“小肥牛多人扫码点餐系统”实施方案

1. 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PAD，数据维护简单、成本高

第三代点餐系统：多人扫码，体验好、成本可控

1. 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V1.0

政策上：合规

1. 需求分析

**顾客**：扫码(每桌一码)，多人同一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。提供个人识别符号，内容任意，且每次可以不同。浏览菜谱：类别、菜品、曾经被点到的次数。点菜，修改数量。点菜同时可以查看同桌台上其他登录人员，可以查看自己所点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**：浏览所有桌台状态（空闲、预约、占用、其它）。为特定顾客预约桌台，修改空闲桌台状态变为“预约”——其它顾客暂时无法使用。若遇到了非扫码顾客，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/修改菜品类别、添加/删除/修改菜品、查看/搜搜所有订单、营业额汇总。

1. 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

子系统技术选择：

(1)数据API子系统：MySQL、Node.js/Express、RESTful、JSON、 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UTF8

|  |  |  |
| --- | --- | --- |
| 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 0-其它 | 当前状态 |

|  |  |  |
| --- | --- | --- |
| 桌台预定信息表：xfn\_reservation | | |
| rid | INT主键 | 信息编号 |
| contactName | VARCHAR(64) | 联系人姓名 |
| phone | VARCHAR(16) | 联系电话 |
| contactTime | BIGINT | 联系时间 |
| 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) | 价格 |
| detail | VARCHAR(128) | 详细描述信息 |
| categoryId | INT 外键，参考类别.cid | 所属类别的编号 |

|  |  |  |
| --- | --- | --- |
| 订单表：xfn\_order | | |
| oid | INT 主键 | 订单编号 |
| startTime | BIGINT | 开始用餐时间 |
| 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.oid | 订单编号，指明所属订单 |

1. 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

1. 编码实现

1.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 "LiWenhua"

git config --global user.email "lwh@tedu.cn"

(2)在Github.com注册账户（官网https://github.com）

登录github，创建远程的项目仓库

数据API子系统：xiaofeiniu-api

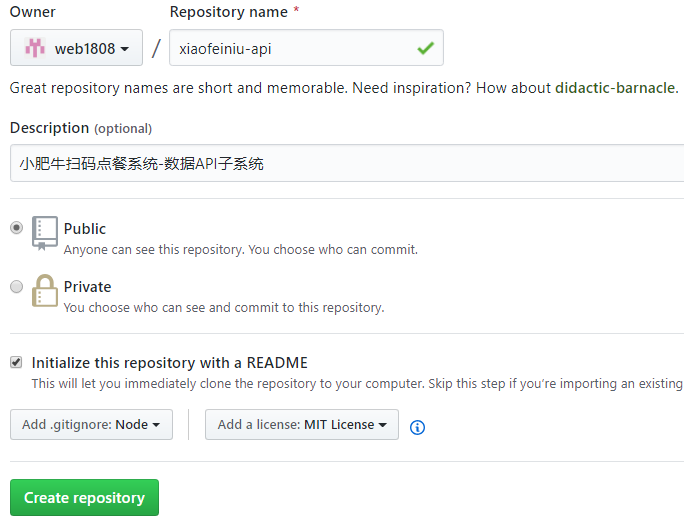
仓库地址：https://github.com/web1808/xiaofeiniu-api.git

管理后台子系统：xiaofeiniu-admin

仓库地址：https://github.com/web1808/xiaofeiniu-admin.git

顾客APP子系统：xiaofeiniu-app

仓库地址：https://github.com/web1808/xiaofeiniu-app.git



(3)把远程仓库克隆到自己的电脑中

git clone 远程仓库地址

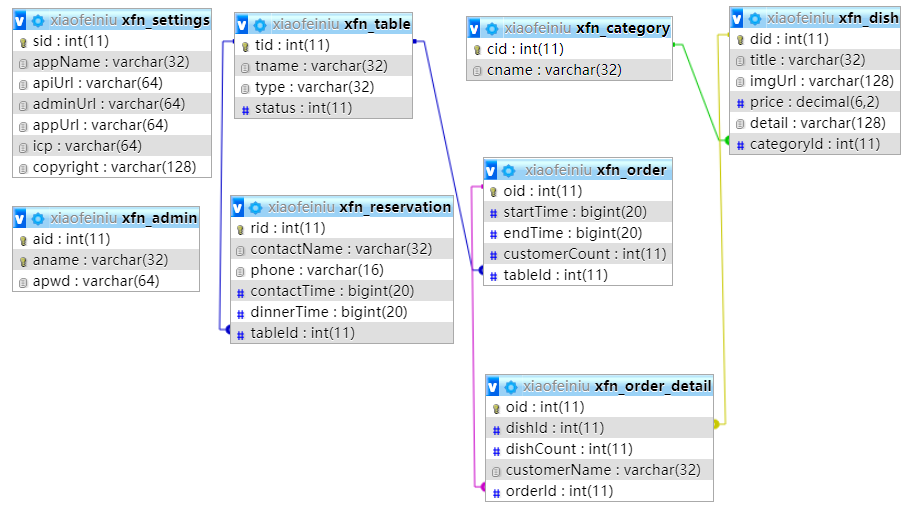
(4)安装VSCode开发工具（地址：https://code.visualstudio.com）

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子系统

(1)安装Node.js运行环境，版本必须是8.x以上

下载地址：https://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

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 --save #请求主体解析的中间件模块

npm i multer --save #处理客户端文件上传的中间件

npm i -g node-dev #在c:/Users/当前用户名/AppData/Roaming/npm目录下安装全局工具node-dev：监视node.js文件的修改，自动重启node应用

(3)编写Express服务器应用

|  |
| --- |
| RESTful风格的API中可用的请求方法：  GET：查询  POST： 添加（非幂等）  PUT： 修改(所有属性) （幂等）  PATCH：修改(部分属性) （幂等）  DELETE：删除（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  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 请求数据字符串 |

|  |
| --- |
| MySQL模块中SQL增删改语句执行完成得到result内容：  对于SELECT请求，result是一个[ … ]；对于INSERT/DELETE/UPDATE请求，result是一个执行结果的描述对象，包含如下属性：  insertId：(仅限INSERT)插入语句生成的自增编号  affectedRows：增删改语句“影响到的”行数  changedRows：增删改语句“改变了的”行数 |
| 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: 'tmp/'})  (3)在特定的路由上使用该中间件  app.post('/admin/dish/image', upload.single('dishImg'), (req, res)=>{  req.file：包含上传的文件信息  req.body：包含上传的字符型K-V对  }) |

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

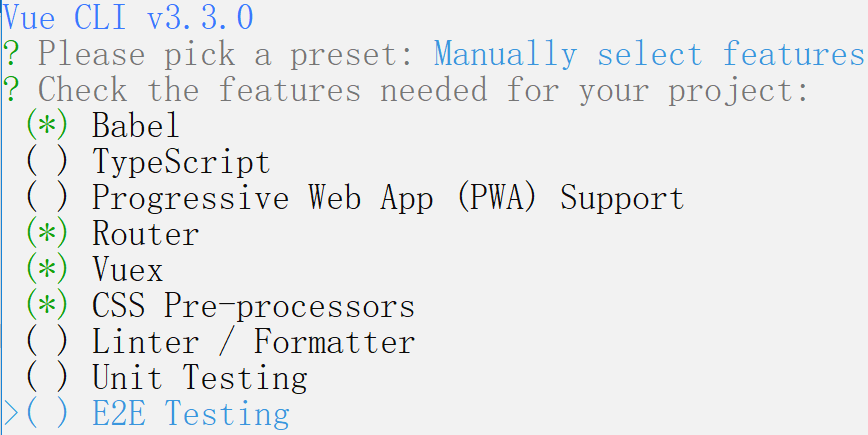
vue create xiaofeiniu-admin #创建一个Vue空白项目

进入上述项目目录，安装必须的第三方依赖模块：

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



|  |
| --- |
| Vue-CLI创建的项目默认有一个配置文件 vue.config.js  module.exports = {  //Webpack开发服务器的设置  devServer: {  host: '127.0.0.1',  port: 8091  }  } |

|  |
| --- |
| 面试题：HTTP协议是没有状态的，有哪些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  1)Cookie：字符串型KV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(大约xKB)，且后续同一服务器上所有的访问都会上传之前的Cookie数据  2)H5 WebStorage：老IE不兼容，但是大小限制(8MB)比较松，后续请求不会上传数据，只能XHR等技术上传  window.sessionStorage：存储在浏览器内存中  window.localStorage：存储在磁盘文件中  3)IndexedDB：  4)Web SQL：  **(2)服务器端存储：**适合于安全要求级别高的数据或核心业务数据  5)数据库：使用恰当的后台语言提交SQL访问数据库  6)session：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><slot></slot></div>  <div><slot name="pfooter"></slot></div>  </div>  </template> |
| 在其他组件中使用上述组件：  <MyPage>  <h2 slot="pheader">购物车</h2>  <p slot="pfooter">版权所有违者必究</p>  <ul><li></li></ul>  </MyPage> |
| **slot-scope：**  自定义组件模板形如MyTable：  <table>  <tr v-for="c in data">  <td><slot></slot></td>  </tr>  </table>  props:['data']  父组件使用该组件：  <MyTable :data="categoryList">  <span>{{cname}}</span>  </MyTable> |

|  |
| --- |
| function f1(user){  console.log(user.name + user.age)  }  f1( {name:'tom', age: 20} ) |
| ES6新特性—— 参数解构  function f1({name, age}){  console.log(name + age)  }  f1( {name:'tom', age: 20, sex:'男'} ) |

|  |
| --- |
| 二维码(Quick Response Code)：  本质就是对数据(如文本、网址、图片、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，为了简化此操作项目特意把上述每个地址转换为二维码粘贴在桌面上。  提示：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 |

七、测试

功能性测试之白盒测试： test/testCategory.html

<button id="bt"></button>

bt.onclick = function(){

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

}

集成测试：

压力测试：

八、部署上线

0.前提准备

(1)注册一个新浪微博账号 weibo.com

(2)使用新浪微博绑定一个新浪云服务账号 sae.sina.com.cn

注意：云服务器的安全邮箱/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 sae.sina.com.cn

(2)创建一个Node.js云应用，获得一个免费的三级域名

例如我的三级域名：xfnapi.applinzi.com

(3)管理云应用，开通MySQL数据库服务

注意：“共享型MySQL”不允许用户自定义新的库或者删库

(4)管理云应用，上传自己的Node.js项目代码

注意：1)数据库连接参数必须修改(参见手册) 2)新浪云中Node.js只能监听5050端口(NginX监听80端口再做请求转发) 3)新浪云服务器启动自动调用 npm i && npm start —— 要求无需上传node\_modules，且package.json中必须有依赖列表；以及scripts/start

git push 新浪云仓库的地址

(5)使用浏览器作为客户端进行测试访问

测试时访问80端口即可！

2.把管理后台子系统部署到新浪云服务器

开发服务器(Node.js代码)： Node.js + Webpack

运行服务器(纯静态文件)：任何一种Web服务器都可以

(0)正式构建出Vue.js项目的build文件

npm run build

会在当前项目根目录下得到dist目录，其中的文件就是可以被部署到任何WEB服务器上的文件！

(1)登录新浪云 sae.sina.com.cn

(2)创建一个任何一种语言的云应用(推荐使用免费的PHP主机空间)，获得一个免费的三级域名

例如我的三级域名：xfnadmin.applinzi.com

(3)管理云应用，上传自己的静态代码

可以使用Git上传，也可以直接上传zip代码包

课后任务：

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

2)修改全局设置

PUT /admin/settings

3)获取所有的桌台信息

GET /admin/table

4)获取预约状态桌台的详情

GET /admin/table/reservation/:tid

5)获取占用状态桌台的详情

GET /admin/table/inuse/:tid

6)修改桌台的状态

PATCH /admin/table

7)添加桌台

POST /admin/table

8)删除桌台

DELETE /admin/table/:tid

课后完成：Main.vue中的组件

提示：

左侧菜单是“导航菜单组件”：el-menu组件

右侧头部需要自定义组件：components/MainHeader.vue

右侧主体部分组件可以根据示例自行完成

课后任务：

1)完成管理后台子系统中的“桌台列表”和“菜品列表”功能

2)参考www.codeboy.com/xfn-app编写App子系统的用户界面