**在线购物网设计与实现**

**一：项目预期功能实现**

在线购物业务系统中需要处理：

1. 用户注册、2.登录/退出、3.业务办理等功能。

其中业务办理等功能包括：

（1） 用户管理，包括用户的登录和注册功能。  
（2） 商品查询功能。  
（3） 商品分类。

（4） 购物车的实现  
（5） 商品购买。

**二：根据需求分析问题**

1. 分析业务系统的结构、实现的功能、涉及到的知识、应用到的编码软件、编码环境等

本系统的软件开发环境如下：

编程语言： Java。

操作系统： Windows 7/8/10/11。

JDK版本： Java SE Development KIT(JDK) Version 10.0。

开发工具： Idea。

数据库： MySQL 8.0。

服务器：Tomcat 9.0.39

**三：选用技术：**

**前端：Vue3 、Vite 、Element Plus 、tailwindcss 、Pinia**

**使用代理和axios解决前后端的通信问题和跨域问题**

export default defineConfig({

  server: {

    proxy: {

      '/api': {

        target: 'http://localhost:8080',

        changeOrigin: true,

        rewrite: (path) => path.replace(/^\/api/,''),

      }

    }

  },

const Instance = axios.create({

    baseURL: "/api",

    timeout: 7000,

})

import { Instance } from "@/axios";

export function getAllProducts() {

    return Instance.get("/product/queryAll")

}

因为获取后端数据较慢，因此使用异步方式等待调用获取完后端数据后再渲染视图

// 在页面加载时获取产品列表

onMounted(async () => {

  try {

    // 调用API获取产品列表数据

    const response = await getAllProducts();

    console.log(response)

    // 将产品列表保存到products变量中

    products.value = response.data.data;

  } catch (error) {

    console.error('获取数据失败:', error);

  }

});

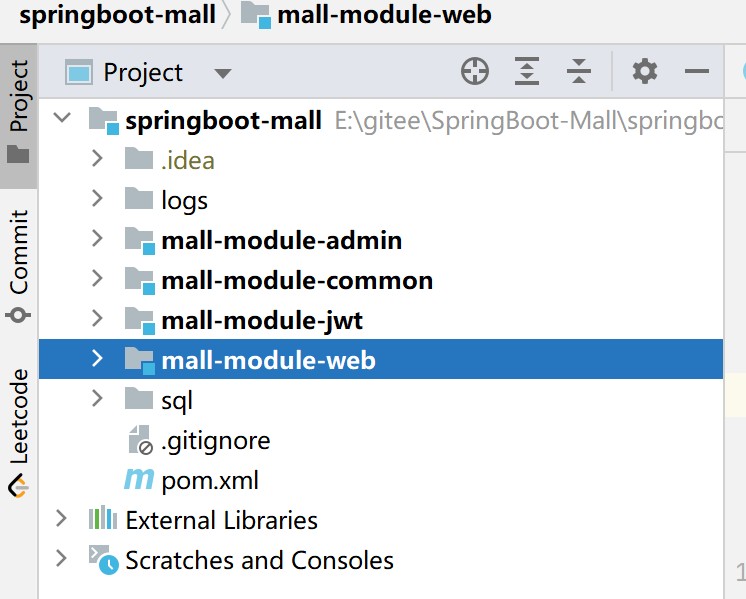
**使用Pinia代替Vuex,实现数据持久化**

import { defineStore } from 'pinia'

export const useUserStore = defineStore('user', () => {})

**后端：SpringBoot、SpringSecurity、JWT 、MySQL 、Mybatis Plus 、Logback 、Knife4j、Maven**

**项目结构**



*<!--子模块管理-->*<modules>

*<!—- 前台 模块 -->*  
 <module>mall-module-web</module>

*<!—- 管理员 模块 -->*  
 <module>mall-module-admin</module>

*<!—- 公共 模块 -->*  
 <module>mall-module-common</module>  
*<!-- JWT 模块 -->* <module>mall-module-jwt</module>  
</modules>

**自定义日志注解：**

@Retention(RetentionPolicy.*RUNTIME*)  
@Target({ElementType.*METHOD*})  
@Documented  
public @interface ApiOperationLog {  
 */\*\*  
 \* API 功能描述  
 \* @return {@link String}  
 \*/* String description() default "";  
}

使用AOP的AspectJ实现日志功能，在ApiOperationLogAspect中以自定义 @ApiOperationLog 注解为切点，凡是添加 @ApiOperationLog 的方法，都会执行环绕中的代码

@Aspect  
@Component  
@Slf4j  
public class ApiOperationLogAspect {@Pointcut("@annotation(com.newone.mall.common.aspect.ApiOperationLog)")  
 public void apiOperationLog() {}

@Around("apiOperationLog()")  
public Object doAround(ProceedingJoinPoint joinPoint) throws Throwable {

//设置日志格式等...

**JWT(JSON Web Token):**

public class JwtAuthenticationFilter extends AbstractAuthenticationProcessingFilter {  
 */\*\*  
 \* 指定用户登录的访问地址  
 \*/* public JwtAuthenticationFilter() {  
 super(new AntPathRequestMatcher("/login","POST"));  
 }

.

.

.

}

POST请求<http://localhost:8080/login>，将被JwtAuthenticationFilter过滤器拦截，然后执行一系列过滤链，

在TokenAuthenticationFilter中，获取请求头中 key 为 Authorization 的值判断值是否以 Bearer 开头，并依赖注入JwtTokenHelper工具类验证token正确性，等一系列操作 后

*// 继续执行下一个过滤器*filterChain.doFilter(request, response);

**业务逻辑:**

用户在购物车中可能同时选中多个商品，同时结算后将多个商品同时写入orders订单表

Controller方法:

@RestController  
@Slf4j  
@RequestMapping("/order")  
public class OrderController {  
 @Autowired  
 private OrderService orderService;  
 @ApiOperation("添加订单")//Knife4j 文档注解  
 @ApiOperationLog(description = "添加订单")//自定义日志注解  
 @PostMapping("/addOrders")  
 public Response addOrders(@RequestBody List<OrderDO> orderList){  
 int i = 0;  
 try {  
 for (OrderDO orderDO : orderList) {  
 i += orderService.addOrder(orderDO);  
 }  
 return Response.*success*(i);  
 } catch (Exception e) {  
 *log*.error("订单写入失败！", e);  
 return Response.*fail*("订单写入失败！");  
 }  
 }

Service方法使用默认的事务管理，传播行为是 REQUIRED，隔离级别是MySQL数据库的默认隔离级别。

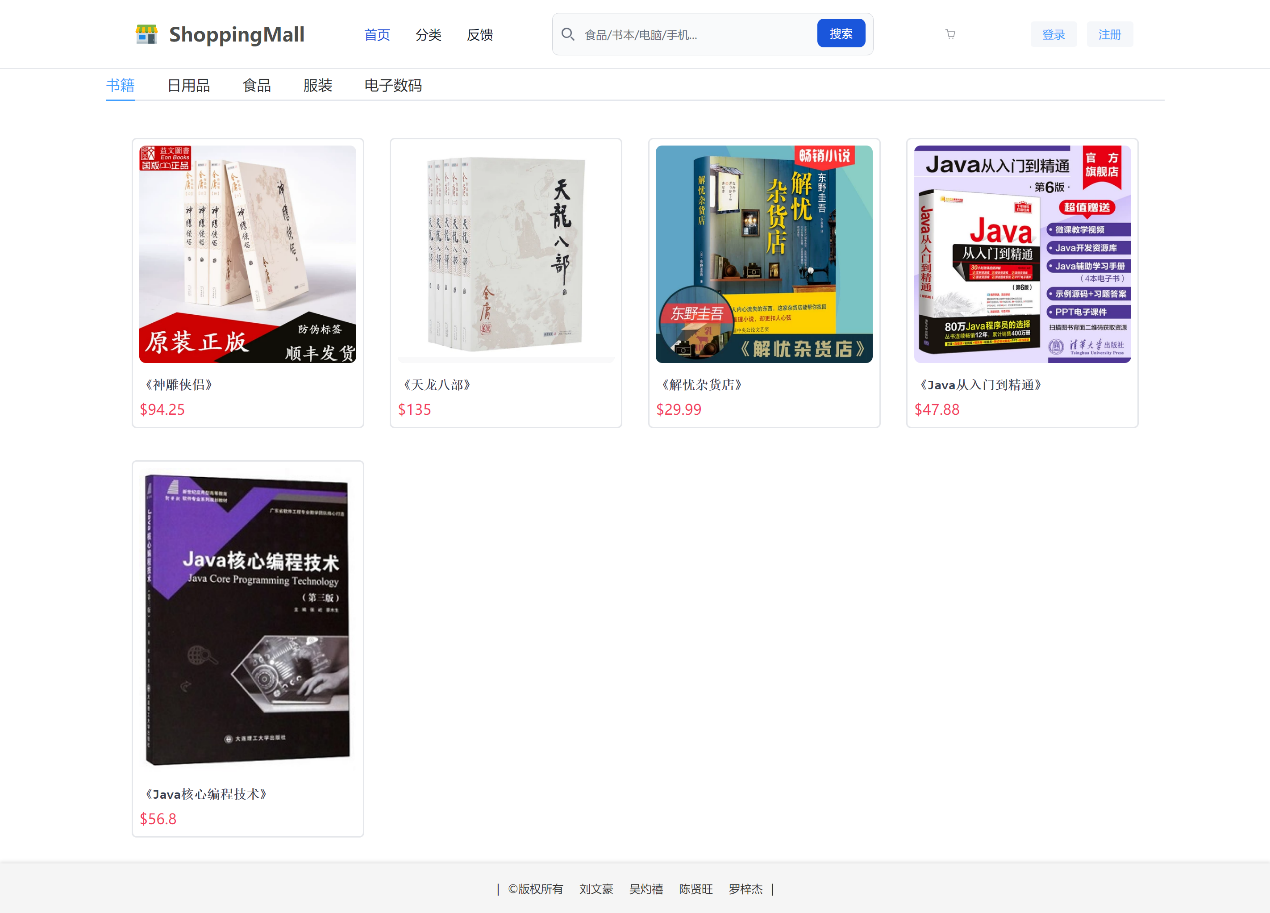
@Service  
@Slf4j  
public class OrderServiceImpl implements OrderService {  
 @Autowired  
 private OrderMapper orderMapper;  
 @Override  
 @Transactional  
 public int addOrder(OrderDO orderDO) {  
 return orderMapper.insert(orderDO);  
 }

**五：运行截图：**

**首页：**



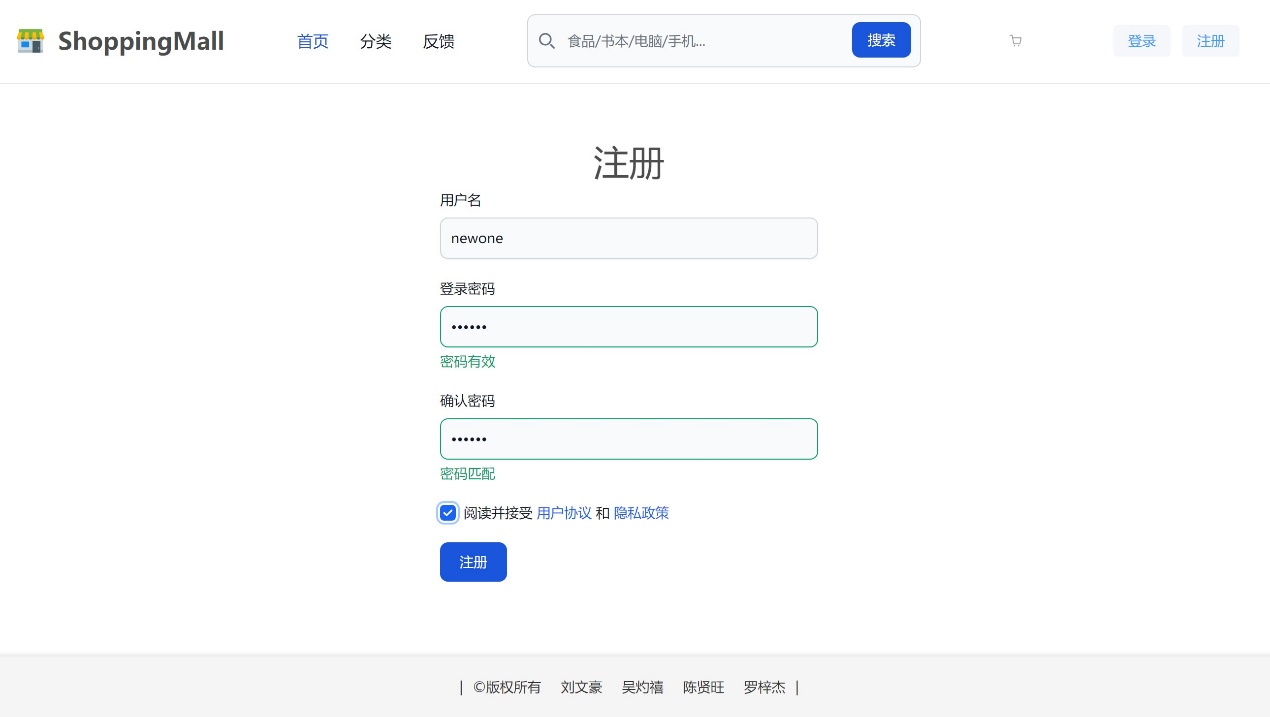
**商品分类：**



**登陆：**



**注册：**



**商品详细信息：**



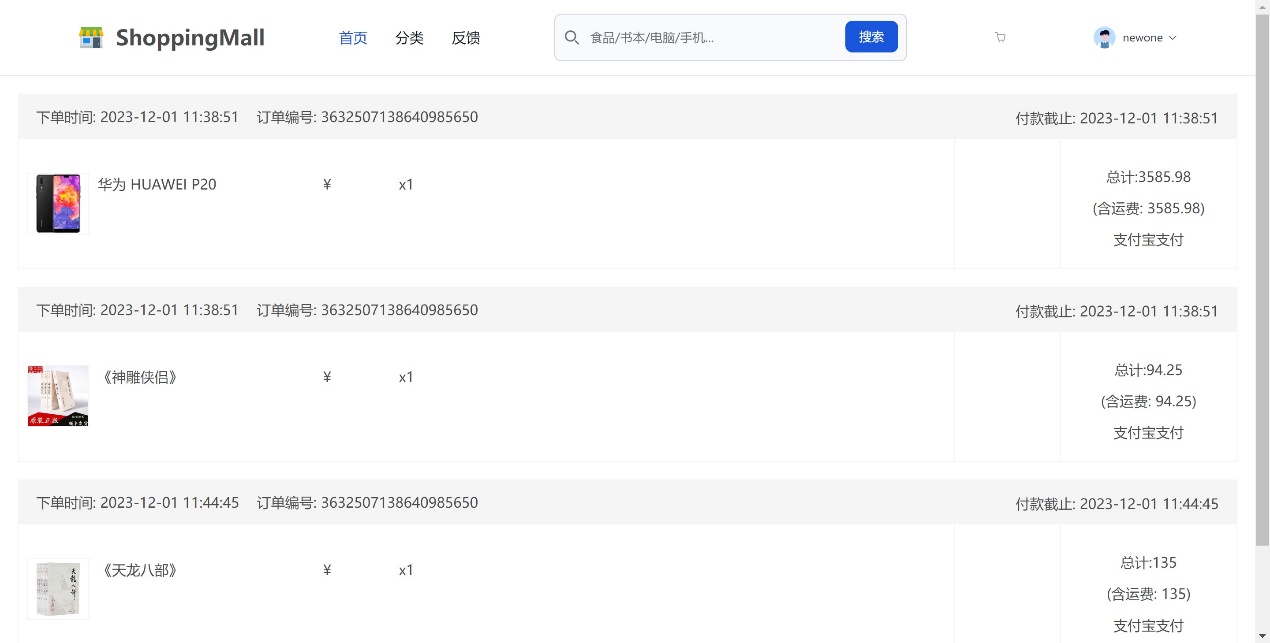
**购物车：**



**支付：**



**订单：**



**六：心得体会：**

在开始项目之前，对需求进行充分的分析是至关重要的。因此我们首先通过对业务系统的结构和功能进行了仔细分析，了解项目的整体框架和核心功能，从而保障后续的技术选型和开发工作有条不紊地进行。

在这个项目中，我们选择了SpringBoot作为后端框架，结合SpringSecurity和JWT实现了用户认证和授权，使用了MySQL数据库进行数据存储，采用了Mybatis Plus简化持久层操作。在前端方面，选择了Vue3作为主要框架，搭配Vite、Element Plus等工具，以及tailwindcss和Pinia，这些都是目前比较流行和高效的技术栈，能够提高开发效率。在开发阶段MySQL驱动设为p6spy监控Mybatis Plus的内置方法的功能，在控制台能够看到sql语句。使用Logback和@Slf4j代替log4j进行日志记录，提高开发效率。使用Knife4j代替Swagger和Postman让项目文档、接口文档等更清晰可见并且可实时测试接口，有助于项目的维护和后期的优化。

我们认识到团队协作和良好的沟通是很重要的。因此我们使用Maven进行多模块项目构建、gitee上进行版本控制，我们团队成员密切合作，及时解决问题，确保项目的顺利进行。

在项目开发过程中也遇到了许多困难，让我认识到要时刻保持着持续学习的态度。新技术和工具层出不穷，及时了解和学习最新的技术。而且要探索出一套解决问题的方法也很重要。

通过这次在线购物网项目的设计与实现，我们深入了解了整个软件开发的流程，提高了自己的技术能力和团队协作能力。这不仅是对课程知识的实际应用，也是对项目开发实践的一次宝贵经验。在以后的学习中，不断拓展提高自己的技术能力，保持学习的热情，更好地应对各种挑战。