目录

第一章 绪论

1.1 系统的目的和意义

随着当今世界进入信息时代，Internet 的飞速发展和在全球的普及给人类生活带来革命性的改变。Internet 将传统意义上的物理空间转变成电子空间，把人们带入了一个网络社会中。成千上万的网络用户选择网上购物这种快速、便捷的购物方式。网上购物操作便易，节省时间、费用，而且一些稀有、时尚用品都能轻松在网上商城找到。网上产品的多样化、新潮、时尚等特点很符合现代人群的消费观念，这正是刺激和吸引网民在网上商城进行购物的主要原因。

我国电子商务自 1998 年开展第一笔网络交易以来，正在以前所未有的速度发展，网上购物这种新的购物方式已经开始逐渐深入到人们的日常生活，并正在为越来越多的人所喜爱。网上购物正在慢慢的影响人们的生活，同时，有越来越多的电子商务的网站出现。

网络商城正是作为电子商务有力的宣传门面和载体出现的，它所传达的是一个商城所有的商品信息，以及整个购物的过程。网站整体布局的合理性，商品信息设置的全面性以及购物流程的人性化等影响顾客群和顾客数量的因素。

针对这种现状，本次课程设计模仿并实现了 TMALL 在线，内容主要分为两个方面：

1．商城的前台设计。

2．商城的后台设计。

第二章 模仿 TMALL 商城需求分析

2.1 引言

对于构建网上购物商城的意义:

1.对消费者来说，让消费者在不出家门的情况下购买到称心如意的商品。

2.网上购物对企业来说，给他们带来更多商机的同时，也会为他们创造更大的利润

本次课程设计模仿 TMALL 构建了一个 B2C 的电商购物平台，即：全天猫没有店铺，就只有唯一一家叫做 Tmall 的商家，销售所有的东西。

目的以及实现: 模仿天猫前端，自行构建后端，通过 SSM（Spring+SpringMVC+MyBatis）框架实现并尽量降低项目的复杂度。

2.2 需求分析

SSM: SSM（Spring+SpringMVC+MyBatis）框架集由 Spring、MyBatis 两个开源框架整

合而成（SpringMVC 是 Spring 中的部分内容）。常作为数据源较简单的 web 项目的框架。

本次课程设计采用 SSM 的框架实现了对知名购物平台 TMALL 的模仿。网站可主要分为前台和后台。

2.3 前台需求分析

设计概念如下:为了简化,规定本网站为”自营”,即只有一个商家.

2.3.1 首页:显示数据库中各种相应数据

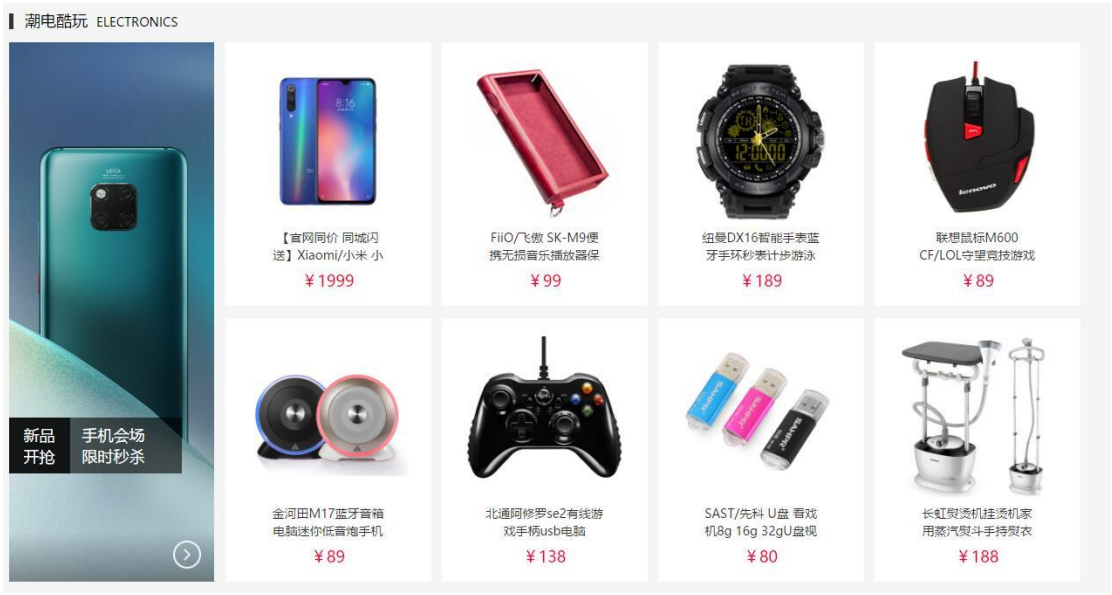
(1) 搜索栏

搜索栏需要实现搜索功能,且下方需要显示的产品可以直接定位查找到.

(2)分类导航

商品分类右边固定的两个链接【天猫超市】和【天猫国际】，还有紧跟着的八个超链，这个可以设计为一个单表，存储它显示的文字和链接过去的地址，然后是具体的 16 个分类以及轮播.

(3)产品推介

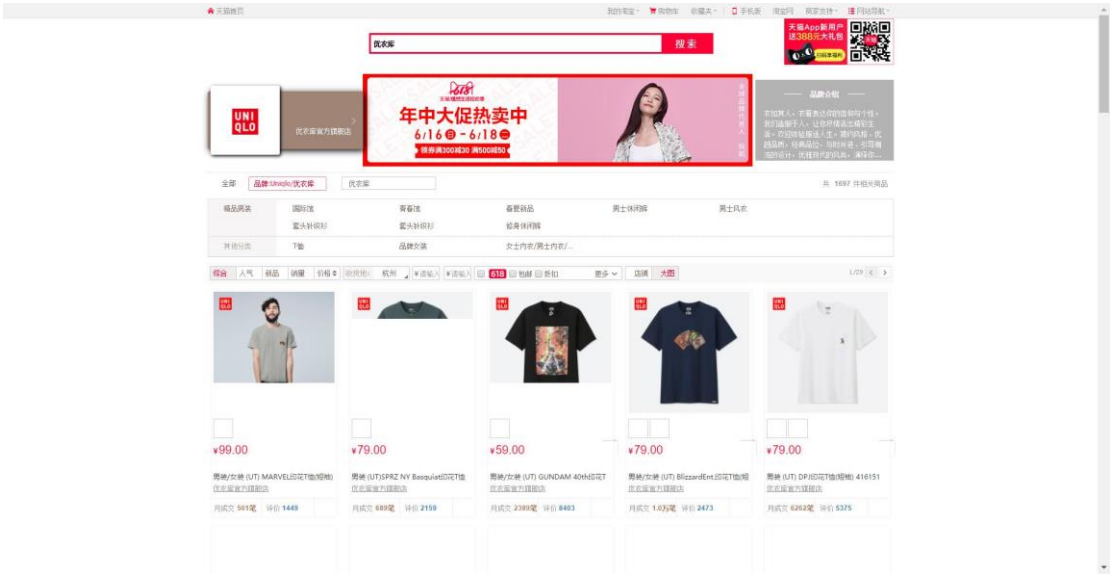


具体产品展示比较复杂，简化如下:展示几个产品比较多的固定的几个分类，其他的就直接舍弃：

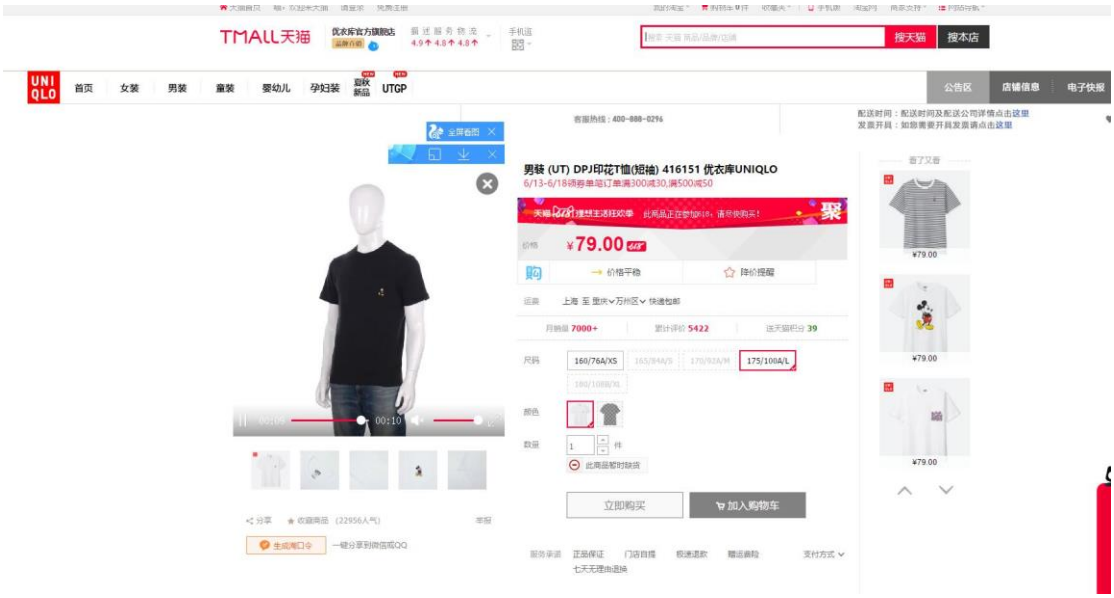
2.3.2 登录/注册页

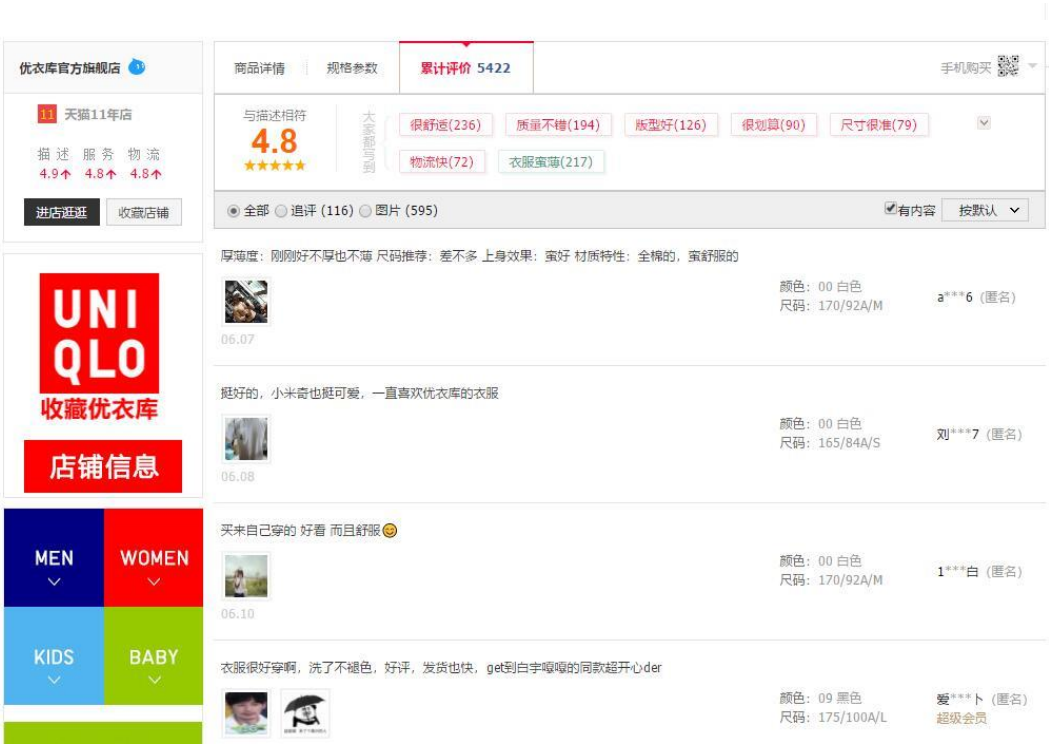
设计一个登录/注册页，能够完成用户的登录和注册功能，并能提供基础的例如判断空值等功能。这个页面尽量模仿天猫登录注册界面.

2.3.3 产品搜索页


优衣库
搜索
UNIQLO
年中大促热卖中
6/16 - 6/18
¥99.00
¥79.00
¥59.00
¥79.00
¥79.00

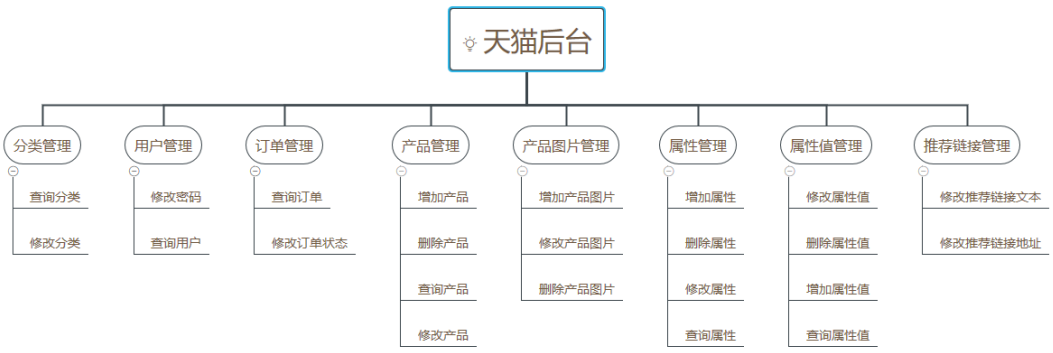
2.3.4 产品展示页


TMALL天猫
优衣库官方旗舰店
UNIQLO
首页
女装
男装
童装
婴幼儿
孕妇装
UTGP
男装 (UT) DPJ印花T恤(短袖) 416151 优衣库UNIQLO
¥79.00
160/76A/XS
175/100A/L
立即购买
加入购物车



2.4 后台需求分析

后台主要需要实现如图的功能：



第三章 仿 TMALL 系统结构分析与设计

3.1 引言

本项目主要可分为如下部分:

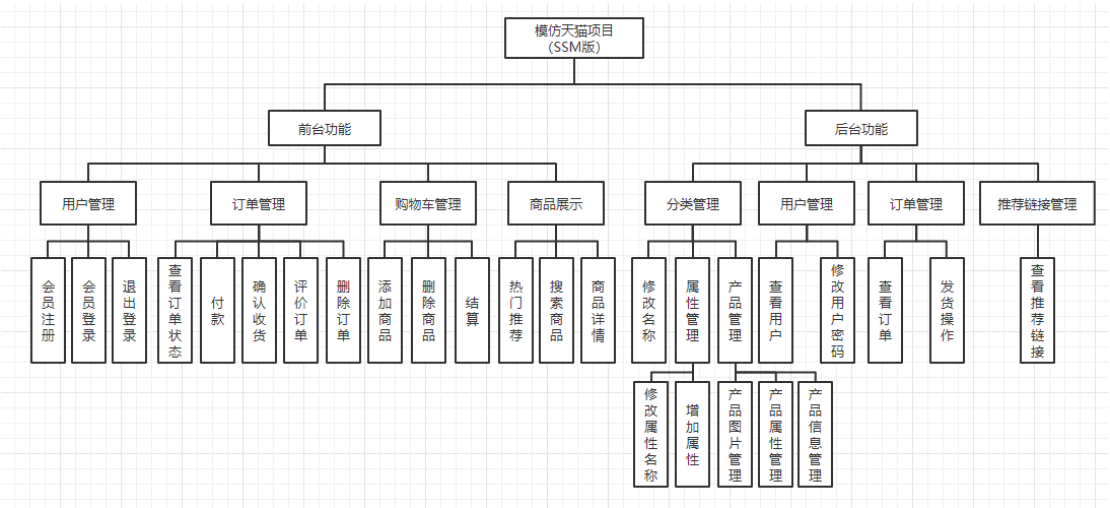
前台展示部分：首页、分类页、搜索页、产品页。首页主要显示分类和产品的内容，产品页则还包括详情图片、评论两块内容。鉴于项目较小，没有再实现一个商铺页，而是统一使用分类页。

用户部分：登录、注册，比较简单，这里实现登录用户的安全监测 Filter，用于过滤绕过登录模块直接访问的非正常访问。

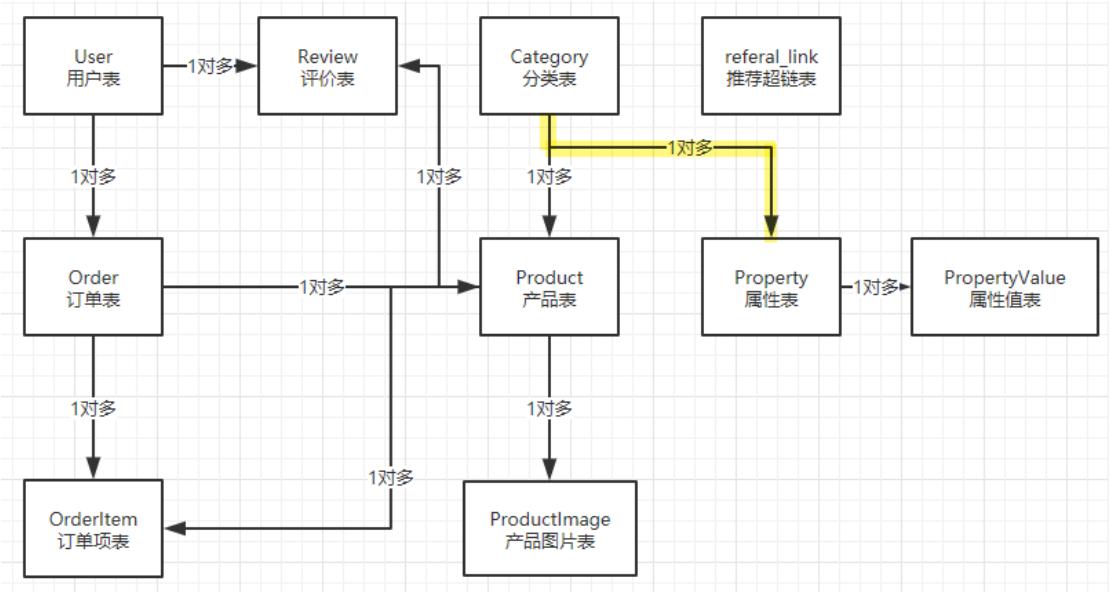
订单部分：加入购物车/直接购买 -> 下单填写地址 -> 支付 -> 发货 -> 确认收货 -> 评价 -> 我的订单。

后台：分类管理 、用户管理、订单管理、产品管理 、产品图片管理、属性管理、属性值管理、推荐链接管理。

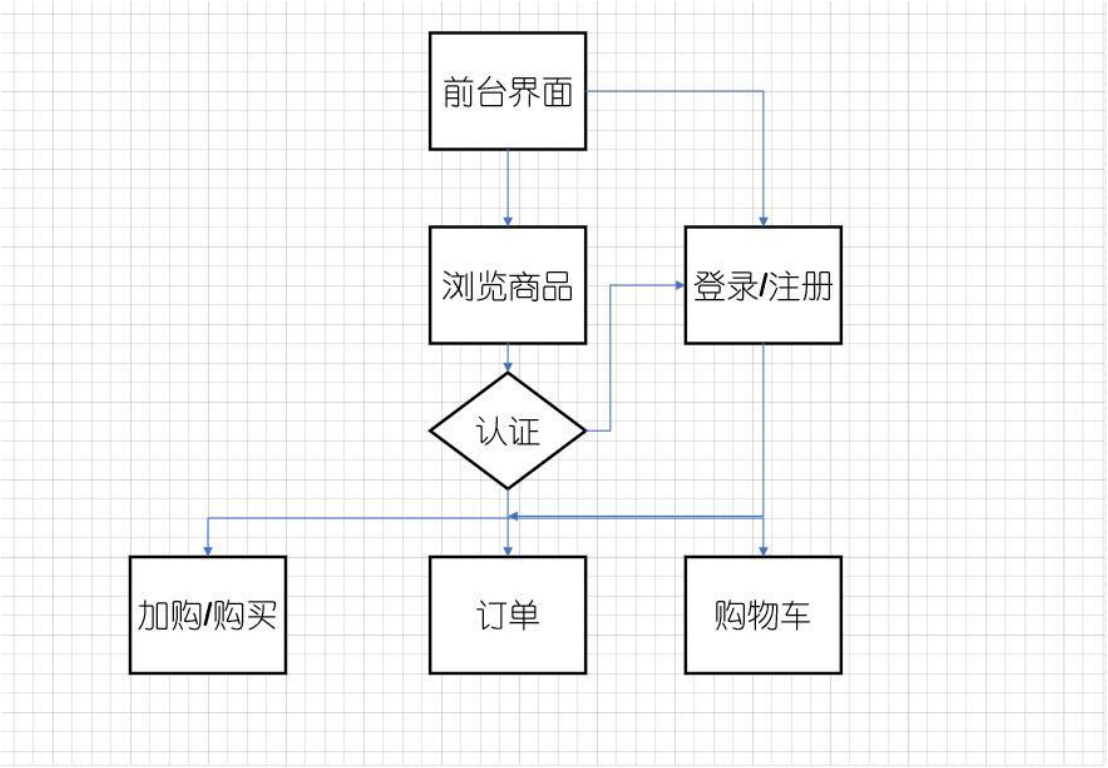
3.2 系统模块结构图



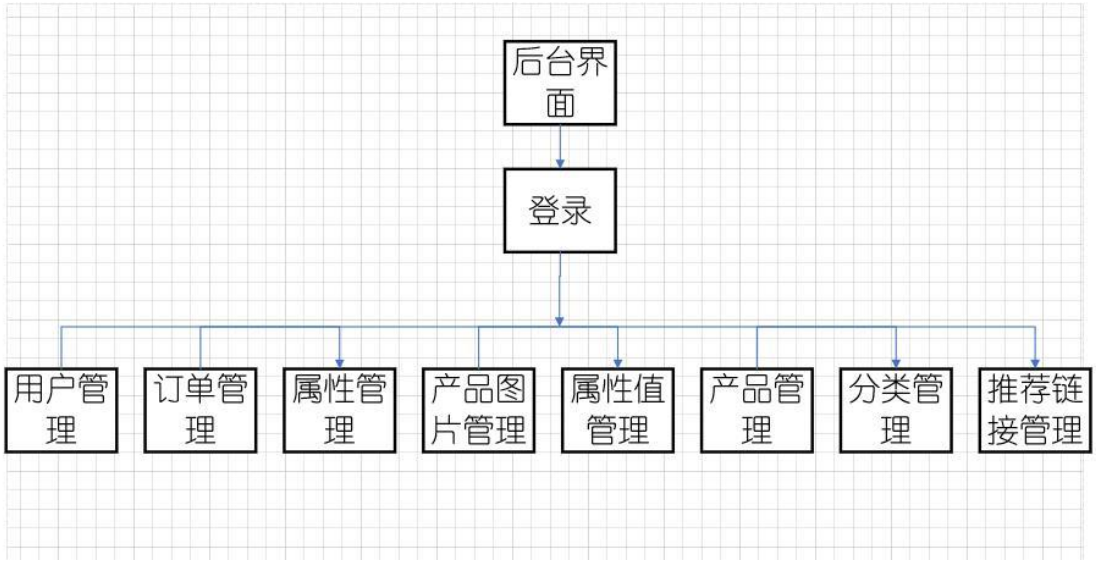
3.3 数据关系图



3.4 前台用户操作



3.5 后台操作



第四章 关键技术研究

4.1 实现技术路线

本项目运行环境为：

操作系统：Windows 10 及 Ubuntu

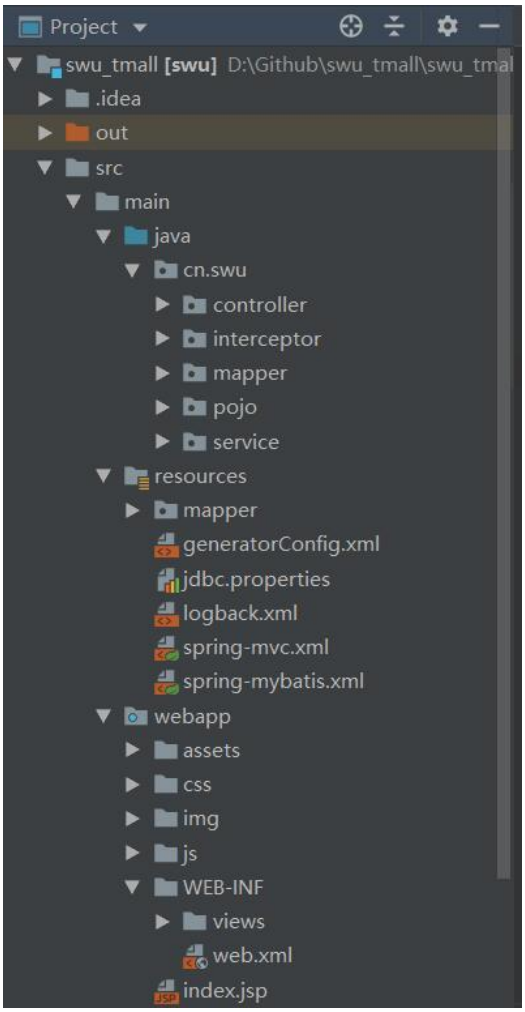
数据库：Mysql5.0 以上版本

服务器软件：Tomcat7.0 以上版本

浏览器：IE、Google Chrome 等浏览器

采用 SSM 框架结构 Spring + SpringMVC + Mybatis，主要语言为 java

4.1.1 使用 IDEA 搭建 SSM 开发环境,项目结构如下:



页面如下：

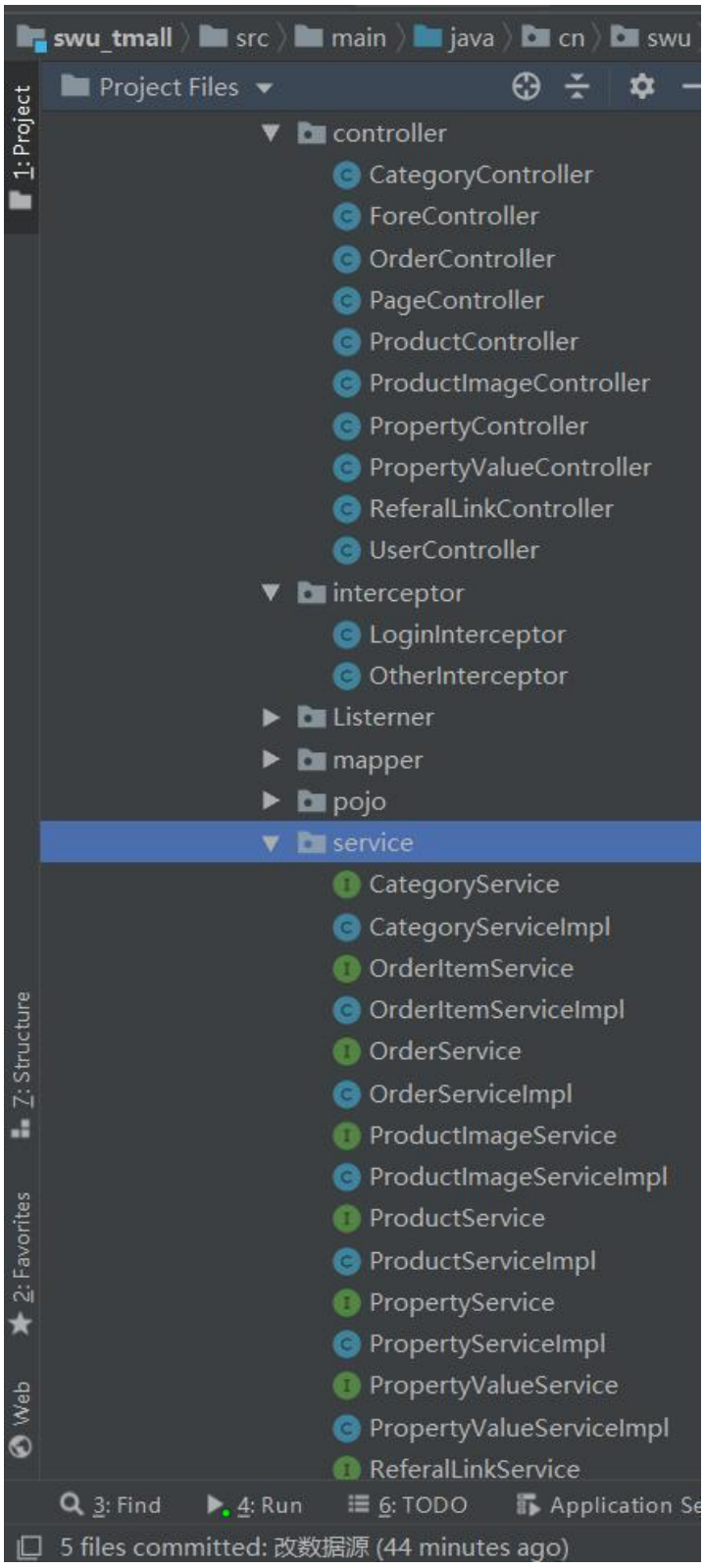
后台页面：后台所需要用到的页面，从名字很好区分功能，其中 index.jsp 只有一行代码用于跳转

公共页面：都是前端页面，从对天猫页面的分析提取出一些复用比较高的页面用于动态

的包含在其他前端页面中。

前台页面：前台相较于后台页面 CSS 更加复杂，交互也更多。

4.1.2 项目主要逻辑类



控制器（Controller）： 用于控制页面的逻辑， 提取出一个 PageController 来专门控制页面的跳转，ForeController 用于前台所有的逻辑操作

拦截器（Interceptor）： LoginInteceptor 用于对登录进行判断，因为有一些页面需要登录之后才能访问的，例如：购物车；OtherInterceptor 用于向页面中添加一些其他的数据，例如：购物车数量。

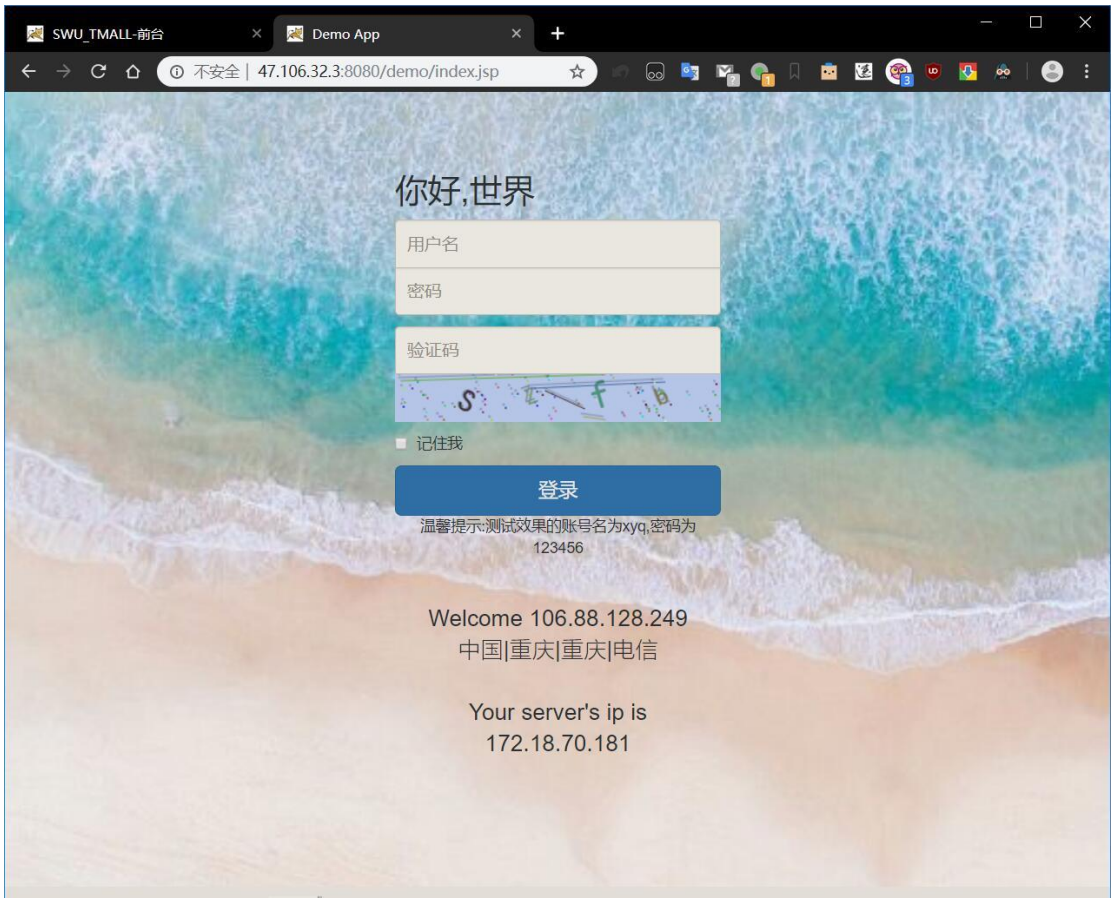
业务层（Service 层）： 业务处理层，其中封装了 Dao 层，用于完成主要的逻辑处理。

4.2 关键技术研究

以下的关键技术说明主要为考试要求的得分点以及针对所遇到的问题、难点和解决。

4.2.1 验证码识别

由于一开始的目的是为了简便的模仿天猫商城网站，因此在登录界面并没有采用验证码的识别功能，转而在管理员登陆页面采用了该验证码识别功能，分析考虑项目实际性之后又删除了管理员注册功能。



4.2.2 检测用户登录

对于电子商城而言，它的商品界面，搜索界面，主页，登录，注册都应该是可以在未登录情况下正常访问的，因此在拦截器中设置可以直接访问的界面：

```
public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Except

    HttpSession session = request.getSession();
    String[] noNeedAuthPage = new String[]{
            "/home",
            "/searchProduct",
            "/sortProduct",
            "/showProduct",
            "/loginPage",
            "/login",
            "/registerPage",
            "/register",
            "/registerSuccessPage",
            "/test",
            "/checkLogin",
            "/admin"
    };
```

4.2.3 关键技术三：内容分页

内容分页为 js 代码，其主要为下载的模板中所带的分页功能：

（1）获取当前页面索引。

```
_api_register( 'page()', function ( action ) {
    if ( action === undefined ) {
        return this.page.info().page; // not an expensive call
    }

    // else, have an action to take on all tables
    return this.iterator( flatten: 'table', type: function ( settings ) {
        _fnPageChange( settings, action );
    } );
} );
```

（2）对当前上下文中的第一个表的信息进行分页

```
_api_register( 'page.info()', function ( action ) {
    if ( this.context.length === 0 ) {
        return undefined;
    }

    var
        settings   = this.context[0],
        start      = settings._iDisplayStart,
        len        = settings._iDisplayLength,
        visRecords = settings.fnRecordsDisplay(),
        all        = len === -1;

    return {
        "page":           all ? 0 : Math.floor( x: start / len ),
        "pages":          all ? 1 : Math.ceil( x: visRecords / len ),
        "start":          start,
        "end":            settings.fnDisplayEnd(),
        "length":         len,
        "recordsTotal":   settings.fnRecordsTotal(),
        "recordsDisplay": visRecords
    };
} );
```

（3）获取当前页面长度

```
_api_register( 'page.len()', function ( len ) {
    // Note that we can't call this function 'length()' because `length`
    // is a Javascript property of functions which defines how many arguments
    // the function expects.
    if ( len === undefined ) {
        return this.context.length !== 0 ?
            this.context[0]._iDisplayLength :
            undefined;
    }

    // else, set the page length
    return this.iterator( flatten: 'table', type: function ( settings ) {
        _fnLengthChange( settings, len );
    } );
} );
```

```
var __reload = function ( settings, holdPosition, callback ) {
    if ( _fnDataSource( settings ) == 'ssp' ) {
        _fnReDraw( settings, holdPosition );
    }
    else {
        // Trigger xhr
        _fnBuildAjax( settings, data: [], fn: function( json ) {
            // xxx can this be reduced?
            _fnClearTable( settings );

            var data = _fnAjaxDataSrc( settings, json );
            for ( var i=0, ien=data.length ; i<ien ; i++ ) {
                _fnAddData( settings, data[i] );
            }

            _fnReDraw( settings, holdPosition );

            if ( callback ) {
                callback( json );
            }
        } );
    }
};
```

4.2.4 关键技术四: 在线人数统计

采用两种方式统计,session 统计,IP 统计。

（1）session 统计

```java
public class MyHttpListen implements HttpSessionListener, ServletContextListener {
    private static int count=-1;

    @Override
    public void contextInitialized(ServletContextEvent servletContextEvent) {
        servletContextEvent.getServletContext().setAttribute( s: "Count", o: 0);
        //System.out.println(count);
    }

    @Override
    public void sessionCreated(HttpSessionEvent httpSessionEvent) {
        ServletContext servletContext = httpSessionEvent.getSession().getServletContext();

        count++;
        servletContext.setAttribute( s: "Count",count);

        if((Integer) servletContext.getAttribute( s: "Count") <= 1){
            servletContext.setAttribute( s: "Count", o: 1);
        }

    }

    @Override
    public void sessionDestroyed(HttpSessionEvent httpSessionEvent) {
        ServletContext servletContext = httpSessionEvent.getSession().getServletContext();
        //count = (Integer) servletContext.getAttribute("Count");
        count--;
        if(count <= 1){
            servletContext.setAttribute( s: "Count", o: 1);
        }
        else{
            servletContext.setAttribute( s: "Count",count);
        }
        //System.out.println(count);
```

（2）IP 统计

```java
package cn.swu.Listerner;
import ...

//本机获取到两个IP,困扰很久,后来增加输出语句调试,发现有一个IP是ipv6,意识到程序并没有问题

@WebListener

public class SessionCounter implements HttpSessionListener,ServletRequestListener {
    private static final String CONTENT_TYPE = "text/html; charset=UTF-8";
    private static int activeSessions = 0;//当前活动的人数
    private HttpServletRequest request;
    private static ArrayList list = new ArrayList();//用来存放不同ip的地址


    public void init() throws ServletException {

    }

    public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOExcep
        response.setContentType(CONTENT_TYPE);
        HttpSession session = request.getSession();
    }

    public void destroy() {
    }
    @Override
    public void requestDestroyed(ServletRequestEvent servletRequestEvent) {

    }
```

```java
public void requestInitialized(ServletRequestEvent sre) { request=(HttpServletRequest)sre.getServletRequest(); }

public void sessionCreated(HttpSessionEvent httpSessionEvent) {
    String sessionId = httpSessionEvent.getSession().getId();
    Timestamp createTime = new Timestamp(System.currentTimeMillis());
    String loginIp = request.getRemoteAddr();
    boolean rs = true;
    if(list.size() > 0){
        for(int i = 0;i < list.size(); i ++){
            if(loginIp.equals(list.get(i))){
                rs = false;
            }
        }
    }
    if(rs){                     //如果队列中存在相同的IP 则SESSION不增加
        list.add(loginIp);
        System.out.println("ipList队列新增ip: "+loginIp);
        activeSessions++;
        System.out.println("新增SESSION,sessionId = " + sessionId +"; createTime = " + createTime
                + "; loginIp = " + loginIp +"; 当前总SESSION值为 "+activeSessions);
    }
}
```

```
public void sessionDestroyed(HttpSessionEvent httpSessionEvent) {
    String sessionId = httpSessionEvent.getSession().getId();
    Timestamp overTime = new Timestamp(System.currentTimeMillis());
    String loginIp = request.getRemoteAddr();
    if(activeSessions>0){
        activeSessions--;
        if(list.size() > 0){     //在用户销毁的时候,从队列中踢出这个IP
            for(int i = 0;i < list.size(); i ++){
                if(loginIp.equals(list.get(i))){
                    list.remove(i);
                    System.out.println("ipList队列移除ip: "+loginIp);
                }
            }
        }

        System.out.println("销毁SESSION,sessionId = " + sessionId +"; overTime = " + overTime
                + "; loginIp = " + loginIp +"; 当前总SESSION值为 "+activeSessions);
    }
}

public static int getActiveSessions() { return activeSessions; }

public void setActiveSessions(int i) { activeSessions = i; }
```

4.2.5 图片上传

产品图片的管理需要涉及到文件的上传操作，我们需要先提供必要的 jar 包依赖：

- commons-fileupload
- commons-io

同样的搜索 maven 库添加依赖到 pom.xml 中：

<!-- 上传文件 fileupload -->

<dependency>

```
    <groupId>commons-fileupload</groupId>
    <artifactId>commons-fileupload</artifactId>
    <version>1.3.3</version>
</dependency>
<dependency>
    <groupId>commons-io</groupId>
    <artifactId>commons-io</artifactId>
    <version>2.6</version>
</dependency>
```

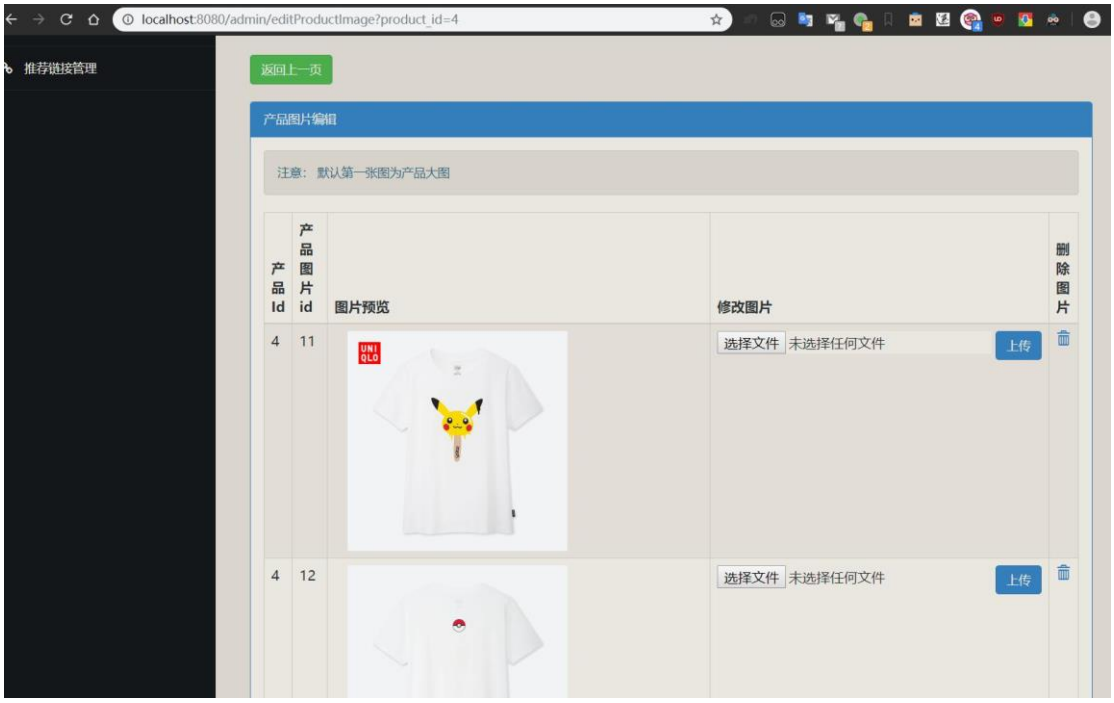
前端代码：

```
<c:forEach items="${productImages}" var="pi">
    <tr>
        <td>${pi.product_id}</td>
        <td>${pi.id}</td>
        <td><img class="col-md-8"
                 src="../img/product/${pi.product_id}/${pi.id%5==0?5:pi.id%5}.jpg"
                 onerror="this.src='../img/product/error.png'"></td>
        <td class="col-md-5">
            <form action="updateProductImage" method="post"
                  enctype="multipart/form-data">
                <input type="hidden" name="id" value="${pi.id}">
                <input type="hidden" name="product_id"
                       value="${pi.product_id}">
                <input type="file" name="picture" class="pull-left">
                <input type="submit" class="btn btn-primary pull-right" value="上传">
            </form>
        </td>
        <td>
            <a href="deleteProductImage?product_id=${pi.product_id}&id=${pi.id}"><span
                    class="glyphicon glyphicon-trash"></span></a></td>
    </tr>
</c:forEach>
```

后端代码：

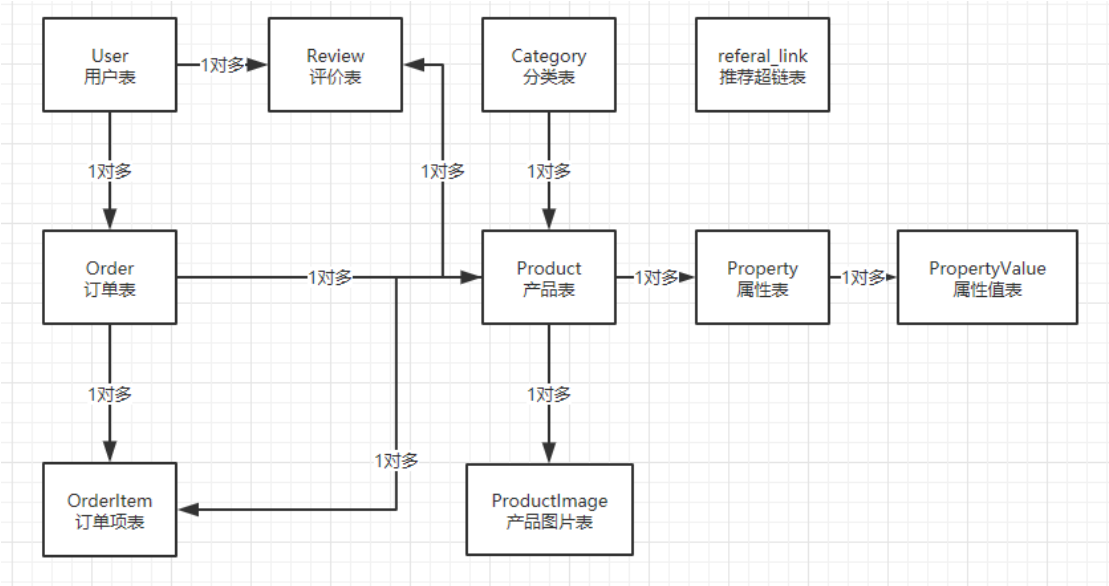
```java
@RequestMapping(value = "/updateProductImage", method = RequestMethod.POST)
public String update(HttpServletRequest request,
                     @RequestParam("productImage") ProductImage productImage,
                     Integer product_id,
                     Integer id,
                     @RequestParam("picture") MultipartFile picture) {

    // 上传文件到指定位置
    String filePath = request.getSession().getServletContext()
            .getRealPath( s: "img/product/" + product_id);
    // 因为 id 是自增长键，所以需要 % 5 来作为文件名
    String fileName = (id % 5 == 0 ? 5 : id % 5) + ".jpg";
    File uploadPicture = new File(filePath, fileName);
    if (!uploadPicture.exists()) {
        uploadPicture.mkdirs();
    }
    // 保存
    try {
        picture.transferTo(uploadPicture);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return "redirect:editProductImage?product_id=" + product_id;
}
```

界面：



4.2.6 数据库设计

数据库设计
起初设计数据库如下



(1)分类表

首先我们需要一个表来存储我们的分类信息，也就是【女装/内衣】、【男装/运动户外】在内的 16 个分类，为了高度一致，这 16 个分类不能多也不能少

为了简化任务，可以观察出，【热门手机】、【特色手机】分栏下的东西都是【手机/数码/电脑办公】类别里的东西，规定每一行显示的产品数目就可以了，这样就简单多了。

```
CREATE TABLE category (
  id int(11) NOT NULL AUTO_INCREMENT COMMENT '唯一索引 id',
  name varchar(255) NOT NULL COMMENT '分类的名字',
  PRIMARY KEY (id)
```

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

(2)商品分类右边的超链表

在【天猫国际】右边的 8 个超链，我们单独新建一个表来存储超链显示的文字和链接的地址，这样就可以任意的修改其内容：


```
CREATE TABLE referal_link (
  id int(11) NOT NULL AUTO_INCREMENT COMMENT '唯一索引 id',
  text varchar(255) NOT NULL COMMENT '超链显示的文字',
  link varchar(255) NOT NULL COMMENT '超链的地址',
  PRIMARY KEY (id)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8;
```

(3)产品表

每个分类下都要一定的产品，这些产品还有自己的一些属性，所以另外需要属性表，这个表另外创建。

产品表需要的参数：

- 用于展示的 5 张图片
- 产品名称
- 小标题（即名称下面一排标红的小字）
- 价格（就一口价，没别的）
- 销量（别月销量了，能简化就简化一下）
- 累计评价（还需要设计一个评价表）

- 库存
- 属性（需要关联另外的属性表）

```
CREATE TABLE product (
  id int(11) NOT NULL AUTO_INCREMENT COMMENT '唯一索引 id',
  name varchar(255) NOT NULL COMMENT '产品的名称',
  sub_title varchar(255) DEFAULT NULL COMMENT '小标题',
  price float DEFAULT NULL COMMENT '价格',
  sale int(11) DEFAULT NULL COMMENT '销量',
  stock int(11) DEFAULT NULL COMMENT '库存',
  category_id int(11) DEFAULT NULL COMMENT '对应的分类 id',
  PRIMARY KEY (id),
  CONSTRAINT fk_product_category FOREIGN KEY (category_id) REFERENCES category (id)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8;
```

其中产品图片，累计评价，属性都作为单独的表存在并让当前表的 id 作为外键

（4）属性表

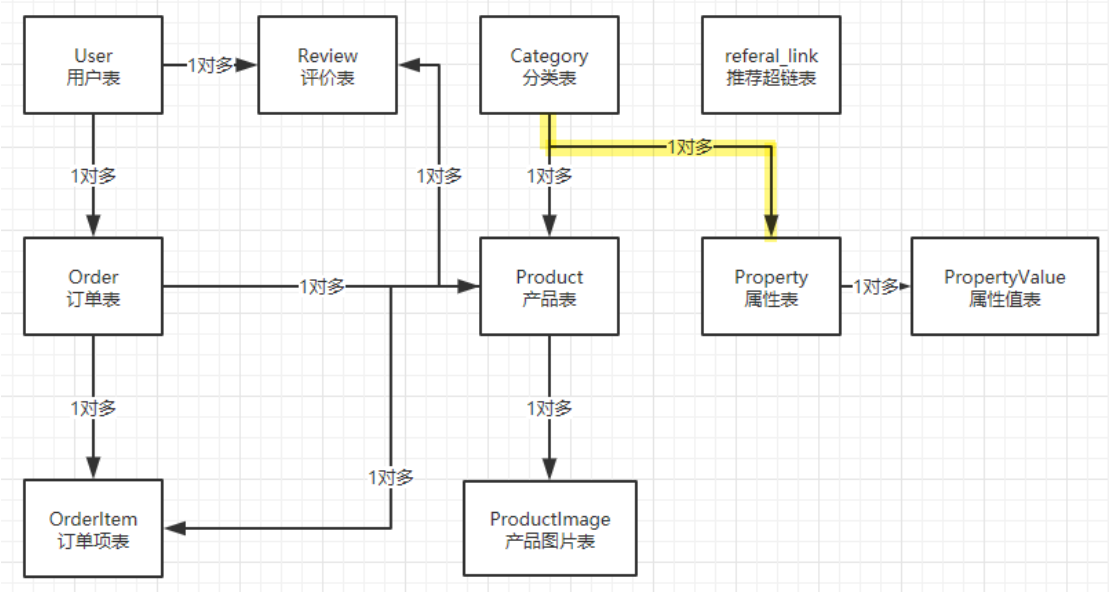

```
CREATE TABLE property (
  id int(11) NOT NULL AUTO_INCREMENT COMMENT '唯一索引 id',
  name varchar(255) DEFAULT NULL COMMENT '属性名称',
  product_id int(11) NOT NULL COMMENT '对应的产品 id',
  PRIMARY KEY (id),
  CONSTRAINT fk_property_product FOREIGN KEY (product_id) REFERENCES product(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

（5）修改数据库

问题：每一个产品的属性其实是很多的，如果每一个产品都对应很多属性的话，数据量会越来越大，不符合现实。

改进方法：

将属性表关联到 category 表上，因为其实每一个分类下的产品的属性差不多。



将属性表的外键修改为 category 的主键：

```
CREATE TABLE property (
  id int(11) NOT NULL AUTO_INCREMENT COMMENT '唯一索引 id',
  name varchar(255) DEFAULT NULL COMMENT '属性名称',
  category_id int(11) NOT NULL COMMENT '对应的分类 id',
  PRIMARY KEY (id),
  CONSTRAINT fk_property_category FOREIGN KEY (category_id) REFERENCES category(id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

（6）属性值表

其中就是保存了对应属性的值，并且应该有两个外键，一个指向 Property 表，而另一个则指向 Product 表

```
CREATE TABLE property_value (
  id int(11) NOT NULL AUTO_INCREMENT COMMENT '唯一索引 id',
  product_id int(11) NOT NULL COMMENT '对应产品 id',
  properti_id int(11) NOT NULL COMMENT '对应属性 id',
  value varchar(255) DEFAULT NULL COMMENT '具体的属性值',
  PRIMARY KEY (id),
  CONSTRAINT fk_property_value_property FOREIGN KEY (properti_id) REFERENCES property (id),
  CONSTRAINT fk_property_value_product FOREIGN KEY (product_id) REFERENCES
```

```
product (id)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8;
```

（7）产品图片表

这个表名义上是保存了产品的图片，其实只是保存了产品图片的位置即图片名称，规定所有的产品图片都放在一个统一的文件夹下面，通过把产品图片的文件命名为 id.jpg ，然后通过 id 来获取对应名称的图片：

```
CREATE TABLE product_image (
  id int(11) NOT NULL AUTO_INCREMENT COMMENT '唯一索引 id',
  product_id int(11) DEFAULT NULL COMMENT '对应的产品 id',
  PRIMARY KEY (id),
  CONSTRAINT fk_product_image_product FOREIGN KEY (product_id) REFERENCES product (id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

（8）用户表

由于前后台分开设计,所以用户表相对简单,并没有设计用户权限

```
CREATE TABLE user (
  id int(11) NOT NULL AUTO_INCREMENT COMMENT '唯一索引 id',
  name varchar(255) NOT NULL COMMENT '用户名称',
  password varchar(255) NOT NULL COMMENT '用户密码',
  PRIMARY KEY (id)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

（9）评价表

评价表对应了用户和产品两个表，也比较简单，省略回复功能和购买参数

```
CREATE TABLE review (
  id int(11) NOT NULL AUTO_INCREMENT COMMENT '唯一索引 id',
  content varchar(4000) DEFAULT NULL COMMENT '评价内容',
  user_id int(11) NOT NULL COMMENT '对应的用户 id',
  product_id int(11) NOT NULL COMMENT '对应的产品 id',
  createDate datetime DEFAULT NULL COMMENT '评价时间',
  PRIMARY KEY (id),
  CONSTRAINT fk_review_product FOREIGN KEY (product_id) REFERENCES product (id),
    CONSTRAINT fk_review_user FOREIGN KEY (user_id) REFERENCES user (id)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8;
```

（10）订单表

由于 Order 是 MySql 的一个关键字，所以我们在订单表的最后添加一个下划线：

```
CREATE TABLE order_ (
  id int(11) NOT NULL AUTO_INCREMENT COMMENT '唯一索引 id',
  order_code varchar(255) NOT NULL COMMENT '订单号',
  address varchar(255) NOT NULL COMMENT '收货地址',
  post varchar(255) NOT NULL COMMENT '邮编',
  receiver varchar(255) NOT NULL COMMENT '收货人姓名',
  mobile varchar(255) NOT NULL COMMENT '手机号码',
  user_message varchar(255) NOT NULL COMMENT '用户备注的信息',
```

```
  create_date datetime NOT NULL COMMENT '订单创建时间',
  pay_date datetime DEFAULT NULL COMMENT '订单支付时间',
  delivery_date datetime DEFAULT NULL COMMENT '发货日期',
  confirm_date datetime DEFAULT NULL COMMENT '确认收货日期',
  user_id int(11) DEFAULT NULL COMMENT '对应的用户 id',
  status varchar(255) NOT NULL COMMENT '订单状态',
  PRIMARY KEY (id),
  CONSTRAINT fk_order_user FOREIGN KEY (user_id) REFERENCES user (id)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8;
```

（11）订单项表

一个订单里面可能有多个订单项，一个产品也可能对应多个订单项，所以这个表应该有两个外键：

```
CREATE TABLE order_item (
  id int(11) NOT NULL AUTO_INCREMENT COMMENT '唯一索引 id',
  product_id int(11) NOT NULL COMMENT '对应产品 id',
  order_id int(11) NOT NULL COMMENT '对应订单 id',
  number int(11) DEFAULT NULL COMMENT '对应产品购买的数量',
  PRIMARY KEY (id) COMMENT '邮编',
  CONSTRAINT fk_order_item_product FOREIGN KEY (product_id) REFERENCES product (id),
  CONSTRAINT fk_order_item_order FOREIGN KEY (order_id) REFERENCES order_ (id)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8;
```

第五章 系统实现

5.1 系统实现介绍

5.1.1 前台

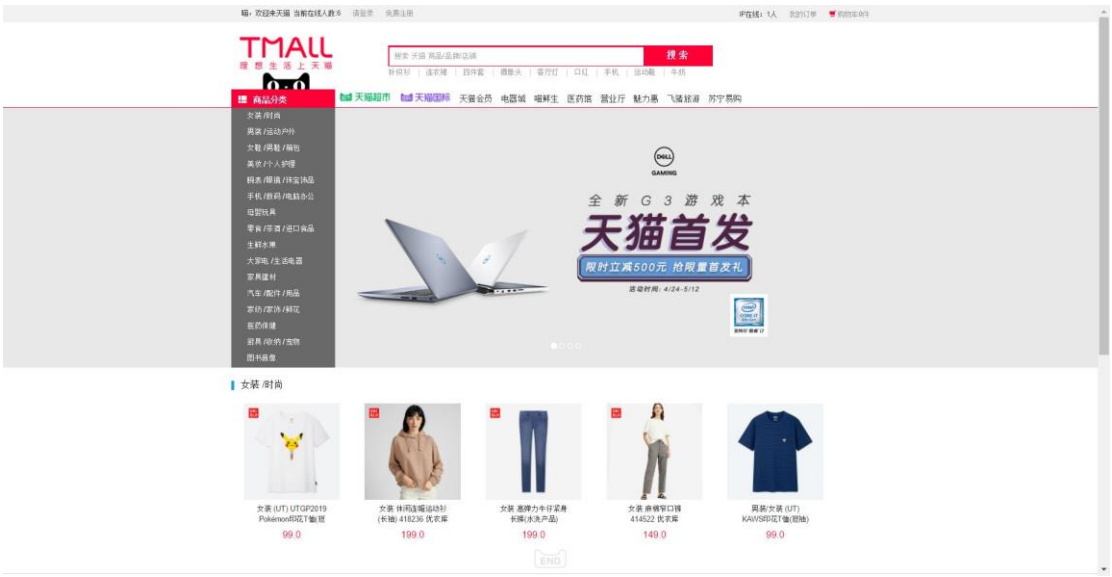
登录页


TMALL天猫
品牌盛宴
超值精品大聚会!
密码登录
登 录
品质保障
七天无理由退换货
特色服务体验
帮助中心
购物指南
天猫保障
支付方式
商家服务
手机天猫

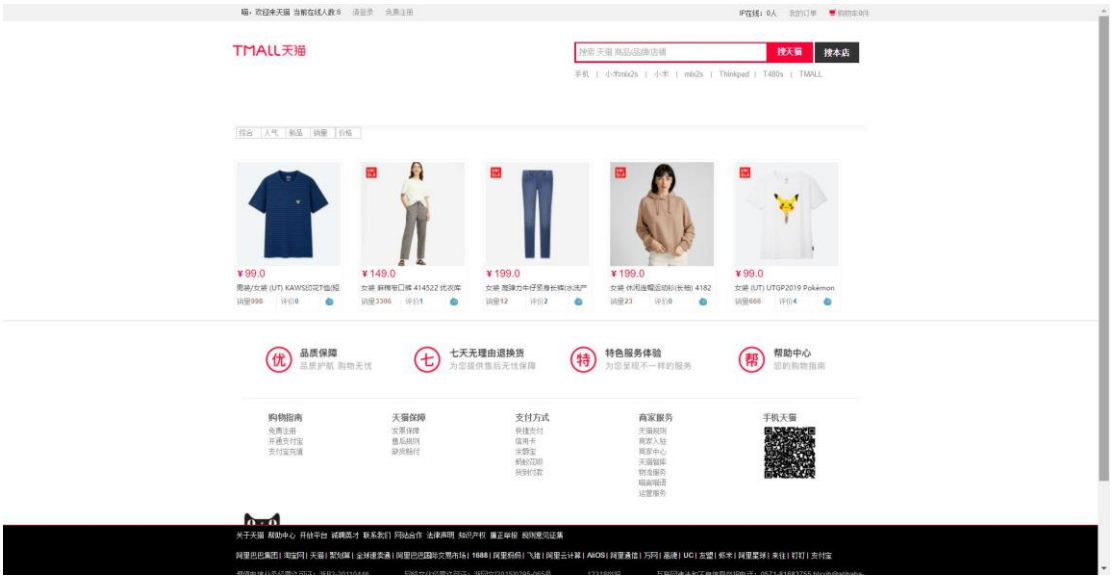
注册页


用户注册
用户账号
用户密码
确认密码
确认注册

主界面：

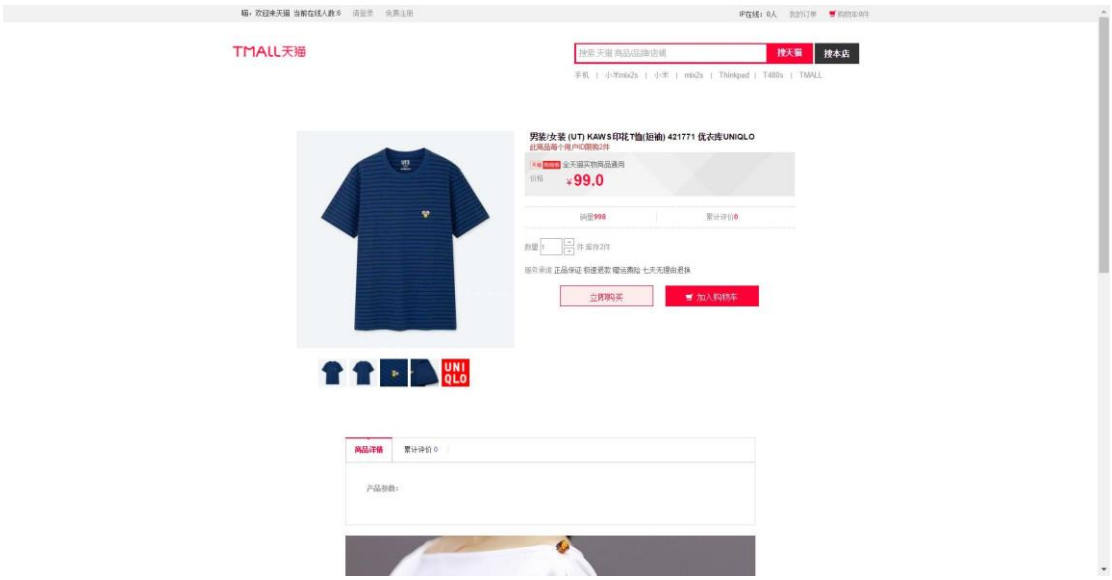


搜索结果展示页：



支持按照【综合（销量*评价）】、【人气（评论量）】、【销量】、【价格】来排序产品。注意只是支持顺序（从小到大）排列。

产品展示页


TMALL天猫
¥99.0
商品详情
累计评价 0
产品参数：

购买页


天猫 TMALL.COM
输入收货地址
确认订单信息
¥199.0
提交订单
品质保障
七天无理由退换货
特色服务体验
帮助中心

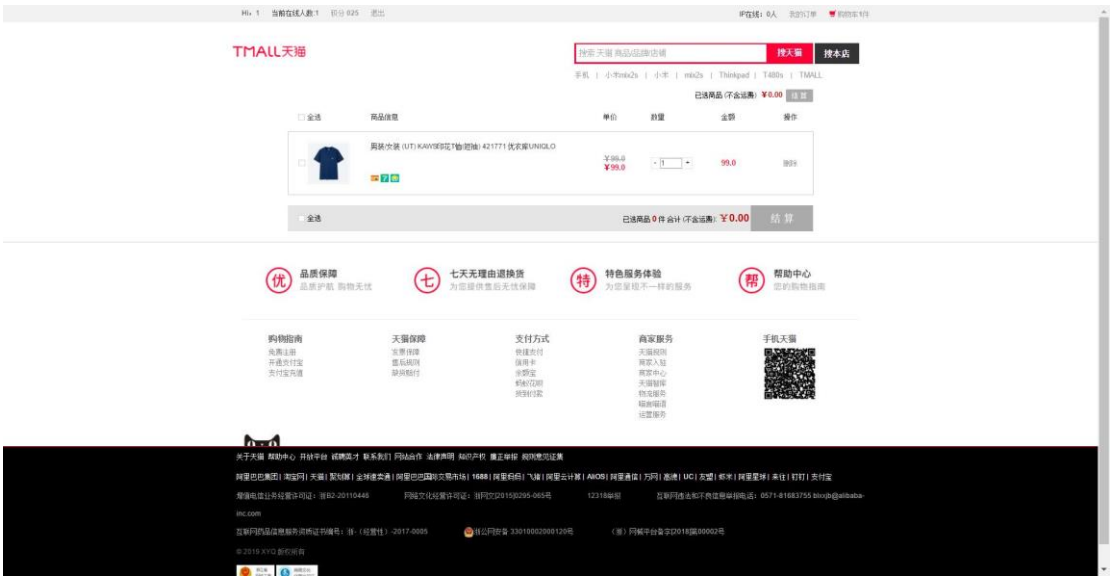
付款页面


天猫 TMALL.COM
扫一扫付款（元）
¥199.0
支付宝
ALIPAY
支付就用支付宝
打开支付宝扫一扫
确认支付

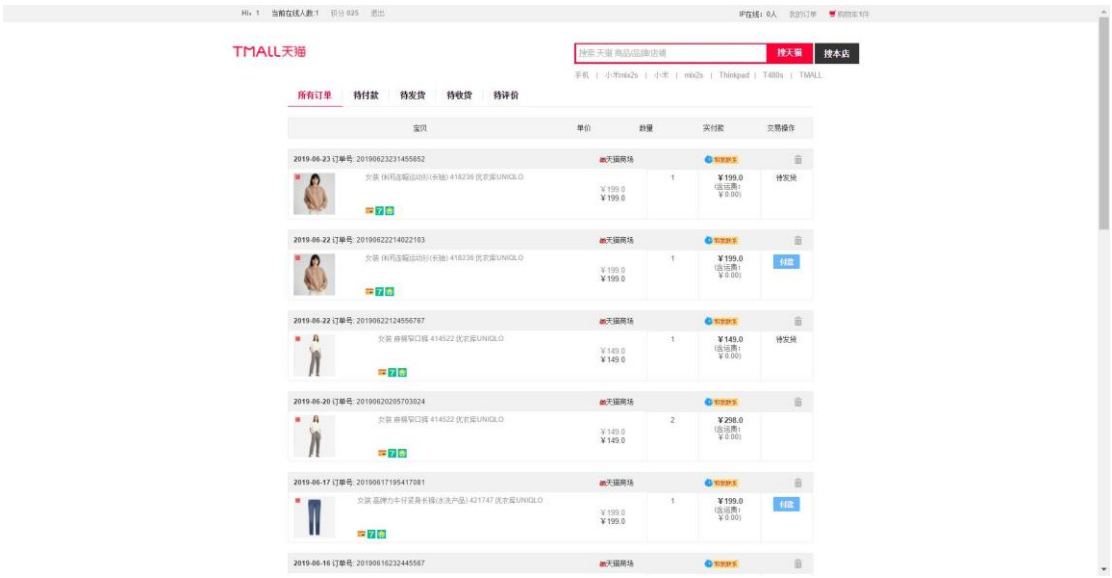
付款成功页


TMALL天猫
搜天猫
搜本店
您已成功付款

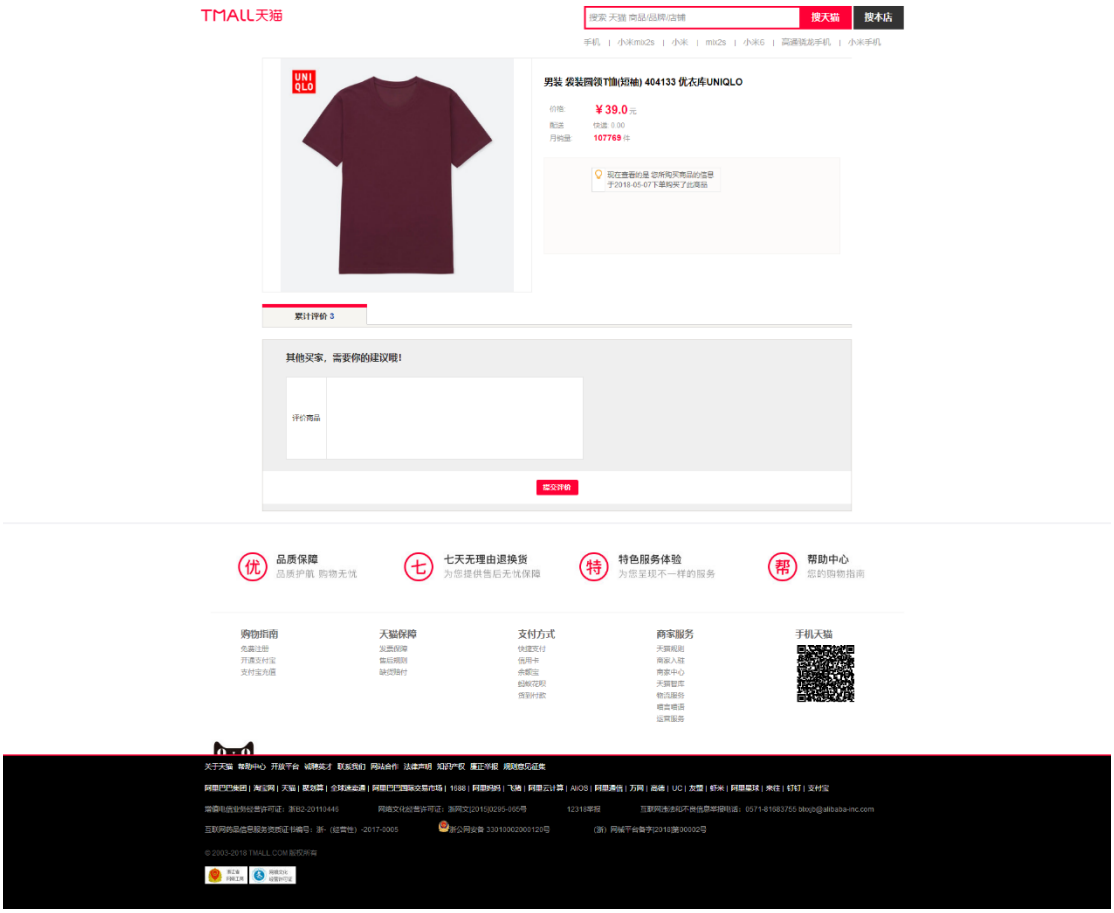
购物车页面



我的订单页

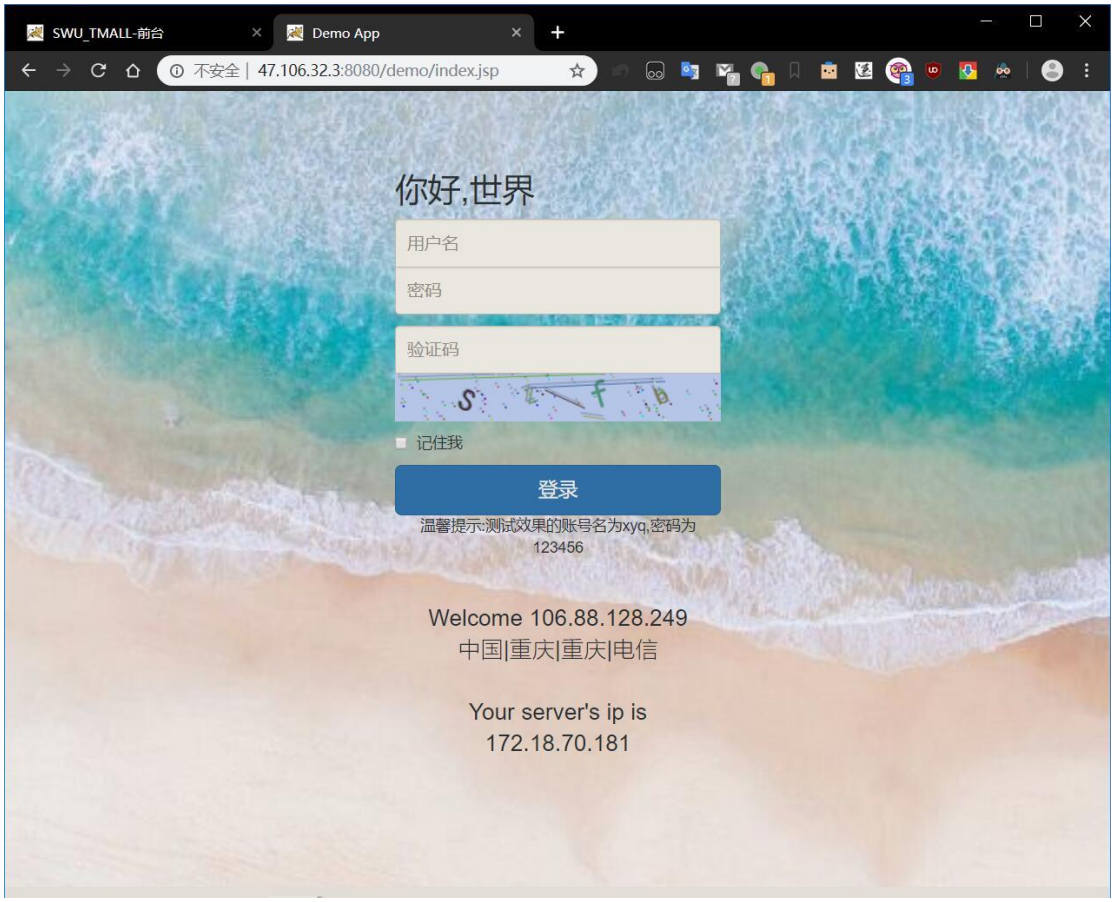


评价页

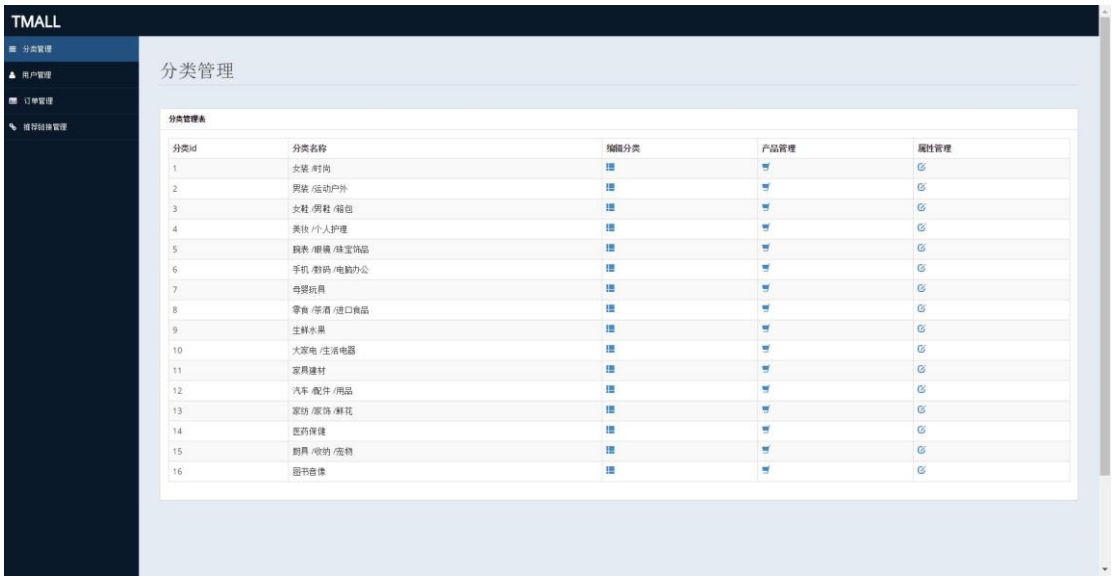

TMALL天猫
搜天猫
搜本店
男装 袋装圆领T恤(短袖) 404133 优衣库UNIQLO
¥ 39.0
累计评价 3
其他买家，需要你的建议哦！
提交评价
品质保障
七天无理由退换货
特色服务体验
帮助中心
购物指南
天猫保障
支付方式
商家服务
手机天猫

5.1.2 后台

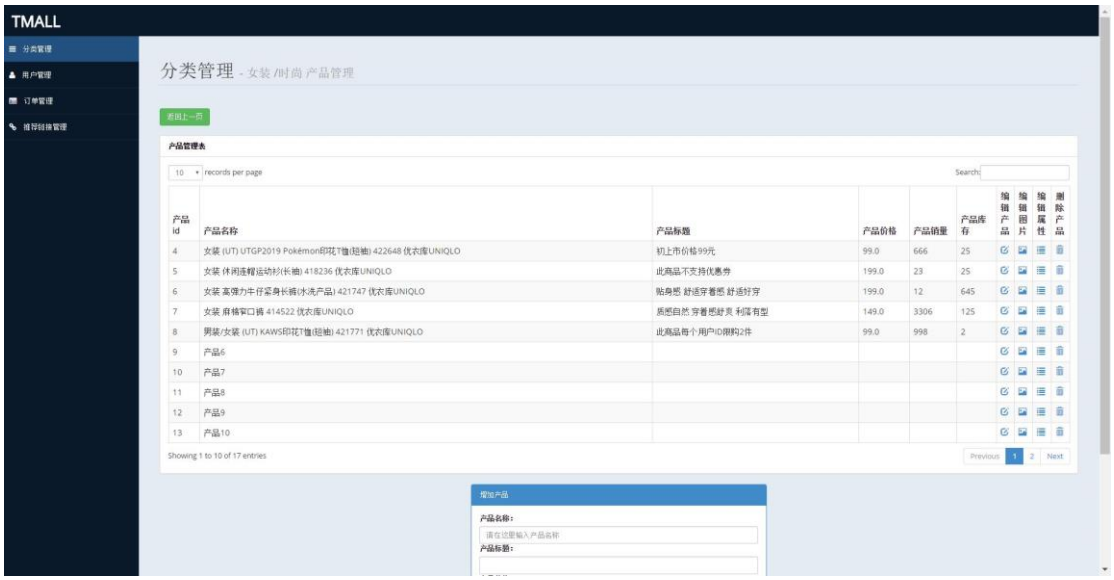
登录:



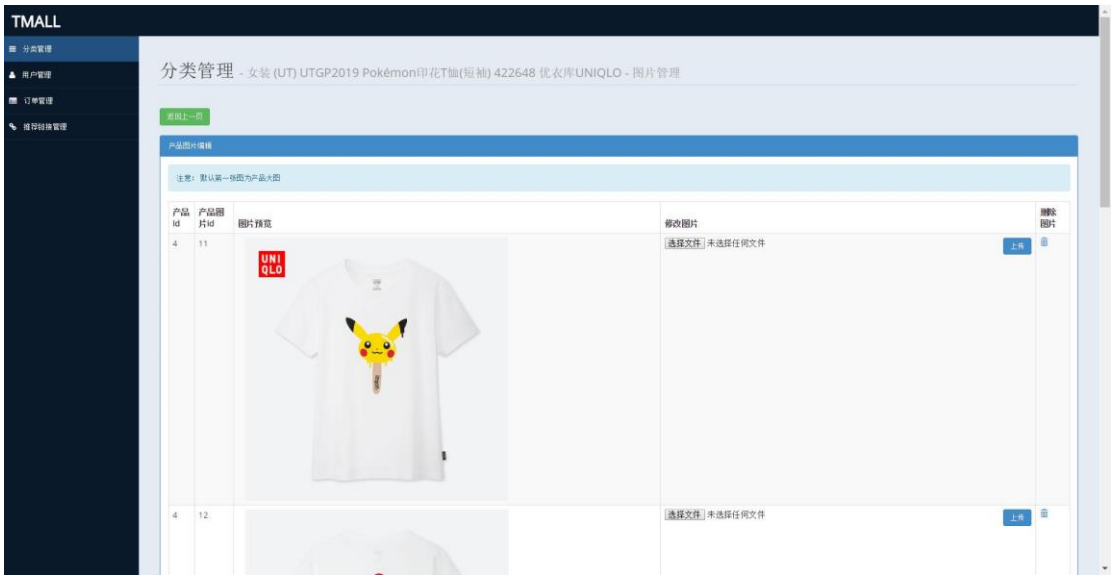
分类管理



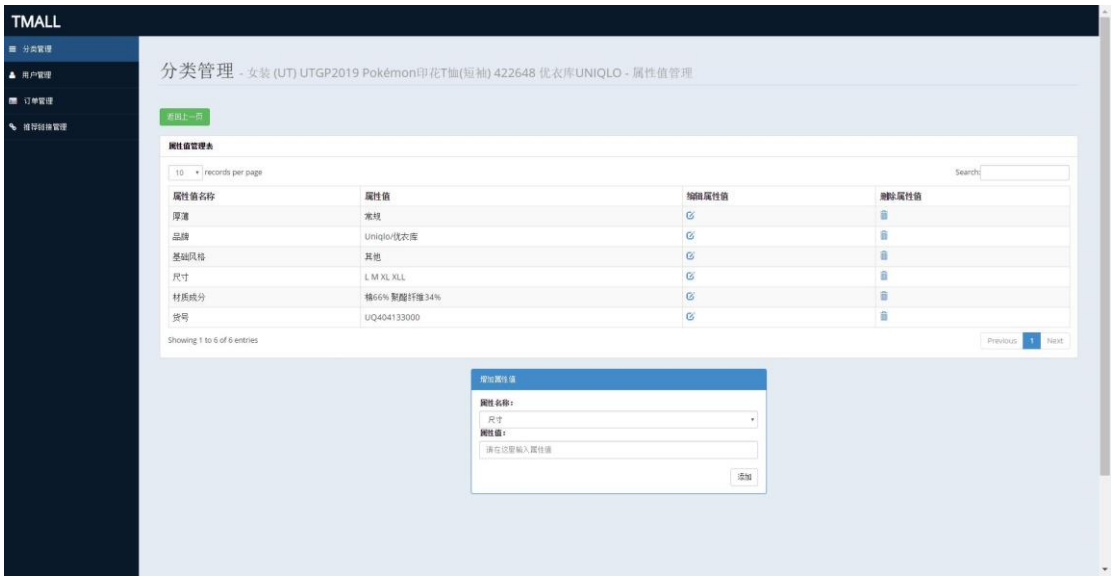
3.2 产品管理页



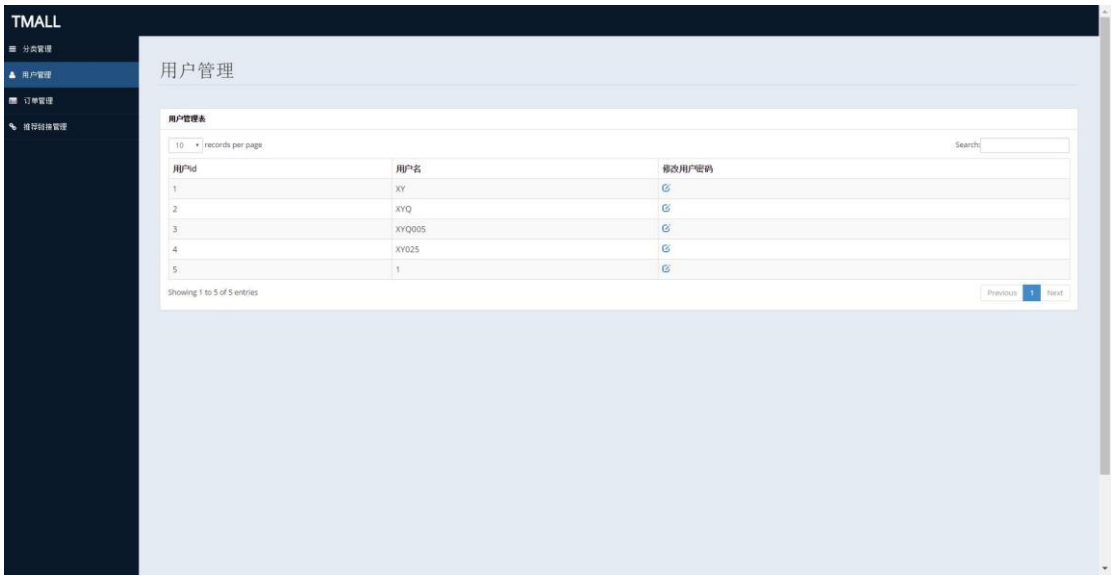
3.3 产品图片管理页



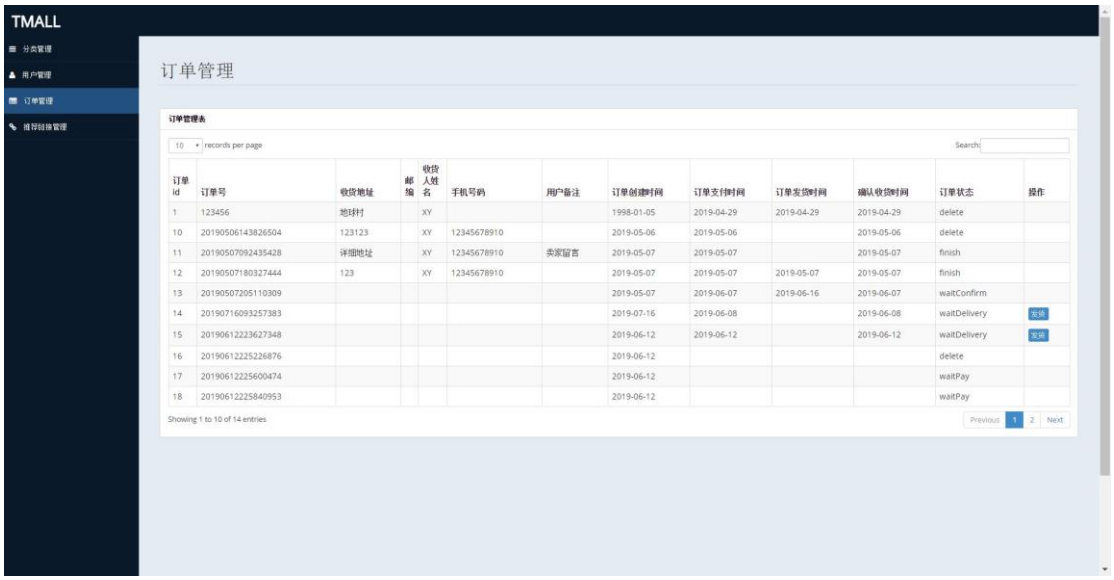
3.3 属性值管理页



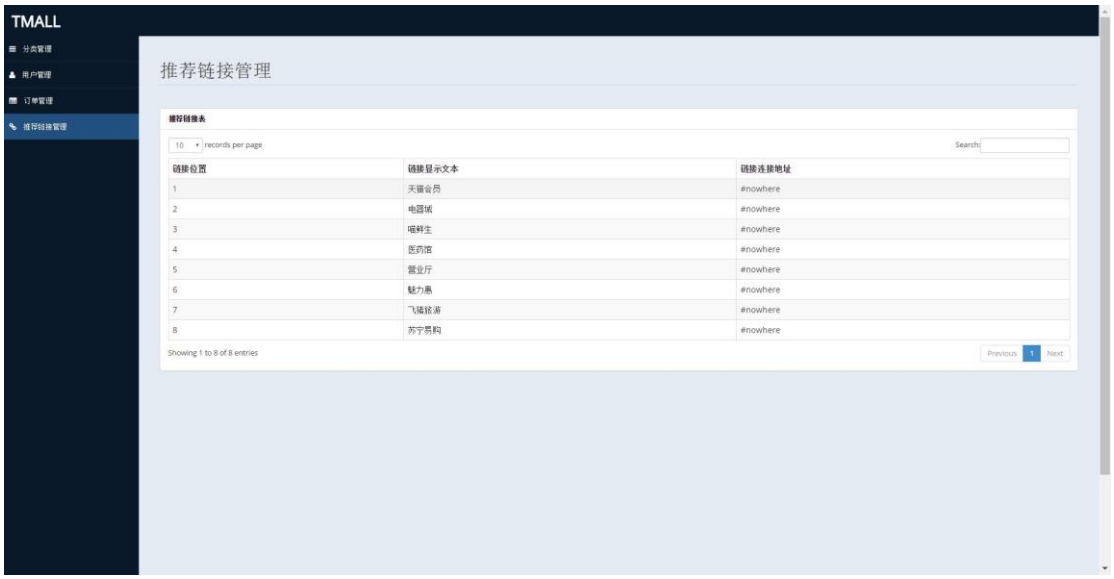
3.4 用户管理



3.5 订单管理页



3.6 推荐链接管理



5.2 系统实现的不足

1.在线人数统计存在一定 bug，采用 session 和 IP 统计也不准确，IP 有时候会显示为 0，但是代码是没有问题的，测试也只能在服务器上测试，本地对于 IP 在线人数统计的测试存在限制。

2.只有文件（图片）的上传功能，未设置下载功能。

3.关于过滤，使用的是 session 和集合 map 以及基于 url 拦截的方式实现过滤。

4.对于数据库用户的设计并没有设计用户权限，而是将后台与前台分开，为了简便，后台并没有设置管理员，后台的登陆界面采用的是以前写的不舍得扔的另一个项目的登陆界

面，这样其实可以直接对后台地址进行访问，当然也可以设置服务器使其不显示真实路径，避免直接访问。想要设置全局 filter 控制多个项目以此实现过滤，没有成功。

5.对于网页界面，未适配移动端，移动端显示不太正常。

6.整个网站存在很多空链接，其实也算是不足，不过也可以算作一个为以后增加功能，或者说在其基础上直接改项目的优势条件。

7.使用了 maven，暂时不能用 ant。