1. flask 部署: nginx + gunicorn + flask
2. Flask-Migrate扩展使得能用脚本管理同步数据库
3. blueprint实现每一部分的逻辑的分离
4. unit test单元测试

项目:

1. csrf_token: 以post/put/delete发送请求体的方式，请求体中body要携带与浏览器中的cookie携带的csrf_token相同的csrf_token: 黑客的脚本不能读取(不同域名/IP,协议或端口)对应的资源: 因为浏览器的同源策略
2. 整个项目目录下不同模块的分拆:
    a. 需求文档 requirements.txt
    b. 配置文件: config.py
    c. 项目启动脚本: manage.py
    d. 整个逻辑目录:
3. 工厂模式: 创建app不自己手动创建，工厂模式创建不同类型的app
4. **init**.py文件中, `csrf=CSRFProtect()` 放在定义函数外面，因为之前说过 `CSRFProtect()` 仅仅是添加了防护机制，但是具体生成cookie, body响应体中的csrf_token还是需要自己做: 既然后面要用到，定义在外面方便import
5. 如果初始化需要app，flask都支持在前面先定义，延迟初始化
6. **最后可以看出,manage.py这个启动文件中只剩下了了manage.run()中的manage对应的需要的Manager和Migrate模块，其他的都是项目中具体要做的，都放在逻辑板块中的init文件了**
7. 先都写在整个启动文件中: manage.py
    a. 从启动文件中拆分: 配置文件拆到config.py
    b. 项目逻辑初始化app相关的拆到**init**.py
    c. 启动项目最后剩下的都在manage.py
8. flask中单独config.py和django中单独settings.py一样的，移植到别的地方只用改配置就可以run了
9. bug: 循环导包: 项目启动文件--> 项目init文件 --> 应用init文件 --> 应用中的视图函数
10. 项目中的文件: static, util(重复的模块抽出来), libs(第三方包)
11. 日志: default访问信息放在terminal 但是为了专门记录，写在一个文件中: 记录访问最多的视图,访问波段 etc. a. `import logging` : 是python中的模块 b.flask中没有自己的logger, 但是current_app中封装了: `current_app.logger.error("xxx")`
12. 设计数据库的时候:
    a. `User` 表中的主键PK是user_id, 就算是手机号登录也不能用手机号作为user_id, 因为用户的手机号可能换了:==> 不能用用户填写的东西作为主键
    b. House表中不包含房子所属的区域信息, 因为房子可能很多，但是区域就只有10-30个之间，这样如果写SQL就是从很多中查找几十条，效率非常低，所以应该单独建立一个 `Area_info` 表用来存储区域信息
    c. House表中用一条冗余字段 `default_image_url` 节约要用的那一条要根据House

表关联查询House_image中的一张图放到列表页这个关联查询的时间和查询(空间换时间)

d. 房屋的配套设施: 本来应该放在House表中, 但是因为可能配套设施可能增加etc.: 单独放在一张表Facility中以免今后改变House表的字段: 类似house_image单独放在一张表中
e. House和Facility两张表是多对多的关系: 一个房子可能有多个设施, 一个设施可能多个房子有: 两张表通过一个中间表House_Facility都是一对多的关系建立多对多的关系
f. 房子的评分: 用这个方子被下的订单数代替: 但是如果从Orders表中由拿到的house_id关系查询sum再排序就效率很低: 用House中的冗余字段order_counts解决问题
g. Orders订单表中的comment实际上是"评价信息"和"拒绝原因"这两个互斥字段其中的一个