“旅有友”

-----旅行系统

目　　录

1、引言

1.1 背景

当前市场上已经有许多旅行软件，比如：马蜂窝、驴妈妈。它们能够给用户上传图片、游记，记录旅行中的美好，并与其他人进行分享。这种方式让我十分喜欢。

在此基础上，我萌发了设计一个系统系统的想法，可以让用户记录旅行的心境。

1.2 要求

本项目构建的旅行目的地智能推荐系统, 针对想要旅行但是又不能决定目的地的用户。用户输入自己的旅行时间、旅行长度、出行方式, 系统会根据自己的算法，对数据库中数据进行排序。然后按照优先顺序向用户推荐适合的出行目的地。并且每个目的地都附有评分、图片、评论、概况、攻略、游记等多项标签，为用户提供更加全面的参考，帮助用户走出选择的困境.

1.3 目标

本项目将利用数据库技术及 php 后端开发知识搭建服务器，连接数据库, 尝试构建 cs 架构的网页界面。在本项目完工后, 用户可以在交互界面输入自己地点.软件将根据数据库输出相关信息。

2、需求分析与可行性分析

2.1 业务调查

通过线下对用户的走访调查，对旅行 app 有了深入的了解：对于管理员而言：日常常用业务包括，用户管理、地点管理、图片管理、游记管理；同时用户常需查询地点信息及个人上传信息。

通过走访调查，我们发现通过建立一个便捷的旅行系统对于减轻管理员的工作强度、方便用户查阅是很有必要的。

2.2 系统目标

本软件主要适用于有旅行需求却缺乏旅行目的地的用户, 在需求 上充分考虑了用户无参考资源，无评论信息，无详实攻略，只有初步想法等 一系列实际情况, 以用户的旅行需求为切入点, 着眼于为相关人群提供出行帮助.

2.3 系统的功能需求

系统针对不同角色的功能需求，分别建立网页服务端以满足其不同功能需求，如下图所示：

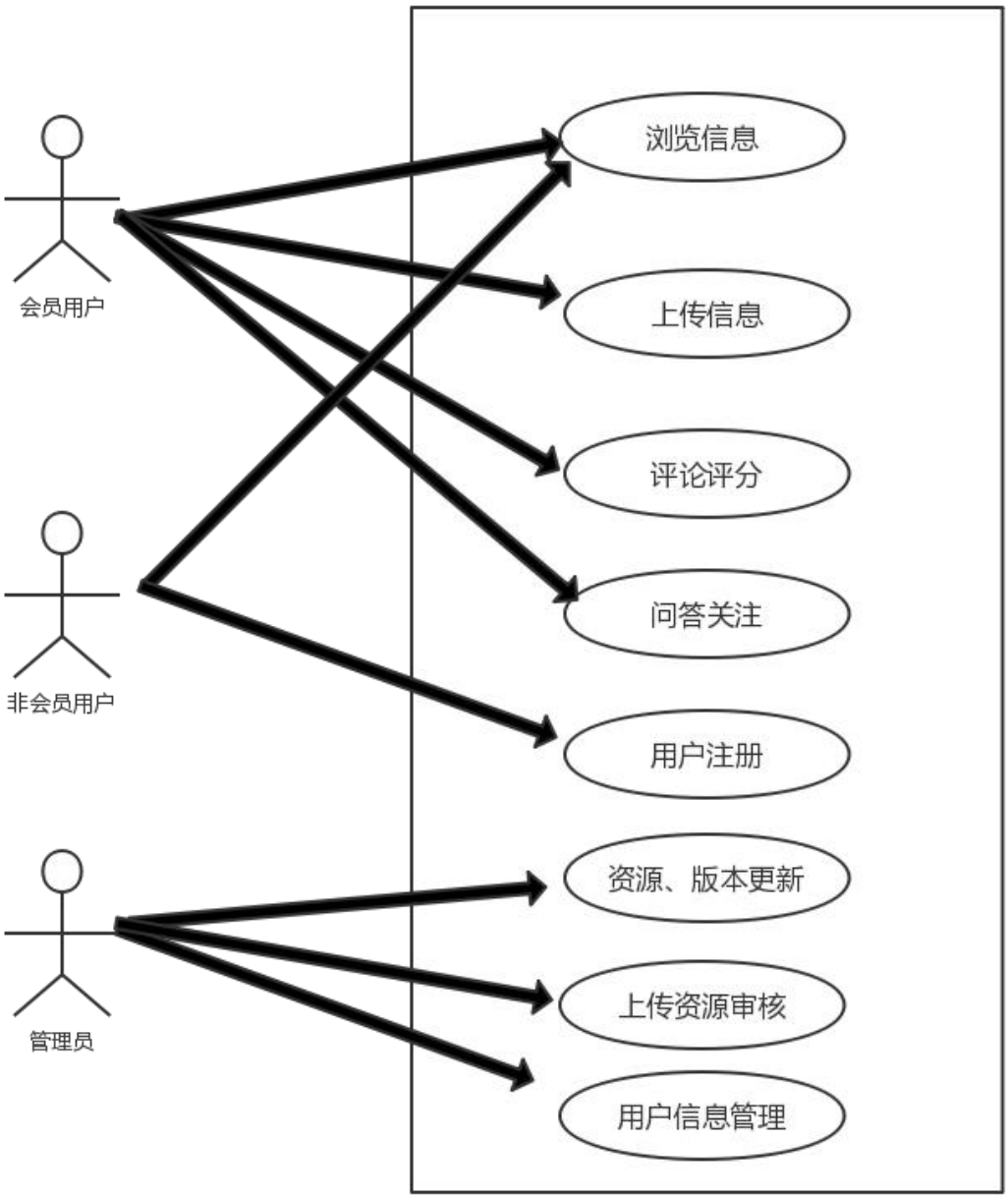


图 2.1 系统用例模型

2.4 系统的性能需求

系统应经常检查数据库中的信息，为用户提供准确的图书信息，达到数据的准确要求。同时，系统界面应设计得美观、大方、简洁，便于用户操作，也便于管理员管理系统中各类信息。另外，系统的响应时间应在用户所能接受的范围之内，这保证足够的数据存储空间和较快的 CPU 响应时间。

2.5 系统的输入输出需求

系统的输入应满足最小量原则、简单性原则以及及早检验原则，即在满足处理要求的前提下应使输入量尽量小，系统能够计算出的数据不要由用户输入；输入过程应尽量简单；对输入数据的检验尽量接近原始数据发生点，使错误能及时得到改正。

系统的输出内容应简洁明了，使得用户一目了然；本系统中输出设备选择在屏幕上输出。同时对于错误输入，进行简洁易懂的错误提示输出。

2.6 可行性分析

2.6.1 技术可行性

技术可行性就是对待开发的系统进行功能、性能和限制条件的分析，确定在现有资源的条件下，技术风险有多大，系统是否能够实现。其中，资源包括已有的或可以活动硬件、软件资源，现有技术人员的技术水平与已有的工作基础。该系统实现的功能相对较简单，通过对该系统需要的技术分析，需要的技术有 php 与 html 交互技术、php 访问 mysql 数据库技术以及对 mysql 数据库的基本操作的技术。

2.6.2 经济可行性

由于本次开发为个人课程设计，不需要考虑经济效益以及开发投入。故经济可行性不做考虑，必可行。

2.6.3 社会可行性

法律方面，图书管理系统，需要使用正版操作系统软件即应用软件平台，避免为此发生法律纠纷。同时本系统为个人自主设计，不存在任何侵权行为，可以放心使用。

使用方面，本图书系统使用简单，速度快，现有人员只需经过简单的培训即可熟练掌握其使用方法。同时，用户交互页面设计相当友好，使用体验较好。

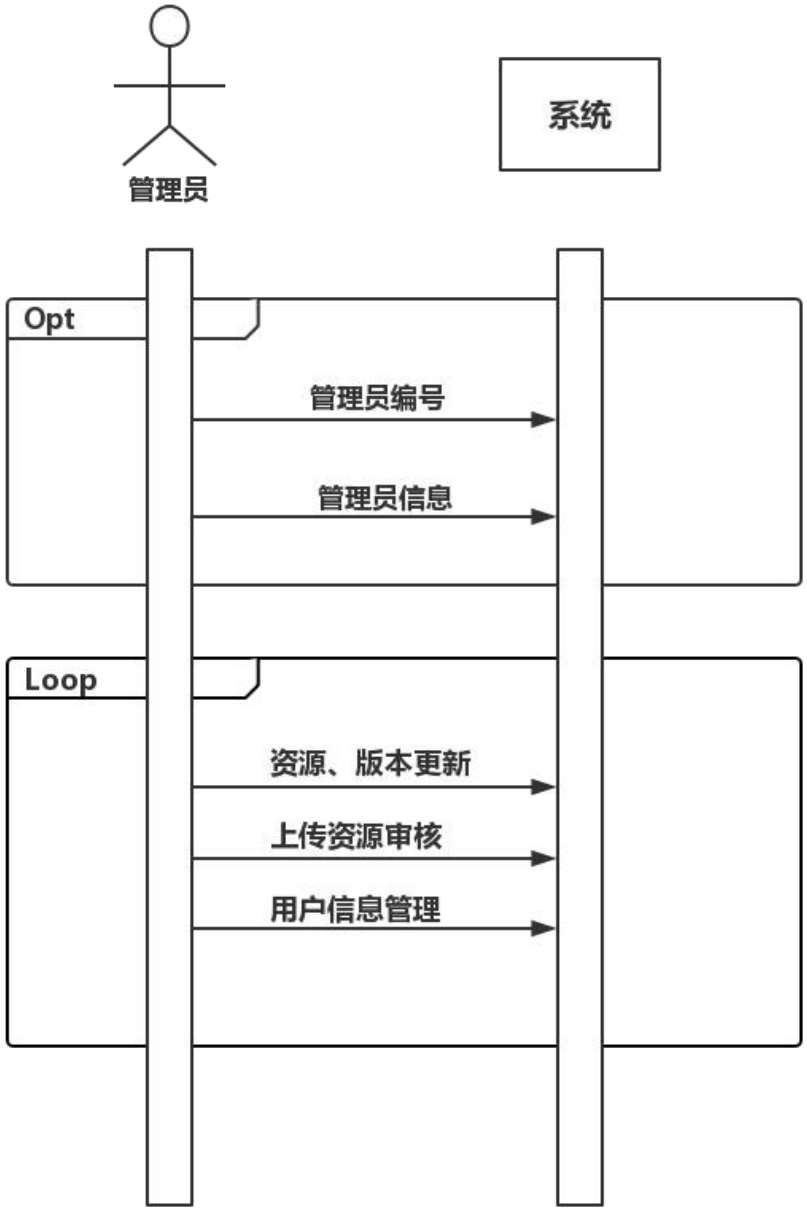
综上，通过对图书系统整体进行可行性分析，该项目无论在操作可行性、技术可行性、经济可行性、社会可行性等上均满足要求。因此开发的构想是可行的，可着手进行实施。

3、系统分析

3.1 系统角色

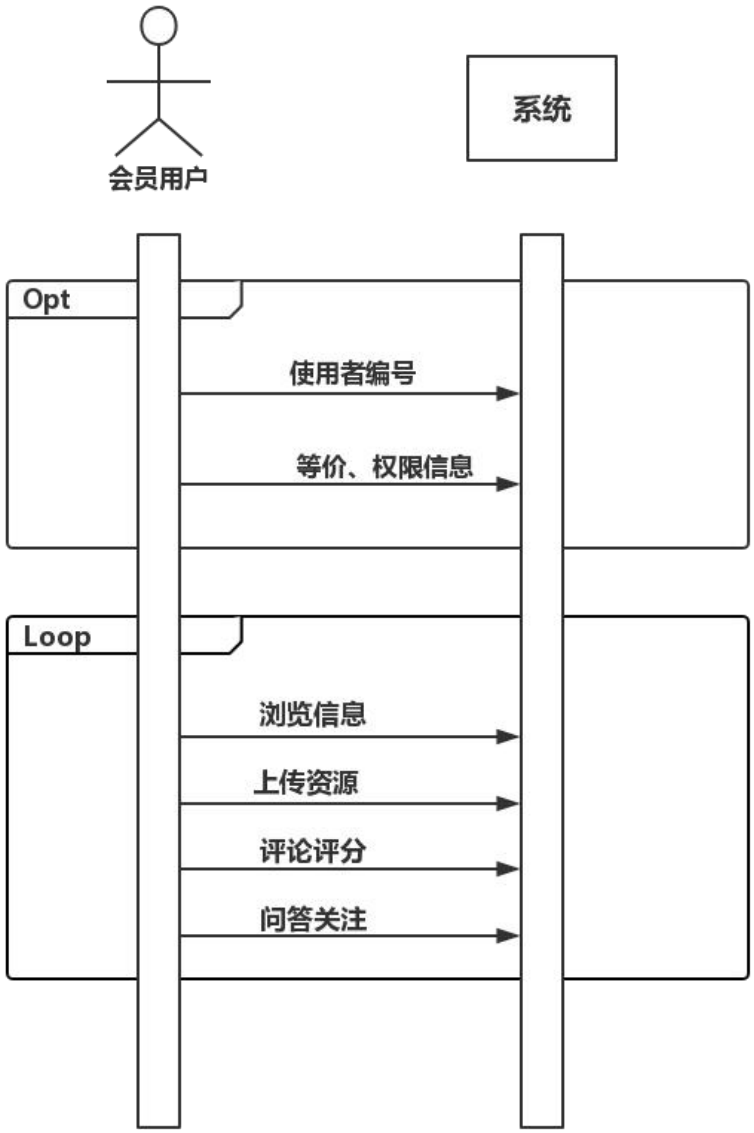
系统角色主要包含以下几类：

（1）系统管理员：完成系统管理与维护, 例如：维护用户数据、管理数据库, 进行资源更新等等；

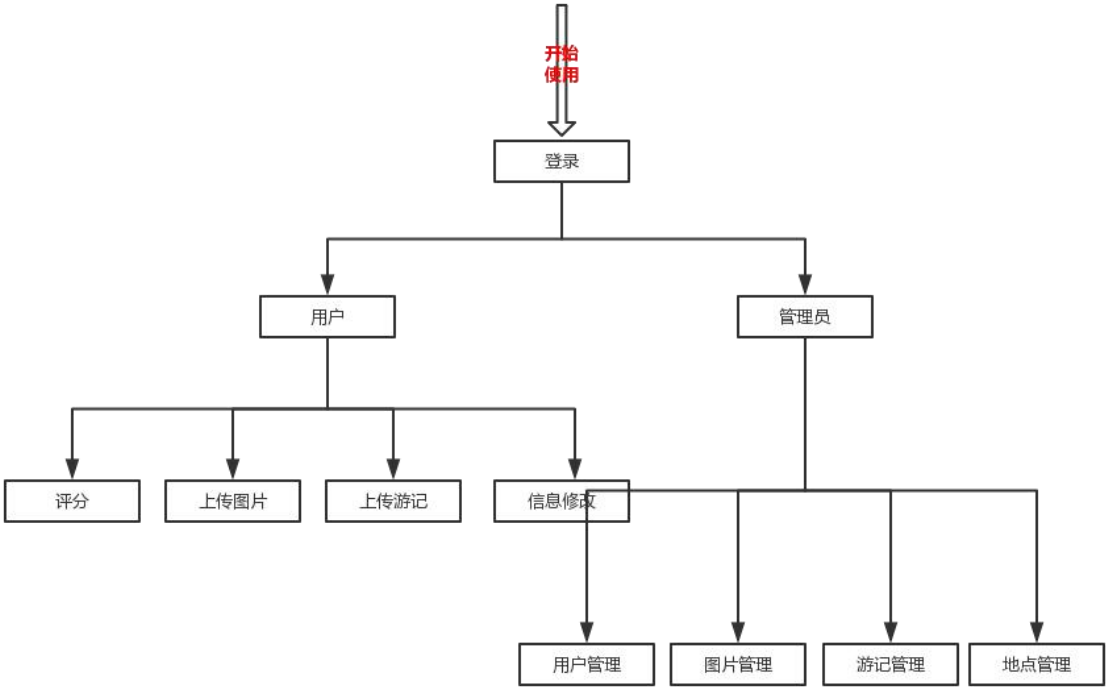


（2）会员用户：可以浏览系统“发现景点”推荐信息，使用“景点查询”查找旅行目的地；进入“我的行程”查看用户的历史出行记录，系统会根据用户

等级为用户保存大小不一的历史消息；上传资源，等级越高就越容易获得通过；给景点写评论、打评分；提问或者回答，关注或者被关注等；



3.1.2 不同角色使用图



使用的角色分为游客、管理员和用户。其中，游客只能在网站的首页面输入地点名，然后系统寻找该地点名的全部信息，包括评分，简介等返回给游客。用户可以在系统已有的地点范围内进行评分，可以上传图片，可以上传游记。管理员可以增加删除用户，增加删除地点（若删除地点，则用户关于该地点的评分会消失，但用户关于该地点的图片和游记不会消失），增加删除图片，增加删除游记。

3.2 用户信息获取与维护

3.2.1 功能说明

初次注册, 系统对用户进行旅行兴趣探究提问, 估测出用户的兴趣爱好、时间布局、景点偏向并记录至系统中, 在用户多次使用系统并给出反馈后，系统对用户的信息记录随之发生更改.

3.2.2 激励/响应序列

刺激：用户注册或初次登录

响应：给出相应问题

刺激：用户选择答案

相应：记录用户兴趣情况信息并给出推荐

3.3 景点查询

3.3.1 功能说明

根据用户输入信息进行旅行目的地的推荐, 针对不同用户的个性化喜好荐不同类型的目的地列表, 由用户进行选择。

3.3.2 激励/响应序列

刺激：用户点击“景点查询”。

响应：进入相应界面, 用户输入信息，系统基于用户信息给出推荐刺激：用户从中选择并确认。

响应：将目的地标记为“推荐成功”，完善用户属性。

刺激：用户旅行结束进行信息。

响应：基于用户反馈，更新系统数据库。

3.4 我的行程

3.4.1 功能说明

按时间顺序记录并保存用户学习的历史出行信息，包括旅游地点、及时感

想、评论和游记。

3.4.2 激励/响应序列

刺激：用户点击“我的行程”。

响应：系统搜索数据库，按照时间、地点顺序反馈。

刺激：用户选择时间、地点。

响应：给出相关结果。

3.5 发现景点

3.5.1 功能说明

系统滚动地为用户提供旅游目的地，用户还可以使用关键字，如：5 公里

海边、森林等进行限制。该模块将为用户的出行提供启发。

3.5.2 激励/响应序列

刺激：用户点击“发现景点”。

响应：滚动展示推荐的旅游点。

刺激：用户设置关键词进行限定。

响应：根据用户的关键词多数据库进行搜索。

3.6 交流圈

3.6.1 功能说明

会员用户可通过此模块查看好友、关注的人的旅游信息。系统也通过该模块

传递系统消息给用户。

3.6.2 激励/响应序列

刺激：用户点击“交流圈”。

响应：与用户有关、系统消息、好友/关注对象的动态都会显示给用户。

刺激：用户选择下一操作。

响应：系统执行相应下一步动作。

3.7 个人信息

3.7.1 功能说明

会员用户填写、完善个人信息。

3.7.2 激励/响应序列

刺激：用户点击“个人信息”。

响应：系统显示用户信息界面。

刺激：用户填写、完善个人信息。

响应：系统检验通过则保存用户信息。

3.8 系统的数据字典

在数据流图的基础上，数据字典从数据项、数据流、数据存储、数据处理、外部实体 5 个方面进行更加详细的解释。

1. 数据项

编号：I1

名称：用户 ID

简述：管理员为用户分配的 id

类型: 整型

长度：10 位

编号：I2

名称：地点 ID

简述：地点在系统内的代码

类型：整型

长度：10 位

编号：I3

名称：图片 ID

简述：图片在系统内的代码

类型：整型

长度：10 位

编号：I4

名称：游记 ID

简述：游记在系统内的代码

类型：整型

长度：10 位

2. 数据流

编号：D1

名称：游记 ID

简述：游记在系统内的代码

类型：整型

长度：10 位

数据流来源：管理

数据流去向：用户信息存储模块

数据流组成：用户 id+姓名+密码

编号：D2

名称：用户信息

简述：用户的相关信息

数据流来源：管理员添加入系统

数据流去向：用户信息存储模块

数据流组成：用户 ID+用户姓名+用户密码

编号：D3

名称：图片信息

简述：用户上传图片的相关信息

数据流来源：用户

数据流去向：图片上传记录信息存储模块

数据流组成：用户 id+地点 id+游览时间

3. 管理员数据存储：

编号：F1

名称：用户信息表

简述：对所有在本系统中注册的用户信息进行汇总存储

组成：用户 ID+用户+密码+电话号码+性别

关键字：用户 ID

编号：F2

名称：地点信息表

简述：对所有添加入本系统的地点信息进行汇总存储

组成：地点 ID+地点名

关键字：地点 ID

编号：F3

名称：图片信息表

简述：对所有添加入本系统的图片信息进行汇总存储

组成：图片 ID+用户 ID+图片名+地点 ID

关键字：图片 ID

编号：F4

名称：游记信息表

简述：对所有添加入本系统的游记信息进行汇总存储

组成：游记 ID+用户 ID+游记名+地点 ID

关键字：游记 ID

编号：F5

名称：首页

简述：显示用户、地点、游记、图片的数目及相关信息

4. 用户数据存储：

编号：F1

名称：用户上传表

简述：对该用户在本系统中在本系统中所上传的所有信息汇总存储

组成：用户 ID+用户名+密码

关键字：用户 ID

编号：F2

名称：地点上传表

简述：对当前用户所上传的所有地点信息汇总存储

组成：地点 ID+地点名+用户 ID

关键字：地点 ID

编号：F3

名称：图片信息表

简述：对当前用户上传图片信息进行汇总存储

组成：图片 ID+图片名+用户 ID

关键字：图片 ID

编号：F4

名称：游记信息表

简述：对当前用户所上传的所有游记信息汇总存储

组成：游记 ID+用户 ID+地点 ID

关键字：游记 ID

编号：F5

名称：评价

简述：当前用户对某个地点的评分，地点的综合评分为所有用户评分的平均值

组成：评分 ID+用户 ID+地点 ID+评分 score

关键字：评分 ID

5. 数据处理

编号：P1

名称：管理员添加/删除用户

输入：用户信息

处理：存储/更新用户信息

输出：用户信息表

编号：P2

名称：用户修改用户信息

输入：用户信息

处理：更新用户信息

输出：用户信息表

编号：P3

名称：搜索地点

输入：地点名

处理：检索地点信息表中相应字段进行匹配

输出：地点信息、评分、评价等信息

编号：P4

名称：增加/删除/修改地点

输入：地点 ID、地点名

处理：增加/删除/修改地点

输出：地点信息表

编号：P5

名称：增加/删除/修改图片

输入：图片 ID

处理：修改图片信息，增加/修改图片记录信息

输出：图片信息表

编号：P6

名称：增加/删除/修改游记

输入：游记 ID

处理：修改游记信息，增加/修改游记记录信息

输出：游记信息表

编号：P7

名称：增加/删除/修改评分

输入：地点 ID、评分 score

处理：更新地点评分，取平均值

输出：评分信息表

6. 外部实体

编号：S1

名称：管理员

简述：系统的维护人员

输入的数据流：用户信息、地点信息、图片信息、游记信息、评分信息

输出的数据流：信息汇总

编号：S2

名称：用户

简述：系统的使用人员

输入的数据流：用户已经上传评价的信息

输出的数据流：用户信息汇总

4、系统设计

4.1 系统配置设计

4.1.1 硬件配置设计

计算机内存：4GB

CPU：Intel Core i7　2.40GHz

4.1.2 软件配置设计

操作系统：Windows 10 64 位

开发工具：JetBrains PhpStorm 2017.3.1　Notepad++

开发语言：PHP5.6.16

数据库：Mysql 5.7

4.1.3 网络配置设计

服务器：Apache 2.4.17

4.2 系统结构设计

4.2.1 系统功能结构设计

在 3.1 节中已经给出了系统的功能划分，下面则是对系统各个功能结构的详细设计，如下图所示：

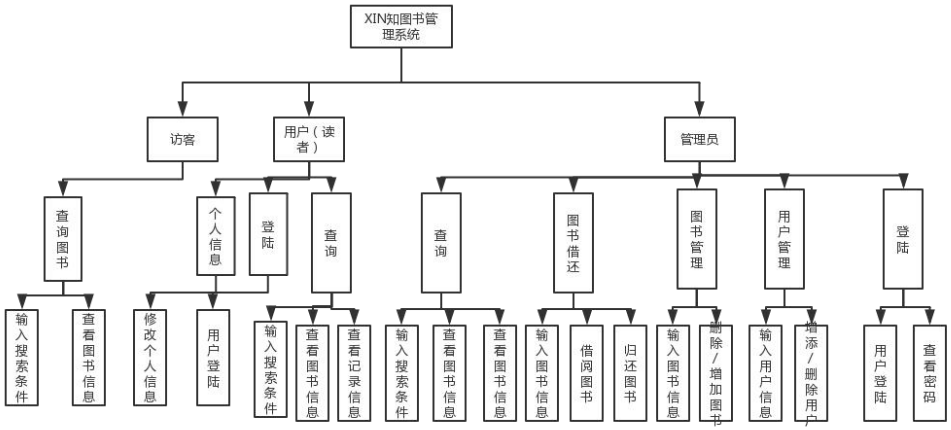


图 4.1　系统功能结构图

4.2.2 系统网络结构设计

本旅行管理系统将采用 B/S（浏览器/服务器）模式，数据存储于 Mysql 数据库中，用户/管理员/访客只需通过浏览器访问网站就可以同数据库进行交互，达到远程分布式存和查询数据以及数据管理的目的。

4.3 原始数据库设计

说明：改进的数据库很大程度上是基于原始的数据库设计。

4.3.1 确定局部应用

系统管理的资源为景点相关资源、游记资源、问答资源等众多资源，系统面向的对象为用户。系统需要做到定期更新资源，审核并管理用户上传资源。同时，也要记录用户的出行情况，更新用户的出行喜好，给用户发推送，当用户关注的人和景点有了更新应该及时提醒用户。

所以，该系统可以确定为两个局部应用：资源信息管理，用户服务管理

4.3.2 确定实体集及其属性

（1）资源信息管理

资源信息管理的实体集有:景点、相关。

各实体集的属性分别为：

景点（编号、名称、地理、评分、游记、提问回答、图片、攻略）

相关（编号、景点编号、景点名称、风俗文化、特色小吃、历史名人）

（2）用户服务管理

用户服务管理的实体集有：用户、个性化、推送

各实体集的属性分别为：

用户（编号、昵称、认证信息、历史信息、照片、关注的人、关注景点、提问回答、游记、等级 ）

个性化（用户编号、历史信息、交通方式、出行时间、旅行长度）

推送（编号、用户编号、景点编号、游记、相关问答）

4.3.3 实体集间的联系

（1）资源信息管理

景点与相关为 1 对多的关系，一个景点可以有多个相关信息。

景点与用户为多对多关系，一个用户可以关注多个景点，发表关于多个景点的游记、问答、照片，一个景点也可以有多个用户的游记、问答、照片。

景点与推送为多对多关系，一个景点可以为多个推送提供游记、问答，一个推送也可以涉及到多个景点。

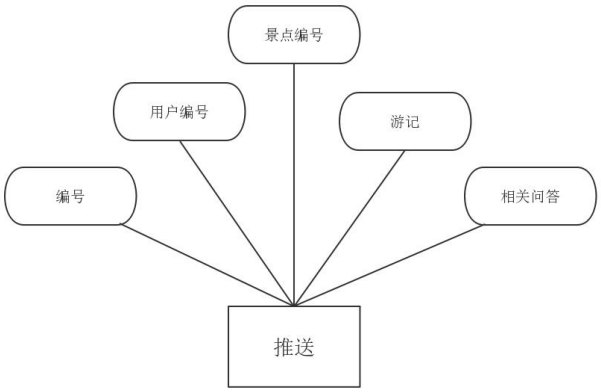
（2）用户服务管理

用户与推送为多对多关系，一个用户因为关注了多个用户和景点而收到多条推送，相同的推送也可以发送给多个用户。

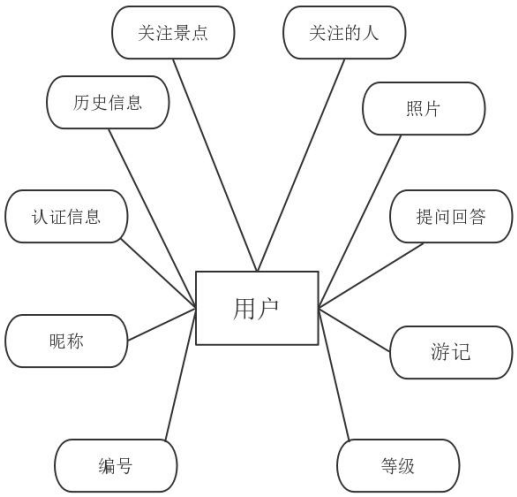
用户与个性化为 1 对 1 关系，一个用户唯一对应一种个性化出行方式

4.3.4 局部 E-R 模型

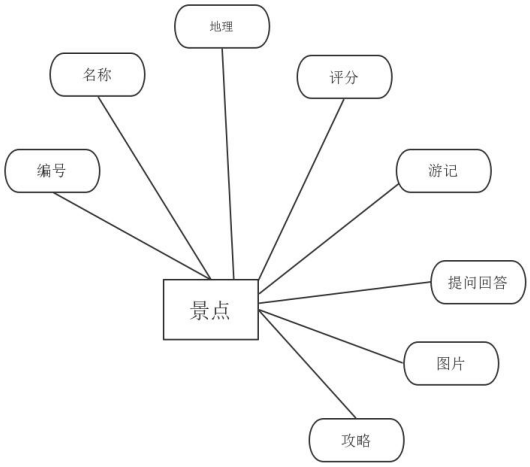
推送 ：



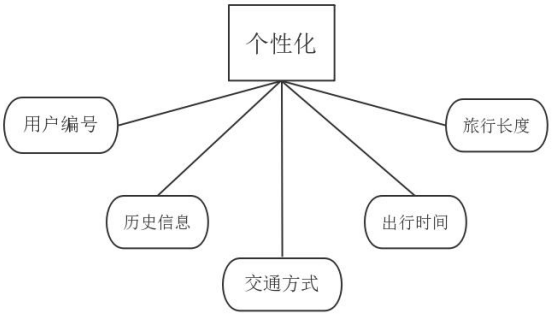
用户：



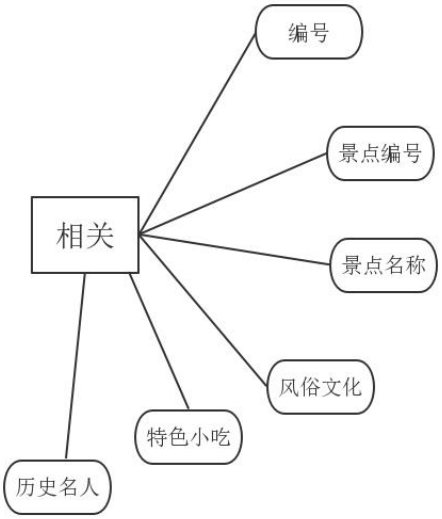
景点：


地理
名称
评分
编号
游记
景点
提问回答
图片
攻略

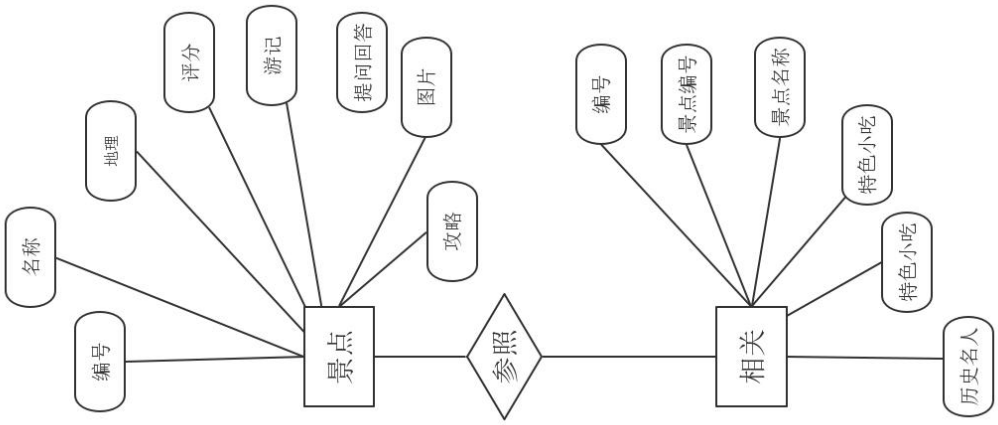
个性化：


个性化
用户编号
旅行长度
历史信息
出行时间
交通方式

相关：

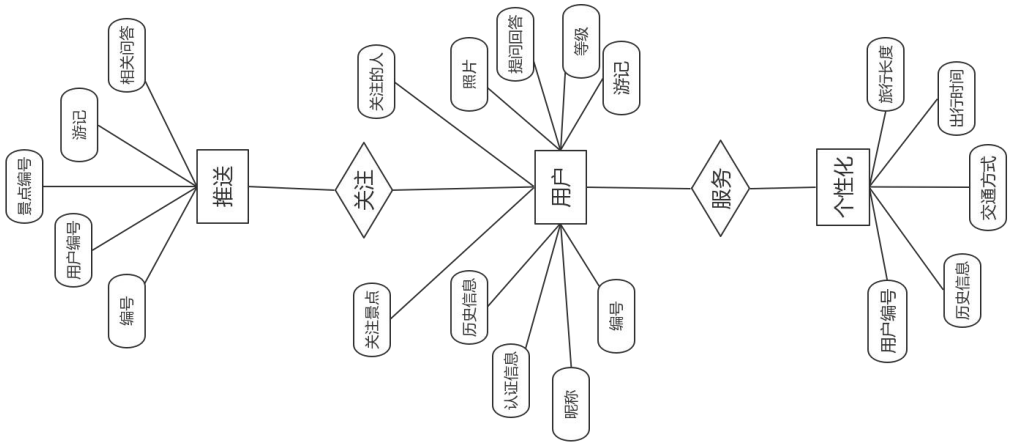

编号
景点编号
相关
景点名称
风俗文化
特色小吃
历史名人

（1）资源信息管理的 E-R 模型



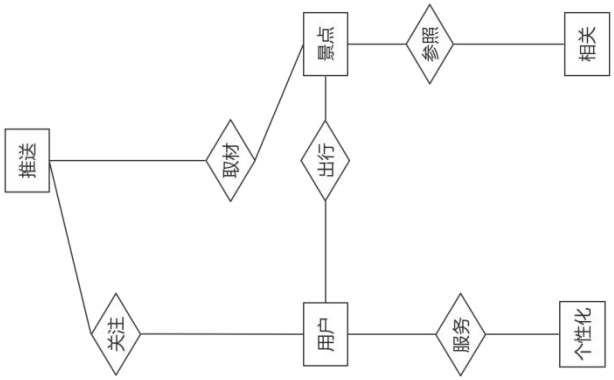
（

（2）用户服务管理的 E-R 模型



4.3.5 集成局部 E-R 模型

将资源信息管理和用户服务管理两个局部应用的 E-R 模型集成为全局 E-R 模型（去除属性），如下：



将上述 E-R 模型转换为关系模型如下：

（1）实体“用户”：

用户（编号，昵称，认证信息，历史信息，关注的人，关注景点，照片，提问回答，游记，等级）；

主键：编号

（2）实体“景点”：

景点（编号，名称，地理，评分，游记，提问回答，图片，攻略）；

主键：编号

（3）联系“出行”：

出行（用户编号，景点编号，出行编号，景点名称，图片，游记）；

主键：用户编号，景点编号，出行编号

（4）实体“推送”：

推送（编号，用户编号，景点编号，游记，相关回答）；

主键：编号

外键：用户编号，景点编号

（5）联系“关注”：

关注（用户编号，推送编号）

主键：用户编号

外键：推送编号

（6）联系“取材”：

取材（推送编号，景点编号）

主键：推送编号

外键：景点编号

（7）实体“个性化”：

个性化（编号，历史信息，交通方式，出行时间，旅行长度）

主键：用户编号

（8）联系“服务”：

服务（用户编号，个性化编号，服务编号）

主键：用户编号、个性化编号

（9）实体“相关”：

相关（编号，景点编号，景点名称，风俗文化，特色小吃，历史名人）；

主键：编号

（10）联系“参照”：

参照（景点编号，相关编号，参照编号）；

主键：景点编号，相关编号

函数依赖检测：

（1）用户（编号，昵称，认证信息，历史信息，关注的人，关注景点，照片，提问回

答，游记，等级）；

编号→昵称

编号→认证信息

编号→历史信息

编号→关注的人

编号→关注景点

编号→照片

编号→提问回答

编号→游记

编号→等级

（2）景点（编号，名称，地理，评分，游记，提问回答，图片，攻略）；

编号→名称

编号→地理

编号→评分

编号→游记

编号→提问回答

编号→图片

编号→功率

（3）出行（用户编号，景点编号，出行编号，景点名称，图片，游记）；

用户编号，景点编号，出行编号->景点名称

用户编号，景点编号，出行编号->图片

用户编号，景点编号，出行编号->游记

（4）推送（编号，用户编号，景点编号，游记，相关回答）；

编号→用户编号

编号->景点编号

编号->游记

编号->相关问答

（5）关注（用户编号，推送编号）；

用户编号->推送编号

（6）取材（推送编号，景点编号）；

推送编号→景点编号

（7）个性化（编号，历史信息，交通方式，出行时间，旅行长度）；

编号→历史信息

编号→交通方式

编号->出行时间

编号->旅行长度

（8）服务（用户编号，个性化编号，服务编号）；

服务编号，个性化编号→服务编号

（9）相关（编号，景点编号，景点名称，风俗文化，特色小吃，历史名人）；

编号→景点编号

编号→景点名称

编号→风俗文化

编号→特色小吃

编号→历史名人

（10）参照（景点编号，相关编号，参照编号）；

景点编号，相关编号→参照编号

优化：

（1）经检验，满足 1NF 要求

（2）消除部分依赖

在“联系” 出行（用户编号，景点编号，出行编号，景点名称，图片，游记）中，景点编号即可以决定景点名称，用户编号可以决定图片、游记，这样就存在部分依赖，可以将将户编号和景点编号变成外键，即：

出行（出行编号，用户编号，景点编号，景点名称，图片，游记）；

主键：出行编号

外键：用户编号，景点编号

函数依赖：

出行编号→用户编号

出行编号→景点编号

出行编号→景点名称

出行编号→图片

出行编号→游记

经过检查，其他的实体与联系都不包含部分依赖

（3）消除传递依赖

“联系”出行经过上面的优化后消除了部分依赖，但是又存在传递依赖：

出行编号->用户编号

用户编号->(图片,游记)

出行编号->景点编号

景点编号->景点名称

所以，要消除传递依赖应该在联系“出行”中去掉属性：用户编号、景点编号。“联系”出行如下：

出行（用户编号，景点名称，图片，游记）；

函数依赖：

用户编号->景点名称

用户编号->图片

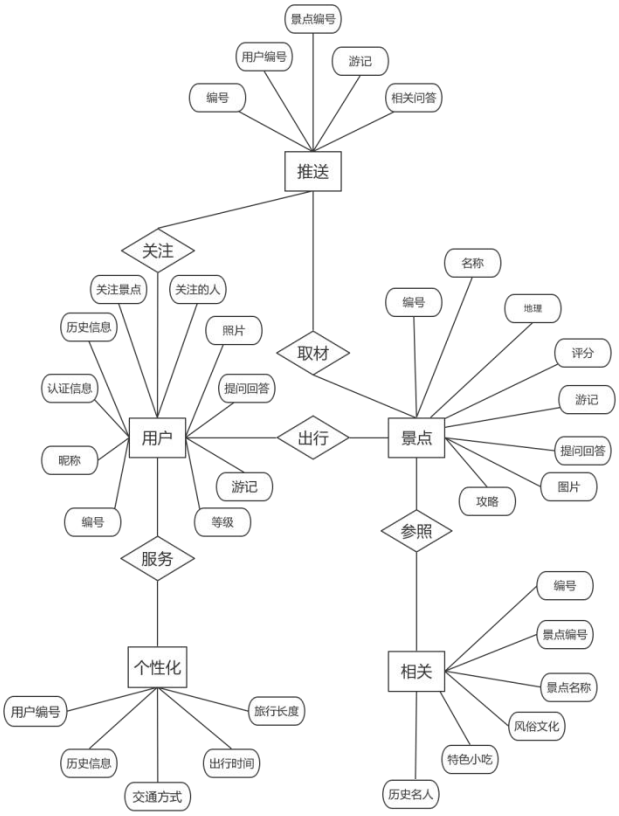
用户编号->游记

经过检验，其他部分不含有传递依赖

4.3.6 检测并消除冲突

不存在结构冲突，命名冲突。

4.3.7 总体 E-R 模型



4.4 改进数据库设计

4.4.1 改进原因及方面

上面的数据库设计是脱离了实现基础的，在开始实现的过程中，发现上面的设计过于繁琐，实现起来比较复杂且相应的界面不好看。所以就把数据库设计简化。

4.4.2 概念结构设计

根据需求分析得出的结果可知，旅行管理系统中主要涉及的实体包括：person、place、photo、diary、mark、makemark 共 6 个实体。一个用户可以游览多个地方，而一个地点可以被多用户浏览；一个用户可以上传多张照片，而一张照片只能由一名用户生成；一个用户可以上传多篇游记，而一篇游记只能由一名用户产生；一个用户关于一个地点只能做一个评分，一个地点的综合评分是由多个用户评分的平均值决定。一个地点的评分总数和总评分人数唯一。这样我们可以设计出数据库的概念数据模型，以 E-R 图的形式表示如下：

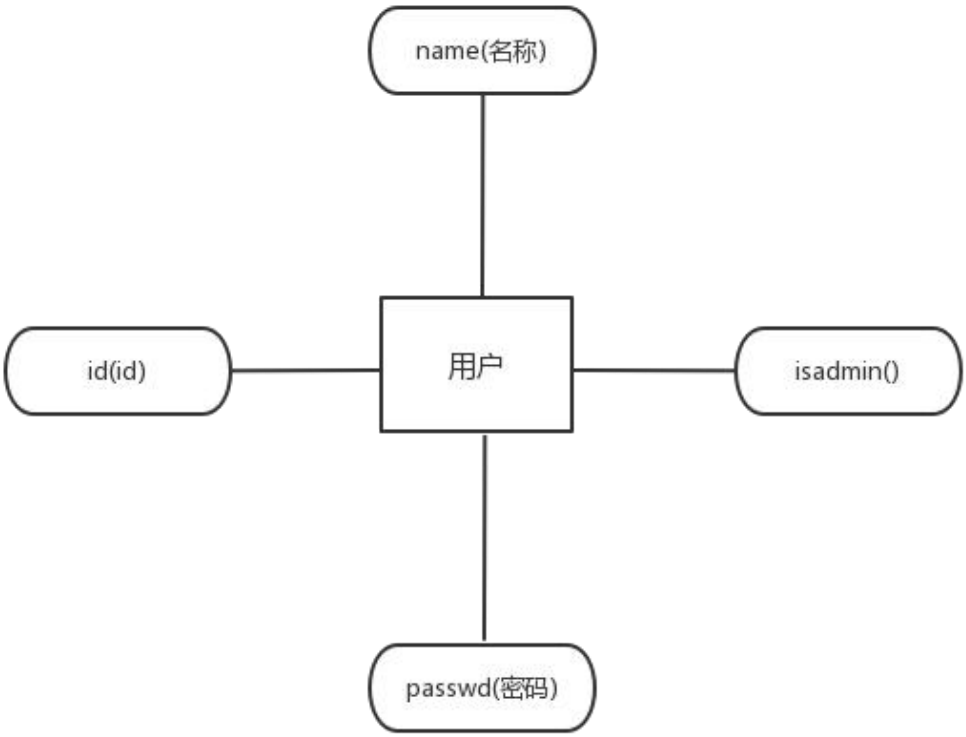
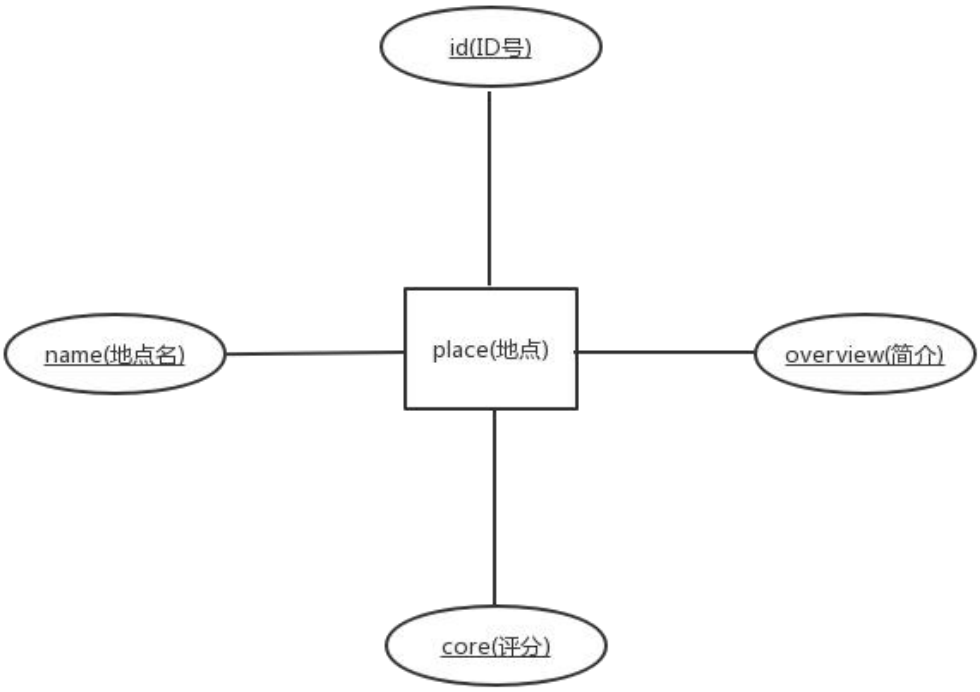
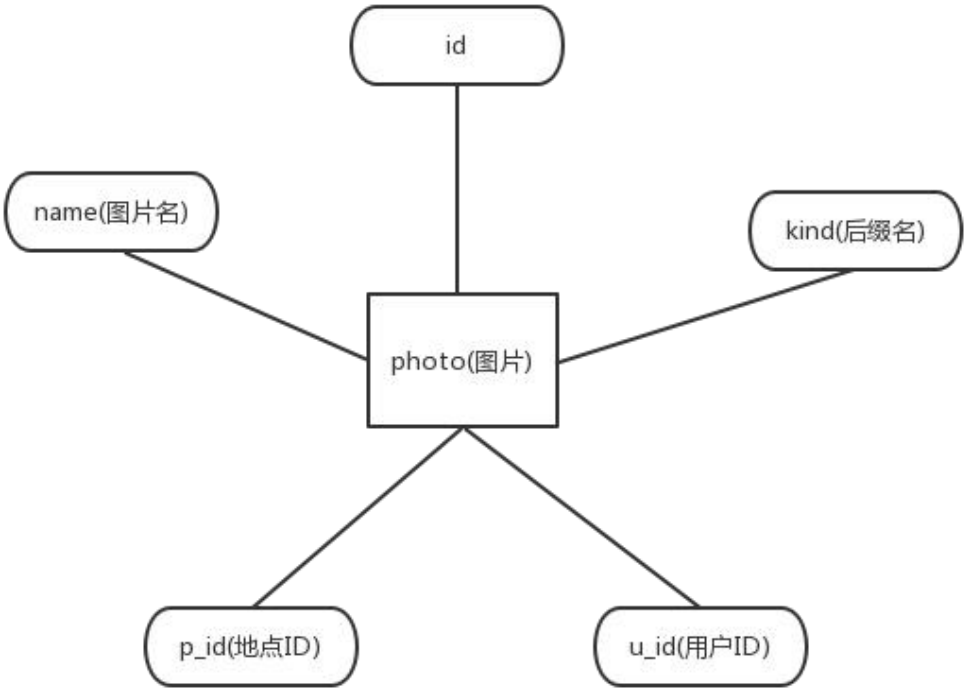


图 4.4　person E-R 图

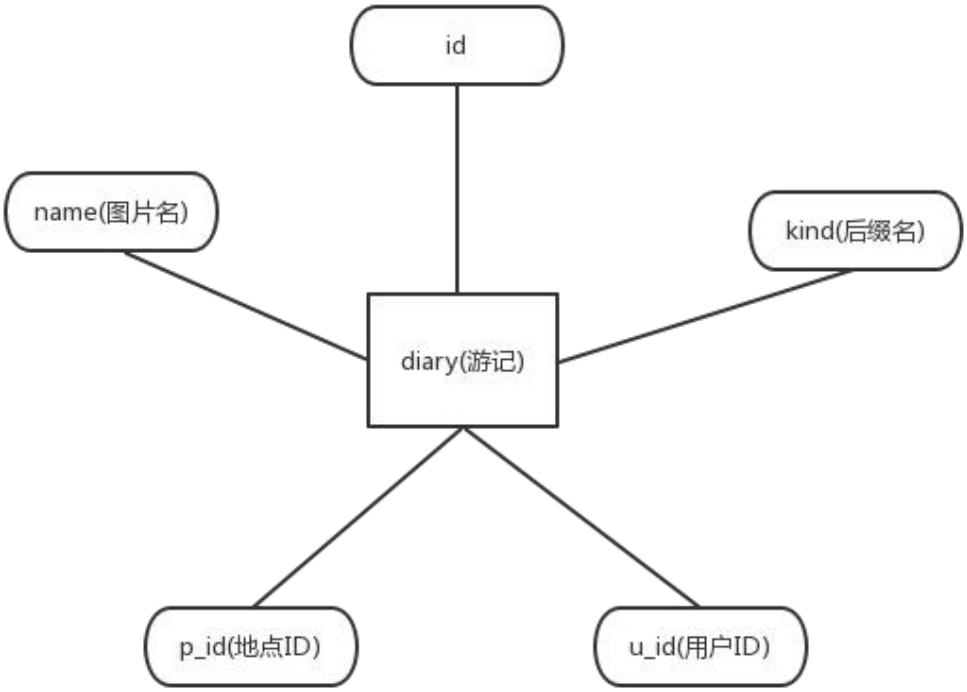
用户：（用户 ID，用户名，用户密码，是否为管理员）



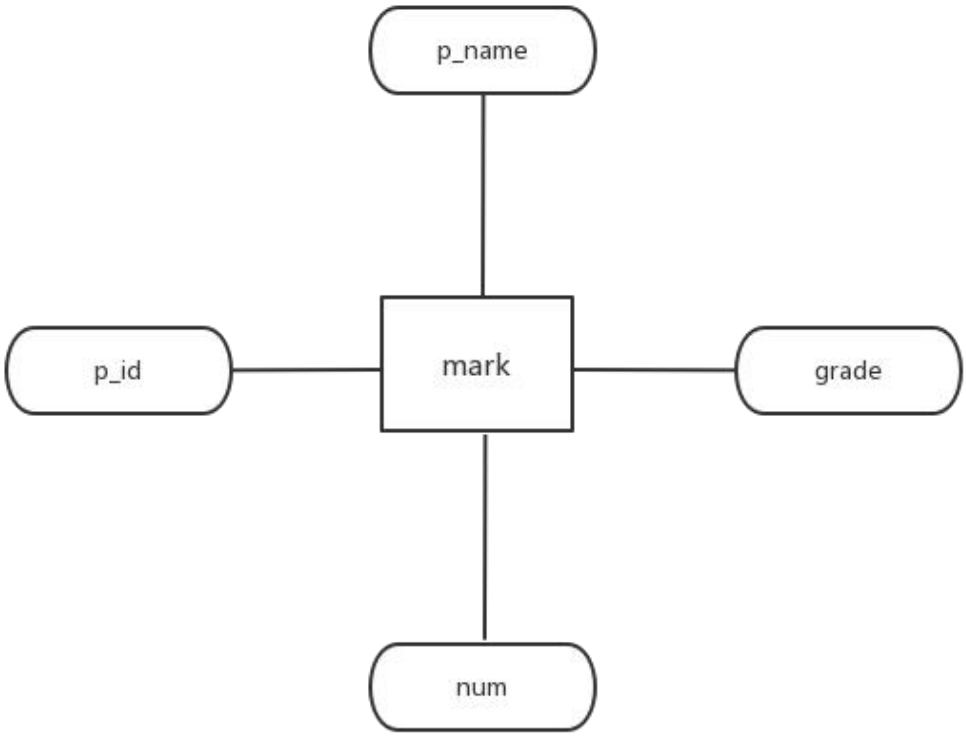
地点：（图书 ID，图书名称，出版社，作者，图价格，图书类别，馆藏位置，馆藏状态）



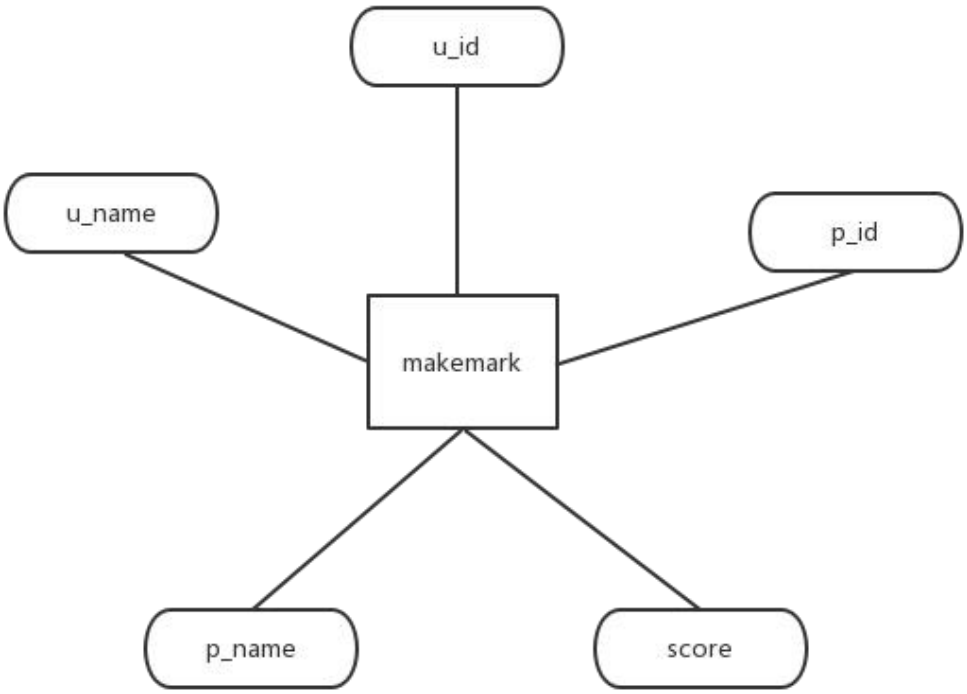
photo：（ID，name，kind，p_id， u_id）



diary：（ID，name，kind，p_id，　u_id）

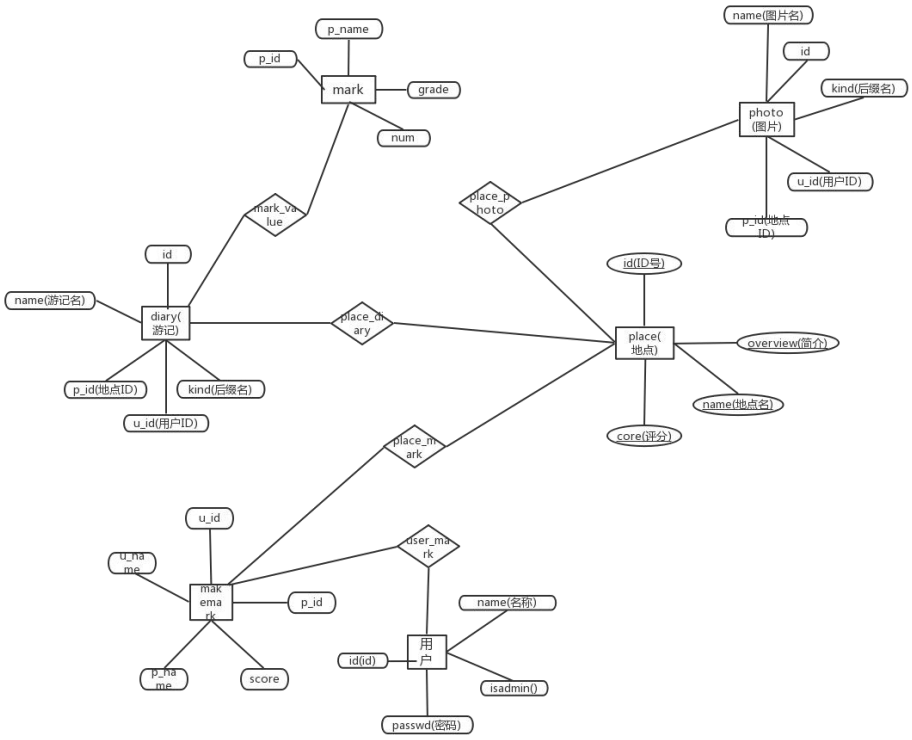


mark：（p_id，p_name，grade,num）



makemark：（u_id，u_name，p_id， p_name， score）

总 ER 图：



其中，有几点需要解释：
1、用户与管理员依靠 isadmin 来区分，用户的 isadmin=0，管理员的 isadmin=1；本系统的使用方式为，管理员向系统中添加地点及地点的简介，用户在管理员添加的地点范围内上传照片，上传游记，并且评分。管理员的信息是在向 mysql 中导入数据库时就写好的，因而管理员的信息及数目都是固定的。用户可以通过管理员向系统中增加和删除。
2、Place 的简介类型为`overview` varchar(150) COLLATE utf8_unicode_ci，支持中英文，要注意 overview 的长度限制。管理员才有权限增加删除地点，用户无权。
3、Photo 和 diary 都有 kind，p_id，u_id 属性，kind 是指图片或游记的文件形式的后缀，比如.jpn、.txt；p_id，u_id 是地点 id 和用户 id，表示图片和游记所在的地点和上传的用户。因而有外键：

```
constraint fk_pid1 foreign key (p_id) references person(id),
constraint fk_uid1 foreign key (u_id) references place(id)
```

4、mark 有属性 p_id，grade，num。p_id 指评分的地点的 id，grade 是所有评分用户关于该地点的总评分，num 是对该地点评分的总用户数目（一个用户关于一个地点只能评分一次）。
5、Makemark 是记录用户关于某个地点是否评分的。当用户在网页上对某个地点评分后，系统就会自动触发，记录用户的该动作，避免用户的二次评分。

为了清楚说明，将数据库的代码复制如下：

```
create table `person`
(
    `id` integer not null,
    `name` varchar(20) not null,
    `passwd` varchar(20) not null,
    `isadmin` integer,
    constraint pk_id primary key (id)
)ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

-- ----------------------
insert into person values(1,'root','1122',1);
insert into person values(2,'hsj','1122',0);
insert into person values(3,'xxc','1122',0);
insert into person values(4,'fzp','1122',0);

-- ----------------------
create table `place`
(
    `id` integer not null,
    `name` varchar(20) COLLATE utf8_unicode_ci not null,
    `core` float not null,
    `overview` varchar(150) COLLATE utf8_unicode_ci,
    constraint pk_id primary key (id)
)ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

```
-- ----------------------

-- ----------------------
create table `photo`
(
    `id` integer not null,
    `name` varchar(20) COLLATE utf8_unicode_ci not null,
    `kind` varchar(20) COLLATE utf8_unicode_ci not null,
    `p_id` integer,
    `u_id` integer,
    constraint pk_id primary key (id),
    constraint fk_pid1 foreign key (p_id) references person(id),
    constraint fk_uid1 foreign key (u_id) references place(id)
)ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

create table `diary`
(
    `id` integer not null AUTO_INCREMENT,
    `name` varchar(20) COLLATE utf8_unicode_ci not null,
    `kind` varchar(20) COLLATE utf8_unicode_ci not null,
    `p_id` integer,
    `u_id` integer,
    constraint pk_id primary key (id),
    constraint fk_pid2 foreign key (p_id) references person(id),
    constraint fk_uid2 foreign key (u_id) references place(id)
)ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

create table `mark`
(
    `p_name` varchar(20) not null,
    `p_id` integer not null,
    `grade` float,
    `num` integer,
    constraint fk_pid3 foreign key (p_id) references place(id)
)ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

create table `makemark`
(
    `u_id` integer not null,
    `u_name` varchar(20) not null,
    `p_id` integer not null,
    `p_name` varchar(20) not null,
    `score` float not null
```

)ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

4.4.2 逻辑结构设计

将概念结构设计阶段完成的概念模型转换成能被数据库管理系统 DBMS 支持的数据结构模型，同时尽量减少不必要的数据冗余，消除修改异常、插入异常和删除异常，使数据库的关系模式规范到符合第三范式的条件，在 XIN 知图书管理系统中，概念模型转换为数据模型如下所示：

（1）一个实体转换为一个关系模式：

用户：（ID，name，passwd，isadmin）

地点：（ID，name，core，overview）

图片：（ID，name，kind，p_id，u_id）

游记：（ID，name，kind，p_id，u_id）

Mark：（p_id，p_name, grade, num）

makemark：（u_id，u_name, p_id, p_name,score）

（2）一对多联系的关系模式：

图片：（ID，p_id，u_id）,对应用户 id 和地点 id

游记：（ID，p_id，u_id）,对应用户 id 和地点 id

makemark：（u_id, p_id），对应用户 id 和地点 id

利用以上 E-R 图转换到关系模式的方法，可以得到关系数据表如下所示：

表 4-1　用户信息表：

列名	数据类型	数据长度	可否为空	备注
ID	int	10	Not null	用户 ID（主键）
Kind	varchar	10	NOT Null	图书类别
Name	varchar	10	NotNull	用户名
Passwd	varchar	10	notNull	用户密码（初始为空）
Isadmin	Int	3	Not Null	是否为管理员

表 4-2　地点信息表：

列名	数据类型	数据长度	可否为空	备注
ID	Int	10	Not null	地点 ID（主键）
Name	varchar	10	null	地点名
Core	Float	20	null	用户评分(初始为 0)
Overview	varchar	500	null	地点简介

表 4-3　图片信息表：

列名	数据类型	数据长度	可否为空	备注
ID	Int	10	Not null	图片 ID（主键）
Name	varchar	10	Niot null	图片名称
Kind	varchar	20	Not null	图片格式后缀
u_id	Int	10	Not null	上传图片用户的 id
p_id	int	10	Not null	图片关联的地点的 id

表 4-4　游记信息表：

列名	数据类型	数据长度	可否为空	备注
ID	Int	10	Not null	游记 ID（主键）
Name	varchar	10	Niot null	游记名称
Kind	varchar	20	Not null	游记格式后缀
u_id	Int	10	Not null	上传游记用户的 id
p_id	int	10	Not null	游记关联的地点的 id

表 4-5　mark 信息表：

列名	数据类型	数据长度	可否为空	备注
P_id	Int	10	Not null	地点 ID
P_Name	varchar	20	Niot null	地点名
grade	Int	10	Not null	地点的总评分
Num	Int	10	Not null	评分的用户数

表 4-6　makemark 信息表：

列名	数据类型	数据长度	可否为空	备注
P_id	Int	10	Not null	地点 ID
P_Name	varchar	20	Not null	地点名
u_id	Int	10	Not null	用户 ID
u_Name	varchar	20	Not null	用户名
Score	Int	10	Not null	用户评分

4.4.3 物理结构设计

1、选择关系模式的存取方法：

在数据库逻辑结构的设计中，我们已经建立起了 6 张数据表主要用来响应快速查询，在物理设计中需要确定它们的存取方法：在地点信息表中，由于地点 ID 号属性唯一决定每个地点元组，所以对关系模式建立以 placeID 号为主关键字的索引，即对地点信息模式采取索引存取方法。同理，对用户信息模式、图片信息模式、游记信息模式均采取索引存取方法。

2、数据库存储结构设计：

考虑到平台可能会频繁的被同学们访问和查询，而且要求以较快的速度响应查询，因

此建立以 placeID 为关键字的索引列，由于其频繁检索特性，因此建立聚簇索引，使得被索引的记录在磁盘物理分配空间中连续存放。建立索引的过程如下所示：

```
use SecondBook
go

create clustered index bookID on books(BookID);
```

4.5 系统界面设计

在旅行系统平台的界面设计中，由于不同角色功能不同，主要分为游客使用界面、用户使用界面和管理员后台管理界面和访客界面 3 个部分，以下为主要功能的界面截图展示：

4.5.1 游客界面设计

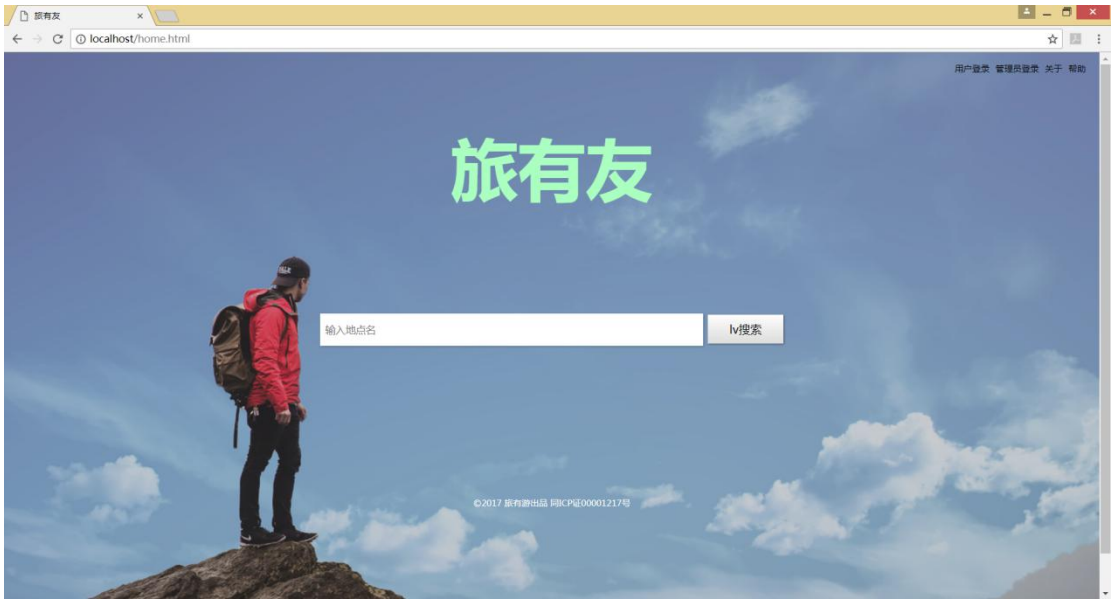


图 4.1　旅行系统首页

图 4.2　旅行系统游客搜索反馈界面

进入网站的初始化界面，游客可以直接在输入框中输入城市名，然后获取相关城市的信息。上面例子是输入上海后返回的搜索结果。

4.5.2 用户界面设计

登录界面

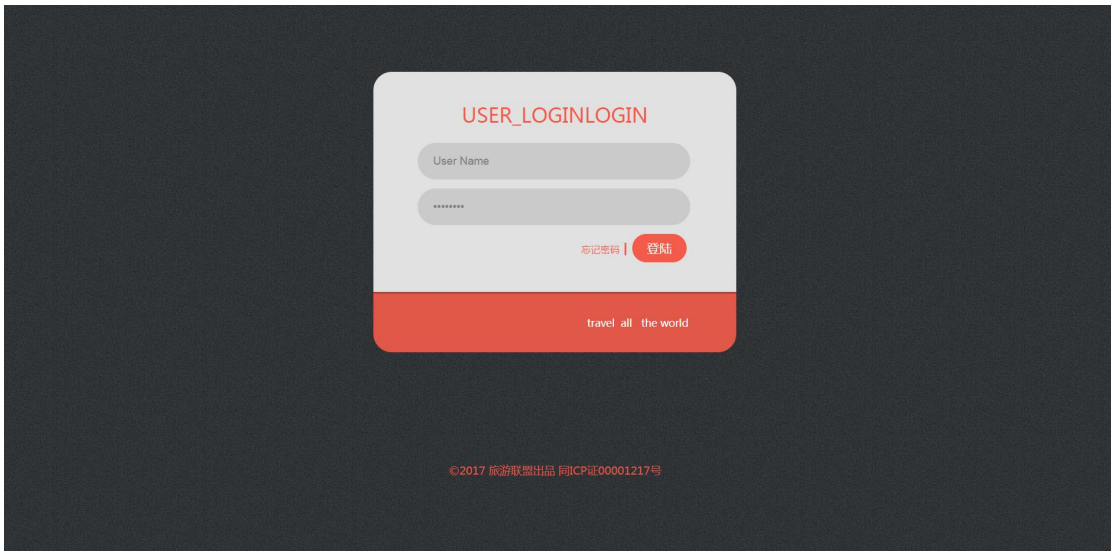


图 4.3　用户登录界面

图 4.4　用户主界面

显示用户的评分信息、上传图片信息、上传游记信息。

评分界面

图 4.4　用户评分界面

图片界面

图 4.5　用户图片界面

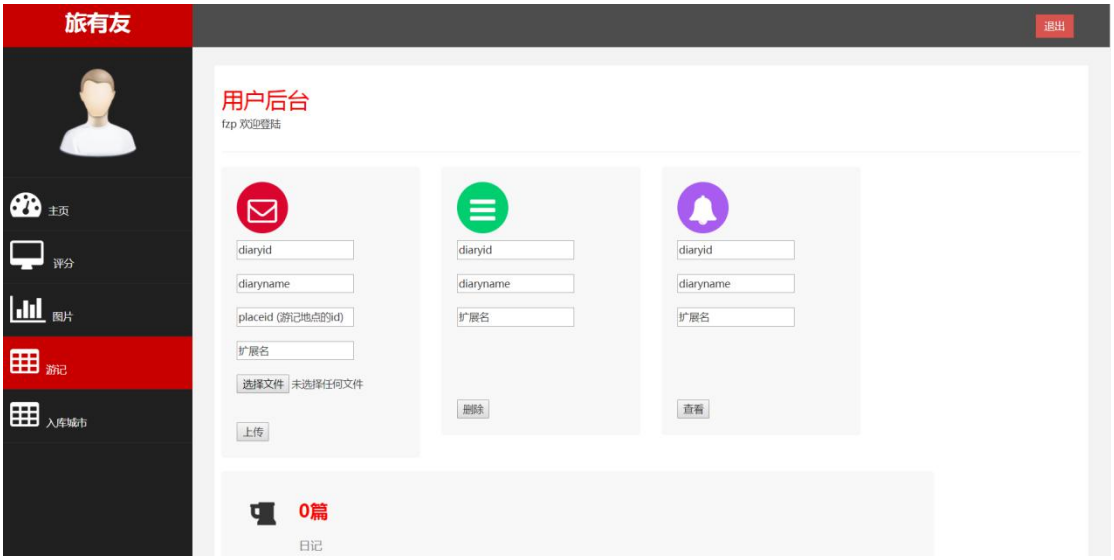


图 4.6　用户图片界面

图 4.7　入库城市界面

在评分界面用户可以评分，也可以删除评分。在图片界面中，用户可以上传图片，进行该操作后，数据库的 photo 中会增加一条记录并且系统的 photo 文件夹中会增加用户上传的图片；用户也可以删除图片，相应的，数据库的 photo 表格中的记录和系统 photo 文件夹里的图片都会被删除；用户可以查看图片，在线查看图片。游记功能与图片类似，不再赘述。用户可以在入库城市界面查看管理员建立的城市。下面结合图片详细介绍。

评分功能展示

id号	名称	评分	简介
1	上海	5.2	上海，简称“沪”或“申”，是中国共产党的诞生地，中华人民
			科技创新中心，首批沿海开放城市。

图 4.8　用户评分前结果显示

在用户 hsj 评分之前，上海的评分为 5.2，评分人数为 1

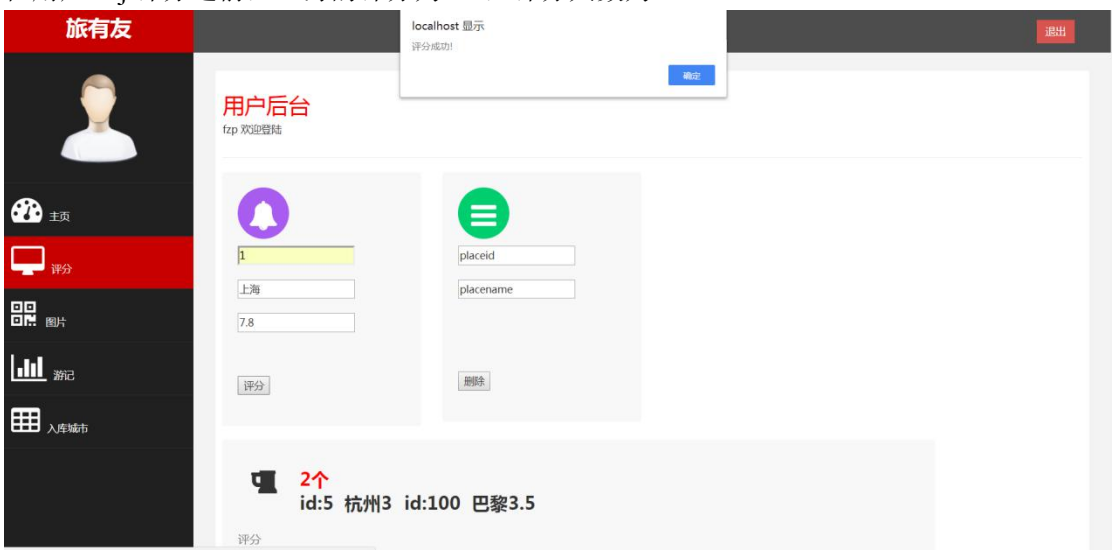


图 4.9　用户评分

id号	名称	评分	简介
1	上海	6.5	上海，简称“沪”或“申”，是中国共产党的诞生地，中华人民
			科技创新中心，首批沿海开放城市。

图 4.9　用户评分后结果显示

用户 hsj 对上海进行评分成功后，上海的评分变成 6.5，评分人数为 2

上传图片功能展示

图 4.10　用户上传图片界面

图 4.11　用户上传图片文件夹结果显示

图 4.12　用户上传图片数据库结果显示

上传图片‘黄浦江.jpg’后，可以看到在系统的 photo 文件夹中已经上传了图片‘黄浦江.jpg’，并且在数据库的 photo 表格中也有了 黄浦江.jpg 的记录

在线查看图片功能展示

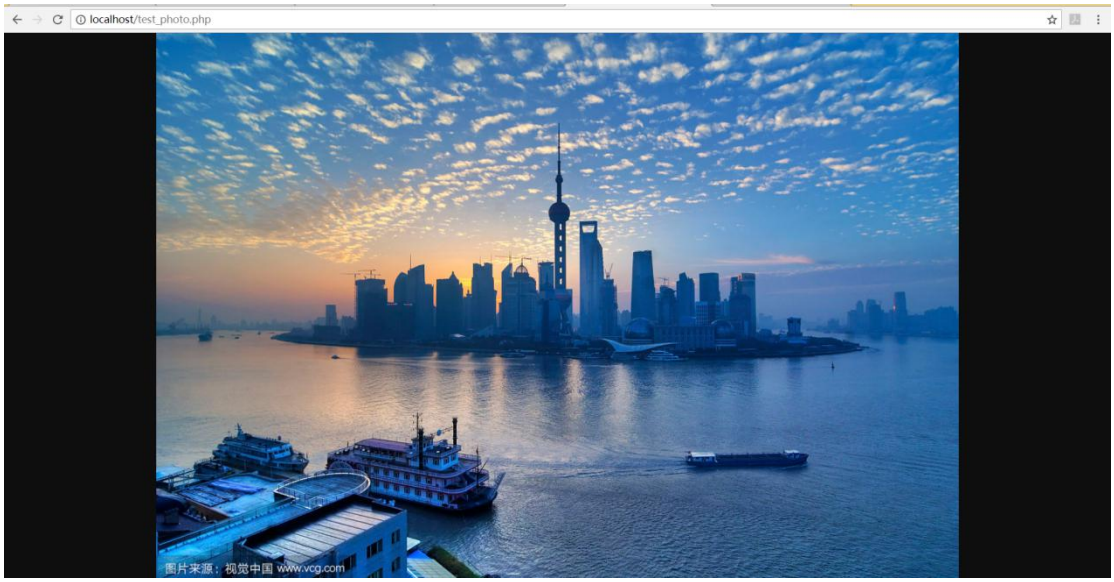


图 4.13　用户在线观看图片显示

上图为在线查看图片的效果。使用另外一个用户登录后，删除刚刚的用户上传的图片，会报错，因为每个用户只有删除自己上传的图片和游记的权限。

非法删除报错功能展示

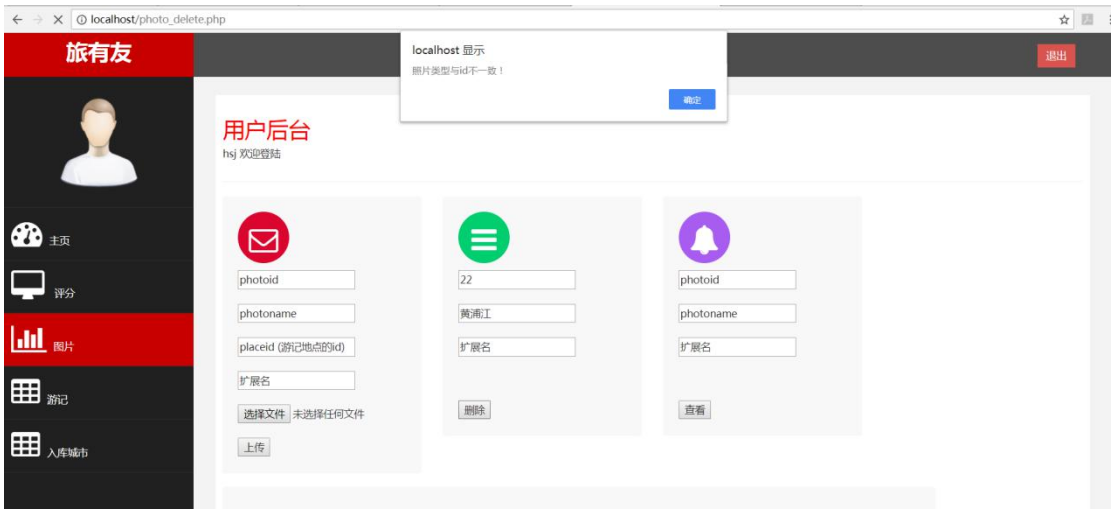


图 4.14　用户非法删除图片报错显示

上传游记功能展示

图 4.15　用户上传游记功能展示 1

图 4.16　用户上传游记功能展示 2

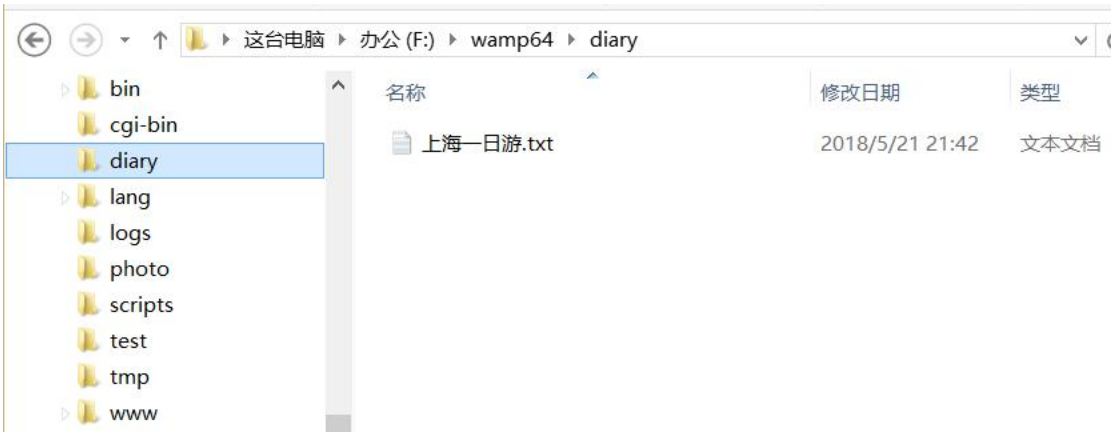


图 4.17　用户上传游记功能展示 3

游记功能的其他操作和图片功能类似，就不再赘述了。

查看入库城市功能展示

图 4.18　用户查看入库城市

用户可以在入库城市中查看管理员录入系统的城市。

4.5.3 管理员界面设计

图 4.19　后台管理登陆首页（即管理员登陆页面）

图 4.20　登陆成功页面

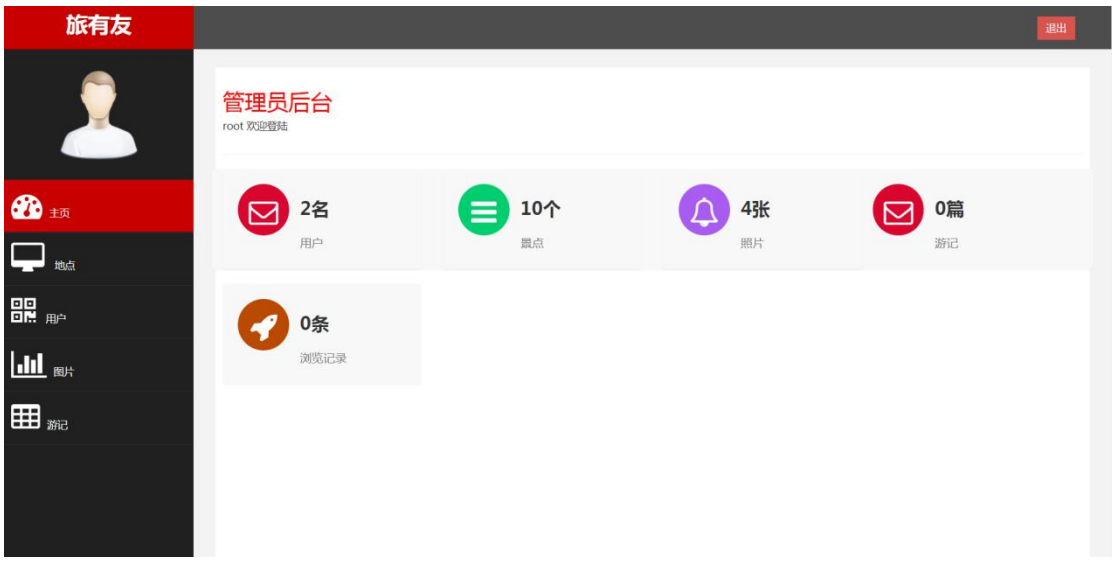


图 4.21　管理员后台主页

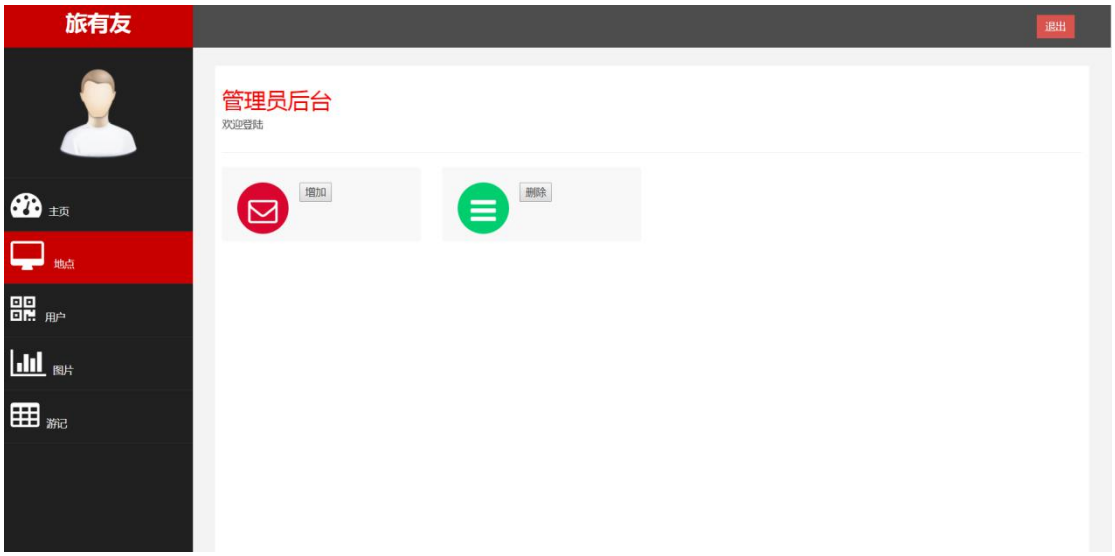


图 4.22　管理员地点页面

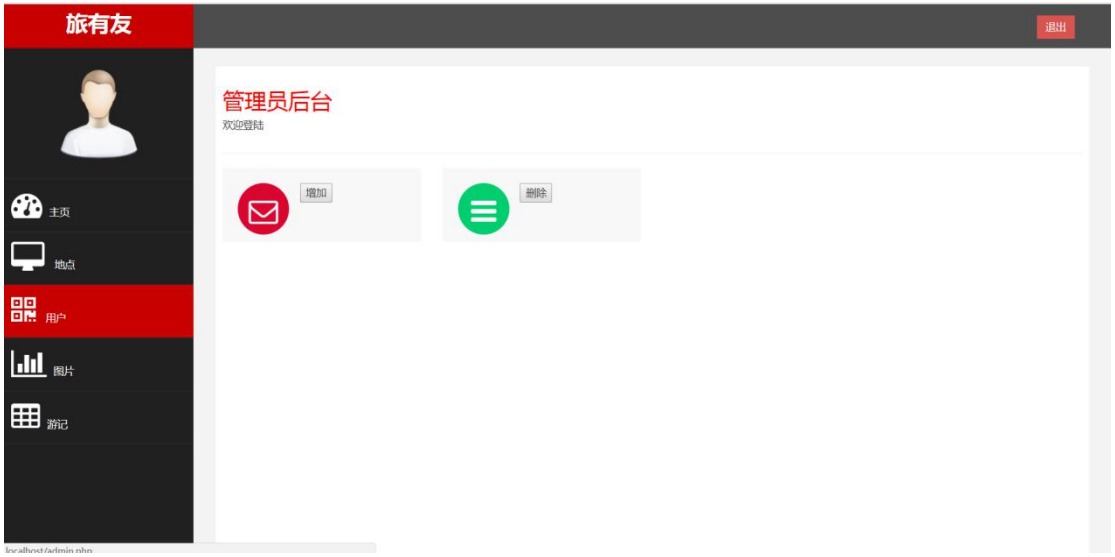


图 4.23　管理员用户管理页面

图 4.24　管理员图片管理页面

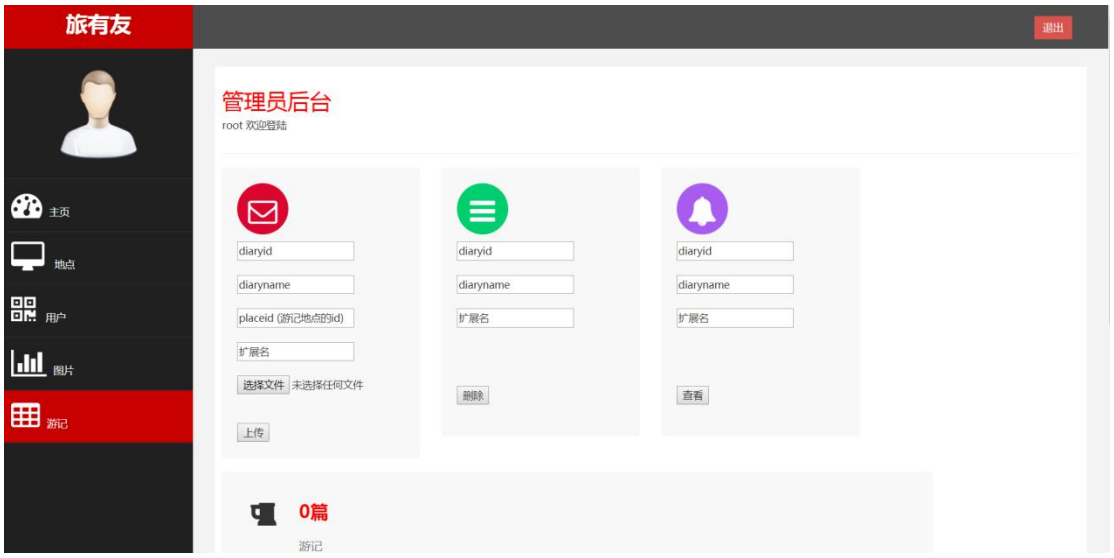


图 4.25　管理员游记管理页面

图 4.26　管理员找回密码界面

管理员的“用户”功能展示

图 4.27　管理员增加用户前

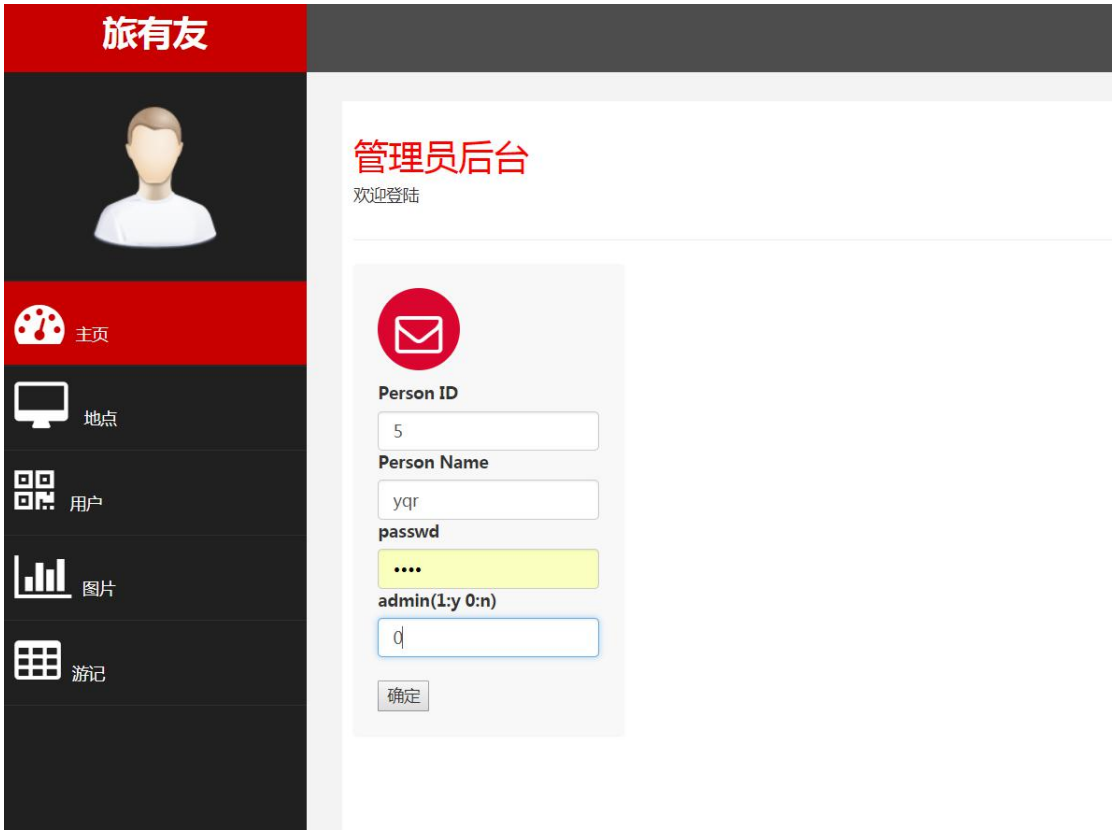


图 4.28　管理员增加用户操作

图 4.29　管理员增加用户后

管理员的地点操作类似。这里就不再赘述了。

管理员的“图片”功能展示

管理员添加图片操作同用户相同。管理员可以删除任何图片，不要求必须是自己上传的。

图 4.30　管理员删除图片前

图 4.31 管理员删除图片操作

+ 选项

	id	name	kind	p_id	u_id
编辑 复制 删除	1	上海图片1	.jpg	1	4
编辑 复制 删除	100	上海图片2	.jpg	1	1
编辑 复制 删除	101	上海图片3	.jpg	1	1

图 4.32 管理员删除图片后

“黄浦江.jpg”是由用户 hsj 上传的图片，而管理员可以删除。

4.6 代码设计（主要程序代码片断）

4.6.1 连接访问数据库

使用 mysql 数据库，建立数据库名为 travel，登录数据库用户名为 root，密码为空。使用了两种方式连接服务数据库。

方式一：使用 mysql PDO 对象访问方式进行数据库操作

```php
<?php
 $result=null;
 $db = new PDO('mysql:host=localhost;dbname=travel','root','');
```

图 4.31 mysql PDO 访问数据库

方式二：使用 mysqli 对象访问方式进行数据库操作

```php
<?php
$servername="localhost";
$username="root";
$userpassword="";
$dbname = "travel";
$id = $_POST['perid'];
$name = $_POST['pername'];
$passwd = $_POST['perpasswd'];
$isadmin = $_POST['isadmin'];
$flag = 0;
$db = new mysqli($servername,$username,$userpassword,$dbname);
```

图 4.33 mysqli 访问数据库

4.6.2 用户/管理员登录功能代码设计

用户登录功能主要是通过读取用户名和密码，与数据库中存储的用户信息是否匹配，若匹配则登录成功，转至相应页面，否则登录失败。

```php
<?php
$username=$_POST['username'];
$password=$_POST['password'];
$db = new PDO('mysql:host=localhost;dbname=library','root','123456');
$statement=$db->prepare('select * from admin where username=:username');
$statement->execute([':username'=>$username]);
$user=$statement->fetch();
if(empty($user)){
   echo "<script>alert('用户不存在！请重新登录'); history.go(-1);</script>";
}

else if($password != $user['password']){
    echo "<script>alert('密码不正确！请重新登录'); history.go(-1);</script>";
}
else{
    $_SESSION['user']=$username;
    echo "<script>alert('登录成功!进入选择业务主页');window.location.href='admi

    //echo "<a href=\"admin.php\">点击进入选择业务主页</a>";
}
?>
```

图 4.34 用户登录

```php
<?php
$username=$_POST['id'];
$password=$_POST['u_password'];
$db = new PDO('mysql:host=localhost;dbname=library','root','123456');
$statement=$db->prepare('select * from users where id=:username and isDelete =0');
$statement->execute([':username'=>$username]);
$user=$statement->fetch();
if(empty($user)){

    echo "<script>alert('用户不存在！请重新登录'); history.go(-1);</script>";

}
else if($password != $user['password']){

    echo "<script>alert('密码不正确！'); history.go(-1);</script>";
    }
else{
    $_SESSION['userid']=$username;

    echo "<script>alert('登录成功!进入选择业务主页');window.location.href='user.php
    //header('refresh:1;url=home.html');
   // echo "<a href=\"admin.php\">点击进入选择业务主页</a>";
}
?>
```

图 4.35 管理员登录

4.6.3 用户图片功能代码设计

上传图片功能：

通过用户输入的图片名、图片 id、地点 id 和图片扩展名，执行 SQL 语句在数据库中插入该图片，并且在系统的文件夹 photo 中增加该图片。

```php
while($photo1=mysqli_fetch_object($statement))
{
    if($photo1->id==$id || $photo1->name==$name)
    {
        echo "<script>alert('照片名称或者id已经使用！');</script>";
        if($isadmin == 1)
            echo "<script>window.location.href='photo_manage.php';</script>";
        else
            echo "<script>window.location.href='user_photo_manage.php';</script>";
        $flag=1;
        break;
    }
}
if($flag==1)
{
    echo "<script>alert('添加失败!');</script>";
    if($isadmin == 1)
        echo "<script>window.location.href='photo_manage.php';</script>";
    else
        echo "<script>window.location.href='user_photo_manage.php';</script>";
}
else{

        #存储上传照片至../photo

if ((($_FILES["file"]["type"] == "image/gif")
    || ($_FILES["file"]["type"] == "image/jpeg")
    || ($_FILES["file"]["type"] == "image/jpg")
    || ($_FILES["file"]["type"] == "image/pjpeg")
    || ($_FILES["file"]["type"] == "image/png"))

if ($_FILES["file"]["error"] > 0)
{
    echo "Return Code: " . $_FILES["file"]["error"] . "<br />";
}
```

图 4.36 用户上传图片

其中，php 代码中使用 select 语句在数据库中查询是否已经存在该图片名。通过 move_uploaded_file 函数将图片从给定路径下移动至 photo 文件夹中。

删除图片功能：
根据用户输入的图片名、图片 id、图片扩展名，使用 select 语句，在数据库中搜寻相应的图片信息。判断当前用户是否有删除照片的权限。若有，则从数据库和 photo 文件夹中，将图片删除。

```php
if($photo1->u_id!=$userid && $isadmin == 0)
{
    echo "<script>alert('您不具备删除该照片的权限！');window.location.href='user_photo_manage.php'; </script>";
    $flag=0;
    break;
}
```

图 4.34 判断用户权限

```php
$deletedata="delete from photo where id = $id";
$statement=$db->query('select * from photo');
```

图 4.35 用户删除图片 1

```php
if($db->query($deletedata)==true)
{
    echo "<script>alert('删除成功!');</script>";
    if($isadmin == 1)
        echo "<script>window.location.href='photo_manage.php';</script>";
    else
        echo "<script>window.location.href='user_photo_manage.php';</script>";
    mysqli_free_result($statement);
}
else
{
     echo "<script>alert('图片删除，数据库信息没有清除！!');</script>";
     if($isadmin == 1)
        echo "<script>window.location.href='photo_manage.php';</script>";
    else
        echo "<script>window.location.href='user_photo_manage.php';</script>";
     #echo "Error insert data: " . $db->error;
}
```

图 4.36 用户删除图片 2

4.6.4 用户评分功能

用户点击评分按钮，系统会判断用户是否已经评价过该地点。若是，报错。

```php
while($isable=mysqli_fetch_object($getabletomark))
{
    $firstuser = 1;
    $flag = 1;
    if($isable->p_id = $id1)
    {
        echo "<script>alert('您已经评价过！');window.location.href='score_manage.php'; </script>";
        $flag=0;
        break;
    }
}
```

图 4.37 用户已经评分

用户符合评分条件，相应的地点 评分人数属性（num）加一，总评分属性（score）加上用户该次评分。重新计算平均分，并且使用 update 语句把 place 的分数属性更新。记录用户该次评分行为到 makemark 中，这样，下次用户重复评分就会报错。

```php
$num = $num + 1;
$grade = $grade + $score;
echo $num;
echo $grade;
$newcore = $grade / $num;
#$insert_makemark = "insert into makemark(u_id,u_name,p_id,p_name,score) values ($userid,$username,$id1,$name1,$score)";
#$res2 = $db->query($insert_makemark);
$update_mark = "UPDATE mark SET grade = $grade, num=$num WHERE p_id = $id1";
#$insert_makemark = "insert into makemark(u_id,u_name,p_id,p_name,score) values ($userid,$username,$id1,$name1,$score)";
$update_place = "UPDATE place SET core = $newcore WHERE id = $id1";
$res1 = $db->query($update_mark);
$insertmakemark="insert into makemark(u_id,u_name,p_id,p_name,score) values($userid,\"$username\",$id1,\"$name1\",$score)";
$db->query($insertmakemark);
$res3 = $db->query($update_place);
mysqli_free_result($getinfo);
mysqli_free_result($getabletomark);
echo "<script>alert('评分成功!');window.location.href='score_manage.php';</script>";
#window.location.href='score_manage.php';
```

图 4.38 用户评分

```php
$pre_getscore = "select score from makemark where u_id = $userid and p_id = $id1";
$getscore = $db->query($pre_getscore);
$tempscore = mysqli_fetch_object($getscore);
$deletable = 1;
if($tempscore==false)
{
    $deletable = 0;
    echo "<script>alert('您尚未评分!');window.location.href='score_manage.php';</script>";
}
if($deletable == 1)
{
$score = $tempscore->score;
$pre_getinfo = "select grade,num from mark where p_id = $id1";
$getinfo = $db->query($pre_getinfo);
if($getinfo==false)
{
    echo "读取info失败！ ";
}
$tempinfo = mysqli_fetch_object($getinfo);
$num = $tempinfo->num;
$grade = $tempinfo->grade;
$num = $num - 1;
$grade = $grade - $score;
if($num == 0)
    $newcore = 0;
else
    $newcore = $grade/$num;
$deletemakemark="delete from makemark where u_id = $userid and p_id = $id1" ;
$update_mark = "UPDATE mark SET grade = $grade, num=$num WHERE p_id = $id1";
$update_place = "UPDATE place SET core = $newcore WHERE id = $id1";
if($db->query($update_mark)==true && $db->query($update_place)==true &&$db->query($deletemakemark)==true){
        #echo "插入数据成功";
        mysqli_free_result($getscore);
        mysqli_free_result($getinfo);
        echo "<script>alert('删除成功!');window.location.href='score_manage.php';</script>";
    }
    else{
        echo "Error : " . $db->error;
    }
```

图 4.39 用户删除评分功能

使用 select 语句搜索 makemark，判断用户之前是否有过关于关于该地点的评分。若用户评分行为合法，则从 makemark 中删除该次评分行为，相应的地点 评分人数属性（num）减一，总评分属性（score）减去用户该次评分。重新计算平均分，并且使用 update 语句把 place 的分数属性更新。

4.6.6 显示用户信息

用户进入主页后，该系统可以显示用户的评分、图片、游记信息：

图 4.40 用户主页

为了实现网页交互的功能，将 HTML 与 PHP 结合，通过 POST 方式实现网页数据的传递。如下图所示：PHP 与 html 网页嵌套使用，<from action=’’, method=’POST’>

```
<div class="text-box" >

      <?php
        $result=null;
        $userid = $_SESSION['id'];
$result=$db->query("select * from makemark where u_id = $userid");
if($result)
{
     $marknum=$result->fetchAll(PDO::FETCH_ASSOC);
     echo '<p style="color:red"; class="main-text">'.count($marknum).'个</p>';
     echo "<br>";echo "<br>";echo "<br>";echo "<br>";
     for($i=0;$i<count($marknum);$i++)
     {
         if($i==0)
         {
         echo " ";echo " ";echo " ";echo " ";}
         echo '<span class="main-text" style="font-size:18px;">'."id:".$marknum[$i]['p_id']." "." ".$marknum[$i]['p_name']." "." ".$marknum[$i]
     }
}
else
    echo '<p style="color:red"; class="main-text">0个</p>';
      ?>
<p class="text-muted">评分</p>
```

图 4.41php 与 html 嵌套代码

4.6.5 管理员用户管理功能

```
$insertdata="insert into person(id,name,passwd,isadmin) values($id,\"$name\",\"$passwd\",$isadmin)";
$statement=$db->query('select * from person');
if($statement==false)
{
    echo "读取数据失败！ ";
}

while($person1=mysqli_fetch_object($statement))
{
    if($person1->id==$id || $person1->name==$name)
    {
        echo "<script>alert('地点名称或者id已经使用！ '); </script>";
        $flag=1;
        break;
    }
}
if($flag==1)
{
    echo "<script>alert('添加失败!');window.location.href='place_add.html';</script>";
}
else{
    if($db->query($insertdata)==true){
        #echo "插入数据成功";
        mysqli_free_result($statement);
    }
    else{
        echo "Error insert data: " . $db->error;
    }
    echo "<script>alert('添加成功!');window.location.href='person_manage.html';</script>";

    //echo "<a href=\"admin.php\">点击进入选择业务主页</a>";
}
mysqli_close($db);
?>
```

图 4.42 管理员增加用户功能

首先使用 select 语句，遍历用户表格，判断用户名和用户 id 是否已经使用。若合法，则使用 insert 语句将用户添加到用户信息表中。

```php
<?php
$servername="localhost";
$username="root";
$userpassword="";
$dbname = "travel";
$id1 = $_POST['perid'];
$name1 = $_POST['pername'];
$flag = 1;
#echo '$id1='.$id1;
#echo '$name1='.$name1;
#echo '$overview1='.$overview1;
#$db = new PDO('mysql:host=localhost;dbname=travel','root','');
$db = new mysqli($servername,$username,$userpassword,$dbname);
$deletedata="delete from person where id = $id1";
$deletephoto="delete from photo where u_id = $id1";
$deletediary="delete from diary where u_id = $id1";
$statement=$db->query('select * from person');
if($statement==false)
{
    echo "读取数据失败！";
}

while($person1=mysqli_fetch_object($statement))
{
    if($person1->id==$id1)
    {
        $flag = 0;
        if($person1->name != "$name1")
        {
            echo "<script>alert('地点名称与id不一致！'); </script>";
            $flag=1;
            break;
        }
    }
}
if($flag==1)
{
    echo "<script>alert('删除失败!');window.location.href='person_del.html';</script>";
}
else{
    if($db->query($deletephoto)==true && $db->query($deletediary)==true&& $db->query($deletedata)==true){
        echo "<script>alert('删除成功!');window.location.href='person_manage.html';</script>";
```

图 4.43 管理员删除用户功能

首先使用 select 语句，遍历用户表格，判断用户名和用户 id 是否已经使用。若合法，则使用 delete 语句将用户从用户信息表中删除。

4.6.6 文件功能表

由于功能较多，但代码结构相似，对于 php 文件不再一一赘余结束，具体作用参看下表：

类别	Php 文件名	功能	
admin	Login_admin.php Login_admin.html	管理员登陆	
	Admin.php	管理员主页	
	Person_manage.php Person_manage.html	管理员用户管理主页面	
	Person_add.php	管理员添加用户页面	
	Person_del.php	管理员删除用户页面	
	Place_manage.php	管理员地点管理主页面	
	Place_add.php	管理员增加地点页面	
	Place_del.php	管理员删除地点页面	
	Photo_manage.php	管理员图片管理主页面	

	Photo_add.php	管理员增加图片页面	
	Photo_del.php	管理员删除图片页面	
	Loginout.php	登出	
	Diary_manage.php	管理员游记管理主页面	
	Diary_add.php	管理员增加游记页面	
	Diary_del.php	管理员删除游记页面	
Others	Help.html	系统使用界面	
	About.html	开发者信息界面	
User	User_longin.php	用户登录界面	
	user.php	用户主界面	
	score_manage.php	用户评分管理主页面	
	Score_add.php	用户增加评分页面	
	Score_del.php	管理员删除评分页面	
	user_diary__manage .php	用户游记管理主页面	
	user_photo__manage .php	用户图片管理主页面	
	Show_city.php	展示入库城市界面	
Visitor	Help.php	游客查询查询城市界面	

表 1　函数功能表

5、系统测试

5.1 系统测试环境

5.1.1 测试的硬件环境

- 计算机内存：4GB
- CPU：Intel Core i7　2.40GHz

5.1.2 测试的软件环境

- 操作系统：Windows 8 64 位
- 数据库：Mysql
- 浏览器：Google Chrome，Microsoft Edge

5.1.3 测试的网络环境

- 服务器：Apache 2.4.17　Wampserver

5.2 系统测试内容

旅有友旅行管理系统将对以下功能进行测试：

游客功能测试：
（1）游客搜索城市功能：测试游客输入城市名，是否能够搜索出正确的结果并予以显示。
（2）帮助功能：测试游客能否查看帮助信息。
（3）开发者信息功能：测试游客能否查看开发者信息。

管理员功能测试：
（1）验证管理员登录功能：管理员用户登录时测试是否能够正常的登录本系统。
（2）管理员主页信息：测试管理员主页是否显示系统的用户，地点，游记，图片等信息。
（3）管理员用户管理功能：测试管理员能否正确添加/删除用户相关信息至数据库中，如果用户非法，是否报错。
（4）管理员地点管理功能：测试管理员能否正确添加/删除地点相关信息至数据库中，如果地点非法，是否报错。
（5）管理员图片管理功能：测试管理员能否正确添加/删除图片相关信息至数据库及 photo

文件夹中，是否对于任何用户上传的图片都可以删除。

（6）管理员游记管理功能：测试管理员能否正确添加/删除游记相关信息至数据库及 diary 文件夹中，是否对于任何用户上传的游记都可以删除。

（7）管理员登出功能：测试管理员能否登出系统。

用户功能测试：

（1）验证用户登录功能：管理员用户登录时测试是否能够正常的登录本系统。

（2）用户评分功能：测试用户能否正确给地点评分至数据库中；对于已经评价过的地点，重复评分是否报错；评分完毕后，是否会自动更新相关地点的评分属性；用户删除评分后，是否会自动更新相关地点的评分属性。

（3）用户图片管理功能：测试用户能否正确添加/删除图片相关信息至数据库及 photo 文件夹中；是否只能删除自己上传的图片；对于用户非法删除是否报错。

（4）用户游记管理功能：测试用户能否正确添加/删游记除图片相关信息至数据库及 diary 文件夹中；是否只能删除自己上传的游记；对于用户非法删除是否报错。

（5）用户登出功能：测试管理员能否登出系统。

（6）用户主页信息：测试用户主页是否显示该用户评分，上传图片，上传游记的信息。

5.3 系统测试方法

由于测试主要是针对系统的功能进行测试，测试是否每个功能都是完善、具有可操作性的，而不太关心系统的内部结构，因此选用黑盒测试方法。

5.4 系统测试设计

5.4.1 管理员/用户登陆功能测试

测试名称	XIN 知图书管理系统——登陆功能测试		
测试用例编号	#1		
测试内容	验证管理员/用户能否成功登陆		
测试输入数据	用户名：root 密码： 1122 用户名:hsj 密码：1122		
预期结果	当输入为空时，系统提示“输入为空”；当密码输入错误时，系统提示“登陆失败”；当管理员输入正确的数据信息后，系统提示“登陆成功，进入业务主页面”，并转至管理员主页；当用户输入正确的数据信息后，系统提示“登陆成功，进入业务主页面”，并转至用户主页；		

测试结果	当输入为空时，系统提示“输入为空”；当密码输入错误时，系统提示“登陆失败”；当管理员输入正确的数据信息后，系统提示“登陆成功，进入业务主页面”，并转至管理员主页；当用户输入正确的数据信息后，系统提示“登陆成功，进入业务主页面”，并转至用户主页； 测试成功！

5.4.2 访客搜索图书功能测试

测试名称	Lv 有友——游客搜索城市功能测试
测试用例编号	#2
测试内容	验证系统是否能够根据访客的输入结果，成功搜索并且在页面上显示出来
测试数据	上海
预期结果	访客输入后，页面成功跳转并显示结果。
测试结果	系统显示搜索结果如下： id号 | 名称 | 评分 | 简介 1 | 上海 | 5.2 | 上海，简称“沪”或“申”，是中国共产党的诞生地，中华人民 科技创新中心，首批沿海开放城市。 测试成功！

5.4.3 用户评分功能测试

测试名称	Lv 有友——游客搜索城市功能测试			
测试用例编号	#3			
测试内容	验证系统能否按照用户的评分条件，记录相应操作， 并且将相应城市的评分更新			
测试数据 （搜索条件）	城市名称	上海		
	城市 id	1		
	评分	5.0		
预期结果	当用户输入完整的评分信息时，系统提示评分成功/失败。当输入的评分信息已经存在时，提示评分失败，编号重复；当输入的评分信息不存在时，系统提示“评分成功!”，并进行相应改名			
测试结果	（已经评价过）当输入评分信息，ID 为 1 时，城市名为“上海”，评分为“5.0”系统提示“已经评价过，评分失败!”，当删除该次评价后再次评分，系统提示，评分成功。查看数据库，发现上海的评分已经修改为 5.0（评分之前为 0）. 当输入删除图书的 ID 为 1000 时，系统提示“编号不存在，删除失败!” 测试成功！			

5.4.4 用户图片功能测试

测试名称	Lv 有友——游客搜索城市功能测试			
测试用例编号	#4			
测试内容	验证系统能否按照用户的图片信息，上传图片到数据库及 photo 文件夹			
测试数据（搜索条件）	图片名称	黄浦江		
	图片 id	1		
	类型	.jpg		
预期结果	当用户输入完整的评分信息时，系统提示评分成功/失败。当输入的评分信息已经存在时，提示评分失败，编号重复；当输入的评分信息不存在时，系统提示“评分成功!”，并进行相应改名			
测试结果	（尚未上传） 当输入图片信息：ID 为 1 时，城市名为“黄浦江”，类型为“.jpg”系统提示”图片上传成功!”。查看数据库，发现图片已经上传到数据库。查看 photo 文件夹，发现图片已经在该文件夹下。 当输入删除图片的 ID 为 1000 时，系统提示“编号不存在，删除失败!” 测试成功!			

5.5 系统测试结果分析

主要对用户评分，用户图片测试结果进行展示。因为这两个操作较其他操作更加复杂。以上测试可以表明系统功能运行良好，具有交互性，与数据库的连接良好，暂未出现系统崩溃、异常错误等情况。并且系统界面友好，UI 设计比较合理，在各浏览器上测试页面均正常无误。

6、遇到的问题及解决方法

6.1 开发环境配置错误

在本机环境编写代码调试过程中，使用了 wampserver 进行调试，在刚开始的阶段，无法正常打开 php 文件和 html 文件。
经查阅资料，发现为 wampserver 的文件目录配置问题，修改了 httpd.conf 中的根目录路径及访问权限，之后重启启动 httpd 服务，成功解决该问题。

6.2 中文编码格式错误

建立数据库.sql 文档时，存入数据时中文乱码，及网页显示中文乱码。

MySQL 会出现中文乱码的原因可能是:

1.server 本身设定问题，例如还停留在 latin1

2.table 的语系设定问题(包含 character 与 collation)

3.客户端程式的连线语系设定问题

统一数据库和 php 的语系设定，即建库和建表时都使用相同的编码格式，php 代码中也规定相同的编码格式。

6.3 php 与 mysql 数据库连接错误

Php 代码编写初期，选用 php 中的 mysql_query 语句进行 mysql 数据库操作，但运行之后始终无法成功连接数据库，多次检查代码，发现代码交互逻辑等正确无误。后仔细查看 php 错误日志，发现 Mysql_query 语言报 warning 警告，原因是当前版本的 php（>5.0）不支持 mysql_query，将其视为安全威胁。

查阅最新的 php 技术手册后发现有 mysqli_query 和 PHP PDO 对象可供使用，最终选择 PHP PDO 方式进行 mysql 数据库操作。

6.4 网页 session 设置问题

出于对网页安全的保护，在 PHP 开发中，使用了 session 的方式，保存用户登陆信息，并在用户访问各功能页面即需要权限的页面时对 session 进行检测，若 session 错误则无法访问。在个别页面访问时发现，报 session 相关错误，经检查发现，由于页面嵌套访问，导致多次调用 session 检查模块，多次开始 session_start() 操作。修改页面访问逻辑及验证次序即可解决。

7、课程设计总结

本次数据库课程设计耗时长达近 3 个星期，从开始的选题到选择编程语言、实现框架、到最后的项目开发，自己对数据库管理系统及数据库都有了较深的认识，并且掌握了系统开发的基本技能。从一开始的茫然不知所措，到现在能够较为熟练的运用 JS、HTML、CSS、Php 等技术，最终完成一个完整的管理系统，一路来收获颇丰。此次课程设计让我主要有了以下几点收获：

（1） 要重视系统设计，利用软件开发技术所学到的项目技术，从需求分析，体系结构设计认真执行。把握好全局，只有先完成总体设计，后续开发才能高效快速；

（2） 要通过实战快速掌握新编程语言，从对 php 完成不懂，再到以 php 实现数据库系统，不仅需要自己查阅网上资料和技术手册，更需要通过实例代码理解代码，转换为编程生产力。同时快速进行场景转换，把实例代码转换应用到实际项目场景中去。

（3） 在数据库设计方面，要充分应用课堂知识，根据设计原则，设计出符号实际生活需求，减少冗余异常的数据库。结合真实的需求分析，耐心地从数据库概念设计、逻

辑设计到物理设计。

（4） 项目开发过程中，注意代码的调试和迭代更新，由于本次系统为个人作品开发，一个人独立完成，而代码工程量较大，需要及时地对各功能模块进行调试，避免出现一股脑写代码，最后调试全崩盘的情况；同时，注意迭代更新，对于部分功能在一开始开发过程中可以设计的较为简单，之后逐步迭代，提高系统性能。

（5） 调试测试时充分利用出错信息以及错误日志文件等信息。

（6） 开发时注意编程语言及编译环境的版本，是否会对开发造成影响。本系统开发中，最初使用 php mysql_query 的方式进行数据库连接等数据库操作，而所用 php 版本为 5.7，在 5.0 以后的版本中该方式均不再支持并会警告；后来改用 php PDO 方式进行数据库操作。

当然，由于时间和精力的限制，lv 有友旅行管理系统目前也存在以下几个问题：

（1）功能不够完整，对管理员可进一步扩展功能；

（2）数据库方面，地点信息可以继续扩展;

（3 仅适用于小规模数据，不能处理大规模数据，系统的并发能力不强。

总之，通过本次课程设计，自己掌握了数据库相关知识，提高了自己的技术能力。

参考文献：

- 《数据库系统基础教程》　原著：Jeffrey D．Ullman、Jennifer Widom
  译者：岳丽华，龚育昌　机械工业出版社出版
- 《软件开发的技术基础》　骆斌　机械工业出版社
- 《计算机软件需求规格说明规范》　GB/T 9385-2008
- 《软件需求》　原著：劳森 译者：刘晓晖　电子工业出版社
- 《SQL server 2000 数据仓库与 Analysis Services》　Bain T 著 中国电力出版社
- 《数据库技术与联机分析处理》　王珊主编　　北京科学出版社