******J I A N G S U U N I V E R S I T Y**

**本 科 毕 业 论 文**

旅游信息分享及交友平台的设计与实现

学 院 名 称 ： 计算机科学与通信工程

专 业 班 级 ： 计算机1402

学 生 姓 名 ： 尹枳熙

指导教师姓名 ： 年 轶

指导教师职称 ： 讲 师

年 月

摘要

1. 绪论

1.1研究背景

1.2研究内容

1.3论文结构

1.4与本课题有关的国内外研究情况

1. 相关技术

2.1微信小程序

2.1.1简介

微信小程序是微信生态下的一种新型的手机应用形态。在2017月1月9日，微信创始人张小龙宣布了微信小程序正式上线。对于开发者而言，微信小程序是一种新的开放能力，开发者可以快速地开发一个小程序，用于传播信息。对于用户来说，用户可以获取小程序，享受小程序带来的信息。

2.1.2 特性

（1）“触手可及、用完即走”

“触手可及、用完即走”是微信小程序的特性。只需要一个二维码，你就能通过微信的扫一扫功能打开小程序，无需下载，无需安装即可使用，实现了应用“触手可及”的梦想。同时这也体现了“用完即走”的特性，用户无需担心手机内存是否不够，是否需要卸载应用，用完即可关闭，并可以随时进行下次使用。

1. 跨平台

当下的手机有android和ios两大平台，原生APP开发需要分别进行android和ios的开发，会消耗大量的开发成本。2018年1月15日，在2018年微信公开课上，微信创始人张小龙透露出微信用户已经达到10亿，可以说，在我国，几乎是每个人的手机上都有一个微信了，那么在微信生态下的微信小程序，自然就享受着跨平台的好处，开发者能够减少开发成本，两个平台的用户能得到相同的体验。

2.1.3 代码构成

进行过web前端开发的人知道，web前端开发采用的是HTML+CSS+JS这样的组合的，HTML用来搭建页面的结构；CSS 用来指明页面的外观，美化界面；JS在web前端开发中一般扮演着与用户交互的角色。

在微信小程序中，有着类似的组合，它们是WXML+WXSS+JS。

(1)JSON配置

#### 小程序的通过JSON文件进行配置。app.json文件对整个微信小程序进行全局配置，决定页面文件的路径、窗口表现、设置网络超时时间、设置多 tab 等，如其中的pages字段，它用于描述当前小程序所有页面路径；又如window字段，用于配置所有颜色，文字颜色等。每个页面有当前的页面配置 page.json，它对本页面的窗口表现进行配置

1. WXML

WXML与HTML一样，用于描述整个页面的结构。WXML和 HTML 非常相似，有标签、属性等等构成。但是也有很多不一样的地方，比如在标签方面做出了改变。同时，WXML用到了MVVM 的开发模式，把渲染和逻辑分离。

(3)WXSS

WXSS负责用来描述页面的样子。WXSS 具有 CSS 大部分的特性，小程序在 WXSS 做了一些扩充和修改。

(4)JS

与web前端开发一样，微信小程序用的也是JavaScript，对页面进行动态编程，与用户进行交互，与后端进行交互。

2.2Wafer2(Node.js环境)

Wafer2是Wafer 团队基于腾讯云强大的 IaaS 能力搭建了一个 PaaS 小程序解决方案。 Wafer2 用户不需要自行对服务器进行操作部署和上传代码，用户只需要开通，即可使用开发者工具上传、部署、调试小程序后端代码，无需了解服务器运维、数据库部署搭建即可使用。Wafer2 由腾讯云统一部署和托管服务器，基于服务器封装Node.js 和 PHP两种语言运行环境，将运行环境的使用权限交给用户，让用户可以直接使用微信开发者工具一键上传、部署、调试小程序后端代码。

Wafer2 的 Node.js环境 采用了 Koa.js 框架编写。Koa 将整个请求过程看做全异步的操作，使用 Node.js 7.6 开始支持的 async/await 语法，大大简化了整个应用开发的繁琐性。Koa是Express原班人马打造的一个更小，基于node.js平台的下一代web开发框架。

2.3Mysql

数据库是用于存放数据的仓库，

MySQL是瑞典MySQL AB 公司开发的关系型数据库管理系统，现在被Oracle收购，并入Oracle旗下。当下MySQL最流行的关系型数据库管理系统之一，因为它拥有体积小、速度快、总体拥有成本低、免费开源等特点。对于比oracle等需要付费的大型数据库，MySQL能够降低开发运行成本，因此很多开发人员的青睐。当然MySQL也会有很多不足的地方，比如它的功能较少、性能相当较弱等等。

1. 系统分析

3.1系统目标

本系统需要做一个关于旅游和交友的微信小程序，要达到两个主要目的。首先，用户能够通过小程序很容易就能获取想要获取的景点信息、旅行攻略等。其次，用户能够通过本系统，找到和自己兴志相投的朋友。

3.2可行性分析

1.经济可行性

微信为微信小程序的开发提供了微信web开发者工具，而这个工具是完全免费的。在后端方面，采用了腾讯云的微信小程序解决方案服务，服务内容包括云端服务器，COS，域名注册和解析，SSL证书等，非常便宜实惠。

1. 技术可行性

我学习并开发过一个小网站，网页编程采用的是HTML+CSS+JS这样的组合，在微信小程序中有着相同的角色，他们是WXML+WXSS+JS，与网页开发有很多相似的地方。经过一年的发展，微信小程序完善了官方开发文档，可以很好地帮助开发者开发小程序。同时，网络上也出现了很多博客和教程，分享很多demo和开发流程，这很好地帮助了初学者的学习进程。而服务器方面，采用了腾讯云的微信小程序解决方案服务，购买后自动分配云端资源，完成域名注册、解析以及SSL证书部署，并且针对小程序，提供有用的demo及SDK案例。

3.环境可行性

本平台的用户会到各地旅行，可以加上适当的引导，促进游客保护景点环境的意识，对自然环境和人文环境的保护做出贡献。

4.法律可行性

本项目开发所用软件均为正版，项目没有侵权或者抄袭等违法情况，也没有被申请过专利，故可行。

3.3功能需求

旅游信息分享及交友平台应该能够方便用户去规划自己的旅程，使自己的行程变得更加完美。在旅游后能够分享自己的旅游心情、经历以及心得，并以此收获朋友。因此，平台的重点在于用户如何能够便捷、准确地获取自己需要的旅游信息以及方便、愉悦地分享自己的旅游经历。因此，我把功能需求分成了四个模块，景点模块、好友模块、个人模块、推荐模块。

3.3.1景点模块

景点模块是用户获取旅游信息的重要定向途径。景点模块由地区标签定位，一个地区包含多个景点，以景点为一个单位，为用户提供信息。用户可以通过搜索、标签定位、景点发现等方式定位到具体景点，获取景点信息。一个景点模块由多个子模块组成，分别为景点简介、用户短评、帖子。景点简介的作用是给用户一个第一印象，能让用户最快速的了解景点。用户短评接在景点简介的下面，用户可以给景点进行简短的评价以及打分，让其他用户获取去过旅行用户的体验。接下来是帖子。帖子可以是旅游攻略，可以是感受分享，可以是任何你想表达的内容，这个部分让用户自由发挥、畅所欲言，记录他们的旅游心情、旅游经历，同时，其他用户可以查看到这些攻略，规划自己的行程，同时还可以结识到新的朋友。

3.3.2好友模块

交流和交友是这个平台的一大功能。由于是本项目是微信小程序，有微信的好友系统的支持，本项目的交流好友系统主要以回复评论，点赞，关注，以及私信的形式进行。用户通过评论，帖子来了解其他游客，通过点赞、关注和私信的方式进行互动、交友。

3.3.3个人模块

用户拥有自己的个人主页，个人主页中显示了自己的资料，曾经发表过的

评论、帖子以及自己关注的景点或用户。在个人主页中，还有需要有一个能够

保存用户旅游回忆的功能，这个功能以时间线的方式，把用户的旅游相片和心情都串起来，做成一个旅游相册。

3.3.4推荐模块

不知道哪里好玩？发现模块给用户一个途径去发现各种好玩景点。发现模块会不定时地给用户推荐引人入胜的景点，以及其他用户给出的优秀的帖子。这个模块不仅仅是想出游的用户可以来这个模块寻找自己想去的景点，这里也是阅读其他用户旅游经历、获取知心朋友的地方。

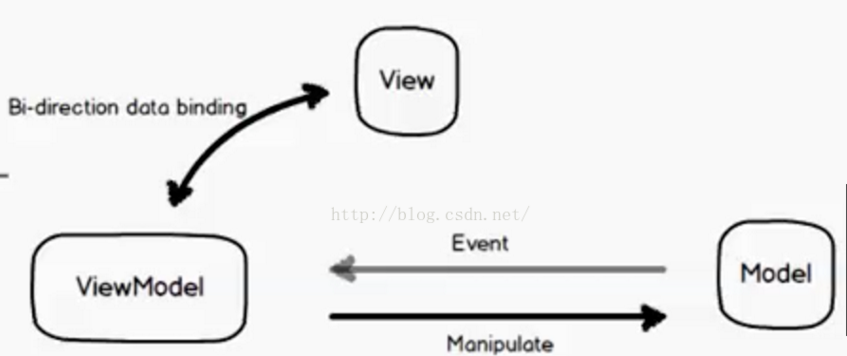
3.4界面需求

界面需要简洁。这是一个能够让用户能够便捷地获取旅游信息的小程序，用户能够通过简洁界面，短时间内熟悉小程序的内容。同时，界面需要友好。通过界面的按钮，图片等内容，用户能够很轻易地知道每一个按钮的作用，界面上每一个部分想要传达出的信息。

1. 系统设计

4.1架构设计

1. MVVC模式

微信小程序采用的是MVVM的开发模式。MVVM是Model-View-ViewModel的缩写，即模型－视图－视图模型。MVVM是MVC模式的的改进版，如下图所示

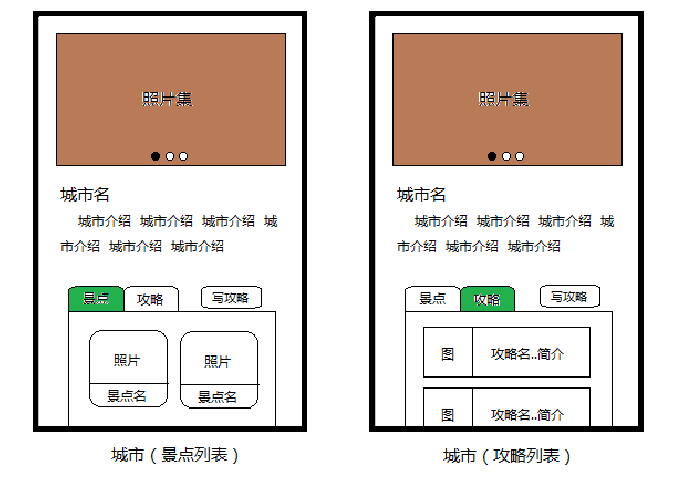
View负责视图显示，Model负责数据保存，ViewModel负责把Model和View关联起来。从图中可以看到View和ViewModel之间存在着数据双向绑定，也就是说，View的变动会自动反映在viewmodel，反之亦然，而这种思想就是MVVM的核心基础。在图中，ViewModel通过操纵使Model发生数据改变，Model通过事件将改变的数据发送给ViewModel，然后通过数据绑定，ViewModel通知View渲染页面。

4.2功能模块设计

4.2.1景点模块

1. 城市

城市模块中，会有城市的基本简介以及该城市所拥有的景点。城市子模块在景点模块里相当于一个容器，把一个城市里的景点都放在一块，让用户能够通过城市去浏览其中的景点，从而获取到景点信息。同时，城市模块中要拥有攻略功能，让用户分享在城市游玩的经历。因此，城市子模块的功能有城市信息、景点列表、攻略列表、跳转至写攻略页面，同时景点列表能够通过点击跳转到景点页面。具体设计如下图。



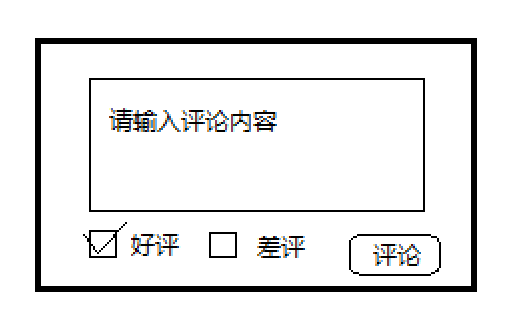
1. 景点

在景点子模块中，会提供景点的具体信息，用户能够通过这些景点信息对景点有一个初步的认识。接下来，景点模块要有评论功能和攻略。评论功能能够让游客抒发对景点的观点，让用户能够看到别的游客的观点，从而对景点有进一步的认识。而攻略功能能够让游客更好的抒发自己的观点，分享自己的游玩经历，同时也让别的用户能更深入地了解景点，以及吸收他人的游玩经验。因此景点模块要有景点信息、评论列表、攻略列表、写评论跳转、写攻略跳转、点攻略打开攻略页面功能、对评论进行观点偏向。具体设计如下图。



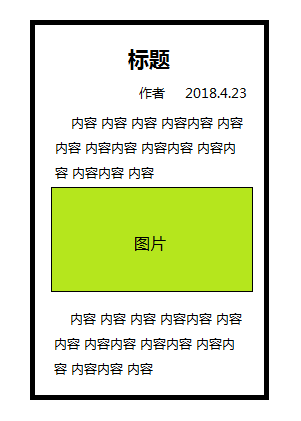
1. 评论

评论需要能够明确地表达出评论者的观点，因此评论设置了好评差评，这也让其他用户能一眼看出来观点的偏向。评论能抒发自己的观点，但是不宜过长，我们需要对评论字数进行限制。最后添加一个其他用户对评论的偏向，使评论也具有互动性。由于评论功能很简洁，因此把它设计为一个弹框。设计如下图。



1. 攻略

攻略功能是一个用户深度表达自己的功能。攻略应该是图文，能够让用户通过文字叙述游玩经历，辅以图片更好地展示自己的游玩经历。



4.2.2好友模块

1. 私信

私信是这个小程序中唯一的能够让用户互相进行交流的途径。由于微信就是一个社交软件，而我认为，在微信生态环境下的微信小程序中，不应该在其中存在一个新的好友系统，而应该是提供一个结交好友的桥梁，这就足够了。因此，在本系统中，只存在私信这种能让用户互相交流的形式，并且做到最简洁的方式。



1. 关注

关注功能能够使用户去关注需要关注的其他用户。

4.2.3个人模块

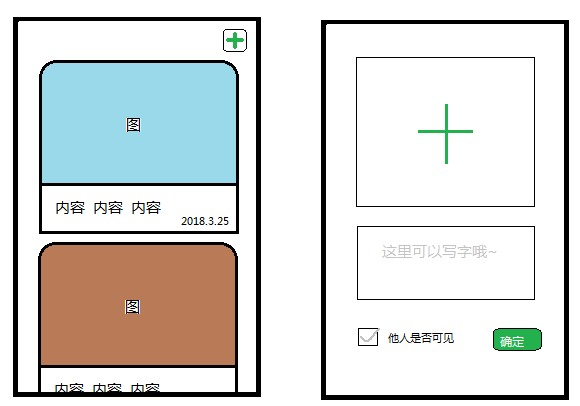
1. 个人资料

个人模块中的个人资料有用户的头像、用户名、性别和简介，反映出用户的基本信息，比较简洁。而个人页面中，会有用户之前的评论，帖子（攻略以及心情感悟），记忆长廊和私信的入口也会在放在这里。在个人页面中，你也可以写下你的心情感悟，它以帖子的形式存在，会在你的帖子列表中出现。设计如下图。



1. 回忆长廊

回忆长廊是一个能够保存用户旅游回忆的功能，这个功能以时间线的方式，把用户的旅游相片和心情都串起来，做成一个旅游相册。我把这个功能设计成一张相片和一句话的形式，让用户能够挑选出珍贵的照片，把它保存成回忆，不要把回忆泛滥化。设计如下图。



1. 他人查看个人页面

在评论、帖子等地方，用户可以通过点击作者头像去查看作者的个人页面。在这个他人的个人页面中，用户可以看到他人的动态，了解他人的旅游历程，他人的旅游观点，然后可以选择关注或者私信他人。设计如下图



4.2.4推荐模块和搜索模块

推荐模块给用户推荐了一些值得一去的景点，给用户提供旅游新思路。我帮推荐模块放在首页，因此同时把搜索模块也放到这里来，搜索模块支持城市和景点的搜索。设计如下图。



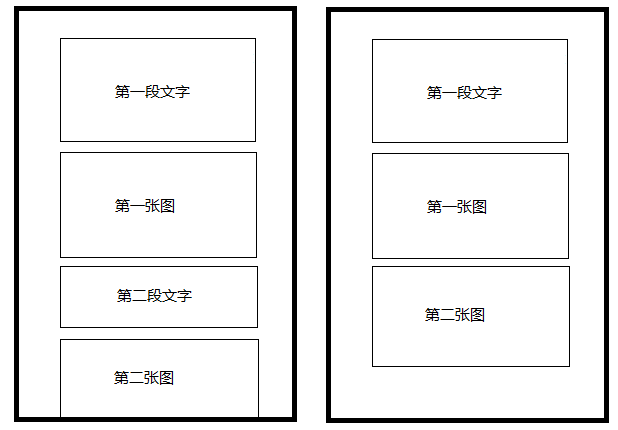
4.3图片存储

在腾讯云提供的小程序解决方案中，提供了图片上传储存的方案。

由于wafer2服务端会在环境初始化时自动配置好腾讯云的接口配置，因此可以直接使用SDK 提供直接上传图片至腾讯云对象储存（COS）的接口，通过微信小程序的API接口wx.uploadFile把图片上传，将请求传入SDK的图片上传接口中，即可自动上传图片到 COS 中，并返回数据,，包括上传图片的URL。然后把图片的URL保存在数据库中，就完成了图片存储这个环节。

4.4图文功能设计

图文功能采用一个“文字-图片”循环的结构实现，如下图所示



上面的左图也是写攻略时候的方式，使用“文字-图片”循环的方式，如果想连续插入两张图的时候，第二段文章为空，那么第二段文章在显示的时候就会被隐藏，最后文章页面就会显示成右图的样子。在存储图文的时候，我采用的是把图片递归上传，把得到的图片url地址拼接成字符串保存到数据库中。同样的，把多段文字拼接成一个字符串，而空段落则用特殊的字符串代替，然后拼接到字符串中，最终存入数据库中。

4.5数据库设计

根据上述功能设计，按照每个页面所需要的数据，设计出下面的数据表。

4.5.1用户表

用户是这个小程序服务的对象，每个用户都有独立的特征，每一个用户都是唯一的。在微信小程序的登陆接口中，会返回每个用户唯一的open\_id，所以，以这个open\_id作为主键，构建用户表。首先用户表与个人模块关联，把个人信息也放进用户表。用户信息有用户名、头像、简介、性别、粉丝数量。同时把个人设置中的**个人信息他人是否可见以及回忆长廊他人是否可见也放在这个表中。**

user\_info表

|  |  |  |  |
| --- | --- | --- | --- |
| **字段** | **类型** | **约束** | **备注** |
| **open\_id** | **VARCHAR(100)** | **NOT NULL**  **PRIMARY KEY** | **唯一ID** |
| **user\_name** | **VARCHAR(30)** | **NOT NULL** | **用户名** |
| **icon\_url** | **VARCHAR(255)** | **NOT NULL** | **头像url** |
| **introduction** | **VARCHAR(100)** | **NOT NULL** | **一句话简介** |
| **sex** | **TINYINT** | **DEFAULT 0** | **性别** |
| **fans\_number** | **INT UNSIGNED** | **DEFAULT 0** | **粉丝数量** |
| **info\_visible** | **TINYINT** | **DEFAULT 1** | **个人信息他人是否可见** |
| **memory\_visible** | **TINYINT** | **DEFAULT 1** | **回忆长廊他人是否可见** |
| **create\_time** | **timestamp** | **DEFAULT CURRENT\_TIMESTAMP** | **创建用户时间** |

4.5.2回忆长廊表

回忆长廊是一个单独的页面，需要一个单独的表去存储。回忆长廊表给一个自增id作为主键，每一个回忆项应该有作者的用户id，用于查找，回忆的内容包括一段文字和一张图片，别人还可以在浏览你的回忆长廊时为你的回忆点赞，所以有点赞数，还需要一个创建回忆项的时间。

memory\_gallery表

|  |  |  |  |
| --- | --- | --- | --- |
| **字段** | **类型** | **约束** | **备注** |
| **id** | **INT UNSIGNED** | **PRIMARY KEY**  **AUTO\_INCREMENT** | **回忆项ID** |
| **user\_id** | **VARCHAR(100)** | **NOT NULL** | **用户open\_id** |
| **pic\_url** | **VARCHAR(255)** | **NOT NULL** | **照片URL** |
| **content** | **VARCHAR(255)** | **NOT NULL** | **内容** |
| **memory\_visible** | **TINYINT** | **DEFAULT 1** | **该回忆项他人是否可见** |
| **like** | **INT UNSIGNED** | **DEFAULT 0** | **点赞数** |
| **timestamp** | **timestamp** | **DEFAULT CURRENT\_TIMESTAMP** | **评论时间** |

4.5.3城市表

城市是一个单独的页面，需要单独的表去存储。首先用城市id作为城市表的主键。由于每个城市的邮编都是唯一的，因此我采用邮编当做城市id。接下来是城市的信息，城市名、具体介绍、介绍图片。然后考虑到推荐模块需要用到图片和文字，而这些图片和文字也是和城市绑定在一起的，所以也放在城市表中。在上面功能设计中，我把城市作为景点的容器，景点也需要有唯一的id。由于我想之后的开发可能城市和景点会出现在同一个模块中，因此我希望城市id和景点id都是唯一的，不会相同，所以对景点id做出了如下设计。我用一个无符号int存储这个景点id，城市的六位id放在十进制的高六位和用自增的3位景点id，每个城市最多可存储999个景点。

city表

|  |  |  |  |
| --- | --- | --- | --- |
| **字段** | **类型** | **约束** | **备注** |
| **city\_id** | **INT UNSIGNED** | **NOT NULL**  **PRIMARY KEY** | **城市id(用邮编)** |
| **city\_name** | **VARCHAR(50)** | **NOT NULL** | **城市名** |
| **brief\_introduction** | **VARCHAR(100)** | **NOT NULL** | **推荐页简介** |
| **introduction** | **VARCHAR(255)** | **NOT NULL** | **具体简介** |
| **next\_scenic\_spot\_id** | **INT UNSIGNED** | **NOT NULL**  **DEFAULT 1** | **该城市下一个景点3位编号** |
| **brief\_pic\_url** | **VARCHAR(255)** |  | **推荐页图片URL** |
| **pic\_url** | **VARCHAR(1023)** |  | **照片集照片URL** |

景点表

景点表的主键为景点id，这个id的组成在城市表的介绍中讲到过，这个id是由城市的六位id以及3位景点id组成的一个无符号整形数。景点表的结构和城市表的结构很相似，有景点的基本信息，包括景点名、简介、图片，还有推荐模块所需要的信息，包括推荐页简介、推荐页图片。由于每个景点都有对应的城市，因此还有一个归属城市id。

scenic\_spot表

|  |  |  |  |
| --- | --- | --- | --- |
| **字段** | **类型** | **约束** | **备注** |
| **scenic\_spot\_id** | **INT UNSIGNED** | **PRIMARY KEY**  **NOT NULL** | **景点id(城市id+景点3位编号)** |
| **scenic\_spot\_name** | **VARCHAR(50)** | **NOT NULL** | **景点名** |
| **belong\_city\_id** | **VARCHAR(6)** | **NOT NULL** | **归属城市的id** |
| **brief\_introduction** | **VARCHAR(100)** | **NOT NULL** | **推荐页简介** |
| **introduction** | **VARCHAR(255)** | **NOT NULL** | **一句话简介** |
| **brief\_pic\_url** | **VARCHAR(255)** |  | **推荐页图片URL** |
| **pic\_url** | **VARCHAR(1023)** |  | **照片集照片URL** |

景点评论表

在景点页面中，会有用户的评论列表。景点评论表使用评论ID作为主键，一个自增的无符号整数。一条评论的构成包括评论用户头像、评论用户名字、评论内容、评论的偏向、赞同数、反对数、评论时间。由于评论是景点页面的一个功能，因此要使用景点主键去查找属于这个景点的评论，因此在景点评论表上也要加上景点id。而在用户个人页面里面也能看到用户以往的评论，因此该表中需要加上用户表的主键，即用户id。

scene\_comment表

|  |  |  |  |
| --- | --- | --- | --- |
| **字段** | **类型** | **约束** | **备注** |
| **id** | **INT UNSIGNED** | **PRIMARY KEY**  **NOT NULL**  **AUTO\_INCREMENT** | **评论ID** |
| **icon\_url** | **VARCHAR(255)** | **NOT NULL** | **头像url** |
| **user\_name** | **VARCHAR(30)** | **NOT NULL** | **用户名** |
| **user\_id** | **VARCHAR(100)** | **NOT NULL** | **评论用户open\_id** |
| **content** | **VARCHAR(255)** | **NOT NULL** | **内容** |
| **scenic\_spot\_id** | **VARCHAR(10)** | **NOT NULL** | **景点id** |
| **scenic\_spot\_name** | **VARCHAR(50)** | **NOT NULL** | **景点名** |
| **evaluation** | **TINYINT** | **NOT NULL** | **评价** |
| **agree** | **INT UNSIGNED** | **DEFAULT 0** | **好评量** |
| **disagree** | **INT UNSIGNED** | **DEFAULT 0** | **差评量** |
| **timestamp** | **timestamp** | **DEFAULT CURRENT\_TIMESTAMP** | **评论时间** |

景点评论点赞表

这个表是用来记录用户对景点评论点赞情况的。用户点赞之后要记录用户的点赞情况以实现用户不会重复点赞一个评论。该表的主键使用一个自增的无符号整数。表中的信息有景点id、景点评论id、用户id和点赞类型这些必要的信息。

scene\_comment\_like表

|  |  |  |  |
| --- | --- | --- | --- |
| **字段** | **类型** | **约束** | **备注** |
| **id** | **INT UNSIGNED** | **PRIMARY KEY**  **AUTO\_INCREMENT** | **ID** |
| **scenic\_spot\_id** | **INT UNSIGNED** | **NOT NUL** | **景点ID** |
| **comment\_id** | **INT UNSIGNED** | **NOT NUL** | **评论ID** |
| **user\_id** | **VARCHAR(100)** | **NOT NUL** | **用户ID** |
| **type** | **TINYINT** | **NOT NUL** | **like,dislike之类的** |

推荐表

推荐表是用于推荐模块的。主要是存储推荐列表信息，这个表非常简单，只存储需要推荐的景点或城市id，并有一个type用于区分景点和城市。这个表起到一个索引的作用。

Recommendation表

|  |  |  |  |
| --- | --- | --- | --- |
| **字段** | **类型** | **约束** | **备注** |
| **id** | **INT UNSIGNED** | **PRIMARY KEY** | **城市或景点ID** |
| **type** | **TINYINT** | **NOT NULL** | **景点或城市** |

文章表

文章表是用来存储帖子的，即攻略和心情经历。文章表的主键为一个自增的无符号整数。文章页面的构成为文章名、作者、写文章日期、内容、图片。在上面的图文功能设计中阐述了功能实现方式，内容和图片都分别用一个字段存储。

article表

|  |  |  |  |
| --- | --- | --- | --- |
| **字段** | **类型** | **约束** | **备注** |
| **id** | **INT UNSIGNED** | **PRIMARY KEY**  **AUTO\_INCREMENT** | **文章ID** |
| **user\_id** | **VARCHAR(100)** | **NOT NULL** | **创建用户open\_id** |
| **belong\_scene\_id** |  | **NOT NULL** | **所属地方的ID** |
| **title** | **VARCHAR(50)** | **NOT NULL** | **文章名字** |
| **author** | **VARCHAR(30)** | **NOT NULL** | **用户名** |
| **pic\_url** | **VARCHAR(1024)** | **NOT NULL** | **照片URL** |
| **mainbody** | **text** | **NOT NULL** | **正文** |
| **like** | **INT UNSIGNED** | **DEFAULT 0** | **点赞数** |
| **timestamp** | **timestamp** | **DEFAULT CURRENT\_TIMESTAMP** | **创建时间** |

1. 系统实现

系统实现部分会展示每个页面的内容。

5.1主页

打开小程序首先进入的是首页。首页的最上部是搜索栏。

搜索栏

5.2功能2

1. 总结与展望

6.1系统总结

6.2系统展望

致谢

参考文献