预警系统前后端开发

已知区域涌水量预警系统后端开发

models.py编写

根据表写对应模型类

字段名称	字段代码	字段类型	非空	备注
日期	date	varchar	false	日期不能为空
经度	longitude	double	false	
纬度	latitude	double	false	
高程	altitude	double	false	
日降水量	rainfall	double	true	可以为空
涌水量	water_quantity	double	true	可以为空，但需要处理

```
#文件backend/crud_demo/models.py

from django.db import models

# Create your models here.
from dvadmin.utils.models import CoreModel


class WaterInfoModel(CoreModel):
    '''
    涌水量信息数据模型，用于预测已知地区的涌水量:
    日期
    横坐标
    纵坐标
    高程
    日降雨量
    涌水量
    '''
    date = models.DateField(verbose_name="日期")
    longitude = models.DecimalField(max_digits=24, decimal_places=6,verbose_name="经
度")
    latitude = models.DecimalField(max_digits=24, decimal_places=6,verbose_name="纬度")
    altitude = models.DecimalField(max_digits=24, decimal_places=6, verbose_name="高
程")
    rainfall = models.DecimalField(max_digits=24, decimal_places=6, verbose_name="日降
水量")
    water_quantity = models.DecimalField(max_digits=24, decimal_places=6,
verbose_name="涌水量")

    class Meta:
        db_table = "water_info"
        verbose_name = '涌水量信息表'
        verbose_name_plural = verbose_name
        ordering = ('longitude','latitude','altitude','date')
```

serializers.py 编写

```
from prediction.models import WaterInfoModel
from dvadmin.utils.serializers import CustomModelSerializer


class WaterInfoModelSerializer(CustomModelSerializer):
    """
    序列化器
    """
#这里是进行了序列化模型及所有的字段
    class Meta:
        model = WaterInfoModel
        fields = "__all__"

#这里是创建/更新时的列化器
class WaterInfoModelCreateUpdateSerializer(CustomModelSerializer):
    """
    创建/更新时的列化器
    """

    class Meta:
        model = WaterInfoModel
        fields = '__all__'
```

view.py编写

```
from prediction_system.models import WaterInfoModel
from prediction_system.serializers import WaterInfoModelSerializer,
WaterInfoModelCreateUpdateSerializer
from dvadmin.utils.viewset import CustomModelViewSet


class WaterInfoModelViewSet(CustomModelViewSet):
    """
    list:查询
    create:新增
    update:修改
    retrieve:单例
    destroy:删除
    """
    queryset = WaterInfoModel.objects.all()
    serializer_class = WaterInfoModelSerializer
    create_serializer_class = WaterInfoModelCreateUpdateSerializer
    update_serializer_class = WaterInfoModelCreateUpdateSerializer
```

urls.py编写

```
#backend/crud_demo/urls.py

from rest_framework.routers import SimpleRouter

from .views import WaterInfoModelViewSet

router = SimpleRouter()
# 这里进行注册路径，并把视图关联上，这里的api地址以视图名称为后缀，这样方便记忆
api/WaterInfoModelViewSet
router.register("api/WaterInfoModelViewSet", WaterInfoModelViewSet)

urlpatterns = [
]
urlpatterns += router.urls
```

在application的urls里导入我们的app

```
My_Urls = (
    [    #这里的crud_demo是指django创建的应用名称crud_demo
        path('',include('crud_demo.urls')),
        path('',include('prediction_system.urls'))
    ]
)

# 这里把自己的路径单独出来，后面再追加在一起
urlpatterns += My_Urls
```

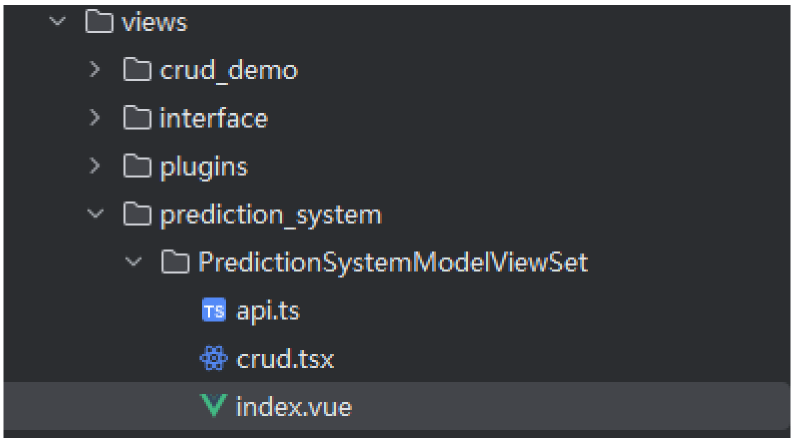
执行迁移

```
python manage.py makemigrations
python manage.py migrate
```

已知区域涌水量预警系统前端开发

前端目录

在web下找到views目录，并创建如下目录以及文件


```
views
  crud_demo
  interface
  plugins
  prediction_system
    PredictionSystemModelViewSet
      api.ts
      crud.tsx
      index.vue
```

api.ts

该文件实现添删改查请求接口，实际开发中，修改apiPrefixr的值即可，你也可以在此页面根据实际业务需求增加和修改方法。

API模块封装，通过函数封装了与后端交互的细节，使得在业务组件中调用变得简单。

注意：export const apiPrefix = '/api/WaterInfoModelViewSet'此处的前缀需要与后端的相同，可以去后端的两个urls文件进行查看。

}

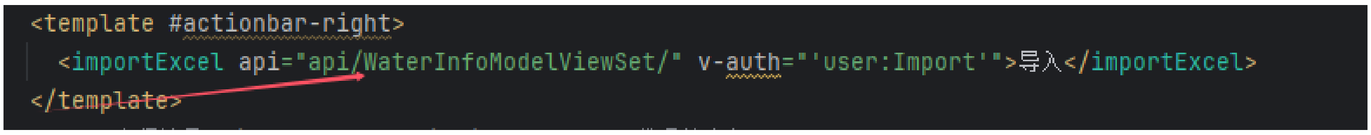
```
import { request,downloadFile } from '/@/utils/service';
import { PageQuery, AddReq, DelReq, EditReq, InfoReq } from '@fast-crud/fast-crud';

export const apiPrefix = '/api/WaterInfoModelViewSet/';

export function GetList(query: PageQuery) {
    return request({
        url: apiPrefix,                  // 请求地址：/api/WaterInfoModelViewSet/
        method: 'get',                   // GET 请求
        params: query,                   // 查询参数（分页、过滤等）
    });
}
export function GetObj(id: InfoReq) {
    return request({
        url: apiPrefix + id,             // 请求地址：/api/WaterInfoModelViewSet/{id}
        method: 'get',                   // GET 请求
    });
}

export function AddObj(obj: AddReq) {
    return request({
        url: apiPrefix,                  // 请求地址：/api/WaterInfoModelViewSet/
        method: 'post',                  // POST 请求
        data: obj,                       // 要添加的对象数据
    });
}

export function UpdateObj(obj: EditReq) {
    return request({
        url: apiPrefix + obj.id + '/',       // 请求地
址：/api/WaterInfoModelViewSet/{id}/
        method: 'put',                       // PUT 请求
        data: obj,                           // 更新后的对象数据
    });
}

export function DelObj(id: DelReq) {
    return request({
        url: apiPrefix + id + '/',           // 请求地
址：/api/WaterInfoModelViewSet/{id}/
        method: 'delete',                    // DELETE 请求
        data: { id },                        // 要删除的对象ID（可选，有些API需要）
    });
}

export function exportData(params:any){
    // 使用专门的下载文件函数
    return downloadFile({
        url: apiPrefix + 'export_data/',         // 导出地
址：/api/WaterInfoModelViewSet/export_data/
```

```
        params: params,                           // 导出参数（过滤条件等）
        method: 'get'                             // GET 请求
    })
}
```

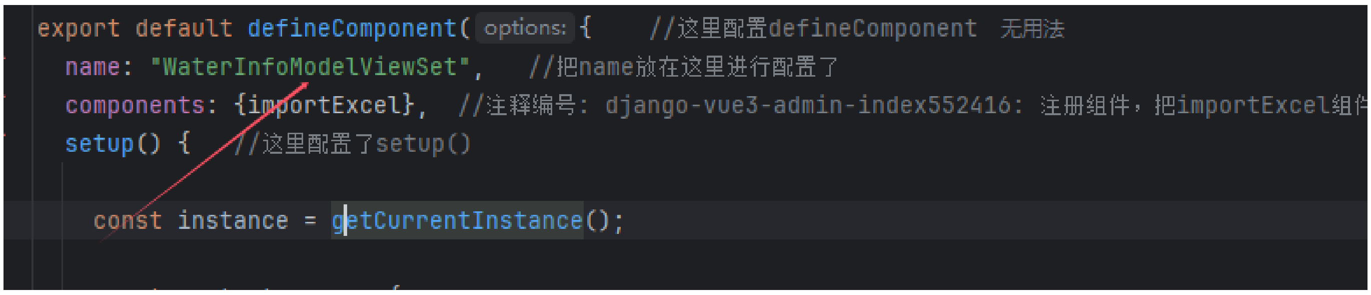
index.vue

文件 `web/src/views/crud_demo/PredictionSystemModelViewSet/index.vue`

主要修改下述api接口以及模块名称：



```
<template #actionbar-right>
  <importExcel api="api/WaterInfoModelViewSet/" v-auth="'user:Import'">导入</importExcel>
</template>
```



```
export default defineComponent( options: {     //这里配置defineComponent  无用法
  name: "WaterInfoModelViewSet",    //把name放在这里进行配置了
  components: {importExcel},  //注释编号: django-vue3-admin-index552416: 注册组件，把importExcel组件
  setup() {   //这里配置了setup()

    const instance = getCurrentInstance();
```

```
<template>
  <fs-page class="PageFeatureSearchMulti">
    <fs-crud ref="crudRef" v-bind="crudBinding">
      <template #cell_url="scope">
        <el-tag size="small">{{ scope.row.url }}</el-tag>
      </template>
      <!-- 注释编号: django-vue3-admin-index442216: -->
      <!-- 注释编号:django-vue3-admin-index39263917:代码开始行-->
      <!--  功能说明:使用导入组件，并且修改api地址为当前对应的api，当前是demo的
api="api/CrudDemoModelViewSet/"-->
      <template #actionbar-right>
        <importExcel api="api/WaterInfoModelViewSet/" v-auth="'user:Import'">导入
</importExcel>
      </template>
      <!--  注释编号:django-vue3-admin-index263917:代码结束行-->

    </fs-crud>
  </fs-page>
</template>

<script lang="ts">
import { onMounted, getCurrentInstance, defineComponent} from 'vue';
import { useFs } from '@fast-crud/fast-crud';
import createCrudOptions  from './crud';

// 注释编号: django-vue3-admin-index192316:导入组件
import importExcel from '/@/components/importExcel/index.vue'


export default defineComponent({    //这里配置defineComponent
  name: "WaterInfoModelViewSet",   //把name放在这里进行配置了
  components: {importExcel},  //注释编号: django-vue3-admin-index552416: 注册组件，把
importExcel组件放在这里，这样<template></template>中才能正确的引用到组件
  setup() {   //这里配置了setup()

    const instance = getCurrentInstance();

    const context: any = {
      componentName: instance?.type.name
    };

    const { crudBinding, crudRef, crudExpose, resetCrudOptions } = useFs({
createCrudOptions, context});


    // 页面打开后获取列表数据
    onMounted(() => {
      crudExpose.doRefresh();
    });
    return {
      //增加了return把需要给上面<template>内调用的<fs-crud ref="crudRef" v-
bind="crudBinding">
      crudBinding,
      crudRef,
    };


  }     //这里关闭setup()
});  //关闭defineComponent

</script>
```

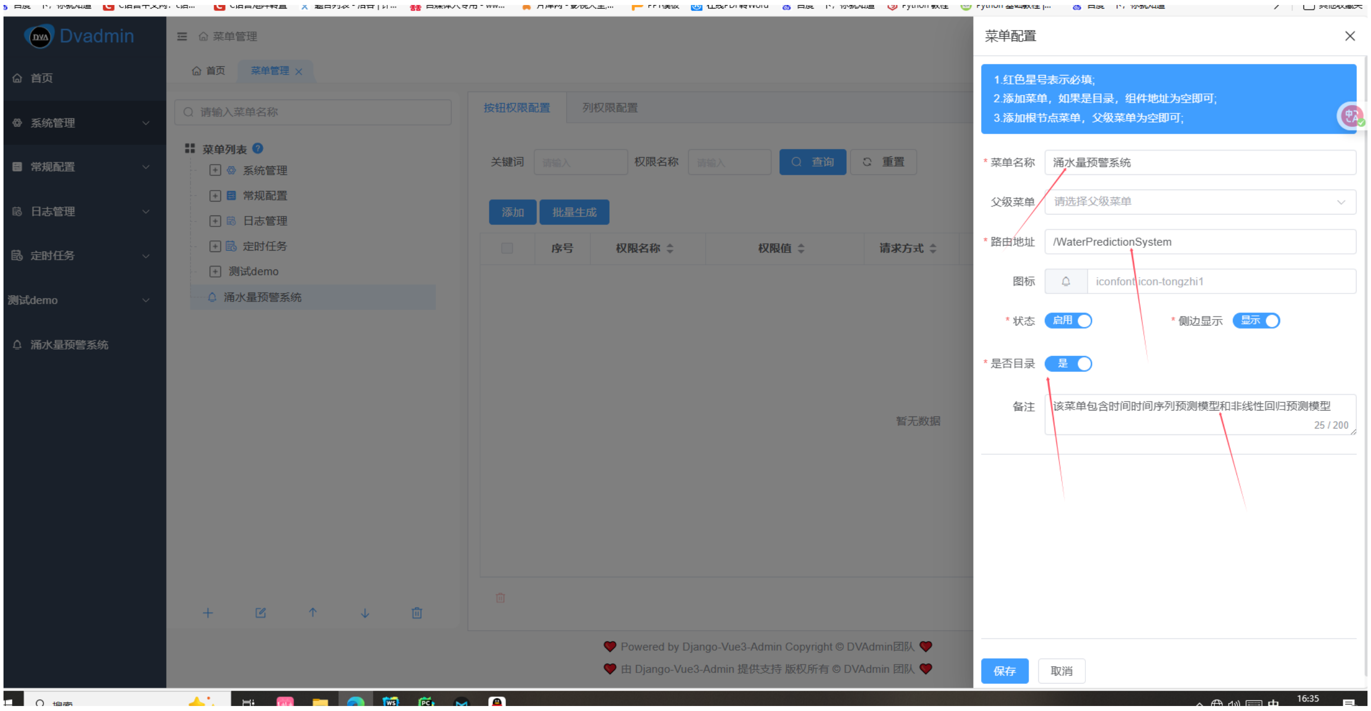
crud.tsx

主要修改columns中的数据字段，确保与后端数据模型一致

```
import { CrudOptions, AddReq, DelReq, EditReq, dict, CrudExpose, UserPageQuery,
CreateCrudOptionsRet} from '@fast-crud/fast-crud';
import _ from 'lodash-es';
import * as api from './api';
import { request } from '/@/utils/service';
import {auth} from "/@/utils/authFunction";

//此处为crudOptions配置
export default function ({ crudExpose}: { crudExpose: CrudExpose}):
CreateCrudOptionsRet {
    const pageRequest = async (query: any) => {
        return await api.GetList(query);
    };
    const editRequest = async ({ form, row }: EditReq) => {
        if (row.id) {
            form.id = row.id;
        }
        return await api.UpdateObj(form);
    };
    const delRequest = async ({ row }: DelReq) => {
        return await api.DelObj(row.id);
    };
    const addRequest = async ({ form }: AddReq) => {
        return await api.AddObj(form);
    };

    const exportRequest = async (query: UserPageQuery) => {
        return await api.exportData(query)
    };

    return {
        crudOptions: {
            request: {
                pageRequest,
                addRequest,
                editRequest,
                delRequest,
            },
            actionbar: {
                buttons: {
                    export:{
                        // 注释编号:django-vue3-admin-crud210716:注意这个auth里面的值，
最好是使用index.vue文件里面的name值并加上请求动作的单词
                        show: auth('WaterInfoModelViewSet:Export'),
                        text:"导出",//按钮文字
                        title:"导出",//鼠标停留显示的信息
                        click(){
                            return exportRequest(crudExpose.getSearchFormData())
                            // return exportRequest(crudExpose!.getSearchFormData())
// 注意这个crudExpose!.getSearchFormData()，一些低版本的环境是需要添加!的
```

```
                    }
                },
                add: {
                    show: auth('WaterInfoModelViewSet:Create'),
                },
            }
        },
        rowHandle: {
            //固定右侧
            fixed: 'right',
            width: 200,
            buttons: {
                view: {
                    type: 'text',
                    order: 1,
                    show: auth('WaterInfoModelViewSet:Retrieve')
                },
                edit: {
                    type: 'text',
                    order: 2,
                    show: auth('WaterInfoModelViewSet:Update')
                },
                copy: {
                    type: 'text',
                    order: 3,
                    show: auth('WaterInfoModelViewSet:Copy')
                },
                remove: {
                    type: 'text',
                    order: 4,
                    show: auth('WaterInfoModelViewSet:Delete')
                },
            },
        },
        columns: {
            date: {
                title: '日期',
                type: 'date',
                search: { show: true},
                column: {
                    minWidth: 120,
                    sortable: 'custom',
                },
                form: {
                    rules: [{ required: true, message: '日期必填' }],
                    component: {
                        format: "YYYY-MM-DD",
                        valueFormat: "YYYY-MM-DD",
                        placeholder: '请选择日期',
                    },
                },
                },
                longitude: {
                    title: '经度',
                    type: 'number',
                    search: { show: true },
                    column: {
                        minWidth: 120,
                        sortable: 'custom',
                    },
                    form: {
                        rules: [{ required: true, message: '经度必填' }],
                        component: {
                            placeholder: '请输入经度',
                            // 控制小数位数为6位
                            step: 0.000001,
                        },
                    },
                },

                latitude: {
                    title: '纬度',
                    type: 'number',
                    search: { show: true },
                    column: {
                        minWidth: 120,
                        sortable: 'custom',
                    },
                    form: {
                        rules: [{ required: true, message: '纬度必填' }],
                        component: {
                            placeholder: '请输入纬度',
                            step: 0.000001,
                        },
                    },
                },
                altitude: {
                    title: '高程',
                    type: 'number',
                    search: { show: false },
                    column: {
                        minWidth: 120,
                        sortable: 'custom',
                    },
                    form: {
                        rules: [{ required: true, message: '高程必填' }],
                        component: {
                            placeholder: '请输入高程',
                            step: 0.000001,
                        },
                    },
                },
                rainfall: {
                    title: '日降雨量',
                    type: 'number',
                    search: { show: false },
                    column: {
                        minWidth: 120,
                        sortable: 'custom',
                    },
                    form: {
                        rules: [{ required: true, message: '日降雨量必填' }],
                        component: {
                            placeholder: '请输入日降雨量',
                            step: 0.000001,
                        },
                    },
                },
                water_quantity: {
                    title: '涌水量',
                    type: 'number',
                    search: {show: false},
                    column: {
                        minWidth: 120,
                        sortable: 'custom',
                    },
                    form: {
                        rules: [{required: true, message: '涌水量必填'}],
                        component: {
                            placeholder: '请输入涌水量',
                            step: 0.000001,
                        },
                    },
                },
            },
        },
    };
}
```

配置菜单

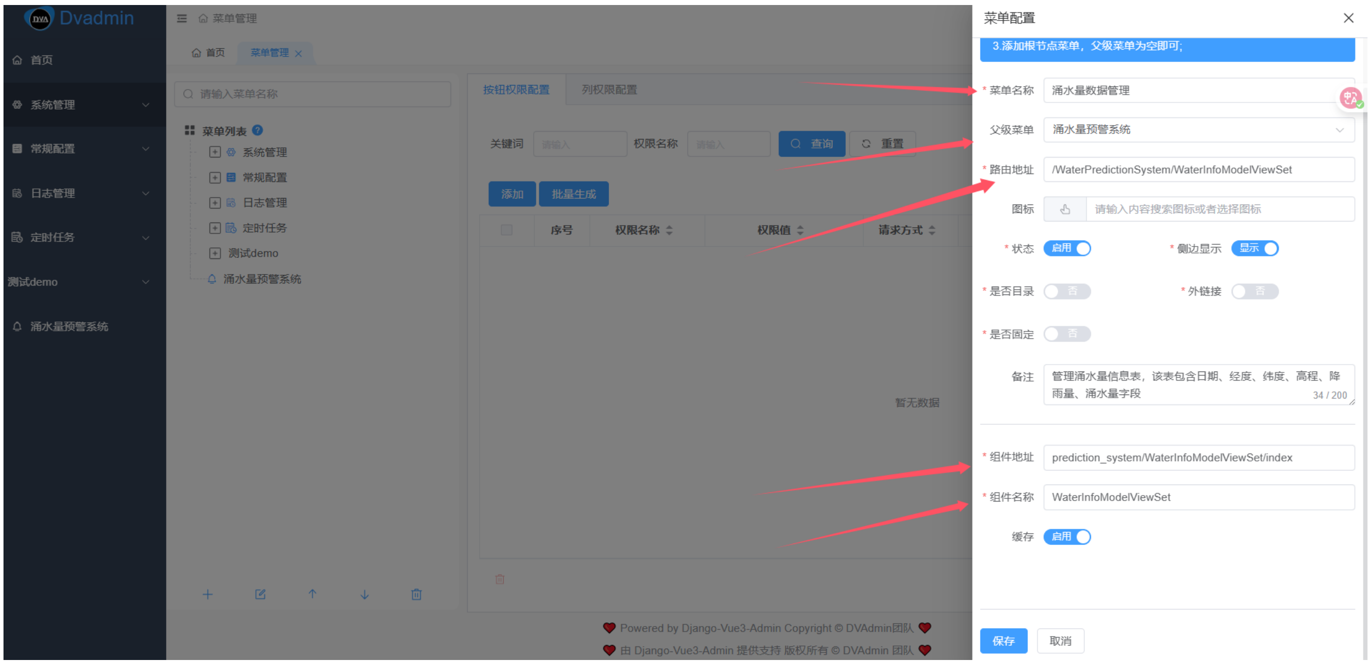

菜单配置

1.红色星号表示必填;
2.添加菜单，如果是目录，组件地址为空即可;
3.添加根节点菜单，父级菜单为空即可;

* 菜单名称	涌水量预警系统
父级菜单	请选择父级菜单
* 路由地址	/WaterPredictionSystem
图标	iconfont icon-tongzhi1
* 状态	启用
* 侧边显示	显示
* 是否目录	是
备注	该菜单包含时间时间序列预测模型和非线性回归预测模型 25 / 200

保存 取消

在此目录下，配置子菜单，注意组件地质为index所在目录，组件地址为index代码里面name取值：


菜单配置

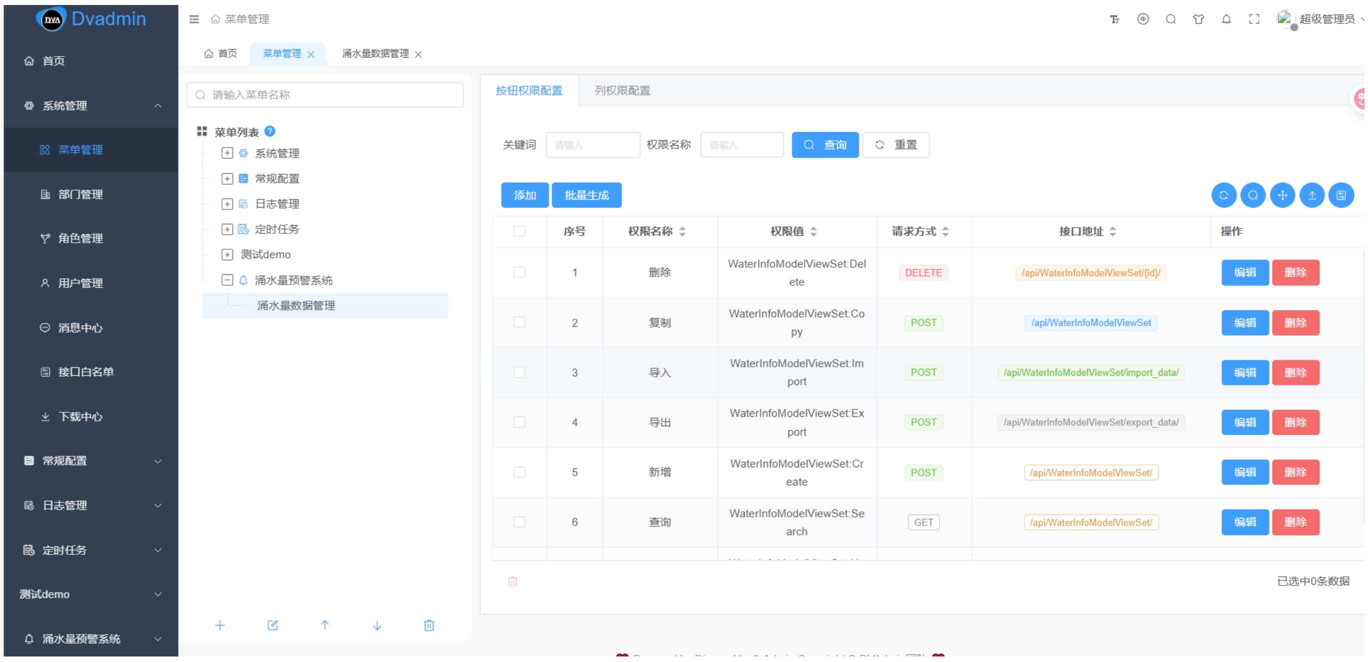
3.添加根节点菜单，父级菜单为空即可;

* 菜单名称	涌水量数据管理
父级菜单	涌水量预警系统
* 路由地址	/WaterPredictionSystem/WaterInfoModelViewSet
图标	请输入内容搜索图标或者选择图标
* 状态	启用
* 侧边显示	显示
* 是否目录	否
* 外链接	否
* 是否固定	否
备注	管理涌水量信息表，该表包含日期、经度、纬度、高程、降雨量、涌水量字段 34 / 200
* 组件地址	prediction_system/WaterInfoModelViewSet/index
* 组件名称	WaterInfoModelViewSet
缓存	启用

保存 取消

配置按钮权限

序号	权限名称	权限值	请求方式	接口地址
1	新增	WaterInfoModelViewSet:Create	POST	/api/WaterInfoModelViewSet
2	编辑	WaterInfoModelViewSet:Update	PUT	/api/WaterInfoModelViewSet/{id}/
3	删除	WaterInfoModelViewSet:Delete	DELETE	/api/WaterInfoModelViewSet/{id}/
4	查询	WaterInfoModelViewSet:Search	GET	/api/WaterInfoModelViewSet
5	查看	WaterInfoModelViewSet:Retrieve	GET	/api/WaterInfoModelViewSet/{id}/
6	导出	WaterInfoModelViewSet:Export	POST	/api/WaterInfoModelViewSet/expo
7	导入	WaterInfoModelViewSet:Import	POST	/api/WaterInfoModelViewSet/impo
8	复制	WaterInfoModelViewSet:Copy	POST	/api/WaterInfoModelViewSet



	序号	权限名称	权限值	请求方式	接口地址	操作
	1	删除	WaterInfoModelViewSet:Delete	DELETE	/api/WaterInfoModelViewSet/{id}/	编辑 删除
	2	复制	WaterInfoModelViewSet:Copy	POST	/api/WaterInfoModelViewSet	编辑 删除
	3	导入	WaterInfoModelViewSet:Import	POST	/api/WaterInfoModelViewSet/import_data/	编辑 删除
	4	导出	WaterInfoModelViewSet:Export	POST	/api/WaterInfoModelViewSet/export_data/	编辑 删除
	5	新增	WaterInfoModelViewSet:Create	POST	/api/WaterInfoModelViewSet/	编辑 删除
	6	查询	WaterInfoModelViewSet:Search	GET	/api/WaterInfoModelViewSet/	编辑 删除

已选中0条数据

导入导出配置

在serializers.py中单独添加导入和导出的序列化器

```
#导入时用到的列化器
class WaterInfoModelImportSerializer(CustomModelSerializer):
    """
    WaterInfoModel导入时的列化器
    """

    class Meta:
        model = WaterInfoModel
        fields = '__all__'

#导出时用到的列化器
class ExportWaterInfoModelSerializer(CustomModelSerializer):
    """
    WaterInfoModel导入时的列化器
    """

    class Meta:
        model = WaterInfoModel
        fields = '__all__'
```

在view.py中配置相关信息

需要引入WaterInfoModelImportSerialize和ExportWaterInfoModelSerializer

```
from prediction_system.models import WaterInfoModel
from prediction_system.serializers import WaterInfoModelSerializer,
WaterInfoModelCreateUpdateSerializer， WaterInfoModelImportSerializer,
ExportWaterInfoModelSerializer
from dvadmin.utils.viewset import CustomModelViewSet


class WaterInfoModelViewSet(CustomModelViewSet):
    """
    list:查询
    create:新增
    update:修改
    retrieve:单例
    destroy:删除
    """
    # 功能说明:导入的配置
    import_field_dict = {
        "date": {
            "title": '日期',
            "display":'date',
            "type": "date"
        },
        "longitude": "经度",
        "latitude": "纬度",
        "altitude": "高程",
        "rainfall": "日降雨量",
        "water_quantity": "涌水量"
    }

    # 导入序列化器
    import_serializer_class = WaterInfoModelImportSerializer

    export_field_label = {
        "date": {
            "title": '日期',
            "display": 'date',
            "type": "date"
        },
        "longitude": "经度",
        "latitude": "纬度",
        "altitude": "高程",
        "rainfall": "日降雨量",
        "water_quantity": "涌水量"
    }

    # 导入序列化器
    import_serializer_class = WaterInfoModelImportSerializer
    # 导出序列化器
    export_serializer_class = ExportWaterInfoModelSerializer
    queryset = WaterInfoModel.objects.all()
    serializer_class = WaterInfoModelSerializer
    create_serializer_class = WaterInfoModelCreateUpdateSerializer
    update_serializer_class = WaterInfoModelCreateUpdateSerializer
```

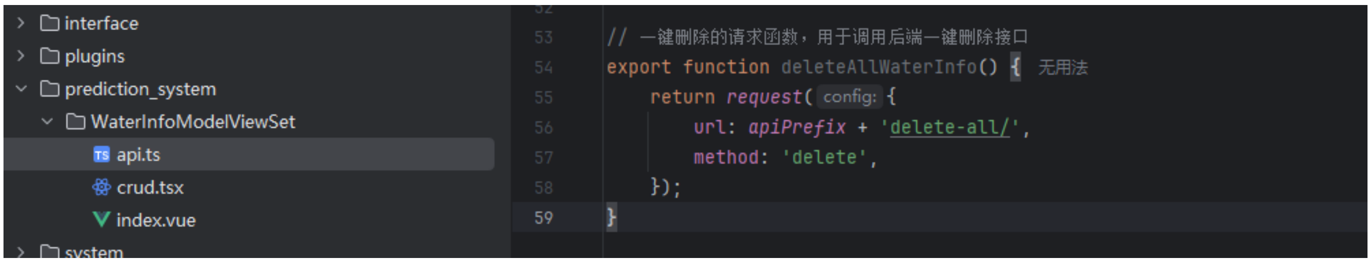
开发一键删除接口

在view.py自定义接口

对于增删改查接口，框架会自己生成相应的接口，而对于一键删除，则需要自定义，定义如下：

```python
from prediction_system.models import WaterInfoModel
from prediction_system.serializers import WaterInfoModelSerializer,
WaterInfoModelCreateUpdateSerializer, WaterInfoModelImportSerializer,
ExportWaterInfoModelSerializer
from dvadmin.utils.viewset import CustomModelViewSet
# 需要引入的模块
from django.db import transaction
from rest_framework.decorators import action
from rest_framework.response import Response
from rest_framework import status



class WaterInfoModelViewSet(CustomModelViewSet):
    """
    list:查询
    create:新增
    update:修改
    retrieve:单例
    destroy:删除
    """
    # 功能说明:导入的配置
    import_field_dict = {
        "date": {
            "title": '日期',
            "display":'date',
            "type": "date"
        },
        "longitude": "经度",
        "latitude": "纬度",
        "altitude": "高程",
        "rainfall": "日降雨量",
        "water_quantity": "涌水量"
    }

    # 导入序列化器
    import_serializer_class = WaterInfoModelImportSerializer

    export_field_label = {
        "date": {
            "title": '日期',
            "display": 'date',
            "type": "date"
        },
        "longitude": "经度",
        "latitude": "纬度",
        "altitude": "高程",
        "rainfall": "日降雨量",
        "water_quantity": "涌水量"
    }
    # 导入序列化器
    import_serializer_class = WaterInfoModelImportSerializer
    # 导出序列化器
    export_serializer_class = ExportWaterInfoModelSerializer

    queryset = WaterInfoModel.objects.all()
    serializer_class = WaterInfoModelSerializer
    create_serializer_class = WaterInfoModelCreateUpdateSerializer
    update_serializer_class = WaterInfoModelCreateUpdateSerializer

    # 一键删除的接口自定义
    @action(detail=False, methods=['delete'], url_path='delete-all')
    def delete_all(self, request):
        """
        删除所有涌水量信息数据
        """
        # 添加权限检查（可选但推荐）
        if not request.user.has_perm('prediction_system.delete_waterinfomodel'):
            return Response({'detail': '权限不足'}, status=status.HTTP_403_FORBIDDEN)

        # 执行删除操作
        try:
            # 使用事务确保操作的原子性
            with transaction.atomic():
                deleted_count, _ = self.get_queryset().delete()
            return Response({'message': f'成功删除 {deleted_count} 条数据'},
status=status.HTTP_200_OK)
        except Exception as e:
            return Response({'detail': f'删除失败: {str(e)}'},
status=status.HTTP_500_INTERNAL_SERVER_ERROR)
```

api.ts中添加请求函数

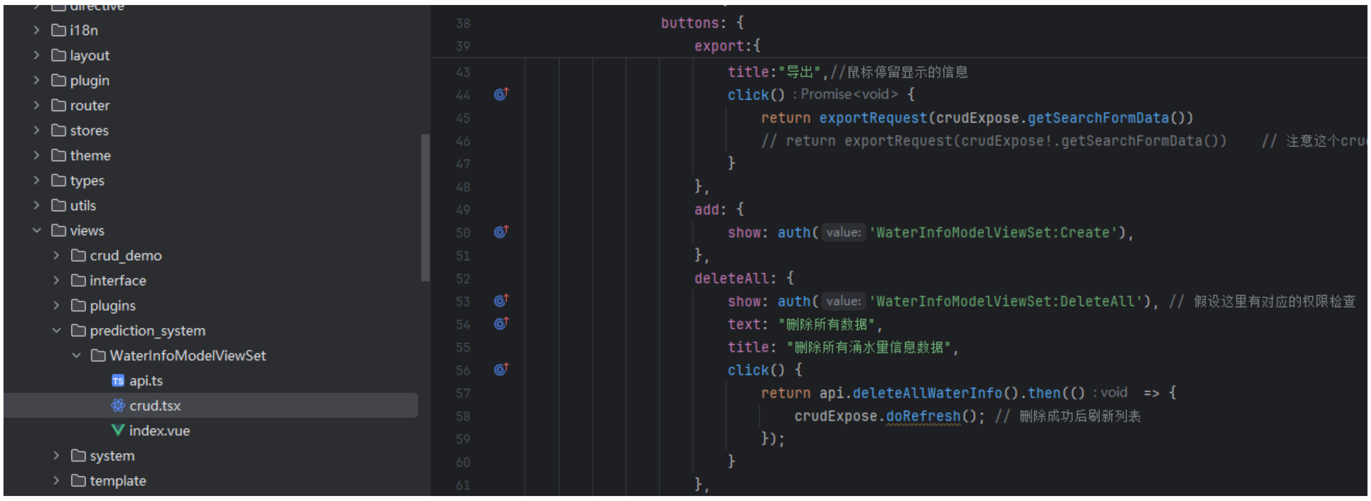


```
// 一键删除的请求函数，用于调用后端一键删除接口
export function deleteAllWaterInfo() {  无用法
    return request( config: {
        url: apiPrefix + 'delete-all/',
        method: 'delete',
    });
}
```

```
// 一键删除的请求函数，用于调用后端一键删除接口
export function deleteAllWaterInfo() {
    return request({
        url: apiPrefix + 'delete-all/',
        method: 'delete',
    });
}
```

在crud.tsx的顶部操作栏中添加一个一键删除的按钮

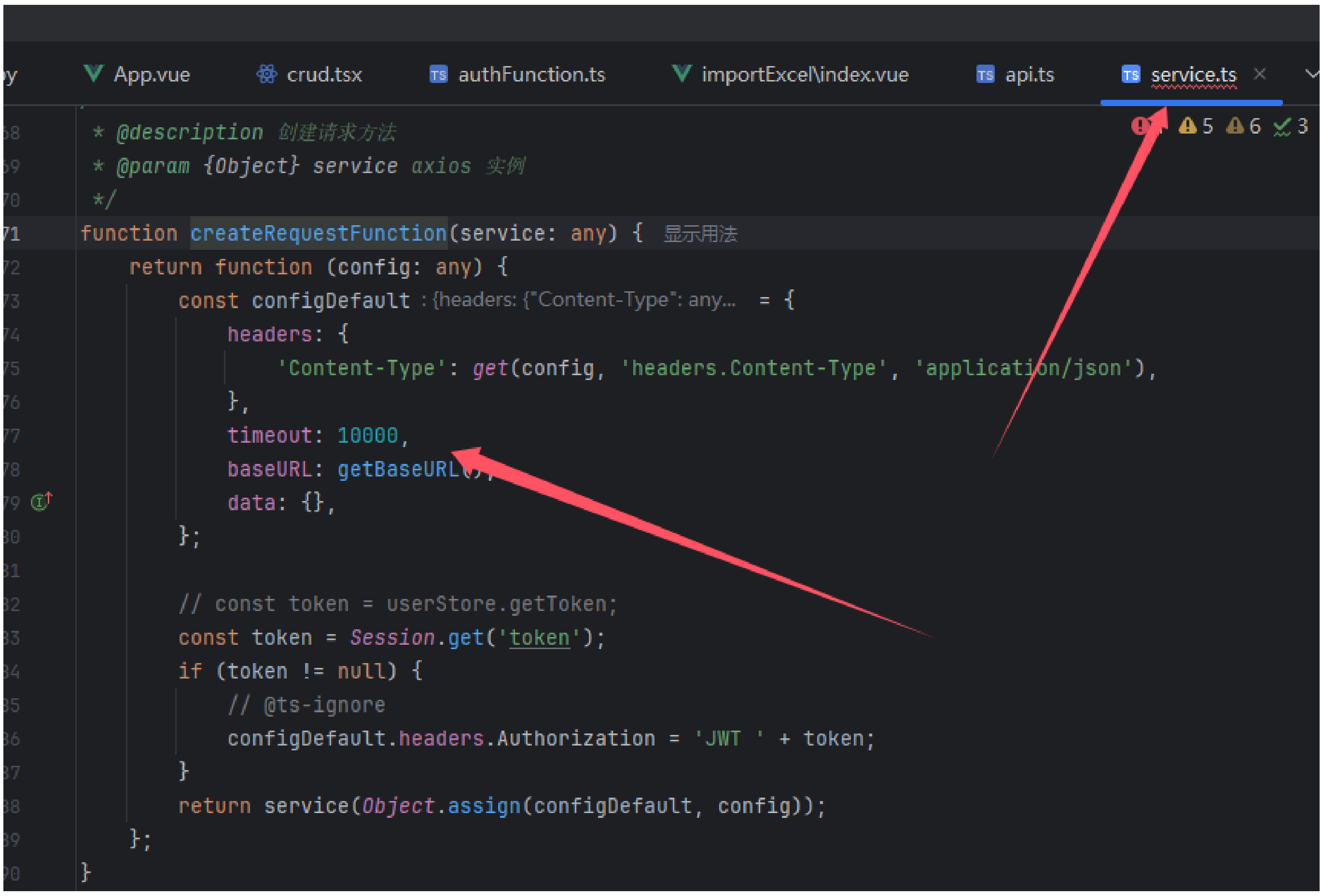
```
buttons: {
                    export:{
                        // 注释编号:django-vue3-admin-crud210716:注意这个auth里面的值，
最好是使用index.vue文件里面的name值并加上请求动作的单词
                        show: auth('WaterInfoModelViewSet:Export'),
                        text:"导出",//按钮文字
                        title:"导出",//鼠标停留显示的信息
                        click(){
                            return exportRequest(crudExpose.getSearchFormData())
                            // return exportRequest(crudExpose!.getSearchFormData())
// 注意这个crudExpose!.getSearchFormData()，一些低版本的环境是需要添加!的
                        }
                    },
                    add: {
                        show: auth('WaterInfoModelViewSet:Create'),
                    },
                    deleteAll: {
                        show: auth('WaterInfoModelViewSet:DeleteAll'), // 假设这里有对
应的权限检查
                        text: "删除所有数据",
                        title: "删除所有涌水量信息数据",
                        click() {
                            return api.deleteAllWaterInfo().then(() => {
                                crudExpose.doRefresh(); // 删除成功后刷新列表
                            });
                        }
                    },
                }
```



```
                buttons: {
                    export:{
                        title:"导出",//鼠标停留显示的信息
                        click() : Promise<void> {
                            return exportRequest(crudExpose.getSearchFormData())
                            // return exportRequest(crudExpose!.getSearchFormData())    // 注意这个cru
                        }
                    },
                    add: {
                        show: auth( value: 'WaterInfoModelViewSet:Create'),
                    },
                    deleteAll: {
                        show: auth( value: 'WaterInfoModelViewSet:DeleteAll'), // 假设这里有对应的权限检查
                        text: "删除所有数据",
                        title: "删除所有涌水量信息数据",
                        click() {
                            return api.deleteAllWaterInfo().then(() : void  => {
                                crudExpose.doRefresh(); // 删除成功后刷新列表
                            });
                        }
                    },
```

修改导入请求的超时时间

进入service.ts中找到对应函数将时间增加到10秒。



```
 * @description 创建请求方法
 * @param {Object} service axios 实例
 */
function createRequestFunction(service: any) {  显示用法
    return function (config: any) {
        const configDefault : {headers: {"Content-Type": any...  = {
            headers: {
                'Content-Type': get(config, 'headers.Content-Type', 'application/json'),
            },
            timeout: 10000,
            baseURL: getBaseURL(),
            data: {},
        };

        // const token = userStore.getToken;
        const token = Session.get('token');
        if (token != null) {
            // @ts-ignore
            configDefault.headers.Authorization = 'JWT ' + token;
        }
        return service(Object.assign(configDefault, config));
    };
}
```