	获得要查询的数据
DB.operate	语法	Function operate ($sql)
	前置条件	Sql 语句合法
	后置条件	执行 sql 语句的操作

6 信息视角

6.1 数据库表

```
DROP TABLE IF EXISTS user;
CREATE TABLE users (
    userid char(64) PRIMARY KEY,
    name char(255) NOT NULL,
    password char(20) NOT NULL,
    grade int,
    birthday char(10),
    gender char(4),
    address char(100),
    introducec char(256)
);

DROP TABLE IF EXISTS sport;
CREATE TABLE sport(
    userid char(64) PRIMARY KEY,
    day char(10) PRIMARY KEY,
    distance double DEFAULT 0,
    time int DEFAULT 0,
);


DROP TABLE IF EXISTS contant;
CREATE TABLE contant(
    hostid char(64) PRIMARY KEY,
    friendid char(64) PRIMARY KEY,

);

DROP TABLE IF EXISTS activty;
CREATE TABLE activty(
    activityid char(72) PRIMARY KEY,
    creator char(64) NOT NULL,
    name char(64) NOT NULL,
    startTime char(18) NOT NULL,
    endTime char(18) NOT NULL,
    peopleNum int DEFAULT 1,
);

DROP TABLE IF EXISTS member;
CREATE TABLE member(
    activityid char(72) PRIMARY KEY,
    memberid char(64) PRIMARY KEY,
);

CREATE VIEW contactView AS
SELECT userid,friendid,name,grade FROM users,contact
WHERE userid=hostid;

CREATE VIEW memberView AS
SELECT activityid,memberid,name,grade FROM users,member
WHERE userid=memberid;
```

图 7 数据库表

6.2 数据格式

用户信息 API 格式：

```
{
        "userid":"String",
        "name":"String",
        "grade": "int",
        "birthday": "String",
        "gender": "String",
        "address": "String",
        "introduce": "String",

}
```

运动数据 API 格式：

```
{
        "userid":"String",
        "daydate":"String",
        "distance": "double",
        "sportTime": "int",

}
```

好友数据 API 格式：

```
{
        "hostid":"String",
        "friendid":"String",

}
```

活动信息 API 格式：

```
{
        "activityid":"String",
        "creator": "String",
        "name":"String",
        "startTime": "String",
        "endTime": "String",
        "peopleNum": "int",
```

}

活动成员信息 API 格式：

```
{
        "activityid":"String",
        "memberid": "String",

}
```