“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：

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

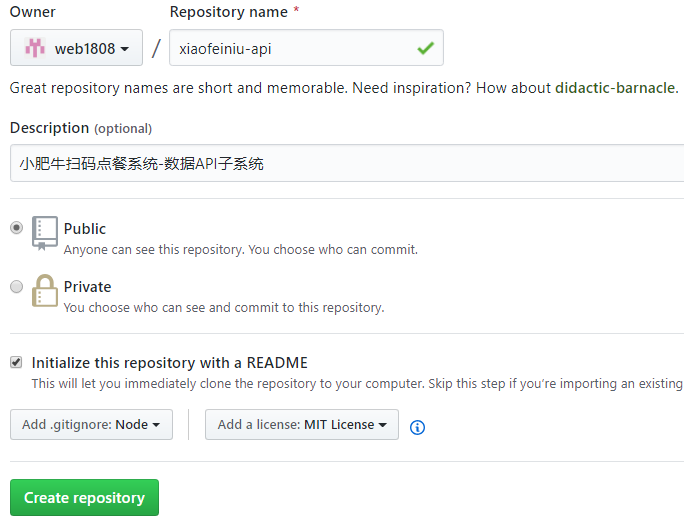
仓库地址：https://github.com/web1808/xiaofeiniu-api.git

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

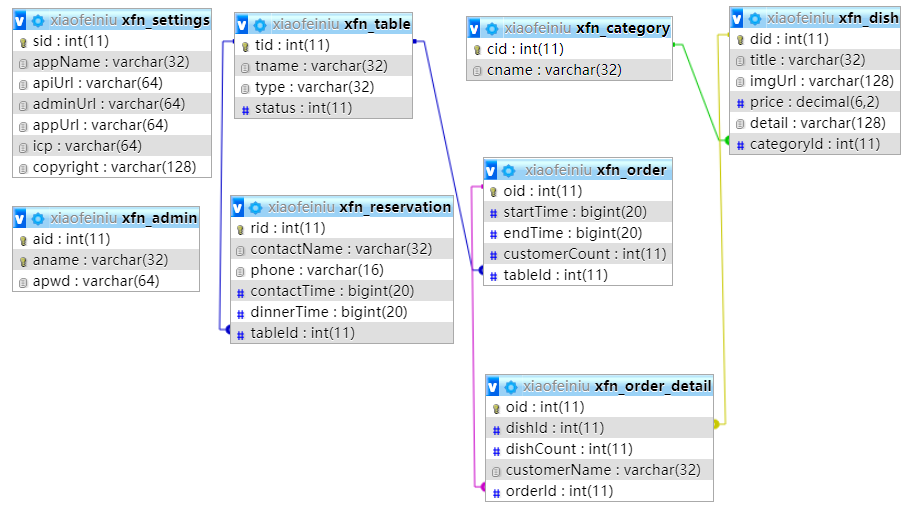
(4)安装VSCode开发工具（地址：https://code.visualstudio.com）

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  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JSON字符串 |

|  |
| --- |
| MySQL模块中SQL增删改语句执行完成得到result内容：  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: 'uploads/'})  (3)在特定的路由上使用该中间件  app.post('/admin/dish/image', upload.single('fieldName'), (req, res)=>{  req.file  req.body  }) |

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

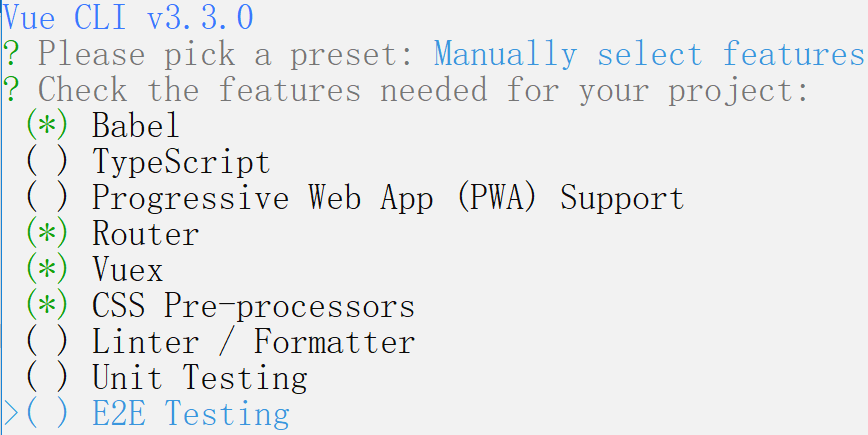
vue create xiaofeiniu-admin #创建一个Vue空白项目

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  }  } |

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

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