2018

数据库课程设计报告

火车票售票系统

软件学院 工科试验 AI16 班

贾 栋 201600301129

InnoHub 票务中心产品文档 *Demo*

修订记录

版本	修订人	日期	编写/修订内容
0.1Base	贾栋	2018.09.10	初步确立需求，明确基本架构
0.2Base	贾栋	2018.09.11	确定 ER 图，建立基本数据模式
0.1Alpha 1	贾栋	2018.09.13	进行数据分析，获取数据
1.0Beta	贾栋	2018.09.17	实现基本功能，进行简单测试
1.0Demo	贾栋	2018.09.21	系统测试，优化用户交互，完善文档

目录

一、 系统开发平台

操作系统	Linux
开发工具	Vim + Google Chrome
题目	火车票购票系统
数据库	Sqlite3

二、 数据库规划

1 任务陈述

为实现对于真实购票系统的高度模拟，实现对于 12306.cn 高仿真再造，本系统采用较为真实的数据进行功能实现，通过网络爬虫抓取 12306.cn 数据，通过筛选整理，导入到本系统的数据库中，再基于此进行用户功能的模拟。
本系统至少应包括但不限于：余票查询，购票，退票，改签，用户注册，登录，中转查询，管理员进行后台数据的管理以及更新，不同角色的管理员等

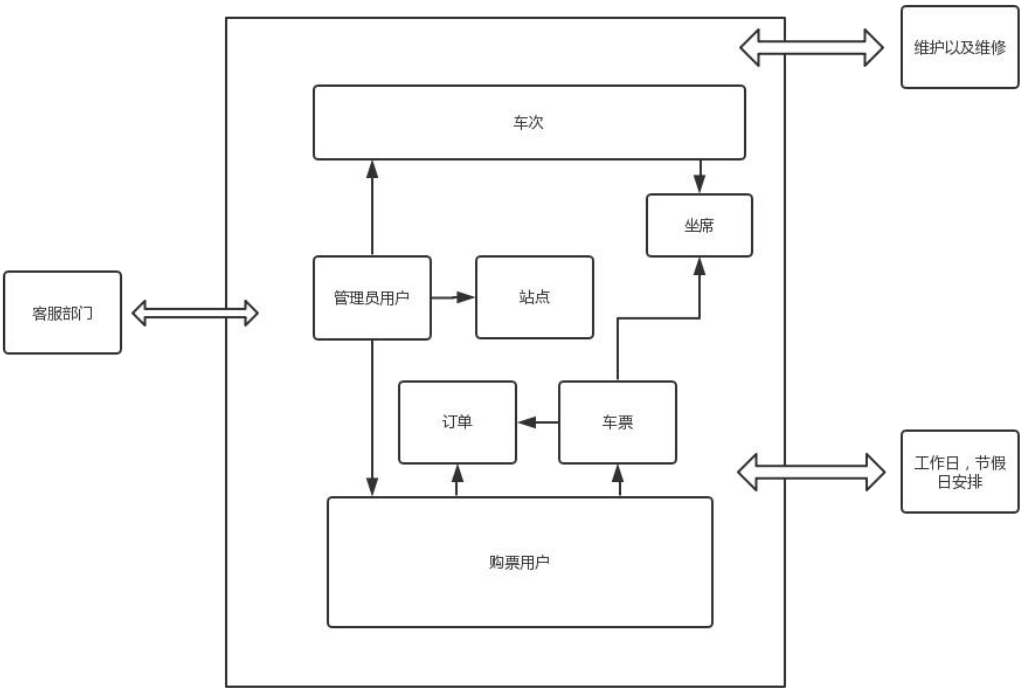
2 任务目标

对于大量数据进行清洗，整理，在系统数据库中存储预售期为一个月的数据量，通过半自动化的方式导入数据；在保证数据量的同时力求系统的运行效率，减少查询等待时间，保证业务的正常执行

在此基础上力求实现系统的长期可用，即实现历史数据的转移，每天新的预售数据的自动生成；实现对于用户购票行为的报表分析，对于不同车次的上座率，客户量的报表分析，从而可以向铁路部门提供直接有效的数据资料，用于更改铁路车次，改变车厢数目，改变车票定价等决策行为中去

三、 系统定义

1 系统边界



四、 需求分析

1 用户需求说明

1.1 用户划分

InnoHub 票务中心是一个仿 12306.com 的铁路购票系统，是一个基于 Django 框架的 web 应用，旨在通过构建必要的数据模式模拟用户网上购票的行为，网站管理员的后台管理行为。为**购票用户**以及**管理员用户**提供简单便捷的 web 服务。所以将基本用户划分为以下两类

1.2 购票用户

余票查询	1. 购票人可以通过在查询页面输入所要查询的车次的时间，出发地，目的地，获取符合当前查询条件的所有车次信息 2. 车票信息要包含：车次，车票类型（二等座，商务座，软卧…），出发到达站，出发到达时间，车票金额,车票和订单都是客观存在的实体，为了便于进行订单管理等操作所以要分别建立单独的表 3. 在余票查询界面提供简单直接的判定，对于存在余票的坐席类型提供购票入口，其他坐席类型则不可以进行购票操作
车票操作	1. 购票人可以进行基本的购票，改签，退票操作，对于每次操作都有相应的相应的订单生成但是只有进行购票，改签操作才会生成

	新的车票对象，进行退票以及改签时相应的旧票都会从数据库 2. 购票人完成相应的操作后，会对其绑定的邮箱账户进行提示 3. 车票模拟实体车票，数据表中需要待打印车票包含所有必要信息
账户管理	1. 注册：模拟实名制购票原则，需要使用邮箱进行用户注册，需要提供相应位数的身份证号码 2. 账户信息的更新，如改密，更新邮箱，修改身份信息 3. *TODO*：时间充裕时可以尝试引入验证码登录等功能
订单查询	1. 可以查询全部订单，订单有五种状态，完成、取消、待付款、可退款可改签、仅可退款 2. 查询结果需要包括订单号，乘车人，车次，车票类型（二等座，商务座，软卧…），出发到达站，出发到达时间，车票金额

1.3 管理员用户

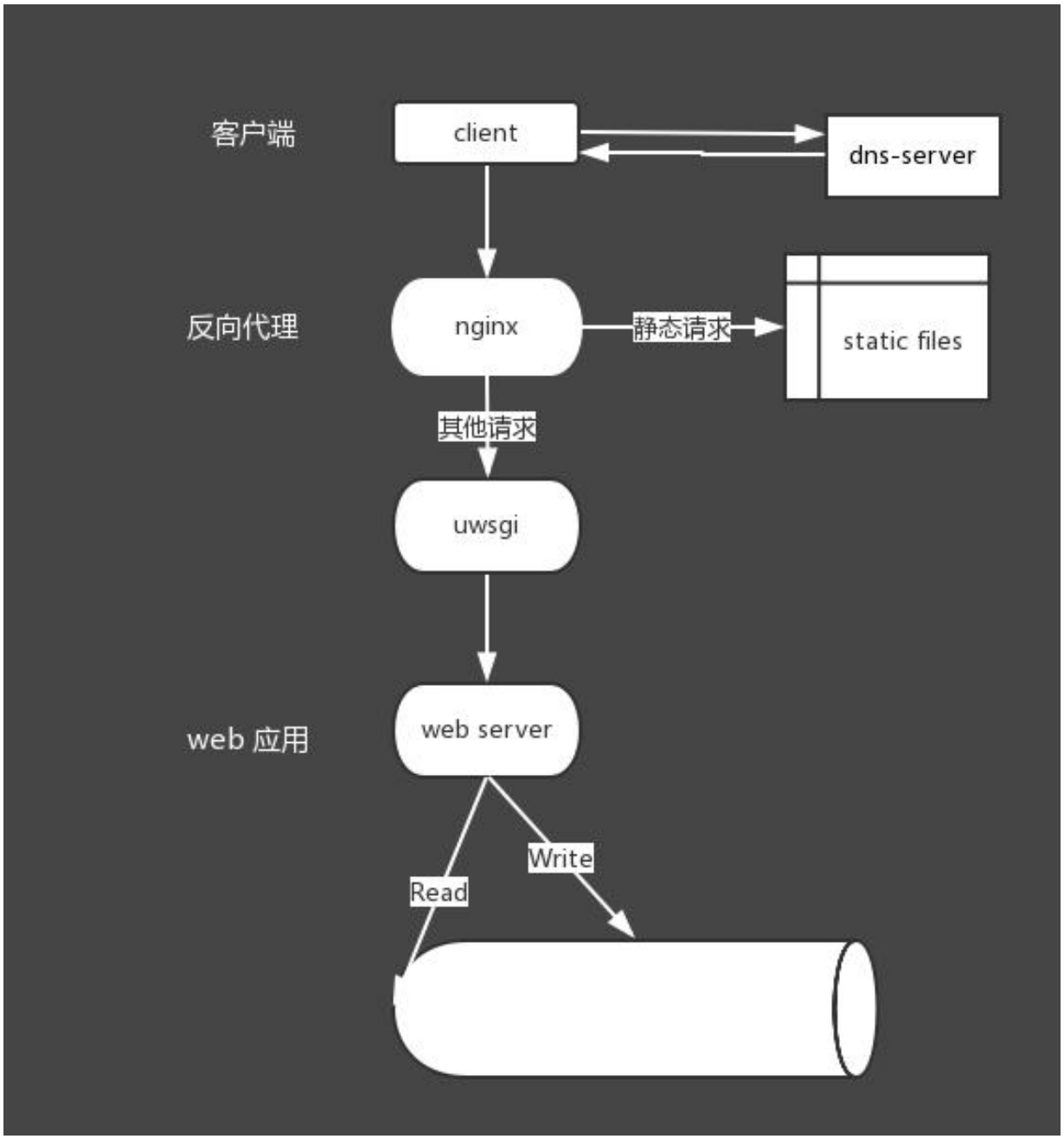
车次管理	**1.** 管理员用户可以直接通过相应的后台修改车次信息，包括增删改等基本操作，该操作会直接映射到数据库中 **2.** 对于车次的基本信息至少包含：车次，起止地点，起止时间，经停站
坐席管理	1. 可以直接根据实际情况修改相应车次的坐席信息，包括坐席类型，车厢，座位号 2. 当有多个管理员用户进行数据修改时要保证 ACID，同时 super admin 进行数据修改时，普通管理员不可以进行修改
用户管理	1. 对于 super admin 用户可以创建普通管理员用户，赋予相应权限，进行后台操作；但是普通管理员只可以进行车次调整，只有超级管理员可以添加管理员用户 2. Super admin 具有添加铁路黑名单的权限，可以将具有违法行为的购票用户加入铁路黑名单，禁止其进行购票

注：管理员分为 Super admin ， common admin 超级管理员具有最高权限

2 系统需求说明

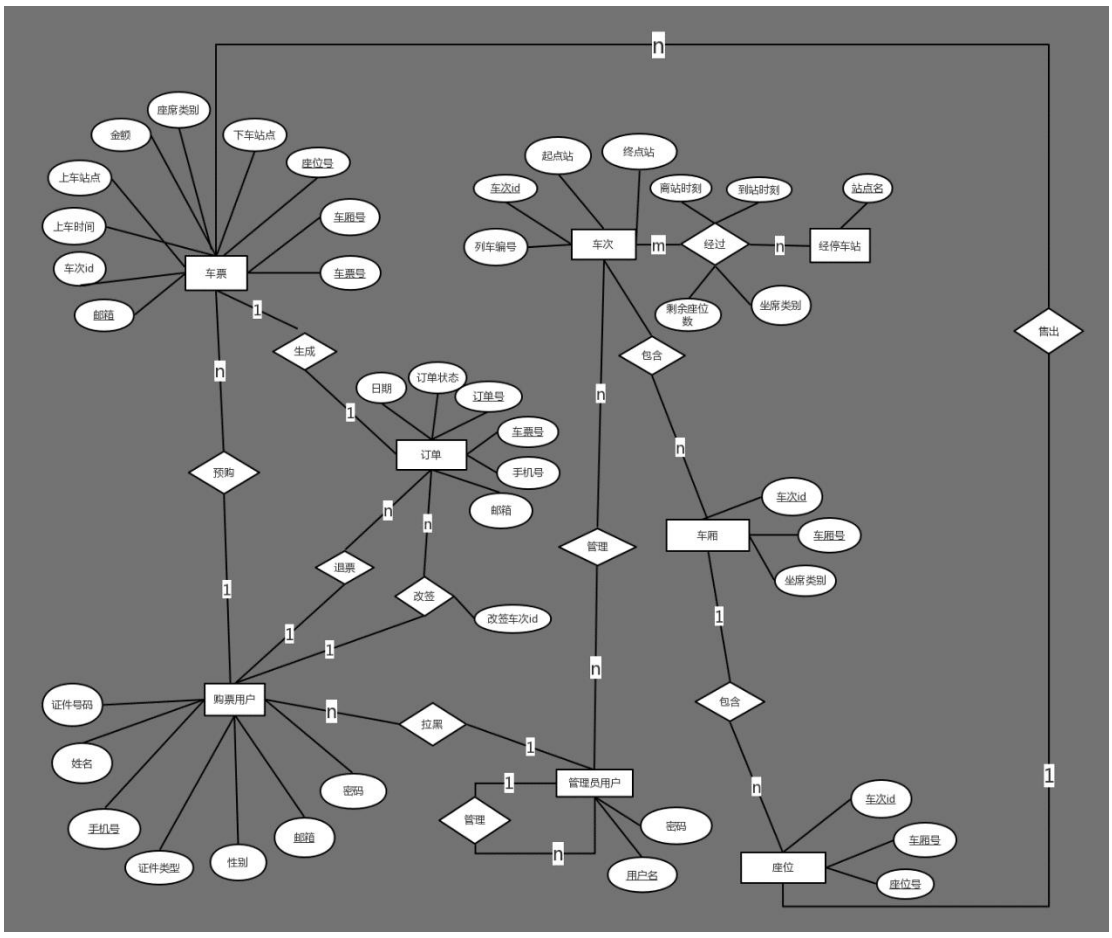
系统描述	本系统采用 MTV（model template view）设计模式，采用开源的 Postgresql 数据库，通过 uwsgi 配置在 nginx 服务器上，是一个典型的 Django 应用，可以为两类用户提供简单便捷的服务
安全	利用 nginx 作为反向代理服务器，只缓存一些静态数据，其它请求再向下分发，提高了安全性，同时，可以在 nginx 和 web 应用间添加防火墙，减少安全隐患（只有一个 web 应用，无法体现负载均衡）
可维护性	1. 根据 MTV 的架构，通过建立基本数据模型（model），可以直接生成相应的数据模式，建立相应的表结构，减少了可能出现的错误 2. 当一个数据模式需要更改时，可以直接修改相应的 model，再进行

	migrate 即可，降低了维护成本
分层架构	使用常见的互联网分层架构： Client <-> nginx <-> uwsgi <-> Django
数据要求	本系统采用对于 12306 系统的高仿真模拟，车票预售期为 30 天，通过从 12306 网站上爬取相应的车次，站点，经停站，坐席，价格等信息，采用高仿真的数据进行系统支持，现存有站点 2800 余个，车次信息 50 万条，车厢信息 11 万条，坐席信息 550 余万条，为方便检索，经停站信息 60 余万条



五、 数据库逻辑

1 ER 图



2 转化为关系模式

车票（<u>车票号</u>，车次 id， 邮箱， 车厢号， 座位号， 上车时间， 上车站点，下车时间，下车站点， 金额， 坐席类别）
购票用户（<u>手机号， 邮箱</u>， 姓名， 证件号码， 证件类型， 性别， 密码）
订单（<u>订单号，</u> 车票号， 手机号， 邮箱， 订单状态， 日期， 改签车次 id， 金额）
车次（<u>车次 id， 列车编号</u>， 起点站， 终点站）
车厢（<u>车次 id， 车厢号，</u> 坐席类别）
座位（<u>车次 id， 车厢号， 座位号</u>）
站点（<u>站点名</u>）
管理员用户（<u>用户名</u>， 密码， isSuper）

车次管理（<u>车次 id</u>， <u>管理员用户名</u>）
经过（<u>车次 id， 站点名</u>， 离站时刻， 到站时刻， 坐席类别， 剩余座位数）

3 数据字典

实体	属性	描述	键	类型	是否 null	多值	约束
车票	车票号	候选码	主码	Varchar(100)	N	N	
	车次 id	N:1 关系		Varchar(80)	N	N	此 4 项可以唯一标示一张车票
	邮箱	N:1 关系		Varchar(100)	N	N	
	车厢号	N:1 关系		Varchar(20)	N	N	
	座位号	N:1 关系		Varchar(20)	N	N	
	上车时间	车票显示		Datetime	N	N	
	上车站点	车票显示		Varchar(100)	N	N	
	下车时间	车票显示		Datetime	N	N	
	下车站点	车票显示		Varchar(100)	N	N	
	金额	车票显示		Decimal(10, 2)	N	N	最多表示 10 位数字，小数点保留 2 位
	坐席类别	车票显示		Varchar(3)	N	N	

实体	属性	描述	键	类型	是否 null	多值	约束
购票用户	手机号	候选码	主码	Varchar(80)	N	N	可以作为用户名登录
	邮箱	候选码	主码	Varchar(80)	N	N	
	姓名	实名制		Varchar(100)	N	N	
	证件类型	24 种可用证件		Varchar(6)	N	N	
	证件号码	实名制		Varchar(80)	N	N	
	性别	F/M		char(1)	N	N	
	密码	登录密码		Varchar(100)	N	N	

实体	属性	描述	键	类型	Null	约束
订单	订单号	候选码	主码	Varchar(100)	N	

	车票号	候选码		Varchar(100)	N	
	手机号	订单反馈		Varchar(80)	N	
	邮箱号	订单反馈		Varchar(80)	N	
	订单状态	包括完成，取消，可退可改签，仅可退款，待付款		Varchar(6)	N	
	日期	订单反馈		Datetime	N	
	改签车次id	没有进行改签操作时为null		Varchar(80)	Y	
	金额	订单反馈		Decimal(10, 2)	N	

实体	属性	描述	键	类型	Null	约束
车次	车次id	候选码	主码	Varchar(80)	N	
	列车编号	列车名称		Varchar(20)	N	
	起点站			Varchar(100)	N	
	终点站			Varchar(100)	N	

实体	属性	描述	键	类型	Null	约束
车厢	车次id	所属车次	主码	Varchar(80)	N	
	车厢号		主码	Varchar(20)	N	
	坐席类型	每个车厢的坐席类型一致		Varchar(3)	N	

实体	属性	描述	键	类型	Null	约束
座位	车次id	所属车次	主码	Varchar(80)	N	
	车厢号		主码	Varchar(20)	N	
	座位号		主码	Varchar(2	N	

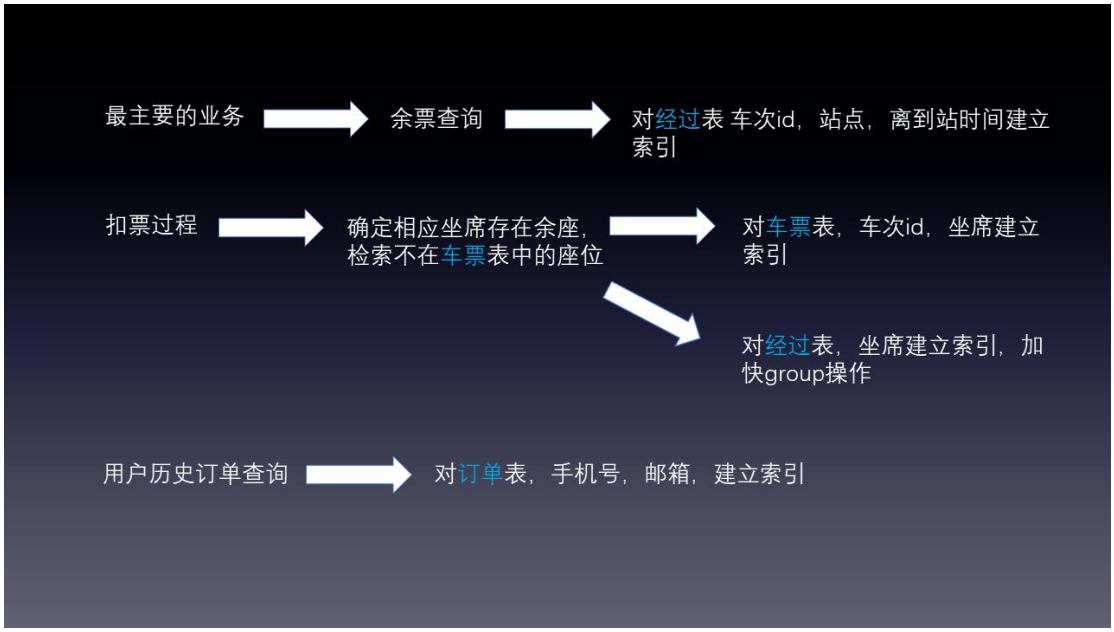
				0)		

实体	属性	描述	键	类型	Null	约束
经停站	车次 id	记录每个车次在每一站的剩余座位，便于进行查询	主码	Varchar(80)	N	
	站点名		主码	Varchar(100)	N	
	到站时刻			Datetime	N	
	离站时刻			Datetime	N	
	坐席类别			Varchar(3)	N	
	剩余座位			integer	N	

六、 数据库物理设计

1 外码建立

除了主码建立索引外，尝试对 where,on,group by,order by 中出现的列使用索引



2 安全机制

2.1 数据库安全

由于本系统是基于 django 框架的 web 应用，使用的嵌入式数据库 Sqlite3 具有高效率，快存储，占用内存小等优点，但是由于所有的数据是写在一个文件中，所有具有相应权限的用户都可以进行数据更改，缺少数据库的用户机制，所以在安全性方面需要多加考

虑

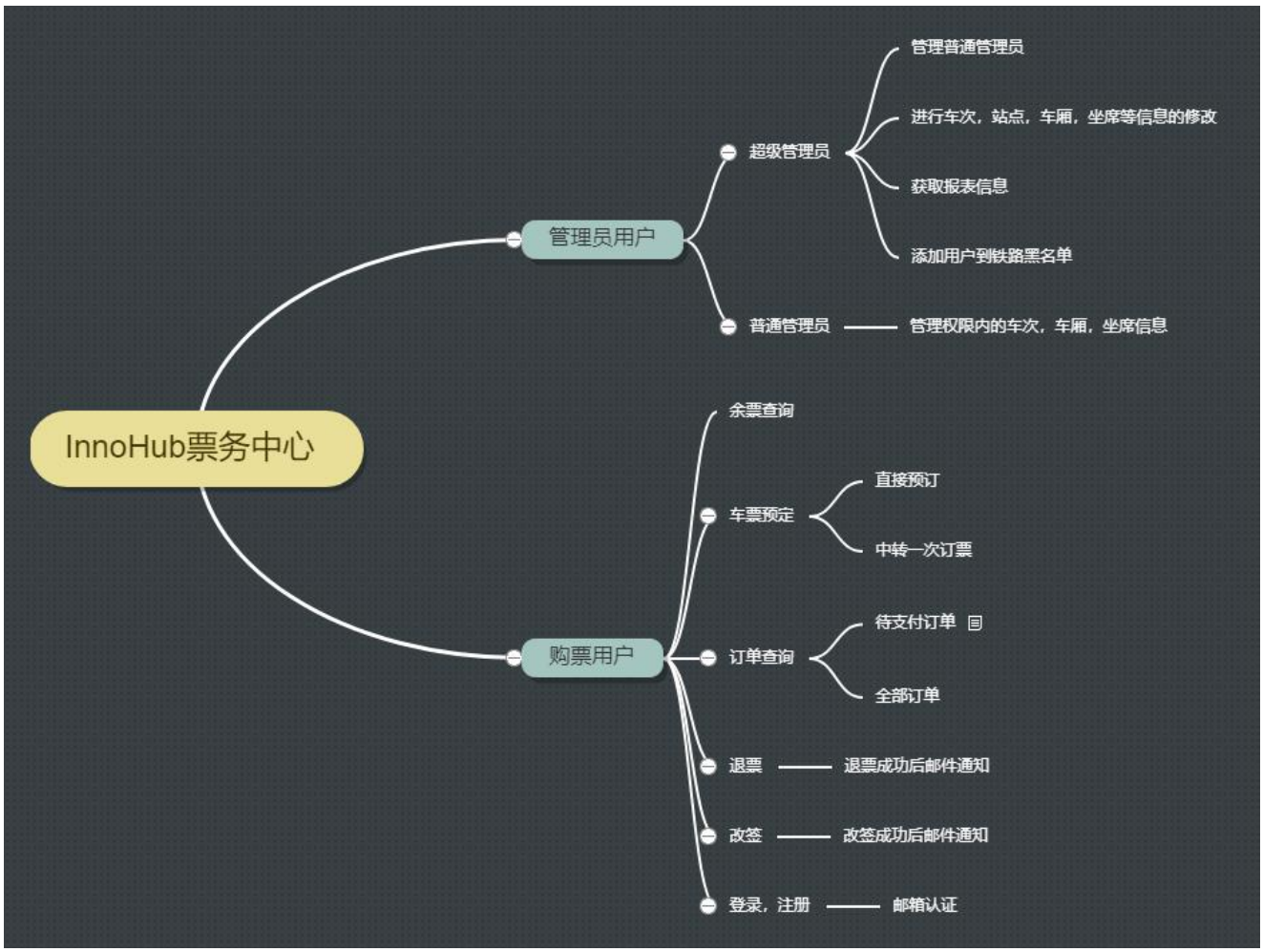
口令认证	对于 sqlite3 数据库的访问依赖于对于文件访问的控制，对于数据库进行创建，查询，增删改等操作必须提供正确的口令
数据库加密	为保持数据存取效率，对于部分敏感数据进行应用层的加密，比如用户密码，手机号，email 采用密文存取的方式
审计机制	由于 sqlite3 不宜调用日志执行审计功能，所以使用特定的文本文件来记录系统的重要事件，如打开数据库，修改口令等
备份和恢复	对于 sqlite 文件进行定时的拷贝备份

2.2 系统安全

本系统通过反向代理服务器及其与应用层的 web server 之间的防火墙实现系统安全防护，通过 nginx 服务器接受所有的请求，再进行分发的方式，避免了对应用服务的直接请求，减少了对于数据库直接访问的可能性，同时在 nginx 和 Django 程序中再加一道防火墙，大大提高了整个系统的安全系数

七、 应用程序设计

1 功能模块

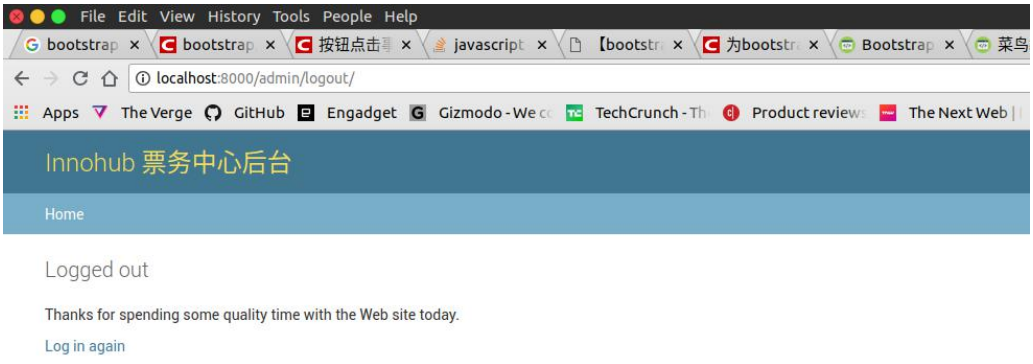


2 界面设计

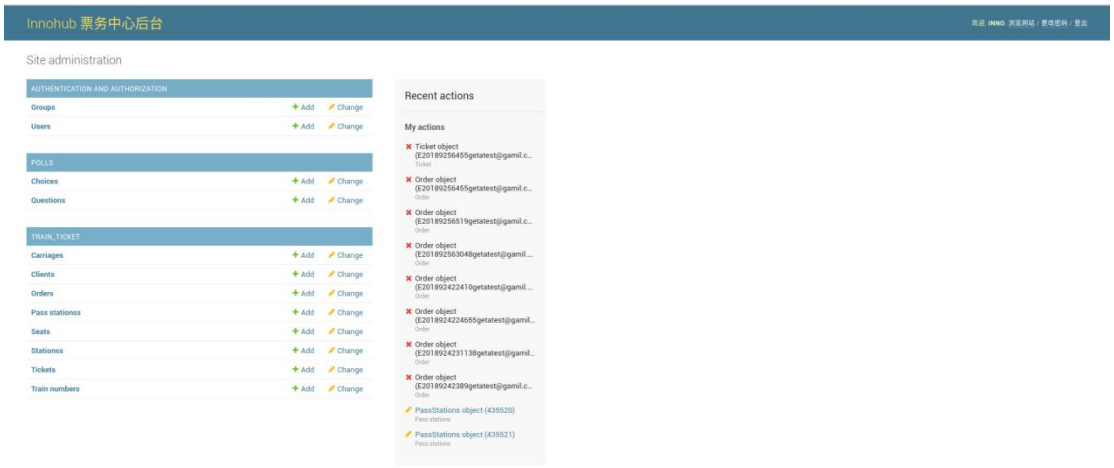
2.1 后台界面

登录登出界面

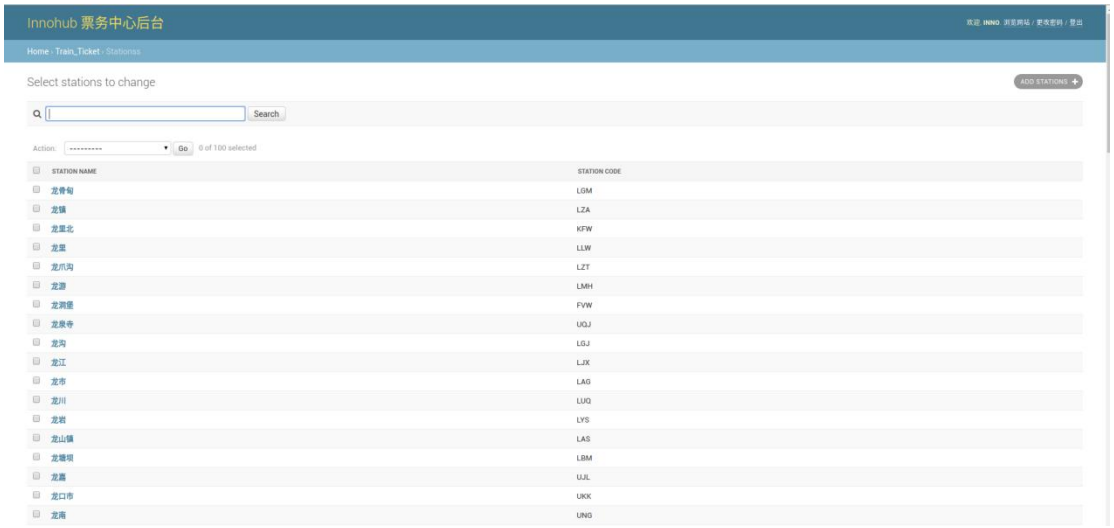

Innohub 票务中心后台
Username:
inno
Password:
Log in


File Edit View History Tools People Help
bootstrap
bootstrap
按钮点击
javascript
【bootstr
为bootstr
Bootstrap
菜鸟
localhost:8000/admin/logout/
Apps
The Verge
GitHub
Engadget
Gizmodo - We
TechCrunch - Th
Product review
The Next Web
Innohub 票务中心后台
Home
Logged out
Thanks for spending some quality time with the Web site today.
Log in again

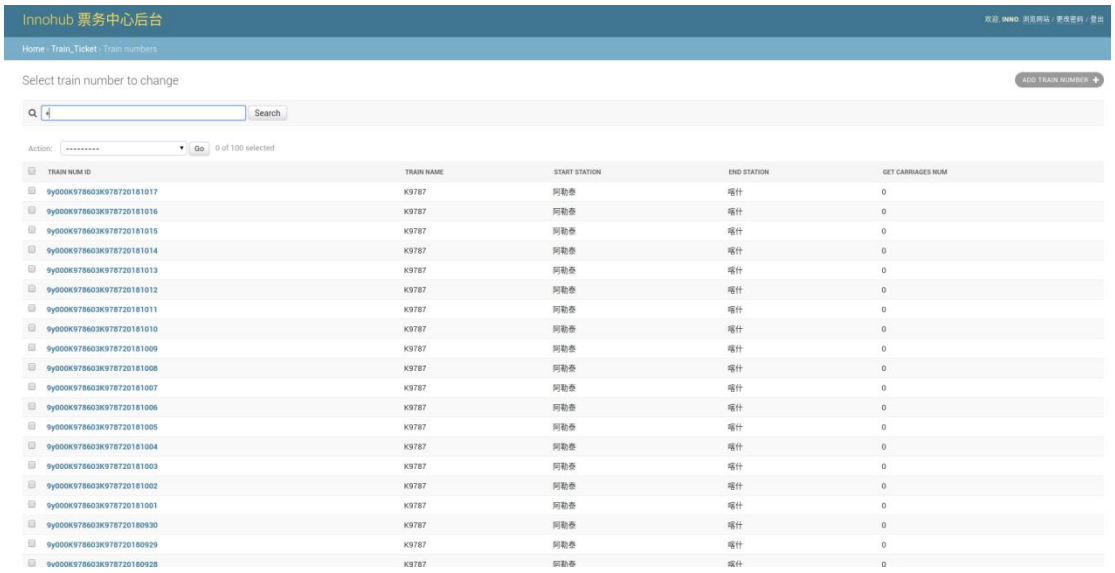
主页



站点列表

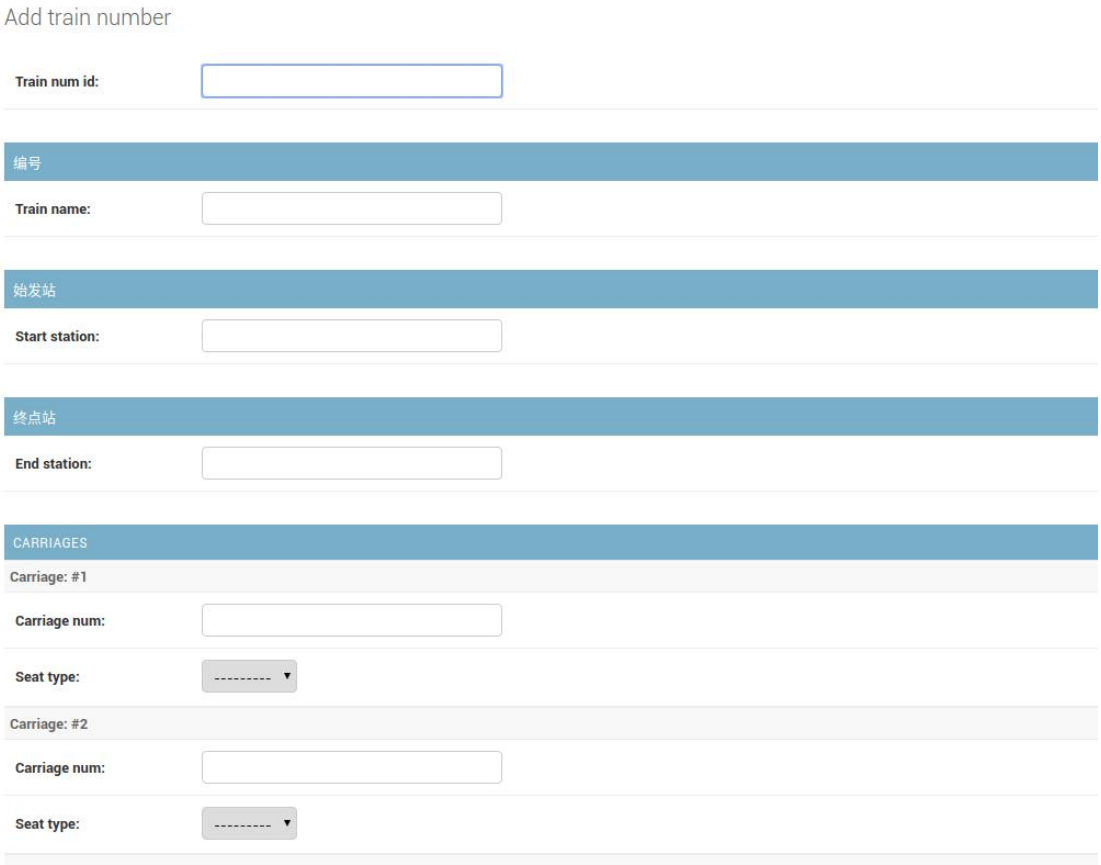


车次列表



车次信息详情

添加新的车次信息

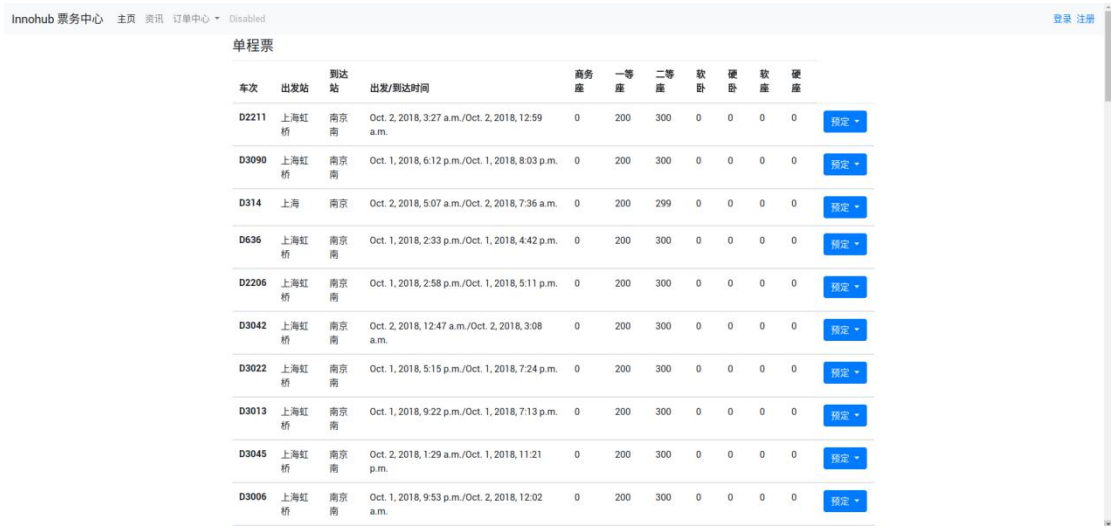


2.2 购票用户界面

登录以及注册

余票查询界面

车次列表以及预订

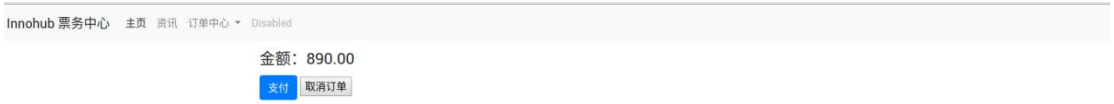


D3136	上海虹桥	南京	Oct. 2, 2018, 1:45 a.m./Oct. 2, 2018, 4:28 a.m.	0	200	300	0	0	0	0	预定
D3032	上海虹桥	南京南	Oct. 1, 2018, 11:16 p.m./Oct. 2, 2018, 1:22 a.m.	0	200	300	0	0	0	0	预定
D3126	上海虹桥	南京	Oct. 2, 2018, 2:47 a.m./Oct. 2, 2018, 5:16 a.m.	0	200	300	0	0	0	0	预定
D3026	上海虹桥	南京南	Oct. 1, 2018, 9:27 p.m./Oct. 1, 2018, 11:32 p.m.	0	200	300	0	0	0	0	预定

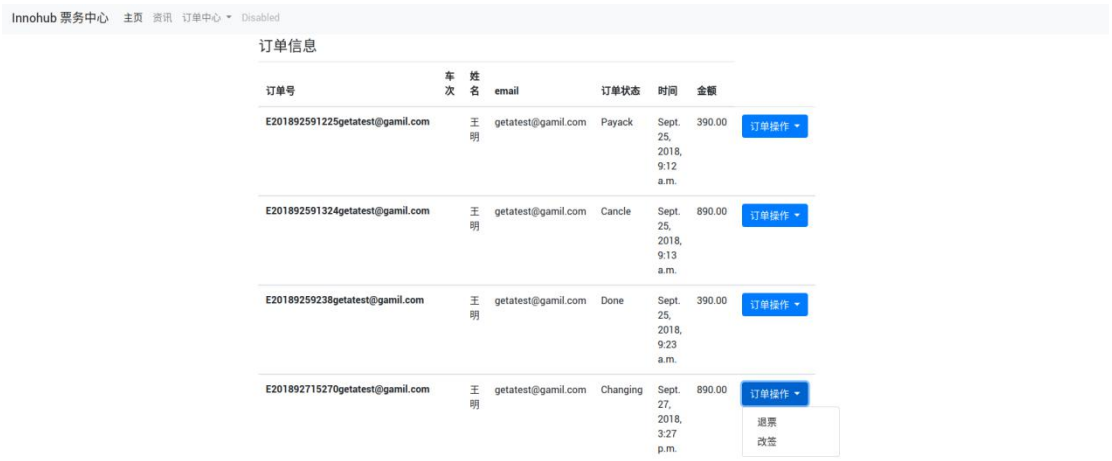
返程票

车次	出发站	到达站	出发/到达时间	商务座	一等座	二等座	软卧	硬卧	软座	硬座	
D956	南京南	上海	Oct. 6, 2018, 7:16 p.m./Oct. 6, 2018, 5:30 p.m.	0	200	300	0	0	0	0	预定
D351	南京南	上海虹桥	Oct. 7, 2018, 4:13 a.m./Oct. 7, 2018, 6:19 a.m.	0	200	300	0	0	0	0	预定
D3005	南京南	上海虹桥	Oct. 6, 2018, 8:46 p.m./Oct. 6, 2018, 11:15 p.m.	0	200	300	0	0	0	0	预定
D3052	南京南	上海虹桥	Oct. 7, 2018, 1:56 a.m./Oct. 6, 2018, 11:57 p.m.	0	200	300	0	0	0	0	预定
D321	南京	上海	Oct. 6, 2018, 2:40 p.m./Oct. 6, 2018, 5:09 p.m.	0	200	300	0	0	0	0	预定

支付



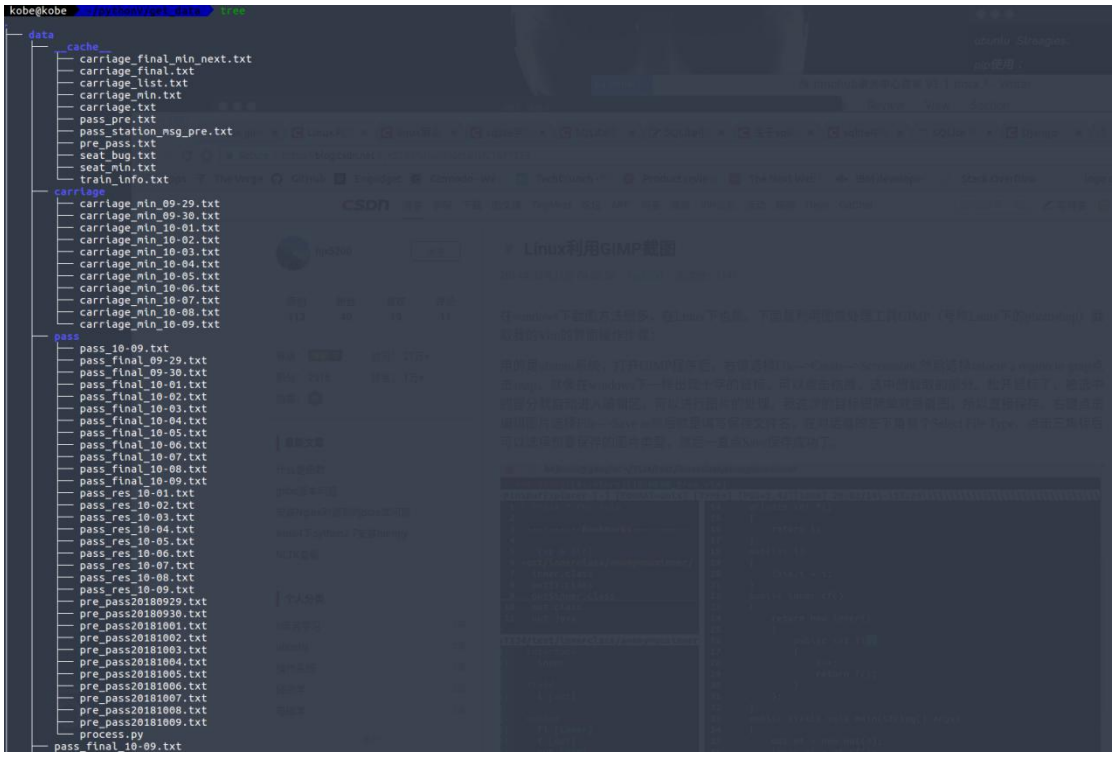
订单查询



3 数据集整理

由于本系统的数据高仿真要求，所以需要进行数据爬取以及整理，将规格化的数据导入到数据库中，全部数据集包含站点 2800 余个，车次 28 万以上，最大的表超过 550 万行，所以在数据收集整理方面花费了大量精力和时间，所有原始数据都是以文本格式存放，按照数据库中的表结构进行组织，方便进行数据导入。关于数据集以及其收集整理过程如下：

3.1 数据集原始文件组织结构



3.2 径行站数据获取：

```
def get_url(train_no, from_station, to_station, date):
    url = ('https://kyfw.12306.cn/otn/czxx/queryByTrainNo?'
           'train_no={}&from_station_telecode={}&'
'to_station_telecode={}&depart_date={}').format(train_no,from_station,station,d
ate)
    print(url)
    return url


def get_pass_stations(train_no, from_station, to_station, date):
    r = requests.get(get_url(train_no, from_station, to_station, date),
verify=False)
    raw_train = r.json()['data']['data']
    train_name = raw_train[0]['station_train_code']
    for item in raw_train:
        station_name = item['station_name']
        arrival_time = item['arrive_time']
        start_time = item['start_time']
        station_no = item['station_no']
        result = [train_no, station_name, arrival_time, start_time, station_no,
train_name]
        yield result
```

3.3 车次信息

```
def parse_train_info(file_path):
    with open(file_path, 'r') as file:
        content = file.read()
    date_sets = re.findall(r'"([0-9]{4}-[0-9]{2}-[0-9]{2})":\{', content)
    train_list = re.findall(r'=(.*)', content)[0]
    train_json = json.loads(train_list)

    trains = []
    for date in date_sets:
        json_date = train_json.get(date)
        keys = json_date.keys()
        for key in keys:
            dicts = json_date.get(key)
            for dic in dicts:
```

```
                station_train_code = dic.get('station_train_code')
                station_msg = re.split(r'[()-]', station_train_code)
                train_name = station_msg[0]
                start_station = station_msg[1]
                end_station = station_msg[2]
                train_no = dic.get('train_no')
                trains.append([train_no, train_name, start_station, end_station,
date])

    return trains
```

3.4 车厢数据处理

```
def get_carriages():
    with open('./train_info_final.txt') as f:
        lines = f.readlines()
    index_t = 1
    for index in range(1000):
        line = lines[index]
        contents = line.split('|')
        train_name = contents[1]
        if train_name[0] == 'G':
            cont = ''
            for index in range(1, 5):
                cont += '{}|c{}'.format(index_t, index) + '|' + 'SWZ' + '|' +
contents[0] + '\n'
                index_t += 1
            for index in range(5, 10):
                cont += '{}|d{}'.format(index_t, index) + '|' + 'YDZ' + '|' +
contents[0] + '\n'
                index_t += 1
            for index in range(10 ,15):
                cont += '{}|e{}'.format(index_t, index) + '|' + 'EDZ' + '|' +
contents[0] + '\n'
                index_t += 1
            cont += '{}|e{}'.format(index_t, 16) + '|' + 'EDZ' + '|' + contents[0]
            print(cont)

        elif train_name[0] == 'D':
            cont = ''
            for index in range(1, 5):
                cont += '{}|c{}'.format(index_t, index) + '|' + 'YDZ' + '|' +
contents[0] + '\n'
```

```
                index_t += 1
            for index in range(5, 10):
                cont += '{}|d{}'.format(index_t, index) + '|' + 'EDZ' + '|' +
contents[0] + '\n'
                index_t += 1
            cont += '{}|e{}'.format(index_t, 10) + '|' + 'EDZ' + '|' + contents[0]
            print(cont)

        elif train_name[0] == 'T':
            cont = ''
            for index in range(1, 5):
                cont += '{}|c{}'.format(index_t, index) + '|' + 'RZ' + '|' +
contents[0] + '\n'
                index_t += 1
            for index in range(5, 10):
                cont += '{}|d{}'.format(index_t, index) + '|' + 'YZ' + '|' +
contents[0] + '\n'
                index_t += 1
            for index in range(10 ,15):
                cont += '{}|e{}'.format(index_t, index) + '|' + 'YW' + '|' +
contents[0] + '\n'
                index_t += 1
            cont += '{}|e{}'.format(index_t, 16) + '|' + 'YW' + '|' + contents[0]
            print(cont)

        elif train_name[0] == 'Y':
            cont = ''
            for index in range(1, 5):
                cont += '{}|c{}'.format(index_t, index) + '|' + 'SWZ' + '|' +
contents[0] + '\n'
                index_t += 1
            for index in range(5, 10):
                cont += '{}|d{}'.format(index_t, index) + '|' + 'YDZ' + '|' +
contents[0] + '\n'
                index_t += 1
            for index in range(10 ,15):
                cont += '{}|e{}'.format(index_t, index) + '|' + 'RZ' + '|' +
contents[0] + '\n'
                index_t += 1
            cont += '{}|e{}'.format(index_t, 16) + '|' + 'RZ' + '|' + contents[0]
            print(cont)
        if train_name[0] == 'K':
            cont = ''
            for index in range(1, 5):
```

```
                cont += '{}|c{}'.format(index_t, index) + '|' + 'RW' + '|' +
contents[0] + '\n'
                index_t += 1
            for index in range(5, 10):
                cont += '{}|d{}'.format(index_t, index) + '|' + 'YW' + '|' +
contents[0] + '\n'
                index_t += 1
            for index in range(10 ,15):
                cont += '{}|e{}'.format(index_t, index) + '|' + 'YZ' + '|' +
contents[0] + '\n'
                index_t += 1
            cont += '{}|e{}'.format(index_t, 16) + '|' + 'YZ' + '|' + contents[0]
            print(cont)
        else:
            cont = ''
            for index in range(1, 5):
                cont += '{}|c{}'.format(index_t, index) + '|' + 'EDZ' + '|' +
contents[0] + '\n'
                index_t += 1
            for index in range(5, 10):
                cont += '{}|d{}'.format(index_t, index) + '|' + 'YZ' + '|' +
contents[0] + '\n'
                index_t += 1
            cont += '{}|e{}'.format(index_t, 10) + '|' + 'YZ' + '|' + contents[0]
            print(cont)
```

4 MTV 具体设计

Django 项目使用 MTV 设计模式，简化了对于数据库的操作，提高了项目的可维护性，便于在实现过程中进行数据模型的更改，提高了设计的灵活性。其中 Models 即数据模型，规定了数据库中每个表的列以及相应的域；templates 用于规定前端页面的模板，可以接受用户操作，将服务器的数据进行格式化展示；views 用于处理具体的事务时对应的实现，负责 web 请求以及反馈，是用户所用的主要功能。

4.1 models

```
class Client(models.Model):
    email = models.CharField(max_length=100)
    phone = models.CharField(max_length=60)
    name  = models.CharField(max_length=80, blank=False)
```

```
    card_num = models.CharField(max_length=120, blank=False)
    CARD_TYPE_LIST = (
        ('EDJ','二代居民身份证'),
        ('LSS', '临时身份证'),
        ('HKB', '户口簿'),
        ('JRB', '中国人民解放军军人保障卡'),
        ('JGZ', '军官证'),
        ('WJJ','武警警官证'),
        ('SBZ','士兵证'),
        ('JDX','军队学员证'),
        ('JDW','军队文职干部证'),
        ('JDL','军队离休干部证'),
        ('HZ','护照'),
        ('GAJ','港澳居民来往内地通行证'),
        ('GAT','中华人民共和国来往港澳通行证'),
        ('TWJ','台湾居民来往大陆通行证'),
        ('DLJ', '大陆居民来往台湾通行证'),
        ('WJL','外国人居留证'),
        ('RJZ','外国人出入境证'),
        ('WJG','外交官证'),
        ('LSG','领事馆证'),
        ('HYZ','海员证'),
        ('WGR','外交部开具的外国人身份证明'),
        ('DFG','地方公安机关出入境管理部门开具的护照报失证明'),
        ('TLG', '铁路公安部门填发的乘坐旅客列车临时身份证明'),
        ('XSZ','身高不足1.5m的未成年人学生证')
    )
    card_type    =    models.CharField(max_length=3,    choices=CARD_TYPE_LIST,
blank=False)
    SEX_LIST = (
        ('M', '男'),
        ('F', '女'),
    )
    sex = models.CharField(max_length = 1, choices=SEX_LIST)
    passwd = models.CharField(_('password'), max_length=128)



    class Meta:
        # 设置联合主码
        unique_together = ('email', 'phone')



class Ticket(models.Model):
```

```
    ticket_num = models.CharField(max_length=100, primary_key=True)
    train_num_id = models.CharField(max_length=80)
    email = models.CharField(max_length=100)
    carriage_num = models.CharField(max_length=20)
    seat_num = models.CharField(max_length=20)
    board_time = models.DateTimeField('board time')
    board_station = models.CharField(max_length=20)
    detrain_time = models.DateTimeField('detrain time')
    detrain_station = models.CharField(max_length=20)
    price = models.DecimalField(max_digits=10, decimal_places=2)
    seat_type = models.CharField(max_length=20)

    # 多对一关系
    ticket_client = models.ForeignKey(Client, on_delete=models.CASCADE)

    class Meta:
        unique_together = ('train_num_id', 'email', 'carriage_num', 'seat_num')



class Order(models.Model):
    order_num = models.CharField(max_length=80, primary_key=True)
    ticket_num = models.CharField(max_length=80, null=True, blank=True)
    phone = models.CharField(max_length=60)
    email = models.CharField(max_length=100)
    ORDER_STATUS_LIST = (
        ('Done', '完成'),
        ('Cancle', '取消'),
        ('Unpay', '待付款'),
        ('Changing', '可改签,可退款'),
        ('Payback', '仅可退款'),
    )
    order_status = models.CharField(max_length=8, choices=ORDER_STATUS_LIST)
    pub_date = models.DateTimeField('创建时间')
    changed_train_num_id = models.CharField(max_length=80, null=True)
    price = models.DecimalField(max_digits=10, decimal_places=2)

    # 多对一关系
    order_client = models.ForeignKey(Client, on_delete=models.CASCADE)



class TrainNumber(models.Model):
    train_num_id = models.CharField(max_length=80, primary_key=True)
```

```
    train_name = models.CharField(max_length=20)
    start_station = models.CharField(max_length=20)
    end_station = models.CharField(max_length=20)
    date = models.DateField()

    def get_carriages_num(self):
        return self.carriage_set.count()

    def get_seat_type_num(self):
        yz, rz, yw, rw, edz, ydz, swz = 0, 0, 0, 0, 0, 0, 0
        if self.get_carriages_num() == 0:
            return {'YZ':0, 'YW':0, 'RZ':0, 'RW':0, 'YDZ':0, 'EDZ':0, 'SWZ':0}
        else:
            for carriage in self.carriage_set.all():
                if carriage.seat_type == 'YZ':
                    yz += carriage.get_seats_num()
                elif carriage.seat_type == 'RZ':
                    rz += carriage.get_seats_num()
                elif carriage.seat_type == 'YW':
                    yw += carriage.get_seats_num()
                elif carriage.seat_type == 'RW':
                    rw += carriage.get_seats_num()
                elif carriage.seat_type == 'EDZ':
                    edz += carriage.get_seats_num()
                elif carriage.seat_type == 'YDZ':
                    ydz += carriage.get_seats_num()
                elif carriage.seat_type == 'SWZ':
                    swz += carriage.get_seats_num()
            return {'YZ':yz, 'RZ':rz, 'YW':yw, 'EDZ':edz, 'YDZ':ydz, 'SWZ':swz,
'RW':rw}

class Carriage(models.Model):
   # id = models.AutoField(primary_key=True)
    belong_to_train = models.ForeignKey(TrainNumber, on_delete=models.CASCADE)
    carriage_num = models.CharField(max_length=20)
    SEAT_TYEP_LIST = (
        ('YZ', '硬座'),
        ('RZ', '软座'),
        ('YW', '硬卧'),
        ('RW', '软卧'),
        ('YDZ', '一等座'),
        ('EDZ', '二等座'),
        ('SWZ', '商务座'),
    )
```

```
    seat_type = models.CharField(max_length=3, choices=SEAT_TYEP_LIST)

    def get_seats_num(self):
        return self.seat_set.count()

    class Meta:
        unique_together = ('belong_to_train', 'carriage_num')



class Seat(models.Model):
    belong_to_carriage = models.ForeignKey(Carriage, on_delete=models.CASCADE)
    seat_num = models.CharField(max_length=20, default=None)
    price = models.DecimalField(max_digits=10, decimal_places=2)

    def get_carriage(self):
        return self.belong_to_carriage.carriage_num

    def get_train(self):
        return self.belong_to_carriage.belong_to_train.train_num_id

    class Meta:
        unique_together = ('belong_to_carriage', 'seat_num')


class Stations(models.Model):
    #station_id = models.CharField(max_length=50, primary_key=True)
    station_name = models.CharField(max_length=50, primary_key=True)
    station_code = models.CharField(max_length=3, blank=True)


class PassStations(models.Model):
    train_num_id = models.CharField(max_length=80)
    station_id = models.CharField(max_length=80)
    arrival_time = models.DateTimeField('arrival time')
    depart_time = models.DateTimeField('depart time')
    station_num = models.IntegerField(default=-1)
    SEAT_TYEP_LIST = (
        ('YZ', '硬座'),
        ('RZ', '软座'),
        ('YW', '硬卧'),
        ('RW', '软卧'),
        ('YDZ', '一等座'),
        ('EDZ', '二等座'),
```

```
        ('SWZ', '商务座'),
    )
    seat_type = models.CharField(max_length=3, choices=SEAT_TYEP_LIST)
    remine_seats_num = models.IntegerField()
    date = models.DateField(blank=True, null=True)



    class Meta:
        unique_together = ('train_num_id', 'station_id', 'seat_type')


class Current(models.Model):
    email = models.CharField(max_length=100)
    client = models.ForeignKey(Client, on_delete=models.CASCADE)

    class Meta:
        unique_together = ('email', 'client')
```

4.2 Views

```
def make_query(start_point, end_point, date):
    """
    进行单次车辆信息的查询:
        return 可用车次信息的 dict
    """
    start_available                                                    =
list(PassStations.objects.filter(date=date).filter(station_id__contains=start_p
oint))
    end_available                                                      =
list(PassStations.objects.filter(date=date).filter(station_id__contains=end_poi
nt))

    # 对数据集进行预处理，出发地不可以是终点站，目的地不可以是起点站
    for item in start_available:
        if item.arrival_time == item.depart_time and item.station_num > 0:
            start_available.remove(item)
    for item in end_available:
        if item.arrival_time is None:
            end_available.remove(item)
        elif item.arrival_time == item.depart_time:
```

```
            end_available.remove(item)

    # 找出车次 id 做交集
    train_num_end = []
    train_num_start = []
    for end in end_available:
        train_num_end.append(end.train_num_id)
    for item in start_available:
        train_num_start.append(item.train_num_id)
    train_num_start = set(train_num_start)
    train_num_end = set(train_num_end)
    target_set = train_num_end & train_num_start

    target_set = list(target_set)
    results_dict = {}
    for proble in target_set:
        proble_pass_start                                                    =
PassStations.objects.filter(date=date).filter(train_num_id=proble).filter(stati
on_id__contains=start_point)
        proble_pass_end                                                      =
PassStations.objects.filter(date=date).filter(train_num_id=proble).filter(stati
on_id__contains=end_point)
        if proble_pass_end[0].arrival_time < proble_pass_start[0].depart_time:
            target_set.remove(proble)
    for item in target_set:
        print(item)#.train_num_id,      item.station_id,      item.arrival_time,
item.depart_time, item.seat_type, item.remine_seats_num)
        train_name = item[12:-8]  #  根据编码规则，得到车次名称
        target_train_item_start                                              =
PassStations.objects.filter(date=date).filter(train_num_id=item).filter(station
_id__contains=start_point)
        target_train_item_end                                                =
PassStations.objects.filter(date=date).filter(train_num_id=item).filter(station
_id__contains=end_point)[0]
        target_start_station = target_train_item_start[0].station_id
        target_end_station = target_train_item_end.station_id
        target_arrival_time = target_train_item_end.arrival_time
        target_depart_time = target_train_item_start[0].depart_time

        # result_train = TrainNumber.objects.get(train_num_id=item)

        yz = target_train_item_start.filter(seat_type='YZ')[0].remine_seats_num
        yw = target_train_item_start.filter(seat_type='YW')[0].remine_seats_num
        rz = target_train_item_start.filter(seat_type='YZ')[0].remine_seats_num
```

```
        rw = target_train_item_start.filter(seat_type='RW')[0].remine_seats_num
        ydz                                                                    =
target_train_item_start.filter(seat_type='YDZ')[0].remine_seats_num
        edz                                                                    =
target_train_item_start.filter(seat_type='EDZ')[0].remine_seats_num
        swz                                                                    =
target_train_item_start.filter(seat_type='SWZ')[0].remine_seats_num
        result = Result(
            train_name,
            target_start_station,
            target_end_station,
            target_depart_time,
            target_arrival_time,
            swz,
            ydz,
            edz,
            rz,
            yz,
            rw,
            yw)
        results_dict[item] = result

    return results_dict



def query_train(request):
    """
    进行余票查询：
    kwargs:
        start_point
        end_point
        date
        back_date
        is_single
    """
    start_point = request.GET.get('start_point')
    end_point = request.GET.get('end_point')
    date = request.GET.get('date')
    back_date =request.GET.get('back_date')
    is_single = request.GET.get('is_single')
    template_name = 'train_ticket/query.html'

    single_set = make_query(start_point, end_point, date)
```

```
    if is_single == 'single':
        return render(request, template_name, {'query_set_single':single_set,
'query_set_double':None})
    else:
        back_set = make_query(end_point, start_point, back_date)
        return render(request, template_name, {'query_set_single':single_set,
'query_set_double':back_set})




def book_ticket(request, train, seat, board_station, board_time, detrain_time,
detrain_station):
    """
    进行车票预订：
    args:
        email
        train_num_id
        seat_type
        board_station
        board_station
        detrain_station
        detrain_time
    """
    template_name = 'train_ticket/buy.html'
    print(train, seat, board_time, board_station)
    client = Current.objects.all()
    if not len(client):
        return render(request, template_name, {'error_msg':'您尚未登录，请登录后
购票！'})
    else:
        email = client[0].email

    carriage_sets       =       Carriage.objects.filter(belong_to_train=train,
seat_type=seat)
    def change_ticket(request, order_num, to_train, to_carriage, to_seat):
        """
         进行车票改签,生成相应新的车票，删除旧票，改变订单状态
        """
        template_name = 'train_ticket/change.html'
        order = Order.objects.get(order_num=order_num)
        ticket_num = order.ticket_num
        old_ticket = Ticket.objects.get(ticket_num=ticket_num)
    print('参与筛选的车厢数量：', len(carriage_sets))
```

```
    # 获取车票信息，检查坐席是否在其中
    tickets = Ticket.objects.filter(
                train_num_id=train,
                email=email,
                board_time=board_time,
                board_station=board_station,
                detrain_station=detrain_station,
                detrain_time=detrain_time,
                seat_type=seat
            )

    not_valid_seat = []
    if len(tickets):
        for ticket in tickets:
            not_valid_seat.append((ticket.carriage_num, ticket.seat_num))
    for carriage in carriage_sets:
        print('carriage num', carriage.carriage_num)
        if carriage.get_seats_num() == 0:
            continue
        carriage_num = carriage.carriage_num
        seat_sets = carriage.seat_set.all()
        for seat_ in seat_sets:
            seat_num = seat_.seat_num
            print('seat number:', seat_num)
            price = seat_.price
            if not_valid_seat == [] or (carriage_num, seat_num) in not_valid_seat:
                # 该 seat 可用,生成相应订单
                client = Client.objects.get(email=email)
                phone = client.phone
                now = timezone.localtime()
                # 修改车辆余座信息
                try:
                    pass_train  =  PassStations.objects.get(train_num_id=train,
station_id=board_station, seat_type=seat)
                    pass_train.remine_seats_num -= 1
                    pass_train.save()
                    print('更改车次信息成功!')
                except PassStations.DoesNotExist:
                    print('更改车次余票信息出错')

                #生成订单以及车票
                order_num           =           'E'+           str(now.year)+
str(now.month)+str(now.day)+str(now.hour)+str(now.minute)+str(now.second)+str(e
mail)
```

```
                create_order = Order(
                    order_num=order_num,
                    ticket_num=None,
                    email=email,
                    phone=phone,
                    order_status='Unpay',
                    pub_date=timezone.localtime(),
                    changed_train_num_id=None,
                    price=price,
                    order_client=client
                )
                create_ticket = Ticket(
                    ticket_num=order_num,
                    train_num_id=train,
                    email=email,
                    carriage_num=carriage_num,
                    seat_num=seat_num,
                    board_time=board_time,
                    board_station=board_station,
                    detrain_station=detrain_station,
                    detrain_time=detrain_time,
                    price=price,
                    seat_type=seat,
                    ticket_client=client
                )
                create_ticket.save()
                create_order.save()
                print(create_ticket.ticket_num,
create_ticket.train_num_id[12:-8],                          create_ticket.email,
create_ticket.carriage_num, create_ticket.board_time, create_ticket.price)
                return         render(request,         'train_ticket/buy.html',
{'order_num':create_order.order_num,     'ticket_num':create_ticket.ticket_num,
'price':str(price)})



def buy(request, order_num, ticket_num):
    template_name = 'train_ticket/buy.html'
    return render(request, template_name, {'success_msg':'订单生成成功,请支付!',
'order_num':order_num, 'ticket':ticket_num})


def pay(request, order_num, ticket_num, payed):
    """
```

```
    用户支付：
    args：
        order
    """
    template_name = 'train_ticket/pay.html'
    order = Order.objects.get(order_num=order_num)
    ticket = Ticket.objects.get(ticket_num=ticket_num)
    email = order.email
    client = order.order_client
    train_num_id = ticket.train_num_id
    station = ticket.board_station
    seat_type = ticket.seat_type

    if payed == 1 and order.order_status == 'Unpay':
        order.order_status = 'Changing'
        order.ticket_num = ticket.ticket_num
        msg = '购票成功，可进行改签退票操作'
    elif payed == 0 and order.order_status == 'Unpay':
        order.order_status = 'Cancle'
        try:
            pass_train  =  PassStations.objects.get(train_num_id=train_num_id,
station_id=station, seat_type=seat_type)
            pass_train.remine_seats_num += 1
            pass_train.save()
        except PassStations.DoesNotExist:
            print('取消订单更新失败!')
        ticket.delete()
        msg = '订单已取消'

    order.save()
    return render(request, template_name, {"msg":msg})



class TicketView(View):
    def get(self, request, *args, **kwargs):
        current = Current.objects.all()
        if not len(current):
            return render(request, 'train_ticket/ticket_result.html', {'msg':'
请登录后查看订单信息!'})
        else:
            cur = current[0]
            email = cur.email
            client = Client.objects.get(email=email)
```

```
            name = client.name
            orders = Order.objects.filter(email=email)
            return      render(request,      'train_ticket/ticket_result.html',
{'order_set':orders,'name':name, 'msg':'订单信息'})



def change_ticket(request, order_num):
    """
     进行车票改签,生成相应新的车票，删除旧票，改变订单状态
    """
    ticket = Ticket.objects.get(ticket_num=order_num)
    order = Order.objects.get(order_num=order_num)
    if order.order_status == 'Changing':
        ticket.delete()
        order.order_status = 'Done'
    order.save()
    return render(request, 'train_ticket/index.html', {'test':None})

def pay_back(request, order_num):
    order = Order.objects.get(order_num=order_num)
    ticket = Ticket.objects.get(ticket_num=order_num)
    if order.order_status == 'Payback' or order.order_status == 'Changing':
        ticket.delete()
        order.order_status = 'Cancle'
        order.save()
        return render(request, 'train_ticket/ticket_result.html', {'msg':'退票成
功!'})

def delete_order(request, order_num):
    order = Order.objects.get(order_num=order_num)
    if order.order_status == 'Cancle':
        order.delete()
        return render(request, 'train_ticket/ticket_result.html', {'msg':'订单已
删除!'})



class IndexView(View):
    template_name = 'train_ticket/index.html'

    def get(self, request, *args, **kwargs):
        return render(request, self.template_name, {'test':None})
```

```
class LoginView(View):
    template_name = 'train_ticket/login.html'

    def get(self, request, *args, **kwargs):
        return render(request, self.template_name, {'test':None})

    def post(self, request, *args, **kwargs):
        current = Current.objects.all()
        if len(current) > 0:
            current[0].delete()
        username = request.POST.get('username')
        pwd = request.POST.get('passwd')
        pattern = re.compile(r'^[0-9]+')
        try:
            if pattern.match(username):
                client = Client.objects.get(phone=username)
            else:
                client = Client.objects.get(email=username)
        except Client.DoesNotExist:
            return render(request, self.template_name, {'error_msg':'请先注册'})
        if check_password(pwd, client.passwd):
            cur = Current(email=client.email, client=client)
            cur.save()
            print('email:{} login sucessfully!'.format(username))
            return redirect(reverse('train_ticket:index'))
        else:
            return render(request, self.template_name, {'error_msg':'密码错误'})



class RegisterView(View):
    template_name = 'train_ticket/register.html'

    def get(self, request, *args, **kwargs):
        return render(request, self.template_name, {'test':None})

    def post(self, request, *args, **kwargs):
        if request.method == 'POST':
            print(request.POST)
            print(request.POST.get('email'))
            print(request.POST.get('name'))
            print(request.POST.get('passwd'))
```

```
            name = request.POST.get('name')
            email= request.POST.get('email')
            phone = request.POST.get('phone')
            card_num = request.POST.get('card_num')
            card_type = request.POST.get('card_type')
            sex = request.POST.get('sex')
            pwd = request.POST.get('passwd')
            pwd = make_password(pwd)

            if not name:
                return(request, self.template_name, {'error_msg':'姓名不可为空!'})
            if not email:
                return(request, self.template_name, {'error_msg':'邮箱不可为空!'})
            if not phone:
                return(request, self.template_name, {'error_msg':'手机号不可为空!'})
            if not sex:
                return(request, self.template_name, {'error_msg':'性别不可为空!'})
            if not card_type:
                return(request,self.template_name,{'error_msg':'证件类型不可为空!'})
            if not name:
                return(request,self.template_name,{'error_msg':'证件号码不可为空!'})



            try:
                if Client.objects.get(email=email):
                    return render(request, self.template_name, {'error_msg':'用户已注册！'},)
            except Client.DoesNotExist:
                client = Client(email=email, phone=phone, name=name, sex=sex, card_type=card_type, card_num=card_num, passwd=pwd)
                client.save()
                return redirect(reverse('train_ticket:login'), {'success_msg':'注册成功！'})
```

4.3 Templates

base.html

作为所有模板的基础模板，其他模板都在此基础上进行扩展，该模板包含了顶部导航栏，以及基本的控件，并且给内容模块，css 模块，script 模块留下了预留的 block 空间.方便后续模板进行扩展

```
{% load staticfiles %}
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!--上述内容必须放到最前面-->
    <title>{% block title %}Base template{% endblock %}</title>

    <!--Bootstrap-->
    <link rel="stylesheet" href="{% static
'train_ticket/bootstrap/css/bootstrap.min.css' %}" media="screen" />
    {% block css %}{% endblock %}
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <a class="navbar-brand" href="#">Innohub 票务中心</a>
        <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent"
aria-expanded="false" aria-label="Toggle navigation">
            <span class="navbar-toggler-icon"></span>
         </button>

        <div class="collapse navbar-collapse" id="navbarSupportedContent">
            <ul class="navbar-nav mr-auto">
                <li class="nav-item active">
                    <a class="nav-link" href="{% url 'train_ticket:index' %}">
主页 <span class="sr-only">(current)</span></a>
                </li>
            <li class="nav-item">
                <a class="nav-link" href="#">资讯</a>
            </li>
            <li class="nav-item dropdown">
                <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown"
role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
                    订单中心
```

```
                </a>
                <div class="dropdown-menu" aria-labelledby="navbarDropdown">
                    <a         class="dropdown-item"         href="{%         url
'train_ticket:ticket' %}">待支付订单</a>
                    <a         class="dropdown-item"         href="{%         url
'train_ticket:ticket' %}">已完成订单</a>
                </div>
            </li>
            <li class="nav-item">
                <a class="nav-link disabled" href="#">Disabled</a>
            </li>
            </ul>
            <!--
            <form class="form-inline my-2 my-lg-0">
                <input      class="form-control      mr-sm-2"      type="search"
placeholder="Search" aria-label="Search">
                <button    class="btn    btn-outline-success    my-2    my-sm-0"
type="submit">Search</button>
            </form>
            -->
            <ul class="nav navbar-bar navbar-right">
                <li class="nav-item"><a href="{% url 'train_ticket:login'  %}">
登录</a></li>  
                <li class="nav-item"><a href="{% url 'train_ticket:register' %}">
注册</a></li>
            </ul>
        </div>
   </nav>
    {% block content %}{% endblock %}
    <script src="{% static 'train_ticket/popper.js' %}"></script>
    <script src="{% static 'train_ticket/jquery-3.3.1.js' %}"></script>
    <script                          src="{%                          static
'train_ticket/bootstrap/js/bootstrap.min.js' %}"></script>
    {% block script %}{% endblock %}
</body>
</html>
```

query.html

查询结果的模板，获取相应的查询集，然后在表格中进行显示

```
{% extends 'train_ticket/base.html' %}
{% load static %}
{% block title %}index{% endblock %}

{% block content%}
    <div class="row-fluid">
```

```
        <div class="container">
            <legend>单程票</legend>
            <table class="table">
                <thead>
                    <th scope="col">车次</th>
                    <th scope="col">出发站</th>
                    <th scope="col">到达站</th>
                    <th scope="col">出发/到达时间</th>
                    <th scope="col">商务座</th>
                    <th scope="col">一等座</th>
                    <th scope="col">二等座</th>
                    <th scope="col">软卧</th>
                    <th scope="col">硬卧</th>
                    <th scope="col">软座</th>
                    <th scope="col">硬座</th>
                </thead>
                {% for key, value in query_set_single.items %}
                <tbody>
                    <tr>
                        <th scope="row">{{value.train_name}}</th>
                        <td>{{value.start_station}}</td>
                        <td>{{value.end_station}}</td>
                        <td>{{value.start_time}}/{{value.end_time}}</td>
                        <td>{{value.swz}}</td>
                        <td>{{value.ydz}}</td>
                        <td>{{value.edz}}</td>
                        <td>{{value.rw}}</td>
                        <td>{{value.yw}}</td>
                        <td>{{value.rz}}</td>
                        <td>{{value.yz}}</td>
                        <td>
                            <div class="nav-item dropdown">
                            <button class="btn btn-primary dropdown-toggle"
type="button" id="dropdownMenuButton" data-toggle="dropdown"
aria-haspopup="true" aria-expanded="false">
                                预定
                            </button>
                            <div class="dropdown-menu"
aria-labelledby="dropdownMenuButton">
                                {% if value.swz > 0 %}
                                <a class="dropdown-item" href="{% url
'train_ticket:book_ticket' key 'SWZ' value.start_station value.start_time
value.end_station value.end_time %}">商务座</a>
                                {% endif %}
```

```
                                    {% if value.ydz > 0 %}
                                    <a   class="dropdown-item"   href="{%   url
'train_ticket:book_ticket'   key   'YDZ'   value.start_station   value.start_time
value.end_station value.end_time %}">一等座</a>
                                    {% endif %}
                                    {% if value.edz > 0 %}
                                    <a   class="dropdown-item"   href="{%   url
'train_ticket:book_ticket'   key   'EDZ'   value.start_station   value.start_time
value.end_station value.end_time %}">二等座</a>
                                    {% endif %}
                                    {% if value.rw > 0 %}
                                    <a  class="dropdown-item"  href="{%    url
'train_ticket:book_ticket'   key   'RW'   value.start_station   value.start_time
value.end_station value.end_time%}">软卧</a>
                                    {% endif %}
                                    {% if value.yw > 0 %}
                                        <a  class="dropdown-item"  href="{%  url
'train_ticket:book_ticket'   key   'YW'   value.start_station   value.start_time
value.end_station value.end_time %}">硬卧</a>
                                    {% endif %}
                                    {% if value.rz > 0 %}
                                    <a   class="dropdown-item"   href="{%   url
'train_ticket:book_ticket'   key   'RZ'   value.start_station   value.start_time
value.end_station value.end_time%}">软座</a>
                                    {% endif %}
                                    {% if value.yz > 0 %}
                                        <a  class="dropdown-item"  href="{%  url
'train_ticket:book_ticket'   key   'YZ'   value.start_station   value.start_time
value.end_station value.end_time%}">硬座</a>
                                    {% endif %}
                                </div>
                            </div>
                        </td>
                    </tr>
                    {% endfor %}
                </tbody>
            </table>
        </div>
    </div>
    <div class="row-fluid">
        {% if query_set_double %}
        <div class="container">
            <legend>返程票</legend>
            <table class="table">
```

```
                <thead>
                    <th scope="col">车次</th>
                    <th scope="col">出发站</th>
                    <th scope="col">到达站</th>
                    <th scope="col">出发/到达时间</th>
                    <th scope="col">商务座</th>
                    <th scope="col">一等座</th>
                    <th scope="col">二等座</th>
                    <th scope="col">软卧</th>
                    <th scope="col">硬卧</th>
                    <th scope="col">软座</th>
                    <th scope="col">硬座</th>
                </thead>
                {% for key, value in query_set_double.items %}
                <tbody>
                    <tr>
                        <th scope="row">{{value.train_name}}</th>
                        <td>{{value.start_station}}</td>
                        <td>{{value.end_station}}</td>
                        <td>{{value.start_time}}/{{value.end_time}}</td>
                        <td>{{value.swz}}</td>
                        <td>{{value.ydz}}</td>
                        <td>{{value.edz}}</td>
                        <td>{{value.rw}}</td>
                        <td>{{value.yw}}</td>
                        <td>{{value.rz}}</td>
                        <td>{{value.yz}}</td>
                        <td>
                            <div class="nav-item dropdown">
                                <button class="btn btn-primary dropdown-toggle"
type="button"           id="dropdownMenuButton"          data-toggle="dropdown"
aria-haspopup="true" aria-expanded="false">
                                    预定
                                </button>
                                <div                        class="dropdown-menu"
aria-labelledby="dropdownMenuButton">
                                    {% if value.swz > 0 %}
                                    <a   class="dropdown-item"   href="{%   url
'train_ticket:book_ticket'   key   'SWZ'   value.start_station   value.start_time
value.end_station value.end_time%}">商务座</a>
                                    {% endif %}
                                    {% if value.ydz > 0 %}
                                    <a   class="dropdown-item"   href="{%   url
'train_ticket:book_ticket'   key   'YDZ'   value.start_station   value.start_time
```

```
value.end_station value.end_time%}">一等座</a>
                                    {% endif %}
                                    {% if value.edz > 0 %}
                                        <a  class="dropdown-item"  href="{%  url
'train_ticket:book_ticket'   key   'EDZ'   value.start_station   value.start_time
value.end_station value.end_time%}">二等座</a>
                                    {% endif %}
                                    {% if value.rw > 0 %}
                                        <a  class="dropdown-item"  href="{%  url
'train_ticket:book_ticket'   key   'RW'   value.start_station   value.start_time
value.end_station value.end_time%}">软卧</a>
                                    {% endif %}
                                    {% if value.yw > 0 %}
                                        <a  class="dropdown-item"  href="{%  url
'train_ticket:book_ticket'   key   'YW'   value.start_station   value.start_time
value.end_station value.end_time%}">硬卧</a>
                                    {% endif %}
                                    {% if value.rz > 0 %}
                                        <a  class="dropdown-item"  href="{%  url
'train_ticket:book_ticket'   key   'RZ'   value.start_station   value.start_time
value.end_station value.end_time%}">软座</a>
                                    {% endif %}
                                    {% if value.yz > 0 %}
                                        <a  class="dropdown-item"  href="{%  url
'train_ticket:book_ticket'   key   'YZ'   value.start_station   value.start_time
value.end_station value.end_time%}">硬座</a>
                                    {% endif %}
                                </div>
                            </div>
                        </td>
                    </tr>
                </tbody>
                {% endfor %}
            </table>
        </div>
        {% endif %}
    </div>
{% endblock %}
{% block script%}
<script>

    $(".btn").click(function(){
        console.log('test');
        $(this).button('loading').delay(1000).queue(function() {
```

```
        // $(this).button('reset');
        // $(this).dequeue();
        });
    });

</script>
{% endblock%}
```

ticket_result.html

```
{% extends 'train_ticket/base.html' %}
{% load static %}
{%block title%}登录{%endblock%}
{% block content %}
<div class="container">
    <div class="col-md-7">
        <legend>{{msg}}</legend>
        {% if order_set %}
        <table class="table">
            <thead>
                <th scope="col">订单号</th>
                <th scoe="col">车次</th>
                <th scope="col">姓名</th>
                <th scope="col">email</th>
                <th scope="col">订单状态</th>
                <th scope="col">时间</th>
                <th scope="col">金额</th>
            </thead>
            {% for order in order_set %}
            <tbody>
                <th scope="row">{{order.order_num}}</th>
                <td>{{order.train_num_id}}</td>
                <td>{{name}}</td>
                <td>{{order.email}}</td>
                <td>{{order.order_status}}</td>
                <td>{{order.pub_date}}</td>
                <td>{{order.price}}</td>
                <td>
                    <div class="nav-item dropdown">
                        <button    class="btn    btn-primary    dropdown-toggle"
type="button"           id="dropdownMenuButton"           data-toggle="dropdown"
aria-haspopup="true" aria-expanded="false">
                                订单操作
                            </button>
                            <div                      class="dropdown-menu"
aria-labelledby="dropdownMenuButton">
```

```
                                {% if order.order_status == 'Changing' %}
                                <a    class="dropdown-item"    href="{%    url
'train_ticket:payback' order.order_num %}">退票</a>
                                <a    class="dropdown-item"    href="{%    url
'train_ticket:change' order.order_num %}">改签</a>
                                 {% elif order.order_status == 'Unpay' %}
                                 <a class="dropdown-item" href="#">支付</a>
                                 {% elif order.order_status == 'Payback' %}
                                 <a    class="dropdown-item"    href="{%    url
'train_ticket:pay_back' order.order_num %}">退票</a>
                                 {% else %}
                                 <a    class="dropdown-item"    href="{%    url
'train_ticket:delete' order.order_num %}">删除</a>
                                 {% endif %}
                        </div>
                    </div>
                </td>
            </tbody>
            {% endfor %}
        </table>
        {% endif %}
    </div>
</div>
{% endblock %}
{% block script %}
<script>
alert(msg);
</script>
{% endblock%}
```

八、 测试和运行

1 黑盒测试

等价类划分

分为有效等价类以及无效等价类,使用有效等价类用于检查是否可以完成需求所要求的功能;使用无效等价类测试用户无效操作的交互提示是否合理

测试页面	有效等价类	无效等价类
注册	姓名:汉字或者英文,必须是在户籍系统中可以	存在不符合有效等价类中的格式的输入

	查到的中国国籍或登录在册的外国籍姓名 性别:二选一,不可为空 邮箱地址:必须是符合正则*r(^[a-zA-Z0-9_-]+@[a-zA-Z0-9_-]+(\.[a-zA-Z0-9_-]+)+$)* 手机号必须是11位中国号码 证件类型必须是24种可用购票证件之一,不可为空 证件号码必须为18位最后一位可以为x 密码至少为6位以上	存在空选项 注册已经注册过的手机号或者邮箱
登录	已经注册过的用户的邮箱或者手机号作为登录名,以及对应的密码	用户名或者密码为空 尚未注册过得手机号或者邮箱,自动重定向到注册页面
余票查询	勾选单程车票时返程日期的时间选择框为不可操作状态;勾选往返车票时,返程日期一栏早于出发日期的时间不可选择 出发地.目的地必须是实际存在站点的地名	没有勾选单程或者返程 出发地或者目的地为空或者存在非法字符 没有选择日期
订购	页面只可以选择预定有余票的坐席 预定扣票成功后,可以取消订单或者选择支付	略
订单列表	对于可进行操作的订单进行操作后,改变相应的状态 只有进行删除操作,订单才会从数据库中删除	略
后台操作	后台管理员登录与用户登录一致 点击相应的表项可以更改对应数据库中信息,每个表中的所有数据都以列表形式展示,点击单项可以进行详情查看	普通管理员尝试更改全限之外的内容 更改内容为非法格式,或者数据已经存在

边界分析

利用边界数据对于程序运行结果进行分析,通过对边界情况的确认,修改可能出现的错误和不兼容

测试项目	边界分析	测试用例
经停站信息修改	到站时间不可以早于timezone.localtime()；离站时间不可以早于到站时间	到:2018-10-01 22:03:00 离:2018-10-01 22:05:00 到:2018-09-11 01:01:00 离:2018-09-10 01:01:00 到:2019-09-11 11:11:11 离:2018-09-11 09:00:00
余票查询	出发地与目的地一致;出发日期返程日一致	出发:北京 到达:北京 出发:2018-10-01 返程:2018-10-01
车次信息	在用数据日期不可以早于timezone.localtime()	DATE: 2017-10-10

九、 总结

本次数据库课程设计通过模拟 12306 购票系统，通过使用 web 框架 django，结合 bootstrap 进行购票系统的开发，让我对于以数据库为基础的现实应用有了初步认知。虽然该系统是我们生活必不可少的部分，具有较为明晰的业务功能需求，但是对于内部业务逻辑的还需要进一步的研究，而且需要组织巨大数量的数据，所以还是很有难度的。在系统设计的初级阶段，通过预期产品定位，需求分析，产品需求文档的书写，逐步加深对于该项目的认识与理解；在数据库设计阶段，由功能出发，不断增加所需的实体联系集，然后发现该系统的设计很多部分是存在问题的。比如对每个座位的设计，由于需要记录每个车厢的每个座位的每个时段的乘车人以及座位状态，但是对于每一个座位建立一个表又是不太现实的，所以采用了“时间换取空间”的策略----在经行站点表中存储每个车次每个坐席类型的余座数目，进行扣票时，在相应的坐席类型下查找车票表，如果在该区间不存在该座位的车票，则该座位可以进行售票。例如此类的问题还遇到了很多，在一步步解决的过程中也逐步提升了对于系统设计的理解与认识。在 ui 设计阶段，增加可能的交互动作，提高系统的用户友好性。虽然可能该设计与现实中的购票系统存在很大差距，毕竟处理的业务以及数据量和自动化程度还是有很大区别的，但是在整个过程中还是有很大收获的。

由于真实购票系统十分庞大复杂，需要处理特别大的数据量。原本计划进行较为真实的模拟，所以在进行数据收集以及整理上花费了大部分的精力，然而自身的系统又不足以支持如此大量的数据，所以不得已删除了部分数据，在整体的方向上有一定的失误。所以在功能实现上并没有达到预期的效果，这是本次课程设计的不足之处。虽然数据在程序中起到了至关重要的作用，但是在课设的情形下，由于时间的压力，过于关注数据导致了功能的不足。

这也是一次不成功的“博弈”，也是以后我需要格外注意的地方。开发不是一蹴而就的事情，需要不断的迭代以及更新，需要着眼于核心功能，尤其是在时间紧迫的情形下。另一方面，由于是使用新的工具开发，在具体实现时也花费了大量的时间阅读文档，查看 API，造成了开发进度缓慢。但我不认为这是一个不好的方面，毕竟在学生时代就是要尝试新的东西，找到适合使用的工具，而且在这种情形下也有助于快速学习了解新的框架，技术，也是十分快乐的体验。

我自身还有许多需要提高的地方，不论是具体技术还是系统设计都是十分浅薄幼稚的，需要不断进行学习，需要积累开发经验。之前阅读过关于产品设计方面的书籍，在自己真正动手操作时才发现里面别有洞天，通过数据库课设我也提高了自己发现问题，解决问题的能力，可谓受益匪浅。

附：参考文档

Django: https://docs.djangoproject.com/en/2.1/

Bootstrap: http://getbootstrap.com/docs/4.1/getting-started/introduction/

Uwsgi: http://getbootstrap.com/docs/4.1/getting-started/introduction/

Nginx: https://docs.nginx.com/

Firewalls:https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/security_guide/sec-using_firewalls#sec-Introduction_to_firewalldl