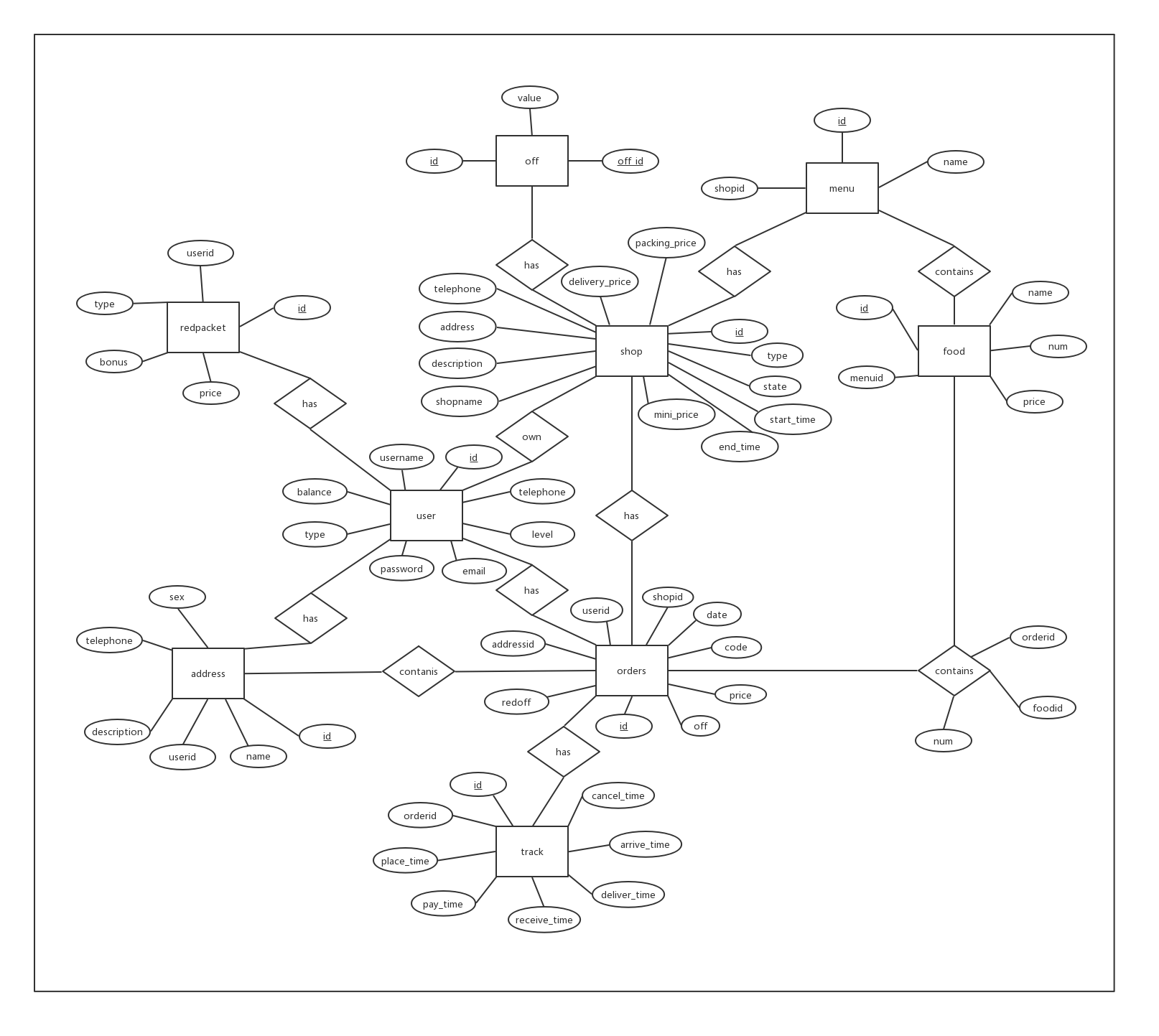
**大作业说明文档**

**161250204 张寅**

**一、数据库设计**

**1、ER图**

****

1. **数据库表（表名，各列名及数据类型，外键关系）**
2. **user表**

|  |  |  |
| --- | --- | --- |
| **列名** | **数据类型** | **备注** |
| id | int(11) | 主键 |
| username | varchar(255) | 无 |
| password | varchar(255) | 无 |
| email | varchar(255) | 无 |
| balance | double(11, 2) | 无 |
| type | varchar(255) | 无 |
| telephone | varchar(255) | 无 |
| level | varchar(255) | 无 |

1. **shop表**

|  |  |  |
| --- | --- | --- |
| **列名** | **数据类型** | **备注** |
| id | int(11) | 主键 |
| userid | int(11) | *外键* |
| shopname | varchar(255) | 无 |
| address | varchar(255) | 无 |
| start\_time | varchar(255) | 无 |
| end\_time | varchar(255) | 无 |
| telephone | varchar(255) | 无 |
| mini\_price | int(11) | 无 |
| delivery\_price | double(11, 1) | 无 |
| packing\_price | double(11, 1) | 无 |
| type | varchar(255) | 无 |
| state | varchar(255) | 无 |
| description | varchar(255) | 无 |

1. **orders表**

|  |  |  |
| --- | --- | --- |
| **列名** | **数据类型** | **备注** |
| id | int(11) | 主键 |
| userid | int(11) | *外键* |
| shopid | int(11) | *外键* |
| addressid | int(11) | *外键* |
| code | varchar(255) | 无 |
| date | varchar(255) | 无 |
| price | double(11, 2) | 无 |
| state | varchar(255) | 无 |
| off | int(11) | 无 |
| redoff | int(11) | 无 |

1. **menu表**

|  |  |  |
| --- | --- | --- |
| **列名** | **数据类型** | **备注** |
| id | int(11) | 主键 |
| shopid | int(11) | *外键* |
| name | varchar(255) | 无 |

1. **food表**

|  |  |  |
| --- | --- | --- |
| **列名** | **数据类型** | **备注** |
| id | int(11) | 主键 |
| menuid | int(11) | *外键* |
| name | varchar(255) | 无 |
| num | int(11) | 无 |
| price | double(11, 2) | 无 |

1. **address表**

|  |  |  |
| --- | --- | --- |
| **列名** | **数据类型** | **备注** |
| id | int(11) | 主键 |
| userid | int(11) | *外键* |
| name | varchar(255) | 无 |
| sex | varchar(255) | 无 |
| telephone | varchar(255) | 无 |
| description | varchar(255) | 无 |

1. **track表**

|  |  |  |
| --- | --- | --- |
| **列名** | **数据类型** | **备注** |
| id | int(11) | 主键 |
| orderid | int(11) | *外键* |
| place\_time | varchar(255) | 无 |
| pay\_time | varchar(255) | 无 |
| receive\_time | varchar(255) | 无 |
| deliver\_time | varchar(255) | 无 |
| arrive\_time | varchar(255) | 无 |
| cancel\_time | varchar(255) | 无 |

1. **orderinfo表**

|  |  |  |
| --- | --- | --- |
| **列名** | **数据类型** | **备注** |
| id | int(11) | 主键 |
| orderid | int(11) | *外键* |
| foodid | int(11) | *外键* |
| num | int(11) | 无 |

1. **off表**

|  |  |  |
| --- | --- | --- |
| **列名** | **数据类型** | **备注** |
| id | int(11) | 主键 |
| offid | int(11) | 主键 |
| value | int(11) | 无 |

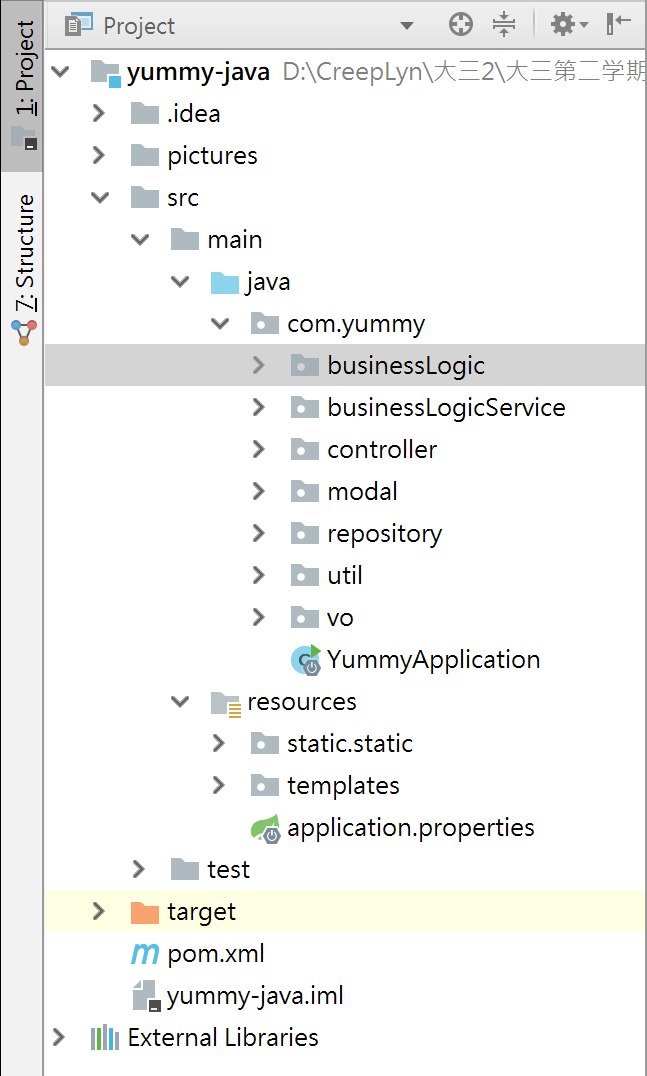
1. **redpacket表**

|  |  |  |
| --- | --- | --- |
| **列名** | **数据类型** | **备注** |
| id | int(11) | 主键 |
| userid | int(11) | *外键* |
| bonus | int(11) | 无 |
| price | int(11) | 无 |
| type | varchar(255) | 无 |

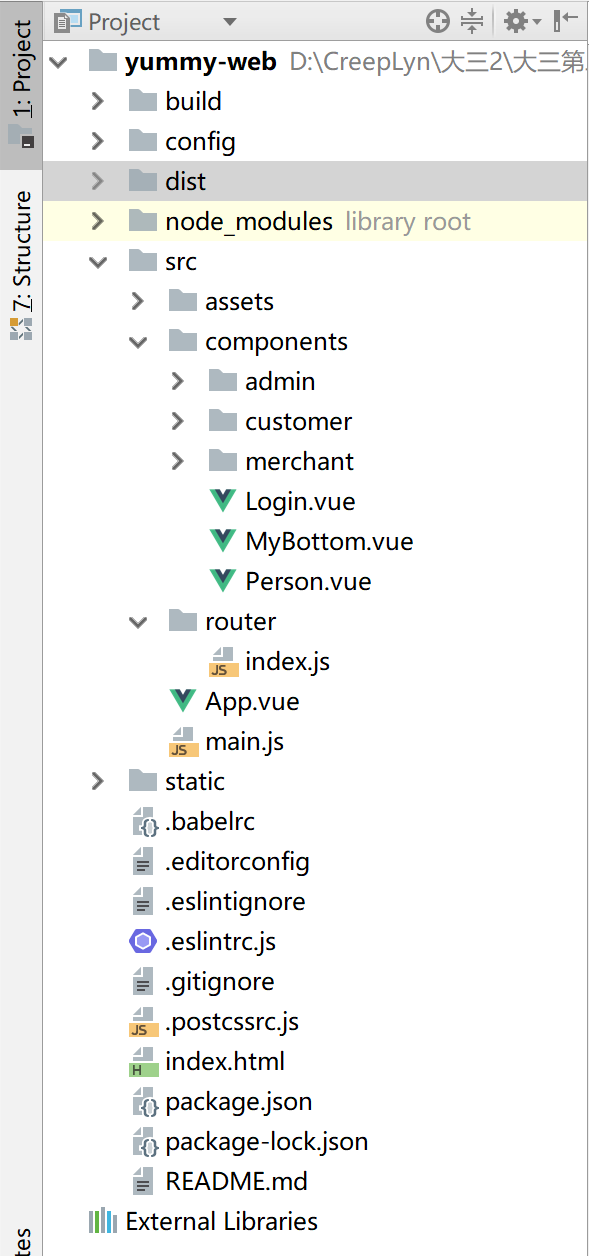
1. **settlement表**

|  |  |  |
| --- | --- | --- |
| **列名** | **数据类型** | **备注** |
| id | int(11) | 主键 |
| orderid | int(11) | 无 |
| date | int(11) | 无 |
| profit | int(11) | 无 |

**二、架构设计1、工程的项目结构截图**

****

（整体项目结构）

****

（前端项目结构）

1. **是否使用框架，如Spring+Hibernate 框架**

使用Spring-boot + Hibernate框架

1. **前端页面是否使用框架，如bootstrap框架**

使用VUE框架

**三、类设计**

**1、各包的类：名称及职责（注：不要使用截图）**

1. **businessLogic包**

|  |  |
| --- | --- |
| **名称** | **职责** |
| AddressBL | AddressBLService的实现类，负责有关地址的所有逻辑 |
| AdminBL | AdminBLService的实现类，负责管理员部分的所有逻辑 |
| OrderBL | OrderBLService的实现类，负责有关订单的所有逻辑 |
| ShopBL | ShopBLService的实现类，负责有关商店的所有逻辑 |
| UserBL | UserBLService的实现类，负责有关用户的所有逻辑 |

1. **businessLogicService包**

|  |  |
| --- | --- |
| **名称** | **职责** |
| AddressBLService | 接口类，负责有关地址的所有接口 |
| AdminBLService | 接口类，负责管理员部分的所有接口 |
| OrderBLService | 接口类，负责有关订单的所有接口 |
| ShopBLService | 接口类，负责有关商店的所有接口 |
| UserBLService | 接口类，负责有关用户的所有接口 |

1. **controller包**

|  |  |
| --- | --- |
| **名称** | **职责** |
| AddressController | 控制类，负责有关地址的所有前后端交互 |
| AdminController | 控制类，负责管理员部分的所有前后端交互 |
| OrderController | 控制类，负责有关订单的所有前后端交互 |
| ShopController | 控制类，负责有关商店的所有前后端交互 |
| UserController | 控制类，负责有关用户的所有前后端交互 |

1. **modal包**

|  |  |
| --- | --- |
| **名称** | **职责** |
| Address | 实体类，负责映射数据库中的address表 |
| Food | 实体类，负责映射数据库中的food表 |
| Menu | 实体类，负责映射数据库中的menu表 |
| OrderInfo | 实体类，负责映射数据库中的orderinfo表 |
| Orders | 实体类，负责映射数据库中的orders表 |
| Redpacket | 实体类，负责映射数据库中的redpacket表 |
| Settlement | 实体类，负责映射数据库中的settlement表 |
| Shop | 实体类，负责映射数据库中的shop表 |
| Track | 实体类，负责映射数据库中的track表 |
| User | 实体类，负责映射数据库中的user表 |

1. **repository包**

|  |  |
| --- | --- |
| **名称** | **职责** |
| AddressRepository | JPA仓库接口类，负责对数据库的address表进行增删改查 |
| FoodRepository | JPA仓库接口类，负责对数据库的food表进行增删改查 |
| MenuRepository | JPA仓库接口类，负责对数据库的menu表进行增删改查 |
| OrderInfoRepository | JPA仓库接口类，负责对数据库的orderinfo表进行增删改查 |
| OrdersRepository | JPA仓库接口类，负责对数据库的orders表进行增删改查 |
| RedpacketRepository | JPA仓库接口类，负责对数据库的redpacket表进行增删改查 |
| SettlementRepository | JPA仓库接口类，负责对数据库的settlement表进行增删改查 |
| ShopRepository | JPA仓库接口类，负责对数据库的shop表进行增删改查 |
| TrackRepository | JPA仓库接口类，负责对数据库的track表进行增删改查 |
| UserRepository | JPA仓库接口类，负责对数据库的user表进行增删改查 |

1. **util包**

|  |  |
| --- | --- |
| **名称** | **职责** |
| ConvertUtil | 工具类，负责po类和vo类的相互转换 |
| PictureUtil | 工具类，负责图片的相关操作 |
| RedPacketUtil | 工具类，负责红包的相关操作 |
| ResultMessage | 常量类，负责操作结果常量 |
| NumberMessage | 常量类，负责数字常量 |
| StateMessage | 常量类，负责状态常量 |
| TypeMessage | 常量类，负责类型常量 |

1. **vo包**

|  |  |
| --- | --- |
| **名称** | **职责** |
| AddressVO | VO类，负责封装页面的address信息 |
| AdminInfo | VO类，负责封装页面的admin信息 |
| FoodVO | VO类，负责封装页面的food信息 |
| MenuInfo | VO类，负责封装页面的menu信息 |
| OffVO | VO类，负责封装页面的off信息 |
| OrderVO | VO类，负责封装页面的order信息 |
| RedpacketVO | VO类，负责封装页面的redpacket信息 |
| SettlementVO | VO类，负责封装页面的settlement信息 |
| ShopVO | VO类，负责封装页面的shop信息 |
| StateVO | VO类，负责封装页面的state信息 |
| TrackVO | VO类，负责封装页面的track信息 |
| UserVO | VO类，负责封装页面的user信息 |

1. **各前端的页面：名称及功能（注：不要使用截图）**

由于前段使用了VUE，页面被组件化，所以在此阐述所有组件的名称和功能

1. **基础组件**

|  |  |
| --- | --- |
| **名称** | **功能** |
| Login | 用户登录注册 |
| Mybottom | 所有界面的footer |
| Person | 用户个人信息的增删改查 |

1. **Admin组件**

|  |  |
| --- | --- |
| **名称** | **功能** |
| AdminExami | 管理员审批商店 |
| AdminHome | 管理员主界面以及各界面的跳转 |
| AdminSettle | 管理员结算 |
| AdminUsers | 管理员查看统计信息 |

1. **Customer组件**

|  |  |
| --- | --- |
| **名称** | **功能** |
| CartOrder | 用户下单 |
| CusAccount | 用户账户充值 |
| CusAddress | 用户地址的增删改查 |
| CusBonus | 用户领取红包 |
| CusHome | 用户主界面以及各界面的跳转 |
| CusInfo | 用户个人中心 |
| CusList | 用户查看商店列表 |
| CusOrder | 用户个人订单 |
| ShopDetail | 用户查看商店详情 |

1. **Merchant组件**

|  |  |
| --- | --- |
| **名称** | **功能** |
| MerAccount | 商家账户提现 |
| MerHome | 商家主界面以及各界面的跳转 |
| MerInfo | 商家个人中心 |
| MerOrder | 商家商店订单 |
| MerShop | 商家商店详情 |
| MerShopChecking | 商店待审核信息 |
| MerShopInitial | 商店初始信息 |

**四、其他**

**1、开发环境（数据库，服务器等）**

（1）数据库：MYSQL

（2）服务器：Apache Tomcat

**2、开发心得体会**

（1）前后端分离开发是大势所趋，但是有利有弊：

优点： a. 可以实现真正的前后端解耦

b. 可以大幅提高开发效率

c. 发现bug，可以快速定位是哪里的问题

d. 减少后端服务器的并发/负载压力，异步加载

缺点： a. 需要制定好接口文档，而在项目开发途中，接口可能会变化，此时修改的 代价会变大

b. 前端需要有机制应对后端请求超时以及后端服务宕机的情况，友好的展示 给用户，所以前端压力变大

（2）对前端来说，有一个好的UI设计图是十分重要的，由于该项目需要自己去构思UI设计，所以花费了大量时间在寻找优秀的UI设计图。

（3）对后端来说，初期对整个项目有一个合理的模块划分是必要的，且必须明确各个模块的职责，否则后期开发逻辑会十分紊乱。

（4）对数据库来说，表的设计最重要，既要全方位考虑表的属性及其数据类型，又要构想好每个表之间的关系，要让数据库的表设计尽量符合规范和原则。

（5）代码风格方面，要及时做好注释，变量以及方法的命名要具有一致性且符合规范。

**3、其他需要补充的**

（1）项目构建工具：maven

（2）前端VUE项目中使用的外部插件：echarts（图表插件）、axios（ajax插件）

（3）前端VUE项目中使用的UI框架：element-ui

（4）后端项目中使用的插件：spring-boot自带的mail（邮箱验证）