项目负责人：

项目开发人员：

项目持续周期：

版本：v1.0; 时间：17-08-03

版本：v1.1; 时间：17-08-08

版本：v1.2; 时间：17-08-15

TVOS项目概要设计说明书

1 项目简介

1.1目的

分析http://120.25.200.39:8081上的数据，并将信息可视化展现在搭建的网址上

超级用户，可以直接修改数据库信息

1.2 实现

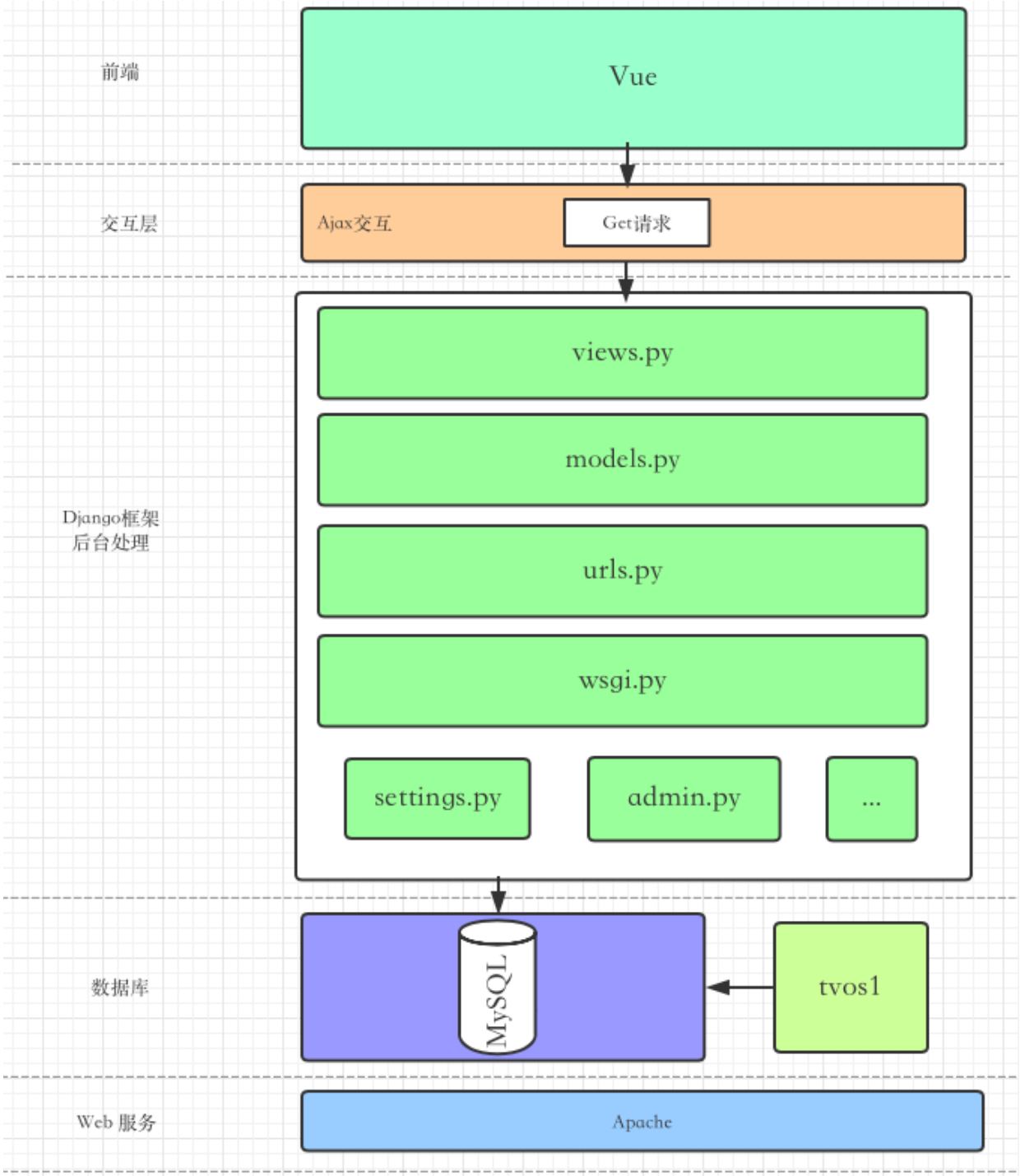
Django+MySQL+ECharts+Vue

运行环境：

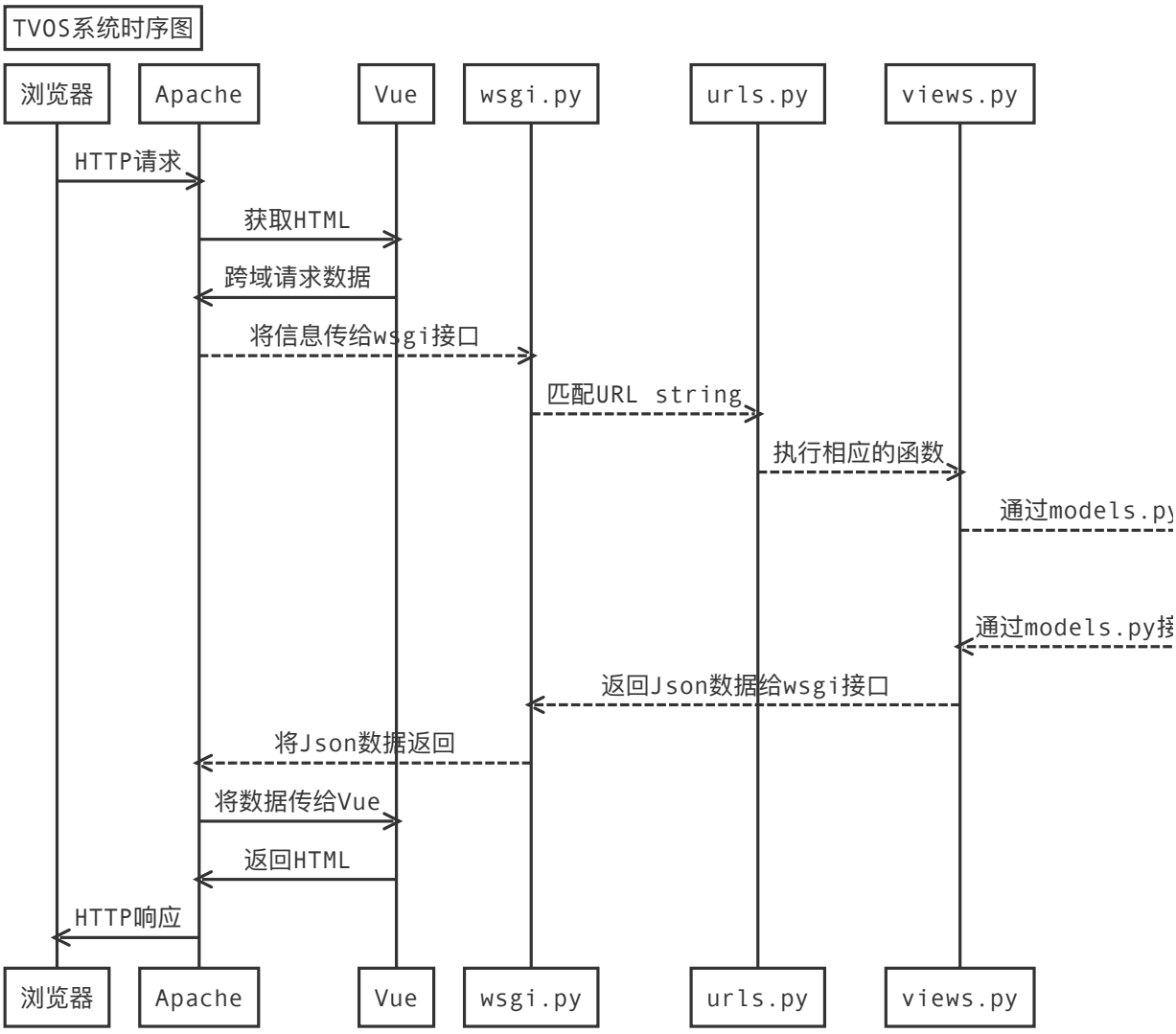
1. Django1.11
2. MySQL5.6
3. Python3.4

1.3 结构图

1.3.1 系统框架图



1.3.2 具体request-response图



2 概要设计

2.1数据获取

2.1.1各版块记录获取

首先访问网址，使用F12，动态抓包（Network->Headers），获取到如下信息

对于open，Merged，Abandoned，显示界面的json文件如下：

- Open http://120.25.200.39:8081/changes/?n=25&O=81
- Merged http://120.25.200.39:8081/changes/?q=status:merged&n=25&O=81
- Abandoned http://120.25.200.39:8081/changes/?q=status:abandoned&n=25&O=81

2.1.2 记录中详细信息的获取

点击网页上面版块任一条目，进入详细信息，其json文件网址为

http://120.25.200.39:8081/changes/1637/revisions/205f3f9694e931de9779cdbaa82c5bb881751899/files

分析上面网址，有两个地方是变化的：

- 一个是1637，一个是205f3f9694e931de9779cdbaa82c5bb881751899

1637代表的是该条目的编号，在对应版块的json文件中可获取(_number)

- 第二个为commit的编号，通过查看Network中显示的文件，发现在下面的json文件中可获取

http://120.25.200.39:8081/changes/1637/detail?O=404

2.1.3 代码实现

见**附录5.1数据获取**

总共下载了1500多个json文件。

2.2 数据分析

2.2.1 all,open,merged,abandoned版块显示部分

1. 项目-修改条数（饼图）：需要的数据是项目名，以及各项目的修改次数
2. 时间-修改量（折线图）：需要的数据是时间，及其对应的代码增加量，代码删除量
3. 用户-修改条数（条形图）：需要的数据是用户名，及其修改的次数
4. 公司-修改条数（条形图）：需要的数据是公司名，及其修改的次数

2.2.2 侧页显示部分

显示项目名，用户名，公司名

2.2.3 侧页点击事件显示部分

项目条目

1. branch-修改条数（饼图）
2. 时间-修改量（折线图）
3. 用户-修改条数（条形图）
4. 公司-修改条数（条形图）

用户条目

1. 时间-修改条数（折线图）
2. 项目-修改条数（条形图）
3. 公司-修改条（条形图？）

公司条目

1. 时间-修改量（折线图）
2. 项目-修改条数（条形图）
3. 用户-修改条数（条形图）

2.3 数据存储

根据上面的数据分析设计数据表结构

2.3.1 数据库表格设计

tvos_record

字段名称	字段描述	字段类型	约束
number	标号	int	PRIMARY KEY
project	项目名	varchar(50)	NOT NULL
branch	分支	varchar(20)	NOT NULL
updated	更新日期	date	NOT NULL
insertions	增量	int	NOT NULL
deletions	删量	int	NOT NULL
owner	修改者	varchar(16)	NOT NULL
section	版块	varchar(10)	NOT NULL
company	公司	varchar(16)	NOTNULL

tvos_info

字段名称	字段描述	字段类型	备注
filepath	修改的文件路径	varchar(250)	PRIMARY KEY
project	所属项目名	varchar(50)	PRIMARY KEY
insertions	总增量	int	NOT NULL
deletions	总删量	int	NOT NULL
num	修改次数	int	NOT NULL

【这里Django中没有联合主键，会默认新建字段id，约束条件是project 和filepath字段唯一】

2.3.2 存储数据

由上在models.py中设置数据表，如下所示：

```
class Record(models.Model):
    number=models.IntegerField(primary_key=True)
    project=models.CharField(max_length=50)
    branch=models.CharField(max_length=20)
    updated=models.DateField()
    insertions=models.IntegerField()
    deletions=models.IntegerField()
    owner=models.CharField(max_length=16)
    company=models.CharField(max_length=16)
    section = models.CharField(max_length=10)
    def __str__(self):
        return self.number


class Info(models.Model):
    filepath=models.CharField(max_length=250)
    project=models.CharField(max_length=50)
    insertions=models.IntegerField()
    deletions=models.IntegerField()
    num=models.IntegerField()

    class Meta:
        unique_together = ('filepath', 'project')
```

在settings中导入pymysql

```
import pymysql
pymysql.install_as_MySQLdb()
```

修改数据库信息（此处需要安装好MySQL环境）：

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'tvosdb',
        'USER': 'root',
        'PASSWORD':'123645',
        'HOST':'localhost',
        'PORT': '',
    }
}
```

终端进入manage.py目录执行下面命令

```
python3 manage.py makemigrations
python3 manage.py migrate
```

会发现Mysql的tvosdb数据库中生成了好几个表，tovs目录下多了一个migrations文件夹

插入数据，这里直接使用pymysql插入数据到MySQL中，具体代码见附录，由**tvos1**项目实现

2.4 Django后台

Django框架中一个项目可以有多个app，在app中完成前后端的交互，分为三个部分类似于MVC模式的MTV模式

2.4.1 Model

负责业务对象和数据库的关系映射，作为views.py与数据库的接口

models.py在2.3.2部分已近给出

2.4.2 Template

负责如何把页面展示给用户

这里对应的时templates目录下的html文件，前端html文件全放于此，对于前端中使用到的css,js等文件，需要再app（或者其他目录）下新建static目录，后台会去查找static目录，见**3.1.1**

此项目并没有使用Django中的T部分，而是前后端分离

详细见**前端部分**

2.4.3 View

负责业务逻辑，并在适当时候调用Model和Template

这里对应的是views.py文件，在其中定义函数，获取数据，并传入前端界面

函数名和前端访问网址对应，在项目中的urls.py中使用正则匹配获取网址，并对应到view.py中的函数

此处可见附录部分示例

2.5 服务器部署

2.5.1 直接部署

安装软件

1. 在服务器上安装apache2和mod_wsgi
2. 安装Django1.11
3. 安装python3.4及以上
4. 安装Mysql

将TVOS项目放在服务器指定位置

设置网址配置文件

激活网址

启动服务器

2.5.2 Docker

【待】

2.6 前端设计

2.6.1 ECharts

按照ECharts官网，下载echarts库，并在html文件的head下导入

```
<script src="/static/echarts.min.js"></script>
```

接下来只需要实现如下脚本在`<script>`中

对需要显示的div初始化如下：

```
var ownerChart=echarts.init(document.getElementById('owner'))
```

设置option,在其中接收传入的数据，以json传入，这里需要使用safe，如下：

```
data:{{ name|safe }}
```

最后设置生效：

```
ownerChart.setOption(option);
```

显示部分见**2.2**

2.6.2 Bootstrap

使用Bootstrap框架来实现前端界面，这里使用官网中的样例

将Bootstrap和样例中的css，js文件放置于static目录下

整个界面分为三部分，由左侧的项目框和各版块的导航条，以及中间的显示部分组成，网页截取一部分如下所示：

这里左侧项目条和上侧导航条部分，在点击过程中，网页不会跳转，只会让中间部分根据不同的数据重新绘制，具体实现如下：

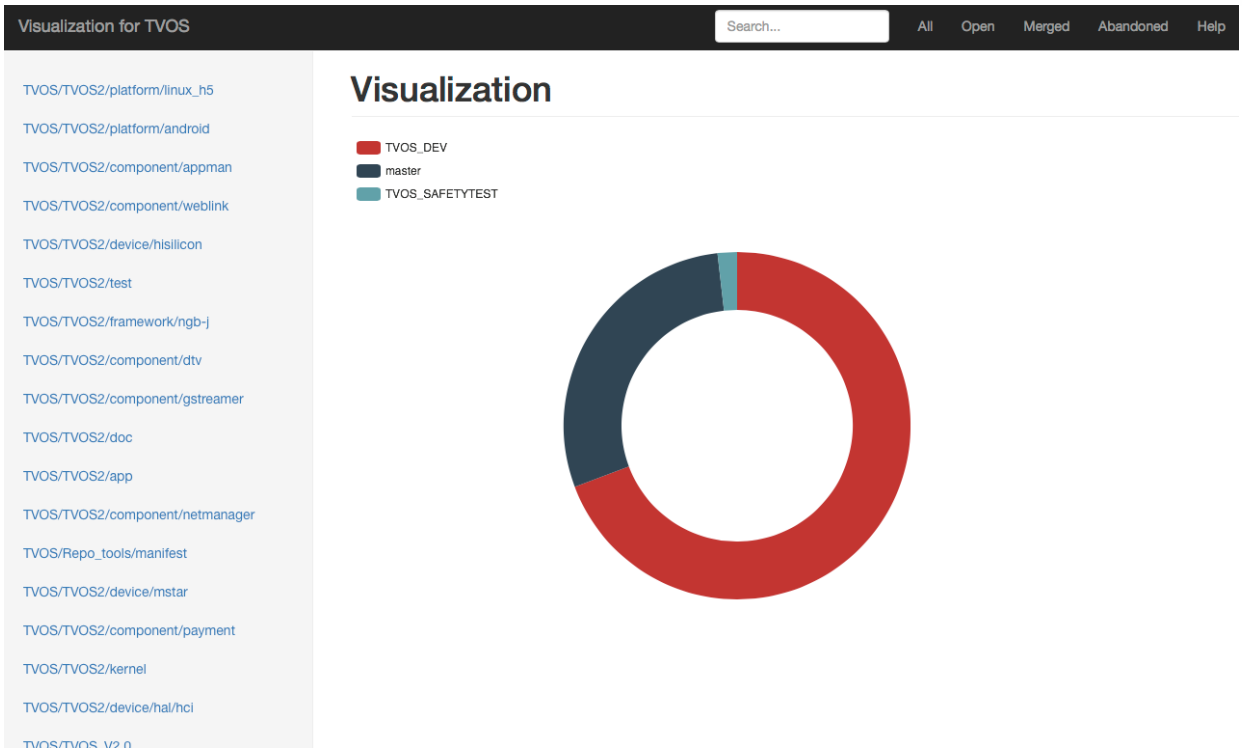
前端部分需要用到jquery，导入jquery文件，以及点击只更新指定div

```
<script type="text/javascript" src="/static/js/jquery-3.2.1.min.js">
</script>
    <script type="text/javascript">
        $(document).ready(function () {
            $("#all").click(function () {
                $('#owner').load('../branch');
            })
        })
    </script>
```

这里载入的`../branch`，是网址对应的上级目录下的branch目录，后台会先在urls.py文件中匹配该网址，找到对应的name参数，再在views.py中找到以name参数命名的函数名，如下

```
def branch(request):
    branch_name = ['TVOS_DEV', 'master', 'TVOS_SAFETYTEST']
    branch_num = [460, 192, 12]
    return render(request, 'branch.html',
{'branchName':json.dumps(branch_name),'branchNum': json.dumps(branch_num)})
```

在该函数中获取数据，加载到branch.html上，branch.html的功能是根据传入的数据绘制Branch图，branch.html就会替换原有的div显示部分，仅该div刷新，如下图：



[这里只是举个例子，将该div显示部分换成Branch图]

2.6.3 网页功能

- 点击导航条的All标签，中间会显示三幅图：修改量-时间，项目分布图，Branch分布图；同理点击Merged和Abandoned标签会显示对应版块下的这三幅图
- 点击左边的项目，会显示该项目修改量-时间的统计，以及该项目各文件夹的活跃程度
- 待增加

3 后记

3.1错误与解决

3.1.1 Django中静态文件路径

比如把`echarts.min.js`文件放置在templates下和index.html同级目录，按理直接使用

```
<script src="echarts.min.js"></script>
```

便可，但是访问时却显示：Not Found charts.min.js

解决： 在app目录下（这个随意，会自动查找static文件夹）新建static文件夹，然后将js文件存放在其中，再在settings.py文件中声明static目录：

```
STATIC_PATH=os.path.join(os.path.dirname(__file__),'static').replace('\\','/')
```

在index.html中使用

```
<script src="/static/echarts.min.js"></script>
```

即可

3.2 tips

3.2.1 CHAR and VARCHAR

在MySQL中

数据类型CHAR表示定长，一般存储身份证号，手机号等，如果不足，会用空格填充，效率极高

数据类型VARCHAR表示不定长，会有一个字节来存储数据的长度，不足不进行填充，效率低，适合变化长度的数据类型

3.2.2 向sql语句中传递参数

传递一条数据：

```
sql='INSERT INTO record_table VALUES ("%d","%s","%s","%s","%d","%d","%s")'
cursor.execute(sql%
(number,project,branch,updated,insertions,deletions,owner))
```

批量导入（一条insert语句插入多条记录，效率高）：

```
sql='INSERT INTO record_table VALUES %s'%'(7,"tvos/das","master","2009-09-
17",22,33,"dang")'
```

这里直接使用executemany效率并不是很高

参考：https://github.com/TsaiZehua/PyMySQL

3.2.3 mysql语句

desc record_table 显示表结构

alter table record_table add column company varchar(16) not null; 添加列

删除表时，如果该表有外键约束，是删除不掉的，需要使用下面语句去除约束：

```
set foreign_key_checks=0
```

3.3 后期安排

1. tvos_info 表的分析问题
2. 数据更新的问题，自动化实现
3. Docker
4. admin静态文件丢失的问题

3.5 不足与改进

随着数据的增多，数值int已经无法表示，需要用到字符串

4 参考

【1】 Django官方文档

【2】自强学堂 Django基础教程

【3】ECharts

【4】Bootstrap官方文档

【5】Django+Vue

绘图processon

跨域

https://segmentfault.com/a/1190000000718840

5 附录

5.1 数据获取部分代码

5.1.1 获取三个版块的json文件

```
import json
import requests


#获取三个版块的json文件
def get_record_data():
    #动态抓包获取各版块的json文件网址,存储在字典中
    section_dict={}
    section_dict['open']='http://120.25.200.39:8081/changes/?n=25&O=81'
    section_dict['merged']='http://120.25.200.39:8081/changes/?
q=status:merged&n=25&O=81'
    section_dict['abandoned']='http://120.25.200.39:8081/changes/?
q=status:abandoned&n=25&O=81'

    filename_list=[]#获取记录的文件名

    #遍历访问网址获取数据，以json存储在data中
    for key ,value in section_dict.items():
        flag = 1
        num = 0
        while flag:
            json_open = requests.get(value + "&S=" + str(num)).text#获取文件
的文本形式
            json_open = json_open.strip().split(']}\'\n')[1]#去除文本中的第一
行
            #这里小于5表示空，即next到头了，结束next；否则将数据写入文件
            if len(json_open) < 5:
                flag = 0
            else:
                with open('data/record/'+key+'_' + str(num) + '.json', 'w')
as f:
                    f.write(json_open)
                    filename_list.append(key+'_'+str(num))
                    print(key+str(num)+"has wirtten !")
            num += 25
    return filename_list
```

5.1.2 获取记录文件中对应的详细json文件

```
#获取记录详细信息
def get_record_info_data(filename_list):
    for filename in filename_list:
        with open('data/record/'+filename+'.json','r') as f:
            records_json=json.load(f)
        for record in records_json:
            number=record['_number']
            print(number)

 url_str2='http://120.25.200.39:8081/changes/'+str(number)+'/detail?O=404'
            detail=requests.get(url_str2).text
            detail = detail.strip().split(']}\'\n')  # 去除文本中的第一行
            if len(detail)<2:continue
            detail=detail[1]
            detail='['+detail+']'
            with open ('data/info/'+filename.split('_')
[0]+'/detail/detail'+str(number)+'.json','w') as f:
                f.write(detail)
            with open ('data/info/'+filename.split('_')
[0]+'/detail/detail'+str(number)+'.json','r') as f:
                detail=json.load(f)
                print(filename+"detail"+str(number)+"has written!")
            revision=detail[0]['current_revision']

            url_str1 =
'http://120.25.200.39:8081/changes/'+str(number)+'/revisions/'+str(revision
)+'/files'
            info=requests.get(url_str1).text
            info = info.strip().split(']}\'\n') # 去除文本中的第一行
            if len(info)<2:continue
            info=info[1]
            with open('data/info/' + filename.split('_')[0] +"/" +
str(number) + '.json','w') as f:
                f.write(info)
                print(filename+str(number)+"has written!")
```

5.2 数据存储部分代码

5.2.1 tvos_record表的数据插入

```
#将全部record数据插入到tvos_record中
def insert_record_with_json():
    filelist = os.listdir('data/record')  # 获取data/record目录下的全部文件名
    batch_list=[]#存放记录
    db=connect_db()#连接数据库
    for filename in filelist:
        with open('data/record/' + str(filename), 'r') as f:
            record_data = json.load(f)
        #获取section
        if re.match(r'^abandoned.*',filename):section='abandoned'
        elif re.match(r'^merged.*',filename):section='merged'
        else:section='open'

        if len(record_data) < 1: continue
        for record in record_data:
            record_list=[]
            number=record['_number']
            project = record['project']
            branch = record['branch']
            updated= record['updated'][:10]
            insertions=record['insertions']
            deletions=record['deletions']
            owner=record['owner']['name']
            company=re.findall(r'@([a-zA-Z\-]+)',record['owner']['email'])
            if len(company)==0:company='None'#找不到公司名，则用None代替
            else:company=company[0]
            record_list.append(number)
            record_list.append(project)
            record_list.append(branch)
            record_list.append(updated)
            record_list.append(insertions)
            record_list.append(deletions)
            record_list.append(owner)
            record_list.append(company)
            record_list.append(section)
            batch_list.append(record_list)

            # 一次insert插入20条记录
            if len(batch_list) >=20:
                insert_db_batch(db, 'tvos_record',batch_list)
                batch_list = []

    if len(batch_list)>=1:
        insert_db_batch(db,'tvos_record', batch_list)
    close_db(db)
```

5.2.2 tvos_info表的数据插入

```
#将全部info数据插入到tvos_info中
def insert_info_with_json():
    dir=['open','merged','abandoned']
    insertions_dict={}
    deletions_dict={}
    num_dict={}
    db=connect_db()
    for d in dir:
        filelist = os.listdir('data/info/'+d)  # 获取data/record目录下的全部文
件名

        for filename in filelist:
            num=re.findall(r'([0-9]+).json$',filename)#获取json文件
            if len(num)!=0:
                #查找number对应的project
                sql='SELECT project FROM tvos_record WHERE number='+num[0];
                cursor=db.cursor()
                cursor.execute(sql)
                result =cursor.fetchall()
                for row in result:
                    project=row[0]

                with open('data/info/' +d+'/'+ filename, 'r') as f:
                    info_data = json.load(f)
                #读取json中的路径以及对应的insert和delete
                for info in info_data:
                    project_filepath=project+'#'+info #以#为分隔符，将项目名和
路径分开
                    if project_filepath not in num_dict.keys():
                        num_dict[project_filepath]=0
                        deletions_dict[project_filepath]=0
                        insertions_dict[project_filepath]=0
                    num_dict[project_filepath]+=1
                    if 'lines_inserted' in info_data[info]:
```

```
                        insertions_dict[project_filepath]+=info_data[info]
['lines_inserted']
                    if 'lines_deleted' in info_data[info]:
                        deletions_dict[project_filepath]+=info_data[info]
['lines_deleted']

    batch_list=[]
    info_list=[]
    id=0
    for profile in num_dict.keys():
        pro_file=profile.split('#')
        if num_dict[profile]<2:continue   #修改次数小于2的记录不添加到数据库
        id+=1
        #按照tvos_info表插入一条记录
        info_list.append(id)#id
        info_list.append(pro_file[1])#filepath
        info_list.append(pro_file[0])#project
        info_list.append(insertions_dict[profile])#insertions
        info_list.append(deletions_dict[profile])#deletions
        info_list.append(num_dict[profile])#num
        batch_list.append(info_list)
        info_list=[]
        # 一次insert插入20条记录
        if len(batch_list) >= 20:
            insert_db_batch(db,'tvos_info', batch_list)
            batch_list = []

    if len(batch_list) >= 1:
        insert_db_batch(db,'tvos_info', batch_list)
    close_db(db)
```

5.3 Django后台搭建部分代码

5.3.1 view.py的部分代码

各版块数据获取

```
#django模板会自动找到app下面的templates文件夹中的模板文件
#显示字符串
def index(request):
    sec=request.GET.get('section')
    if(sec==None or sec=='all'):
        record=Record.objects.all()#获取全部记录
    else:
        record = Record.objects.filter(section=sec)#获取各版块的记录

    project_dict={}
    owner_dict={}
    company_dict={}
    date_list=[]
    insert_list=[]
    delete_list=[]
    project_name=[]
    project_num=[]
    owner_name=[]
    owner_num=[]
    company_name=[]
    company_num=[]

    #读取数据
    for row in record:
        if row.project not in project_dict.keys():
            project_dict[row.project]=0
        if row.owner not in owner_dict.keys():
            owner_dict[row.owner]=0
        if row.company not in company_dict.keys():
            company_dict[row.company]=0
        project_dict[row.project]+=1
        owner_dict[row.owner]+=1
        company_dict[row.company]+=1
        date_list.append(str(row.updated))
        insert_list.append(row.insertions)
        delete_list.append(row.deletions)

    for key,value in project_dict.items():
        project_name.append(key)
        project_num.append(value)
    for key,value in owner_dict.items():
        owner_name.append(key)
        owner_num.append(value)
    for key,value in company_dict.items():
        company_name.append(key)
```

```
        company_num.append(value)

    return HttpResponse(json.dumps({'dateList': date_list,
                    'insertList': insert_list, 'deleteList': delete_list,
                    'ownerName': owner_name, 'ownerNum': owner_num,
                    'companyName': company_name, 'companyNum': company_num,
                    'projectName': project_name, 'projectNum':
project_num}))
```