# 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
  2. 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. 用户功能模块
   1. 用户账号注册功能模块
      1. 需求
2. 用户输入正确的手机号进行注册，6-16位的密码，然后再次输入密码，后台确认两次密码输入相同，然后点击获取验证码
3. 后台验证用户输入的验证码是否正确，最后点击注册
   * 1. 流程
4. 用户输入手机号，form验证手机号输入的格式，以及是否已经被注册
5. form校验输入的密码是否是6-16位，并且两次输入的密码是否一致
6. 阿里云请求发送验证码，将验证码保存到redis数据库中，生命周期为2分钟，判断用户输入的验证码是否和redis数据库中保存的验证码一致
   * 1. 难点及解决方案

1.难点一：用户表的设计

gender\_choices = (  
 (1, "男"),  
 (2, "女"),  
)  
phone = models.CharField(max\_length=11, verbose\_name="手机号")  
nick\_name = models.CharField(max\_length=50, null=True, blank=True, verbose\_name="昵称")  
password = models.CharField(max\_length=32, verbose\_name="密码")  
gender = models.SmallIntegerField(choices=gender\_choices, null=True, blank=True, verbose\_name="性别")  
birthday = models.DateField(null=True, blank=True, verbose\_name="出生日期")  
school = models.CharField(max\_length=50, null=True, blank=True, verbose\_name="学校")  
hometown = models.CharField(max\_length=50, null=True, blank=True, verbose\_name="家乡")  
address = models.CharField(max\_length=50, null=True, blank=True, verbose\_name="收货地址")

2.难点二：模型继承

通常都会在我们的模型中第一三个字段   
create\_time 创建时间   
upodate\_time 修改时间   
is\_delete 删除时间

如果每个模型都去定义，非常浪费时间，我们可以定义一个基础模型类，让所有模型都继承该模型

在项目根目录下创建一个db包文件夹，在里面创建一个文件 base\_model.py

1. from django.db import models
2. class BaseModel(models.Model):
3. create\_time = models.DateTimeField(verbose\_name="创建时间",
4. auto\_now\_add=True,
5. )
6. update\_time = models.DateTimeField(verbose\_name="更新时间",
7. auto\_now=True,
8. )
9. is\_delete = models.BooleanField(verbose\_name="是否删除",
10. default=False,
11. )
12. class Meta:
13. # 说明是一个抽象模型类，不会被迁移
14. abstract = True

Meta 中abstract用于声明该类为抽象类只能用于继承

之后所有的模型都继承BaseModel类

3.难点三：验证手机号码是否被注册和两次输入的密码是否一致

class UserRegisterForm(forms.ModelForm):  
 password1 = forms.CharField(max\_length=16,  
 min\_length=6,  
 error\_messages={  
 "required": "密码必填，请填写密码",  
 "max\_length": "密码长度不能大于16位，请重新输入",  
 "min\_length": "密码长度不能小于6位，请重新输入"  
 })  
 password2 = forms.CharField(error\_messages={  
 "required": "确认密码必填，请填写密码"  
 })  
  
 class Meta:  
 model = UserModel  
 fields = ['phone']  
 error\_messages = {  
 "phone": {  
 "required": "手机号码必须填写，请输入手机号"  
 }  
 }  
  
 # 验证两次输入的密码是否一致  
 def clean\_password2(self):  
 pwd1 = self.cleaned\_data.get("password1")  
 pwd2 = self.cleaned\_data.get("password2")  
 if pwd1 and pwd2 and pwd1 != pwd2:  
 raise forms.ValidationError("两次输入的密码不一致，请重新输入")  
 else:  
 return pwd2  
  
 # 验证手机号是否已经注册  
 def clean\_phone(self):  
 phone = self.cleaned\_data.get("phone")  
 # 判断手机号是否存在  
 res = UserModel.objects.filter(phone=phone).exists()  
 if res:  
 raise forms.ValidationError("该手机号码已经注册")  
 else:  
 return phone

1. 难点四：安全的加密方法

def set\_password(password):  
 *"""  
 定义新的加密方法，返回加密后的密码  
 """* new\_password = "{}{}".format(password, settings.SECRET\_KEY)  
 h = hashlib.md5(new\_password.encode("utf-8"))  
 new\_password = h.hexdigest()  
 return new\_password

* 1. 用户登陆功能模块
     1. 需求
        1. 用户填写用户名和密码，然后登陆
        2. 通过form验证用户名是否存在以及密码是否正确
        3. 如果正确跳转到个人中心，并保存session
     2. 流程
        1. 用户分别输入用户名和密码
        2. 通过form分别判断手机号是否存在，再判断密码是否正确，不正确把错误信息渲染到页面
        3. 在登陆的视图函数上获取数据和创建form对象，验证数据的合法性
        4. 如果不合法把错误信息渲染到页面，如果合法则跳转到个人中心，并且保存session
     3. 难点及解决方案
        1. 难点一：验证密码是否正确

class UserLoginForm(forms.ModelForm):  
 class Meta:  
 model = UserModel  
 fields = ["phone", "password"]  
 error\_messages = {  
 "phone": {  
 "required": "手机号必须填写，请输入手机号"  
 },  
 "password": {  
 "required": "密码必须填写，请输入密码"  
 }  
 }  
  
 # 验证用户名和密码  
 def clean(self):  
 data = self.cleaned\_data  
 phone = data.get("phone")  
 password = data.get("password")  
 # 判断用户是否都输入用户名和密码  
 if all([phone, password]):  
 # 根据用户输入的phone查找数据库中的用户user  
 try:  
 user = UserModel.objects.get(phone=phone)  
 except UserModel.DoesNotExist:  
 raise forms.ValidationError({"phone": "手机号/用户名不存在！"})  
 # 判断密码是否正确  
 if set\_password(password) != user.password:  
 raise forms.ValidationError({"password": "密码输入错误，登陆失败"})  
 else:  
 # 正确就将信息保存到data中  
 data["user"] = user  
 return data  
 else:  
 return data

* + - 1. 难点二：判断是否有session的装饰器

def verify\_session(old):  
 *"""  
 定义获取session的装饰器  
  
 """* def check\_login(request, \*args, \*\*kwargs):  
 if request.session.get("id") is None:  
 return redirect("user:login")  
 else:  
 return old(request, \*args, \*\*kwargs)  
  
 return check\_login

* 1. 个人资料页面
     1. 需求
        1. 回显用户的信息（如果存在）
        2. 用户可以补全个人资料，要验证输入数据的合法性
        3. 用户提交之后回显用户信息
     2. 流程
        1. 回显数据库中存在的用户信息
        2. 修改前端页面相应的form标签，type，name等属性
        3. form验证用户填写的信息的合法性
     3. 难点及解决方案
        1. 难点一：上传文件头像

# 设置静态文件的上传目录  
MEDIA\_URL = "/static/media/"  
  
# 配置该URL对应的物理目录存储地址  
MEDIA\_ROOT = os.path.join(BASE\_DIR, 'static/media')

前端页面的form

<form action="" method="post" enctype="multipart/form-data">

* + - 1. 难点二：日期回显格式

<input type="date" name="birthday" value="{{ user.birthday|date:'Y-m-d' }}" class="infor-tele" placeholder="出生日期"/>

1. 商品功能模块
   1. 商品增删改查模块
      1. 需求
         1. 商家登录后台系统；
         2. 添加||删除||修改||查看 商店或商品信息；
         3. 添加完成后，添加的信息能够写入数据库，并且能够通过前端渲染到页面上。
      2. 流程
         1. 设计好数据库；
         2. 将数据库内供商家控制的模型添加到admin控制模块中；
         3. 创建超级用户来管理后端数据；
         4. 通过后台账户进入到后台；
         5. 对数据进行增删查改。
      3. 设计要点（数据库和页面交互）

class ShopCategory(BaseModel):  
 *"""  
  
 分类名  
 分类简介  
 添加时间  
 修改时间  
 是否删除  
 """* name = models.CharField(max\_length=50, verbose\_name="分类名")  
 intro = models.CharField(max\_length=255, null=True, blank=True, verbose\_name="分类简介")  
  
 class Meta:  
 db\_table = "ShopCategory"  
 verbose\_name = "商品分类管理"  
 verbose\_name\_plural = verbose\_name  
  
 def \_\_str\_\_(self):  
 return self.name  
  
  
class ShopSpu(BaseModel):  
 *"""  
 商品SPU表  
 ID  
 名称  
 详情  
 """* name = models.CharField(max\_length=50, verbose\_name="名称")  
 detail = models.CharField(max\_length=255, null=True, blank=True, verbose\_name="详情")  
  
 class Meta:  
 db\_table = "ShopSpu"  
 verbose\_name = "商品SPU管理"  
 verbose\_name\_plural = verbose\_name  
  
 def \_\_str\_\_(self):  
 return self.name  
  
  
class ShopUnit(BaseModel):  
 *"""  
 商品单位表  
 ID  
 单位名（斤，箱）  
 添加时间  
 修改时间  
 是否删除  
 """* name = models.CharField(max\_length=50, verbose\_name="单位名（斤，箱）")  
  
 class Meta:  
 db\_table = "ShopUnit"  
 verbose\_name = "商品单位管理"  
 verbose\_name\_plural = verbose\_name  
  
 def \_\_str\_\_(self):  
 return self.name  
  
  
class ShopSku(BaseModel):  
 *"""  
 商品SKU表  
 ID  
 商品名  
 简介  
 价格  
 单位  
 库存  
 销量  
 LOGO地址  
 是否上架  
 商品分类ID  
 商品spu\_id  
 """* name = models.CharField(max\_length=50, verbose\_name="商品名称")  
 intro = models.CharField(max\_length=255, null=True, blank=True, verbose\_name="简介")  
 price = models.DecimalField(max\_digits=9, decimal\_places=2, verbose\_name="价格", default=0)  
 unit = models.ForeignKey(to=ShopUnit, verbose\_name="单位")  
 stock = models.IntegerField(verbose\_name="库存", default=0)  
 sales = models.IntegerField(verbose\_name="销量", default=0)  
 url = models.ImageField(upload\_to="shop\_sku/%Y%m/%d", verbose\_name="LOGO地址")  
 isAdded = models.BooleanField(default=True, verbose\_name="是否上架")  
 category = models.ForeignKey(to=ShopCategory, verbose\_name="商品分类")  
 spu = models.ForeignKey(to=ShopSpu, verbose\_name="商品SPU")  
  
 class Meta:  
 db\_table = "ShopSku"  
 verbose\_name = "商品SKU管理"  
 verbose\_name\_plural = verbose\_name  
  
 def \_\_str\_\_(self):  
 return self.name  
  
  
class ShopPicture(BaseModel):  
 *"""  
 图片地址  
 商品ID  
 """* url = models.ImageField(upload\_to="shop\_picture/%Y%m/%d", verbose\_name="图片地址")  
 shop\_sku = models.ForeignKey(to=ShopSku, verbose\_name="商品SKU")  
  
 class Meta:  
 db\_table = "shop\_picture"  
 verbose\_name = "商品相册"  
 verbose\_name\_plural = verbose\_name  
  
 def \_\_str\_\_(self):  
 return "图片地址"  
  
  
class LunBoModel(BaseModel):  
 *"""  
 首页轮播商品  
 ID  
 名称  
 商品SKUID  
 图片  
 排序（order）  
 添加时间  
 修改时间  
 是否删除  
 """* name = models.CharField(max\_length=50, verbose\_name="名称")  
 shop\_sku = models.ForeignKey(to=ShopSku, verbose\_name="商品SKUID")  
 picture = models.ImageField(upload\_to="lunbo/%Y%m/%d", verbose\_name="图片")  
 order = models.SmallIntegerField(verbose\_name="排序", default=0)  
  
 class Meta:  
 db\_table = "lunbo"  
 verbose\_name = "轮播图管理"  
 verbose\_name\_plural = verbose\_name  
  
 def \_\_str\_\_(self):  
 return self.name  
  
  
class Activity(BaseModel):  
 *"""  
 首页活动表  
 ID  
 名称  
 图片地址  
 url地址  
  
 """* name = models.CharField(max\_length=50, verbose\_name="活动名称")  
 picture = models.ImageField(upload\_to="activity/%Y%m/%d", verbose\_name="图片地址")  
 url = models.URLField(verbose\_name="url地址")  
  
 class Meta:  
 db\_table = "activity"  
 verbose\_name = "首页活动管理"  
 verbose\_name\_plural = verbose\_name  
  
 def \_\_str\_\_(self):  
 return self.name  
  
  
class ActivityZone(BaseModel):  
 *"""  
 首页活动专区  
 ID  
 名称  
 描述  
 排序  
 是否上架  
 """* name = models.CharField(max\_length=50, verbose\_name="活动专区名称")  
 desc = models.CharField(max\_length=255, null=True, blank=True, verbose\_name="描述")  
 order = models.SmallIntegerField(default=0, verbose\_name="排序")  
 isAdded = models.BooleanField(default=True, verbose\_name="是否上架")  
  
 class Meta:  
 db\_table = "activity\_zone"  
 verbose\_name = "活动专区管理"  
 verbose\_name\_plural = verbose\_name  
  
 def \_\_str\_\_(self):  
 return self.name  
  
  
class ActivityGoods(BaseModel):  
 *"""  
 首页专区活动商品表  
 ID  
 专区ID  
 商品SKU ID  
 添加时间  
 修改时间  
 是否删除  
  
 """* zone = models.ForeignKey(to=ActivityZone, verbose\_name="活动专区ID")  
 shop\_sku = models.ForeignKey(to=ShopSku, verbose\_name="商品SKU\_ID")  
  
 class Meta:  
 db\_table = "activity\_goods"  
 verbose\_name = "首页专区活动商品管理"  
 verbose\_name\_plural = verbose\_name  
  
 def \_\_str\_\_(self):  
 return "首页专区活动商品"

* 1. 首页模块
     1. 需求
        1. 渲染首页的banner部分；
        2. 渲染首页不规则展示推荐信息；
        3. 查找出商品活动信息，渲染出活动以及该活动下的商品信息。
     2. 流程
        1. 渲染不规则推荐信息部分

查询出不规则展示信息的，将所有的信息以字典形式传递到主页，在主页上渲染出所有不规则信息。

* + - 1. 渲染banner部分

查询出所有banner信息，将banner信息传递到首页上，首页遍历出所有的banner图片，渲染到首页的banner框内；

* + - 1. 渲染首页的活动信息

在视图函数内查询出所有的活动表；

遍历表内的信息，得到所有的活动；

通过活动信息，利用外键查询，得到所有的商品信息；

遍历商品信息，得到所有的商品。

* 1. 商品分类模块
     1. 需求
        1. 得到所有的分类信息；
        2. 遍历分类信息，将所有分类显示到页面上；
        3. 通过分类查询出该分类下的商品；
        4. 将查询到的商品信息渲染到对应的分类目录下。
     2. 流程
        1. 查询出所有的分类信息；
        2. 在静态页面中遍历并渲染分类信息；
        3. 利用分类信息查询到该目录下的所有商品；
        4. 遍历出所有商品，将商品渲染到相应的分类下。

1. 常见面试问题