# 项目介绍

* 1. 项目描述简介

通过完成一个仿饿了么的在线订餐平台，让大家掌握企业开发特点，以及解决问题的能力。

该项目会涉及以后工作中的一些非常有代表性的功能。

用户端采用前后端分离开发模式，前端通过调用后端API接口来获取和提交数据。掌握项目中前后端分离的开发模式。

我们使用git作为项目的版本控制软件。

在项目中会使用很多其他的知识，比如系统架构、维护、优化等等。

* 1. 主要功能模块

用户模块,商品模块,购物车模块,订单模块

* 1. 开发环境和技术

|  |  |
| --- | --- |
| 开发环境 | Window |
| 开发工具 |  |
| 相关技术 |  |

* 1. 项目人员组成周期成本
     1. 人员组成

|  |  |  |
| --- | --- | --- |
| 职位 | 人数 | 备注 |
| 项目经理和组长 | 1 | 一般小公司由项目经理负责管理，中大型公司项目由项目经理或组长负责管理 |
| 开发人员 | 2~3 |  |
| UI设计人员 | 1 |  |
| 前端开发人员 | 1 | 专业前端不是必须的，所以前端开发和UI设计人员可以同一个人 |
| 测试人员 | 1~3 | 有些公司并未有专门的测试人员，测试人员可能由开发人员完成测试。  公司有测试部，测试部负责所有项目的测试。  项目测试由产品经理进行业务测试。 |

* + 1. 项目周期成本

|  |  |  |
| --- | --- | --- |
| 人数 | 周期 | 备注 |
| 3 | 四个月  第一个月需求设计  第二三个月编码实现  第四个月编码实现、测试 |  |

1. 用户注册功能模块
   1. 需求

用户按照页面要求输入数据并提交,输入接收到的验证码,对相关数据进行验证,通过,注册成功,跳转登录页面,失败,返回注册页面,提示错误信息

* 1. 流程

1. 用户按照页面要求输入数据并提交
2. 对数据进行判断,判断通过,保存数据到数据库,并跳转到登录页面
3. 判断不通过,返回注册页面,并提示错误信息
   1. 设计要点（数据库和页面交互）

Phone 手机号

Password 密码

Username 用户名

Head\_portrait 头像

Sex 性别

School 学校

Address 地址

Hometown 故乡

Is\_delete 删除

Create\_time 创建时间

* 1. 要点难点及解决方案

验证码:

1. # 发送手机验证码
2. def send\_phone\_code(request):
3. """
4. :param request: HttpRequest 请求对象
5. :param phone: 手机号码
6. :return: 返回结果
7. """
8. try:
9. # 获取手机号码
10. phone = request.GET.get('phone')
11. # 验证手机号是否正确
12. phone\_re = re.compile('^1[3-9]\d{9}$')
13. res = re.search(phone\_re, phone)
14. if res:
15. # 生成随机验证码
16. code = "".join([str(random.randint(0, 9)) for \_ in range(4)])
17. print(code)
18. print("===========================")
19. # 保存到redis中 ,等你验证的时候使用
20. r = get\_redis\_connection('default')
21. r.set(tel, code)
22. # 设置过期时间 redis
23. r.expire(tel, 120)
24. # 发送短信验证码
25. \_\_business\_id = uuid.uuid1()
26. # 信息
27. params = "{\"code\":\"%s\"}" % code
28. rs = send\_sms(\_\_business\_id, phone, "模板名称", "SMS\_141905190", params)
29. print(rs.decode('utf-8'))
30. return {'ok': 1, 'code': 200}
31. else:
32. return {'ok': 0, 'code': 500, 'msg': '手机号码格式错误！'}
33. except:
34. return {'ok':0,'code':500,'msg':'短信验证码发送失败'}
35. # 发送短信
36. def send\_sms(business\_id, phone\_numbers, sign\_name, template\_code, template\_param=None):
37. # 注意：不要更改
38. REGION = "cn-hangzhou"
39. PRODUCT\_NAME = "Dysmsapi"
40. DOMAIN = "dysmsapi.aliyuncs.com"
41. # acs\_client = AcsClient(const.ACCESS\_KEY\_ID, const.ACCESS\_KEY\_SECRET, REGION)
42. acs\_client = AcsClient(settings.ACCESSKEYID, settings.ACCESSKEYSECRET, REGION)
43. region\_provider.add\_endpoint(PRODUCT\_NAME, REGION, DOMAIN)
44. smsRequest = SendSmsRequest.SendSmsRequest()
45. # 申请的短信模板编码,必填
46. smsRequest.set\_TemplateCode(template\_code)
47. # 短信模板变量参数
48. if template\_param is not None:
49. smsRequest.set\_TemplateParam(template\_param)
50. # 设置业务请求流水号，必填。
51. smsRequest.set\_OutId(business\_id)
52. # 短信签名
53. smsRequest.set\_SignName(sign\_name)
54. # 数据提交方式
55. # smsRequest.set\_method(MT.POST)
56. # 数据提交格式
57. # smsRequest.set\_accept\_format(FT.JSON)
58. # 短信发送的号码列表，必填。
59. smsRequest.set\_PhoneNumbers(phone\_numbers)
60. # 调用短信发送接口，返回json
61. smsResponse = acs\_client.do\_action\_with\_exception(smsRequest)
62. # TODO 业务处理
63. return smsResponse

创建视图函数并绑定路由

1. # 短信发送 视图函数
2. def sendMsg(request):
3. return JsonResponse(send\_phone\_code(request))
4. # 8. 路由绑定
5. urlpatterns = [
6. url(r'^sendMsg/$', sendMsg, name='sendMsg'), # 发短信
7. ]

Ajax点击按钮js代码

1. {% block footer\_js %}
2. <script>
3. $(function () {
4. // 验证码按钮上绑定点击事件
5. $(".yzm-hq").on('click', function () {
6. // 由于this会在很多匿名函数中使用，先将其保存在一个变量中
7. // 因为其他匿名函数中有自己的this,会和点击事件的this冲突
8. var that = this;
9. // 获取手机号码
10. phone = $("input[name='phone']").val();
11. // 通过ajax发送get请求
12. $.get('{% url "sp\_user:sendMsg" %}', {'phone': phone}, function (data) {
13. //判断是否请求成功
14. if (data.ok == 1) {
15. // 设定 60 秒后可以重新发送验证码
16. var time = 60;
17. // 立马 将按钮禁用 防止重复点击提交
18. $(that).attr('disabled', true);
19. // 颜色设置为 灰色
20. $(that).css({'backgroundColor': "gray"});
21. // 将按钮提示文字改变
22. showMsg(that, time);
23. // 设置周期性执行， 周期性改变显示文字内容
24. var timer = window.setInterval(function () {
25. // 时间减少
26. --time;
27. // 显示文字
28. showMsg(that, time);
29. // 判断是否到期
30. if (time == 0) {
31. // 到期后清除定时器
32. window.clearInterval(timer);
33. // 启用按钮
34. $(that).attr('disabled', false);
35. // 改变颜色和文字内容
36. $(that).css({'backgroundColor': "#76bb2a"});
37. $(that).val("获取验证码");
38. }
39. }, 1000)
40. } else {
41. alert(data.msg);
42. }
43. })
44. });
45. //改变按钮 倒计时显示内容
46. function showMsg(obj, time) {
47. var msg = time + "秒后重新发送!";
48. $(obj).val(msg);
49. }
50. })
51. </script>
52. {% endblock %}

在form表单中创建一个方法验证验证码

· verify\_code = forms.CharField(error\_messages={"required": "请填写验证"})

· · · # 单独使用一个方法校验 验证码

· def clean\_verify\_code(self):

· # 验证验证码是否填写正确

· # 获取redis中的验证码

· r = get\_redis\_connection()

· tel = self.cleaned\_data.get('phone')

· s\_verify\_code = r.get(tel)

· if not s\_verify\_code:

· raise forms.ValidationError("验证码已经过期")

· # 表单传入的验证码

· verify\_code = self.cleaned\_data.get('verify\_code')

· # sid\_verify\_code = self.data.get('sid\_verify\_code')

· if int(verify\_code) != int(s\_verify\_code):

· raise forms.ValidationError("验证码输入有误")

· return verify\_code

建表时报错 django.db.utils.InternalError: (1366, "Incorrect string value:...

编码问题,重新创建数据库 create database 库名 default charset=utf8;

1. 用户登录功能模块
   1. 需求

用户来到登录界面,按照提示输入数据,通过,跳转到商品首页,不通过,返回登录页

* 1. 流程

1. 用户按照提示输入数据
2. 获取数据,对数据进行判断,判断通过,跳转到商品首页
3. 判断失败,返回至登录页面并提示错误信息
   1. 设计要点（数据库和页面交互）

Phone 手机号

Password 密码

Username 用户名

Head\_portrait 头像

Sex 性别

School 学校

Address 地址

Hometown 故乡

Is\_delete 删除

Create\_time 创建时间

* 1. 要点难点及解决方案

密码需要加密

通过哈希算法加密

# 4.用户详细信息及修改模块

4.1需求

显示用户的详细信息

4.2流程

用户打开详细信息页面,展示用户的详细信息,并修改用户修改的信息

4.3设计要点

Phone 手机号

Password 密码

Username 用户名

Head\_portrait 头像

Sex 性别

School 学校

Address 地址

Hometown 故乡

Is\_delete 删除

Create\_time 创建时间

4.4 难点及解决方案

获取用户信息:

使用session来获取用户的唯一信息，再使用唯一信息进行数据查询，将查询到的数据渲染到页面上进行显示。

用户头像: 在settings中设置上传图片路径

MEDIA\_URL = ‘static/media/’

MEDIA\_ROOT = os.path.join(BASE\_DIR,’/static/media’

设置头像字段属性 upload\_to=’head/%Y%m’ ,设置默认头像default = ‘head/图片’

5.商品详细信息模块

5.1需求

* + - 1. 点击某个商品后，会跳转到该商品的详情页面；
      2. 该详情页面会渲染出商品的基本信息，商品图片，商品名称，价格，单位，详情。

5.2流程

用户点击商品,跳转到详细信息页面

渲染页面,

5.3设计要点

* + - 1. 商品分类表

Classification\_name 分类名 CharField

Introduction 简介 TextField

Add\_time 添加时间 DateTimeField

Update\_time 修改时间 DateTimeField

Is\_delete 删除 BooleanField

2.商品单位表

Conversion\_name 单位名 CharField

Add\_time 添加日期 DateTimeField

Upda\_time 修改日期 DateTimeField

Is\_delete 删除 BooleanField

3.商品SKU表

Commodity\_name 商品名 CharField

Introduce 简介 TextField

Price 价格 DecimalField

Conversion 单位 ForeignKey

Stock 库存 CharField

Sale 销量 CharField

Logo logo地址 ImageField

Is\_on\_sale 上架 BooleanField

Classification\_id 商品分类ID ForeiginKey

Goods\_spu 商品spu\_id ForeignKey

Add\_time 添加时间 DateTimeField

update\_time 修改时间 DateTimeField

is\_delete 删除 BooleanField

4.商品SPU表

Name 名称 CharField

Details 详情 RichTextUploadingField

1. 商品相册

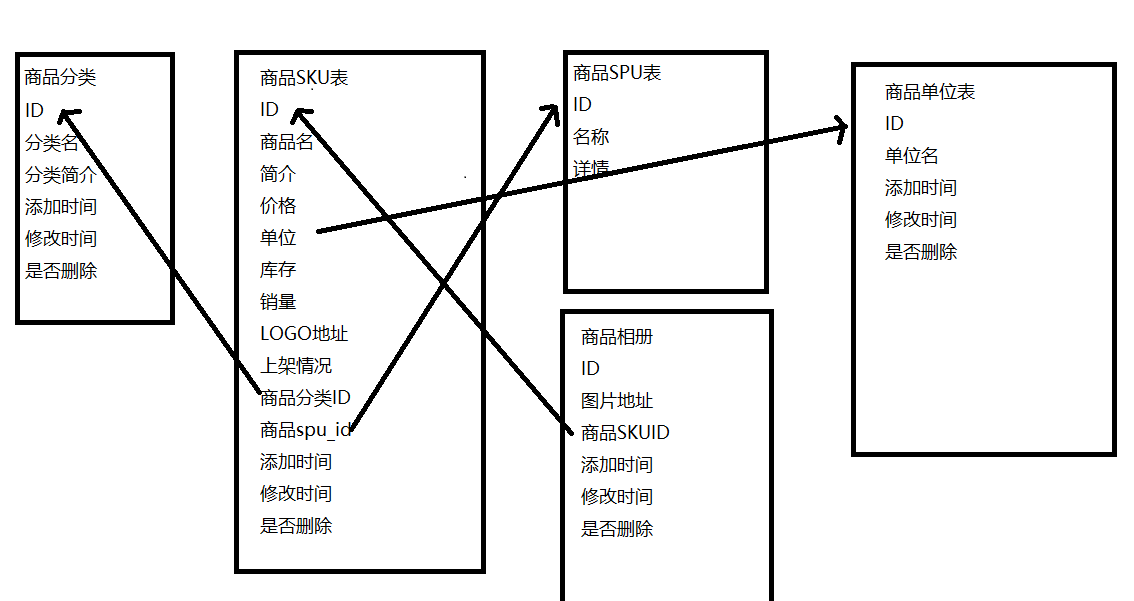
Picture\_address 图片地址 ImageField

Commodity\_id 商品id ForeignKey

Add\_time 添加时间 DateTimeField

Update\_time 修改时间 DateTimeField

is\_delete 删除 BooleanField



1. 首页轮播

Name 名称 CharField

Img\_url 图片地址 ImageField

Order 排序 SmallIntegerField

Add\_time 添加时间 DateTimeField

Update\_time 修改时间 DateTimeField

Is\_delete 删除 BooleanField

1. 首页活动表

title 名称 CharField

Img\_url 图片 ImageField

Url 活动地址 URLField

首页活动专区

Title 活动专区名 CharField

Brief 专区简介 CHarField

Order 排序 SmallIntegerField

is\_on\_sale 上架 BooleanField

goods\_sku 商品 ManyToManyField



5.4要点难点及解决方案

富文本编辑配置 看文档 https://www.zybuluo.com/yqcdwg/note/1244768

商品详情页 需要传递id参数

6.商品列表模块

6.1 需求

用户来到列表页面,显示所有分类信息

用户点击该分类,显示出该分类下的所有商品

6.2流程

1. 查询出所有的分类信息；

2.利用分类信息查询到该目录下的所有商品；

3.遍历出所有商品，将商品渲染到相应的分类下。

6.3设计要点

商品分类表

Classification\_name 分类名 CharField

Introduction 简介 TextField

Add\_time 添加时间 DateTimeField

Update\_time 修改时间 DateTimeField

Is\_delete 删除 BooleanField

商品SKU表

Commodity\_name 商品名 CharField

Introduce 简介 TextField

Price 价格 DecimalField

Conversion 单位 ForeignKey

Stock 库存 CharField

Sale 销量 CharField

Logo logo地址 ImageField

Is\_on\_sale 上架 BooleanField

Classification\_id 商品分类ID ForeiginKey

Goods\_spu 商品spu\_id ForeignKey

Add\_time 添加时间 DateTimeField

update\_time 修改时间 DateTimeField

is\_delete 删除 BooleanField

商品SPU表

Name 名称 CharField

Details 详情 RichTextUploadingField

商品相册

Picture\_address 图片地址 ImageField

Commodity\_id 商品id ForeignKey

Add\_time 添加时间 DateTimeField

Update\_time 修改时间 DateTimeField

is\_delete 删除 BooleanField

6.4难点及解决方案

跳转商品详细信息需要传递该商品的sku id参数

7.商品排序

7.1需求

1. 进入超市页面,点击不同方式的排序按钮能够使商品信息按照相应的规律进行排序;

7.2流程

1.点击综合排序按照综合信息进行排序;

2.点击销量,商品以销量进行降序排序;

3.点击价格:

价格按升序排序

按价格降序排序

* + - 1. 点击新品,按照商品发布时间进行升序排序。

7.3设计要点

商品分类表

Classification\_name 分类名 CharField

Introduction 简介 TextField

Add\_time 添加时间 DateTimeField

Update\_time 修改时间 DateTimeField

Is\_delete 删除 BooleanField

商品SKU表

Commodity\_name 商品名 CharField

Introduce 简介 TextField

Price 价格 DecimalField

Conversion 单位 ForeignKey

Stock 库存 CharField

Sale 销量 CharField

Logo logo地址 ImageField

Is\_on\_sale 上架 BooleanField

Classification\_id 商品分类ID ForeiginKey

Goods\_spu 商品spu\_id ForeignKey

Add\_time 添加时间 DateTimeField

update\_time 修改时间 DateTimeField

is\_delete 删除 BooleanField

7.4难点及解决方案

价格需要升序和降序两种排序方式

利用ajax进行判断；默认的排序方式是升序,用户再次点击转为降序

商品搜索:

1.在虚拟环境中安装包

pip install django-haystack

pip install whoosh

pip install jieba

1. 在settings.py添加应用
2. INSTALLED\_APPS = [
3. 'django.contrib.admin',
4. 'django.contrib.auth',
5. 'django.contrib.contenttypes',
6. 'django.contrib.sessions',
7. 'django.contrib.messages',
8. 'django.contrib.staticfiles',
9. 'haystack', # 全文检索框架
10. 'sp\_user.apps.SpUserConfig', # 用户模块
11. 'sp\_goods.apps.SpGoodsConfig', # 商品模块
12. 'ckeditor', # 添加ckeditor富文本编辑器
13. 'ckeditor\_uploader', # 添加ckeditor富文本编辑器上传部件
14. ]
15. 配置全文搜索框架

· HAYSTACK\_CONNECTIONS = {

· 'default': {

· # 配置搜索引擎

· 'ENGINE': 'haystack.backends.whoosh\_backend.WhooshEngine',

· # 配置索引文件目录

· 'PATH': os.path.join(BASE\_DIR, 'whoosh\_index'),

· },

· }

· #当添加、修改、删除数据时，自动生成索引

· HAYSTACK\_SIGNAL\_PROCESSOR = 'haystack.signals.RealtimeSignalProcessor'

1. 在应用目录下,创建search\_indexes.py文件
2. # 导入全文检索框架索引类
3. from haystack import indexes
4. from sp\_goods.models import GoodsSKU
5. class GoodsSKUSearchIndex(indexes.SearchIndex, indexes.Indexable):
6. # 设置需要检索的主要字段内容 use\_template表示字段内容在模板中
7. text = indexes.CharField(document=True, use\_template=True)
8. # 获取检索对应对的模型
9. def get\_model(self):
10. return GoodsSKU
11. # 设置检索需要使用的查询集
12. def index\_queryset(self, using=None):
13. """Used when the entire index for model is updated."""
14. return self.get\_model().objects.all()
15. 创建模板文件,检索的字段定义在模板中
    1. templates/search/indexes/应用名/模型名小写\_text.txt
16. # 指定搜索的字段
17. {{object.sku\_name}}
18. {{object.brief}}
19. {{object.goods\_spu.content}}

6.生成索引文件

python manage.py rebuild\_index

7.在主路由中配置全文搜索框架

1. urlpatterns = [
2. url(r'^admin/', admin.site.urls),
3. #全文搜索框架
4. url(r'^search/', include('haystack.urls')),
5. # 添加自己的应用的子路由
6. ]

8.创建搜索表单

1. <form action="/search/" method="get">
2. <input type="text" name="q" value="" class="cate-input" placeholder="请输入您要的搜索的产品关键词"/>
3. <input type="submit" class="cate-btn" value=""/>
4. </form>

9.处理结果

1. {% if query %}
2. <ul class="list-pro" style="margin-top: 20px">
3. {% for result in page %}
4. <li>
5. <a href="{% url 'sp\_goods:detail' result.object.pk %}"><img
6. src="{{ MEDIA\_URL }}{{ result.object.logo }}" class="list-pic1"/></a>
7. <div class="shop-list-mid" style="width: 65%;">
8. <div class="tit">
9. <a href="{% url 'sp\_goods:detail' result.object.pk %}">
10. {{ result.object.goods\_spu.spu\_name }}{{ result.object.sku\_name }}
11. </a>
12. </div>
13. <div class="am-gallery-desc">￥{{ result.object.price }}</div>
14. <p>销量：{{ result.object.sale\_num }}件</p>
15. </div>
16. </li>
17. {% empty %}
18. <li>没有找到您搜索的产品！</li>
19. {% endfor %}
20. </ul>
21. {% endif %}

8.购物车商品添加

8.1 需求

在商品详情页面，点击商品下的+号，选择加入购物车的数量， 点击加入购物车，将该数量的商品加入购物车。

8.2 流程

用户点击页面上的+号按钮,该数量的商品加入购物车。

8.3设计要点

用redis保存数据

用户ID user\_id

商品sku\_id sku\_id

商品数量 count

8.4难点及解决方案

需要用到redis保存购物车内的信息

安装reids

redis-server.exe --service-install redis.windows.conf --loglevel verbose

服务器操作

1. net start redis 开启服务
2. net stop redis 停止服务
3. sc delete redis 删除服务

配置reids

1. # 缓存配置
2. CACHES = {
3. "default": {
4. "BACKEND": "django\_redis.cache.RedisCache",
5. "LOCATION": "redis://127.0.0.1:6379/1",
6. "OPTIONS": {
7. "CLIENT\_CLASS": "django\_redis.client.DefaultClient",
8. }
9. }
10. }

需要将 sku\_id 和 count转换为 int

9.购物车商品显示

9.1需求

进入购物车页面，显示出加入购物车的商品

9.2流程

1.从数据库中取出sku\_id, count；

2.从数据库取出完整的商品信息；

3.计算总金额和总商品数量；  
4.在购物车页面渲染出商品的信息，总价,总数量。

9.3设计要点

Redis数据库

用户ID user\_id

商品sku\_id sku\_id

商品数量 count

根据sku\_id查询完整的商品信息

商品SKU表

Commodity\_name 商品名 CharField

Introduce 简介 TextField

Price 价格 DecimalField

Conversion 单位 ForeignKey

Stock 库存 CharField

Sale 销量 CharField

Logo logo地址 ImageField

Is\_on\_sale 上架 BooleanField

Classification\_id 商品分类ID ForeiginKey

Goods\_spu 商品spu\_id ForeignKey

Add\_time 添加时间 DateTimeField

update\_time 修改时间 DateTimeField

is\_delete 删除 BooleanField

9.4难点及解决方案

1. 10.订单功能模块
   1. 10.1确认订单
      1. 10.1.1需求
         1. 在购物车页面，选择好要购买的商品后，点击去结算按钮，进入到确认订单页面；
         2. 在订单页面内显示出用户在购物车内选择好的商品；
         3. 用户选择或添加地址；
         4. 点击确认订单，进入下单页面。
      2. 10.1.2流程
         1. 登录状态的用户选择要结算的商品点击提交订单；
         2. 在redis数据库中查询出从购物车提交过来的商品id和商品数量；
         3. 根据商品的数量和价格得到总价，在加上用户选择的运输方式的价格得到应付总金额；
         4. 选择或添加地址信息，当选择某个地址为默认地址时，之前的默认地址会被修改为普通地址，默认地址被渲染到页面上。
      3. 10.1.3 设计要点

user = models.ForeignKey(to='User', verbose\_name='用户ID')  
name = models.CharField(max\_length=50, verbose\_name='收货人')  
phone = models.CharField(max\_length=11,  
 validators=[  
 RegexValidator('^1[3-9]\d{9}$', '电话号码格式错误')  
 ],  
 verbose\_name='收货电话')  
hcity = models.CharField(max\_length=50, blank=True, null=True, verbose\_name='省')  
hproper = models.CharField(max\_length=50, blank=True, null=True, verbose\_name='市')  
harea = models.CharField(max\_length=50, verbose\_name='区')  
brief = models.CharField(max\_length=100, verbose\_name='详细地址')  
isdefault = models.BooleanField(default=False, verbose\_name='默认地址')

* + 1. 10.1.4难点及解决方案
  1. 10.2 确认支付
     1. 10.2.1需求
        1. 提交订单后，产生订单编号；
        2. 使用订单编号查询出订单信息；
        3. 查询出订单内的商品信息；
        4. 将成功创建的订单显示出来。
     2. 10.2.2流程
        1. 登录状态的用户在确认订单页面点击提交订单按钮；
        2. 创建订单成功，订单编号以参数的形式传递到url内，使用ajax进行页面跳转；
        3. 将创建好的订单内的商品和价格信息渲染出来。
     3. 10.2.3 设计要点

class OrderGoods(BaseModel):  
 order = models.ForeignKey(to="Order", verbose\_name="订单ID")  
 goods\_sku = models.ForeignKey(to="goods.GoodsSKU", verbose\_name="订单商品ID")  
 price = models.DecimalField(max\_digits=9, decimal\_places=2, verbose\_name="商品价格")  
 count = models.SmallIntegerField(verbose\_name="订单商品数量")  
  
 def \_\_str\_\_(self):  
 return "{}:{}".format(self.order.order\_sn, self.goods\_sku.sku\_name)  
  
 class Meta:  
 verbose\_name = "订单商品管理"  
 verbose\_name\_plural = verbose\_name

class Order(BaseModel):  
 order\_status\_choices = (  
 (0, "未支付"),  
 (1, "已支付"),  
 (2, "已发货"),  
 (3, "未评价"),  
 (4, "已完成"),  
 (5, "退发货"),  
 (6, "取消订单"),  
 )  
 user = models.ForeignKey(to="user.User", verbose\_name="用户")  
 order\_sn = models.CharField(max\_length=64, verbose\_name="订单编号")  
 goods\_total\_price = models.DecimalField(max\_digits=9, decimal\_places=2, default=0, verbose\_name="商品总金额")  
 transport\_price = models.DecimalField(max\_digits=9, decimal\_places=2, verbose\_name="运费")  
 transport = models.CharField(max\_length=50, verbose\_name="运输方式")  
 username = models.CharField(max\_length=50, verbose\_name="收货人姓名")  
 phone = models.CharField(max\_length=11, verbose\_name="收货人电话号码")  
 address = models.CharField(max\_length=250, verbose\_name="收货人地址")  
 order\_price = models.DecimalField(max\_digits=9, decimal\_places=2, default=0, verbose\_name="订单总金额")  
 order\_status = models.SmallIntegerField(choices=order\_status\_choices, default=0, verbose\_name="订单状态")  
 payment = models.ForeignKey(to="Payment", null=True, blank=True, verbose\_name="支付方式")  
  
 pay\_time = models.DateTimeField(verbose\_name="支付时间", null=True, blank=True)  
 deliver\_time = models.DateTimeField(verbose\_name="发货时间", null=True, blank=True)  
 finish\_time = models.DateTimeField(verbose\_name="完成时间", null=True, blank=True)

class Payment(BaseModel):  
 name = models.CharField(max\_length=50, verbose\_name="支付方式")  
 brief = models.CharField(max\_length=200, verbose\_name="说明")  
 logo = models.ImageField(upload\_to="payment/%Y", verbose\_name="支付LOGO")  
 is\_default = models.BooleanField(default=False,verbose\_name='是否默认')  
  
 def \_\_str\_\_(self):  
 return self.name  
  
 class Meta:  
 verbose\_name = "支付方式管理"  
 verbose\_name\_plural = verbose\_name

* + 1. 10.2.4 难点及解决方法
       1. 支付宝支付
       2. 安装python工具包

· pip uninstall pycrypto

· # 安装python-alipay-sdk

· pip install python-alipay-sdk --upgrade

判断支付方式,初始化

if payment.name == '支付宝':  
 app\_private\_key\_string = open(os.path.join(settings.BASE\_DIR, "alipay/user\_private\_key.txt")).read()  
 alipay\_public\_key\_string = open(os.path.join(settings.BASE\_DIR, 'alipay/alipay\_public\_key.txt')).read()  
 # 初始化  
 alipay = AliPay(  
 appid="2016092400582068",  
 app\_notify\_url=None,  
 app\_private\_key\_string=app\_private\_key\_string,  
 # 支付宝公钥  
 alipay\_public\_key\_string=alipay\_public\_key\_string,  
 sign\_type="RSA2",  
 debug=True  
 )

* + - 1. 构造地址

# 构造地址  
order\_string = alipay.api\_alipay\_trade\_wap\_pay(  
 out\_trade\_no=order.order\_sn,  
 total\_amount=str(order.order\_price),  
 subject='酱油超市支付',  
 return\_url="http://127.0.0.1:8001/orders/pay",  
 notify\_url=None # 可选, 不填则使用默认notify url  
)

* + - 1. 拼接地址

# 拼接地址  
 url = "https://openapi.alipaydev.com/gateway.do?" + order\_string  
 return JsonResponse(json\_msg(0, '创建支付地址成功', data=url))  
else:  
 return JsonResponse(json\_msg(4,'不支持该支付方式'))

# 查询订单是否交易成功  
# 构造支付请求  
app\_private\_key\_string = open(os.path.join(settings.BASE\_DIR, "alipay/user\_private\_key.txt")).read()  
alipay\_public\_key\_string = open(os.path.join(settings.BASE\_DIR, 'alipay/alipay\_public\_key.txt')).read()  
  
# 初始化对象  
alipay = AliPay(  
 appid="2016092400582019",  
 app\_notify\_url=None, # 默认回调url  
 app\_private\_key\_string=app\_private\_key\_string,  
 # 支付宝的公钥，验证支付宝回传消息使用，不是你自己的公钥,  
 alipay\_public\_key\_string=alipay\_public\_key\_string,  
 sign\_type="RSA2", # RSA 或者 RSA2  
 debug=True # 默认False  
)  
  
# 获取订单编号  
order\_sn = request.POST.get('out\_trade\_no')  
order = Order.objects.get(order\_sn=order\_sn)  
# check order status  
paid = False  
for i in range(10):  
 # 根据订单编号查询  
 result = alipay.api\_alipay\_trade\_query(out\_trade\_no=order\_sn)  
 print(result)  
 if result.get("trade\_status", "") == "TRADE\_SUCCESS":  
 # 支付成功  
 paid = True  
 break  
  
 # 继续执行  
 # check every 3s, and 10 times in all  
 sleep(3)  
 print("not paid...")  
  
# 判断支付是否成功  
# 修改订单状态  
if paid is True:  
 # 支付成功  
 order.order\_status = 1  
 order.save()  
  
return HttpResponse("success")