中期检查报告

项目需求与设计

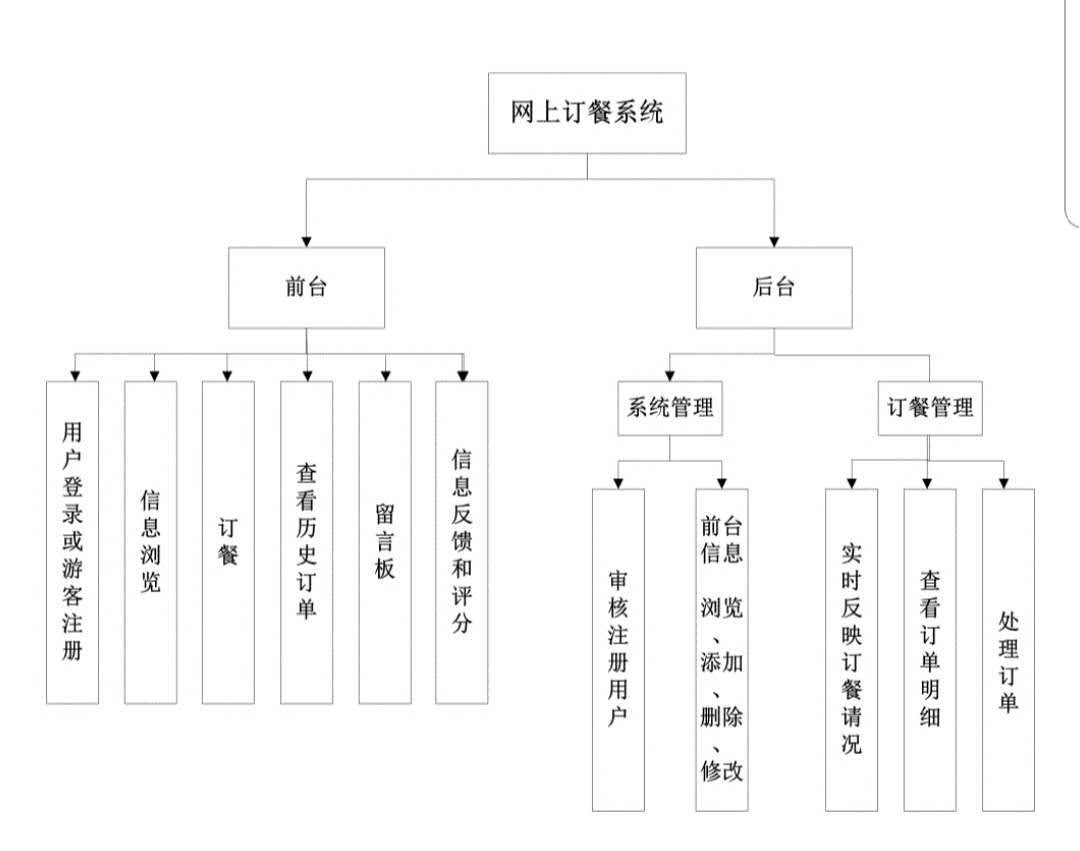
1、 需求部分

现在普遍使用的订餐方式是进行网上预定，这种预订方式方便，错误率也比较低，给人们带来方便，对顾客和餐厅都有很大的帮助。

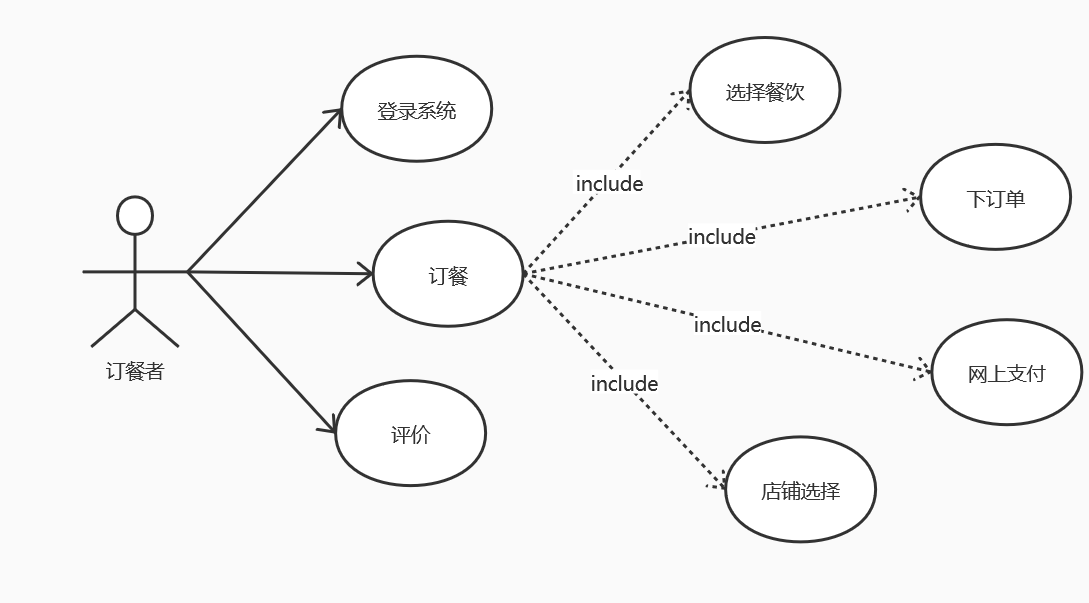
顾客模块的功能包括个人信息管理，美食选购，美食评价三大功能。其中，在个人信息管理中可以对个人信息进行修改、还可以查阅以往订过的美食信息；美食评价中顾客可以这对这次服务质量，留下自己相对餐厅说的话，完成用户与餐厅之间的交互。

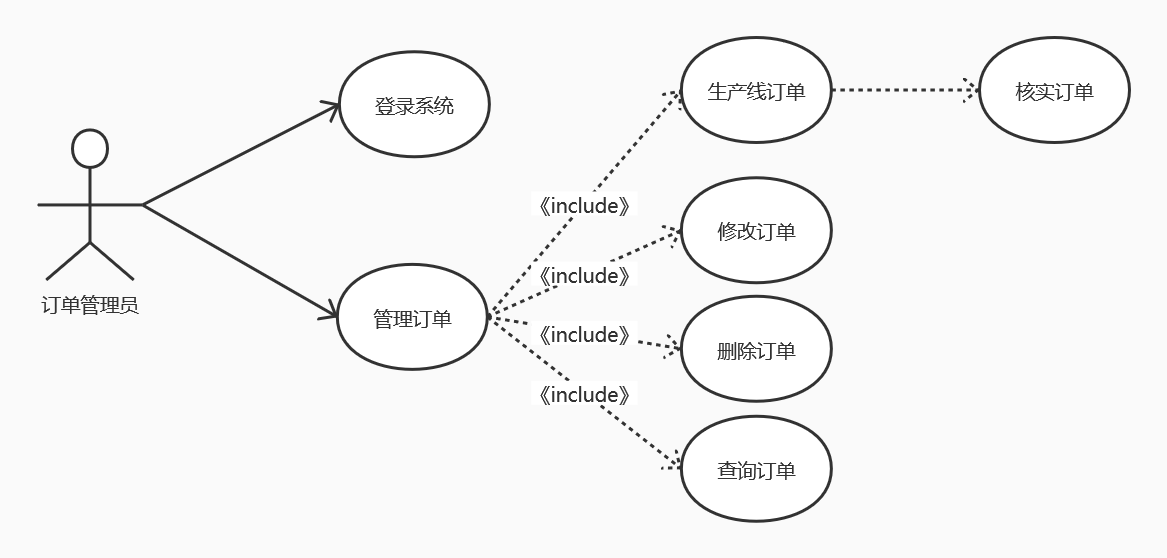
管理员模块的功能包括菜品管理,订单管理,会员信息管理三大功能。其中,在个菜品管理中可以对菜品信息进行添加、修改和查询操作;在订单管理中可以通过未确认、已确认、已下单三种形式进行管理。

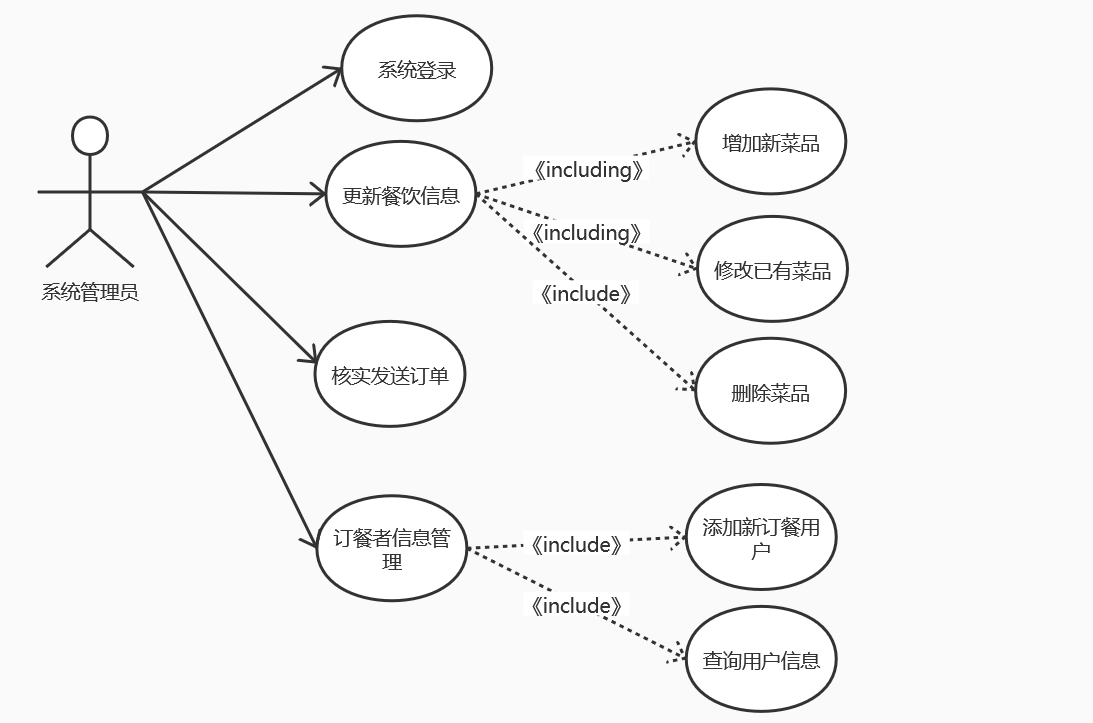
功能模块图



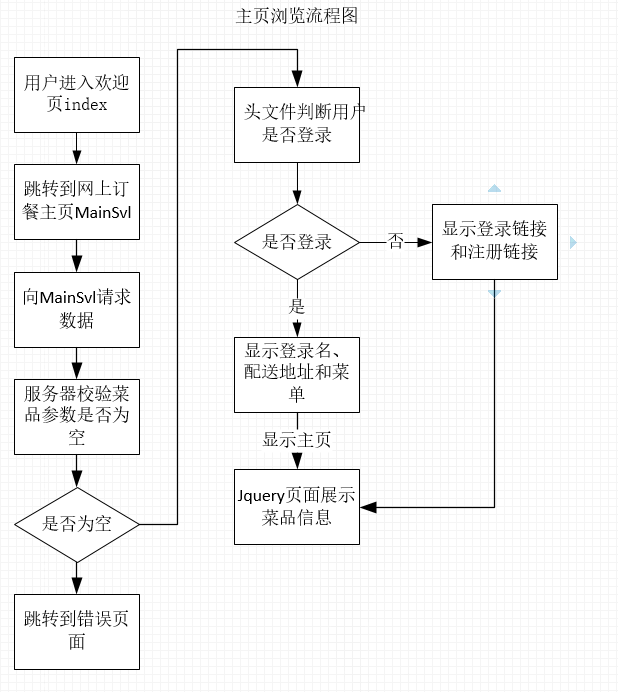
用例图

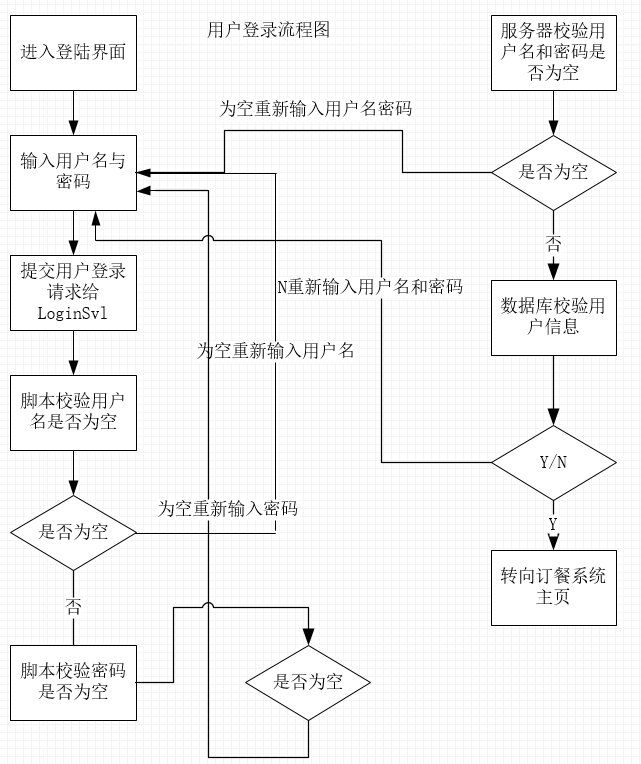


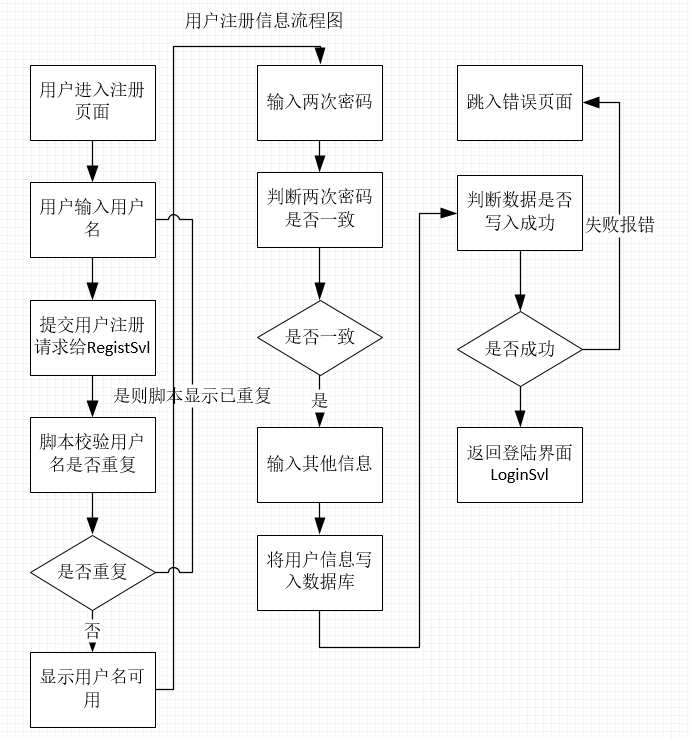


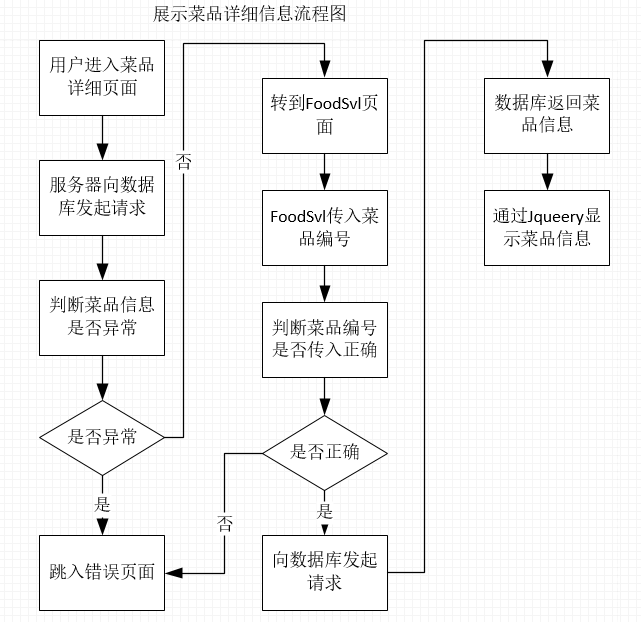


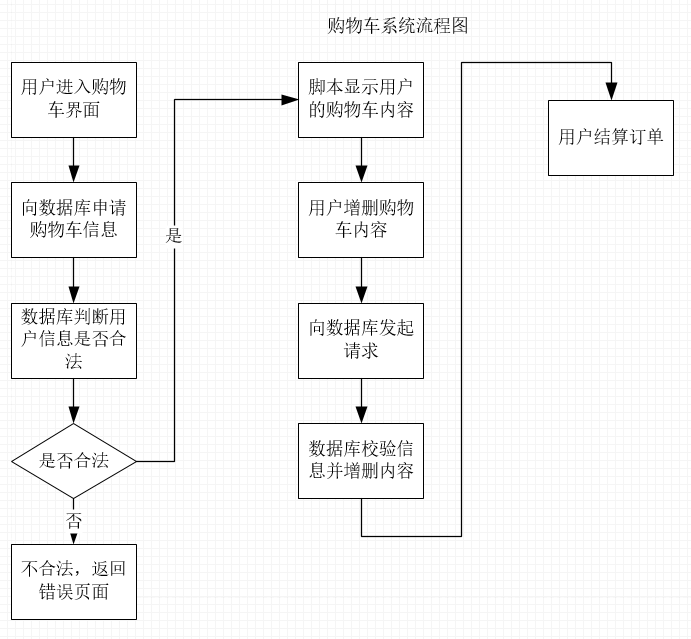
流程图





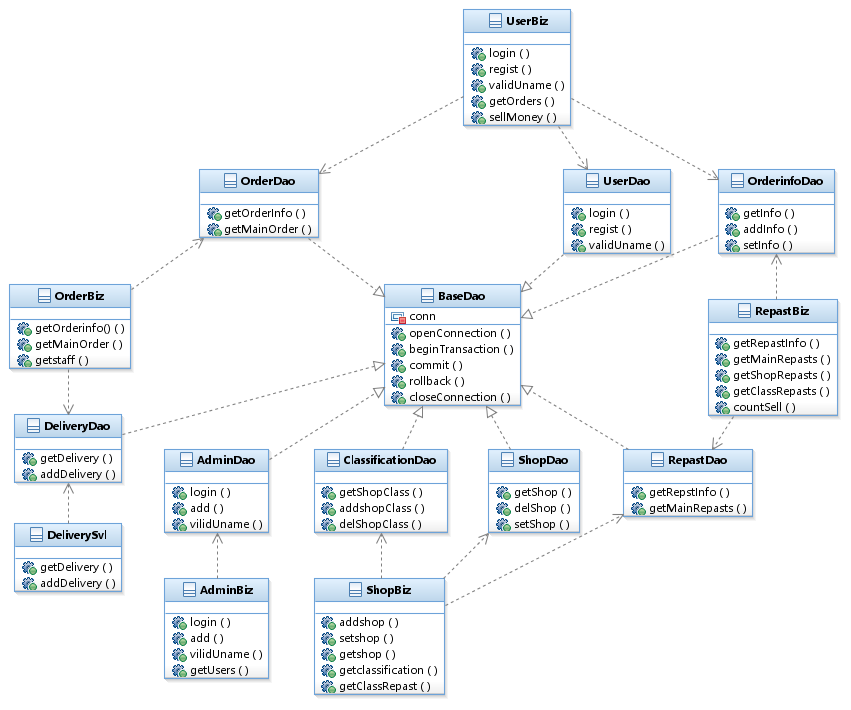




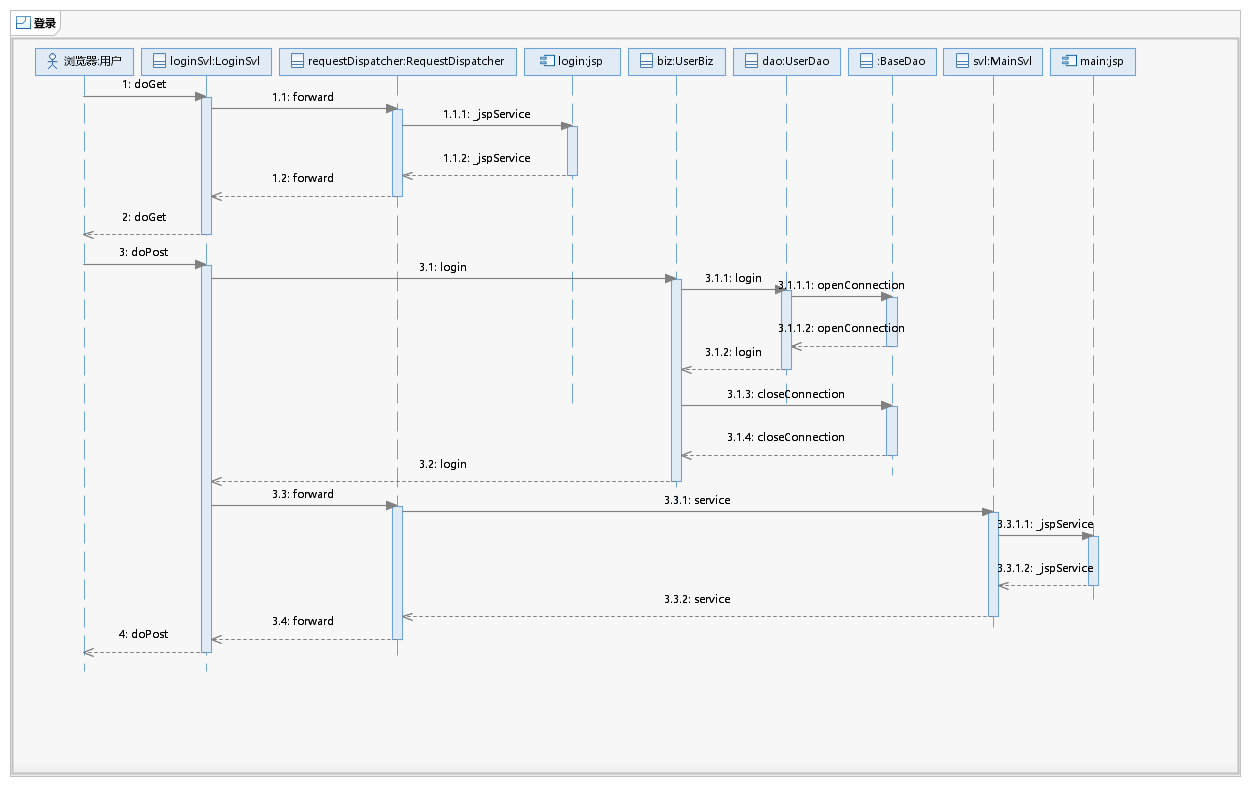


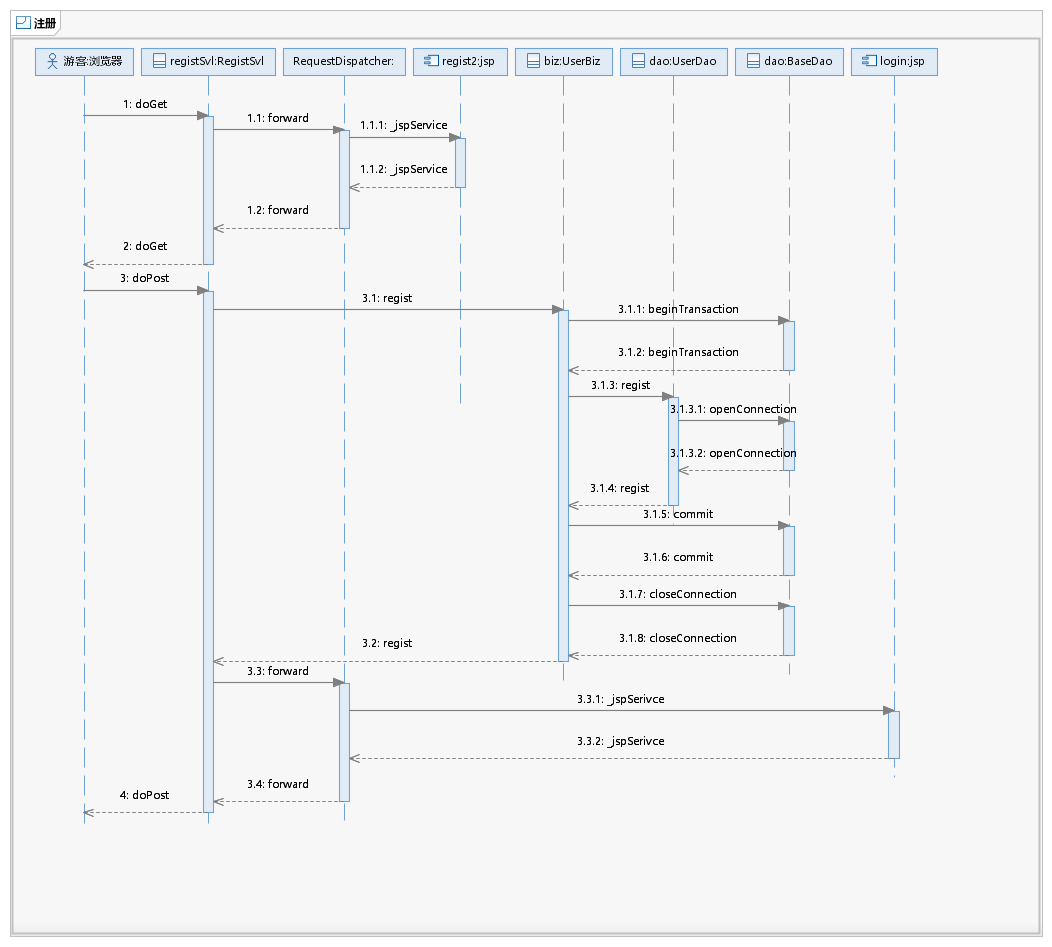
1. 设计部分

类图

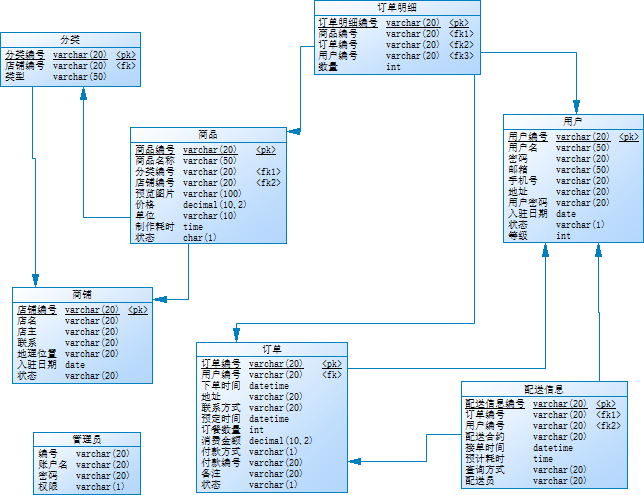


时序图





表的物理模型图



3、 项目的核心代码

public abstract class BaseDao {

    protected Connection conn;   //此处一定要使用protected关键字

    public void openConnection() throws ClassNotFoundException,SQLException, IOException{

        try {

            //打开数据库 ,记录重用数据库的connection

            if(this.conn == null || this.conn.isClosed()) {

                Properties pro = new Properties();

                pro.load(BaseDao.class.getClassLoader().getResourceAsStream("db.properties"));

                String driver = pro.getProperty("mysql.driver");

                String url = pro.getProperty("mysql.url");

                String username = pro.getProperty("mysql.username");

                String password = pro.getProperty("mysql.password");

                Class.forName(driver);

                conn = DriverManager.getConnection(url, username, password);

                Log.logger.info("打开数据库连接.....");

            }else {

                Log.logger.info("重用已经打开的数据库连接.....");

            }

        }catch (ClassNotFoundException e) {

            Log.logger.error("检查数据库驱动包是否存在", e);

            throw e;

        }catch (SQLException e) {

            Log.logger.error("检查数据库的连接串是否正确", e);

        }

    }

    /\*\*

     \* 开启事务

     \* @throws Exception

     \*/

    public void beginTransaction() throws Exception{

        this.openConnection();

        this.conn.setAutoCommit(false);  //默认是自动提交模式，false表示收到提交事务

        Log.logger.info("开启事务...");

    }

    /\*\*

     \* 提交事务

     \* @throws Exception

     \*/

    public void commit()throws Exception {

        if(this.conn != null) {

            this.conn.commit();

            Log.logger.info("提交事务...");

        }

    }

    /\*\*

     \* 回滚事务

     \* @throws Exception

     \*/

    public void rollback() throws Exception{

        if(this.conn != null) {

            this.conn.rollback();

            Log.logger.info("回滚事务...");

        }

    }

    public void closeConnection() {

        if(this.conn != null) {

            try {

                this.conn.close();

                Log.logger.info("关闭数据库连接...");

            } catch (Exception e) {

                Log.logger.error(e.getMessage(),e);

            }

        }

    }

}

4、 测试用例

一、测试用例的选择原则：

测试用例的确定，依靠等价类的划分。

本系统的测试中，测试用例的选择原则是：

（1）设计尽可能少的测试用例，覆盖所有的有效等价类。

（2）针对每一个无效等价类，设计一个测试用例来覆盖它。

二、需要测试的功能：

后台 ： （1）订单管理 前台： （1）登录功能

（2）点餐菜单管理 （2）注册功能

（3）结账功能 （3）用户购物车

（4）用户管理

1. 测试用例

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例编号 | 测试用例名称 | 用例状态 | 测试结果 |
| Bug-001 | 用户登录 | 已执行 | 测试通过 |
| Bug-002 | 用户注册 | 已执行 | 测试通过 |
| Bug-003 | 用户购物车 | 已执行 | 测试通过 |
| Bug-004 | 订单管理 | 已执行 | 测试通过 |
| Bug-005 | 菜单管理 | 已执行 | 测试通过 |
| Bug-006 | 结账 | 已执行 | 测试通过 |
| Bug-007 | 用户管理 | 已执行 | 测试通过 |



