# 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. 用户账号功能注册
      1. 需求
         1. 用户填写基本信息:正确的手机号，长度为8位16位的手机号，和密码相同的确认密码，完成后得到正确的验证码。
         2. 填写正确的验证码后，点击立即注册，跳转到登陆页面。
      2. 流程
         1. 用户填写基本信息，填写正确的数据完毕后点击获取验证码，
         2. 得到验证码后，用户填写正确的验证码后，点击立即注册，数据被拿到form表单进行验证。
         3. form表单验证数据是否合法，手机号码是否正确且已存在，密码是否正确，
         4. 如果填写的数据错误，则在界面提示错误信息，并提示用户重新注册
         5. 如果填写的密码正确，则表示注册成功，跳转到登录页面
      3. 设计要点（数据库和页面交互）
         1. 数据库基本字段：

ID：主键

phone：用户手机号码 Charfield

password:用户密码 Charfield

nickname:用户昵称 Charfield

gender:性别 choices

school:学校 Charfield

Position:所在位置 Charfield

HomeTown:家乡 Charfield

birth\_of\_data :出生日期 Datefield

add\_time:添加时间 Datetimefield

update\_time:更新时间 Datetimefield

is\_delete：是否删除 BooleanField

* + - 1. 前端页面对数据的校验

1. 在每一个文本框下面加一个标签显示错误信息
2. 文本框进行空字段校验,如果为空,则在该文本框下面提示用户该文本框位必填项
3. 对必填的文本框进行数据合法性的验证,如果用户填写的字数多或者少,对用户进行提示, 并重新填写.
4. 对手机号进行正则判断,如果格式不对,则提示用户填写正确格式的手机号码
5. 对验证码进行校验.
   * 1. 要点难点及解决方法

难点一:

* 1. 用户登录功能模块:
     1. 需求
        1. 用户填写手机号和验证码,点击登录,并将数据提交
        2. 对用户提交的数据进行验证,如果数据与数据库内的数据一直,则登陆成功,跳转到个人详情界面
     2. 流程
        1. 用户填写正确的数据
        2. 数据被拿到form表单进行验证
        3. 如果用户填写的手机号和密码没有问题,将数据交给单独封装的一个函数进行hashMD5加密,
        4. 将加密后的数据与数据库中加密的数据进行比较,如果一样,则登录成功,跳转到个人详情页面
     3. 设计流程(数据库和页面交互)
        1. 数据库基本字段:

ID：主键

phone：用户手机号码 Charfield

password:用户密码 Charfield

nickname:用户昵称 Charfield

gender:性别 choices

school:学校 Charfield

Position:所在位置 Charfield

HomeTown:家乡 Charfield

birth\_of\_data :出生日期 Datefield

add\_time:添加时间 Datetimefield

update\_time:更新时间 Datetimefield

is\_delete：是否删除 BooleanField

* + - 1. 前端数据对页面的验证

1. 判断文本框是否为空,如果为空,在对应文本框下提示失败,提示用户重新填写登录信息
2. 对用户填写的数据进行校验,判断用户如的数据手否规范
3. 在对应文本框下添加标签,提示用户输入正确的的信息
   * 1. 要点难点及解决方案

难点一:用户填写的为明文密码,如何与数据库加密后的密码进行比较

解决方案:将用户填写的密码同样进行哈希加密,再与数据库中加密的数据进行比较,判断是否一致

难点二:用户登录后需要保存session,以便跳转到需要session的页面,不需要重复登录

解决方案:定义一个函数保存session,并且设置过期时间,然后定义一个装饰器,将函数传入到定义的这个装饰器中,需要session的页面就需要继承这个视图类,这里用到了@method\_decorator,可以传入参数

* 1. 用户修改功能模块
     1. 需求
        1. 用户登录商城
        2. 用户的资料渲染到个人资料界面
        3. 用户在个人资料页面下可以修改自己的资料
        4. 点击确认按钮,保存用户修改的信息,提交到数据库
     2. 流程
        1. 用户填写好基本信息后,点击修改按钮进行提交
        2. Form表单验证提交过来的数据
        3. 如果校验成功,返回到当前页面
     3. 设计要点(数据库和页面交互)
        1. 数据库要和注册数据一致
        2. 交互设计

1. 为文本框添加placeorder属性，提示用户需要填写的信息
2. 对修改的信息进行form验证,对用户输入信息验证是否合法
3. 对必填字段进行空字段验证,向用户提示哪些字段必填
   * 1. 要点难点及解决方案
        1. 用户登录后如何将用户的信息渲染到个人资料页面

解决:使用session来获取用户的信息,在将用户的信息查询出来,再将数据渲染到个人详情页面

* + - 1. 用户登录后的状态怎么保存

解决:用session将用户的唯一信息进行保存，当用户登录后，session会被传递到浏览器，用户的信息也将被保存。

* 1. 流程
  2. 设计要点（数据库和页面交互）
  3. 要点难点及解决方案

1. 商品功能模块
   1. 商品详情展示模块
      1. 需求
         1. 用户点击某个商品后,会跳转到商品的详情页面
         2. 详情页面上展示商品的基本信息,价格,商品的详情,单位
      2. 流程
         1. 详情页能够轮播该商品的图片信息
         2. 视图函数有一个参数id, 根据id查询出商品信息，再利用商品信息查询出商品的轮播图片。
      3. 设计要点(数据库和页面交互)
         1. 数据库

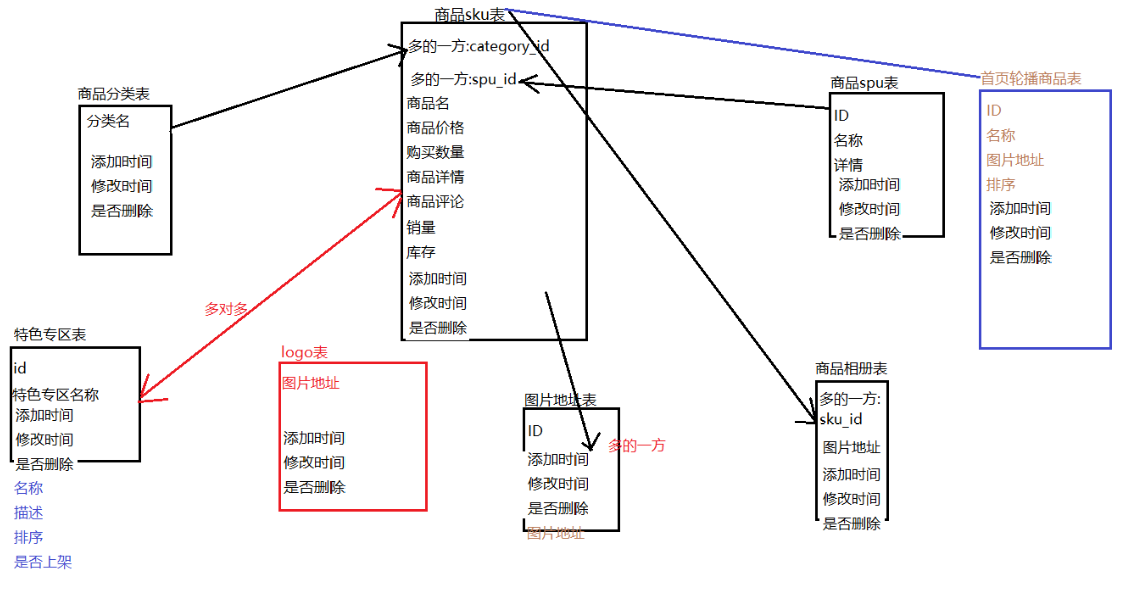
1)商品sku表

class Goods\_sku(models.Model):  
 *"""商品sku表"""* trade\_name = models.CharField(max\_length=100,  
 verbose\_name="商品sku名")  
 breif = models.CharField(max\_length=200,  
 null=True,  
 blank=True,  
 verbose\_name="商品简介")  
 price = models.DecimalField(max\_digits=9,  
 decimal\_places=2,  
 verbose\_name="商品价格",  
 default=0)  
  
 stock = models.IntegerField(verbose\_name="商品库存",  
 default=0)  
  
 sales = models.IntegerField(default=0,  
 verbose\_name="商品销量")  
  
 logo = models.ImageField(upload\_to='goods/%Y%m/%d',  
 verbose\_name="封面图片")  
  
 is\_shelf = models.BooleanField(verbose\_name="是否上架",  
 choices=is\_on\_sale\_choices,  
 default=False)  
  
 unit = models.ForeignKey(to="Unit",  
 verbose\_name="单位")  
 commodity = models.ForeignKey(to="Commodity",  
 verbose\_name="商品分类")  
 goods\_spu = models.ForeignKey(to="Goods\_spu",  
 verbose\_name="商品spu")  
 activity = models.ForeignKey(to="Activity",  
 verbose\_name="活动分类")  
  
 add\_time = models.DateTimeField(auto\_now\_add=True, verbose\_name="创建时间")  
 update\_time = models.DateTimeField(auto\_now=True, verbose\_name="更新时间")  
 is\_delete = models.BooleanField(default=False, verbose\_name="是否删除")  
  
 def \_\_str\_\_(self):  
 return self.trade\_name  
  
 class Meta:  
 verbose\_name = "商品sku管理"  
 verbose\_name\_plural = verbose\_name

1. 商品相册表

class Album(models.Model):  
 *"""商品相册管理"""* picture\_address = models.ImageField(upload\_to="goods\_pictures/%Y%m/%d",  
 verbose\_name="商品图片地址")  
 goods\_sku = models.ForeignKey(to="Goods\_sku",  
 verbose\_name="商品sku")  
  
 add\_time = models.DateTimeField(auto\_now\_add=True, verbose\_name="创建时间")  
 update\_time = models.DateTimeField(auto\_now=True, verbose\_name="更新时间")  
 is\_delete = models.BooleanField(default=False, verbose\_name="是否删除")  
  
 def \_\_str\_\_(self):  
 return "商品相册:{}".format(self.picture\_address.name)  
  
 class Meta:  
 verbose\_name = "商品相册管理"  
 verbose\_name\_plural = verbose\_name

* + 1. 要点及难点及解决方案
       1. 关联查询时要注意正逆关系,你想查询时要用 一对象.模型类名\_set.all(),正像查询时用 多对象.关联字段
       2. 在多的一方添加外键
       3. 分析好模型及模型关系后才能建表,注意外键添加是否已经添加完,
    2. 建表分析过程



* 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,将所有的分类信息渲染都页面上
        3. 点击分类信息可以查询到下面的商品信息
        4. 遍历所有的商品,将所有的商品渲染到相应的分类下
     2. 流程
        1. 查询出所有的分类信息
        2. 在静态页面中遍历并得到分类信息
        3. 在分类信息下可以查询到给分类下的所有商品信息
        4. 遍历所有的商品,将所有的商品渲染到相应的分类下
     3. 设计要点(数据库和页面交互)
        1. 数据库

2)商品分类表

class Commodity(models.Model):  
 *"""商品分类模型"""* classification\_name = models.CharField(max\_length=50,  
 verbose\_name="分类名", )  
  
 add\_time = models.DateTimeField(auto\_now\_add=True, verbose\_name="创建时间")  
 update\_time = models.DateTimeField(auto\_now=True, verbose\_name="更新时间")  
 is\_delete = models.BooleanField(default=False, verbose\_name="是否删除")  
  
 def \_\_str\_\_(self):  
 return self.classification\_name  
  
 class Meta:  
 verbose\_name = "商品分类管理"  
 verbose\_name\_plural = verbose\_name

2)商品spu表

class Goods\_spu(models.Model):  
 *"""商品spu表"""* spu\_name = models.CharField(max\_length=50,  
 verbose\_name="商品spu名")  
  
 content = RichTextUploadingField(verbose\_name="商品详情")  
 add\_time = models.DateTimeField(auto\_now\_add=True, verbose\_name="创建时间")  
 update\_time = models.DateTimeField(auto\_now=True, verbose\_name="更新时间")  
 is\_delete = models.BooleanField(default=False, verbose\_name="是否删除")  
  
 def \_\_str\_\_(self):  
 return self.spu\_name  
  
 class Meta:  
 verbose\_name = "商品spu名"  
 verbose\_name\_plural = verbose\_name

1. 商品单位表

class Unit(models.Model):  
 *"""商品单位表"""* unit\_name = models.CharField(max\_length=20,  
 verbose\_name="商品单位")  
 add\_time = models.DateTimeField(auto\_now\_add=True, verbose\_name="创建时间")  
 update\_time = models.DateTimeField(auto\_now=True, verbose\_name="更新时间")  
 is\_delete = models.BooleanField(default=False, verbose\_name="是否删除")  
  
 def \_\_str\_\_(self):  
 return self.unit\_name  
  
 class Meta:  
 verbose\_name = "商品单位"  
 verbose\_name\_plural = verbose\_name

1. 商品sku表

class Goods\_sku(models.Model):  
 *"""商品sku表"""* trade\_name = models.CharField(max\_length=100,  
 verbose\_name="商品sku名")  
 breif = models.CharField(max\_length=200,  
 null=True,  
 blank=True,  
 verbose\_name="商品简介")  
 price = models.DecimalField(max\_digits=9,  
 decimal\_places=2,  
 verbose\_name="商品价格",  
 default=0)  
  
 stock = models.IntegerField(verbose\_name="商品库存",  
 default=0)  
  
 sales = models.IntegerField(default=0,  
 verbose\_name="商品销量")  
  
 logo = models.ImageField(upload\_to='goods/%Y%m/%d',  
 verbose\_name="封面图片")  
  
 is\_shelf = models.BooleanField(verbose\_name="是否上架",  
 choices=is\_on\_sale\_choices,  
 default=False)  
  
 unit = models.ForeignKey(to="Unit",  
 verbose\_name="单位")  
 commodity = models.ForeignKey(to="Commodity",  
 verbose\_name="商品分类")  
 goods\_spu = models.ForeignKey(to="Goods\_spu",  
 verbose\_name="商品spu")  
 activity = models.ForeignKey(to="Activity",  
 verbose\_name="活动分类")  
  
 add\_time = models.DateTimeField(auto\_now\_add=True, verbose\_name="创建时间")  
 update\_time = models.DateTimeField(auto\_now=True, verbose\_name="更新时间")  
 is\_delete = models.BooleanField(default=False, verbose\_name="是否删除")  
  
 def \_\_str\_\_(self):  
 return self.trade\_name  
  
 class Meta:  
 verbose\_name = "商品sku管理"  
 verbose\_name\_plural = verbose\_name

* + - 1. 页面的交互

每个分类下都有对应的该分类下的所有商品信息,如果没有,则提示给分类下没有商品信息

* + 1. 要点难点及解决方案
       1. 每个商品都有对应的id

解决方案:为了方便排序,url地址使用两个参数,一个是商品的分类的id信息,一个是排序的序号信息

* + - 1. 轮播图片有个js代码必须加上
  1. 首页模块
     1. 需求
        1. 渲染首页的轮播部分
        2. 渲染首页的特色专区模块
     2. 流程
        1. 渲染轮播部分

查询出所有轮播信息，将轮播信息传递到首页上，首页遍历出所有的轮播图片，渲染到首页的轮播框内；

* + - 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. 设计要点(数据库和页面交互)
       1. 数据库

1. 首页特色专区表

class ActivityZone(models.Model):  
 *"""  
 首页特色专区  
 """* title = models.CharField(verbose\_name='活动专区名称', max\_length=150)  
 brief = models.CharField(verbose\_name="活动专区的简介",  
 max\_length=200,  
 null=True,  
 blank=True,  
 )  
 order = models.SmallIntegerField(verbose\_name="排序",  
 default=0,  
 )  
 is\_on\_sale = models.BooleanField(verbose\_name="是否上线",  
 choices=is\_on\_sale\_choices,  
 default=0,  
 )  
 goods\_sku = models.ManyToManyField(to="Goods\_sku", verbose\_name="商品")  
  
 add\_time = models.DateTimeField(auto\_now\_add=True, verbose\_name="创建时间")  
 update\_time = models.DateTimeField(auto\_now=True, verbose\_name="更新时间")  
 is\_delete = models.BooleanField(default=False, verbose\_name="是否删除")  
  
 def \_\_str\_\_(self):  
 return self.title  
  
 class Meta:  
 verbose\_name = "特色专区管理"  
 verbose\_name\_plural = verbose\_name

1. 轮播图片表

class Wheel\_planting(models.Model):  
 *"""轮播图片管理"""* name = models.CharField(max\_length=50,  
 verbose\_name="轮播活动名称")  
  
 picture\_address = models.ImageField(verbose\_name='轮播图片地址',  
 upload\_to='banner/%Y%m/%d')  
  
 order = models.SmallIntegerField(verbose\_name="排序",  
 default=0, )  
  
 add\_time = models.DateTimeField(auto\_now\_add=True, verbose\_name="创建时间")  
 update\_time = models.DateTimeField(auto\_now=True, verbose\_name="更新时间")  
 is\_delete = models.BooleanField(default=False, verbose\_name="是否删除")  
  
 def \_\_str\_\_(self):  
 return self.name  
  
 class Meta:  
 verbose\_name = "轮播图片管理"  
 verbose\_name\_plural = verbose\_name

* + - 1. 页面交互
      2. 无
    1. 要点难点及解决方案

难点1：

查询活动以及活动下的商品信息

解决方案：

先查询出所有活动信息，遍历活动信息，利用，每一个活动来查询出活动模块下的所有商品信息，遍历这些信息，利用遍历得到的信息渲染出活动模块。

* 1. 商品排序模块
     1. 需求
        1. 点击进入超市页面,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
        2. 点击综合排序按照综合信息进行排序
        3. 点击销量,商品以销量进行降序排序
        4. 点击价格:

价格按升序排序

* + - 1. 按价格降序排序
      2. 点击新品,按照商品发布时间进行升序排序
    1. 流程
       1. 当点击某一项方式进行排序时，所有的商品按照该方式进行排序
       2. 视图函数中传递两个参数，用来方便排序以及显示分类下的商品信息；第一个参数是分类的com\_id,第二个参数是排序方式的order
       3. 先用第一个参数作为主键来查询商品分类，通过分类得到分类下的所有商品
       4. 设置一个排序列表，列表中的字段是用于排序的字段
       5. 商品排序，将排序列表作为参数传递给排序方法，将第二个参数order传递给作为参数传递给排序列表
       6. 在前端渲染页面上，将排序的参数传递到相应的位置，进行排序
    2. 设计要点（数据库和页面交互)
       1. 数据库

1. 商品分类表

class Commodity(models.Model):  
 *"""商品分类模型"""* classification\_name = models.CharField(max\_length=50,  
 verbose\_name="分类名", )  
  
 add\_time = models.DateTimeField(auto\_now\_add=True, verbose\_name="创建时间")  
 update\_time = models.DateTimeField(auto\_now=True, verbose\_name="更新时间")  
 is\_delete = models.BooleanField(default=False, verbose\_name="是否删除")  
  
 def \_\_str\_\_(self):  
 return self.classification\_name  
  
 class Meta:  
 verbose\_name = "商品分类管理"  
 verbose\_name\_plural = verbose\_name

1. 商品sku表

class Goods\_sku(models.Model):  
 *"""商品sku表"""* trade\_name = models.CharField(max\_length=100,  
 verbose\_name="商品sku名")  
 breif = models.CharField(max\_length=200,  
 null=True,  
 blank=True,  
 verbose\_name="商品简介")  
 price = models.DecimalField(max\_digits=9,  
 decimal\_places=2,  
 verbose\_name="商品价格",  
 default=0)  
  
 stock = models.IntegerField(verbose\_name="商品库存",  
 default=0)  
  
 sales = models.IntegerField(default=0,  
 verbose\_name="商品销量")  
  
 logo = models.ImageField(upload\_to='goods/%Y%m/%d',  
 verbose\_name="封面图片")  
  
 is\_shelf = models.BooleanField(verbose\_name="是否上架",  
 choices=is\_on\_sale\_choices,  
 default=False)  
  
 unit = models.ForeignKey(to="Unit",  
 verbose\_name="单位")  
 commodity = models.ForeignKey(to="Commodity",  
 verbose\_name="商品分类")  
 goods\_spu = models.ForeignKey(to="Goods\_spu",  
 verbose\_name="商品spu")  
 activity = models.ForeignKey(to="Activity",  
 verbose\_name="活动分类")  
  
 add\_time = models.DateTimeField(auto\_now\_add=True, verbose\_name="创建时间")  
 update\_time = models.DateTimeField(auto\_now=True, verbose\_name="更新时间")  
 is\_delete = models.BooleanField(default=False, verbose\_name="是否删除")  
  
 def \_\_str\_\_(self):  
 return self.trade\_name  
  
 class Meta:  
 verbose\_name = "商品sku管理"  
 verbose\_name\_plural = verbose\_name

* + 1. 要点难点及解决方案
       1. 难点1： 对价格进行排序

价格既可以升序排列，又可以降序排列，两个排序方式都需要提供给用户，如何才能实现在用户点击升序或者降序排列的按钮时能实现相应的排序。

解决方案：

利用ajax进行判断；默认的排序方式是升序，为排序添加一个类，用该类来作为判断标识；当再次点击时，判断排序的序号为降序或升序，如果为降序，则将他的排序方式修改为升序，反之，则修改为升序。

* 1. 商品增删改差模块
     1. 需求
        1. 商家登录后台系统
        2. 完成添加||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. 将设计好的数据库内供商家控制的模型添加到admin控制模块中
        2. 创建超级管理员账户来操作后端数据
        3. 通过后台账户进入到后台
        4. 对数据进行增删改查
     3. 设计要点（数据库和页面交互）
        1. 数据库

1. 商品分类表

class Commodity(models.Model):  
 *"""商品分类模型"""* classification\_name = models.CharField(max\_length=50,  
 verbose\_name="分类名", )  
  
 add\_time = models.DateTimeField(auto\_now\_add=True, verbose\_name="创建时间")  
 update\_time = models.DateTimeField(auto\_now=True, verbose\_name="更新时间")  
 is\_delete = models.BooleanField(default=False, verbose\_name="是否删除")  
  
 def \_\_str\_\_(self):  
 return self.classification\_name  
  
 class Meta:  
 verbose\_name = "商品分类管理"  
 verbose\_name\_plural = verbose\_name

1. 商品单位表

class Unit(models.Model):  
 *"""商品单位表"""* unit\_name = models.CharField(max\_length=20,  
 verbose\_name="商品单位")  
 add\_time = models.DateTimeField(auto\_now\_add=True, verbose\_name="创建时间")  
 update\_time = models.DateTimeField(auto\_now=True, verbose\_name="更新时间")  
 is\_delete = models.BooleanField(default=False, verbose\_name="是否删除")  
  
 def \_\_str\_\_(self):  
 return self.unit\_name  
  
 class Meta:  
 verbose\_name = "商品单位"  
 verbose\_name\_plural = verbose\_name

1. 商品spu表

class Goods\_spu(models.Model):  
 *"""商品spu表"""* spu\_name = models.CharField(max\_length=50,  
 verbose\_name="商品spu名")  
  
 content = RichTextUploadingField(verbose\_name="商品详情")  
 add\_time = models.DateTimeField(auto\_now\_add=True, verbose\_name="创建时间")  
 update\_time = models.DateTimeField(auto\_now=True, verbose\_name="更新时间")  
 is\_delete = models.BooleanField(default=False, verbose\_name="是否删除")  
  
 def \_\_str\_\_(self):  
 return self.spu\_name  
  
 class Meta:  
 verbose\_name = "商品spu名"  
 verbose\_name\_plural = verbose\_name

1. 商品sku表

class Goods\_sku(models.Model):  
 *"""商品sku表"""* trade\_name = models.CharField(max\_length=100,  
 verbose\_name="商品sku名")  
 breif = models.CharField(max\_length=200,  
 null=True,  
 blank=True,  
 verbose\_name="商品简介")  
 price = models.DecimalField(max\_digits=9,  
 decimal\_places=2,  
 verbose\_name="商品价格",  
 default=0)  
  
 stock = models.IntegerField(verbose\_name="商品库存",  
 default=0)  
  
 sales = models.IntegerField(default=0,  
 verbose\_name="商品销量")  
  
 logo = models.ImageField(upload\_to='goods/%Y%m/%d',  
 verbose\_name="封面图片")  
  
 is\_shelf = models.BooleanField(verbose\_name="是否上架",  
 choices=is\_on\_sale\_choices,  
 default=False)  
  
 unit = models.ForeignKey(to="Unit",  
 verbose\_name="单位")  
 commodity = models.ForeignKey(to="Commodity",  
 verbose\_name="商品分类")  
 goods\_spu = models.ForeignKey(to="Goods\_spu",  
 verbose\_name="商品spu")  
 activity = models.ForeignKey(to="Activity",  
 verbose\_name="活动分类")  
  
 add\_time = models.DateTimeField(auto\_now\_add=True, verbose\_name="创建时间")  
 update\_time = models.DateTimeField(auto\_now=True, verbose\_name="更新时间")  
 is\_delete = models.BooleanField(default=False, verbose\_name="是否删除")  
  
 def \_\_str\_\_(self):  
 return self.trade\_name  
  
 class Meta:  
 verbose\_name = "商品sku管理"  
 verbose\_name\_plural = verbose\_name

* + - 1. 页面交互

1. 在添加商品信息时，用户添加的图片能够显示在管理界面
2. 每一张表能有多个字段可以点击跳转到该表的管理界面，而不仅仅是只能通过点击id字段
   * 1. 要点难点及解决方案
        1. 如何在添加详细说明的时候能不仅仅只添加文字信息，还可以进行排版以及添加图片等等信息
        2. 解决方案:使用一个富文本插件：ckeditor后端编辑器
        3. Ckeditor是django的一个插件，配置好该插件后可以使用RichTextUploadingField属性，该属性可以对上传的信息进行编辑，并且能上传图片、排版、修改文字颜色等
3. 购物车功能模块
   * 1. 购物车商品添加
        1. 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
        2. 在商品分类页面，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
     2. 流程
        1. 前端通过ajax post请求方式，添加sku\_id（sku商品id）和count（商品的数量）
        2. 验证是否登录，如果没有登录，ajax中的js代码实现跳转到登录页面location.href=url
        3. 验证数据的合法性，必须都为整数
        4. 验证商品是否存在
        5. 验证商品库存是否充足
        6. 保存到redis中
     3. 设计要点(数据库和页面交互)
        1. 使用hsah保存对象
        2. Key:user\_id
        3. Field:sku\_id
        4. Value:count
        5. 页面交互
        6. 用ajax进行数据库交互以及页面跳转
     4. 要点难点及解决方案
   1. 流程
   2. 设计要点（数据库和页面交互）
   3. 要点难点及解决方案
4. 常见面试问题