

核心功能开发

资产是资产类型的实例，资产类型中定义了该类型的应该具有哪些字段。资产类型和资产之间形成了一对多关系。

CIType功能

可以对资产类型进行增删改查，但是要注意，删除和修改对该类型已有资产实例的影响。

后台代码

user/views.py 菜单项

```
1 from django.contrib.auth.models import AbstractUser
2
3 @api_view(['GET'])
4 @permission_classes([IsAuthenticated]) # 覆盖全局配置单独设置权限
5 def menulist_view(request: Request):
6     user: AbstractUser = request.user
7     auth = request.auth
8     print(user, auth)
9     # print(dir(user)) # .is_superuser
10    menulist = {
11        'data': []
12    }
13    if user.is_superuser: # 用户管理必须管理员
14        p1 = PermissionItem(101, '用户管理')
15        p1.append(PermissionItem(102, '用户列表', '/users/'))
16        p1.append(PermissionItem(103, '角色列表', '/users/roles/'))
17        p1.append(PermissionItem(104, '权限列表', '/users/perms/'))
18        menulist['data'].append(p1)
19    # if user.has_perm('user.view_userprofile'):
20    #     pass
21
22    # cmdb
23    p5 = PermissionItem(501, '资产管理')
24    p5.append(PermissionItem(502, 'CI类型列表', '/cmdb/citypes/'))
25    p5.append(PermissionItem(503, 'CI列表', '/cmdb/cis/'))
26    menulist['data'].append(p5)
27
28    return Response(menulist)
```

cmdb/models.py

```
1 from mongoengine import (
2     Document, EmbeddedDocument,
3     StringField, IntField, BooleanField,
4     ListField, EmbeddedDocumentField
5 )
6
7 class CiTypeField(EmbeddedDocument):
8     meta = {'collection': 'citypes'}
```

```

9     name = StringField(required=True, max_length=24)
10    label = StringField(max_length=24)
11    type = StringField(max_length=24)
12    required = BooleanField(default=False)
13
14    def __str__(self):
15        return "<F {},{}>".format(self.name, self.type)
16
17    class CiType(Document):
18        meta = {'collection': 'citypes'}
19        name = StringField(required=True, unique_with='version', max_length=24)
20        label = StringField(max_length=24)
21        version = IntField(required=True, default=1)
22        fields = ListField(EmbeddedDocumentField(CiTypeField))
23
24    def __str__(self):
25        return "<CiType {}: {}, {}>".format(
26            self.name, self.version, self.fields)

```

创建cmdb/serializers.py

```

1  from rest_framework_mongoengine import serializers
2  from .models import CiType
3
4  # 使用DRFMongoEngine的序列化器类
5  class CiTypeSerializer(serializers.DocumentSerializer):
6      class Meta:
7          model = CiType
8          #fields = ('id', 'name', 'label', 'version')
9          exclude = ['fields']

```

cmdb/views.py

```

1  from rest_framework_mongoengine.viewsets import ModelViewSet
2  from .models import CiType
3  from .serializers import CiTypeSerializer
4
5  class CiTypeViewSet(ModelViewSet):
6      # CiType是ODM，因此使用DRFM的ModelViewSet
7      queryset = CiType.objects.all()
8      serializer_class = CiTypeSerializer
9      permission_classes = [] # 移除所有权限要求

```

cmdb/urls.py

```

1 from django.urls import path
2 from .views import CiTypeViewSet
3 from rest_framework_mongoengine.routers import SimpleRouter
4
5 router = SimpleRouter()
6 router.register('citytypes', CiTypeViewSet)
7
8 urlpatterns = [] + router.urls
9
10 print('~' * 30)
11 print(urlpatterns)
12 print('~' * 30)

```

前台代码

在组件目录下新建cmdb目录创建CiTypes.vue。

src/router/index.js, 省略无关代码

```

1 import CiType from '../components/cmdb/CiTypes.vue'
2
3 const routes = [
4   { path: '/', redirect: '/login' },
5   { path: '/login', component: Login },
6   {
7     path: '/home',
8     component: Home,
9     redirect: '/welcome',
10    children: [
11      { path: '/welcome', component: Welcome },
12      { path: '/users', component: User },
13      { path: '/users/perms', component: Perm },
14      { path: '/users/roles', component: Role },
15      { path: '/cmdb/citytypes', component: CiType }
16    ]
17  }
18 ]

```

新建src/components/cmdb/CiTypes.vue

```

1 <template>
2   <div>
3     <el-breadcrumb separator-class="el-icon-arrow-right">
4       <el-breadcrumb-item :to="{ path: '/home' }">首页</el-breadcrumb-item>
5       <el-breadcrumb-item>资产管理</el-breadcrumb-item>
6       <el-breadcrumb-item>资产类型</el-breadcrumb-item>
7     </el-breadcrumb>
8     <el-card class="box-card">
9       <el-table :data="dataList" border style="width: 100%">
10         <el-table-column type="index" label="序号"> </el-table-column>
11         <el-table-column prop="label" label="名称"> </el-table-column>

```

```

12     <el-table-column label="操作">
13         <el-tooltip content="分配权限" effect="light">
14             <el-button type="info" icon="el-icon-plus" size="mini"></el-
button>
15         </el-tooltip>
16         <el-tooltip content="修改" effect="light">
17             <el-button type="success" icon="el-icon-edit" size="mini"></el-
button>
18         </el-tooltip>
19         <el-tooltip content="删除" effect="light">
20             <el-button type="danger" icon="el-icon-delete" size="mini"></el-
button>
21         </el-tooltip>
22     </el-table-column>
23 </el-table>
24 <el-pagination
25     @current-change="handleCurrentChange"
26     :current-page="pagination.page"
27     :page-size="pagination.size"
28     layout="total, prev, pager, next, jumper"
29     :total="pagination.total"
30 >
31 </el-pagination>
32 </el-card>
33 </div>
34 </template>
35
36 <script>
37 export default {
38     created() {
39         this.getList()
40     },
41     data() {
42         return {
43             dataList: [],
44             pagination: { page: 1, size: 1, total: 0 }
45         }
46     },
47     methods: {
48         handleCurrentChange(val) {
49             console.log(`当前页: ${val}`)
50             this.getList(val)
51         },
52         async getList(page = 1) {
53             if (!page) {
54                 page = 1
55             }
56             const { data: response } = await this.$http.get('cndb/citytypes/', {
57                 params: {
58                     page
59                 }
60             })
61             if (response.code) {
62                 return this.$message.error(response.message)
63             }
64             this.dataList = response.results
65             this.pagination = response.pagination
66         }

```

```

67     }
68 }
69 </script>
70
71 <style>
72 </style>

```

增删改功能，请参照之前代码，自行补全。

CI功能

对资产的增删改查，资产属于某一个类型。

后端代码

由于资产类别有很多种，每一种的字段也都不一样，产生的资产实例字段完全不一样。

需要为每一种类别建一个Collection吗？

不需要。将所有资产都放到一个Collection中，命名为cis。cis的字段完全不固定，怎么设计模型类呢？

<http://docs.mongoengine.org/guide/defining-documents.html#dynamic-document-schemas>

cmdb/models.py

```

1  from mongoengine import (
2      Document, EmbeddedDocument, DynamicDocument,
3      StringField, IntField, BooleanField,
4      ListField, EmbeddedDocumentField
5  )
6
7  class CiTypeField(EmbeddedDocument):
8      meta = {'collection': 'citypes'}
9      name = StringField(required=True, max_length=24)
10     label = StringField(max_length=24)
11     type = StringField(max_length=24)
12     required = BooleanField(default=False)
13
14     def __str__(self):
15         return "<F {},{}>".format(self.name, self.type)
16
17     class CiType(Document):
18         meta = {'collection': 'citypes'}
19         name = StringField(required=True, unique_with='version', max_length=24)
20         label = StringField(max_length=24)
21         version = IntField(required=True, default=1)
22         fields = ListField(EmbeddedDocumentField(CiTypeField))
23
24         def __str__(self):
25             return "<CiType {}: {}, {}>".format(
26                 self.name, self.version, self.fields)
27
28     class Ci(DynamicDocument):
29         # 字段取决于CiType中某一种类型在fields中定义的字段，可以说几乎每种类型的字段都不一样
30         meta = {'collection': 'cis'}

```

cmdb/serializers.py

```
1 from rest_framework_mongoengine import serializers
2 from .models import CiType, Ci
3
4 # 使用DRFMongoEngine的序列化器类
5 class CiTypeSerializer(serializers.DocumentSerializer):
6     class Meta:
7         model = CiType
8         #fields = ('id', 'name', 'label', 'version')
9         exclude = ['fields']
10
11 class CiSerializer(serializers.DynamicDocumentSerializer):
12     class Meta:
13         model = Ci
14         fields = '__all__'
```

cmdb/views.py

增加一个不分页的返回全部CI分类的功能

```
1 from rest_framework.response import Response
2 from rest_framework_mongoengine.viewsets import ModelViewSet
3 from rest_framework.decorators import action
4 from .models import CiType, Ci
5 from .serializers import CiTypeSerializer, CiSerializer
6
7 class CiTypeViewSet(ModelViewSet):
8     # CiType是ODM，因此使用DRFM的ModelViewSet
9     queryset = CiType.objects.all()
10    serializer_class = CiTypeSerializer
11    permission_classes = [] # 移除所有权限要求
12
13    @action(detail=False) # 默认GET
14    def all(self, request):
15        """返回所有分类。相当于重写list，去掉分页"""
16        queryset = self.filter_queryset(self.get_queryset())
17        serializer = self.get_serializer(queryset, many=True)
18        data = serializer.data
19        return Response(data)
20
21 class CiViewSet(ModelViewSet):
22     queryset = Ci.objects
23     serializer_class = CiSerializer
24     permission_classes = [] # 移除所有权限要求
```

cmdb/urls.py

```
1 from .views import CiTypeViewSet, CiViewSet
2 from rest_framework_mongoengine.routers import SimpleRouter
3
4 router = SimpleRouter()
5 router.register('citytypes', CiTypeViewSet)
6 router.register('cis', CiViewSet)
7
8 urlpatterns = [] + router.urls
```

动态改变字段

需求：列表页和详情页看到的字段不一样。

解决方案：

1. 不用视图集，定义列表页、详情页2个视图类，各自指定不同的序列化器
2. 使用视图集，内部区分列表页、详情页，来使用不同的序列化器

cmdb/serializers.py

```
1 from rest_framework_mongoengine import serializers
2 from .models import CiType, Ci
3
4 # 使用DRFMongoEngine的序列化器类
5 class CiTypeSerializer(serializers.DocumentSerializer):
6     class Meta:
7         model = CiType
8         #fields = ('id', 'name', 'label', 'version')
9         exclude = ['fields']
10
11 class CiTypeFieldsSerializer(serializers.DocumentSerializer):
12     class Meta:
13         model = CiType
14         fields = ('id', 'fields')
15
16 class CiSerializer(serializers.DynamicDocumentSerializer):
17     class Meta:
18         model = Ci
19         fields = '__all__'
```

访问某一个CiType的时候，也就是详情页，需要返回所有字段定义，这和列表页不一样。

如何区分列表页、详情页？用是否有pk或id参数。

如何更换序列化器？这就要覆盖get_serializer_class了。

```
1 from rest_framework.response import Response
2 from rest_framework_mongoengine.viewsets import ModelViewSet
3 from rest_framework.decorators import action
4 from .models import CiType, Ci
5 from .serializers import CiTypeSerializer, CiSerializer,
6 CiTypeFieldsSerializer
7
8 class CiTypeViewSet(ModelViewSet):
9     # CiType是ODM，因此使用DRFM的ModelViewSet
10     queryset = CiType.objects.all()
11     serializer_class = CiTypeSerializer
12     permission_classes = [] # 移除所有权限要求
13
14     @action(detail=False) # 默认GET
15     def all(self, request):
16         """返回所有分类。相当于重写list，去掉分页"""
17         queryset = self.filter_queryset(self.get_queryset())
18         serializer = self.get_serializer(queryset, many=True)
19         data = serializer.data
20         return Response(data)
```

```

21     def get_serializer_class(self):
22         if 'id' in self.kwargs: # 有id就是详情页替换序列化器
23             return CiTypeFieldsSerializer
24         return super().get_serializer_class()
25
26
27 class CiViewSet(ModelViewSet):
28     queryset = Ci.objects
29     serializer_class = CiSerializer
30     permission_classes = [] # 移除所有权限要求

```

前端代码

新建Cis.vue组件文件。

src/router/index.js 路由配置

```

1  import CiType from '../components/cmdb/CiTypes.vue'
2  import Ci from '../components/cmdb/Cis.vue'
3
4  const routes = [
5    { path: '/', redirect: '/login' },
6    { path: '/login', component: Login },
7    {
8      path: '/home',
9      component: Home,
10     redirect: '/welcome',
11     children: [
12       { path: '/welcome', component: Welcome },
13       { path: '/users', component: User },
14       { path: '/users/perms', component: Perm },
15       { path: '/users/roles', component: Role },
16       { path: '/cmdb/citytypes', component: CiType },
17       { path: '/cmdb/cis', component: Ci }
18     ]
19   }
20 ]

```

src/plugins/element.js

```

1  import Vue from 'vue'
2  import {
3    // 省略重复代码
4    Select,
5    Option,
6    DatePicker
7  } from 'element-ui'
8
9  Vue.use(Select)
10 Vue.use(Option)
11 Vue.use(DatePicker)

```

Select选择框、DatePicker日期


```

1  <template>
2    <div>
3      <el-breadcrumb separator-class="el-icon-arrow-right">
4        <el-breadcrumb-item :to="{ path: '/home' }">首页</el-breadcrumb-item>
5        <el-breadcrumb-item>资产管理</el-breadcrumb-item>
6        <el-breadcrumb-item>资产列表</el-breadcrumb-item>
7      </el-breadcrumb>
8      <el-card class="box-card">
9        <el-row :gutter="20">
10         <el-col :span="12">
11           <el-button type="primary" @click="addDialogVisible = true">增加资
12           产</el-button>
13         </el-col>
14       </el-row>
15       <el-table :data="dataList" border style="width: 100%">
16         <el-table-column type="index" label="序号"> </el-table-column>
17         <el-table-column prop="label" label="名称"> </el-table-column>
18         <el-table-column label="操作">
19           <el-tooltip content="分配权限" effect="light">
20             <el-button type="info" icon="el-icon-plus" size="mini"></el-
21             button>
22           </el-tooltip>
23           <el-tooltip content="修改" effect="light">
24             <el-button type="success" icon="el-icon-edit" size="mini"></el-
25             button>
26           </el-tooltip>
27           <el-tooltip content="删除" effect="light">
28             <el-button type="danger" icon="el-icon-delete" size="mini">
29           </el-button>
30           </el-tooltip>
31         </el-table-column>
32       </el-table>
33       <el-pagination
34         @current-change="handleCurrentChange"
35         :current-page="pagination.page"
36         :page-size="pagination.size"
37         layout="total, prev, pager, next, jumper"
38         :total="pagination.total"
39       >
40     </el-pagination>
41   </el-card>
42   <!-- 添加对话框 -->
43   <el-dialog title="增加" :visible.sync="addDialogVisible"
44     @close="resetForm('add')">
45     <el-form :model="addForm" :rules="addRules" ref="add" label-
46     width="100px">
47       <el-form-item label="名称">
48         <el-select v-model="addForm.citype" placeholder="请选择"
49           @change="handleCiTypeChange">
50           <el-option v-for="item in citypes" :key="item.id"
51             :label="item.label" :value="item.id"> </el-option>
52         </el-select>
53       </el-form-item>
54     </el-form>
55     <div slot="footer" class="dialog-footer">

```

```

48     <el-button @click="addDialogVisible = false">取 消</el-button>
49     <el-button type="primary" @click="add">确 定</el-button>
50   </div>
51 </el-dialog>
52 </div>
53 </template>
54
55 <script>
56 export default {
57   created() {
58     this.getList()
59     this.getAllTypeList()
60   },
61   data() {
62     return {
63       dataList: [],
64       pagination: { page: 1, size: 1, total: 0 },
65       citytypes: [],
66       //
67       addDialogVisible: false,
68       addForm: {
69         citytype: null
70         // name: ''
71       },
72       addRules: {
73         // name: [
74         //   { required: true, message: '请输入名称', trigger: 'blur' },
75         //   { min: 1, max: 16, message: '长度在 1 到 16 个字符', trigger:
76         'blur' }
77         // ]
78       }
79     },
80     methods: {
81       handleCurrentChange(val) {
82         console.log(`当前页: ${val}`)
83         this.getList(val)
84       },
85       resetForm(name) {
86         this.$refs[name].resetFields()
87       },
88       async getList(page = 1) {
89         if (!page) {
90           page = 1
91         }
92         const { data: response } = await this.$http.get('cmdb/cis/', {
93           params: {
94             page
95           }
96         })
97         if (response.code) {
98           return this.$message.error(response.message)
99         }
100         this.dataList = response.results
101         this.pagination = response.pagination
102       },
103       async getAllTypeList(page = 1) {
104         if (!page) {

```

```

105     page = 1
106   }
107   const { data: response } = await this.$http.get('cndb/citytypes/all/')
108   if (response.code) {
109     return this.$message.error(response.message)
110   }
111   this.citytypes = response
112 },
113 async handleCiTypeChange() {
114   console.log(this.addForm)
115   // 去请求类型的字段
116   const { citytype: id } = this.addForm
117   const { data: response } = await
118   this.$http.get(`cndb/citytypes/${id}/`)
119   if (response.code) {
120     return this.$message.error(response.message)
121   }
122   console.log(response) // 返回所有定义的字段
123 },
124 add() {}
125 }
126 </script>
127
128 <style>
129 </style>

```

动态增加表单项**

参考 <https://element.eleme.cn/#/zh-CN/component/form#dong-tai-zeng-jian-biao-dan-xiang>

由于Ci Type定义不同的类型，每个类型对应的字段不一样，所以，表单项也要变化，不能提前构建。

返回类型的fields数组后，遍历它。el-form-item组件中使用v-for指令遍历创建表单项。

```

<el-form :model="addForm" :rules="addRules" ref="add" label-width="100px">
  <el-form-item label="名称" prop="citytype">
    <el-select v-model="addForm.citytype" placeholder="请选择" @change="handleCiTypeChange">
      <el-option v-for="item in citytypes" :key="item.id" :label="item.label" :value="item.id"> </el-option>
    </el-select>
  </el-form-item>
  <!-- 动态表单项 -->
  <el-form-item
    v-for="(domain, index) in addForm.fields"
    :label="domain.label"
    :key="domain.name"
    :prop="'fields.' + index + '.value'"
    :rules="
      domain.required
      ? {
        required: true,
        message: domain.label + '不能为空',
        trigger: 'blur'
      }
      : {}
    "
  >
    <el-input v-model="domain.value"></el-input>
  </el-form-item>
</el-form>

```

```

data() {
  return {
    addForm: {
      citytype: null,
      fields: []
    },
    addRules: {
      citytype: [{ required: true, message: '请选择资产类型', trigger: 'blur' }]
    }
  },
  methods: {
    async handleCiTypeChange() {
      const { citytype: id } = this.addForm
      const { data: response } = await this.$http.get(`cndb/citytypes/${id}/`)
      if (response.code) {
        return this.$message.error(response.message)
      }
      this.addForm.fields = response.fields
      // {id, fields:[{name, label, type, required}]}
    }
  }
}

```

:prop="'fields.' + index + '.value'"，这里是v-bind，说明写的是表达式。表达式里面的值不可以随便给，就是给下面的:rules规则安个名字，但要和addForm中字段值fields对应。之所以用index，是因为数组用 .index 访问第几个。

反应式对象

确切地讲，在Vue中，被观察的Reactive（响应式，反应式）对象，发生改变会引起视图更新。

```
1 <script>
2 export default {
3   created() {
4     this.getList()
5     this.getAllTypeList()
6   },
7   data() {
8     return {
9       addForm: {
10         citytype: null
11         // fields: [] // 注释这一句看看
12       },
13       addRules: {
14         citytype: [{ required: true, message: '请选择资产类型', trigger: 'blur'
15       }]
16     }
17   },
18   methods: {
19     async handleCiTypeChange() {
20       const { citytype: id } = this.addForm
21       const { data: response } = await this.$http.get(`cmdb/citytypes/${id}/`)
22       if (response.code) {
23         return this.$message.error(response.message)
24       }
25       this.addForm.fields = response.fields
26       console.log(this.addForm) // 注释fields，观察注释前后并对比
27       // {id, fields:[{name, label, type, required}]}
28     }
29   }
30 }
31 }
32 </script>
```

注意观察fields是否后面是3点表示

未注释	注释
<pre>▼ {__ob__: Observer} ⓘ citytype: (...) fields: (...)</pre>	<pre>▼ {fields: Array(5), __ob__: Observer} ⓘ citytype: (...) ▼ fields: Array(5) ► 0: {name: 'name', label: '名称', type: 'str', required: true} ► 1: {name: 'IP Address', label: 'IP', type: 'str', required: true} ► 2: {name: 'Mac Address', label: 'MAC', type: 'str', required: true} ► 3: {name: 'GateWay', label: '网关', type: 'str', required: false} ► 4: {name: 'Mask', label: '掩码', type: 'str', required: false} length: 5</pre>

如果是3点表示，它就被观察的Reactive对象，它的改变会引起Vue的视图更新。

把addForm中的fields注释掉，动态为这对象增加fields属性，这个属性就不是Reactive对象。这需要Vue.set全局函数，或者使用其在实例上的别名this.\$set。

```
1 // this.addForm.fields = response.fields
2 this.$set(this.addForm, 'fields', response.fields) // 增加Reactive对象
```

列表类型字段处理***

网络接口的type为 `list:Network Interface`，怎么处理？

冒号后面写的是类型的名称，名称是唯一键，查询也可以，未来还要生成多版本。类型新版信息名称不变但是会生成同名新版本记录，新的id。

修改类型测试代码，重新生成一次数据

```
1 CiTypeField(name='Network Interfaces', label='网络接口',
2             type='list:{}:{}'.format(ctni.name, ctni.version))
```

遇到这种问题的时候，必须知道 `Network Interface:1`，对应的字段是谁？什么时候去查询字段？。

采用懒查询策略，也就是用户不增加，这里不去查询 `Network Interface:1` 对应的字段。

通过 名称 和 版本 请求后台，如下

```
1 this.$http.get(`/cmdb/citytypes/${name}/${version}/`)
```

后台Django视图类中要增加对应的action装饰的handler。

cmdb/view.py

增加action `get_by_name_and_version`

```
1 from rest_framework.response import Response
2 from rest_framework_mongoengine.viewsets import ModelViewSet
3 from rest_framework.decorators import action
4 from .models import CiType, Ci
5 from .serializers import CiTypeSerializer, CiSerializer,
6   CiTypeFieldsSerializer
7
8 class CiTypeViewSet(ModelViewSet):
9     # CiType是ODM，因此使用DRFM的ModelViewSet
10    queryset = CiType.objects.all()
11    serializer_class = CiTypeSerializer
12    permission_classes = [] # 移除所有权限要求
13
14    @action(detail=False) # 默认GET
15    def all(self, request):
16        """返回所有分类。相当于重写list，去掉分页"""
17        queryset = self.filter_queryset(self.get_queryset())
18        serializer = self.get_serializer(queryset, many=True)
19        data = serializer.data
20        return Response(data)
21
22    @action(detail=False, url_path='(?P<name>[^\./]+)/(?P<version>\d+)')
23    # /cmdb/citytypes/Network Interface/1/
24    #url_path不支持最新3.x中path的<int:version>语法，只能使用正则。但是version拿到
    的是str类型
25    def get_by_name_and_version(self, request, name, version):
```

```

25         print(name, version, "+++")
26         obj = self.get_queryset().get(name=name, version=version) # 名字加版本
必须唯一
27         serializer = CiTypeFieldsSerializer(obj)
28         data = serializer.data
29         return Response(data)
30
31     def get_serializer_class(self):
32         if 'id' in self.kwargs: # 有id就是详情页替换序列化器
33             return CiTypeFieldsSerializer
34         return super().get_serializer_class()
35
36
37 class CiViewSet(ModelViewSet):
38     queryset = Ci.objects
39     serializer_class = CiSerializer
40     permission_classes = [] # 移除所有权限要求

```

src/components/cmdm/Cis.vue

遇到list类型，那么还可以动态增加表单项

```

1 <template>
2   <div>
3     <el-breadcrumb separator-class="el-icon-arrow-right">
4       <el-breadcrumb-item :to="{ path: '/home' }">首页</el-breadcrumb-item>
5       <el-breadcrumb-item>资产管理</el-breadcrumb-item>
6       <el-breadcrumb-item>资产列表</el-breadcrumb-item>
7     </el-breadcrumb>
8     <el-card class="box-card">
9       <el-row :gutter="20">
10        <el-col :span="12">
11          <el-button type="primary" @click="addDialogVisible = true">增加资
产</el-button>
12        </el-col>
13      </el-row>
14      <el-table :data="dataList" border style="width: 100%">
15        <el-table-column type="index" label="序号"> </el-table-column>
16        <el-table-column prop="label" label="名称"> </el-table-column>
17        <el-table-column label="操作">
18          <el-tooltip content="分配权限" effect="light">
19            <el-button type="info" icon="el-icon-plus" size="mini"></el-
button>
20          </el-tooltip>
21          <el-tooltip content="修改" effect="light">
22            <el-button type="success" icon="el-icon-edit" size="mini"></el-
button>
23          </el-tooltip>
24          <el-tooltip content="删除" effect="light">
25            <el-button type="danger" icon="el-icon-delete" size="mini">
</el-button>
26          </el-tooltip>
27        </el-table-column>
28      </el-table>
29      <el-pagination
30        @current-change="handleCurrentChange"

```

```

31         :current-page="pagination.page"
32         :page-size="pagination.size"
33         layout="total, prev, pager, next, jumper"
34         :total="pagination.total"
35     >
36     </el-pagination>
37 </el-card>
38 <!-- 添加对话框 -->
39 <el-dialog title="增加" :visible.sync="addDialogVisible"
@close="resetForm('add')" width="90%">
40     <el-form :model="addForm" :rules="addRules" ref="add" label-
width="100px">
41         <el-form-item label="名称" prop="citytype">
42             <el-select v-model="addForm.citytype" placeholder="请选择"
@change="handleCityChange">
43                 <el-option v-for="item in citytypes" :key="item.id"
:label="item.label" :value="item.id"> </el-option>
44             </el-select>
45         </el-form-item>
46         <!-- 动态表单项 -->
47         <el-form-item
v-for="(domain, index) in addForm.fields"
:label="domain.label"
:key="domain.name"
:prop="'fields.' + index + '.value'"
:rules="
53             domain.required
54             ? {
55                 required: true,
56                 message: domain.label + '不能为空',
57                 trigger: 'blur'
58             }
59             : {}
60         "
61     >
62         <div v-if="!domain.type.startsWith('list:')"><el-input v-
model="domain.value"></el-input></div>
63         <div v-else>
64             <el-button icon="el-icon-plus" size="mini" plain type="success"
@click="handleAddChild(domain)"></el-button>
65             <el-card v-for="(def, i) in domain.fieldsDef"
:key="`${domain.name}.def.${i}`">
66                 <!-- <div>{{ def }}</div> -->
67                 <!-- 动态子表单项 -->
68                 <el-form-item
v-for="(subDomain, j) in def.fields"
:label="subDomain.label"
:key="subDomain.name"
:prop="'fields.' + index + '.fieldsDef.' + i + '.fields.' +
j + '.value'"
73                 :rules="
74                     subDomain.required
75                     ? {
76                         required: true,
77                         message: subDomain.label + '不能为空',
78                         trigger: 'blur'
79                     }
80                     : {}

```

```

81         "
82     >
83     <el-input v-model="subDomain.value"></el-input>
84 </el-form-item>
85 </el-card>
86 </div>
87 </el-form-item>
88 </el-form>
89 <div slot="footer" class="dialog-footer">
90     <el-button @click="addDialogVisible = false">取 消</el-button>
91     <el-button type="primary" @click="add">确 定</el-button>
92 </div>
93 </el-dialog>
94 </div>
95 </template>
96
97 <script>
98 export default {
99     created() {
100         this.getList()
101         this.getAllTypeList()
102     },
103     data() {
104         return {
105             dataList: [],
106             pagination: { page: 1, size: 1, total: 0 },
107             citypes: [],
108             // 增加资产对话框
109             addDialogVisible: false,
110             addForm: {
111                 citype: null
112                 // fields: [] // 注释这一句看看
113             },
114             addRules: {
115                 citype: [{ required: true, message: '请选择资产类型', trigger: 'blur'
116             }
117         }
118     },
119     methods: {
120         handleCurrentChange(val) {
121             console.log(`当前页: ${val}`)
122             this.getList(val)
123         },
124         resetForm(name) {
125             console.log(this.$refs[name])
126             this.$refs[name].resetFields()
127         },
128         async getList(page = 1) {
129             if (!page) {
130                 page = 1
131             }
132             const { data: response } = await this.$http.get('cmdb/cis/', {
133                 params: {
134                     page
135                 }
136             })
137             if (response.code) {

```



```

138         return this.$message.error(response.message)
139     }
140     this.dataList = response.results
141     this.pagination = response.pagination
142 },
143 async getAllTypeList(page = 1) {
144     if (!page) {
145         page = 1
146     }
147     const { data: response } = await this.$http.get('cndb/citytypes/all/')
148     if (response.code) {
149         return this.$message.error(response.message)
150     }
151     this.citytypes = response
152 },
153 async handleCityTypeChange() {
154     const { citytype: id } = this.addForm
155     const { data: response } = await
this.$http.get(`cndb/citytypes/${id}/`)
156     if (response.code) {
157         return this.$message.error(response.message)
158     }
159     // this.addForm.fields = response.fields
160     this.$set(this.addForm, 'fields', response.fields) // 增加Reactive对象
161     console.log(this.addForm) // 注释fields, 观察注释前后并对比
162     // {id, fields:[{name, label, type, required}]}
163 },
164 async handleAddChild(domain) {
165     console.log(domain)
166     console.log(domain.type.split(':'))
167     const [, name, version] = domain.type.split(':')
168     const { data: response } = await
this.$http.get(`cndb/citytypes/${name}/${version}/`)
169     if (response.code) {
170         return this.$message.error(response.error)
171     }
172     if ('fieldsDef' in domain === false) {
173         // domain.fieldsDef = [] // 这种新增方式无法触发视图更新
174         this.$set(domain, 'fieldsDef', [])
175     }
176     domain.fieldsDef.push(response)
177     console.log(this.addForm)
178 },
179 add() {
180     console.log(this.addForm)
181     console.log(this.addRules)
182     const name = 'add'
183     this.$refs[name].validate(async (valid) => {
184         if (valid) {
185             // const { data: response } = await
this.$http.patch(`users/mgr/${id}/`, this.editUserForm)
186             // if (response.code) {
187             //     return this.$message.error(response.message)
188             // }
189             // this.addDialogVisible = false
190             this.resetForm(name)
191             // this.getUserList(this.pagination.page) // 刷新用户列表
192         }

```

```

193     })
194   }
195 }
196 }
197 </script>
198
199 <style lang="less" scoped>
200 .el-form-item .el-card .el-form-item {
201   margin-bottom: 22px; /*解决card内form item没有下方空白*/
202 }
203 </style>

```

特别注意 :prop="'fields.' + index + '.fieldsDef.' + i + '.fields.' + j + '.value'",
只有这样才能对应addForm中的字段

网络接口

The diagram illustrates the mapping between the form fields and the data structure. The form fields are: * 名称 (Name), * IP, * MAC, 网关 (Gateway), and 掩码 (Mask). The data structure is a nested object with 'fieldsDef' and 'fields' arrays. The 'fieldsDef' array contains two objects, each with a 'fields' array. The 'fields' array contains five objects, each with a 'label', 'name', 'required', and 'type' property. The 'fields' array also contains five objects, each with a 'label', 'name', 'required', and 'type' property. The 'fields' array also contains five objects, each with a 'label', 'name', 'required', and 'type' property. The 'fields' array also contains five objects, each with a 'label', 'name', 'required', and 'type' property.

可以看出，表单再复杂，也不过如此了，只要耐心，可以完成的。

增加CI

利用CiViewSet的POST方法，接收提交的数据。

src/components/cmdb/Cis.vue

```

1 <template>
2   <div>
3     <el-breadcrumb separator-class="el-icon-arrow-right">
4       <el-breadcrumb-item :to="{ path: '/home' }">首页</el-breadcrumb-item>
5       <el-breadcrumb-item>资产管理</el-breadcrumb-item>
6       <el-breadcrumb-item>资产列表</el-breadcrumb-item>
7     </el-breadcrumb>
8     <el-card class="box-card">
9       <el-row :gutter="20">
10        <el-col :span="12">
11          <el-button type="primary" @click="addDialogvisible = true">增加资
12          产</el-button>

```

```

13     </el-row>
14     <el-table :data="dataList" border style="width: 100%">
15       <el-table-column type="index" label="序号"> </el-table-column>
16       <el-table-column label="名称">
17         <template #default="{ row }">
18           {{ row.fields[0].value }}
19         </template>
20       </el-table-column>
21       <el-table-column label="操作">
22         <el-tooltip content="修改" effect="light">
23           <el-button type="success" icon="el-icon-edit" size="mini"></el-
button>
24           </el-tooltip>
25           <el-tooltip content="删除" effect="light">
26             <el-button type="danger" icon="el-icon-delete" size="mini">
</el-button>
27           </el-tooltip>
28         </el-table-column>
29       </el-table>
30       <el-pagination
31         @current-change="handleCurrentChange"
32         :current-page="pagination.page"
33         :page-size="pagination.size"
34         layout="total, prev, pager, next, jumper"
35         :total="pagination.total"
36       >
37     </el-pagination>
38   </el-card>
39   <!-- 添加对话框 -->
40   <el-dialog title="增加" :visible.sync="addDialogVisible"
@close="resetForm('add')" width="90%">
41     <el-form :model="addForm" :rules="addRules" ref="add" label-
width="100px">
42       <el-form-item label="名称" prop="citytype">
43         <el-select v-model="addForm.citytype" placeholder="请选择"
@change="handleCityTypeChange">
44           <el-option v-for="item in citytypes" :key="item.id"
:label="item.label" :value="item.id"> </el-option>
45         </el-select>
46       </el-form-item>
47       <!-- 动态表单项 -->
48       <el-form-item
49         v-for="(domain, index) in addForm.fields"
50         :label="domain.label"
51         :key="domain.name"
52         :prop="'fields.' + index + '.value'"
53         :rules="
54           domain.required
55           ? {
56             required: true,
57             message: domain.label + '不能为空',
58             trigger: 'blur'
59           }
60           : {}
61       >
62       <div v-if="!domain.type.startsWith('list:')"><el-input v-
model="domain.value"></el-input></div>

```

```

64         <div v-else>
65             <el-button icon="el-icon-plus" size="mini" plain type="success"
@click="handleAddChild(domain)"></el-button>
66             <el-card v-for="(def, i) in domain.fieldsDef"
:key="`${domain.name}.def.${i}`">
67                 <!-- <div>{{ def }}</div> -->
68                 <!-- 动态子表单项 -->
69                 <el-form-item
70                     v-for="(subDomain, j) in def.fields"
71                     :label="subDomain.label"
72                     :key="subDomain.name"
73                     :prop="'fields.' + index + '.fieldsDef.' + i + '.fields.' +
j + '.value'"
74                     :rules="
75                         subDomain.required
76                         ? {
77                             required: true,
78                             message: subDomain.label + '不能为空',
79                             trigger: 'blur'
80                         }
81                         : {}
82                     "
83                 >
84                     <el-input v-model="subDomain.value"></el-input>
85                 </el-form-item>
86             </el-card>
87         </div>
88     </el-form-item>
89 </el-form>
90 <div slot="footer" class="dialog-footer">
91     <el-button @click="addDialogVisible = false">取 消</el-button>
92     <el-button type="primary" @click="add">确 定</el-button>
93 </div>
94 </el-dialog>
95 </div>
96 </template>
97
98 <script>
99 export default {
100     created() {
101         this.getList()
102         this.getAllTypeList()
103     },
104     data() {
105         return {
106             dataList: [],
107             pagination: { page: 1, size: 1, total: 0 },
108             citypes: [],
109             // 增加资产对话框
110             addDialogVisible: false,
111             addForm: {
112                 citype: null
113                 // fields: [] // 注释这一句看看
114             },
115             addRules: {
116                 citype: [{ required: true, message: '请选择资产类型', trigger: 'blur'
117             }
118         }

```

```

118     }
119   },
120   methods: {
121     handleCurrentChange(val) {
122       console.log(`当前页: ${val}`)
123       this.getList(val)
124     },
125     resetForm(name) {
126       console.log(this.$refs[name])
127       this.$refs[name].resetFields()
128     },
129     async getList(page = 1) {
130       if (!page) {
131         page = 1
132       }
133       const { data: response } = await this.$http.get('cmdb/cis/', {
134         params: {
135           page
136         }
137       })
138       if (response.code) {
139         return this.$message.error(response.message)
140       }
141       this.dataList = response.results
142       this.pagination = response.pagination
143     },
144     async getAllTypeList(page = 1) {
145       if (!page) {
146         page = 1
147       }
148       const { data: response } = await this.$http.get('cmdb/citytypes/all/')
149       if (response.code) {
150         return this.$message.error(response.message)
151       }
152       this.citytypes = response
153     },
154     async handleCiTypeChange() {
155       const { citytype: id } = this.addForm
156       const { data: response } = await
this.$http.get(`cmdb/citytypes/${id}/`)
157       if (response.code) {
158         return this.$message.error(response.message)
159       }
160       // this.addForm.fields = response.fields
161       this.$set(this.addForm, 'fields', response.fields) // 增加Reactive对象
162       console.log(this.addForm) // 注释fields, 观察注释前后并对比
163       // {id, fields:[{name, label, type, required}]}
164     },
165     async handleAddChild(domain) {
166       console.log(domain)
167       console.log(domain.type.split(':'))
168       const [, name, version] = domain.type.split(':')
169       const { data: response } = await
this.$http.get(`cmdb/citytypes/${name}/${version}/`)
170       if (response.code) {
171         return this.$message.error(response.error)
172       }
173       if ('fieldsDef' in domain === false) {

```

```

174     // domain.fieldsDef = [] // 这种新增方式无法触发视图更新
175     this.$set(domain, 'fieldsDef', [])
176   }
177   domain.fieldsDef.push(response)
178   console.log(this.addForm)
179 },
180 add() {
181   // console.log(this.addForm)
182   const name = 'add'
183   this.$refs[name].validate(async (valid) => {
184     if (valid) {
185       const { data: response } = await this.$http.post('cndb/cis/',
this.addForm)
186       if (response.code) {
187         return this.$message.error(response.message)
188       }
189       this.addDialogVisible = false
190       this.resetForm(name)
191       this.getList() // 刷新列表
192     }
193   })
194 }
195 }
196 }
197 </script>
198
199 <style lang="less" scoped>
200 .el-form-item .el-card .el-form-item {
201   margin-bottom: 22px; /*解决card内form item没有下方空白*/
202 }
203 </style>

```

至此，核心功能基本开发完成。望同学们能抛砖引玉，解决生产需求。