# 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. 2用户登录功能模块
   1. 需求
      * 1. 用户填写手机号和密码，点击注册按钮后，数据提交；
        2. 对用户提交过来的数据进行校验，如果数据与数据库内的信息一致，登录成功，跳转到商城主页；
   2. 流程

用户填写数据；

填写的数据提交到form验证层进行基本的校验；

如果数据手机号不为空，字数没有问题，符合格式要求，并且密码字数没有问题，将数据提交给view视图函数， 对密码进行hash256加密；

将加密好的密码和手机号码与数据库内的数据进行比较，如果数据一致，用户登录成功。

* 1. 设计要点（数据库和页面交互）

1. 数据库的基本字段：

Id 主键

Create\_time 创建时间 datetime（用户注册的时间）

Update\_time 更新时间 datetime （用户修改信息的时间）

Is\_delete 是否删除 boolean （假删除）

Nickname 用户名 varchar

Telephone 手机号 varchar

Password 密码 varchar (哈希加密)

Gender 性别 choice (男， 女， 保密)

Birthday 生日 datetime

School 学校 varchar

Location 地址 varchar

Hometown 地址 varchar

2. 交互设计

1. 对文本框进行空字段校验，如果为空，登录失败，提示用户填写基本信息；
2. 对填写的字段进行基本判断，判断用户输入的字符长度是否符合规范；
3. 为文本框添加placeorder属性，提示用户需要填写的信息。
   1. 要点难点及解决方案

难点1：验证码的获取

解决方案：通过阿里云发送，申请一个模板和签名，当用户点击获取验证码时，阿里云向该手机号发送验证码

难点2：对用户填写的验证码进行校验

解决方案：通过ajax将用户从第三方平台得到的校验码获取到，用户填写校验码后将用户填写的和从第三方平台获取的进行比较，如果一致那么用户填写的校验码没有问题，如果不一致，则用户填写的校验码有错误，注册失败并提示用户。

难点3: 数据库的设置

解决方案:前期数据库没有设计好,导致后面需要添加一些字段的时候又重新迁移,十分麻烦.以后工作中,第一件事就是先把数据库整规矩.. 照到模板设置 ,要把不保存到数据库的字段设置为空 Model

难**点**4:form验证,错误类型提示

解决方案:借鉴强哥的操作 迁移时遇到: \_\_init\_\_() got an unexpected keyword argument 'Null'的错误 原因是把**null写成了Null,注意大小写 渲染模板的时候一步一步的来,有时候报错是乱报的 反向寻址的时候注意写法{%url ‘namespace:name ’%}**

**难点5:个人资料的更改和更新,更新后保存到数据库.日期对象的格式 电话号码要根据登录后session获得**

**解决方案:问吴娟的**

**点击获取验证码时,ajax没反应,原因是前端代码少些了一个花括号’}’ 解决办法:慢慢调试,看哪步错了**

1. 2产品功能模块
   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）

数据库

商品分类表：GoodsCategoriesModel

Classname 分类名 varchar

Classintro 分类简介 varchar

Is\_delete 是否删除 Boolean

Create\_time 添加时间 datetime

Update\_time 更新时间 datetime

商品单位表: GoodsUnitModel

Create\_time 添加时间 datetime

Update\_time 更新时间 datetime

Is\_delete 是否删除 Boolean

Unitname 单位名称 varchar

商品SPU: SpuModel

Create\_time 添加时间 datetime

Update\_time 更新时间 datetime

Is\_delete 是否删除 Boolean

Name 商品SPU名称 varchar

Desc 商品spu描述 RichTextUploadingField

商品SKU表：GoodsSkuModel

Create\_time 添加时间 datetime

Update\_time 更新时间 datetime

Is\_delete 是否删除 Boolean

Goodsname 商品信息 char

Goodsintro 商品介绍 text

Price 商品价格 decimal

Unitinfo 商品单位 foreignkey

Num 库存 positiveinteger

Sellnum 销售量 positiveinteger

Logo 图片 ImageField

Is\_putaway 上架 Boolean

Goodcate 商品分类 foreignkey

Goodsspu 商品spu分类 foreignkey

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. 如何将图片在后台管理界面中显示出来让用户能够预览到；

在设计数据库时，为表添加一个显示图片的函数，在该函数中添加图片的方法;

def show\_img(self):  
 return "<img src='%s%s'>"%(settings.MEDIA\_URL, self.img)  
show\_img.allow\_tags = True  
show\_img.short\_description = "商品图片"

2. 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；

使用一个富文本插件：ckeditor后端编辑器

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；

使用方法：

* + - 1. 1.在您的python路径中安装或添加django-ckeditor。
      2. pip install django-ckeditor
      3. 2.将ckeditor添加到您的INSTALLED\_APPS设置中。
      4. INSTALLED\_APPS = [
      5. 'django.contrib.admin',
      6. 'django.contrib.auth',
      7. 'django.contrib.contenttypes',
      8. 'django.contrib.sessions',
      9. 'django.contrib.messages',
      10. 'django.contrib.staticfiles',
      11. 'sp\_user.apps.SpUserConfig', # 用户模块
      12. 'sp\_goods.apps.SpGoodsConfig', # 商品模块
      13. 'ckeditor', # 添加ckeditor富文本编辑器
      14. ]
      15. 3.配置静态文件路径
      16. 看文档：https://docs.djangoproject.com/en/dev/howto/static-files/
      17. 删除之前配置的STATICFILES\_DIRS（本来上线就要删除），修改为STATIC\_ROOT
      18. #STATICFILES\_DIRS = [
      19. # os.path.join(BASE\_DIR, "static"),
      20. #]
      21. # 设置静态文件根目录 上线的时候使用
      22. STATIC\_ROOT = os.path.join(BASE\_DIR, "static")
      23. 使用以下命令，将django中安装在INSTALLED\_APPS的其他应用所有的静态文件收集到STATIC\_ROOT指定目录下
      24. python.exe manage.py collectstatic
      25. 这个时候 我们在 static 目录下就能看的 ckeditor 目录，里面就是ckeditor使用到的所有的静态文件（css,js,image）。
      26. 这个时候又将之前的修改改回去，现在还没有上线，还得用之前的配置。
      27. STATICFILES\_DIRS = [
      28. os.path.join(BASE\_DIR, "static"),
      29. ]
      30. # 设置静态文件根目录 上线的时候使用
      31. # STATIC\_ROOT = os.path.join(BASE\_DIR, "static")
      32. 4.设置ckeditor的上传目录
      33. 这个目录是相对目录，相对与 MEDIA\_ROOT
      34. CKEDITOR\_UPLOAD\_PATH = "uploads/"
      35. 将ckeditor\_uploader添加到您的INSTALLED\_APPS设置中。
      36. INSTALLED\_APPS = [
      37. 'django.contrib.admin',
      38. 'django.contrib.auth',
      39. 'django.contrib.contenttypes',
      40. 'django.contrib.sessions',
      41. 'django.contrib.messages',
      42. 'django.contrib.staticfiles',
      43. 'sp\_user.apps.SpUserConfig', # 用户模块
      44. 'sp\_goods.apps.SpGoodsConfig', # 商品模块
      45. 'ckeditor', # 添加ckeditor富文本编辑器
      46. 'ckeditor\_uploader', # 添加ckeditor富文本编辑器文件上传部件
      47. ]
      48. 5.在主路由中配置 ckeditor 上传文件使用到的url地址
      49. ckeditor上传文件内部机制使用 ajax 异步上传文件，需要配置上传访问地址才能被访问到。
      50. urlpatterns = [
      51. url(r'^admin/', admin.site.urls),
      52. # 上传部件自动调用的上传地址
      53. url(r'^ckeditor/', include("ckeditor\_uploader.urls")),
      54. # 用户模块
      55. url(r'^user/', include("sp\_user.urls", namespace="sp\_user")),
      56. # 其他模块
      57. ]
      58. 6.安装pillow(已经安装的不用重复安装)
      59. ckeditor上传文件使用pillow
      60. 7.设置ckeditor在后台显示的样式（显示效果）
      61. 使用默认配置就行，具体看文档修改
      62. # 编辑器样式配置
      63. CKEDITOR\_CONFIGS = {
      64. 'default': {
      65. 'toolbar': 'full',
      66. },
      67. }
      68. 8.将商品SPU表的详情字段修改RichTextUploadingField，使用ckeditor为们单独提供的字段类型，修改完后，后台自动显示富文本编辑器样式，可以上传图片
      69. # 导入ckeditor上富文本编辑器自带字段
      70. from ckeditor\_uploader.fields import RichTextUploadingField
      71. class GoodsSPU(BaseModel):
      72. """
      73. 商品SPU表
      74. """
      75. spu\_name = models.CharField(verbose\_name='商品SPU名称',
      76. max\_length=20,
      77. )
      78. # 使用ckeditor为我们提供的字段，不用重新迁移就可以
      79. content = RichTextUploadingField(verbose\_name="商品详情")
      80. def \_\_str\_\_(self):
      81. return self.spu\_name
      82. class Meta:
      83. verbose\_name = "商品SPU"
      84. verbose\_name\_plural = verbose\_name
      85. 9.前台模板显示

{{content|safe}}

* 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. 设计要点（数据库和页面交互）
       1. 数据库

首页轮播商品表：DisplayGoods

Name 名称 char

Skuinfo 商品sku信息 foreignkey

Pic 图片 image

Goods\_order 排序 smallint

Create\_time 创建时间 datetime

Updatetime 更新时间 datetime

Is\_delete 是否删除 Boolean

商品活动表：ActivateModel

Create\_time 创建时间 datetime

Updatetime 更新时间 datetime

Is\_delete 是否删除 Boolean

Name 名称 char

Img 商品图片 Image

url 商品地址 URL

首页专区活动和商品：ActivateGoodsModel

Create\_time 创建时间 datetime

Updatetime 更新时间 datetime

Is\_delete 是否删除 Boolean

Area 首页活动 foreignkey

Goods 商品 foreignkey

* + - 1. 页面交互

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。

难点2:数据库的创建,数据库的创建是很重要的,如果前期没有设计好的话,后面改起来很麻烦,所以先多花点时间在创建数据库上面

**难点2: 遍历的使用,在遍历时,for写在了li的里面 出现了很多小问题...自己解决了**

**难点3:在反向路由的时候,如果反过去的路由有参数,在url 后面要加上参数**

**{url ‘good:详情’ pk=goods\_id}**

难点4:连表查询,注意外键和关系.一对一,一对多,多对多

1. 常见面试问题