**目 录**

[1 系统分析 3](#_Toc13407671)

[1.1 系统开发背景与目的 3](#_Toc13407672)

[1.2系统的需求分析 3](#_Toc13407673)

[1.3 开发软硬件平台 4](#_Toc13407674)

[2 系统设计 5](#_Toc13407675)

[2.1 系统的数据库设计 5](#_Toc13407676)

[2.1.1数据库ER图 5](#_Toc13407677)

[2.1.2 ER图转为关系模式 5](#_Toc13407678)

[2.1.3数据表结构 6](#_Toc13407679)

[2.1.4各种表结构详细信息 7](#_Toc13407680)

[2.1.5数据库结构总图 11](#_Toc13407681)

[2.1.6 SQL语句建表 12](#_Toc13407682)

[2.2系统的总体设计 16](#_Toc13407683)

[2.2.1系统层次方框图 16](#_Toc13407684)

[2.2.2各个模块之间传递的数据以及结构 17](#_Toc13407685)

[2.3系统的详细设计 18](#_Toc13407686)

[2.3.1各模块具体设计 18](#_Toc13407687)

[3 系统实现 22](#_Toc13407688)

[3.1 登录模块 22](#_Toc13407689)

[3.2 注册模块 23](#_Toc13407690)

[3.3系统综合模块 24](#_Toc13407691)

[3.4顾客查看餐厅信息模块 24](#_Toc13407692)

[3.5预定模块 25](#_Toc13407693)

[3.6订单查询模块 26](#_Toc13407694)

[3.7评论查询模块 27](#_Toc13407695)

[3.8餐厅餐桌管理模块 27](#_Toc13407696)

[3.9餐厅菜单管理模块 28](#_Toc13407697)

[4 总结感想 29](#_Toc13407698)

[参考文献 29](#_Toc13407699)

[附录 30](#_Toc13407700)

[代码 30](#_Toc13407701)

[1．前端登陆相关JS： 30](#_Toc13407702)

[2. 后端UserController类： 34](#_Toc13407703)

# 1 系统分析

### 1.1 系统开发背景与目的

日常生活中，有许多人不想点外卖，想要与好朋友一起去餐厅吃。但是餐厅的选择又是挺要人纠结的，结果转来转去都没找到一个合适的，就算找到合适了，进去后也没有位置了。所以有些人就很早的出去预定位置，打电话或者提前去餐厅预定都有很大的弊端，不是没时间就是电话里说不清楚。该系统就是解决这样的事情，用户能提前看到餐厅的所有信息，方便预定，不用外出。也能根据评论来判断餐厅的饭菜口感和服务态度

### 1.2系统的需求分析

用户特点

用户分为两种，一种为经常去餐厅吃饭，但又常常等很久的顾客，另外一种是餐厅管理员，对本餐厅的基本操作和订单处理。对数据的准确性有要求，要求快速响应。

用户界面

a 将采用图形用户界面或产品系列的风格

b 屏幕布局或解决方案的限制

c 将出现在每个屏幕的标准按钮功能或导航链接

d 错误信息显示标准

功能需求

本系统分为两种：

a.一种为顾客设计，实现的功能为修改自己的基本信息，查询所需店铺，查看店铺的席位，菜单，并预定位置和所需菜类。查看其他客户的评论并自己参与评论。

b.另一种为餐厅管理员设计。修改自己餐厅的基本信息，主招牌菜。管理客户的订单，并提示还有多久开始。

非功能需求

a 性能需求

同时支持的最大顾客数为30，餐厅管理员数为20.

当有30个以上的用户同时对系统执行查询操作时，系统的相应时间应当不多于2秒，页面刷新频率应当在0.2次/秒~0.3次/秒。

b安全和保密需求

用户通过邮箱验证来找回密码。防止非法访问系统功能及数据丢失而要求用户身份确认，防止病毒入侵和黑客进攻而增加的警告拦截等功能

c 质量需求

本软件无故障运行一段时间的概率很高，，也有很强的可维护性。软件操作也非常简单。

### 1.3 开发软硬件平台

IDEA编译器

# 2 系统设计

### 2.1 系统的数据库设计

#### 2.1.1数据库ER图



图2.1 数据库ER图

#### 2.1.2 ER图转为关系模式

用户(用户编号，用户名，用户类型，密码，邮箱，用户人脸数据)

餐厅(餐厅编号，餐厅名称，地址，好评率，特色，用户编号)

顾客(顾客编号，姓名，手机号，口味偏好，地址，用户编号)

菜单(菜品编号，菜名，单价，类别，简介，餐厅编号)

餐桌(餐桌编号，容量，位置，餐厅编号)

预定(订单编号，顾客编号，餐厅编号，人数，餐桌号，时间，状态)

预定菜单(订单编号，菜品编号，备注)

评论(顾客编号，餐厅编号，评论类型，评价简述，时间)

#### 2.1.3数据表结构

表2-1数据表结构

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **编号** | **表名** | **主键** | **外键** | **功能** |
| **1** | **用户**  **（USERINFO）** | **用户名**  **（ID）** | **无** | **存储用户的账户密码和邮箱** |
| **2** | **餐厅**  **（RESTINFO）** | **餐厅编号**  **（ID）** | **用户名**  **（USER\_ID）** | **存储餐厅的一般信息** |
| **3** | **顾客**  **（CUSTINFO）** | **顾客编号**  **（ID）** | **用户名**  **（USER\_ID）** | **存储顾客的一般信息** |
| **4** | **菜单**  **（CARTEINFO）** | **菜名编号**  **（ID）** | **餐厅编号**  **（REST\_ID）** | **存储每个餐厅的所有菜的基本信息** |
| **5** | **餐桌**  **（TABLEINFO）** | **餐桌号**  **（ID）** | **餐厅编号**  **（REST\_ID）** | **存储每个餐厅的餐桌信息** |
| **6** | **预定**  **（ORDERINFO）** | **订单编号**  **（ID）** | **顾客编号和餐厅编号**  **（REST\_ID，CUST\_ID）** | **保存用户对某个餐厅的订单信息** |
| **7** | **预定菜单（ORDERFOOD）** | **菜单编号**  **（ID）** | **菜名编号，订单编号**  **（FOOD\_ID，ORDER\_ID）** | **保存订单信息的菜名** |
| **8** | **评论**  **（COMMENTINFO）** | **评论编号**  **（ID）** | **餐厅编号，顾客编号（REST\_ID，CUST\_ID）** | **保存用户对餐厅的评论** |

#### 2.1.4各种表结构详细信息

USERINFO表结构

表2-2 USERINFO表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **编号** | **属性名** | **类型，精度** | **主键/外键** | **约束** |
| **1** | **用户编号**  **(ID)** | **INT** | **PK** | **NOT NULL**  **UNQUIE** |
| **2** | **用户名（USERNAME）** | **VARCHAR**  **（12）** |  | **NOT NULL**  **UNQUIE** |
| **3** | **用户类型**  **（USERTYPE）** | **VARCHAR**  **（10）** |  | **NOT NULL** |
| **4** | **密码（PASSWORD）** | **VARCHAR**  **（10）** |  | **NOT NULL** |
| **5** | **邮箱**  **（MAILBOX）** | **VARCHAR**  **（50）** |  | **NOT NULL**  **UNQUIE** |
| **6** | **人脸数据**  **（USER\_IMAGE）** | **VARCHAR**  **（50）** |  | **UNQUIE** |

RESTINFO表结构

表2-3 RESTINFO表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **编号** | **属性名** | **类型，精度** | **主键/外键** | **约束** |
| **1** | **餐厅编号**  **（REST\_ID）** | **INT** | **PK** | **NOT NULL**  **UNQUIE** |
| **2** | **餐厅名称（REST\_NAME）** | **VARCHAR**  **（50）** |  | **NOT NULL** |
| **3** | **地址**  **（REST\_ADDRESS）** | **VARCHAR**  **（100）** |  | **NOT NULL** |
| **4** | **好评率**  **（PRAISE\_RATE）** | **VARCHAR**  **（5）** |  | **NOT NULL** |
| **5** | **特色**  **(FEATURE)** | **VARCHAR**  **（50）** |  |  |
| **6** | **用户编号（USER\_ID）** | **INT** |  | **NOT NULL** |

CUSTINFO表结构

表2-4 CUSTINFO表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **编号** | **属性名** | **类型，精度** | **主键/外键** | **约束** |
| **1** | **顾客编号**  **（CUST\_ID）** | **INT** | **PK** | **NOT NULL**  **UNQUIE** |
| **2** | **姓名**  **（CUST\_NAME）** | **VARCHAR**  **（10）** |  |  |
| **3** | **手机号**  **（CUST\_NUMBER）** | **VARCHAR**  **（12）** |  | **NOT NULL** |
| **4** | **地址**  **（CUST\_ADDRESS）** | **VARCHAR**  **（100）** |  |  |
| **5** | **口味偏好**  **（CUST\_FLAVOR）** | **VARCHAR**  **（100）** |  |  |
| **6** | **用户编号**  **（USER\_ID）** | **VARCHAR**  **（12）** | **FK** | **NOT NULL** |

FOODINFO表结构

表2-5 FOODINFO表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **编号** | **属性名** | **类型，精度** | **主键/外键** | **约束** |
| **1** | **菜品编号**  **（ID）** | **INT** | **PK** | **NOT NULL**  **UNQUIE** |
| **2** | **菜名（FOOD\_NAME）** | **VARCHAR**  **（50）** |  | **NOT NULL** |
| **3** | **单价**  **（FOOD\_PRICE）** | **INT** |  | **NOT NULL** |
| **4** | **类别**  **（FOOD\_TYPE）** | **VARCHAR**  **（20）** |  | **NOT NULL** |
| **5** | **简介**  **（FOOD\_INTRO）** | **VRACHAR**  **（50）** |  | **NOT NULL** |
| **6** | **餐厅编号**  **（REST\_ID）** | **INT** | **FK** | **NOT NULL** |

TABLEINFO表结构

表2-6 TABLEINFO表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **编号** | **属性名** | **类型，精度** | **主键/外键** | **约束** |
| **1** | **餐桌编号**  **(ID)** | **INT** | **PK** | **NOT NULL** |
| **2** | **餐桌号**  **（TABLE\_ID）** | **INT** |  | **NOT NULL** |
| **3** | **容量**  **（TABLE\_CAPA）** | **INT** |  | **NOT NULL** |
| **4** | **位置**  **（TABLE\_POSITION）** | **VARCHAR（20）** |  | **NOT NULL** |
| **5** | **餐厅编号**  **（REST\_ID）** | **INT** | **FK** | **NOT NULL** |

ORDERINFO表结构

表2-7 ORDERINFO表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **编号** | **属性名** | **类型，精度** | **主键/外键** | **约束** |
| **1** | **订单编号**  **（ID）** | **INT** | **PK** | **NOT NULL**  **UNQUIE** |
| **2** | **顾客编号**  **（CUST\_ID）** | **INT** | **FK** | **NOT NULL** |
| **3** | **餐厅编号**  **（REST\_ID）** | **INT** | **FK** | **NOT NULL** |
| **4** | **人数**  **（PEOPLE\_NUM）** | **INT** |  | **NOT NULL** |
| **5** | **餐桌号**  **（TABLE\_ID）** | **INT** | **FK** | **NOT NULL** |
| **6** | **时间**  **（START\_TIME）** | **TIMESAMP** |  | **NOT NULL** |
| **7** | **订单状态**  **（ORDER\_STATE）** | **VARCHAR**  **(20)** |  | **NOT NULL** |

ORDERFOOD 表结构

表2-8 ORDERFOOD表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **编号** | **属性名** | **类型，精度** | **主键/外键** | **约束** |
| **1** | **编号**  **(ID)** | **INT** | **PK** | **NOT NULL**  **UNQUIE** |
| **2** | **订单编号**  **（ORDER\_ID）** | **INT** |  | **NOT NULL**  **UNQUIE** |
| **3** | **菜品编号**  **（FOOD\_ID）** | **INT** |  | **NOT NULL** |
| **4** | **备注**  **（REMARK）** | **VARCHAR**  **（50）** |  | **NOT NULL** |

COMMENTINFO表结构

表2-9 COMMENTINFO表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **编号** | **属性名** | **类型，精度** | **主键/外键** | **约束** |
| **1** | **评论编号**  **（ID）** | **INT** | **FK** | **NOT NULL** |
| **2** | **订单编号**  **（ORDER\_ID）** | **INT** |  | **NOT NULL** |
| **3** | **评论类型**  **（COMMENT\_TYPE）** | **VARCHAR**  **（10）** |  | **NOT NULL** |
| **4** | **评论简述**  **（COMMENT\_INFO）** | **VARCHAR**  **（100）** |  |  |
| **5** | **时间**  **（COMMENT\_TIME）** | **DATETIME** |  | **NOT NULL** |

#### 2.1.5数据库结构总图

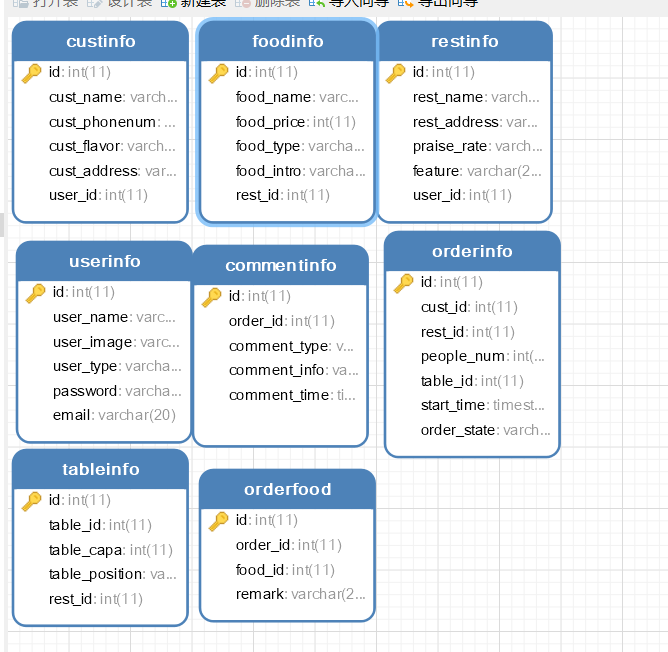


图2.2数据库结构总图

#### 2.1.6 SQL语句建表

**评论表commentinfo：**

**CREATE TABLE `commentinfo` (**

**`id` int(11) NOT NULL AUTO\_INCREMENT,**

**`order\_id` int(11) NOT NULL,**

**`comment\_type` varchar(20) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,**

**`comment\_info` varchar(255) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,**

**`comment\_time` timestamp(0) NULL DEFAULT NULL,**

**PRIMARY KEY (`id`) USING BTREE**

**)**

**顾客表custinfo**

**CREATE TABLE `custinfo` (**

**`id` int(11) NOT NULL AUTO\_INCREMENT,**

**`cust\_name` varchar(20) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,**

**`cust\_phonenum` varchar(12) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,**

**`cust\_flavor` varchar(20) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,**

**`cust\_address` varchar(255) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,**

**`user\_id` int(11) NOT NULL,**

**PRIMARY KEY (`id`) USING BTREE,**

**UNIQUE INDEX `cust\_phonenum`(`cust\_phonenum`) USING BTREE)**

**菜品表foodinfo：**

**CREATE TABLE `foodinfo` (**

**`id` int(11) NOT NULL AUTO\_INCREMENT,**

**`food\_name` varchar(20) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,**

**`food\_price` int(11) NOT NULL,**

**`food\_type` varchar(10) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,**

**`food\_intro` varchar(50) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL DEFAULT '无',**

**`rest\_id` int(11) NOT NULL,**

**PRIMARY KEY (`id`) USING BTREE**

**)**

**订单菜单表orderfood：**

**DROP TABLE IF EXISTS `orderfood`;**

**CREATE TABLE `orderfood` (**

**`id` int(11) NOT NULL AUTO\_INCREMENT,**

**`order\_id` int(11) NOT NULL,**

**`food\_id` int(11) NOT NULL,**

**`remark` varchar(200) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL DEFAULT '无',**

**PRIMARY KEY (`id`) USING BTREE**

**)**

**订单表orderinfo：**

**CREATE TABLE `orderinfo` (**

**`id` int(11) NOT NULL AUTO\_INCREMENT,**

**`cust\_id` int(11) NOT NULL,**

**`rest\_id` int(11) NOT NULL,**

**`people\_num` int(11) NOT NULL,**

**`table\_id` int(11) NOT NULL,**

**`start\_time` timestamp(0) NOT NULL,**

**`order\_state` varchar(10) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT '预定中',**

**PRIMARY KEY (`id`) USING BTREE**

**)**

**餐厅表restinfo：**

**CREATE TABLE `restinfo` (**

**`id` int(11) NOT NULL AUTO\_INCREMENT,**

**`rest\_name` varchar(20) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,**

**`rest\_address` varchar(20) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,**

**`praise\_rate` varchar(20) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL DEFAULT '暂无评论',**

**`feature` varchar(255) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,**

**`user\_id` int(11) NOT NULL,**

**PRIMARY KEY (`id`) USING BTREE**

**)**

**餐桌表tableinfo：**

**CREATE TABLE `tableinfo` (**

**`id` int(11) NOT NULL AUTO\_INCREMENT,**

**`table\_id` int(11) NOT NULL,**

**`table\_capa` int(11) NOT NULL DEFAULT 2,**

**`table\_position` varchar(20) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,**

**`rest\_id` int(11) NOT NULL,**

**PRIMARY KEY (`id`) USING BTREE,**

**UNIQUE INDEX `table\_id`(`table\_id`) USING BTREE**

**)**

**用户表userinfo：**

**CREATE TABLE `userinfo` (**

**`id` int(11) NOT NULL AUTO\_INCREMENT COMMENT '用户ID',**

**`user\_name` varchar(20) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,**

**`user\_image` varchar(255) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,**

**`user\_type` varchar(20) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,**

**`password` varchar(20) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,**

**`email` varchar(20) CHARACTER SET utf8 COLLATE utf8\_general\_ci NOT NULL,**

**PRIMARY KEY (`id`) USING BTREE,**

**UNIQUE INDEX `user\_name`(`user\_name`) USING BTREE,**

**UNIQUE INDEX `user\_image`(`user\_image`) USING BTREE,**

**UNIQUE INDEX `email`(`email`) USING BTREE**

**)**

### 2.2系统的总体设计

#### 2.2.1系统层次方框图



图2.3系统的层次方框图

#### 2.2.2各个模块之间传递的数据以及结构

表2-10 模块总结构表

|  |  |  |  |
| --- | --- | --- | --- |
| 模块编号 | 模块名称 | 输入数据 | 输出数据 |
| M1 | 注册模块 | 用户名，用户密码，和电子邮箱，人脸数据 | 将信息输入到数据库中 |
| M2 | 登陆模块 | 用户名，密码或者人脸登陆 | 用户类型 |
| M3 | 用户基本信息模块 | 顾客或餐厅的本信息 | 将信息输入到数据库中 |
| M4 | 顾客查看餐厅模块 | 所有餐厅的编号 | 每个餐厅的基本信息 |
| M5 | 顾客预定模块 | 顾客指定的餐厅编号，预定到达时间，人数，饭菜名自动选择合适的餐桌 | 将信息输入到数据库中，输出订单编号 |
| M6 | 顾客评论模块 | 时间，评论类型，评论简述 | 输入到数据库中 |
| M7 | 订单显示模块 | 订单编号 | 该订单的所有信息 |
| M8 | 评论显示模块 | 用户的编号 | 通过用户的编号获取对应的所有评论 |
| M9 | 餐厅餐桌信息 | 餐厅的编号 | 该餐厅对应的所有餐桌信息 |
| M10 | 餐桌菜单信息 | 餐厅的编号 | 该餐厅对应的所有饭菜信息 |

### 2.3系统的详细设计

#### 2.3.1各模块具体设计

用户登陆模块结构如下所示

表2-11 用户登陆模块IPO图

|  |  |  |  |
| --- | --- | --- | --- |
| 模块编号 | M1 | | |
| 模块名称 | 用户登陆 | | |
| 被调用模块 | M3,M4,M7,M8,M9,M10 | | |
| 调用的模块 | Login的模态对话框 | | |
| 输入数据 | 用户名，用户密码，人脸数据 | 输出数据 | 处理结果，信息字符串 |
| 处理 | 当用户输入用户名为空时，直接在前端判断，提示用户没有输入用户名。当用户输入密码后点击登陆时，如果该用户输入的用户名和密码在该用户所在的用户类型一一对应，则登陆成功，刷新当前页面，否则密码错误，登陆失败。如果用户选择用人脸登陆，打开摄像头，会自动扫描10次，最终结果后台判断。 | | |
| 特殊数据结构 | List<string> 和ArrayList | | |
| 特殊算法 | 穷举法 | | |
| 代码名称 | 前端：sLogin和Login 后端：loginByUserName和loginByFace | | |

用户注册模块结构如下所示

表2-12 用户注册模块IPO图

|  |  |  |  |
| --- | --- | --- | --- |
| 模块编号 | M2 | | |
| 模块名称 | 用户注册 | | |
| 被调用模块 | M1 | | |
| 调用的模块 | REGISTER html页面和基本信息页 | | |
| 输入数据 | 用户名，用户密码，确认密码，用户类型，验证码，还有人脸数据等等 | 输出数据 | 注册成功，注册失败 |
| 处理 | 用户输入所有数据后，判断邮箱格式是否正确，如果正确，则发送验证码到该邮箱中，用户看到后在验证码输入框内输入，完成验证。用户也可以选择注册人脸数据方便以后面部识别登陆 | | |
| 特殊数据结构 | List<string> | | |
| 特殊算法 | 穷举法 | | |
| 代码名称 | 前端：sendMessage() register() upImage() addCustInfo() addRestInfo()  后端：getCode() addCustInfo() addRestInfo() | | |

用户信息模块结构如下所示

表2-13用户信息模块IPO图

|  |  |  |  |
| --- | --- | --- | --- |
| 模块编号 | M3 | | |
| 模块名称 | 用户信息 | | |
| 被调用模块 | 无 | | |
| 调用的模块 | M1 | | |
| 输入数据 | 用户的基本信息 | 输出数据 | True，false |
| 处理 | 通过用户ID获取用户所有的信息并展现给前端 | | |
| 特殊数据结构 | List<string> | | |
| 特殊算法 | 穷举法 | | |
| 代码名称 | 前端：无 后端：showCustInfo() showRestInfo() | | |

查看餐厅信息模块结构如下所示

表2-14查看餐厅信息模块IPO图

|  |  |  |  |
| --- | --- | --- | --- |
| 模块编号 | M4 | | |
| 模块名称 | 查看餐厅信息 | | |
| 被调用模块 | 无 | | |
| 调用的模块 | M1 | | |
| 输入数据 | 无 | 输出数据 | 餐厅信息 |
| 处理 | 在用户没有登陆的情况下也可以查看各个餐厅的基本信息 | | |
| 特殊数据结构 | List<string> | | |
| 特殊算法 | 穷举法 | | |
| 代码名称 | 后端：showRestInfos() | | |

顾客预定餐厅信息处理模块结构如下所示

表2-15 顾客预定餐厅信息处理模块IPO图

|  |  |  |  |
| --- | --- | --- | --- |
| 模块编号 | M5 | | |
| 模块名称 | 预定处理 | | |
| 被调用模块 | 无 | | |
| 调用的模块 | 无 | | |
| 输入数据 | 用户到达时间，人数，餐桌号，饭菜信息 | 输出数据 | True，false |
| 处理 | 用户输入预定的一些基本信息，根据这些基本信息来判断是否已经有人在你前面在该时间段，预定了该餐厅的餐桌号.如果没有，则预定成功，如果没有则，预定失败，也可以在选择菜品的时候添加每样菜的备注 | | |
| 特殊数据结构 | List<string> | | |
| 特殊算法 | 穷举法 | | |
| 代码名称 | 前端：addRemark() addOrder() 后端：addOrder()等 | | |

查看预定信息模块结构如下所示

表2-16查看预定信息模块IPO图

|  |  |  |  |
| --- | --- | --- | --- |
| 模块编号 | M7 | | |
| 模块名称 | 预定查看 | | |
| 被调用模块 | 无 | | |
| 调用的模块 | M1 | | |
| 输入数据 | 无 | 输出数据 | 预定信息 |
| 处理 | 如果用户没用登陆则不会有相关信息出现，用户必须先登陆在通过自己的编号来获取所有的订单信息，无论是已完成的还是已取消的还是在预定中。还可以查看菜单，评论取消等等 | | |
| 特殊数据结构 | List<string> | | |
| 特殊算法 | 穷举法 | | |
| 代码名称 | 前端： cancelOrder()等 后端：showCustOrderInfo()等 | | |

评论模块结构如下所示

表2-17评论模块IPO图

|  |  |  |  |
| --- | --- | --- | --- |
| 模块编号 | M6 | | |
| 模块名称 | 评论餐厅 | | |
| 被调用模块 | M7 | | |
| 调用的模块 | 无 | | |
| 输入数据 | 评论类型，评论简述 | 输出数据 | True，false |
| 处理 | 当用户完成订单后，点击后面的评论按钮会进行对该餐厅的评论，评论完成后输入到数据库中，提示顾客评论成功 | | |
| 特殊数据结构 | List<string> | | |
| 特殊算法 | 穷举法 | | |
| 代码名称 | 前端：commenting() 后端：addComment() | | |

查看评论信息模块结构如下所示

表2-18查看评论信息模块IPO图

|  |  |  |  |
| --- | --- | --- | --- |
| 模块编号 | M8 | | |
| 模块名称 | 评论查看 | | |
| 被调用模块 | 无 | | |
| 调用的模块 | M1 | | |
| 输入数据 | 无 | 输出数据 | 评论信息 |
| 处理 | 如果用户没用登陆则不会有相关信息出现，用户必须先登陆在通过自己的编号来获取所有的评论信息 | | |
| 特殊数据结构 | List<string> | | |
| 特殊算法 | 穷举法 | | |
| 代码名称 | 前端：deleteComment()等 后端：showCustComment()等 | | |

菜单信息模块结构如下所示

表2-19菜单信息模块IPO图

|  |  |  |  |
| --- | --- | --- | --- |
| 模块编号 | M9 | | |
| 模块名称 | 餐厅菜单信息 | | |
| 被调用模块 | 无 | | |
| 调用的模块 | M1 | | |
| 输入数据 | 无 | 输出数据 | 餐厅菜单信息 |
| 处理 | 餐厅登陆后，通过餐厅编号来获取所有的饭菜的详细信息，也可以批量删除和增加 | | |
| 特殊数据结构 | List<string> | | |
| 特殊算法 | 穷举法 | | |
| 代码名称 | AddFood() deleteFood()等等 | | |

餐桌信息模块结构如下所示

表2-20 餐桌信息模块IPO图

|  |  |  |  |
| --- | --- | --- | --- |
| 模块编号 | M10 | | |
| 模块名称 | 餐厅餐桌信息 | | |
| 被调用模块 | M1 | | |
| 调用的模块 | M1 | | |
| 输入数据 | 无 | 输出数据 | 餐厅餐桌信息 |
| 处理 | 餐厅登陆后，通过餐厅编号来获取所有的餐桌的详细信息，也可以批量删除和增加 | | |
| 特殊数据结构 | List<string> | | |
| 特殊算法 | 穷举法 | | |
| 代码名称 | AddTable() deleteTable()等 | | |

# 3 系统实现

### 3.1 登录模块

登陆界面是一个前端的模态对话框，在每一个基本界面都有，可以在登陆后直接刷新本界面显示相关信息。如果用户选择账户密码登陆，用户输入账户后会自动判断该账户存不存在，弹出提示框，证明是有这个用户的。如果用户选择人脸识别。则直接对准摄像头就行



图3.1登陆界面

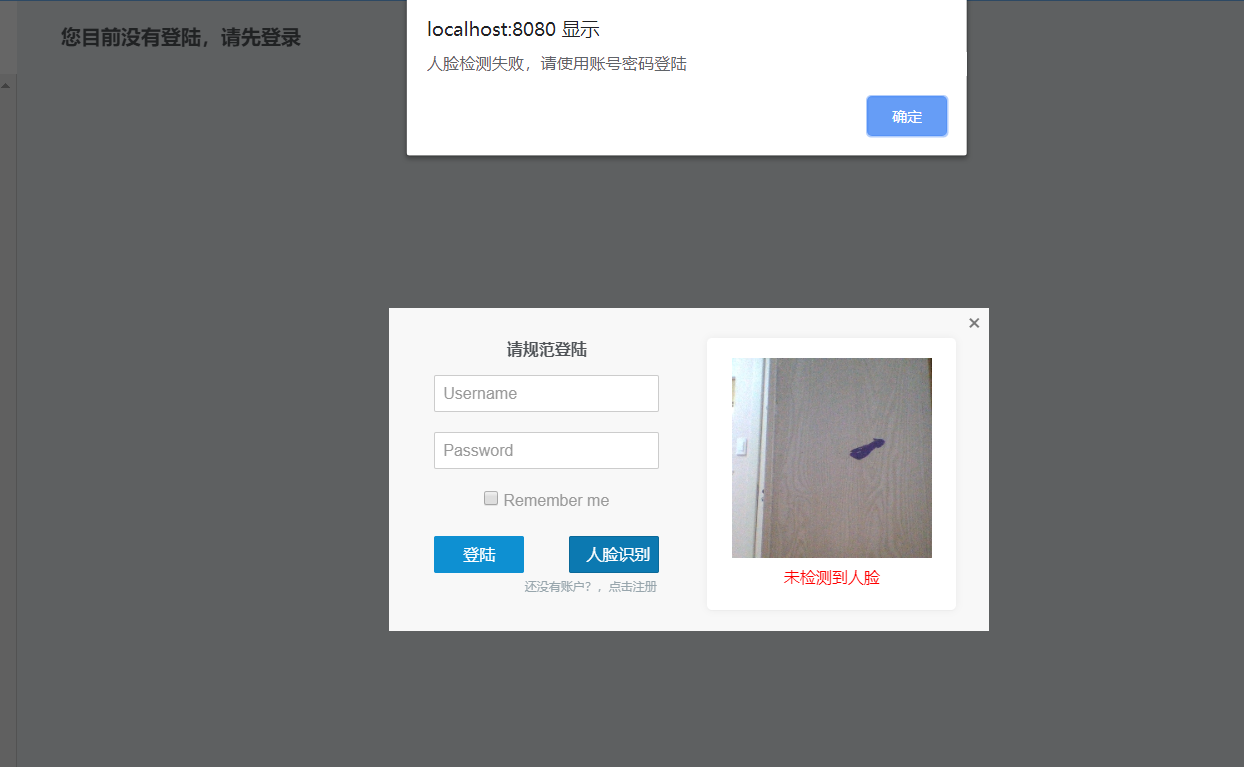


图3.2人脸识别

### 3.2 注册模块

注册界面是包含两部分，会判断用户输入的符不符合要求，不符合要求就提示用户，当然我们的注册是用邮箱注册的，当输入邮箱后，点击发送会获取验证码，输入验证码后完成验证，还需要填写基本信息。

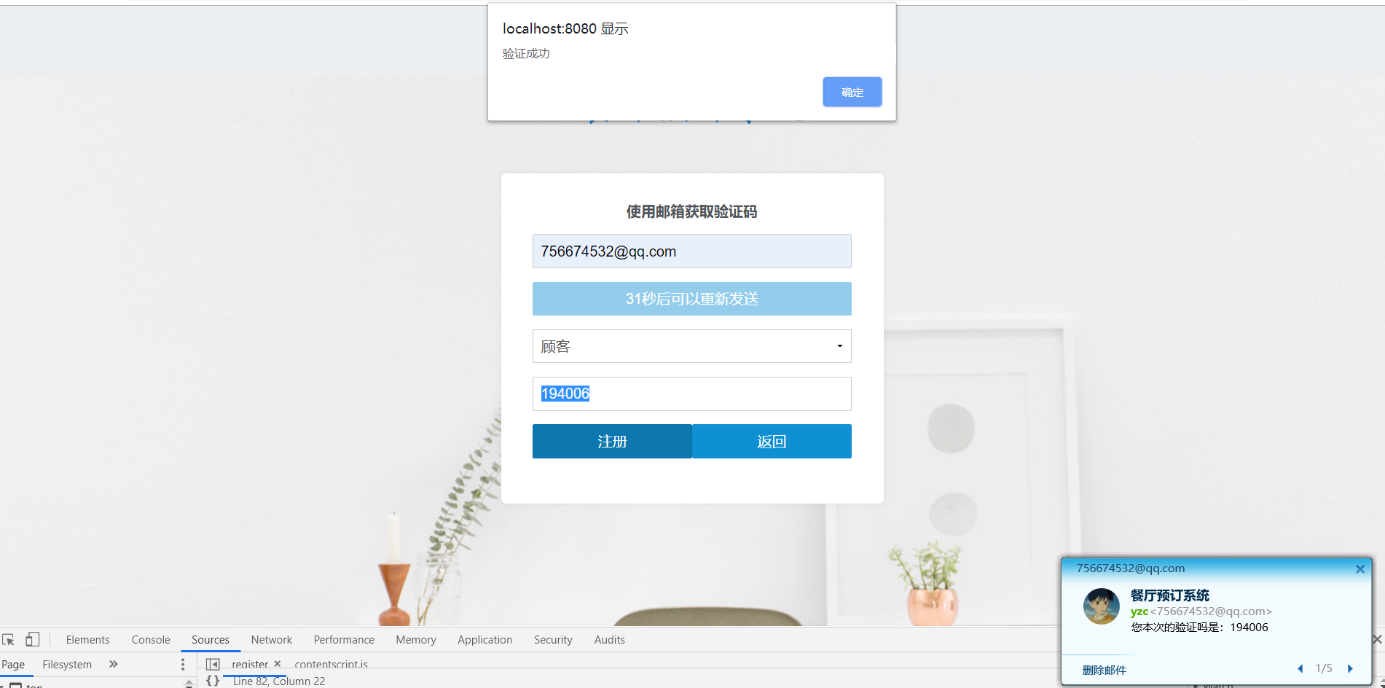


图3.3验证阶段

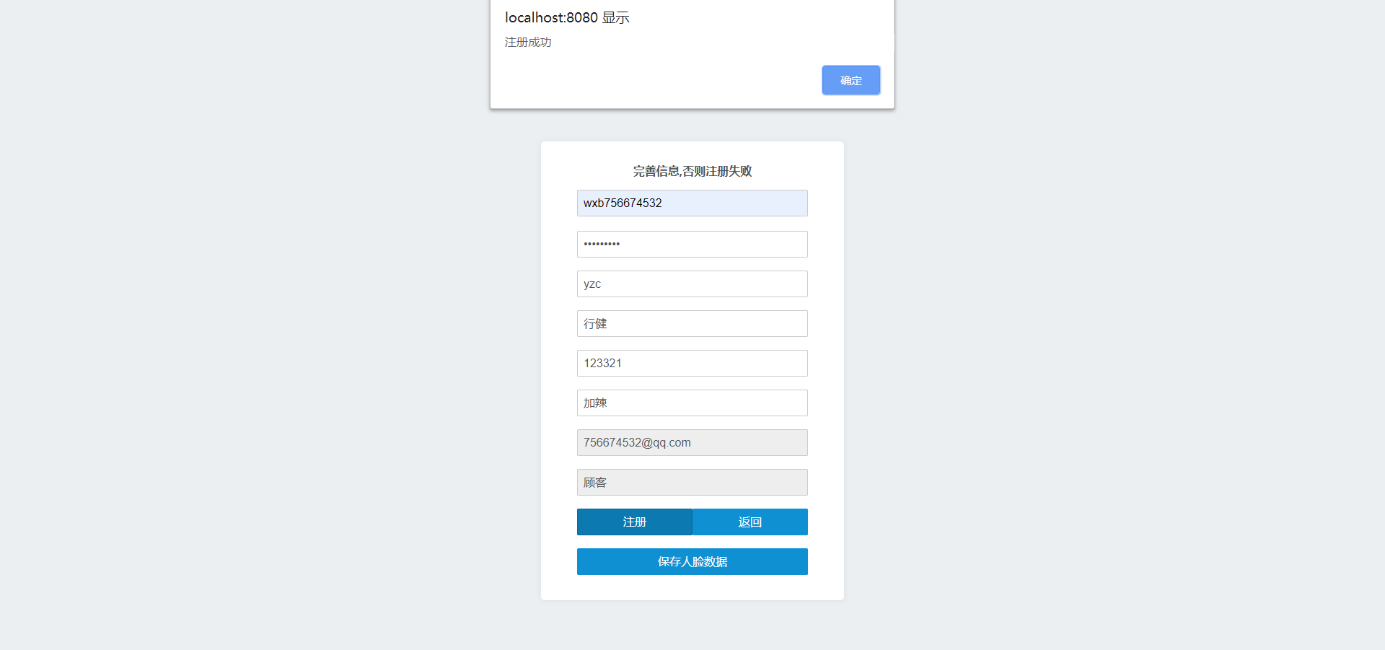


图3.4注册阶段

### 3.3系统综合模块

系统综合实现是与用户一些列操作的转换站，界面直接显示用户基本信息

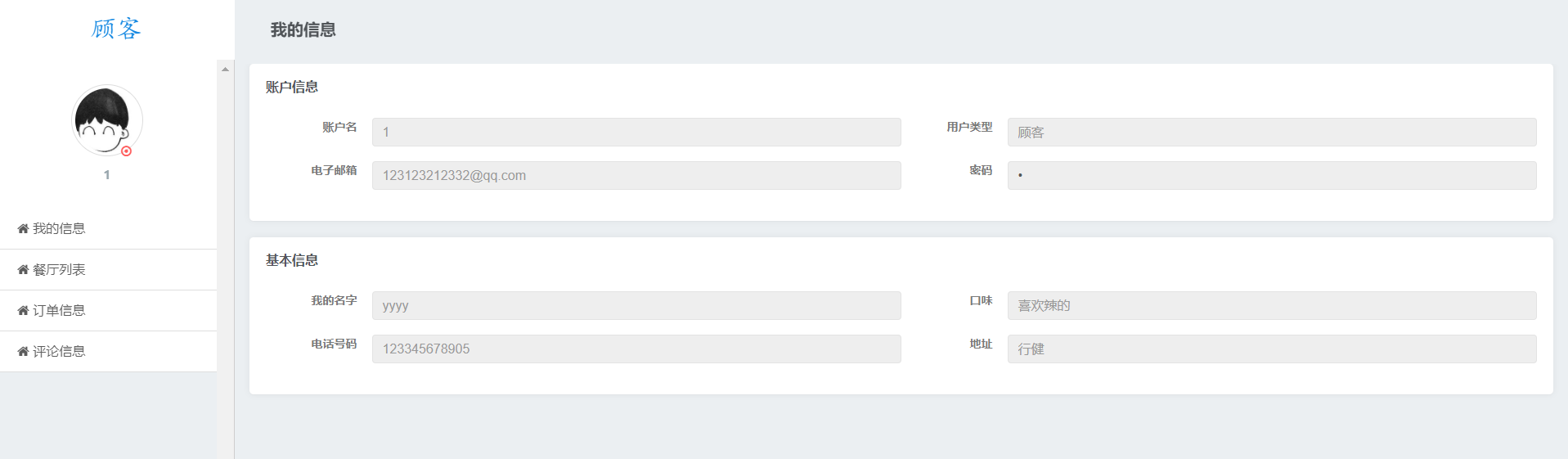


图3.5顾客系统界面

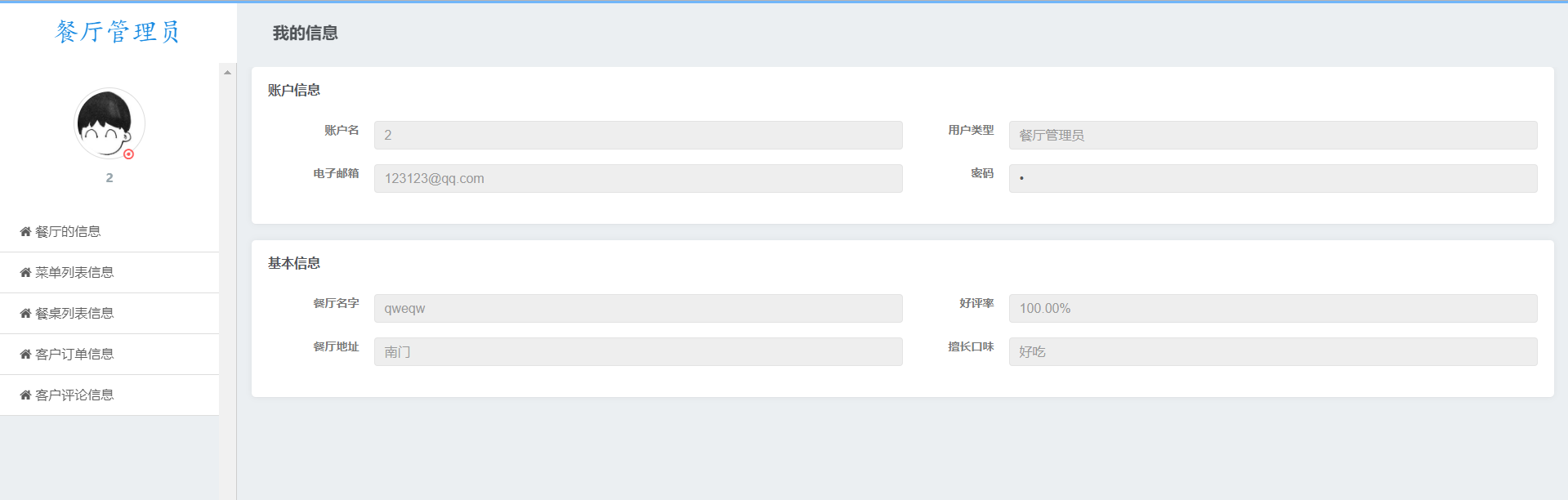


图3.6餐厅管理员系统界面

### 3.4顾客查看餐厅信息模块

用户点击餐厅信息后会弹出所有的餐厅的基本信息，地址，好评率，口味特色的，让用户直接观察到自己想要去的餐厅



图3.7查看餐厅信息界面

### 3.5预定模块

用户进入餐厅后，可以查看餐厅的评论信息，如果觉得好的话，可以直接选择菜品，点击添加订单，然后选择好时间，人数，系统会自动给顾客选择最佳的餐桌号。

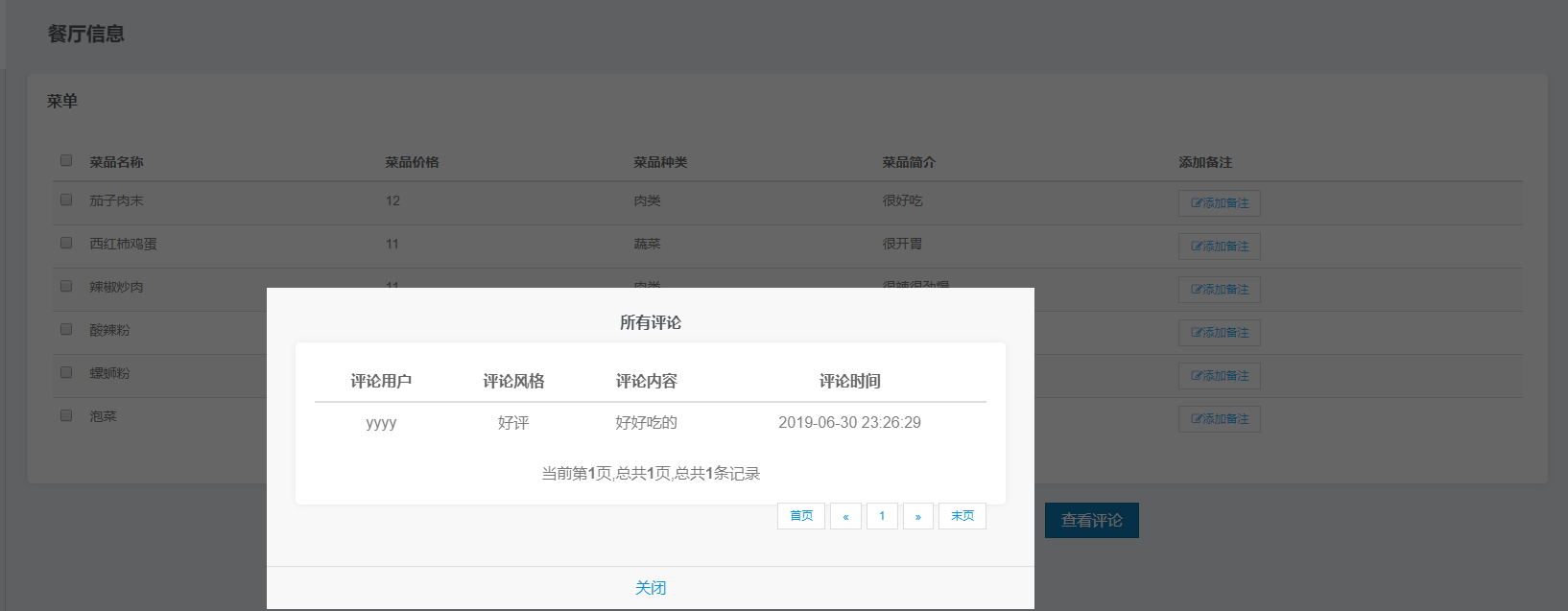


图3.8预定界面

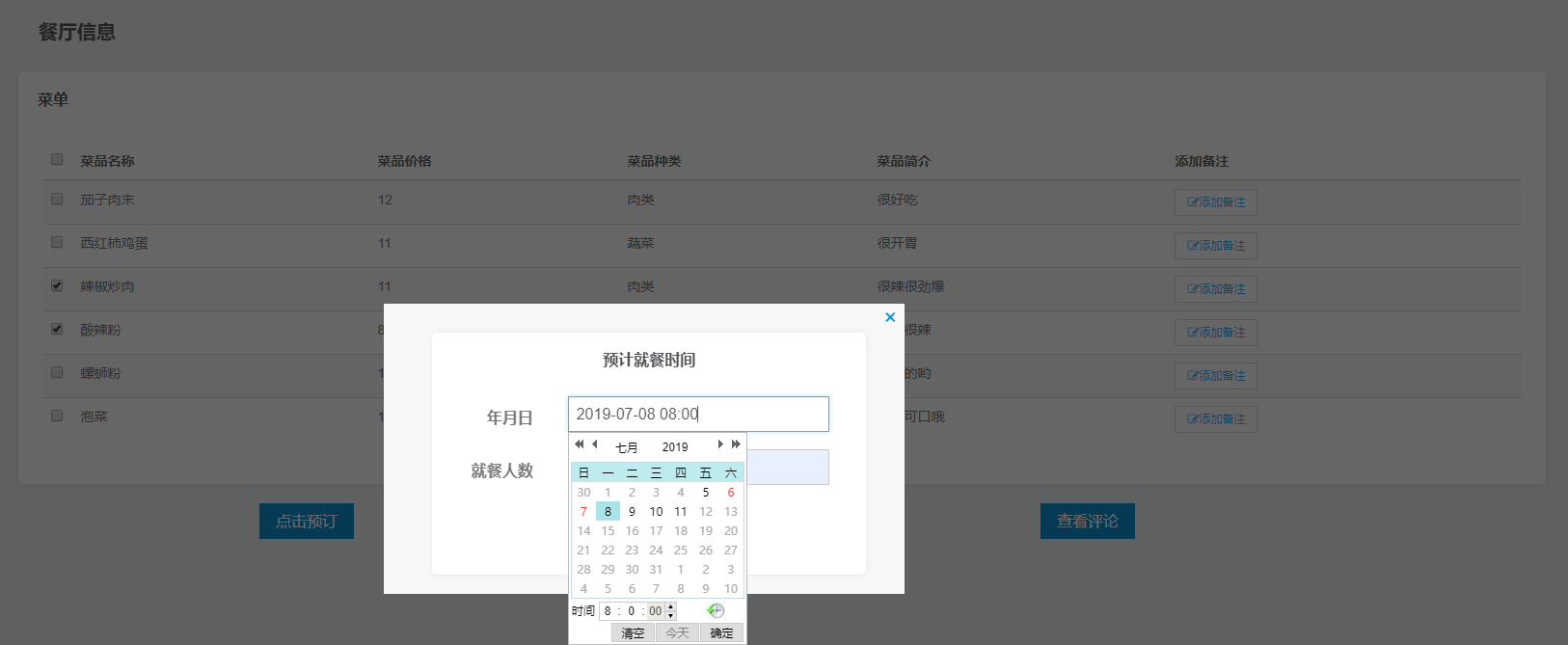


图3.9预定界面



图3.10预定界面

### 3.6订单查询模块

该模块分为两个模块，一个是顾客查看自己所有的订单，无论是已完成订单还是已取消订单还是预定中的订单，并且能随时取消和评论订单。另一个是餐厅的订单查询，查看的是所有顾客的订单，可以随时提示餐厅管理员，客户的还有多久到，是否取消，双击订单能查看点菜的信息。

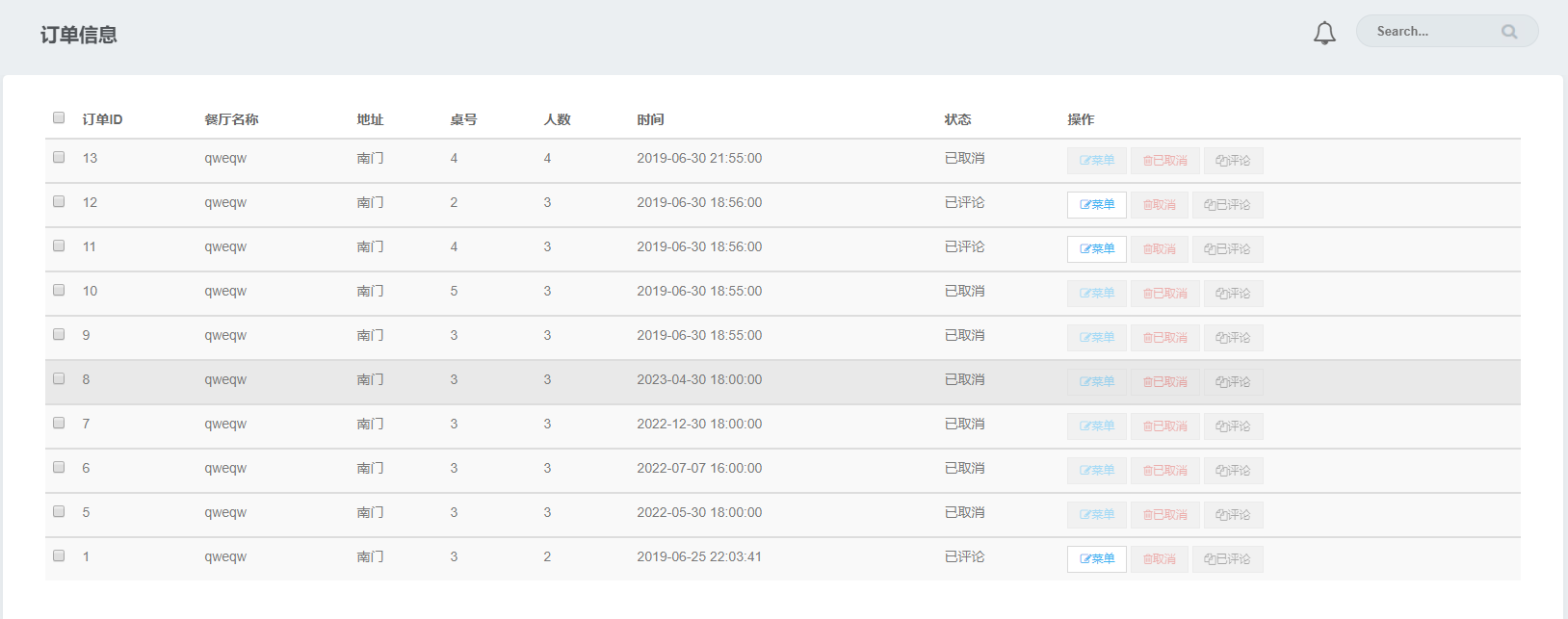


图3.11顾客订单查询



图3.12餐厅订单管理

### 3.7评论查询模块

该模块也分为两种，一个是顾客的评论查询，可以查看自己的锁有评论，并管理评论，另一个是餐厅评论，可以查到所有顾客对他的评论，双击可以查看详细信息



图3.13顾客评论界面



图3.14餐厅评论界面

### 3.8餐厅餐桌管理模块

如图所示，该界面用来显示餐厅的所有餐桌信息，餐桌编号，并能添加餐桌

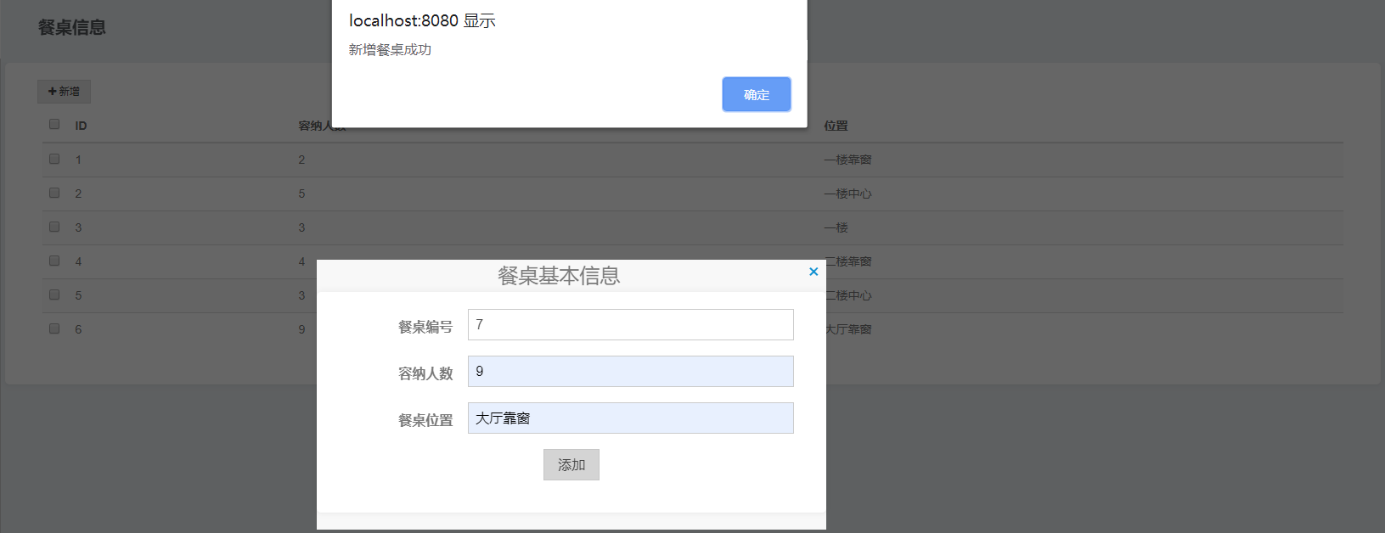


图3.15餐厅餐桌界面

### 3.9餐厅菜单管理模块

如图所示，该界面用来显示餐厅的所有菜品信息，菜品编号，并能批量删除和增加食品



图3.16餐厅菜单管理



图3.17餐厅菜单管理

# 4 总结感想

通过这次课程设计，我收获了很多，一方面学习到了许多以前没学过的专业知识与知识的应用，另一方面还提高了自己动手做项目的能力。本次课程设计，是对我能力的进一步锻炼，也是一种考验。从中获得的诸多收获，也是很可贵的，是非常有意义的。

在课程设计中我学到了许多新的知识。是一个让我把书本上的理论知识运用于实践中的好机会，原来，学的时候感叹学的内容太难懂，现在想来，有些其实并不难，关键在于理解。

在这次课程设计中还锻炼了我其他方面的能力，提高了我的综合素质。首先，它锻炼了我做项目的能力，提高了独立思考问题、自己动手操作的能力，在工作的过程中，复习了以前学习过的知识，并掌握了一些应用知识的技巧。

这次我也遇到很多的难题，就比如说如何才能解决两个顾客同时预定同一个餐厅同一时间段同一餐桌号，这个是一个非常严重的问题，我想了很久，终于能解决一点点了，至少不能同时预定了。还有一个问题，就是用到多线程的时候，程序总是报未知错误，也找了很久，但是还是不能解决，只好先放弃，到后面在解决！

总的来说，我这次好好认清了自己，知道自己的长处和短处，下次一定能做的更好。

# 参考文献

[1] 王珊，萨师煊**.** 数据库系统概论. 第五版. 北京：高等教育出版社，2014

[2] 李春葆，曾平，喻丹丹**.** C#程序设计教程. 第三版. 北京：清华大学出版社，2015

[3] 张海，牟永敏**.** 软件工程导论. 第六版. 北京：清华大学出版社，2013

# 附录

### 代码

#### 1．前端登陆相关JS：

**var *time***=0;  
**var *videoStream***;  
**var *usertype*** = ***document***.getElementById(**"userType"**).**innerHTML**;  
  
**function** *show*(none) {  
 **if**(!none)  
 {  
 ***document***.getElementById(**"videoC"**).**style**.**display**=**"none"**;  
 ***document***.getElementById(**"imgBlog"**).**style**.**display**=**"block"**;  
 }  
 **else** {  
 ***document***.getElementById(**"videoC"**).**style**.**display**=**"block"**;  
 ***document***.getElementById(**"imgBlog"**).**style**.**display**=**"none"**;  
 }  
}  
  
**function** *sLogin*()  
{  
 **$**(**'#my-alert'**).**modal**();  
 *show*(**false**);  
 **$**(**"#tishi"**).text(**""**);  
}  
  
**function** *exit*()  
{  
 *///关闭摄像头资源* **if**(***videoStream***!=**null**)  
 {  
 ***videoStream***.stop();  
 }  
 **$**(**'#my-alert'**).**modal**(**'close'**);  
}  
  
  
**function** *getUserMedia*(constraints, success, error) {  
 **if** (***navigator***.**mediaDevices**.getUserMedia) {  
 *//最新的标准API* ***navigator***.**mediaDevices**.getUserMedia(constraints).then(success).catch(error);  
 } **else if** (***navigator***.webkitGetUserMedia) {  
 *//webkit核心浏览器* ***navigator***.**webkitGetUserMedia**(constraints,success, error)  
 } **else if** (***navigator***.mozGetUserMedia) {  
 *//firfox浏览器* ***navigator***.**mozGetUserMedia**(constraints, success, error);  
 } **else if** (***navigator***.getUserMedia) {  
 *//旧版API* ***navigator***.getUserMedia(constraints, success, error);  
 }  
}  
  
**function** *success*(stream) {  
 video.**srcObject** = stream;  
 ***videoStream***=stream.getTracks()[0];  
 video.play();  
 *setTimeout*(*uploadImage*,1000)  
}  
  
**function** *error*(error) {  
 ***console***.log(**'访问用户媒体设备失败${error.name}, ${error.message}'**);  
}  
**function** *showVideo*()  
{  
 **if**(***document***.getElementById(**"videoC"**).**style**.**display**==**"none"**) {  
 *show*(**true**);  
 *<!--js代码-->* **if** (***navigator***.**mediaDevices**.getUserMedia || ***navigator***.getUserMedia || ***navigator***.webkitGetUserMedia || ***navigator***.mozGetUserMedia) {  
 *getUserMedia*({**video**: {**width**: 200, **height**: 200}}, *success*, *error*);  
 } **else** {  
 *alert*(**'不支持访问用户媒体'**);  
 }  
 }  
}  
  
**function** *uploadImage*(){  
 **var** video = ***document***.getElementById(**'video'**);  
 **var** canvas = ***document***.getElementById(**'canvas'**);  
 **var** context = canvas.getContext(**'2d'**);  
  
 canvas.**width** = 300;  
 canvas.**height** = 300;  
 context.drawImage(video, 0, 0, 300, 300);  
 **var** imgData = canvas.toDataURL();  
  
 ***time***+=1;  
 **if**(***time***==10)  
 {  
 *alert*(**"人脸检测失败，请使用账号密码登陆"**);  
 *show*(**false**);  
 *exit*();  
 **return** ;  
 }  
  
 *//上传到后台。* **var** uploadAjax = **$**.ajax({  
 **type**: **"post"**,  
 *//后端需要调用的地址* **url**:**"/facelogin"**,  
 **data**: {**"imgData"**: imgData,**"userType"**:***usertype***},  
 *//设置超时* **timeout**:10000,  
 **async**: **true**,  
 success: **function** (data) {  
 *//成功后回调* **if**(data.**success**)  
 {  
 *alert*(data.**datas**.**message**);  
 ***window***.**history**.go(0);  
 }**else** {  
 **$**(**"#tishi"**).text(data.**datas**.**message**);  
 *uploadImage*();  
 }  
  
 },  
 error: **function**(data) {  
 *alert*(**"内部错误"**);  
 *show*(**false**);  
 **$**(**'#my-alert'**).**modal**(**'close'**);  
 },  
 *//调用执行后调用的函数* complete: **function** (XMLHttpRequest, textStatus) {  
 **if**(textStatus == **'timeout'**){  
 uploadAjax.abort(); *//取消请求  
 //超时提示：请求超时，请重试  
 alert*(**"请求超时，请重试"**)  
 *//请求超时返回首页* closeCard();  
 }  
 }  
 });  
}  
  
  
  
**function** *Login*() {  
  
 **var** username=**$**(**"#username"**).val();  
 **var** password=**$**(**"#password"**).val();  
  
 **if**(username==**""**||password==**""**)  
 {  
 *alert*(**"用户名或者密码不能为空"**);  
 **$**(**"#username"**).focus();  
 **return**;  
 }  
  
  
 **$**.post(**"/login"**,{**"username"**:username,**"password"**:password,**"userType"**:***usertype***},**function** (data) {  
  
 *alert*(data.**datas**.**message**);  
 **if**(data.**success**)  
 {  
 ***window***.**history**.go(0);  
 }  
 })  
}

#### 2. 后端UserController类：

**package** com.yzc.shixun.controller;  
  
**import** com.yzc.shixun.dto.CustInfo;  
**import** com.yzc.shixun.dto.UserInfo;  
**import** com.yzc.shixun.result.ResultEnum;  
**import** com.yzc.shixun.result.ResultInfo;  
**import** com.yzc.shixun.result.ReturnResult;  
**import** com.yzc.shixun.service.UserService;  
**import** com.yzc.shixun.tool.MailUtil;  
**import** com.yzc.shixun.tool.face.FaceDetect;  
**import** com.yzc.shixun.tool.face.FaceSearch;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.stereotype.Controller;  
**import** org.springframework.ui.Model;  
**import** org.springframework.web.bind.annotation.GetMapping;  
**import** org.springframework.web.bind.annotation.PostMapping;  
**import** org.springframework.web.bind.annotation.RequestParam;  
**import** org.springframework.web.bind.annotation.ResponseBody;  
**import** javax.servlet.http.HttpServletRequest;  
**import** java.io.File;  
**import** java.util.Random;  
@Controller  
**public class** UserController {  
  
 @Autowired  
 HttpServletRequest **httpServletRequest**;  
  
 @Autowired  
 UserService **userService**;  
  
 @GetMapping(**"/index"**)  
 **public** String getIndex()  
 {  
 **return "index"**;  
 }  
 @PostMapping(**"/login"**)  
 @ResponseBody  
 **public** ReturnResult loginByUserName(String username,String password,String userType)  
 {  
 UserInfo userInfo = **userService**.selectByUserName(username,userType);  
  
 **if**(userInfo==**null**)**return new** ReturnResult(**new** ResultInfo(ResultEnum.***LOGIN\_FAIL\_USERNAME***));  
  
 **if**(!userInfo.getPassword().equals(password))  
 **return new** ReturnResult(**new** ResultInfo(ResultEnum.***LOGIN\_FAIL\_PASSWORD***));  
  
 **httpServletRequest**.getSession().setAttribute(**"userInfo"**,userInfo);  
 **return new** ReturnResult(**true**,**new** ResultInfo(ResultEnum.***LOGIN\_SUCCESS***));  
 }  
 @PostMapping(**"/facelogin"**)  
 @ResponseBody  
 **public** ReturnResult loginByFace(String imgData,String userType){  
  
 String image=imgData.split(**","**)[1];  
 String detect = FaceDetect.*detect*(image);  
 **if**(detect.equals(**"0"**))**return new** ReturnResult(**new** ResultInfo(ResultEnum.***FACE\_IDENTIFY\_DETECT\_NOTPEOPLE***));  
 **if**(detect.compareTo(**"1"**)>0)**return new** ReturnResult(**new** ResultInfo(ResultEnum.***FACE\_IDENTIFY\_DETECT\_MANYPEOPLE***));  
  
 UserInfo userInfo = **userService**.selectByFace(image,userType);  
 **if**(userInfo==**null**)**return new** ReturnResult(**new** ResultInfo(ResultEnum.***FACE\_IDENTIFY\_SEARCH\_FAIL***));  
 **httpServletRequest**.getSession().setAttribute(**"userInfo"**,userInfo);  
 **return new** ReturnResult(**true**,**new** ResultInfo(ResultEnum.***FACE\_IDENTIFY\_SEARCH\_SUCCESS***));  
 }  
  
 @GetMapping(**"/register"**)  
 **public** String showRegister()  
 {  
 **return "register"**;  
 }  
 @GetMapping(**"/getCode"**)  
 @ResponseBody  
 **public** ReturnResult getCode(String email)  
 {  
 **boolean** b = **userService**.selectByEmail(email);  
 **if**(b) **return new** ReturnResult(**new** ResultInfo(ResultEnum.***GET\_CODE\_FAIL***));  
 Integer code=**new** Random().nextInt(899999) + 100000;  
 **httpServletRequest**.getSession().setAttribute(**"code"**,code.toString());  
 **new** Thread(**new** Runnable() {  
 @Override  
 **public void** run() {  
 MailUtil.*send*(email, code.toString());  
 }  
 }).start();  
 **return new** ReturnResult(**true**,**new** ResultInfo(ResultEnum.***GET\_CODE\_SUCCESS***));  
 }  
 @GetMapping(**"/verification"**)  
 @ResponseBody  
 **public** ReturnResult verification(String code)  
 {  
 String codes= (String) **httpServletRequest**.getSession().getAttribute(**"code"**);  
 **if**(code.equals(codes))  
 {  
 **return new** ReturnResult(**true**, **new** ResultInfo(ResultEnum.***VERIFICATION\_SUCCESS***));  
 }**else** {  
 **return new** ReturnResult(**false**,**new** ResultInfo(ResultEnum.***VERIFICATION\_FAIL***));  
 }  
 }  
  
 @GetMapping(**"/setCustInfo"**)  
 **public** String showSetCustInfo(Model model,@RequestParam(**"email"**)String email,@RequestParam(**"userType"**)String type){  
 model.addAttribute(**"email"**,email);  
 model.addAttribute(**"userType"**,type);  
 **return "setCustInfo"**;  
 }  
 @GetMapping(**"/setRestInfo"**)  
 **public** String showSetRestInfo(Model model,@RequestParam(**"email"**)String email,@RequestParam(**"userType"**)String type){  
 model.addAttribute(**"email"**,email);  
 model.addAttribute(**"userType"**,type);  
 **return "setCustInfo"**;  
 }  
  
 @PostMapping(**"/addImage"**)  
 @ResponseBody  
 **public** ReturnResult getImage(String imgData)  
 {  
 String image=imgData.split(**","**)[1];  
 String detect = FaceDetect.*detect*(image);  
 **if**(detect.equals(**"0"**))**return new** ReturnResult(**new** ResultInfo(ResultEnum.***FACE\_IDENTIFY\_DETECT\_NOTPEOPLE***));  
 **if**(detect.compareTo(**"1"**)>0)**return new** ReturnResult(**new** ResultInfo(ResultEnum.***FACE\_IDENTIFY\_DETECT\_MANYPEOPLE***));  
  
 **httpServletRequest**.getSession().setAttribute(**"imageData"**,image);  
 **return new** ReturnResult(**true**,**new** ResultInfo(ResultEnum.***GET\_IMAGE***));  
 }  
  
 @PostMapping(**"/saveCustInfo"**)  
 @ResponseBody  
 **public** ReturnResult addCustInfo(CustInfo custInfo,UserInfo userInfo)  
 {  
 Object image=**httpServletRequest**.getSession().getAttribute(**"imageData"**);  
 String imageString =image==**null**?**"无"**:image.toString();  
  
 **try** {  
 **userService**.saveCustInfo(custInfo,userInfo,imageString);  
 } **catch** (Exception e) {  
 e.printStackTrace();;  
 **if**(!imageString.equals(**"无"**))  
 {  
 File file =**new** File(**"D:\\javaWeb\\shixun\\src\\main\\resources\\static\\img\\"**+userInfo.getUserName()+**".jpg"**);  
 file.delete();  
 }  
 **return new** ReturnResult(**new** ResultInfo(ResultEnum.***REGISTER\_FAIL***));  
 }  
 **return new** ReturnResult(**true**,**new** ResultInfo(ResultEnum.***REGISTER\_SUCCESS***));  
 }  
  
}

后端OrderController类：

**package** com.yzc.shixun.controller;  
  
**import** com.yzc.shixun.dto.\*;  
**import** com.yzc.shixun.model.CustOrderInfo;  
**import** com.yzc.shixun.result.ResultEnum;  
**import** com.yzc.shixun.result.ResultInfo;  
**import** com.yzc.shixun.result.ReturnResult;  
**import** com.yzc.shixun.service.Custervice;  
**import** com.yzc.shixun.service.OrderService;  
  
**import** com.yzc.shixun.service.RestService;  
**import** com.yzc.shixun.service.TableService;  
**import** net.sf.json.JSONArray;  
**import** net.sf.json.JsonConfig;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.stereotype.Controller;  
**import** org.springframework.ui.Model;  
**import** org.springframework.web.bind.annotation.\*;  
  
**import** javax.servlet.http.HttpServletRequest;  
**import** java.sql.SQLException;  
**import** java.text.DateFormat;  
**import** java.text.ParseException;  
**import** java.text.SimpleDateFormat;  
**import** java.util.Date;  
**import** java.util.List;  
  
@Controller  
**public class** OrderController {  
  
 @Autowired  
 HttpServletRequest **httpServletRequest**;  
  
 @Autowired  
 Custervice **custervice**;  
 @Autowired  
 OrderService **orderService**;  
 @Autowired  
 RestService **restService**;  
  
 @Autowired  
 TableService **tableService**;  
 @GetMapping(**"/restOrderInfos"**)  
 **public** String showRestOrderInfos(Model model){  
  
 UserInfo userInfo= (UserInfo) **httpServletRequest**.getSession().getAttribute(**"userInfo"**);  
 **if**(userInfo!=**null**) {  
 RestInfo restInfo = **restService**.selectByUserId(userInfo.getId());  
  
 List<CustOrderInfo> custOrderInfos = **orderService**.getRestOrderInfos(restInfo.getId());  
 **for**(CustOrderInfo custOrderInfo:custOrderInfos)  
 {  
 **if**(custOrderInfo.getOrderState().equals(**"预定中"**))  
 {  
 Date date=**new** Date();  
  
 **if**((date.getTime()-custOrderInfo.getStartTime().getTime()) >= 7200000)  
 {  
 **try** {  
 **orderService**.setState(custOrderInfo.getOrderid(),**"已完成"**);  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
 custOrderInfo.setOrderState(**"已完成"**);  
 }  
 }  
 }  
  
 model.addAttribute(**"custOrderInfos"**,custOrderInfos);  
 }  
 **return "restOrderInfos"**;  
 }  
  
 @GetMapping(**"/custOrderInfos"**)  
 **public** String showCustOrderInfos(Model model){  
  
 UserInfo userInfo= (UserInfo) **httpServletRequest**.getSession().getAttribute(**"userInfo"**);  
 **if**(userInfo!=**null**) {  
 CustInfo custInfo = **custervice**.selectById(userInfo.getId());  
 List<CustOrderInfo> custOrderInfos = **orderService**.getCustOrderInfos(custInfo.getId());  
 **for**(CustOrderInfo custOrderInfo:custOrderInfos)  
 {  
 **if**(custOrderInfo.getOrderState().equals(**"预定中"**))  
 {  
 Date date=**new** Date();  
  
 **if**((date.getTime()-custOrderInfo.getStartTime().getTime()) >= 7200000)  
 {  
 **try** {  
 **orderService**.setState(custOrderInfo.getOrderid(),**"已完成"**);  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
 custOrderInfo.setOrderState(**"已完成"**);  
 }  
 }  
 }  
  
 model.addAttribute(**"custOrderInfos"**,custOrderInfos);  
 }  
 **return "custOrderInfos"**;  
 }  
  
  
 @PostMapping(**"/addOrder"**)  
 @ResponseBody  
 **public** ReturnResult addOrder(String foodList, Integer restId, String date, Integer peopleNum)  
 {  
 DateFormat simpleDateFormat=**new** SimpleDateFormat(**"yyyy-MM-dd HH:mm"**);  
 System.***out***.println(date);  
 Date startTime;  
 **try** {  
 startTime=simpleDateFormat.parse(date);  
 System.***out***.println(startTime.toString());  
 } **catch** (ParseException e) {  
 **return new** ReturnResult(**new** ResultInfo(ResultEnum.***TIME\_FORMAT\_FAIL***));  
 }  
 Integer tableId = **tableService**.getTableIdByTime(restId, startTime, peopleNum);  
 **if**(tableId==-1)  
 {  
 **return new** ReturnResult(**new** ResultInfo(ResultEnum.***ADD\_ORDER\_FAIL***));  
 }  
  
 UserInfo userInfo = (UserInfo) **httpServletRequest**.getSession().getAttribute(**"userInfo"**);  
 CustInfo custInfo=**custervice**.selectById(userInfo.getId());  
 OrderInfo orderInfo=**new** OrderInfo();  
 orderInfo.setCustId(custInfo.getId());  
 orderInfo.setPeopleNum(peopleNum);  
 orderInfo.setRestId(restId);  
 orderInfo.setTableId(tableId);  
 orderInfo.setStartTime(startTime);  
  
 JSONArray foodJson=JSONArray.*fromObject*(foodList);  
 List<OrderFood> orderFoods=JSONArray.*toList*(foodJson, **new** OrderFood(), **new** JsonConfig());  
  
 **try** {  
 **orderService**.addOrder(orderInfo, orderFoods);  
 **return new** ReturnResult(**true**,**new** ResultInfo(ResultEnum.***ADD\_ORDER\_SUCCESS***));  
 } **catch** (Exception e) {  
 **return new** ReturnResult(**new** ResultInfo(ResultEnum.***SYSTEM\_FAIL***));  
 }  
 }  
 @PostMapping(**"/cancelOrder"**)  
 @ResponseBody  
 **public** ReturnResult cancelOrder(Integer orderId)  
 {  
 **try** {  
 **orderService**.cancelOrder(orderId);  
 **return new** ReturnResult(**true**,**new** ResultInfo(ResultEnum.***CANCEL\_ORDER\_SUCCESS***));  
 } **catch** (Exception e) {  
 **return new** ReturnResult(**new** ResultInfo(ResultEnum.***SYSTEM\_FAIL***));}  
 }  
}