

《餐厅点餐系统》

详 细 分 析 与 设 计

V 1.0.0

**目 录**

[第一部分 引言 5](#_Toc428303611)

[一、编写目的 5](#_Toc428303612)

[二、项目背景 5](#_Toc428303613)

[三、参考资料 5](#_Toc428303613)

[第二部分 项目概述 6](#_Toc428303614)

[第三部分 总体设计](#_Toc318310232) 7

[一、技术架构设计](#_Toc318310233) 7

[1、MVC模型设计](#_Toc428303628) 7

[2、Model,Controller,View相互通讯的规则](#_Toc428303628) 8

[3、Web Service技术](#_Toc428303628) 8

[二、SSH Web环境下的集成配置](#_Toc428303628) 9

[1、web.iml](#_Toc428303628) 9

[2、misc.xml](#_Toc428303628) 9

[3、modules.xml](#_Toc428303628) 9

[4、workspace.xml](#_Toc428303628) 10

[第四部分 界面设计 1](#_Toc428303629)3

[一、PC界面设计 1](#_Toc428303630)3

[1、用户登录主页面 1](#_Toc428303628)3

[2、前台主页面 1](#_Toc428303628)4

[3、菜品管理员主页面 1](#_Toc428303628)4

[4、最高管理员主页面 1](#_Toc428303628)5

[5、订单菜品管理页面 1](#_Toc428303628)5

[6、订单结算页面 1](#_Toc428303628)6

[7、厨师页面 1](#_Toc428303628)6

[8、经营流水查看页面 1](#_Toc428303628)6

[二、移动端界面设计 1](#_Toc428303631)8

[1、登录界面设计 1](#_Toc428303628)8

[2、主界面 1](#_Toc428303628)9

[第五部分 单元模块设计 2](#_Toc428303632)0

[一、类图 2](#_Toc428303633)0

[二、数据访问层设计](#_Toc428303633) 20

[1、类的详细设计描述](#_Toc318310284) 20

[1.1、AdminDao接口设计 2](#_Toc318310284)0

[1.2、DishMapper接口设计 2](#_Toc318310284)1

[1.3、OrderMapper接口设计 2](#_Toc318310284)2

[1.4、UserMapper接口设计 2](#_Toc318310284)2

[1.5、DataMapper接口设计 23](#_Toc318310284)

[1.6、PowerMapper接口设计 23](#_Toc318310284)

[1.7、DAO接口实现类汇总 2](#_Toc318310284)4

[三、业务逻辑层设计 2](#_Toc318310284)4

[1、类的详细设计描述 2](#_Toc318310284)4

[1.1、AdminService接口设计 2](#_Toc318310284)4

[1.2、DishService接口设计 2](#_Toc318310284)5

[1.3、OrderService接口设计 2](#_Toc318310284)5

[1.4、UserService接口设计 2](#_Toc318310284)6

[1.5、DataService接口设计 2](#_Toc318310284)6

[1.6、PowerService接口设计 2](#_Toc318310284)7

[1.7、业务接口实现类汇总 2](#_Toc318310284)7

[四、系统管理 2](#_Toc318310284)7

[1、员工管理 2](#_Toc428303634)7

[1.1、功能汇总 2](#_Toc428303634)7

[1.2、类图设计 2](#_Toc428303634)8

[1.3、时序图 2](#_Toc428303634)9

[1.4、实体类和接口详细设计描述 2](#_Toc428303634)9

[1.4.1、AdminController描述 2](#_Toc428303634)9

[1.4.2、DataController描述 3](#_Toc428303634)0

[1.5、功能实现 3](#_Toc428303634)0

[2、用户登录注册 3](#_Toc318310284)0

[2.1、功能汇总 3](#_Toc428303634)0

[2.2、类图设计 3](#_Toc428303634)1

[2.3、时序图 3](#_Toc428303634)2

[2.4、实体类和接口详细设计描述 3](#_Toc428303634)2

[2.4.1、UserController描述 3](#_Toc428303634)2

[2.5、功能实现 3](#_Toc428303634)3

[3、订单管理 3](#_Toc318310284)3

[3.1、功能汇总 3](#_Toc428303634)3

[3.2、类图设计 3](#_Toc428303634)4

[3.2.1、类关系结构图 3](#_Toc428303634)4

[3.2.2、时序图 3](#_Toc428303634)4

[3.2.2.1、订单管理时序图设计 3](#_Toc428303634)4

[3.2.2.2、订单菜品管理时序图设计 3](#_Toc428303634)5

[3.2.2.3、订单结算时序图设计 3](#_Toc428303634)5

[3.3、实体类和接口详细设计描述 3](#_Toc428303634)6

[3.3.1、OrderController描述 3](#_Toc428303634)6

[3.3.2、DishController描述 3](#_Toc428303634)7

[3.3.3、页面描述 3](#_Toc428303634)8

[3.4、功能实现 3](#_Toc428303634)9

[4、经营数据流水查询 3](#_Toc318310284)9

[4.1、功能汇总 3](#_Toc428303634)9

[4.2、类图设计 4](#_Toc428303634)0

[第六部分呢 数据库设计 4](#_Toc318310284)1

[一、数据库整体结构图 4](#_Toc318310284)1

[二、数据表清单 4](#_Toc318310284)1

[1、user表结构 4](#_Toc318310284)2

[2、order表结构 4](#_Toc318310284)2

[3、order\_dish表结构 43](#_Toc318310284)

[4、dish表结构 43](#_Toc318310284)

[5、admin表结构 4](#_Toc318310284)4

[三、外键清单 4](#_Toc318310284)4

[第七部分 补充设计和说明 4](#_Toc318310284)5

[一、编译运行环境设计 4](#_Toc318310284)5

[二、包路径与WEB目录结构设计 4](#_Toc318310284)7

[1、包路径设计 4](#_Toc318310284)7

[2、WEB目录结构设计 4](#_Toc318310284)9

第一部分 引言

### 一、编写目的

编写本设计的目的是为了准确阐述基于餐厅点餐系统的具体实现思路和方法，即系统的详细架构和实现逻辑，主要包括程序系统的结构以及各层次中每个程序的设计考虑。预期读者为项目全体成员，包括运行维护和测试人员。

### 二、项目背景

1、系统名称：餐厅点餐系统

2、任务提出者：中国高科

3、开发者：武大实训02组

### 三、参考资料

1、本项目的经核准的计划任务书或合同、上级机关的批文；

《每日工作计划与管理》

《里程碑计划》

2、属于本项目的其他已发表的文件；

《需求说明》

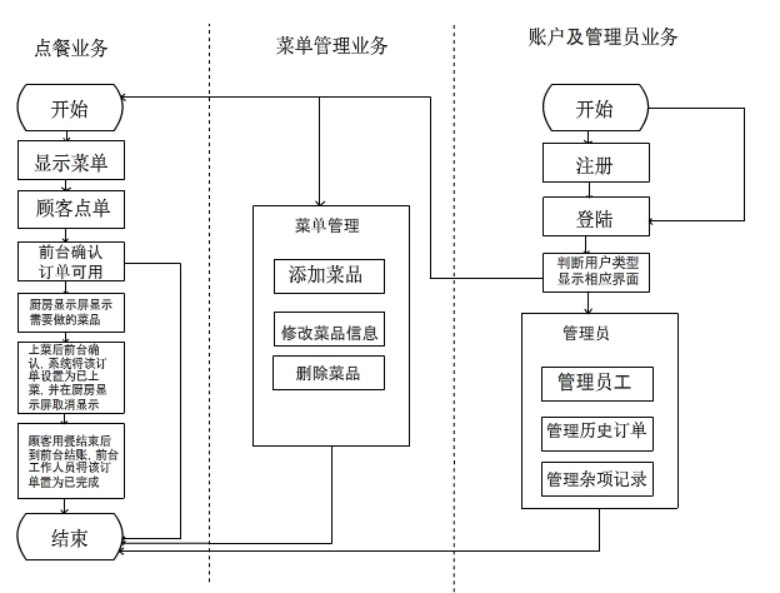
《概要设计》

3、本文件中各处引用到的文件资料，包括所要用到的软件开发标准。

《Java语言编码规范(Java Code Conventions)》

第二部分 项目概述

随着现代社会的城市化，城市规模和人数不断增长，同时也带动了餐饮业的发展，但是餐馆由于受到空间大小的影响，盈利率很难提高，想要增加更多的盈利就必须加快服务的效率，同时带动消费效率的提高。当前餐厅中的大多数环节都是由服务员跑动来完成，这样会造成餐厅使用的劳动力过多，而且费时费力，运营效率低下。顺应现代餐馆的要求，我们研发了一个全新的餐厅点餐系统来解决这些问题。客户通过餐厅提供的二维码即可实现手机点餐，餐厅服务人员也能及时收到反馈提供服务，大大提升了服务效率，同时带动消费效率提高，使得餐厅盈利率提高。

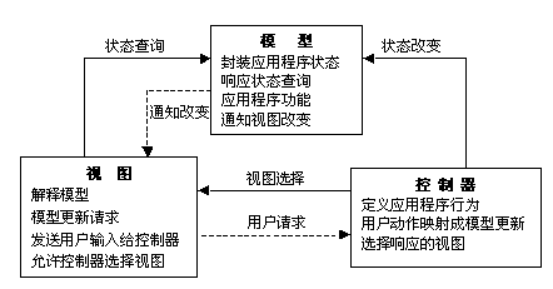


第三部分 总体设计

### 一、技术架构设计

#### 1、MVC模型设计

系统实现的MVC系统架构，使用Android SDK29.0.5标准库，UIView的界面层，UIViewController负责架构结合，NSUserDefaults和CoreData作为模型层进行数据的持久化。



1）Controller层

每个业务功能的主操作场景采用一个核心控制器类，利用iPhone的SDK对移动终端设备的先进场景切换设计，由核心控制器控制不同业务处理视图的切换。

View层通过事件、或者异步线程模式调用控制器的函数，达到把客户响应汇集到控制器。

Model层通过Delegate托管，回调Controller中实现的协议方法把数据汇集到控制器。

2）View层

View负责数据显示与数据输入。数据的输入通过事件交给Model层处理，数据显示通过Model层的托管实现刷新。

3）Model层

Model层服务数据处理，负责向服务器请求数据，并把数据处理好传递给托管。交给View层完成。

#### 2、Model，Controller，View相互通讯的规则

1） Controller可以直接和Model通信

2） Controller也可以直接和View通信

3） Model和View永远不能直接通信

4） View和Controller的通信是透明和固定的，主要通过outlet和action实现

5） View使用Delegate接口和Controller同步信息

6） View不直接和数据通信，使用dataSource接口从Controller处获取数据

7） View的delegate和dataSource一般就是Controller

8） Controller负责为View翻译和格式化Model的数据

9） Model使用Notification & KVO的方式分发数据更新信息，Controller可以有选择的监听自己感兴趣的信息。

10） View也可以监听广播信息，但一般不是Model发出的信息

11） 一个完整的App就是很多MVC的集合

#### 3、Web Service技术

移动端的所有数据请求采用Web Service技术，在后台应用设计一个WebService服务器服务提供，通信的数据格式采用XML。

在移动端大量处理XML的解析处理。

### 二、SSH Web环境下的集成配置

#### 1、web.iml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <module type="WEB\_MODULE" version="4">  <component name="NewModuleRootManager">  <content url="file://$MODULE\_DIR$">  <excludeFolder url="file://$MODULE\_DIR$/.tmp" />  <excludeFolder url="file://$MODULE\_DIR$/temp" />  <excludeFolder url="file://$MODULE\_DIR$/tmp" />  </content>  <orderEntry type="inheritedJdk" />  <orderEntry type="sourceFolder" forTests="false" />  </component>  </module> |

#### 2、misc.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <project version="4">  <component name="JavaScriptSettings">  <option name="languageLevel" value="ES6" />  </component>  </project> |

#### 3、modules.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <project version="4">  <component name="ProjectModuleManager">  <modules>  <module fileurl="file://$PROJECT\_DIR$/.idea/web.iml" filepath="$PROJECT\_DIR$/.idea/web.iml" />  </modules>  </component>  </project> |

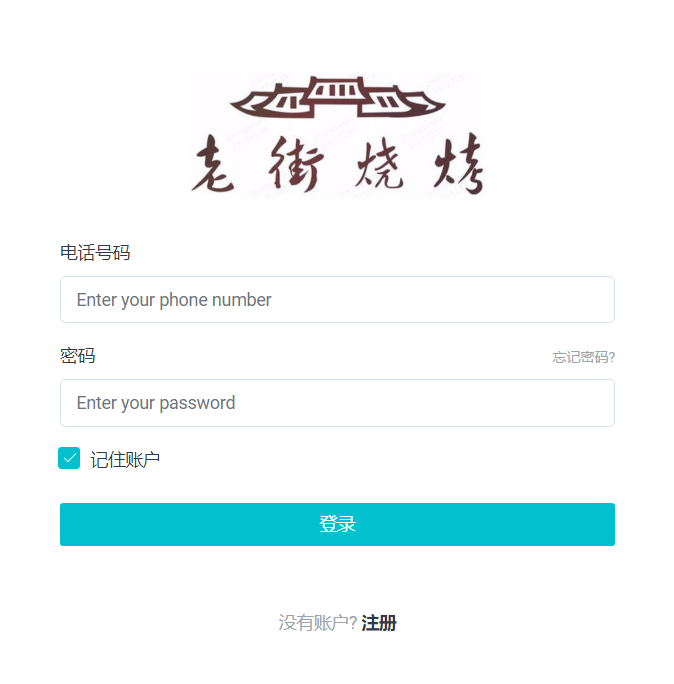
#### 4、workspace.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <project version="4">  <component name="ChangeListManager">  <list default="true" id="1e28a0b1-2afc-458a-85cb-414b1c97054b" name="Default Changelist" comment="" />  <option name="SHOW\_DIALOG" value="false" />  <option name="HIGHLIGHT\_CONFLICTS" value="true" />  <option name="HIGHLIGHT\_NON\_ACTIVE\_CHANGELIST" value="false" />  <option name="LAST\_RESOLUTION" value="IGNORE" />  </component>  <component name="Git.Settings">  <option name="RECENT\_GIT\_ROOT\_PATH" value="$PROJECT\_DIR$/.." />  </component>  <component name="ProjectConfigurationFiles">  <option name="files">  <list>  <option value="$PROJECT\_DIR$/.idea/web.iml" />  <option value="$PROJECT\_DIR$/.idea/misc.xml" />  <option value="$PROJECT\_DIR$/.idea/modules.xml" />  <option value="$PROJECT\_DIR$/.idea/vcs.xml" />  </list>  </option>  </component>  <component name="ProjectId" id="1fvowIlh5F6xbJvNC8O6nE3A4JH" />  <component name="ProjectLevelVcsManager" settingsEditedManually="true" />  <component name="ProjectViewState">  <option name="hideEmptyMiddlePackages" value="true" />  <option name="showExcludedFiles" value="true" />  <option name="showLibraryContents" value="true" />  </component>  <component name="PropertiesComponent">  <property name="RunOnceActivity.ShowReadmeOnStart" value="true" />  <property name="SHARE\_PROJECT\_CONFIGURATION\_FILES" value="true" />  <property name="WebServerToolWindowFactoryState" value="false" />  <property name="aspect.path.notification.shown" value="true" />  <property name="last\_opened\_file\_path" value="$PROJECT\_DIR$" />  <property name="nodejs\_interpreter\_path.stuck\_in\_default\_project" value="undefined stuck path" />  <property name="nodejs\_npm\_path\_reset\_for\_default\_project" value="true" />  <property name="settings.editor.selected.configurable" value="project.propVCSSupport.Mappings" />  </component>  <component name="SvnConfiguration">  <configuration />  </component>  <component name="TaskManager">  <task active="true" id="Default" summary="Default task">  <changelist id="1e28a0b1-2afc-458a-85cb-414b1c97054b" name="Default Changelist" comment="" />  <created>1596451115901</created>  <option name="number" value="Default" />  <option name="presentableId" value="Default" />  <updated>1596451115901</updated>  <workItem from="1596451117021" duration="465000" />  <workItem from="1596545683967" duration="206000" />  <workItem from="1596771722354" duration="4424000" />  <workItem from="1597108164227" duration="3898000" />  <workItem from="1597197153860" duration="1275000" />  <workItem from="1597198758489" duration="960000" />  <workItem from="1597282153892" duration="958000" />  <workItem from="1597284287874" duration="709000" />  <workItem from="1597285776324" duration="1921000" />  <workItem from="1597287889207" duration="845000" />  <workItem from="1597288855944" duration="895000" />  <workItem from="1597299455216" duration="13000" />  <workItem from="1597368324532" duration="2365000" />  <workItem from="1597453639031" duration="466000" />  <workItem from="1597488593834" duration="8674000" />  <workItem from="1597579352340" duration="6000" />  <workItem from="1597627882983" duration="125000" />  <workItem from="1597638260048" duration="130000" />  <workItem from="1597638518970" duration="248000" />  </task>  <servers />  </component>  <component name="TypeScriptGeneratedFilesManager">  <option name="version" value="1" />  </component>  <component name="WindowStateProjectService">  <state x="549" y="168" key="FileChooserDialogImpl" timestamp="1597638765105">  <screen x="0" y="0" width="1536" height="824" />  </state>  <state x="549" y="168" key="FileChooserDialogImpl/0.0.1536.824@0.0.1536.824" timestamp="1597638765105" />  <state x="256" y="65" key="SettingsEditor" timestamp="1597285788876">  <screen x="0" y="0" width="1536" height="824" />  </state>  <state x="256" y="65" key="SettingsEditor/0.0.1536.824@0.0.1536.824" timestamp="1597285788876" />  </component>  </project> |

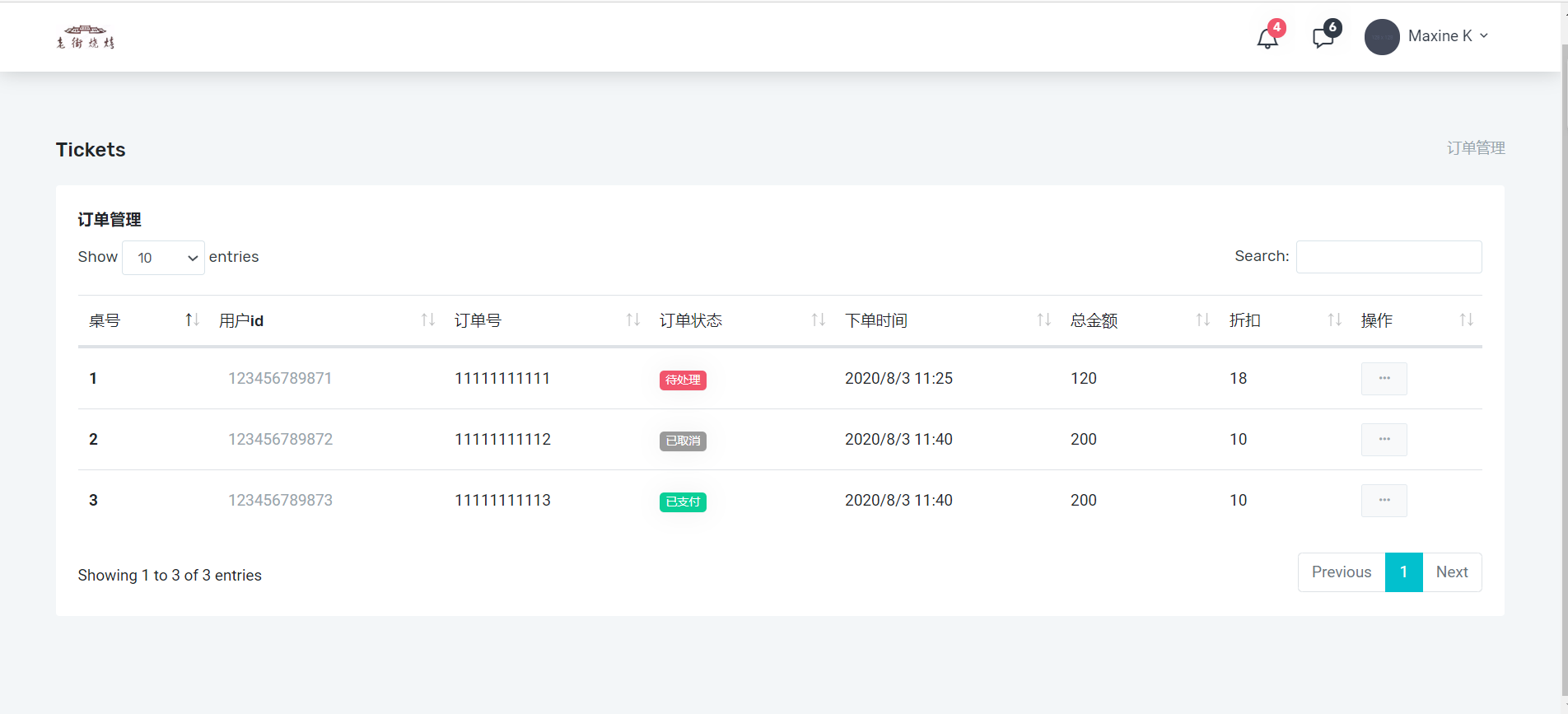
第四部分 界面设计

### 一、PC界面设计

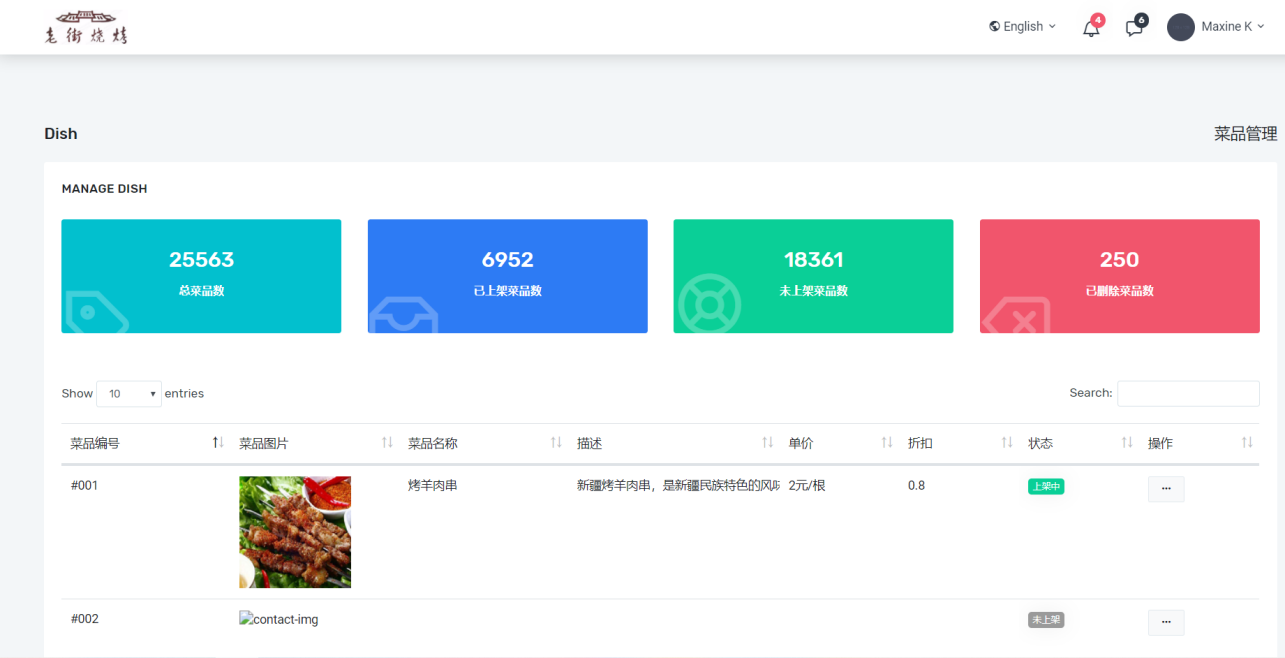
#### 1、用户登录主页面



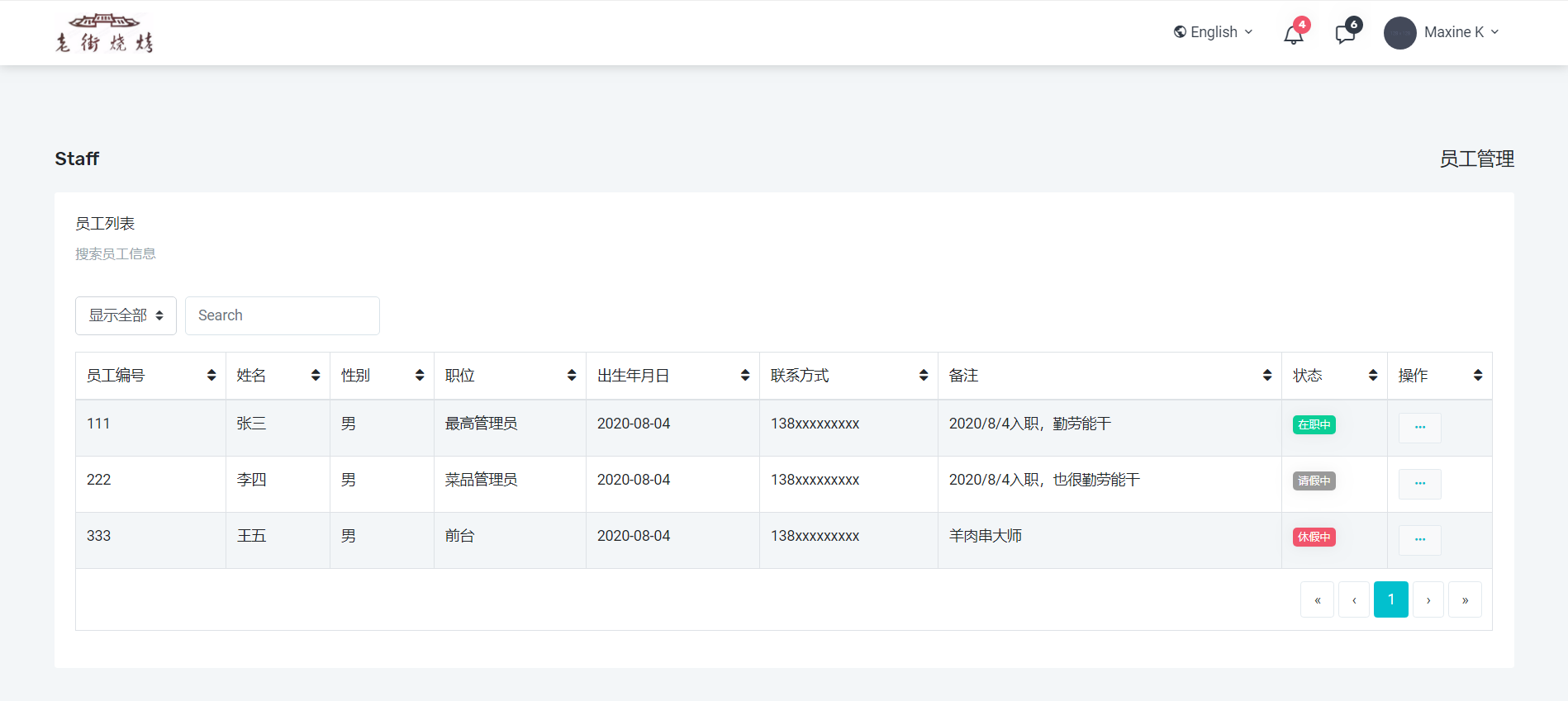
#### 2、前台主页面



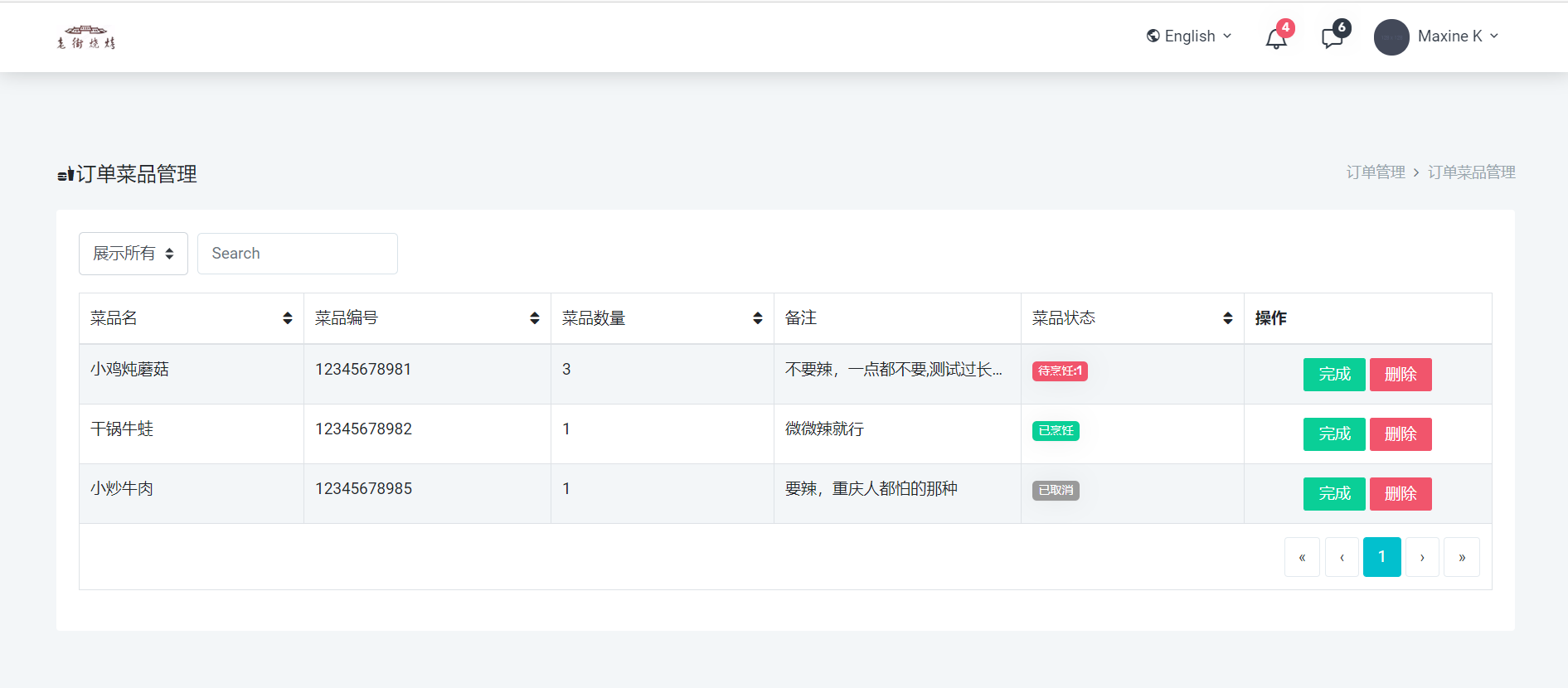
#### 3、菜品管理员主页面



#### 4、最高管理员主页面



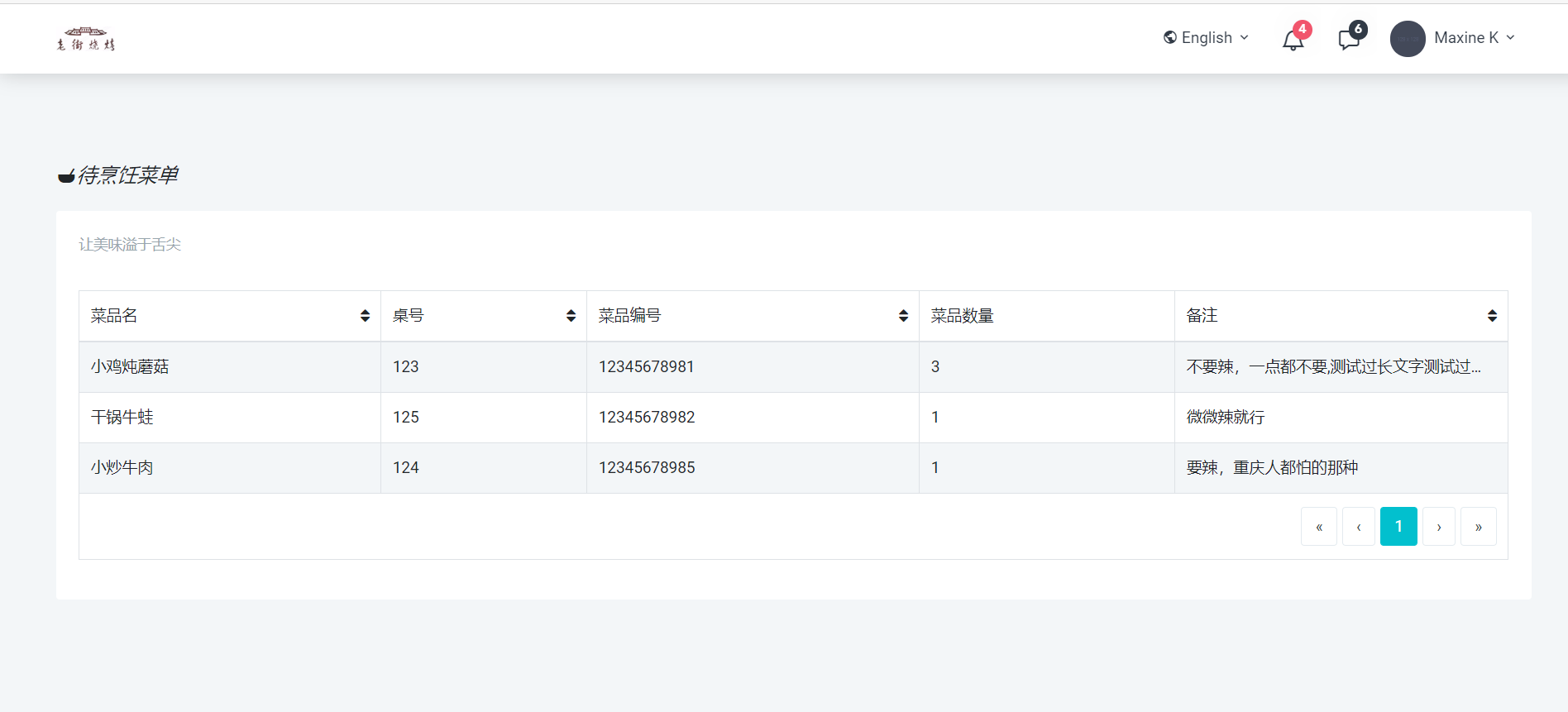
#### 5、订单菜品管理页面



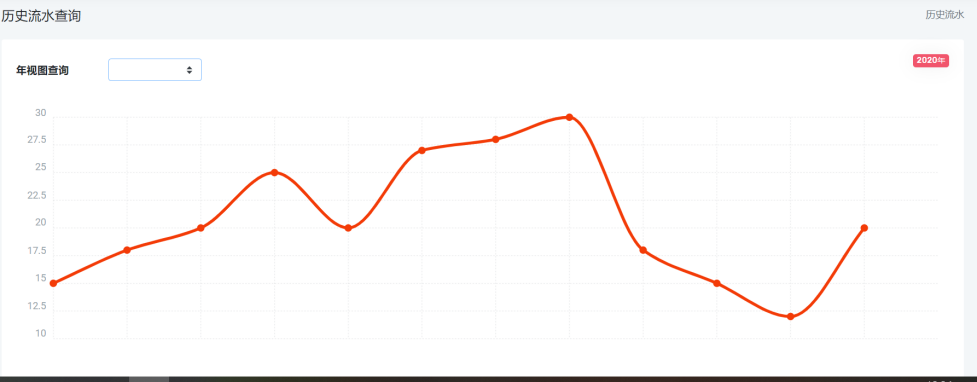
#### 6、订单结算页面



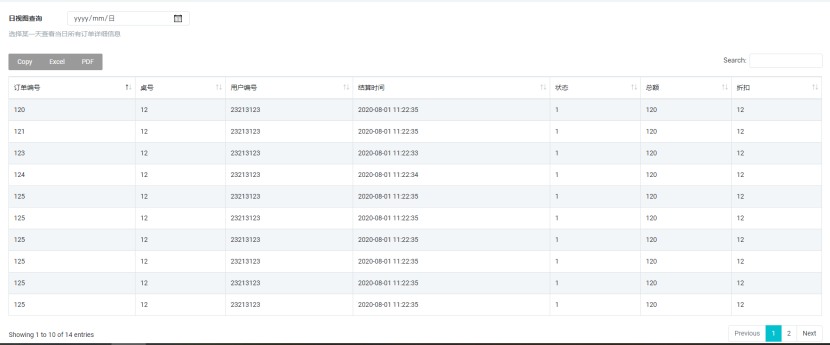
#### 7、厨师界面



#### 8、经营流水查看页面

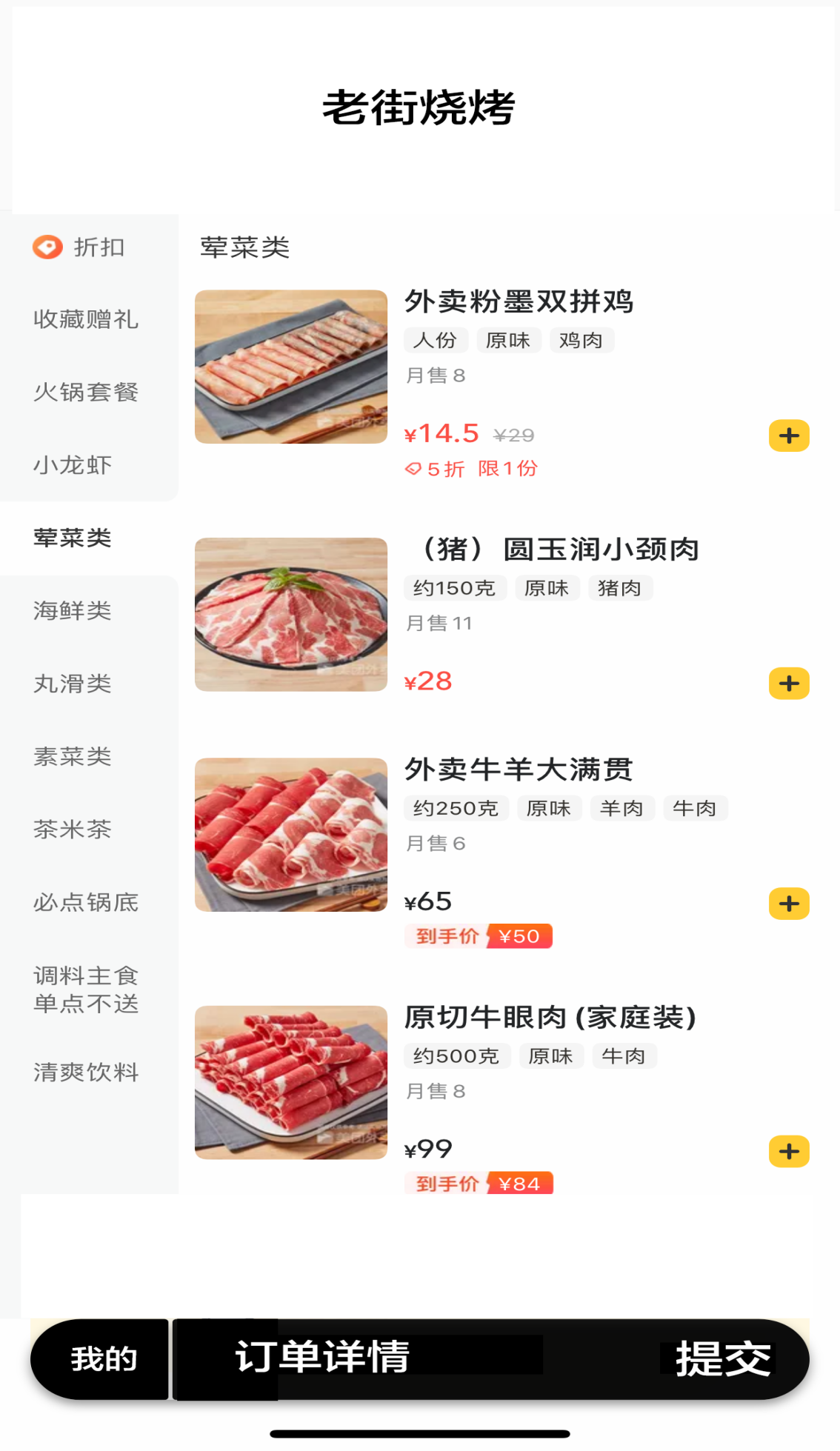






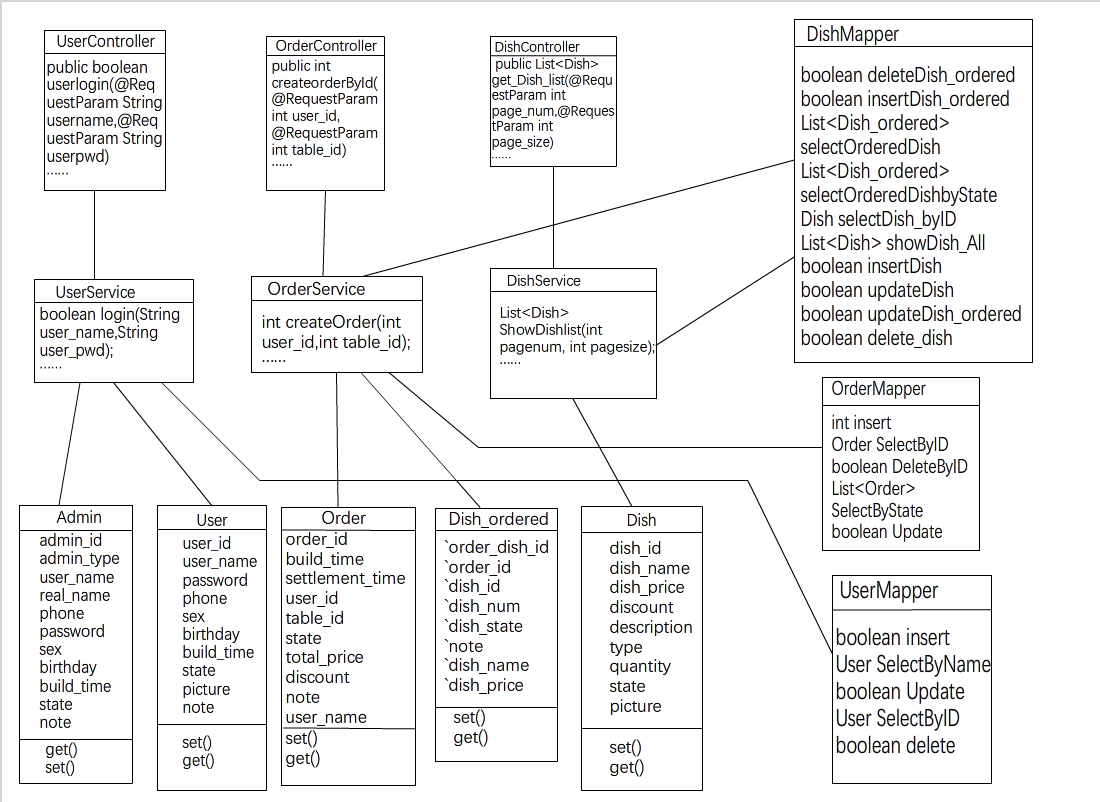
### 二、移动端界面设计

**1、登录界面设计**

**2、主界面**

第五部分 单元模块设计

### 一、类图



类图（总）

### 二、数据访问层设计

#### 1、类的详细设计描述

##### 1.1 AdminDao接口设计

详细描述：

|  |
| --- |
| AdminMapper系统管理员 |
| List<AdminBrief> getAdminBrief();  获取全部员工信息 |
| List<Admin> getAdmin(@Param("id")int id);  获取某个员工的详细信息 |
| int newAdmin(@Param("admin")Admin admin);  新建员工 |
| int deleteAdmin(@Param("admin\_id")int id);  删除员工 |
| int updateAdminById(@Param("id")  int id,@Param("admin\_type")  int admin\_type,@Param("user\_name")  String name,@Param("real\_name")  String real\_name,@Param("phone")  String phone,@Param("password")  String password,@Param("sex")String sex,@Param("birthday")  String birthday,@Param("build\_time")  String build\_time,@Param("state")  String state,@Param("note")String note);  更新员工 |
| List<AdminBrief> getAdminBriefByPage(@Param("pageNO")  int pageNo,@Param("pageSize")int pageSize);  分页查询全部员工信息 |
| int getCount();  获取员工数量 |

##### 1.2 DishMapper接口设计

详细描述：

|  |
| --- |
| boolean deleteDish\_ordered(int order\_id,int dish\_id)：根据订单号和菜品号删除订单内的菜品 |
| boolean insertDish\_ordered(Dish\_ordered dish\_ordered);  插入订单中的菜品 |
| List<Dish\_ordered> selectOrderedDish(int order\_id,int offset,int pagesize);  根据订单号查询订单中包含的菜品 |
| List<Dish\_ordered> selectOrderedDishbyState(int state,int offset,int pagesize);  根据制作状态查询分页查询被点菜品 |
| Dish selectDish\_byID(int dish\_id);  根据菜品号查询菜品 |
| List<Dish> showDish\_All(int offset,int pagesize);  分页查询所有菜品 |
| boolean insertDish(Dish dish);  插入菜品 |
| boolean updateDish(Dish dish);  更新菜品 |
| boolean updateDish\_ordered(Dish\_ordered dish\_ordered);  更新订单内的菜品 |
| boolean delete\_dish(int dish\_id);  删除菜品 |

##### 1.3 OrderMapper接口描述

详细描述：

|  |
| --- |
| int insert(Order order);  插入订单 |
| Order SelectByID(int order\_id);  根据订单号查询订单 |
| boolean DeleteByID(int order\_id);  根据订单号删除订单 |
| List<Order> SelectByState(int State,int offset,int pagesize);  根据订单状态显示订单 |
| boolean Update(Order order);  更新订单 |

##### 1.4 UserMapper接口设计

详细描述：

|  |
| --- |
| boolean insert(User user);  插入用户 |
| User SelectByName(String name);  通过用户名查询用户 |
| boolean Update(User user);  更新用户信息 |
| User SelectByID(int user\_id);  通过用户id查询 |
| boolean delete(int user\_id);  通过用户id删除用户 |

##### 1.5 DataMapper接口设计

|  |
| --- |
| DataMapper 经营数据 |
| List<Order> getAllOrder();  获取全部数据 |
| List<Order> getOrderBetween(@Param("begintime")  String begintime,@Param("endtime")String endtime);  获取某个时间段数据 |
| List<Order> getTodayOrder(@Param("date")String date);  获取今天数据 |
| int getokcount(@Param("state")int state);  获取数量 |

##### 1.6 PowerMapper接口设计

|  |
| --- |
| powerMapper系统管理员 |
| List<Power> getPower();  获取全部权限信息 |
| Power getPowerByid(@Param("username")String username);  获取某个id的权限 |
| int setPowerById(@Param("id")int id,@Param("power")String power);  修改权限 |

##### 1.7 DAO接口实现类汇总

|  |  |
| --- | --- |
| **接口** | **实现** |
| DishMapper | DishMapper.xml |
| OrderMapper | OrderMapper.xml |
| ICheckDao | ICheckDaoImpl |

### 三、业务逻辑层设计

#### 1、类的详细设计描述

##### 1.1 AdminService接口设计

|  |
| --- |
| AdminService管理员服务 |
| List<AdminBrief> getAdminBrief(); 获得全部信息 |
| List<Admin> getAdmin(int id) 获取详细信息通过id |
| int newAdmin(Admin admin); 新建管理员 |
| int deleteAdmin(int id); 删除管理员通过id |
| int updateAdminById(  int id,  int admin\_type,  String name,  String real\_name,  String phone,  String password,  String sex,  String birthday,  String build\_time,  String state,  String note);  更新管理员信息通过id |
| List<AdminBrief> getAdminBriefByPage(int pageNo,int pageSize);分页查询 |
| int getCount();  获取数量 |
| int adminLogin(String username,String password);  管理员登录 |
| int adminLoginPhone(String phone,String password);  管理员登录 |

##### 1.2 DishService接口设计

详细描述：

|  |
| --- |
| List<Dish> ShowDishlist(int pagenum, int pagesize);  分页显示菜品列表 |
| boolean insertdish(String dish\_name,float dish\_price,float discount,String discription,String type,int quantity,int state,String picture);  根据菜品信息添加菜品 |
| boolean updatedish\_byid(int dish\_id,String dish\_name,float dish\_price,float discount,String description,String type,int quantity,int state,String picture);  根据前台发送的数据更新菜品信息 |
| boolean delete\_dish(int dish\_id);  根据前台发送的菜品号删除菜品 |

##### 1.3 OrderService接口设计

详细描述：

|  |
| --- |
| int createOrder(int user\_id,int table\_id);  根据用户id和桌号创建订单 |
| boolean insertDishbyid(int order\_id,int dish\_id,int dish\_num);  根据订单号添加菜品 |
| boolean deleteDishbyid(int order\_id,int dish\_id);  根据订单号删除菜品 |
| List<Dish> ShowDishlist(int pagenum, int pagesize);  分页显示菜品列表 |
| List<Order> ShowOrderByState(int state,int pagenum, int pagesize);  根据状态显示订单 |
| List<Dish\_ordered> ShowDish\_orderedByState(int state,int pagenum, int pagesize);  根据状态显示菜品列表 |
| Order Show\_totalPrice(int order\_id,int user\_id);  计算总价 |
| List<Dish\_ordered> ShowOrderedDish(int order\_id,int pagesize,int pagenum);  显示订单内菜品 |
| boolean UpdateDish\_Ordered(int order\_id,int dish\_id,int dish\_num,int dish\_state,String note);  更新订单内菜品 |
| boolean UpdateOrder(int order\_id,int order\_state,float order\_discount,String note);  更新订单 |

##### 1.4 UserService接口设计

详细描述：

|  |
| --- |
| boolean login(String user\_name,String user\_pwd);  web端用户登录 |
| boolean register(String user\_name,String user\_pwd,String user\_phone,String user\_birthday,String user\_sex,String user\_note);  web端用户注册 |
| User login(String user\_name,String user\_pwd);  ios端用户登录 |
| boolean logout(String User\_id);  用户退出登录 |

##### 1.5 DataService接口设计

|  |
| --- |
| List<Order> getAllOrder();获取全部数据 |
| List<Order> getOrderBetween(String begintime, String endtime);获取区间数据 |
| List<Order> getTodayOrder(String date);  获取今日数据 |
| int getstateCount(int state);  获取数量 |

##### 1.6 PowerService接口设计

|  |
| --- |
| List<Power> getPower(); |
| int setPowerById(int id,String power); |

##### 1.7 业务接口实现类汇总

|  |  |
| --- | --- |
| **接口** | **实现类** |
| DishService | DishServiceImpl |
| OrderService | OrderServiceImpl |
| UserService | UserServiceImpl |

### 四、系统管理

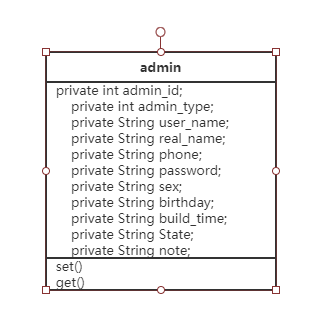
#### 1、员工管理

##### 1.1、功能汇总

管理系统中的所有员工的信息。

|  |  |
| --- | --- |
| **功能** | **描述** |
| 新增 | 新增用户，新增加一个员工 |
| 修改 | 修改用户信息 |
| 删除 | 删除用户信息 |
| 查询 | 根据现有查询条件，查询用户信息 |

##### 1.2、类图设计



##### 1.3、时序图

##### 图片

##### 1.4、类的详细设计描述

##### 1.4.1、AdminController描述

|  |
| --- |
| public List<AdminBrief> getEmployers()  获取员工信息 |
| public List<AdminBrief> getAdminBriefByPage(@RequestParam(value = "pageNO")int pageNo,@RequestParam(value = "pageSize")int pageSize)  public List<Admin> getAdmin(@RequestParam(value = "admin\_id")int id)分页查询 |
| public int deleteAdmin(@RequestParam(value = "admin\_id") int id)删除员工信息 |
| public int newadmin(@RequestBody Admin admin)  新建员工信息 |
| public int updateadmin(@RequestBody Admin admin)  更新员工信息 |
| public int getAdminCount()  获取员工数量 |
| public String adminLoginByUserName(@RequestParam("username")String username,@RequestParam("password")String password)  管理员登录 |
| public String adminLoginByPhone(@RequestParam("phone")String phone,@RequestParam("password")String password)  管理员登陆 |

##### 1.4.2 DataController描述

|  |
| --- |
| public List<Order> getAllOrder() |
| public List<Order> getOrderBetween(@RequestParam("begintime")String begintime,@RequestParam("endtime")String endtime) |
| public List<Order> getTodayOrder() |

1.3.3PowerController

|  |
| --- |
| public List<Power> getEmployers() |
| public int setPowerById(@RequestParam("id")int id,@RequestParam("power")String power) |

##### 1.5、功能实现

管理员登录、信息增删查改、查询经营数据、查询设置权限

#### 2、用户登录注册

##### 2.1、功能汇总

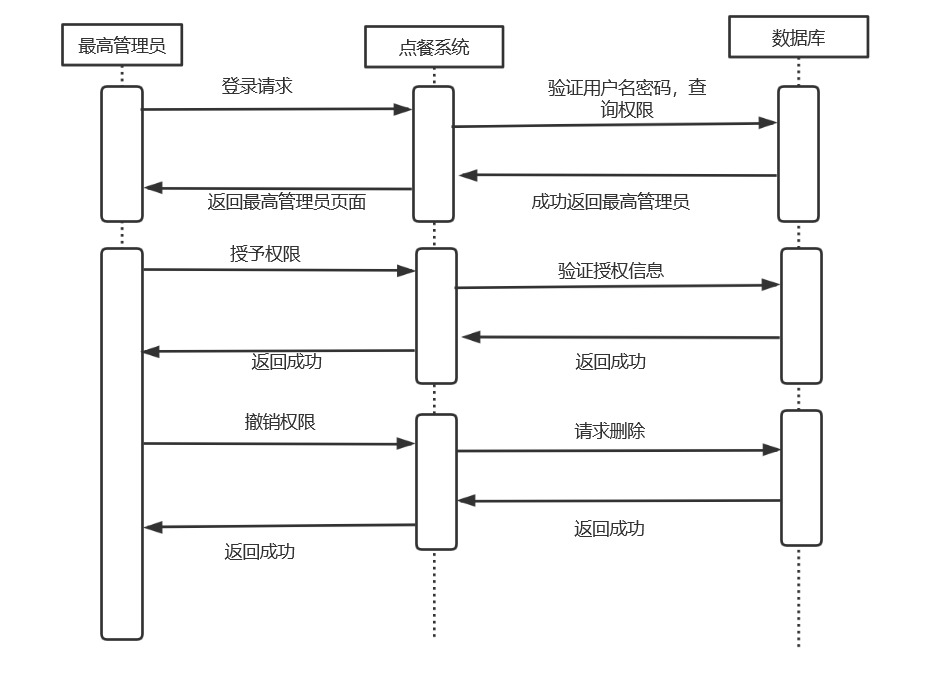
供工作人员登录系统使用。

|  |  |
| --- | --- |
| **功能** | **描述** |
| 用户登录 | 用户登录 |
| 管理员登陆 | 管理员登陆 |
| 用户注册 | 用户注册 |
| 退出登录 | 用户退出登录 |

##### 2.2、类图设计



##### 2.3、时序图



##### 2.4、类的详细设计描述

##### 2.4.1、UserController描述

UserController用于前端提供请求接口，具体包含

@PostMapping(value="/user\_login")

public boolean userlogin(@RequestParam String username,@RequestParam String userpwd){return userService.login(username,userpwd);}

@GetMapping(value="/user\_login\_app")

public User userlogin\_app(@RequestParam String username, @RequestParam String userpwd){return userService.login\_app(username,userpwd);}

@PostMapping(value="/Admin\_login")

public boolean Adminlogin(@RequestParam String Admin\_name,@RequestParam String Admin\_pwd){return userService.login(Admin\_name,Admin\_pwd);}

@PostMapping(value="/register")

public boolean userregieter(@RequestParam String username,

@RequestParam String userpwd,

@RequestParam String phone,

@RequestParam String birthday,

@RequestParam String sex,

@RequestParam String note){return userService.register(username,userpwd,phone,birthday,sex,note);}

@GetMapping(value="logout")

public boolean logout(@RequestParam int UserID){return userService.logout(UserID);

##### 2.5、功能实现

UserController提供了用户注册登陆注销等和连接相关的接口，控制用户的登陆状态。  
 必须在该位置建立连接后，才可使用系统提供的其他功能。

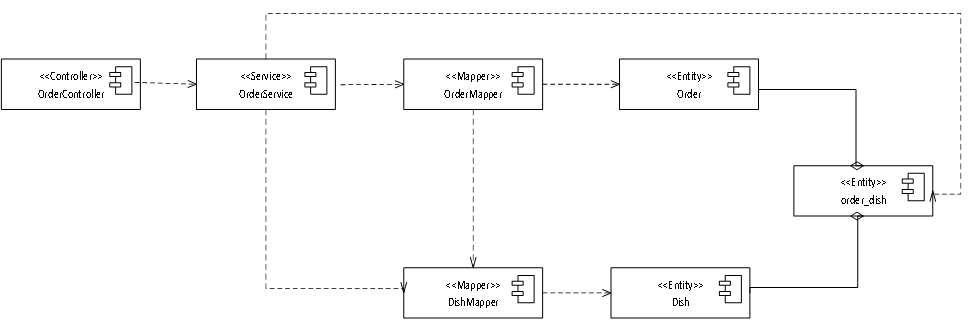
#### 3、订单管理

##### 3.1 功能汇总

|  |  |
| --- | --- |
| **功能** | **描述** |
| 取消订单 | 取消订单，将订单状态改为已取消 |
| 获取订单列表 | 获取所有待处理的订单的列表 |
| 查看菜品 | 查看订单中的菜品 |
| 结算 | 结账，生成结账清单 |
| 修改菜品 | 修改菜品的状态信息，确认烹饪完成 |
| 取消菜品 | 取消订单下指定的的菜品 |

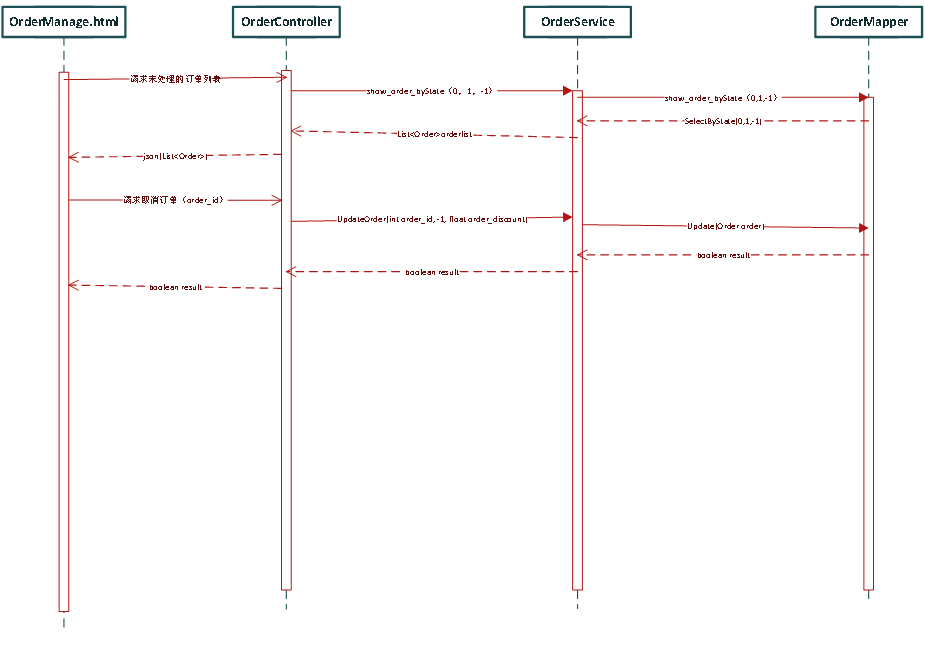
##### 3.2 类图设计

##### 3.2.1 类关系结构图

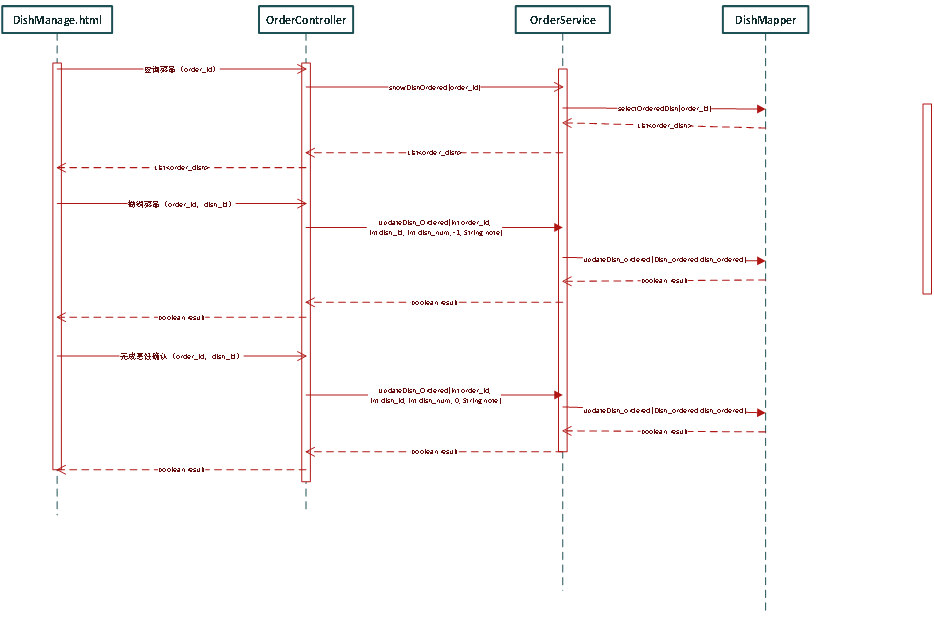


##### 3.2.2 时序图

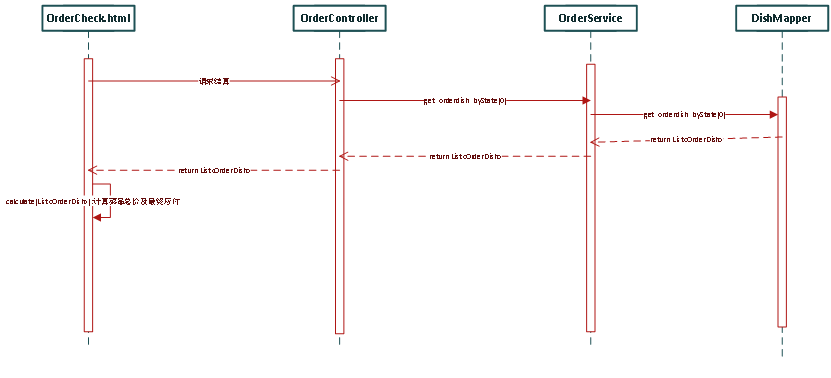
##### 3.2.2.1 订单管理时序图设计



##### 3.2.2.2 订单菜品管理时序图设计：

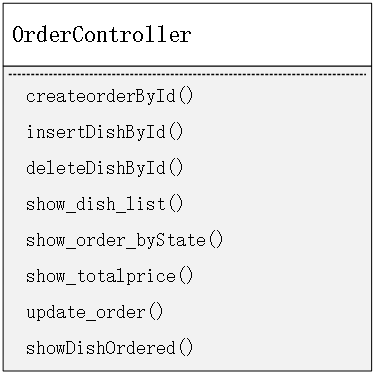


##### 3.2.2.3 订单结算时序图设计



##### 3.3 类的详细设计描述

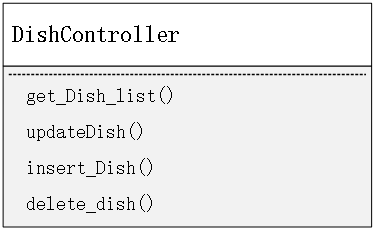
##### 3.3.1 OrderController类的详细设计



描述：

|  |
| --- |
| **OrderController：用来处理前端订单相关的请求，包括菜品订单列表的生成，订单的撤销，订单结算，菜品的撤销，菜品烹饪完成状态的修改** |
| public int createorderById(@RequestParam int user\_id, @RequestParam int table\_id)：创建订单，填充订单初始数据 |
| public boolean insertDishById(@RequestParam int order\_id, @RequestParam int dish\_id,@RequestParam int dish\_num)：新建订单下菜品，点餐时调用 |
| public boolean deleteDishById(@RequestParam int order\_id, @RequestParam int dish\_id)：删除订单下的菜品 |
| public List<Dish> show\_dish\_list(@RequestParam int page\_num,@RequestParam int page\_size)：显示所有订单 |
| public List<Order> show\_order\_byState(@RequestParam int state,@RequestParam int page\_num,@RequestParam int page\_size)：分页查询指定状态的订单 |
| public boolean update\_dish\_ordered(@RequestParam int order\_id,                                         @RequestParam int dish\_id,                                         @RequestParam int dish\_num,                                         @RequestParam int dish\_state,                                         @RequestParam String note)：更新订单下的菜品信息 |
| public boolean update\_order(@RequestParam int order\_id,                                         @RequestParam int order\_state,                                         @RequestParam float order\_discount                                         )：更新订单信息 |

##### 3.3.2 DishController设计：

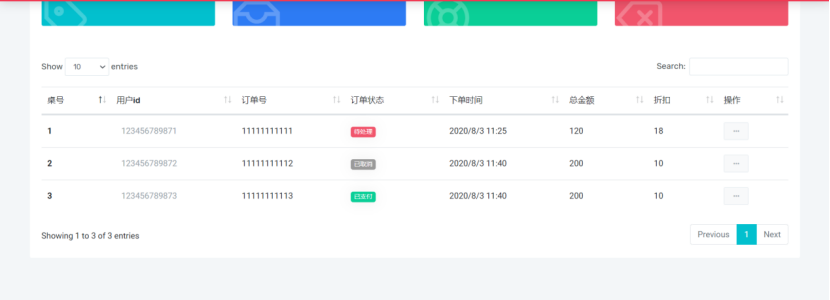


详细描述：

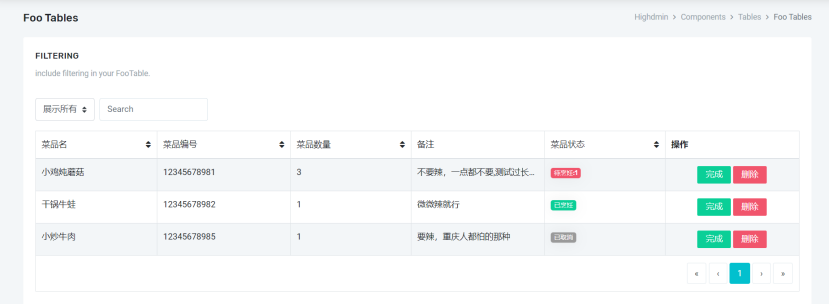
|  |
| --- |
| **DishController：用于处理菜品信息** |
| public List<Dish> get\_Dish\_list(@RequestParam int page\_num,@RequestParam int page\_size)：得到所有菜品的基本信息 |
| public boolean insert\_Dish(@RequestParam String dish\_name,                                 @RequestParam float dish\_price,                                 @RequestParam float discount,                                 @RequestParam String discription,                                 @RequestParam String type,                                 @RequestParam int quantity,                                 @RequestParam int state,                                 @RequestParam String picture)：插入新的菜品 |
| public boolean updateDish(@RequestParam int dish\_id,                                @RequestParam String dish\_name,                                @RequestParam float dish\_price,                                @RequestParam float discount,                                @RequestParam String discription,                                @RequestParam String type,                                @RequestParam int quantity,                                @RequestParam int state,                                @RequestParam String picture      )：更新菜品基本信息 |
| public boolean delete\_dish(@RequestParam int dish\_id)：根据菜品id删除菜品 |

##### 3.3.3 页面描述

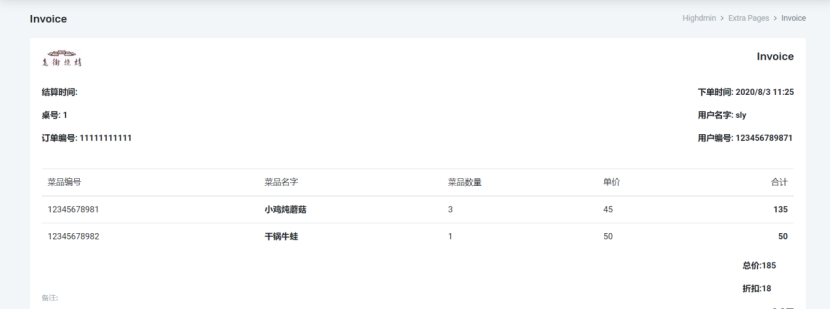
OrderManage.html:OrderManage.html是前台顶单管理的首页页面，是待处理的订单的列表的形式，具体见下图：



DishManage.html是订单详情页，用来处理订单下的菜品相关的操作，具体设计如下图：



OrderCheck.html是订单结算页，用来处理订单结算相关操作，具体设计如下图：



##### 3.4 功能实现

Ordermanage.html为前台提供所有待处理订单的订单列表，其中包含这些订单的id，对应的桌号，用户id，订单当前状态，下单时间，总金额，折扣，并为每个订单数据项设置了操作，包含编辑菜单，结算，撤销订单三种操作，编辑菜单会进入订单详情页DishManage.html,内含订单的具体所点的菜的列表，结算会进入OrderCheck页面

DishManage.html是前台点击编辑订单后进入的页面，包含订单下的菜品列表，此列表包含菜品id，菜品数量，菜品名称，备注，菜品当前状态，以及对应的删除与完成操作，点击删除将对应的菜品状态改为已取消，点击完成将对应的菜品状态改为已完成。

OrderCheck.html：OrderCheck页面是订单页下点击结算跳转到的页面，内含订单的信息和订单下菜品的简略信息，并通过获取到的订单菜品列表计算总金额，根据订单的dishcount折扣，计算应付金额，提供打印操作，可打印小票。

#### 4、经营数据流水查询

##### 4.1、功能汇总

最高管理员查看经营流水，了解餐厅营业情况

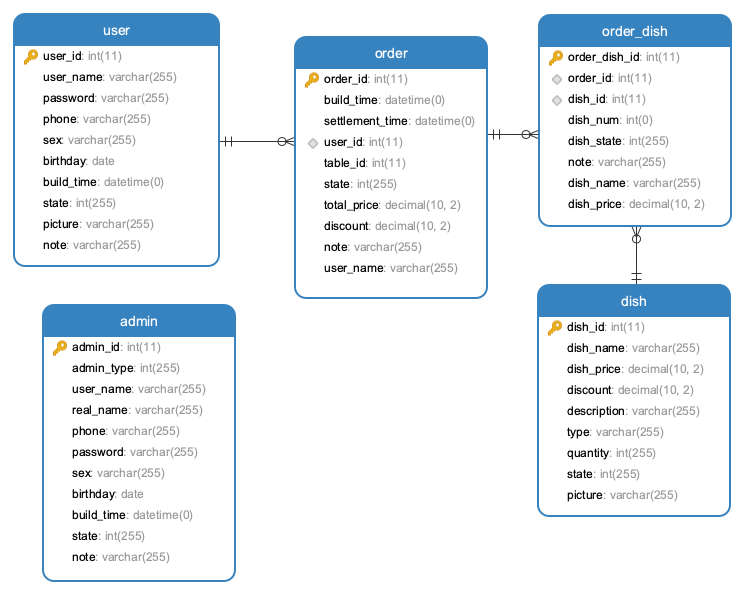
|  |  |
| --- | --- |
| **功能** | **描述** |
| 按年查询订单 | 按年份查询订单，展示每一年的销售总金额情况 |
| 按月查询订单 | 按照某年某月对某一月的订单数，总金额等流水进行查询 |
| 按日查询订单 | 按照某年某月某日对某一天的订单进行查询，展示那一天所有的顶单情况 |

##### 4.2 类图设计

|  |
| --- |
| **DataController** |
| public List<Order> getAllOrder() |
| public List<Order> getYearOrder() |
| public List<Order> getOrderBetween(@RequestParam("begintime")String begintime,@RequestParam("endtime")String endtime) |
| public List<Order> getTodayOrder() |
| public List<Order> getMonthOrder() |

第六部分 数据库设计

### 一、数据库整体结构图



### 二、数据表清单

|  |  |  |
| --- | --- | --- |
| **序号** | **表名** | **注释** |
| 1 | user | 用户表 |
| 2 | order | 订单数据表 |
| 3 | order\_dish | 订单菜品数据表 |
| 4 | dish | 菜品数据表 |
| 5 | admin | 管理员表 |

#### 1、user表结构

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **英文名** | **中文名** | **数据类型** |
| 1 | user\_id | 顾客编号 | Int |
| 2 | user\_name | 用户名 | String |
| 3 | password | 密码 | String |
| 4 | phone | 手机号 | String |
| 5 | sex | 性别 | String |
| 6 | birthday | 生日 | DATE |
| 7 | build\_time | 注册时间 | DATETIME |
| 8 | state | 激活状态 | Int |
| 9 | picture | 用户头像 | String |
| 10 | note | 备注 | String |

#### 2、order表结构

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **英文名** | **中文名** | **数据类型** |
| 1 | order\_id | 订单号 | Int |
| 2 | build\_time | 生成时间 | DATETIME |
| 3 | settlement\_time | 结账时间 | DATETIME |
| 4 | user\_id | 点单顾客编号 | Int |
| 5 | table\_id | 桌号 | Int |
| 6 | state | 订单状态 | Int |
| 7 | total\_price | 订单总价 | decimal |
| 8 | discount | 折扣 | decimal |
| 9 | note | 备注 | String |
| 10 | user\_name | 用户名 | String |

#### 3、order\_dish表结构

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **英文名** | **中文名** | **数据类型** |
| 1 | order\_dish\_id | 订单菜品号 | Int |
| 2 | order\_id | 订单号 | Int |
| 3 | dish\_id | 菜品号 | Int |
| 4 | dish\_num | 点菜数量 | Int |
| 5 | dish\_state | 菜品状态 | Int |
| 6 | note | 备注 | String |
| 7 | dish\_name | 菜品名 | String |
| 8 | dish\_price | 菜品价格 | decimal |

#### 4、dish表结构

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **英文名** | **中文名** | **数据类型** |
| 1 | dish\_id | 菜品号 | Int |
| 2 | dish\_name | 菜品名 | String |
| 3 | dish\_price | 菜品价格 | Double |
| 4 | discount | 菜品优惠 | Double |
| 5 | description | 菜品描述 | String |
| 6 | type | 菜品类型 | String |
| 7 | quantity | 菜品剩余数量 | Int |
| 8 | state | 菜品可选状态 | Int |
| 9 | picture | 菜品图片 | String |

#### 5、admin表结构

|  |  |  |  |
| --- | --- | --- | --- |
| **编号** | **英文名** | **中文名** | **数据类型** |
| 1 | admin\_id | 管理员编号 | Int |
| 2 | admin\_type | 类型 | Int |
| 3 | user\_name | 用户名 | String |
| 4 | real\_name | 真实姓名 | String |
| 5 | phone | 手机号 | String |
| 6 | password | 密码 | String |
| 7 | sex | 年龄 | String |
| 8 | birthday | 生日 | DATE |
| 9 | build\_time | 注册时间 | DATETIME |
| 10 | state | 激活状态 | Int |
| 11 | note | 备注 | String |

### 三、外键清单

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **外键名称** | **父表** | **父键列** | **子表** | **外键列** | **关系** | **说明** |
| user\_id | user | user\_id | order | user\_id | 0..\* | 一个用户可以有多个订单 |
| order\_id | order | order\_id | order\_dish | order\_id | 0..\* | 一个订单可以有多个菜品 |
| dish\_id | dish | dish\_id | order\_dish | dish\_id | 0..\* | 一个菜品可以与多个订单中的菜品相关 |

第七部分 补充设计和说明

### 一、编译运行环境设计

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.3.2.RELEASE</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.gaoke</groupId>

<artifactId>demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>Order-system</name>

<description>Demo for gaoke internship</description>

<properties>

<java.version>11</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

<exclusions>

<exclusion>

<groupId>org.junit.vintage</groupId>

<artifactId>junit-vintage-engine</artifactId>

</exclusion>

</exclusions>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!-- mysql驱动包 -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.0.13</version>

</dependency>

<dependency>

<groupId>org.xmlunit</groupId>

<artifactId>xmlunit-core</artifactId>

</dependency>

<dependency>

<groupId>org.mybatis.spring.boot</groupId>

<artifactId>mybatis-spring-boot-starter</artifactId>

<version>1.2.0</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

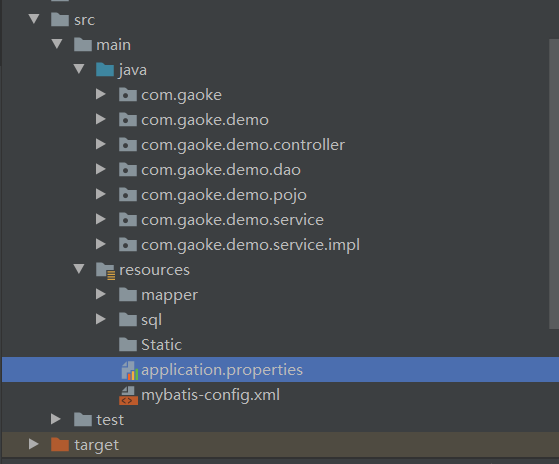
</plugins>

</build>

</project>

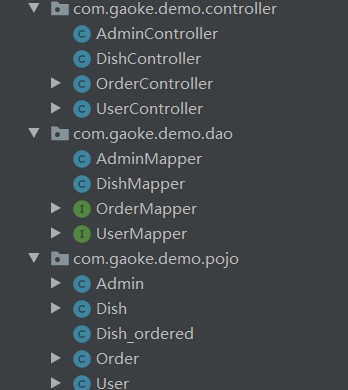
### 二、包路径与WEB目录结构设计

#### 1、包路径设计

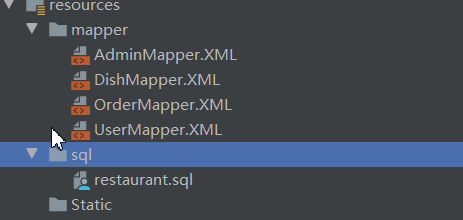


采用springboot典型三层结构，controller+dao+service，在pojo中封装用到的实体类，在controller中处理与前端的交互，在service 中封装接口，在serviceimpl中编写接口具体操作，在dao中处理与数据库的交互

部分结构如下



此外，由于使用了mybatis 在resource目录的mapper下存放相应的xml文件



#### 2、WEB目录结构设计

