“小肥牛多人扫码点餐系统”实施方案

一．项目背景

第一代点餐系统：传统纸媒，体验不好，成本高

第二代点餐系统：提供式平板(PAD)，数据维护简单，成本高

第三代点餐系统：二维码点餐，多人扫码，体验好，成本可控

二．可行性分析

技术上：已有成熟产品

团队上：成员技术可以实现

资金上：自筹

时间上：大约两周完成 Version1.0

政策上：合规

三．需求分析

顾客：扫码(每桌一码)，多人同一个时间段同时扫码即认为在同一桌，时间起点：第一个顾客扫码，时间截止：结账完成。

提供个人的识别符号（内容任意，每次可以不同）。浏览菜谱：类别、菜品、曾经被点到的次数，点菜，修改菜品

数量。点菜同时可以查看同桌上其他登录人员，可以查看自己点的菜品以及其他人所点菜品。最后下单，结账前查

看最终订单内容

前台收银员：浏览所有的桌台状态(空闲/预约/占用/其他)，为特定顾客预约桌台，修改空闲桌台状态变为“预约”——其他

顾客暂时无法使用。若遇到非扫码客户，则手工修改桌台状态为占用，并代替顾客录入订单信息。查看用餐桌

台详情(用餐人、菜品&数量、开始用餐时间)，顾客付款收账后修改桌台状态变为空闲。

店长老板：添加/删除/修改桌台，添加/删除/修改类别，添加/删除/修改菜品，查看/搜索所有订单，营业额汇总

四．概要设计

子系统划分： ——前后端分离架构设计

(1)数据API子系统:提供后台管理、顾客App所需的数据

(2)后台管理子系统：提供前台收银和店长老板后台管理

(3)顾客App子系统：提供点餐下单功能

子系统后台选择：

(1)数据API子系统：MySQL、Node.js/Express、RESTful、JSON、跨域(JSONP 只能get/CORS 必须修改响应头，请求次

数变多/服务器端代理方案 服务器端压力会增加)选用CORS

(2)后台管理子系统：ApacheHttpd(运行服务器)、Node.js/Webpack(开发服务器)、Vue.js/Vue-CLI、Element-UI、Axios、

Canvas/QRCode、Vuex

(3)顾客App子系统：ApacheHttpd(运行服务器)、Node.js/Webpack(开发服务器)、Vue.js/Vue-CLI、Mint-UI、Axios、Vuex

数据结构设计：

数据库名：xiaofeiniu 编码方式UTF-8

|  |  |  |
| --- | --- | --- |
| 管理员信息表 xfn\_admin | | |
| aid | INT主键 | 管理员编号 |
| aname | VARCHAR(32)唯一 | 管理员用户名 |
| apwd | VARCHAR(64)加密存储 | 管理员密码 |
| role | …… | 管理员角色(V2实现) |

|  |  |  |
| --- | --- | --- |
| 项目全局设置 xfn\_settings | | |
| sid | INT 主键 | 编号 |
| appName | VARCHAR(32) | 应用/店家名称 |
| apiUrl | VARCHAR(64) | 数据API子系统地址 |
| adminUrl | VARCHAR(64) | 管理后台子系统地址 |
| appUrl | VARCHAR(64) | 顾客App子系统地址 |
| icp | VARCHAR(64) | 系统备案号 |
| copyright | VARCHAR(128) | 系统版权声明 |

|  |  |  |
| --- | --- | --- |
| 桌台信息表 xfn\_table | | |
| tid | INT 主键 | 桌台编号 |
| tname | VARCHAR(64) 允许空 | 桌台昵称 |
| type | VARCHAR(16) | 桌台类型 如3-4人桌 |
| status | INT  1.空闲 2-预定 3-占用 4-其他 | 桌台状态 |

|  |  |  |
| --- | --- | --- |
| 桌台的预定表 xfn\_reservation | | |
| rid | INT 主键 | 预定编号 |
| contactName | VARCHAR(64) | 联系人姓名 |
| phone | VARCHAR(16) | 联系电话 |
| contactTime | BIGINT | 联系时间，客户打电话时间 |
| dinnerTime | BIGINT | 预约的用餐时间 |
| tableId | INT 外键 | 预定的桌台号 |

|  |  |  |
| --- | --- | --- |
| 菜品分类表 xfn\_category | | |
| cid | INT 主键 | 类别编号 |
| cname | VARCHAR(32) | 类别名称 |

|  |  |  |
| --- | --- | --- |
| 菜品信息表 xfn\_dish | | |
| did | INT 主键 | 菜品编号,起始值为100000 |
| title | VARCHAR(32) | 菜品名称/标题 |
| imgUrl | VARCHAR(128) | 图片地址 |
| price | DECIMAL(6,2) 例:9999.99 | 菜品价格 |
| detail | VARCHAR(128) | 详细描述信息 |
| categoryId | INT 外键 参考类别的cid | 所属类别编号 |

|  |  |  |
| --- | --- | --- |
| 订单表 xfn\_order | | |
| oid | INT 主键 | 订单编号 |
| startTime | BIGINT | 用餐开始时间 |
| endTime | BIGINT | 用餐结束时间 |
| customerCount | INT | 用餐人数 |
| tableId | INT 外键 参考桌台tid | 桌台编号 |

|  |  |  |
| --- | --- | --- |
| 订单详情表 xfn\_order\_detail | | |
| did | INT 主键 | 订单编号 |
| dishId | INT 外键，参考菜品did | 菜品编号 |
| dishCount | INT | 菜品数量 |
| customerName | VARCHAR(64) | 点餐用户的称呼 |
| orderId | INT 外键 参考订单表oid | 订单编号，指明所属订单 |

五．详细设计

数据API子系统需要提供的API接口说明

详细请见：路由器文件中的注释

六．编码实现

软件准备:

(1)Git客户端软件（下载地址：<https://git-scm.com/>）

确保在命令行中可以输入执行git命令;

在当前电脑中设置git用户名和邮箱 git config --global user.name “ly” git config --global user.email “123@163.com”

(2)在Github.com注册账户(官网：<http://github.com>)

登录GitHub 创建远程仓库：

数据API子系统：xiaofeiniu-api

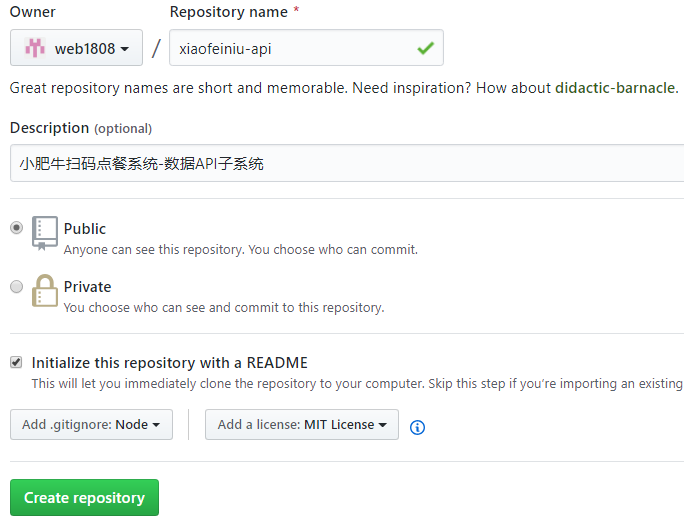
仓库地址：https://github.com/aichaoer/xiaofeiniu-api.git

后台管理子系统：xiaofeiniu-admin

仓库地址：https://github.com/aichaoer/xiaofeiniu-admin.git

顾客App子系统：xiaofeiniu-app

仓库地址：<https://github.com/aichaoer/xiaofeiniu-app.git>



(3)把远程仓库克隆到自己的电脑

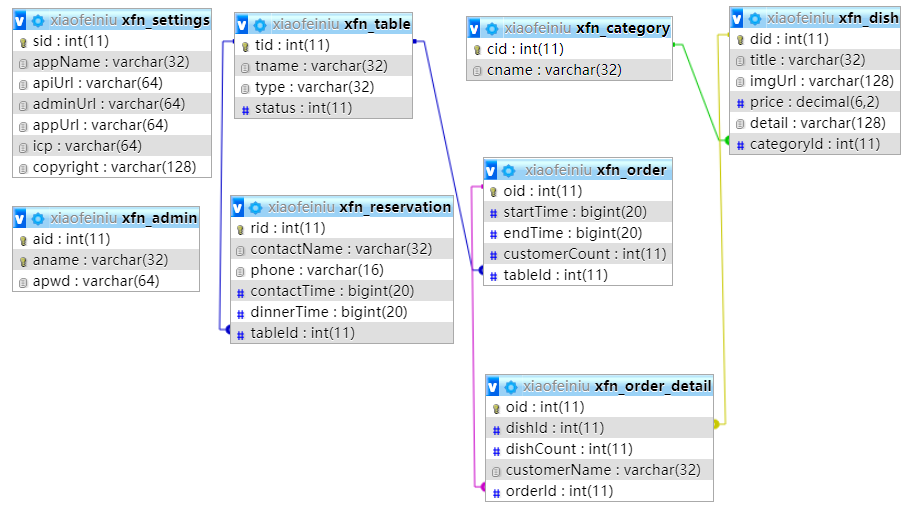
git clone 远程仓库地址

(4)安装VSCode安装工具(地址：<http://code.visualstudio.com>)

打开本地仓库所在目录，开始编辑

编辑 -> 添加到本地Git暂存区 -> 提交到本地Git仓库 -> 将本地仓库推送到远程仓库

2．根据数据库结构图编写sql脚本



3.实现数据API系统

(1)安装Node.js运行环境，版本必须是8.x以上 (node -v)

下载地址：<http://nodejs.org>

安装完成后，运行node –v进行确认

(2)下载Node.js/Express后台应用必需第三方模块 (新版本 –save可加可不加)

npm –h start stop test可以不加run，其余都得加npm run server

npm init #创建项目必需的package.json

npm i express –save #下载依赖并修改package.json

npm i mysql –save #数据库访问模块

npm i cors –save #解决跨域问题的中间件模块

npm i body-parser #请求主体解析的中间件模块

npm i –g node-dev # C:\Users\web\AppData\Roaming\npm\node\_modules下安装node -dev,监听node.js文件的修

改, 自动重启node应用

npm I multer –save #实现文件上传功能的中间件模块

(3)编写Express 服务器应用

|  |
| --- |
| RESTful风格的API中可用的请求方法：   * GET：读取（Read） * POST：新建（Create）（非幂等） * PUT：更新（Update） （幂等） * PATCH：更新（Update），通常是部分更新 （幂等） * DELETE：删除（Delete） （非幂等） |
| 注意：restful风格的接口，浏览器在正式调用前，一般会先发起一个“Preflight Request(预取请求)”——请求方法为OPTIONS |

|  |
| --- |
| 常见的请求消息内容类型：  text/plain 未经编码的k=v encodeURIComponent()  application/x-www-form-urlencoded 请求数据是k=v对  multipart/form-data 请求数据是可能包含文件的二进制数据  application/json 请求数据是JSON字符串 |

|  |
| --- |
| MySQL模块中SQL增删改查语句执行完成得到的result内容：  affectedRows: 0, 增删改语句“影响到的”行数  changedRows: 0, 增删改语句“更新到的”行数  insertId: 0, （仅限INSERT）插入语句生成的自增编号 |
| 数据：  1 TOM 男  2 JOE 男  下面这条语句，“影响到2行”，只能“更新到1行”  UPDATE emp SET name=’TOM’ WHERE sex=‘男’ |

|  |
| --- |
| Node.js/Express实现文件上传功能  (1)下载模块 npm I multer –save  (2)引入中间件:  const multer=require(“multer”);  var upload = multer({ dest: 'uploads/' })  (3)在特定的路由下使用该中间件  app.post('/admin/dish/image',upload.single('avatar'),function (req, res, next) {}) 单文件  app.post('/photos/upload', upload.array('photos', 12), function (req, res, next) {  console.log(req.file); console.log(req.body);  }) |

|  |  |  |  |
| --- | --- | --- | --- |
| ES6新特性: ——参数解构   |  |  |  | | --- | --- | --- | |  |  |  | |

|  |
| --- |
| 二维码又称二维条码，常见的二维码为QR Code，QR全称Quick Response：  本质就是对数据如文本，图片，网址，音视频等进行图形化编码。  小肥牛项目中每个桌台访问地址形如：<http://127.0.0.1:8092/#/order?tableId=3>  这种地址可以让顾客在手机浏览器中输入以实现在特定桌台点餐目的，为了简化此操作，特把上述每个地址转换为二维码贴在桌面上。  二维码图片的生成可以使用服务器端技术，也可以使用客户端技术——H5 Canvas绘图技术+二维码绘图规则——推荐使用第三方模块qrcode来实现 |

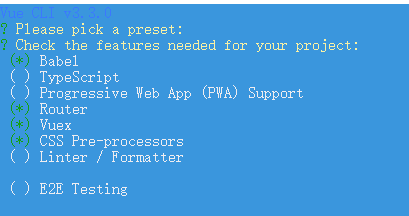
4.实现管理后台子系统

搭建开发环境

Vue-CLI Babel Vue-Router Vuex SCSS

npm I –g @vue/cli #Vue项目的全局脚手架

vue create xiaofeiniu-admin #创建一个空白的Vue项目



进入上述项目目录，安装必须得第三方依赖模块

npm i element-ui #UI组件系统

npm i axios #异步请求工具

npm i qrcode #二维码生成工具

七．测试

功能性测试之白盒测试：test/category.html

<button id=”btn”></button>

btn.onclick=function(){

axios.get(url).then().catch()

}

完成“数据API子系统”中的如下接口实现：

1. 获取全部的全局设置

GET /admin/settings

1. 修改全局设置

PUT /admin/settings

1. 获取所有的桌台信息

GET /admin/table

1. 获取预约状态桌台的详情

GET /admin/table/reservation/:tid

1. 获取占用状态桌台的详情

GET /admin/table/inuse/:tid

1. 修改桌台状态

PATCH /admin/table

1. 添加桌台

POST /admin/table

1. 删除桌台

DELETE /admin/table/:tid

八．部署上线

0.前期准备：注册一个新浪账号，绑定一个新浪云服务账号

注意：云服务器的安全邮箱和安全密码与微博账号不同

1.把数据API子系统部署到新浪云服务器

(1)登录新浪云

(2)创建一个node.js云应用，获得一个免费的三级域名

(3)管理云应用，开通mysql数据库服务

(4)管理云应用，上传自己的Node.js项目代码

|  |
| --- |
| 面试题：HTTP协议是没有状态的,请问有那修方案可以在不同的页面间记录共享数据?  (1)客户端(即浏览器)存储： 只适合存储安全要求级别低的数据  ①Cookie：字符串型KV(键值对)可以存储在内存/磁盘，兼容性好，但是只能存字符数据，且浏览器有大小限制(比较小，大约x KB)，且后续同一个服务器上所有的访问都会上传之前的Cookie数据  ②H5 WebStorage:老IE不兼容，但是大小限制(最大8MB)比较松，后续请求不会上传数据，只能用XHR等技术上传  window.sessionStorage.getItem(“adminName”):存储在浏览器内存中 .setItem  window.localStorage：存储在磁盘文件中，跨会话  ③indexedDB：  ④Web SQL：  (2)服务器端存储：适合于安全要求级别高的数据或者核心业务数据  ⑤数据库：使用恰当的后台语言，提交SQL访问数据库  ⑥服务器端session技术：把当前浏览器对应的私有数据存储在服务器的内存/文件系统/数据库中，类似于银行中的保管箱 |

|  |
| --- |
| Vue.js中的插槽概念(slot)  假设有一个自定义组件：MyPage.vue  <template>  <div>  <div><slot name=”pheader”></slot><div>  <div>页面主体<div>  <div><p>版权声明</p><div>  </div>  </template>  不写name，为默认插槽 |
| 在其他组件中使用上述组件：  <MyPage>  <h2 slot=”pheader”>购物车</h2>  </MyPage>  ↓↓↓  <div><h2>购物车</h2><div>  <MyPage></MyPage> |