

# 目 录

一. 系统需求分析 .....	1
1. 系统描述 .....	1
2. 数据存储需求.....	1
3. 系统常做的查询与更新.....	2
4. 应用程序功能.....	3
二. 数据库概念设计 .....	6
1. 确定实体和属性.....	6
2. E-R 图 .....	7
三. 数据库逻辑结构设计.....	8
1. 关系模式设计.....	8
2. 基本表设计 .....	9
四. 数据库物理设计和实施.....	15
1. 数据库的创建.....	15
2. 创建基本表 .....	15
3. 存储过程/函数设计.....	20
五. 应用程序设计 .....	22
1. 开发及运行环境介绍.....	22
2. 流程图 .....	23
3. 部分功能设计.....	24
4. 主要界面 .....	28

## 一. 系统需求分析

### 1. 系统描述

随着网络技术的完善，信息时效性的发展不断为商业模式带来变革，餐饮行业便是其中的受益者之一。相比于传统的线下销售模式，外卖系统使得消费者不再拘于距离的限制，完善而遍历的产业链刺激了顾客的消费欲望，提高了商家的利润收益，并为市场提供了骑手这一新兴职业。

"饱了么"正是应用于该行业的一款信息管理系统，系统的三种用户分为顾客，商家，骑手。而联系这三种实体的数据集便是订单，订单由顾客向用户直接发起，商家接单后由骑手介入，从而完成一次订单的创建。

此外三种实体之间又有各自的联系集，例如顾客可以为骑手和商家进行打分和评论。商家可以与骑手之间签署契约，只有与商家签约的骑手才能为该商家进行订单的配送。

对于每种实体本身也有一系列信息可以维护，除了最基本的密码修改、邮箱电话修改、更新头像外，还可以查看自己的历史订单，管理自己的菜品。每种实体也设计了等级制度，用户等级可分为普通用户和 VIP 用户，骑手等级分普通骑手和高级骑手，商家分为美食类，医药类，果蔬类和超市类。

外卖销售系统消除了部分空间的限制，给商业流通带来了变革，也给消费者带来了便捷。

### 2. 数据存储需求

用户信息涉及用户名、密码、手机号、邮箱和用户类型，无论何种用户类型用户名都唯一，即某个用户名若被消费者注册，则商家和骑手均不可再使用该用户名。邮箱和手机号同样不可重复，因为该字段信息用于找回密码时的验证。

顾客信息包含用户名、顾客等级、账户余额、消费积分、用户头像和用户地址，用户可以对余额进行充值，在下单时用户只能使用余额进行支付。积分会根据用户下单的金额进行奖励，积分用于兑现余额。用户等级分为普通和会员，会

员可以兑换积分时的倍率更高。

骑手信息包含用户名,注册日期,骑手等级,总收入,完成的订单数和评分。

商家信息包含用户名,注册日期,销售类型,店铺名称,总订单数,评分,店铺地址,营业时间与描述信息。顾客可以通过店铺名称和描述信息进行对店家的搜索。

订单信息包含了订单号、顾客、骑手与商家的用户名、下单时间、用户评价,评价内容包含对商家、骑手的打分和内容。订单的流程为:顾客下单—>商家接单—>与商家签约的骑手接单。

商品信息包含了商品 id, 价格, 销售量, 图片路径, 剩余量, 商品描述以及属于哪个店家;订单包含的商品项需要通过联系集实现。

### 3. 系统常做的查询与更新

经常做的查询

- 根据用户名查询用户全部信息
- 根据用户名查询所以与该用户相关的订单
- 根据关键词和销售类型查询商家
- 根据评价号查询评价内容、评价撰写者和评价对象
- 查询当前使用的最大评价号、商品号
- 通过订单号查询所有订单包含的所有商品
- 通过商家用户名查询其店下的所有商品
- 查询某个用户名/邮件/电话号码是否已经存在
- 查询某个骑手所有签约商家的可接订单
- 查询某骑手/商家的所有评价内容
- 查询某商家签约的全部骑手/某骑手签约的全部商家

关于更新

- 用户注册后, 对应的用户类型需要更新
- 用户更改头像时, 图片路径需要更新
- 用户更改邮箱、电话、密码、地址时, 相关字段需要更新
- 用户成功支付时, 钱包余额和积分需要更新, 订单状态变为 0

- 商家成功接单时，订单状态更新为 1，接单时间更新
- 骑手成功配送时，订单状态更新为 2，配送时间更新
- 骑手成功送达时，订单状态更新 3，订单完成时间更新，顾客、骑手、商家订单数更新，商家和骑手收入更新
- 用户等级变更时，相关信息需要更新
- 顾客评价订单时，商家和骑手的评分需要更新
- 商家管理菜品时，商品名称、价格、库存需要更新
- 商家向骑手发送签约申请时，申请列表需要更新
- 商家与骑手双方达成协议时，契约列表需要更新

## 4. 应用程序功能

### 4.0 登录/注册界面

#### 1) 登录中心:

- 正常登录: 通过输入账户和密码判断用户身份，若为顾客则跳转到 4.1 顾客版本，若为商家跳转到 4.2 商家版本，若为骑手跳转到 4.3 商家版本
- 验证码系统: 每一次登录/点击验证码都会刷新验证码，防止恶意登录
- 忘记密码: 通过邮箱验证的方式找回密码
- 关于我们: 通过点击界面最下方的字样获取开源地址、作者信息。

#### 2) 注册中心:

- 用户名不得与已存在的冲突
- 密码不得过短，两次输入密码需一致
- 邮箱通过正则表达式检验，需要输入正确的邮箱地址
- 电话号码需要用数字（区号）的正确格式，也会通过正则表达式检验

#### 3) 完成注册

- 选定注册身份:顾客/商家/骑手
- 选定地址、工作时间、头像等选项
- 发送邮箱验证码完成注册: 您可以使用@test.com 的域名作为邮箱后缀，对于这类邮箱将通过正则表达式识别出，然后输入任意验证码均可通过注册

#### 4.1 顾客

##### 1) 主菜单:

- a. 分类板块: 根据不同的商家类型进行筛选, 主要分为美食外卖、超市便利、健民药房和鲜果时蔬四大类。
- b. 搜索系统: 根据店名、店铺关键字进行搜索筛选。
- c. 浏览系统: 根据 a,b 筛选出的全部商店将以分页的形式供用户浏览, 每页设置八个店铺的信息, 包括店名、距离、评分等。
- d. 点餐系统: 查询出店家旗下的全部产品, 全部罗列在界面中, 并设置购物车来统计用户的购买信息。用户结算后将会进行订单的创建。

##### 2) 订单中心:

- a. 浏览订单: 以分页的形式呈现该用户的全部订单, 每页一个订单的信息, 包括订单号、订单状态、下单商家、配送骑手、订单金额以及所有购买的商品, 都将罗列出。
- b. 评价系统: 对于已经送达的商品, 用户可以进行打分和内容的评价, 提交后将会更新到商家和骑手的个人信息当中。

##### 3) 个人中心:

- a. 用户注销: 切换用户
- b. 修改信息: 修改密码、邮箱、头像、电话和地址信息。
- c. 会员充值: 提升普通会员到 VIP 会员
- d. 积分兑换: 使用积分兑换现金, VIP 兑换倍率双倍
- e. 余额充值: 增加用户的余额数值

#### 4.2 商家

##### 1) 主菜单:

- a. 接单中心: 查询所有顾客下单自己仍为处理的订单
- b. 菜品管理: 对店铺的商品进行增删, 也可以更改已有菜品的价格、库存、名称。
- c. 我的评价: 查看已经收到的全部评价内容和评分

- d. 签约骑手: 可以通过输入骑手 ID 的方式发出签约申请, 也可以处理收到的骑手申请。
- 2) 订单中心: 以分页的形式呈现该用户的全部订单, 每页一个订单的信息, 包括订单号、订单状态、下单商家、配送骑手、订单金额以及所有购买的商品, 都将罗列出。
- 3) 个人中心:
  - a. 用户注销: 切换用户
  - b. 修改信息: 修改店名、邮箱、头像、电话和地址信息。
  - c. 收入查询: 查看店铺的收入
  - d. 评分查询: 查看店铺的平均得分

### 4.3 骑手

- 1) 主菜单:
  - a. 接单中心: 查询所有自己已经签约的店铺的可接订单
  - b. 骑手排行: 以送单数为关键字排序, 查看当前所有骑手的排行
  - c. 我的评价: 查看已经收到的全部评价内容和评分
  - d. 签约商家: 可以通过输入商家 ID 的方式发出签约申请, 也可以处理收到的商家申请。
- 2) 订单中心: 以分页的形式呈现该用户的全部订单, 每页一个订单的信息, 包括订单号、订单状态、下单商家、配送骑手、订单金额以及所有购买的商品, 都将罗列出。
- 3) 个人中心:
  - a. 用户注销: 切换用户
  - b. 修改信息: 修改密码、邮箱、头像、电话和地址信息。
  - c. 收入查询: 查看骑手当前的收入
  - d. 评分查询: 查看店铺的平均得分
  - e. 等级提升: 提升当前骑手的配送等级

## 二. 数据库概念设计

### 1. 确定实体和属性

分析外卖信息管理系统的系统需求，将系统中设计的人、物进行抽象，得到了系统的实体如下：

- 1) 用户信息实体集。属性包括：用户名、密码、电话、邮箱、用户类型。
- 2) 顾客实体集。属性包括：用户名、用户等级、账户余额、消费积分、头像路径、总下单次数、注册日期、地址。
- 3) 骑手实体集。属性包括：用户名、注册日期、头像路径、骑手等级、订单数、评价次数、评价总分、地址、收入。
- 4) 商家实体集。属性包括：用户名、注册日期、销售类型、订单总数、评价次数、评价总分、地址、营业时间、头像路径、描述信息、店名、头像路径。
- 5) 订单实体集。属性包括：订单号、下单时间、接单时间、配送时间、订单状态、送达时间、商家编号、顾客编号、骑手编号、评价编号。
- 6) 评价实体集。属性包括：评价号、评价内容、评价分数。
- 7) 商品实体集。属性包括：商品号、价格、销量、剩余量、商品名、所属店家、图片路径。

## 2. E-R 图

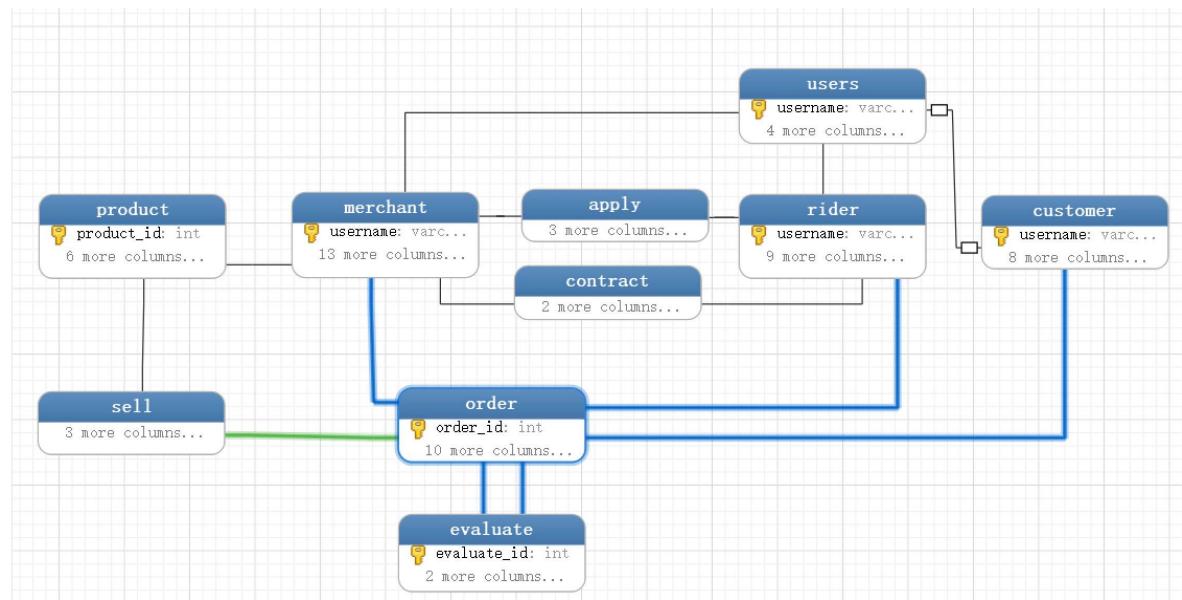


图 2-1 E-R 图

### 三. 数据库逻辑结构设计

#### 1. 关系模式设计

根据概念结构设计得到的 E-R 图和转换规则, 得到如下关系模式

- a. 用户信息表(用户名, 登录密码, 用户类型, 电话号码, 邮箱)
- b. 顾客信息表(用户名[引用 a 的用户名], 用户等级, 账户余额, 消费积分, 头像路径, 总订单数, 注册日期, 经度, 纬度, )
- c. 骑手信息表(用户名[引用 a 的用户名], 注册日期, 骑手等级, 头像路径, 总订单数, 评价总分, 评价次数, 经度, 纬度, 收入)
- d. 商家信息表(用户名[引用 a 的用户名], 注册日期, 销售类型, 总订单数, 评价总分, 评价次数, 经度, 纬度, 营业开始时间, 营业结束时间, 描述信息, 头像路径, 收入, 店铺名称)
- e. 评价信息表(评价号, 评价分数, 评论内容)
- f. 订单信息表(订单号, 下单时间, 接单时间, 配送时间, 送达时间, 顾客编号[引用 b 的用户名], 骑手编号[引用 c 的用户名], 商家编号[引用 d 的用户名], 对商家评价[引用 e 的评价号], 对骑手评价[引用 e 的评价号], 订单状态)
- g. 商品信息表(商品号, 商品价格, 销售量, 图片路径, 商品名称, 描述信息, 归属商家[引用 d 的用户名], 剩余量)
- h. 销售信息表(订单号[引用 f 的订单号], 商品号[引用 g 的商品号], 销售量)
- i. 申请信息表(骑手用户名[引用 c 的用户名], 商家用户名[引用 d 的用户名], 申请方向)
- j. 契约信息表(骑手用户名[引用 c 的用户名], 商家用户名[引用 d 的用户名])

## 2. 基本表设计

表 3-1：用户信息表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
username	char(50)	否			主键	用户名
password	varchar(20)	否				登录密码
user_type	enum	否	"customer" or "rider" or "merchant"	customer		用户类型
telephone	varchar(20)	否				电话号码
email	varchar(50)	否				邮箱

表 3-2：顾客信息表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
username	char(50)	否			主键; 外键:引用表 3-1 的 username	用户名
user_level	enum	否	"common" or "VIP"	"common"		用户等级
balance	numeric(12,2)	否		0		账户余额
points	int	否		0		消费积分
portrait	varchar(200)	是				头像路径
sum_order	int	否		0		总订单数
register_date	date	否				注册日期

longitude	int	否				经度
latitude	int	否				纬度

表 3-3: 骑手信息表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
username	char(50)	否			主键; 外键:引用表 3-1 的 username	用户名
register_date	date	否				注册日期
rider_type	enum	否	"common" or "medium" or "expert"	"common"		骑手等级
sum_order	int	否		0		订单总数
evaluate_sum	int	否		0		评价总分
evaluate_cnt	int	否		0		评价次数
portrait	varchar(200)	是				头像路径
longitude	int	否				经度
latitude	int	否				纬度
income	decimal(12,4)	否		0		收入

表 3-4: 商家信息表的设计

属性名	数据类型	是否可	列约束	默认值	键	解释
-----	------	-----	-----	-----	---	----

		空				
username	char(50)	否			主键; 外键:引用 表 3-1 的 username	用户名
register_date	date	否				注册日期
sale_type	enum	否		"restaurant" or "supermarket" or "drugstore"	"restaurant"	商家类型
sum_order	int	否		0		订单总数
evaluate_sum	int	否		0		评价总分
evaluate_cnt	int	否		0		评价次数
longitude	int	否				经度
latitude	int	否				纬度
begin_time	time	否				营业时间
end_time	time	否				关门时间
description	varchar(200)	是				描述信息
portrait	varchar(200)	是				头像路径
name	varchar(200)	否				店铺名称
income	decimal(12,4)	否		0		收入

表 3-5: 评价信息表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
evaluate_id	int	否			主键	评价号
score	int	否	{1,2,3,4,5}			评价分数
comment	varchar(200)	是				评价内容

表 3-6：订单信息表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
order_id	int	否			主键	订单号
order_time	datetime	否				下单时间
merchant_accept_time	datetime	是				接单时间
rider_accept_time	datetime	是				配送时间
finish_time	datetime	是				送达时间
customer_name	varchar(50)	否			外键:引用表 3-2 username	顾客编号
rider_name	varchar(50)	否			外键:引用表 3-3 username	骑手编号
merchant_name	varchar(50)	否			外键:引用表 3-4 username	商家编号
merchant_evaluate_id	int	是			外键:引用表 3-5 evaluate_id	顾客对商家的评价
rider_evaluate_id	int	是			外键:引用表 3-5 evaluate_id	顾客对骑手的评价
state	int	否	{0,1,2,3}			订单状态

表 3-7: 商品信息表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
product_id	int	否			主键	商品编号
price	decimal(12,2)	否				商品价格
sales	int	否				销售量
picture_path	varchar(200)	否				图片路径
name	varchar(200)	否				商品名称
master	varchar(50)	否			外键:引用表 3-4 username	归属商家
left	int	否				剩余量

表 3-8: 销售信息表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
order_id	int	否			外键:引用表 3-6 order_id	订单编号
product_id	int	否			外键:引用表 3-7 product_id	产品编号

表 3-9: 申请信息表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释

rider_name	varchar(50)	否			外键:引用表 3-3 username	骑手编号
merchant_name	varchar(50)	否			外键:引用表 3-4 username	商家编号
direction	int	否	{0,1}			申请方向

表 3-10: 契约信息表的设计

属性名	数据类型	是否可空	列约束	默认值	键	解释
rider_name	varchar(50)	否			外键:引用表 3-3 username	骑手编号
merchant_name	varchar(50)	否			外键:引用表 3-4 username	商家编号

## 四. 数据库物理设计和实施

### 1. 数据库的创建

使用华为 GaussDB(for MySQL)云数据库建立"饱了没"信息管理系统数据库。

### 2. 创建基本表

```
SET FOREIGN_KEY_CHECKS=0;

-- Table structure for apply
-- Table structure for contract

DROP TABLE IF EXISTS `apply`;
CREATE TABLE `apply` (
  `rider_name` varchar(50) DEFAULT NULL,
  `merchant_name` varchar(50) DEFAULT NULL,
  `direction` int(11) DEFAULT NULL,
  KEY `rider_name` (`rider_name`),
  KEY `merchant_name` (`merchant_name`),
  CONSTRAINT `apply_ibfk_1` FOREIGN KEY (`rider_name`) REFERENCES `rider` (`username`),
  CONSTRAINT `apply_ibfk_2` FOREIGN KEY (`merchant_name`) REFERENCES `merchant` (`username`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

DROP TABLE IF EXISTS `contract`;
CREATE TABLE `contract` (
  `rider_name` varchar(50) DEFAULT NULL,
  `merchant_name` varchar(50) DEFAULT NULL,
  KEY `rider_name` (`rider_name`),
  KEY `merchant_name` (`merchant_name`),
  CONSTRAINT `contract_ibfk_1` FOREIGN KEY (`rider_name`) REFERENCES `rider` (`username`),
  CONSTRAINT `contract_ibfk_2` FOREIGN KEY (`merchant_name`) REFERENCES `merchant` (`username`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

```

-----  

-- Table structure for customer  

-----  

DROP TABLE IF EXISTS `customer`;  

CREATE TABLE `customer` (  

  `username` varchar(50) NOT NULL,  

  `user_level` enum('common','VIP') DEFAULT 'common',  

  `balance` decimal(12,2) DEFAULT '0.00',  

  `points` int(11) DEFAULT '0',  

  `portrait` varchar(200) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai  

_ci DEFAULT './image/default_customer.jpg',  

  `sum_order` int(11) DEFAULT '0',  

  `register_date` date DEFAULT NULL,  

  `longitude` int(11) unsigned DEFAULT '0',  

  `latitude` int(11) unsigned DEFAULT '0',  

  PRIMARY KEY (`username`),  

  CONSTRAINT `customer_ibfk_1` FOREIGN KEY (`username`) REFERENCES `use  

rs` (`username`) ON DELETE CASCADE ON UPDATE CASCADE  

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;  

-----  

-- Table structure for evaluate  

-----  

DROP TABLE IF EXISTS `evaluate`;  

CREATE TABLE `evaluate` (  

  `evaluate_id` int(11) NOT NULL AUTO_INCREMENT,  

  `score` int(11) unsigned NOT NULL,  

  `comment` varchar(200) DEFAULT NULL,  

  PRIMARY KEY (`evaluate_id`),  

  CONSTRAINT `evaluate_chk_1` CHECK (((`score` >= 0) and (`score` <= 5))  

)  

) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb  

4_0900_ai_ci;  

-----  

-- Table structure for merchant  

-----  

DROP TABLE IF EXISTS `merchant`;  

CREATE TABLE `merchant` (  

  `username` varchar(50) NOT NULL,  

  `register_date` date DEFAULT NULL,  

  `sale_type` enum('cate','supermarket','drug','fruit') DEFAULT 'cate',  

  `sum_order` int(11) DEFAULT '0',  

  `evaluate_sum` int(11) unsigned DEFAULT '0',

```

```

`name` varchar(200) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci
DEFAULT NULL,
`longitude` int(11) DEFAULT NULL,
`latitude` int(11) DEFAULT NULL,
`begin_time` time DEFAULT NULL,
`end_time` time DEFAULT NULL,
`portrait` varchar(200) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai
_ci DEFAULT './image/default_merchant.jpg',
`description` varchar(200) DEFAULT NULL,
`evaluate_cnt` int(11) unsigned DEFAULT '0',
`income` decimal(12,4) unsigned DEFAULT NULL,
PRIMARY KEY (`username`),
CONSTRAINT `merchant_ibfk_1` FOREIGN KEY (`username`) REFERENCES `use
rs`(`username`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

-----
-- Table structure for order
-----

DROP TABLE IF EXISTS `order`;
CREATE TABLE `order` (
`order_id` int(11) NOT NULL AUTO_INCREMENT,
`order_time` datetime DEFAULT NULL,
`merchant_accept_time` datetime DEFAULT NULL,
`rider_accept_time` datetime DEFAULT NULL,
`finish_time` datetime DEFAULT NULL,
`state` int(11) unsigned DEFAULT NULL,
`customer_name` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4_090
0_ai_ci DEFAULT NULL,
`rider_name` varchar(50) DEFAULT NULL,
`merchant_name` varchar(50) DEFAULT NULL,
`merchant_evaluate_id` int(11) DEFAULT NULL,
`rider_evaluate_id` int(11) DEFAULT NULL,
PRIMARY KEY (`order_id`),
KEY `customer_name`(`customer_name`),
KEY `rider_name`(`rider_name`),
KEY `merchant_name`(`merchant_name`),
KEY `merchant_evaluate_id`(`merchant_evaluate_id`),
KEY `rider_evaluate_id`(`rider_evaluate_id`),
CONSTRAINT `order_ibfk_1` FOREIGN KEY (`customer_name`) REFERENCES `c
ustomer`(`username`),
CONSTRAINT `order_ibfk_2` FOREIGN KEY (`rider_name`) REFERENCES `ride
r`(`username`),

```

```

    CONSTRAINT `order_ibfk_3` FOREIGN KEY (`merchant_name`) REFERENCES `merchant`(`username`),
    CONSTRAINT `order_ibfk_4` FOREIGN KEY (`merchant_evaluate_id`) REFERENCES `evaluate`(`evaluate_id`),
    CONSTRAINT `order_ibfk_5` FOREIGN KEY (`rider_evaluate_id`) REFERENCES `evaluate`(`evaluate_id`)
) ENGINE=InnoDB AUTO_INCREMENT=18 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

-----
-- Table structure for product
-----

DROP TABLE IF EXISTS `product`;
CREATE TABLE `product` (
    `product_id` int(11) NOT NULL AUTO_INCREMENT,
    `price` decimal(12,2) DEFAULT NULL,
    `sales` int(11) DEFAULT '0',
    `picture_path` varchar(200) DEFAULT NULL,
    `name` varchar(200) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci
    DEFAULT NULL,
    `master` varchar(50) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci
    DEFAULT NULL,
    `left` int(11) DEFAULT NULL,
    PRIMARY KEY (`product_id`),
    KEY `master`(`master`),
    CONSTRAINT `product_ibfk_1` FOREIGN KEY (`master`) REFERENCES `merchant`(`username`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=197 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

-----
-- Table structure for rider
-----

DROP TABLE IF EXISTS `rider`;
CREATE TABLE `rider` (
    `username` varchar(50) NOT NULL,
    `register_date` date DEFAULT NULL,
    `portrait` varchar(200) CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci
    DEFAULT './image/default_rider.jpg',
    `rider_type` enum('common','medium','expert') DEFAULT 'common',
    `sum_order` int(11) DEFAULT '0',
    `evaluate_sum` int(11) unsigned DEFAULT '0',
    `longitude` int(11) unsigned DEFAULT '0',
    `latitude` int(11) unsigned DEFAULT NULL,

```

```

`evaluate_cnt` int(11) unsigned DEFAULT '0',
`income` decimal(12,4) unsigned DEFAULT '0.0000',
PRIMARY KEY (`username`),
CONSTRAINT `rider_ibfk_1` FOREIGN KEY (`username`) REFERENCES `users`(`username`)
ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

-----
-- Table structure for sell
-----

DROP TABLE IF EXISTS `sell`;
CREATE TABLE `sell` (
`order_id` int(11) DEFAULT NULL,
`product_id` int(11) DEFAULT NULL,
`count` int(11) unsigned NOT NULL,
KEY `order_id` (`order_id`),
KEY `product_id` (`product_id`),
CONSTRAINT `sell_ibfk_1` FOREIGN KEY (`order_id`) REFERENCES `order`(`order_id`),
CONSTRAINT `sell_ibfk_2` FOREIGN KEY (`product_id`) REFERENCES `product`(`product_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

-----
-- Table structure for users
-----

DROP TABLE IF EXISTS `users`;
CREATE TABLE `users` (
`username` varchar(50) NOT NULL,
`password` varchar(20) NOT NULL,
`user_type` enum('customer','rider','merchant','admin') DEFAULT 'customer',
`telephone` varchar(20) DEFAULT NULL,
`email` varchar(50) DEFAULT NULL,
PRIMARY KEY (`username`),
UNIQUE KEY `telephone` (`telephone`),
UNIQUE KEY `email` (`email`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

```

### 3. 存储过程/函数设计

```

-- get_biggest_evaluate_id: 搜索 evaluate 表中最大 id 值来设置插入时的新 id 值
-- 没用使用自增是因为在操作的时候需要用该 id 值更新其他属性, 使用自增无法获取定位刚才插入的列, 所以撰写了该函数, 还有一个 get_biggest_order_id 函数同理
-- 

DROP FUNCTION IF EXISTS `get_biggest_evaluate_id`;
DELIMITER ;;
CREATE DEFINER=`DB_USER066`@`%` FUNCTION `get_biggest_evaluate_id`() RE
URNS int(11)
READS SQL DATA
BEGIN
    declare result int;
    declare cnt int;
    select count(*) into cnt from `evaluate`;
    if cnt = 0 then
        return 1;
    else
        select `evaluate_id` into result
            from `evaluate`
            order by `evaluate_id` desc
            limit 1;
        return result+1;
    end if;
    RETURN 0;
END
;;
DELIMITER ;
-- Function structure for get_biggest_order_id
-- 

DROP FUNCTION IF EXISTS `get_biggest_order_id`;
DELIMITER ;;
CREATE DEFINER=`DB_USER066`@`%` FUNCTION `get_biggest_order_id`() RETUR
NS int(11)
READS SQL DATA
BEGIN
    declare result int;
    declare cnt int;
    select count(*) into cnt from `order`;
    if cnt = 0 then
        return 1;
    else

```

```

    select `order_id` into result
    from `order`
    order by `order_id` desc
    limit 1;
    return result+1;
end if;
END
;;
DELIMITER ;
-----
-- sumMoney:计算一个订单的总金额
-----
DROP FUNCTION IF EXISTS `sumMoney`;
DELIMITER ;;
CREATE DEFINER=`DB_USER066`@`%` FUNCTION `sumMoney` (orderId int) RETURN
S int(11)
    READS SQL DATA
begin
    declare result int;
    select sum(`product`.price*`sell`.count) into result
    from `sell` inner join `product`
    on `sell`.product_id = `product`.product_id
    where `sell`.order_id = orderId;
    return result;
end
;;
DELIMITER ;
-----
-- Function structure for getCustPort
-----
DROP FUNCTION IF EXISTS `getCustPort`;
DELIMITER ;;
CREATE DEFINER=`DB_USER066`@`%` FUNCTION `getCustPort` (cname varchar(50
)) RETURNS varchar(200) CHARSET utf8mb4
    READS SQL DATA
begin
    declare result varchar(200);
    select portrait into result
    from customer
    where `username` = cname;
    return result;
end
;;
DELIMITER ;

```

## 五. 应用程序设计

### 1. 开发及运行环境介绍

操作系统: Win10

开发语言: Python3.9

开发工具: PyQt5

运行方法: 在 ./MySQL/目录下使用 python main.py 打开登录界面

需要 pip 安装的包有:py-emails, PyQt5, pymysql

import 部分的代码一共有:

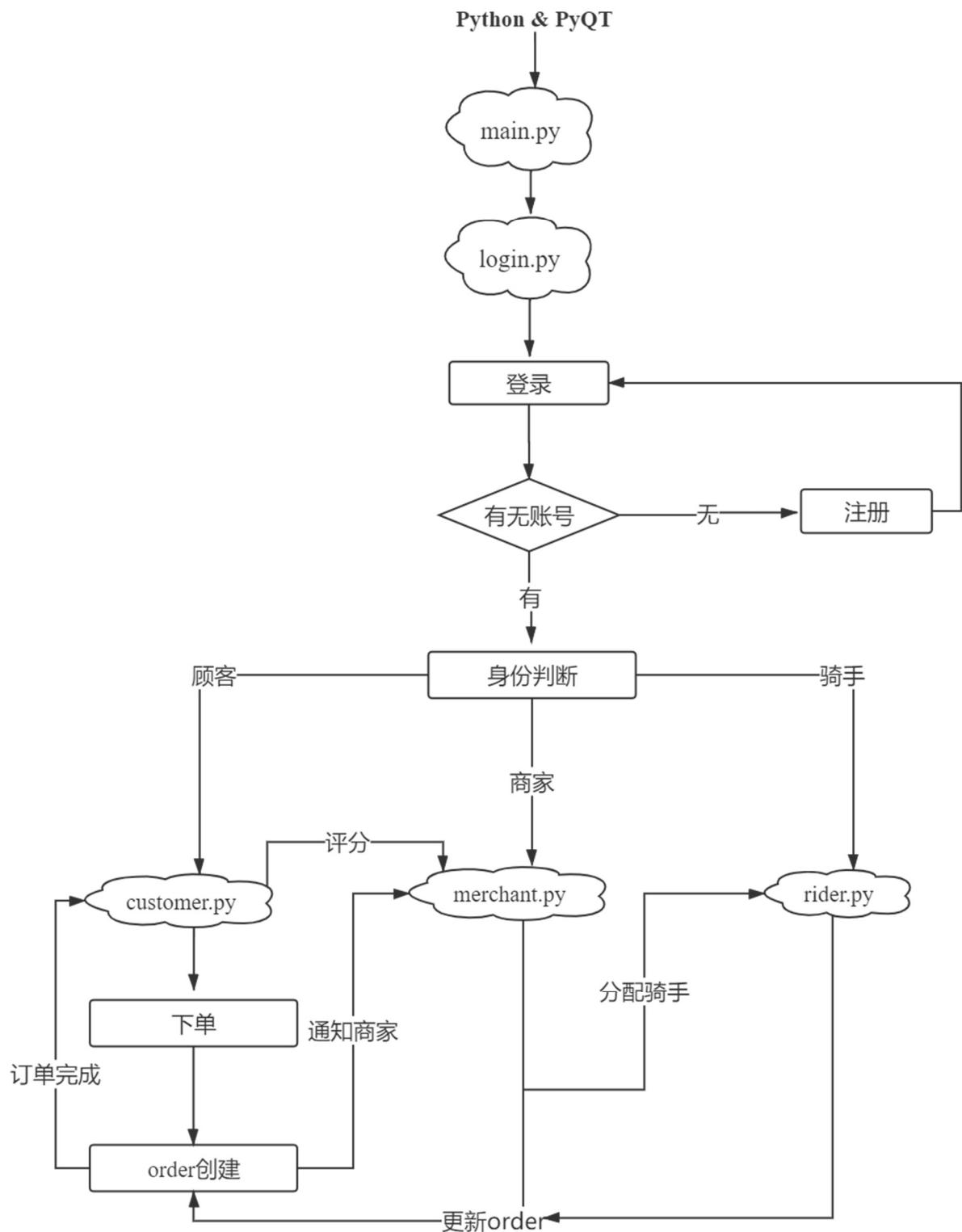
```
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
from PyQt5.QtGui import *
from pymysql import *
from os import listdir
from email.mime.text import MIMEText
import smtplib, random, sys, time, os, re, requests
```

整体文件为一个 Pycharm 工程项目

cover, image, product, verification: 图片文件	
.idea, __pycache__, venv: Pycharm 工程文件	
customer.py: 顾客客户端	login.py: 登录注册界面
main.py: 主函数文件	merchant.py: 商家客户端
rider.py: 骑手客户端	sent_email.py: 发送邮箱验证码函数
util.py: 小工具类	verification_code.py: 生成验证码函数

📁 .idea	6/21/2021 4:16 AM	File folder
📁 __pycache__	6/21/2021 3:31 AM	File folder
📁 cover	6/21/2021 2:36 AM	File folder
📁 image	6/20/2021 3:42 PM	File folder
📁 product	6/16/2021 11:52 PM	File folder
📁 venv	6/9/2021 10:35 PM	File folder
📁 verification	6/9/2021 10:38 PM	File folder
📄 customer.py	6/21/2021 3:57 AM	Python File 54 KB
📄 login.py	6/16/2021 10:06 PM	Python File 31 KB
📄 main.py	6/20/2021 4:15 AM	Python File 2 KB
📄 merchant.py	6/21/2021 4:09 AM	Python File 48 KB
📄 rider.py	6/20/2021 5:24 PM	Python File 49 KB
📄 sent_email.py	6/21/2021 3:29 AM	Python File 2 KB
📄 util.py	6/19/2021 11:24 PM	Python File 1 KB
📄 verification_code.py	6/21/2021 3:17 AM	Python File 1 KB

## 2. 流程图



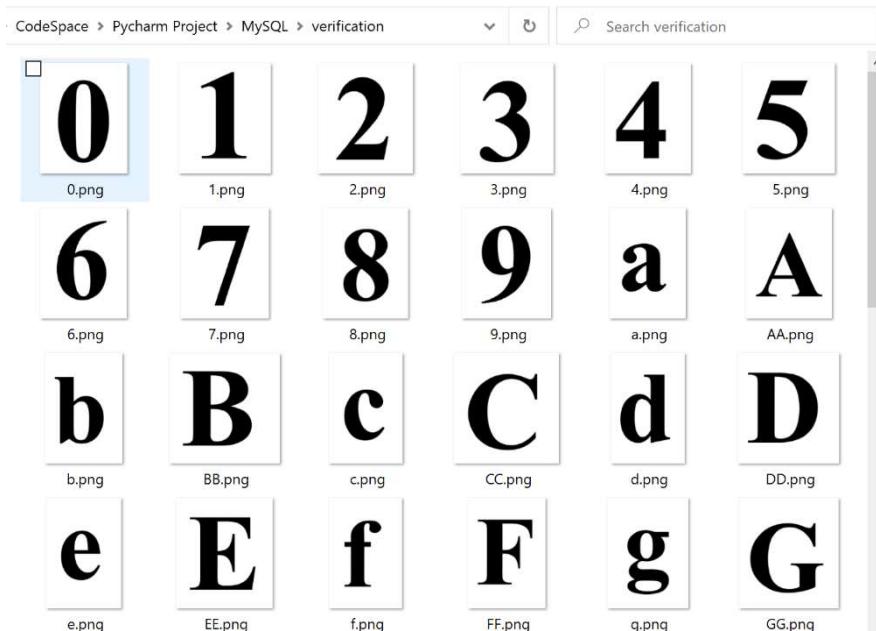
### 3. 部分功能设计

(1) 数据库连接: 使用 pymysql 的接口

```
connect = Connect(
    host="124.71.228.59",
    port=3306,
    user="DB_USER066",
    passwd="DB_USER066@123",
    db="user066db",
    charset="utf8"
)
```

(2) 登录界面的验证码

首先在./MySQL/verification 下准备好验证码图片, 如图: 图片的文件名即代表了验证码的答案。



然后通过如下代码读取该文件夹, 随机选取四张图片, 将这四张图片的文件名作为参数返回, 图片的文件名相拼凑就得到了验证码的答案, 然后在界面中调用这四张图片路径拼凑就可以成功展示验证码:

验证码

W 7 T R

```
def random_verification_code():
    code_list = listdir("./verification/")
    n = len(code_list)
```

```

ans = "" # 验证码答案
path_list = []
for i in range(4):
    code_index = random.randint(0, n-1)
    ans += code_list[code_index][0]
    path_list.append('./verification/%s' % code_list[code_index])
)
return ans, path_list

```

每一次登录失败/点击验证码(通过 mouseEvent 实现)就会重新调用该函数,实现验证码的刷新。

### (3) 注册界面的邮箱和手机号格式检测

通过正则表达式实现:

手机号码: `re.compile(r"^\+?\d+(-\d+)*$")`

邮箱地址:

```

re.compile("^[a-zA-Z0-9_-]+@[a-zA-Z0-9_-]+(\.[a-zA-Z0-9_-]+)+$")
re.compile("^[a-zA-Z0-9_-]+@test+(\.[a-zA-Z0-9_-]+)+$")

```

### (4) 邮箱验证码的发送

使用 `python` 的 `SMTP` 服务

```

import random
import smtplib
from email.mime.text import MIMEText

def sent_email(receiver):
    mail_host = "smtp.qq.com"
    mail_user = "963491881@qq.com"
    mail_pass = "mxdkglodrquqbbjc"
    sender = "963491881@qq.com"
    title = "【饱了没】验证码"
    code = ""
    for i in range(6):
        code += str(random.randint(0, 9))
    content = "【饱了没】验证码: %s" % code
    message = MIMEText(content, "plain", "utf-8")
    message["From"] = "{}".format(sender)
    message["To"] = ",".join(receiver)
    message["Subject"] = title
    try:
        smtp_object = smtplib.SMTP_SSL(mail_host, 465)

```

```
smtp_object.login(mail_user, mail_pass)
smtp_object.sendmail(sender, receiver, message.as_string())
return code
except smtplib.SMTPException as error:
    return str(error)
```

### 饱了没-邮箱验证码 ☆

发件人: **Kuroko** <963491881@qq.com>   
时间: 2021年6月16日 (星期三) 上午2:07  
收件人: Kuroko <963491881@qq.com>

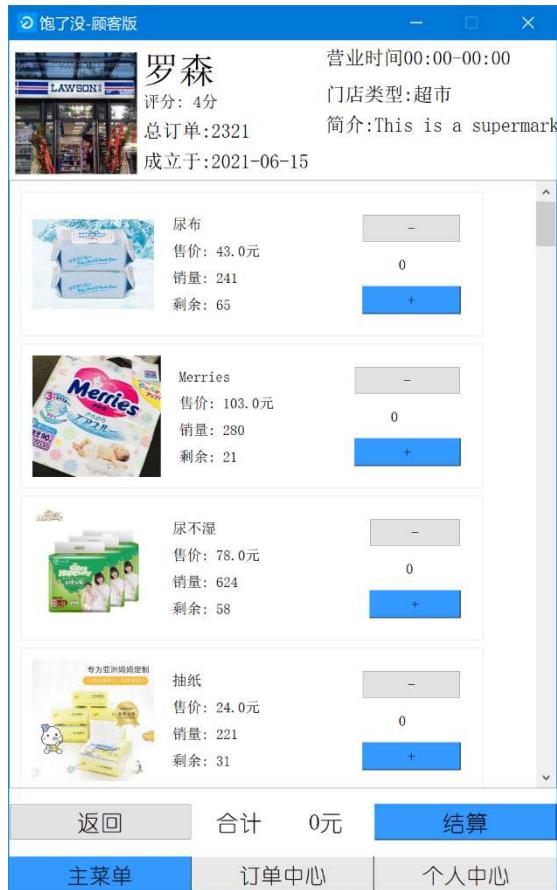
【饱了没】验证码: 062372, 请在 5 分钟内完成操作。如非本人操作, 请忽略。

### (5) 界面之间的跳转

点击"登录"后通过 `pyqtSignal()` 的接口 `emit()` 发射信号来触发槽函数

然后打开相应的界面。

### (6) 滚动界面的设计: 以下图为例子



```
# product_info 包含了全部的商品信息
# product_list 是一个 QScrollArea 对象
items = QWidget() # 创建一个窗口
vLayout = QVBoxLayout(items) # 在窗口中创建垂直布局
for item in product_info:
    # 添加一个滚动 item
    vLayout.addWidget(self.create_product_item(item))
self.product_list.setWidget(items) # 设置窗口

# 设计一个滚动 item
def create_product_item(self, item):
    groupBox = QGroupBox(self)
    text_layout = QVBoxLayout() # 创建垂直布局
    text_layout.addWidget(name) # 设置商品名称
    text_layout.addWidget(price) # 设置商品价格
    text_layout.addWidget(sales) # 设置商品销量
    text_layout.addWidget(left) # 设置商品剩余
    text_widget = QWidget() # 创建窗口
    text_widget.setLayout(text_layout) # 设置布局

    button_layout = QVBoxLayout() # 创建垂直布局
    button_layout.addWidget(minus_button) # 设置加减和数量
    button_layout.addWidget(self.number[item[0]]) # 设置商品数量
    button_layout.addWidget(add_button)
    button_widget = QWidget() # 创建窗口
    button_widget.setLayout(button_layout) # 设置布局

    main_layout = QHBoxLayout() # 整体水平布局
    main_layout.addWidget(pic) # 添加图片
    main_layout.addWidget(text_widget) # 嵌套布局
    main_layout.addWidget(button_widget) # 嵌套布局
    groupBox.setLayout(main_layout) # 设置主布局
    return groupBox # 返回
```

#### 4. 主要界面



图 5-1 登陆界面



图 5-2 注册界面



图 5-3 注册错误信息提示



图 5-4 顾客确认注册界面



图 5-5 骑手确认注册界面



图 5-6 商家确认注册界面

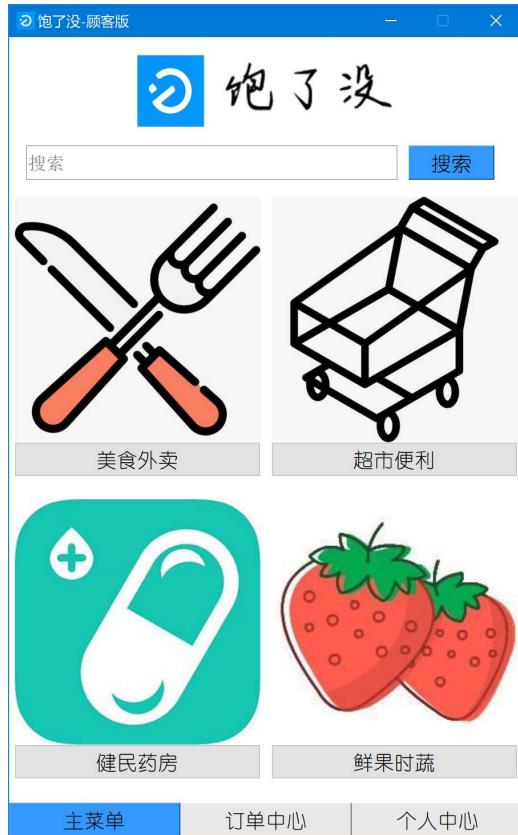


图 5-7 顾客版-主菜单界面



图 5-8 顾客版-浏览界面

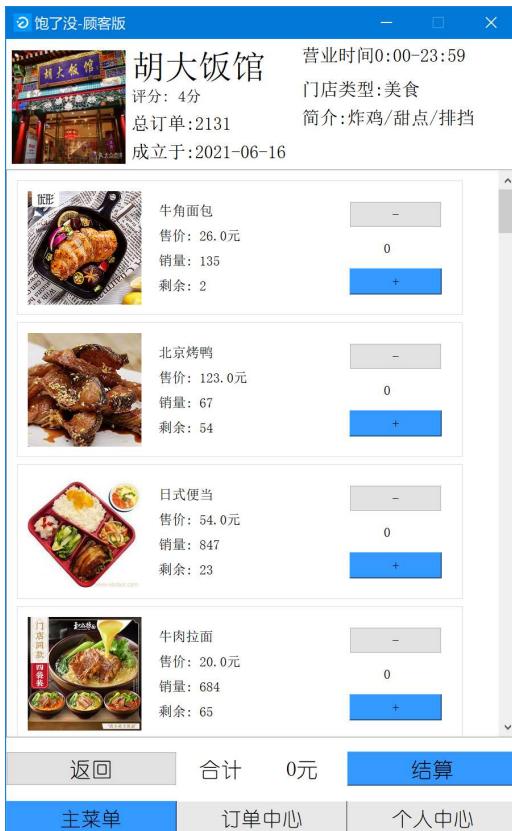


图 5-9 顾客版-点单界面

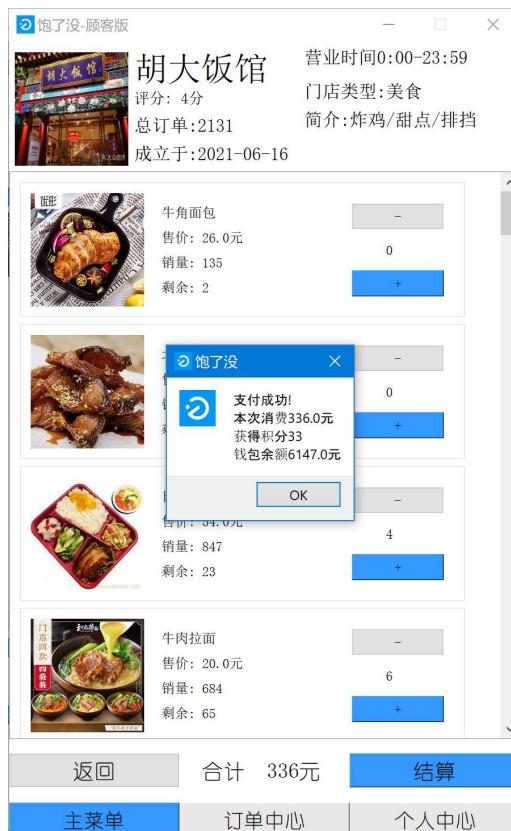


图 5-10 顾客版-消费界面



图 5-11 顾客版-搜索界面

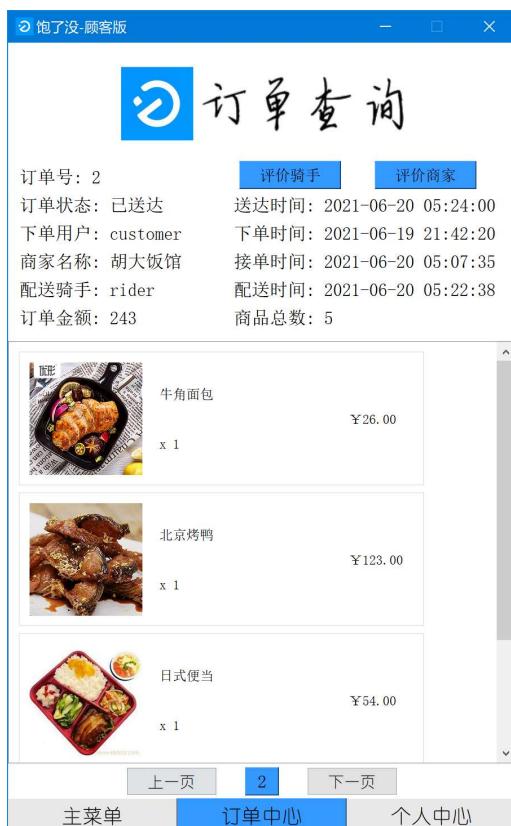


图 5-12 顾客版-订单浏览界面



图 5-13 顾客版-点单界面



图 5-14 顾客版-个人中心界面



图 5-15 顾客版-修改密码界面



图 5-16 顾客版-钱包充值界面



图 5-17 邮箱修改、电话修改、积分兑换、地址修改



图 5-18 商家版-主菜单



图 5-19 商家版-浏览界面

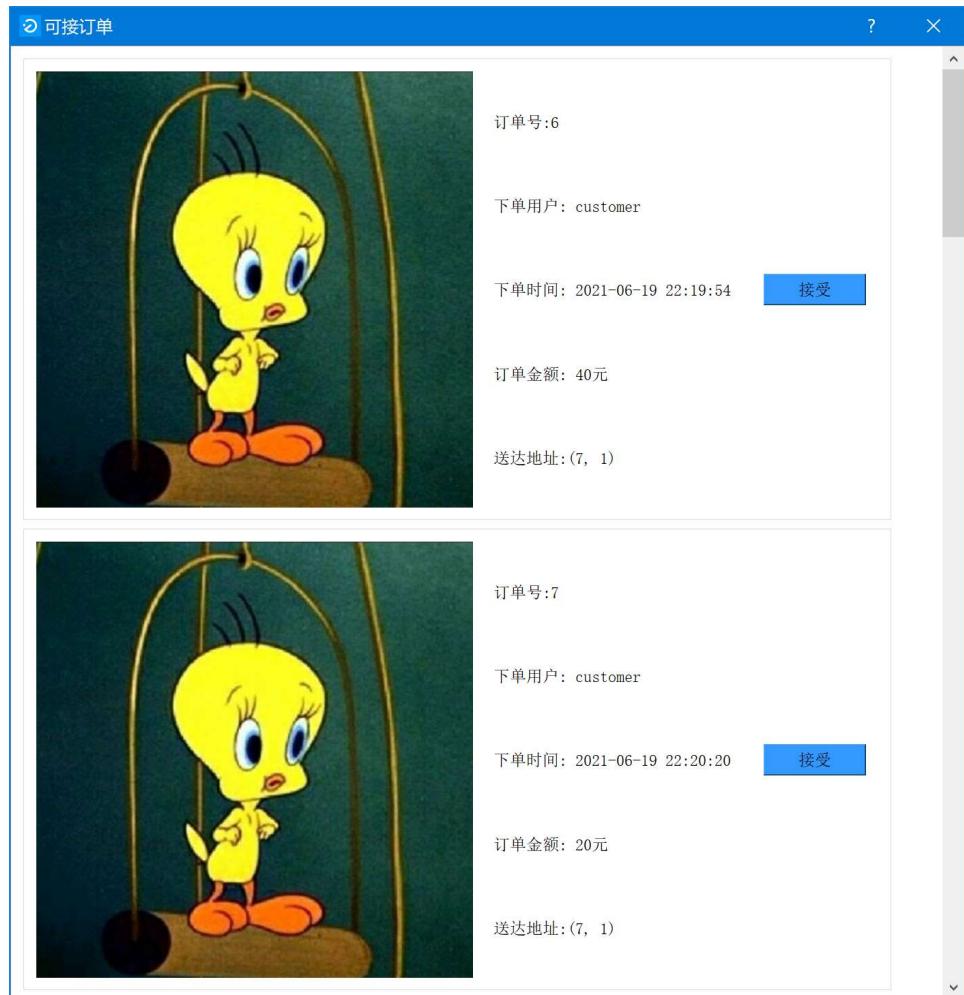


图 5-20 商家版-接单中心

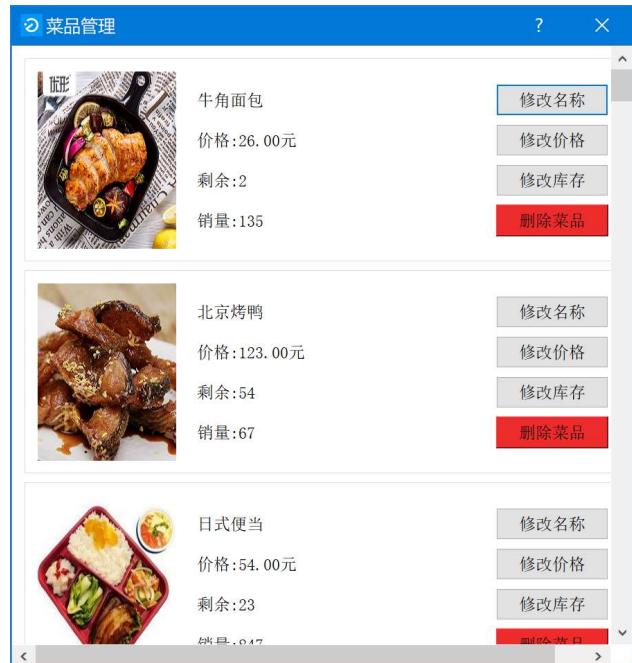


图 5-21 商家版-菜品管理

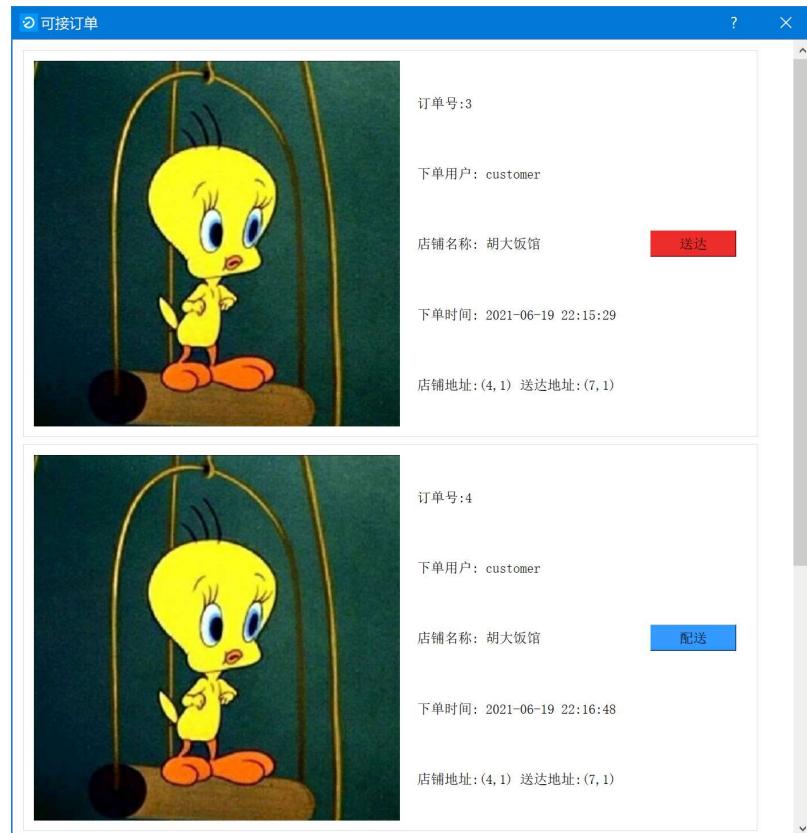


图 5-22 商家版-评价中心

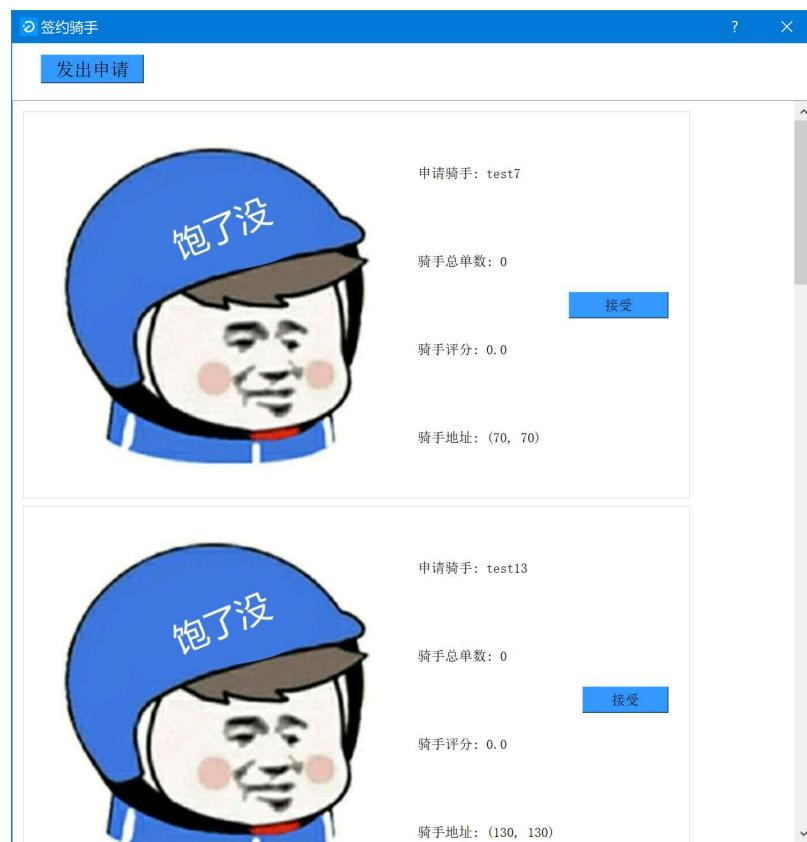


图 5-23 商家版-签约中心



图 5-24 商家版-个人中心



图 5-25 骑手版-主菜单



图 5-26 骑手版-订单中心



图 5-27 骑手版-个人中心

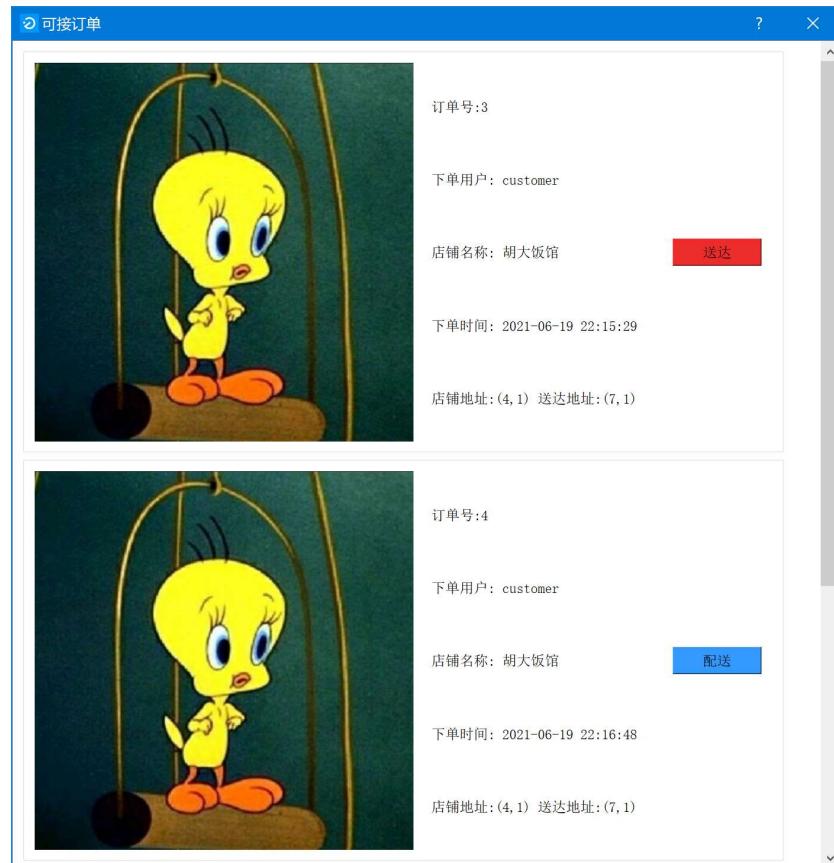


图 5-28 骑手版-接单中心



图 5-29 骑手版-骑手排行



图 5-30 骑手版-评价中心

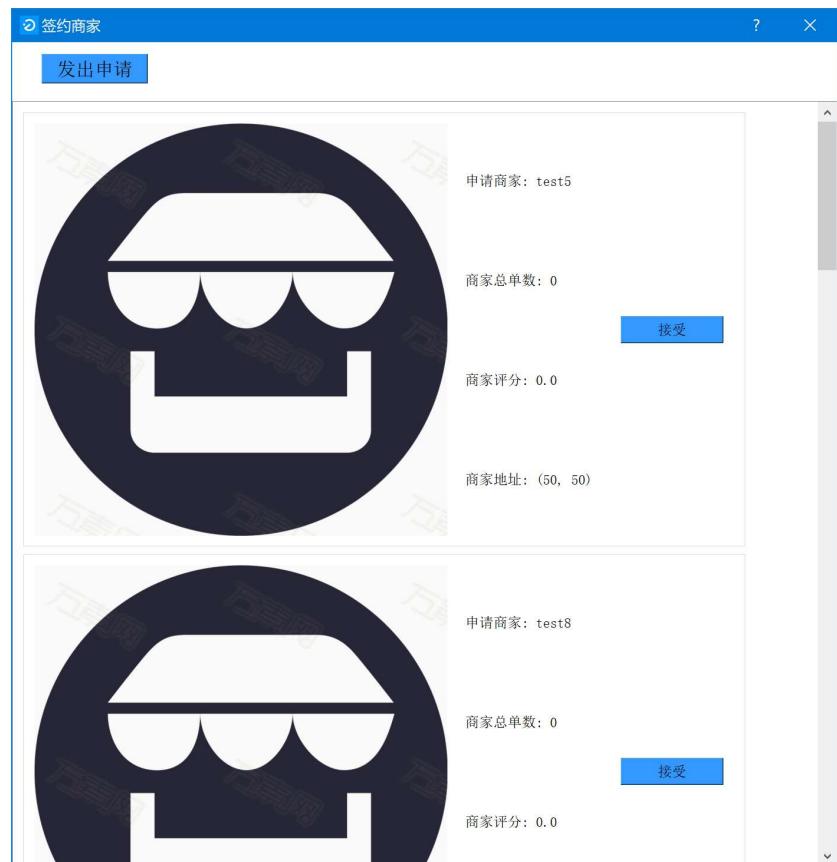


图 5-31 骑手版-签约中心