华南理工大学

《网络应用开发》课程实验报告

实验题目： 设计并实现一个电子商务网站的开发和在线部署 \_\_

姓名： 曾震 学号： 202030443356 \_\_\_\_

班级： 计科2

指导教师： 布社辉

|  |
| --- |
| **实验概述** |
| 【实验题目】  设计并实现一个电子商务网站的开发和在线部署.  【基本功能要求】  顾客：  用户的注册、登录、注销  购买流程（浏览/查询->添加至购物篮->付款->发送电子邮件确认收货）  销售：  商品目录的管理（包括最基本的增删改等操作）  后台销售统计报表，销售状态  客户的浏览/购买日志记录  【实验环境和工具】  开发环境：Windows11 + IDEA + VsCode  部署环境：Ubuntu 20.04  工具：  前端：Vue3  后端：SpringBoot + SpringSecurity + JWT + Redis + Minio + 积木报表 + MyBatisPlus  运维：Jenkins + Nginx |
| **实验内容**   1. 系统设计：   部署网站：<http://www.feijizhadan.cn/>  GitHub项目地址：<https://github.com/feijizhadandan/Market-Total>   1. 项目概述   该项目为前后端分离项目，前端通过Vue3框架实现，后端使用SpringBoot作为基础框架。  前端：使用Vue3的组件式思想，将各个界面设置为一个组件，分别实现，其次引入了BootStrap和ElementUI组件库前端的UI设计。  后端：主要分为几个功能模块：   1. 用户的授权和认证模块，主要通过整合SpringSecurity和JWT实现，同时引入Redis作为用户登录信息Toekn的缓存工具。 2. 引入Minio高性能对象存储工具，实现图片文件等信息保存。 3. 具体业务逻辑的实现，主要包括各层Controller、Service、Dao，数据库采用MySQL+Druid连接池。 4. 数据库设计：   核心部分Product表的设计，需要设定一个唯一id，用于后台进行数据搜索链接，该id不会在前端进行展示，但是他是该表的主键，和一般作为其他表对product的外键，使用雪花算法生成。另外code则是商品编号，可以对用户进行展示，也是唯一的。  CreateBy和updateBy用来存储创建者和最近一次更新着的id（也是雪花算法生成的），使用id作为连接字段更加安全合理。  del\_flag用作逻辑删除的标志位，任何前端操作对商品信息不进行彻底删除，0表示未删除，2表示删除，删除后后端在使用MyBatisPlus获取数据时会进行默认筛选，值选择值为0的未删除商品。如果手写SQL，则需要手动补上where条件。  Version字段用于实现乐观锁，防止多线程下数据的读写不安全，并且效率远高于悲观锁。  Is\_show字段表示是否将该商品在商城中展示给用户。  以上的几个特殊字段都可以在application.yml文件中的MyBatisPlus属性进行配置，指定字段，实现自动更新。    （Product表数据库字段）    （application.yml配置信息）    （自动填充注解）  用户-认证-权限系统的数据库表设计：使用 权限-角色-用户，三角关系设计，1角色 -- n权限；1用户 -- n角色；从而实现用户权限的存储。    （角色表）    （用户表）    （功能表）  其三者的连接表就不贴出来了，都是id-id的双字段表。   1. 后端代码实现 2. 用户登录系统：我将其放在framework子模块的security文件夹中。   Login和Register的Controller层就是普通的接口，调用Service层的服务。其中还包含通过用户传入的Token，到Redis中检验登录状态的方法。    （Login和Register的控制层）  主要是Service层中对身份校验的执行流程比较复杂。首先在Service层中将前端传来的username和password封装成UsernamePasswordAuthenticationToken对象（是一个Security中的一个类），然后在Service中自动装配AuthenticationManager对象，并使用他的authenticate方法将UsernamePasswordAuthenticationToken传入进行身份校验。    （Service层认证流程）  其中authenticate会调用UserDetailService的实现类进行验证，因此我们还需要创建一个UserDetailServiceImpl来实现UserDetail，其中需要重写一个方法loadUserByUsername，如果不重写该方法其账号密码验证只能从本地获取，重写后可以通过自定义，到数据库中获取账号密码信息（不是进行比对，而是取出其信息，即一个对象User），如果不存在则抛出异常即可（异常会进行同一捕获处理）；找到了，则将User封装成Security中的一个UserDetail类的自定义子类LoginUser（也是需要自己重写的），return即可。    （自定义重写的UserDetailServiceImpl子类中的loadUserByUsername方法）  UserDetail在SpringSecurity中是一个重要的类，验证从数据库中返回的User对象都需要封装成UserDetail或其子类，其中包含很多信息，最重要的就是三个：username，password，和permission（用来做权限认证的）。    （SpringSecurity的UserDetails）  由于其信息不足以实现我们的业务要求，因此我们需要设计其子类，其中需要包含更多的自定义信息，比如User的个人资料，还有Token等。    （自定义的LoginUser -- 继承自UserDetail）  同时这里设定了一个permission列表，是用来存用户的权限信息的，在上一步从数据库中查询用户信息的时候，createLoginUser就是用来获取权限信息的，这就涉及到了SpringSecurity的数据库设计，主要分成了几个表，分别用来存角色，角色和权限的关系，和用户和角色的关系，因此就能通过用户来查询到角色。    （数据库设计）  然后在LoginUser中设置permissions属性，在后续权限认证的时候，会调用LoginUser的getAuthorities方法，返回的值就是权限信息列表，这里就需要用到流处理。这样就能将权限转换成SpringSecurity能处理的格式，从而进行权限控制，然后在后续接口上添加@PreAuthorize("hasAuthority('system:manager')")注解，就能够对该接口实现权限控制了，表示有system:manager权限的才能访问该接口，否则抛出异常。    （比如该接口就需要管理权限才能访问）    （重写后的方法）  返回的LoginUser信息会被一个叫DaoAuthenticationProvider的处理器进行处理，获取其中的Username和password，并在配置文件SecurityConfig中注入一个解码器，系统就会对password进行Bcrypt解码（存在数据库中的是加密过后的形式）。然后在和前端传来的用户输入进行比对，从而判断是否登录成功。紧接着就是通过tokenService中的工具函数，生成Token，并将用户信息和对应的Token存入Redis中。      （部分工具函数）  而对于权限认证方面，主要是添加一个自定义的过滤器JwtAuthenticationTokenFilter，其继承自OncePerRequestFilter，也需要在SecurityConfig中注入到IOC容器中。  其作用主要是判断前端来的请求是否有携带Token，过滤分为几种情况  1、没有token, 直接放行(只能访问登录接口) (提示用户未登录)  2、有token, 但是缓存已经过期, 直接放行 (提示用户未登录)  3、有token, 但是token非法, 直接放行 (提示用户未登录)  (后两种情况都是getLoginUser中返回null的)   1. 有token, 验证通过, 将其信息和权限列表加入Holder, 交给后续拦截器统一处理   先根据用户信息从数据库中获取并封装出LoginUser对象。如果对象不为空，即存在该用户且token没过期，就存入SecurityContextHolder中，后续交给拦截器进行处理。否则就直接放行。因为后续的权限认证信息需要从SecurityContextHolder获取，如果其中没有信息，就不会允许访问对应接口。    （过滤器内容）   1. 具体业务逻辑   主要分为几个部分：用户在商城的接口、管理员管理商品的接口、购物车接口（包括支付）、以及管理员获取统计报表的接口。  （1）首先是商品-用户接口ProductBuyerController：就三种，展示用户可见商品、展示某个商品的详细信息、用户关键词搜索。其中获取详细信息需要buy的权限（即意味着要登录，因为需要统计用户浏览记录），其他都不需要权限，即匿名用户也可以查看。    （用户-商品接口）  其Service层的实现都比较简单，部分可以直接用MyBatisPlus获取数据库信息返回。    （获取所有商品数据/获取某个商品的详细数据实现）  搜索商品就需要用到手写SQL语句。（需要注意is\_show字段和del\_flag字段需要是用户可见的情况）使用concat进行模糊搜索。     1. 商品-管理员接口ProductManagerController：其接口比较多，包含简单的增删查改接口，同时包括一个上传商品图片的接口。其中比较有代表性的是获取商品详细信息的接口，其中用到了手写SQL，和定义了一个专用的返回类ProductVo，在mapper.xml文件中通过左外连接sys\_user表来获取管理员的名称信息，因为在product表中记录的外表信息都是id，为了展示给前端因此需要进行联表查询。     （查询具体的商品信息SQL语句）  同时在添加商品的方法中，需要进行特殊判断，判断其有无重名的商品，如果有就返回error，并附上错误信息。同时根据默认传入的参数HttpServletRequest，获取其中的token，到redis中取出用户信息，就能获取到用户id，后续在存入数据库信息的时候，就能有createBy的信息了，updateBy同理。    （添加商品方法）  删除商品方法也比较特殊，因为使用的是MyBatisPlus，因此如果直接用delete删除，会导致原本配置的updateTime、version之类的信息都不能自动更新。又因为我这里定义的删除都是逻辑删除，不是直接删数据，因此可以用update方法来进行逻辑删除，并通过token获取updateBy信息（同时还需要去数据库中取出原本数据，再进行update，否则version信息不会自动+1，因为前端传来的数据没有version）。    （删除商品方法）  上传图片方法需要用到Minio，在配置好Minio后，可以写一个Minio的Util工具类，来完成基本的上传、删除操作。首先在设计该上传系统的时候，基本思路都是一个商品对应一个图片，对应一个图片的URL，因此如果在原有图片的情况下，再上传，就应该实现直接覆盖的情况，因此我这里的fileName定义为bucketName + productName，就能保证一个商品只有一个图片。同时商品图片的URL也是Product表中的一个字段，默认为null，上传之后才进行更新。    （Minio图片上传的工具函数）    （上传图片的方法）   1. 购物车接口（包括支付）CartController：都是普通的增删查改操作，提供修改数量。有特殊处理的地方是在添加购物车的接口。对于发来的商品id，需要先到数据库中进行查询，查询该用户的购物车中是否本来就有该商品，如果有则添加数量即可，没有才新增一条记录。     （添加商品到购物车的方法）  其中支付函数比较特殊，其中包括了删除操作和新增若干条支付记录，同时还需要发送邮件。邮件使用Spring自带的JavaMail包，需要在application.yml文件中进行配置。    （JavaMail配置）    （支付功能的方法）   1. 管理员获取统计报表。因为报表需要有打印等功能，自行实现比较复杂，这里通过整合积木报表进行实现。其中代码内容较少，主要是在积木报表的图形化界面中进行操作，进行SQL语句的查询，以及数据表格的排版，最后会生成一个基于部署服务器的url，访问该url即可得到报表。需要提醒的是报表需要在WebMvcConfig中作静态资源映射。     （静态资源映射）   1. 前端代码实现：   页面分成几个部分：1、登录注册界面 2、商城界面 3、购物车界面 4、管理界面。  前端使用了一点点vuex，是为了方便各组件共享数据，比如token这种数据，很多组件在发送axios时都会用到，因此使用vuex统一管理会比较方便。  Router的设计中为了实现某些页面需要登录后才能访问，需要登录状态的检验，我添加了一个路由守卫，如果发现是未登录状态，则弹出message框，并跳转到登录界面。  Token存在本地的localStorage中。    （路由守卫）  同时还有一个实现登录情况的验证，就是在根组件下App.vue，钩子函数onMounted中，每次刷新后都会执行该函数，该函数会携带token访问后端的check接口，如果后端检查到redis中还存在登录状态，那么就返回true。由于我在vuex中设置的登录情况是false，因此如果有，则修改状态，如果没有，则不动就行了，这样就实现了每次刷新后都能保证登录状态的检验。    （验证登录状态函数）  同时在开发环境中存在跨域问题，需要在vue.config.js中进行代理配置。    （跨域配置）   1. 登录注册界面   登录界面比较简单，实现简单的两个input框，进行v-model的双向绑定，然后点击登录后发送axios请求即可，如果登录成功，需要将Token存入vuex中，并且修改登录状态为true。  注册界面比较复杂，涉及到表单的检验，使用的组件是ElementUI中的el-form，属性rules用来绑定校验规则，需要在js中进行定义。    （html元素）    （填入信息的校验规则）  又因为这个校验只能在填入后进行校验，不是很保险，比如啥都不填他就不校验，因此还需要在点击提交后再次进行校验。因此还需要设计一个函数进行异步判断。无误后就可以进行上传。    （点击后校验）    （注册信息的提交）   1. 商城界面   商城界面也会有一个onMounted钩子函数，每次进入商城后都需要向后端发一次请求，因此将请求的函数放在钩子函数中，然后渲染。渲染时直接通过v-for循环生成div.card元素即可。    （html函数）  查看详情需要一个对话框，这里对话框的实现是使用el-dialog实现，每个对话框都绑定一个visible属性，true时就会出现，false就隐藏。其逻辑是，在点击某个商品的查看详情时，直接将该商品框的原始数据p传入到函数中，然后通过p.id向后端发送请求，前端接收到信息后存储到detailData对象中，然后将detailvisible属性设为true展示即可。    （点击详情信息后的处理函数）  关键词查询有一点特判，如果关键字为空，那么就不向后端的keyword接口发数据，而是调用类似钩子函数中的获取所有商品的接口。   1. 购物车界面   购物车界面提供了修改、删除、结算功能。同样每次进入购物车都要用onMounted钩子函数请求一次数据，另外修改购物车数量、删除购物车项目之后也需要刷新一次。  同时这里使用了ElementUI中的el-table，绑定了一个函数selection-change="handleSelectionChange"，当用户通过radio选中某一项后，会将当前所有被勾选的数据列表，作为默认参数传到handleSelectionChange函数中，然后我们就可以保存起来，在支付功能时，将该列表发送到后端即可。    （html元素）    （支付函数）   1. 管理界面   管理界面主要有几个功能：新增商品，查看商品详情，修改商品信息，删除商品，打开用户浏览/购买记录报表。  其中查看、修改都是通过el-dialog实现对话框功能，在其中设置一个表单，查看则禁止表单输入，修改则允许表单输入，同时进行部分数据的检查（类似于注册时的检查rules）    （修改信息窗口）  在点击修改信息后，将该商品信息赋值到一个updateForm对象中进行临时存储，并记录一个imgIndex改行表格的下标，用来展示图片信息。每次完成后重置更新表单并刷新管理界面。    （更新所需的一些对象和函数）  另外删除功能，删除之前弹出一个提示框是否确认删除，这里使用ElMessageBox组件，在js中进行生成，点击确认后触发删除函数。、    （确认删除框）    （删除函数）  用户浏览记录和用户购买记录点击后直接跳转到一个新页面即可，页面的URL为后端提前生成的一个积木报表网址，也是在本地的，直接进行新创页面跳转即可。  上传图片功能，使用input，type=file，可以实现文件上传，同时将文件信息作为Post请求体中的一个变量传递即可，是以二进制方式进行的。    （更新商品图片函数）   1. 功能测试：   管理员测试账号/密码：root/root  普通用户测试账号/密码：zhen/zhen   1. 输入网址 <http://www.feijizhadan.cn/> 进入商城首页。      1. 点击商城，在未登录状态下可以正常浏览，但是想查看详情或加入购物车就会提示用户未登录。      1. 点击购物车或管理，会提示未登录，并跳转到登录界面。      1. 注册一个新的账户（都是普通用户，管理员需要后台数据库手动添加） 2. 会对输入的消息进行校验，如果有误则会报错，点击提交也无效      1. 注册一个新的账号test/test，然后进行登录，提示登录成功。      1. 进行正常的商城操作，比如查看详情，添加购物车。      1. 到购物车进行移除、修改数量操作。        1. 选中商品进行支付。所绑定的邮箱会收到邮件（如果邮箱有误，后台进行了异常捕获，在前端会提示邮件发送失败）        1. 点击管理界面，普通用户会被阻拦，并跳转到商城页面。      1. 注销，登录管理员用户root/root。      1. 点击管理界面      1. 点击查看详情，可以查看商品信息。      1. 对HUAWEI 手机的商品简介进行修改。并上传一个新的图片。      1. 可以看到商城内的信息也更新成功了。      1. 在管理界面更改商品的可见。将iPhone修改成用户不可见。这样用户在商城就不能找到该商品了。      1. 新增一个商品。      1. 在修改信息界面上传一个图片。就可以在商城看到了。      1. 点击管理界面的用户浏览记录、用户购买记录，可以看到所有记录的报表，并支持打印操作。        1. 应用部署：   前端的部署主要通过手动执行命令生成dist文件，拷贝到服务器的指定位置。  后端的部署则是通过Jenkins的Maven项目自动打Jar包后拷贝到服务器执行脚本运行。  关于服务器、域名的租用就不再赘述，域名只需要进行解析对应IP地址就能使用。  前端部署：   1. 在项目的根目录下执行 npm run build 指令，会在文件夹下生成dist文件，拷贝到Linux系统的文件夹下即可（我这里放在/root/mydist/dist下，和nginx的配置对应）   部署服务器环境配置：   1. 下载并安装JDK11、Minio、Redis、MySQL（这些都已经下载好了，因为在开发环境中连接的也是服务器上的数据库和缓存），以及Nginx、Docker、Jenkins。 2. 配置Nginx的端口转发和反向代理。下载好Nginx后，找到/etc/nginx/conf.d文件夹，在其中可以自定义我们想要的Nginx配置。创建market.conf文件，在其中配置。然后 service nginx restart 重启nginx。      1. 拉取Docker官方的Jenkins镜像，并进行一系列配置和初始化（这几步都跟着网上的教程走，具体网站在参考文献中）。 2. 新建一个Maven项目，绑定对应GitHub项目的cloneUrl和网址，并在GitHub项目中创建一个WebHuk，并开启GitHub hook trigger for GITScm polling，实现push到仓库后，Jenkins能收到指令，自动拉取最近代码进行打包部署。最后配置Send files or execute commands over SSH，表示在Jenkins打包完成后，需要发送到哪个服务器上，并执行某个脚本。这里进行如下配置，发送到/root/myjar/target，并执行/root/myjar/script/deploy.sh脚本。       （脚本的意思是执行JARPATH/JARFILE文件，并删除原本运行的jar包，执行新的jar包，并将日志输出到out.log中）  这样就能完成部署了。访问域名即可。   1. 参考文献   Jenkins部署：  <https://www.51cto.com/article/715617.html>  <https://www.cnblogs.com/Chenjiabing/p/13953130.html> |
| **小结** |
| 本次实验主要是对一个网站项目前后端从代码实现到部署的全方面考验，很大的增强了我后端的代码编写能力以及结合数据库的使用，并且对前端的设计也有了很多新的认知。另外Jenkins也是之前从未涉及的领域，经过研究也终于是成功部署了一次，实现了push新的代码，就能自动拉取代码并部署到部署服务器上并运行，是一个很大的突破。总之这次实验很考验人，也让我收益良多！ |
| **指导教师评语及成绩** |
| 评语：  成绩：    指导教师签名：                                                  批阅日期： |