# 项目介绍

* 1. 项目描述简介

通过完成一个仿淘宝的在线购物超市，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 | Pycharm,Django |
| 相关技术 | 数据库,Django,python |

* 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. 用户功能模块
   1. 用户注册功能模块
      1. 需求

1.用户填写基本信息 ,手机号为移动电话11位数字,密码最少6位并且不得以123456。

2.用户点击获取验证码,填写验证码,当验证码填写正确时点击立即注册按钮提交注册成功,否则注册失败并提示用户。

* + 1. 流程

1.用户填写表单信息。

2.点击获取验证码,并填写验证码。

3.必须同意用户协议。

4.点击立即注册时,提交表单至后台API。

5.使用form表单验证数据是否合法,验证合法时,将数据提交之数据库;当数据不合法时,提示错误,返回注册界面并返回提示用户信息。

* + 1. 设计要点（数据库和页面交互）

1.数据库字段设计

ID 主键

User\_ID varchar 用户手机号

User\_sex 性别(男,女)

Nick\_name varchar 昵称

Borm\_Date data 出生日期

School varchar 学校

Address varchar 地址

Hometown varchar 故乡

Password varchar 用户密码(md5加密)

Balance decimal 账户余额

Integral int 积分

Coupons int 优惠券

Collection int 收藏

Create\_date datatime 注册日期

Head\_portrait varchar 头像

Pay\_password varchar 支付密码(md5加密)

2.前端界面设计

a.页面form表单中input框使用placeholder属性,提示用户填写信息。

b.手机号,密码通过js检测,当合格时点击获取验证码才能获取,否则不可获取验证码。

c.密码框采用掩码形式显示

d.验证码采用4位数字验证

e.当验证码填写后点击立即注册有效,否则点击无效。

* + 1. 要点难点及解决方案

难点1:验证码的获取

解决方案1:后台通过random方法随机生成一个验证码,并响应给前端。

解决方案2: 在阿里云申请一个短信服务接口来发送验证码实现。

* 1. 登陆功能模块
     1. 需求

1.用户填写登陆表单信息

2.也可以通过第三方快捷登陆方式登陆

3.当填写表单后点击我要登录.提示登录成功或失败!

* + 1. 流程

1.填写表单或选择第三方快捷登陆

2.点击登陆,将form表单值提交至后台API接口

3.当API接收到前端页面post提交过来的值时,去除空白符;通过form层验证表单值是否合格,当不合格是直接返回错误并提示用户;当验证合格时,进行数据库查询并判断,结果为True时,返回登录成功标识并提示用户登录成功及跳转至个人中心;结果为False时,返回错误并提示用户登录失败及失败原因!

* + 1. 设计要点（数据库和页面交互）

1.数据设计:

与用户注册数据库公用

2.前台设计:

a.添加input输入框的placeholder属性,来提示用户输入相应的值

b.当点击登录时,判断用户是否已经填写信息并且是否合法

* + 1. 要点难点及解决方案

难点1:

ajax前端页面验证数据是否合法

解决方案:

使用jQuery,对按钮绑定点击触发事件,当被点击时,获取form表单的value值并进行逻辑判断;为True发起ajax请求。

难点2:

第三方快捷登录

解决方案:

QQ登录: 去QQ互联中心去申请一个接口

微信登录: 微信开放平台申请一个智能登录接口

微博登录: 微博开放平台申请一个API

* 1. 用户中心模块
     1. 需求

1.用户登录后展示个人中心。

2.用户能够对个人资料的添加与修改。

3.用户能够查看,添加与修改收货地址。

4.用户可以查看已付款订单,待付款订单,以及待收货。

5.用户可安全退出。

* + 1. 流程

1.使用登录模块功能登录后,跳转至个人中心页面。

2.点击个人资料(通过登录成功并将用户ID存入session到Redis缓存中)读取相应数据至数据库查询用户信息数据,并进行回显。

3.到用户修改后提交时,使用post提交方式提交至后台,当后台得到数据后进行验证成功后存入数据库并调转回个人中心。

4.收货地址管理,用户点击收货地址进入管理页面,显示所有收货地址信息。点击+时,填写完整收货信息,提交时方式为post提交.先到数据库索引当前用户是否超过六条地址,当大于六条地址时,返回添加失败,并提示用户最多只能添加六条。

5.点击编辑时.进入地址详情页并回显数据,当修改数据后提交时(post),修改该信息重新保存至数据库。

6.点击删除时,直接删除此条记录.

* + 1. 设计要点（数据库和页面交互）
       1. 数据库字段设计

收货地址:

Recipient 收件人

Province 省份

City 市/州

Area 区/县

Detail\_address 详细地址

Recipient\_tel 收件人电话

Is\_default 设为默认(True,False)

User\_id 主键

订单

Order\_NO 订单号

Place\_date 下单时间

Accounts\_payable 应付款

Real\_payment 实付款

Contact 联系人

Tel 手机号

State 付款状态(待,已,未)

* + 1. 要点难点及解决方案
       1. 难点:

收货地址的添加与修改.

解决方案:

从session中拿到用户ID ,对ID进行地址添加,添加之前先索引改ID用户地址是否大于6条;修改时调用该地址id进行修改数据库即可。

1. 商品相关功能模块
   1. 首页功能及分类列表页模块
      1. 需求
         1. 通过轮播图展示最新活动。
         2. 点击充值可进入对移动电话和宽带的充值缴费,以及电费,煤气费。
         3. 点击签到可对会员进行加积分。
         4. 活动列表,显示近期的活动促销。
         5. 特色专区可以显示所属特色商品列表。
      2. 流程
         1. 将数据库数据渲染至首页。
         2. 根据渲染出的页面进行轮播图数据库设计。
         3. 根据渲染出的页面进行活动列表的数据库设计。
         4. 根据渲染出的页面进行特色专区数据库设计。
         5. 将数据模型安装至后台管理中心,实现对数据的添加,修改,删除。
      3. 设计要点（数据库和页面交互）
         1. 轮播图/活动列表(继承基础模型)

#### 商品名称 [trade\_name](javascript:;)

商品ID trade\_ID

图片url url\_add

排序(order ) priority

是否轮播 is\_carousel

是否活动 is\_activity

活动地址(URL) url\_activity

* + - 1. 活动专区列表(继承基础模型)(少)

#### 商品名称 [trade\_name](javascript:;)

专区ID trade\_ID

描述 describe

排序(order ) priority

是否上架 is\_ shelves

* + - 1. 专区商品列表(继承基础模型)(多)

专区ID zone\_ID

商品ID trade\_ID

* + - 1. 特色专区

特色专区  
*class* Features(BaseModel):  
 trade\_naem = models.CharField(max\_length=50, verbose\_name='专区名称')  
 trade\_ID = models.CharField(max\_length=20, verbose\_name='商品ID')  
 describe = models.CharField(max\_length=255, verbose\_name='商品描述',null=*True*)  
 priority = models.SmallIntegerField(verbose\_name='排序',default=0)  
 is\_shelves = models.BooleanField(choices=((1, '是'), (0, '否')), verbose\_name='是否上架',default=1)  
  
 *def \_\_str\_\_*(self):  
 *return* self.trade\_naem  
  
 *class* Meta:  
 db\_table = 'features'  
 verbose\_name = '专区分类表'  
 verbose\_name\_plural = verbose\_name

* + - 1. 商品分类

#商品分类  
*class* Goods\_class(BaseModel):  
 class\_name = models.CharField(max\_length=30, verbose\_name='分类名称')  
 introduction = models.CharField(max\_length=255, verbose\_name='分类简介',null=*True*)  
  
 *def \_\_str\_\_*(self):  
 *return* self.class\_name  
  
 *class* Meta:  
 db\_table = 'goods\_class'  
 verbose\_name = '商品分类'  
 verbose\_name\_plural = verbose\_name

* + - 1. 商品SPU表

*class* Goods\_SPU(BaseModel):  
 trade\_naem = models.CharField(max\_length=50, verbose\_name='名称')  
 details = RichTextUploadingField(verbose\_name='详情')  
  
 *def \_\_str\_\_*(self):  
 *return* self.trade\_naem  
  
 *class* Meta:  
 verbose\_name = '商品SPU'  
 verbose\_name\_plural = verbose\_name

* + - 1. 商品相册

*class* Photo\_album(BaseModel):  
 img\_url = models.ImageField(upload\_to='goods/%Y%m/%d', verbose\_name='图片地址')  
 trade\_ID = models.ForeignKey(to='Goods\_SKU', verbose\_name='商品ID')  
  
 # 后台显示缩略图  
 *def* img\_urls(self):  
 *return* "<img style='width:50px;height:30px' src='{}{}'/>".format(settings.MEDIA\_URL,self.img\_url)  
 img\_urls.allow\_tags = *True* img\_urls.short\_description = "img\_url"  
  
 *def \_\_str\_\_*(self):  
 *return* self.trade\_ID  
  
 *class* Meta:  
 verbose\_name = '商品相册'  
 verbose\_name\_plural = verbose\_name

* + - 1. 单位

*class* Unit(BaseModel):  
 unit\_name = models.CharField(max\_length=10,verbose\_name='单位')  
 note = models.CharField(max\_length=20,verbose\_name='备注')  
  
 *def \_\_str\_\_*(self):  
 *return* self.unit\_name  
 *class* Meta:  
 verbose\_name = '产品单位'  
 verbose\_name\_plural = verbose\_name

* + - 1. 商品SKU

*class* Goods\_SKU(BaseModel):  
 goods\_name = models.CharField(max\_length=100, verbose\_name='商品名')  
 goods\_abstract = models.CharField(max\_length=250, verbose\_name='商品简介',null=*True*)  
 goods\_price = models.DecimalField(decimal\_places=2, max\_digits=10, verbose\_name='价格')  
 goods\_unit = models.ForeignKey(to='Unit',verbose\_name='单位')  
 repertory = models.IntegerField(verbose\_name='库存')  
 sales = models.SmallIntegerField(verbose\_name='销量')  
 goods\_LOGO = models.ImageField(upload\_to='goods/%Y%m', verbose\_name='logo地址')  
 is\_shelves = models.BooleanField(choices=((*True*, '是'), (*False*, '否')), verbose\_name='是否上架',default=*True*)  
 goods\_class = models.ForeignKey(to='Goods\_class', verbose\_name='商品分类ID')  
 goods\_spu\_id = models.ForeignKey(to='Goods\_SPU', verbose\_name='商品SPU\_id')  
  
 # 自定义字段  
 *def* show\_logo(self):  
 *return* "<img style='width:50px;height:30px' src='{}{}'/>".format(settings.MEDIA\_URL,self.goods\_LOGO)  
 show\_logo.allow\_tags = *True* show\_logo.short\_description = "img\_url"  
  
 *def \_\_str\_\_*(self):  
 *return* self.goods\_name  
  
 *class* Meta:  
 verbose\_name = '商品SKU表'  
 verbose\_name\_plural = verbose\_name

* + 1. 要点难点及解决方案

难点:

分类列表前端渲染时,即时传递参数改变样式

解决方案:

通过变量进行ID传递自传

难点:

前端上传文件,保存文件不成功

解决方案:

上传文件保存时必须使用get点对象save(提交) 的方式 ,才能成功的存储文件路径

难点:

列表页对数据的排序的优先级

解决方案:

通过自定义参数变量,进行传递传入进行相应排序控制

重点: 模型类中没有明确的外键时,逆向方法(模型对象.别模型类\_set.属性),有明确外键时,正向方法(模型对象.外键属性.别类的属性

* 1. 商品详情页功能模块
     1. 需求

当用户点击对应产品跳转进对应商品详情页

详情页显示产品相册展示商品

显示商品的分类名,商品名,价格,单位商品详情

商品评论,有全部,好评,中评,差评三个等级,除显示评论内容外,还应显示评论日期

* + 1. 流程

建立详情页(detail)视图函数(视图类)

绑定视图到子路由

渲染前端静态详情页

分析前端页面,分析所需数据设计数据库模型及数据库属性

编写数据模型类并添加测试数据

注: 由于添加数据只能从后台添加数据,所以这里省去了验证数据的方法

将数据库数据渲染至前端页面上

对页面进行优化

* + 1. 设计要点（数据库和页面交互）

数据设计:

数据库已有,不在设计

前端页面:

已提供

* + 1. 要点难点及解决方案

难点:

多表查询时,思路不是很明了

解决方案:

参考提供的数据分析表

难点:

商品详情的调用渲染(数据库设计有误)

解决方案:

调整数据模型并重新迁移,查看分析表对一对一,一对多,多对多的不同方式查询渲染

1. 订单功能模块
   1. XXX功能模块
      1. 需求
      2. 流程
      3. 设计要点（数据库和页面交互）
      4. 要点难点及解决方案
   2. XXX功能模块
      1. 需求
      2. 流程
      3. 设计要点（数据库和页面交互）
      4. 要点难点及解决方案
   3. XXX功能模块
      1. 需求
      2. 流程
      3. 设计要点（数据库和页面交互）
   4. 要点难点及解决方案
2. 常见面试问题