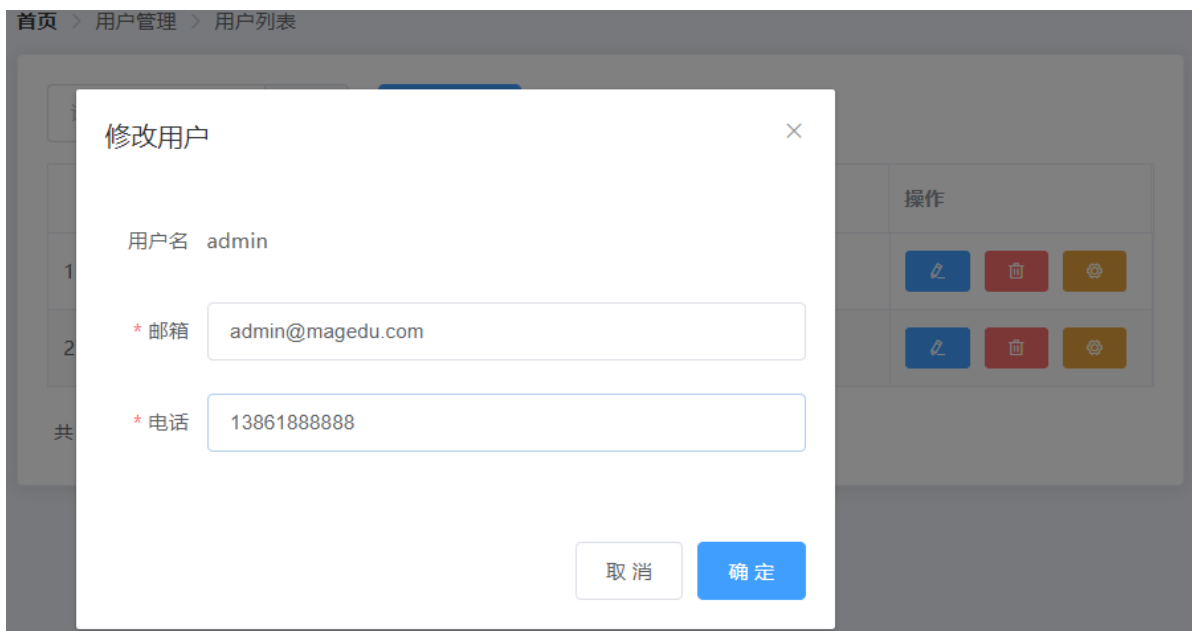


用户功能实现

功能有：

- 用户列表、分页
- 添加用户、用户修改、用户删除
- 激活、禁用用户
- 用户搜索
- 角色配置





组件和路由

新建components/user/Users.vue

```
1 <template>
2   <div>用户列表页</div>
3 </template>
4
5 <script>
6   export default {}
7 </script>
8
9 <style>
10  </style>
```

路由配置 (省略未变化代码)

```
1 import User from '../components/user/Users.vue'
2
3 const routes = [
```

```

4   { path: '/', redirect: '/login' },
5   { path: '/login', component: Login },
6   {
7     path: '/home',
8     component: Home,
9     redirect: '/welcome', // 访问/home重定向到/welcome, 进入子路由
10    children: [
11      { path: '/welcome', component: Welcome },
12      { path: '/users', component: User }
13    ]
14  }
15 ]

```

UI组件

参考 <https://element.eleme.cn/#/zh-CN/component/breadcrumb>

简单卡片 <https://element.eleme.cn/#/zh-CN/component/card#jian-dan-qia-pian>

搜索框 <https://element.eleme.cn/#/zh-CN/component/input#fu-he-xing-shu-ru-kuang>

表格 <https://element.eleme.cn/#/zh-CN/component/table>

src/plugins/element.js

```

1  import Vue from 'vue'
2  import {
3    Form, FormItem, Input, Button, Message, Container,
4    Header, Aside, Main, Menu, MenuItem, Submenu, Breadcrumb,
5    BreadcrumbItem, Card, Row, Col, Table, TableColumn
6  } from 'element-ui'
7  import 'element-ui/lib/theme-chalk/index.css' // UI组件样式
8
9  Vue.use(Breadcrumb)
10 Vue.use(BreadcrumbItem)
11 Vue.use(Card)
12 Vue.use(Row)
13 Vue.use(Col)
14 Vue.use(Table)
15 Vue.use(TableColumn)

```

src/assets/css/main.css全局样式中加入样式，控制所有面包屑、表格

```

1  /* 全局样式表 */
2  html, body, #app {
3    height: 100%;
4    margin: 0;
5    padding: 0;
6    min-height: 200px;
7  }
8  .el-breadcrumb {
9    margin-bottom: 15px;
10 }
11
12 .el-table {
13   margin: 15px 0px;
14 }

```

```
1 <template>
2   <div>
3     <el-breadcrumb separator="/">
4       <el-breadcrumb-item :to="{ path: '/home' }">首页</el-breadcrumb-item>
5       <el-breadcrumb-item>用户管理</el-breadcrumb-item>
6       <el-breadcrumb-item>用户列表</el-breadcrumb-item>
7     </el-breadcrumb>
8     <el-card class="box-card">
9       <el-row :gutter="20">
10        <el-col :span="12">
11          <el-input placeholder="请输入内容" v-model="input3">
12            <el-button slot="append" icon="el-icon-search"></el-button>
13          </el-input>
14        </el-col>
15        <el-col :span="12">
16          <el-button type="primary">增加用户</el-button>
17        </el-col>
18      </el-row>
19      <el-table :data="tableData" border style="width: 100%">
20        <el-table-column prop="date" label="日期" width="180"> </el-table-
column>
21        <el-table-column prop="name" label="姓名" width="180"> </el-table-
column>
22        <el-table-column prop="address" label="地址"> </el-table-column>
23      </el-table>
24    </el-card>
25  </div>
26 </template>
27
28 <script>
29 export default {
30   data() {
31     return {
32       tableData: [
33         {
34           date: '2016-05-02',
35           name: '王小虎',
36           address: '上海市普陀区金沙江路 1518 弄'
37         },
38         {
39           date: '2016-05-04',
40           name: '王小虎',
41           address: '上海市普陀区金沙江路 1517 弄'
42         },
43         {
44           date: '2016-05-01',
45           name: '王小虎',
46           address: '上海市普陀区金沙江路 1519 弄'
47         }
48       ]
49     }
50   }
51 }
```

```
52 </script>
53
54 <style lang="less" scoped>
55 </style>
```

重建User模型类

User模型类是Django内建的，对应auth_user表

```
1 CREATE TABLE `auth_user` (
2   `id` int(11) NOT NULL AUTO_INCREMENT,
3   `password` varchar(128) NOT NULL,
4   `last_login` datetime(6) DEFAULT NULL,
5   `is_superuser` tinyint(1) NOT NULL,
6   `username` varchar(150) NOT NULL,
7   `first_name` varchar(150) NOT NULL,
8   `last_name` varchar(150) NOT NULL,
9   `email` varchar(254) NOT NULL,
10  `is_staff` tinyint(1) NOT NULL,
11  `is_active` tinyint(1) NOT NULL,
12  `date_joined` datetime(6) NOT NULL,
13  PRIMARY KEY (`id`),
14  UNIQUE KEY `username` (`username`)
15 ) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=utf8;
```

Django可以保持User模型的基础上，增加1对1关系的从表来扩展一些字段。但是，这等于有2个模型类，后面序列化、操作都较为麻烦。参考 <https://docs.djangoproject.com/en/3.2/topics/auth/customizing/#extending-the-existing-user-model>

Django用户重建User模型

- 注册应用
- **一定要使用** AUTH_USER_MODEL=我的应用.User模型类 这样的配置
- 使用 django.contrib.auth.get_user_model() 获得模型类

settings.py

```
1 INSTALLED_APPS = [
2     'django.contrib.admin',
3     'django.contrib.auth',
4     'django.contrib.contenttypes',
5     'django.contrib.sessions',
6     'django.contrib.messages',
7     'django.contrib.staticfiles',
8     'user',
9 ]
10
11 AUTH_USER_MODEL = 'user.UserProfile'
```

user/models.py

```

1 from django.db import models
2 from django.contrib.auth.models import AbstractUser
3
4
5 # 替换原有User模型类
6 class UserProfile(AbstractUser):
7     class Meta:
8         db_table = "auth_user" # 保持原来名字, 方便使用
9         verbose_name = "用户详细信息"
10        phone = models.CharField(max_length=32, verbose_name='电话号码',
            null=True, blank=True)

```

由于是替换原有的模型类, 所以, 把数据库所有表情况, 重建。

事实上, 重建User模型类应该项目开始时就做, 而不是现在, 应该一开始就规划好。

```

1 $ python manage.py makemigrations
2 $ python manage.py migrate
3
4 $ python manage.py createsuperuser

```

```

1 CREATE TABLE `auth_user` (
2   `id` bigint(20) NOT NULL AUTO_INCREMENT,
3   `password` varchar(128) NOT NULL,
4   `last_login` datetime(6) DEFAULT NULL,
5   `is_superuser` tinyint(1) NOT NULL,
6   `username` varchar(150) NOT NULL,
7   `first_name` varchar(150) NOT NULL,
8   `last_name` varchar(150) NOT NULL,
9   `email` varchar(254) NOT NULL,
10  `is_staff` tinyint(1) NOT NULL,
11  `is_active` tinyint(1) NOT NULL,
12  `date_joined` datetime(6) NOT NULL,
13  `phone` varchar(32) DEFAULT NULL,
14  PRIMARY KEY (`id`),
15  UNIQUE KEY `username` (`username`)
16 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;

```

序列化器

user/serializers.py

```

1 from rest_framework import serializers
2 from .models import UserProfile
3
4
5 class UserSerializer(serializers.ModelSerializer):
6     class Meta:
7         model = UserProfile
8         fields = [
9             'id', 'username', 'first_name', 'last_name', 'is_superuser',
10            'email', 'is_active', 'phone', 'password'
11            # 'groups', 'user_permissions'
12        ]
13        extra_kwargs = {

```

```

14         'username': {'max_length': 16, "min_length":4}, # 重新限定用户名长
    度
15         'password' : {"write_only":True},
16         'is_superuser': {'default': False},
17         'is_active': {'default':False}
18     }

```

后台视图类

功能有

- 列表页：用户列表、增加用户
- 详情页：用户详情页、修改用户、删除用户

所以，采用ModelViewSet比较合适。

```

1 from rest_framework.viewsets import ModelViewSet
2 from django.contrib.auth import get_user_model
3 from .serializers import UserSerializer
4
5 class UserViewSet(ModelViewSet):
6     queryset = get_user_model().objects.all()
7     serializer_class = UserSerializer
8     # permission_classes = [IsAdminUser] # 必须是管理员才能管理用户

```

user/urls.py

```

1 from django.urls import path
2 from .views import menulist_view, UserViewSet
3 from rest_framework.routers import SimpleRouter
4
5 router = SimpleRouter()
6 router.register('', UserViewSet)
7
8 urlpatterns = [
9     path('menulist/', menulist_view),
10 ] + router.urls
11

```

新增用户

注册组件

使用el-dialog对话框组件，参考 <https://element.eleme.cn/#/zh-CN/component/dialog>

```

1 import Vue from 'vue'
2 import {
3   Form, FormItem, Input, Button, Message, Container,
4   Header, Aside, Main, Menu, MenuItem, Submenu, Breadcrumb,
5   BreadcrumbItem, Card, Row, Col, Table, TableColumn,
6   Dialog
7 } from 'element-ui'
8 import 'element-ui/lib/theme-chalk/index.css' // UI组件样式
9
10 vue.use(Dialog)

```

Dialog组件使用

参考嵌套表单的Dialog

- 对话框 `:visible.sync="dialogvisible"` 来控制对话框的显示和消失
- 对话框关闭 `@close` 重置表单, 利用 `$refs` 找到表单 `resetFields()`, `<el-form-item label="邮箱" prop="email">`, 有 `prop` 的才能被重置
- 确定 按钮 `@click` 保存数据, 发给后台
- 取消 按钮 `@click` 让对话框消失

```

1 <template>
2   <div>
3     <el-breadcrumb separator="/">
4       <el-breadcrumb-item :to="{ path: '/home' }">首页</el-breadcrumb-item>
5       <el-breadcrumb-item>用户管理</el-breadcrumb-item>
6       <el-breadcrumb-item>用户列表</el-breadcrumb-item>
7     </el-breadcrumb>
8     <el-card class="box-card">
9       <el-row :gutter="20">
10        <el-col :span="12">
11          <el-input placeholder="请输入内容" v-model="keywords">
12            <el-button slot="append" icon="el-icon-search"></el-button>
13          </el-input>
14        </el-col>
15        <el-col :span="12">
16          <el-button type="primary" @click="addUserDialogVisible = true">增
17          加用户</el-button>
18        </el-col>
19      </el-row>
20      <el-table :data="tableData" border style="width: 100%">
21        <el-table-column prop="date" label="日期" width="180"> </el-table-
22        column>
23        <el-table-column prop="name" label="姓名" width="180"> </el-table-
24        column>
25        <el-table-column prop="address" label="地址"> </el-table-column>
26      </el-table>
27    </el-card>
28    <!-- 添加用户对话框 -->
29    <el-dialog title="增加用户" :visible.sync="addUserDialogVisible"
30      @close="resetForm('addUser')">
31      <el-form :model="addUserForm" :rules="addUserRules" ref="addUser"
32        label-width="100px">
33        <el-form-item label="用户名" prop="username">
34          <el-input v-model="addUserForm.username"></el-input>
35        </el-form-item>

```



```

31     <el-form-item label="密码" prop="password">
32         <el-input type="password" v-model="addUserForm.password"></el-
input>
33     </el-form-item>
34     <el-form-item label="确认密码" prop="checkPass">
35         <el-input type="password" v-model="addUserForm.checkPass"></el-
input>
36     </el-form-item>
37     <el-form-item label="电话" prop="phone">
38         <el-input v-model="addUserForm.phone"></el-input>
39     </el-form-item>
40     <el-form-item label="姓" prop="last_name">
41         <el-input v-model="addUserForm.last_name"></el-input>
42     </el-form-item>
43     <el-form-item label="名" prop="first_name">
44         <el-input v-model="addUserForm.first_name"></el-input>
45     </el-form-item>
46     <el-form-item label="邮箱" prop="email">
47         <el-input v-model="addUserForm.email"></el-input>
48     </el-form-item>
49 </el-form>
50 <div slot="footer" class="dialog-footer">
51     <el-button @click="addUserDialogVisible = false">取 消</el-button>
52     <el-button type="primary" @click="addUser">确 定</el-button>
53 </div>
54 </el-dialog>
55 </div>
56 </template>
57
58 <script>
59 export default {
60     data() {
61         const validatePass = (rule, value, callback) => {
62             if (value !== this.addUserForm.password) {
63                 callback(new Error('两次输入密码不一致!'))
64             } else {
65                 callback()
66             }
67         }
68         return {
69             keywords: '',
70             // 添加用户对话框
71             addUserDialogVisible: false,
72             addUserForm: {
73                 username: '',
74                 password: '',
75                 checkPass: '',
76                 last_name: '',
77                 first_name: '',
78                 email: '',
79                 phone: ''
80             },
81             addUserRules: {
82                 username: [
83                     { required: true, message: '请输入登录用户名', trigger: 'blur' },
84                     { min: 4, max: 16, message: '长度在 4 到 16 个字符', trigger:
'blur' }
85                 ],

```

```

86     password: [
87       { required: true, message: '请输入密码', trigger: 'blur' },
88       { min: 4, max: 16, message: '长度在 4 到 16 个字符', trigger:
'blur' }
89     ],
90     checkPass: [
91       { required: true, message: '请输入密码', trigger: 'blur' },
92       { validator: validatePass, trigger: 'blur' }
93     ]
94   },
95   // 用户列表表格
96   tableData: [
97     {
98       date: '2016-05-02',
99       name: '王小虎',
100      address: '上海市普陀区金沙江路 1518 弄'
101    },
102    {
103      date: '2016-05-04',
104      name: '王小虎',
105      address: '上海市普陀区金沙江路 1517 弄'
106    },
107    {
108      date: '2016-05-01',
109      name: '王小虎',
110      address: '上海市普陀区金沙江路 1519 弄'
111    }
112  ]
113 },
114 },
115 methods: {
116   resetForm(formName) {
117     this.$refs[formName].resetFields() // 重置表单, 表单必须有rules
118   },
119   addUser() {
120     const name = 'addUser'
121     this.$refs[name].validate(async (valid) => {
122       if (valid) {
123         console.log(valid, '+++')
124         // 表单验证成功, 数据post到后端/users/
125         const { data: response } = await this.$http.post('users/',
this.addUserForm)
126         if (response.code) {
127           return this.$message.error(response.message)
128         }
129         this.addUserDialogVisible = false
130         this.resetForm(name)
131       }
132     })
133   }
134 }
135 }
136 </script>
137
138 <style lang="less" scoped>
139

```

提交后发现成功创建用户，但是，数据库表中密码不对，原来是自动生成的类，没有调用密码生成函数。

手动修改校验器函数，校验完返回一个加密的密码。

user/serializers.py

```
1 from rest_framework import serializers
2 from .models import UserProfile
3 from django.contrib.auth.models import make_password
4
5
6 class UserSerializer(serializers.ModelSerializer):
7     class Meta:
8         model = UserProfile
9         fields = [
10             'id', 'username', 'first_name', 'last_name', 'is_superuser',
11             'email', 'is_active', 'phone', 'password'
12             # 'groups', 'user_permissions'
13         ]
14         extra_kwargs = {
15             'username': {'max_length': 16, "min_length":4}, # 重新限定用户名长度
16             'password': {"write_only":True},
17             'is_superuser': {'default': False},
18             'is_active': {'default':False}
19         }
20
21     def validate_password(self, value):
22         if 4 <= len(value) <= 16:
23             # print(value, make_password(value), '+++++')
24             return make_password(value)
25         raise serializers.ValidationError("The length of password must be between 4 and 16")
```

用户列表

表格参考 <https://element.eleme.cn/#/zh-CN/component/table>

本组件开始初始化的时候，就需要访问后台，返回用户表格数据。

- getUserList()对后台访问返回用户分页列表
- created()中调用getUserList()
- el-table中绑定数据 :data="userList"
- el-table-column中 prop="id" 选择字段值, label="id" 设置标题

Users.vue如下

```
1 <template>
2   <div>
3     <el-breadcrumb separator="/">
4       <el-breadcrumb-item :to="{ path: '/home' }">首页</el-breadcrumb-item>
5       <el-breadcrumb-item>用户管理</el-breadcrumb-item>
6       <el-breadcrumb-item>用户列表</el-breadcrumb-item>
7     </el-breadcrumb>
8     <el-card class="box-card">
9       <el-row :gutter="20">
```

```

10     <el-col :span="12">
11         <el-input placeholder="请输入内容" v-model="keywords">
12             <el-button slot="append" icon="el-icon-search"></el-button>
13         </el-input>
14     </el-col>
15     <el-col :span="12">
16         <el-button type="primary" @click="addUserDialogVisible = true">增
加用户</el-button>
17     </el-col>
18 </el-row>
19 <el-table :data="userList" border style="width: 100%">
20     <el-table-column prop="id" label="id"> </el-table-column>
21     <el-table-column prop="username" label="登录名"> </el-table-column>
22     <el-table-column prop="is_active" label="激活"> </el-table-column>
23     <el-table-column prop="is_superuser" label="管理员"> </el-table-
column>
24     <el-table-column prop="phone" label="电话"> </el-table-column>
25 </el-table>
26 </el-card>
27 <!-- 添加用户对话框 -->
28 <el-dialog title="增加用户" :visible.sync="addUserDialogVisible"
@close="resetForm('addUser')">
29     <el-form :model="addUserForm" :rules="addUserRules" ref="addUser"
label-width="100px">
30         <el-form-item label="用户名" prop="username">
31             <el-input v-model="addUserForm.username"></el-input>
32         </el-form-item>
33         <el-form-item label="密码" prop="password">
34             <el-input type="password" v-model="addUserForm.password"></el-
input>
35         </el-form-item>
36         <el-form-item label="确认密码" prop="checkPass">
37             <el-input type="password" v-model="addUserForm.checkPass"></el-
input>
38         </el-form-item>
39         <el-form-item label="电话" prop="phone">
40             <el-input v-model="addUserForm.phone"></el-input>
41         </el-form-item>
42         <el-form-item label="姓" prop="last_name">
43             <el-input v-model="addUserForm.last_name"></el-input>
44         </el-form-item>
45         <el-form-item label="名" prop="first_name">
46             <el-input v-model="addUserForm.first_name"></el-input>
47         </el-form-item>
48         <el-form-item label="邮箱" prop="email">
49             <el-input v-model="addUserForm.email"></el-input>
50         </el-form-item>
51     </el-form>
52     <div slot="footer" class="dialog-footer">
53         <el-button @click="addUserDialogVisible = false">取 消</el-button>
54         <el-button type="primary" @click="addUser">确 定</el-button>
55     </div>
56 </el-dialog>
57 </div>
58 </template>
59
60 <script>
61 export default {

```

```

62   created() {
63       this.getUserList()
64   },
65   data() {
66       const validatePass2 = (rule, value, callback) => {
67           if (value === '') {
68               callback(new Error('请再次输入密码'))
69           } else if (value !== this.addUserForm.password) {
70               callback(new Error('两次输入密码不一致!'))
71           } else {
72               callback()
73           }
74       }
75       return {
76           keywords: '',
77           // 添加用户对话框
78           addUserDialogvisible: false,
79           addUserForm: {
80               username: '',
81               password: '',
82               checkPass: '',
83               last_name: '',
84               first_name: '',
85               email: '',
86               phone: ''
87           },
88           addUserRules: {
89               username: [
90                   { required: true, message: '请输入登录用户名', trigger: 'blur' },
91                   { min: 4, max: 16, message: '长度在 4 到 16 个字符', trigger:
'blur' }
92               ],
93               password: [
94                   { required: true, message: '请输入密码', trigger: 'blur' },
95                   { min: 4, max: 16, message: '长度在 4 到 16 个字符', trigger:
'blur' }
96               ],
97               checkPass: [{ validator: validatePass2, trigger: 'blur' }]
98           },
99           // 用户列表表格
100          userList: []
101      }
102  },
103  methods: {
104      resetForm(formName) {
105          this.$refs[formName].resetFields() // 重置表单, 表单必须有rules
106      },
107      addUser() {
108          const name = 'addUser'
109          this.$refs[name].validate(async (valid) => {
110              if (valid) {
111                  console.log(valid, '+++')
112                  // 表单验证成功, 数据post到后端/users/
113                  const { data: response } = await this.$http.post('users/',
this.addUserForm)
114                  if (response.code) {
115                      return this.$message.error(response.message)
116                  }

```

```

117         this.addUserDialogvisible = false
118         this.resetForm(name)
119         this.getUserList() // 刷新用户列表
120     }
121 })
122 },
123 async getUserList() {
124     const { data: response } = await this.$http.get('users/')
125     if (response.code) {
126         return this.$message.error(response.message)
127     }
128     this.userList = response
129 }
130 }
131 }
132 </script>
133
134 <style lang="less" scoped>
135 </style>

```

分页功能

后台分页

rest_framework.pagination.PageNumberPagination 和前台分页组件需要的数据不匹配，前台分页组件往往需要total总数、page-size页条目数、current-page当前页码，这里的数据明显不符合要求。

所以，在后台自定义一个分页类，方便使用。

新建utils/paginations.py，新建自定义分页类PageNumberPagination

```

1  from rest_framework import pagination
2  from rest_framework.response import Response
3
4
5  class PageNumberPagination(pagination.PageNumberPagination):
6      def get_paginated_response(self, data):
7          return Response({
8              'pagination': {
9                  'total': self.page.paginator.count,
10                 'size': self.page_size,
11                 'page': self.page.number
12             },
13             'results': data,
14         })

```

```

1 REST_FRAMEWORK = {
2     'EXCEPTION_HANDLER': 'utils.exceptions.global_exception_handler',
3     'DEFAULT_AUTHENTICATION_CLASSES': [
4         'rest_framework_simplejwt.authentication.JWTAuthentication'
5     ],
6     'DEFAULT_PERMISSION_CLASSES': [
7         'rest_framework.permissions.IsAuthenticated', # 全局，要求被认证，也就是
            登录成功
8     ],
9     'DEFAULT_PAGINATION_CLASS': 'utils.paginations.PageNumberPagination',
10    'PAGE_SIZE': 20,
11 }

```

前台分页

组件注册

```

1 import Vue from 'vue'
2 import {
3     Form, FormItem, Input, Button, Message, Container,
4     Header, Aside, Main, Menu, MenuItem, Submenu, Breadcrumb,
5     BreadcrumbItem, Card, Row, Col, Table, TableColumn,
6     Dialog, Pagination
7 } from 'element-ui'
8 import 'element-ui/lib/theme-chalk/index.css' // UI组件样式
9
10 Vue.use(Pagination)

```

分页组件

- layout="total, prev, pager, next, jumper" 包含哪些部件。sizes可以不要，一般不让改
- :total="1200"
- :page-size="20"
- :current-page="4"

```

1 <template>
2   <div>
3     <el-breadcrumb separator="/">
4       <el-breadcrumb-item :to="{ path: '/home' }">首页</el-breadcrumb-item>
5       <el-breadcrumb-item>用户管理</el-breadcrumb-item>
6       <el-breadcrumb-item>用户列表</el-breadcrumb-item>
7     </el-breadcrumb>
8     <el-card class="box-card">
9       <el-row :gutter="20">
10        <el-col :span="12">
11          <el-input placeholder="请输入内容" v-model="keywords">
12            <el-button slot="append" icon="el-icon-search"></el-button>
13          </el-input>
14        </el-col>
15        <el-col :span="12">
16          <el-button type="primary" @click="addUserDialogVisible = true">增
            加用户</el-button>
17        </el-col>
18      </el-row>

```

```

19     <el-table :data="userList" border style="width: 100%">
20       <el-table-column prop="id" label="id"> </el-table-column>
21       <el-table-column prop="username" label="登录名"> </el-table-column>
22       <el-table-column prop="is_active" label="激活"> </el-table-column>
23       <el-table-column prop="is_superuser" label="管理员"> </el-table-
column>
24       <el-table-column prop="phone" label="电话"> </el-table-column>
25     </el-table>
26     <el-pagination
27       @size-change="handleSizeChange"
28       @current-change="handleCurrentChange"
29       :current-page="pagination.page"
30       :page-size="pagination.size"
31       layout="total, prev, pager, next, jumper"
32       :total="pagination.total"
33     >
34   </el-pagination>
35 </el-card>
36 <!-- 添加用户对话框 -->
37 <el-dialog title="增加用户" :visible.sync="addUserDialogVisible"
@close="resetForm('addUser')">
38   <el-form :model="addUserForm" :rules="addUserRules" ref="addUser"
label-width="100px">
39     <el-form-item label="用户名" prop="username">
40       <el-input v-model="addUserForm.username"></el-input>
41     </el-form-item>
42     <el-form-item label="密码" prop="password">
43       <el-input type="password" v-model="addUserForm.password"></el-
input>
44     </el-form-item>
45     <el-form-item label="确认密码" prop="checkPass">
46       <el-input type="password" v-model="addUserForm.checkPass"></el-
input>
47     </el-form-item>
48     <el-form-item label="电话" prop="phone">
49       <el-input v-model="addUserForm.phone"></el-input>
50     </el-form-item>
51     <el-form-item label="姓" prop="last_name">
52       <el-input v-model="addUserForm.last_name"></el-input>
53     </el-form-item>
54     <el-form-item label="名" prop="first_name">
55       <el-input v-model="addUserForm.first_name"></el-input>
56     </el-form-item>
57     <el-form-item label="邮箱" prop="email">
58       <el-input v-model="addUserForm.email"></el-input>
59     </el-form-item>
60   </el-form>
61   <div slot="footer" class="dialog-footer">
62     <el-button @click="addUserDialogVisible = false">取 消</el-button>
63     <el-button type="primary" @click="addUser">确 定</el-button>
64   </div>
65 </el-dialog>
66 </div>
67 </template>
68
69 <script>
70 export default {
71   created() {

```



```

72     this.getUserList()
73   },
74   data() {
75     const validatePass2 = (rule, value, callback) => {
76       if (value === '') {
77         callback(new Error('请再次输入密码'))
78       } else if (value !== this.addUserForm.password) {
79         callback(new Error('两次输入密码不一致!'))
80       } else {
81         callback()
82       }
83     }
84     return {
85       keywords: '',
86       // 添加用户对话框
87       addUserDialogVisible: false,
88       addUserForm: {
89         username: '',
90         password: '',
91         checkPass: '',
92         last_name: '',
93         first_name: '',
94         email: '',
95         phone: ''
96       },
97       addUserRules: {
98         username: [
99           { required: true, message: '请输入登录用户名', trigger: 'blur' },
100          { min: 4, max: 16, message: '长度在 4 到 16 个字符', trigger:
101            'blur' }
102        ],
103        password: [
104          { required: true, message: '请输入密码', trigger: 'blur' },
105          { min: 4, max: 16, message: '长度在 4 到 16 个字符', trigger:
106            'blur' }
107        ],
108        checkPass: [{ validator: validatePass2, trigger: 'blur' }]
109      },
110      // 用户列表表格
111      userList: [],
112      pagination: { page: 1, size: 20, total: 0 }
113    },
114    methods: {
115      resetForm(formName) {
116        this.$refs[formName].resetFields() // 重置表单, 表单必须有rules
117      },
118      addUser() {
119        const name = 'addUser'
120        this.$refs[name].validate(async (valid) => {
121          if (valid) {
122            console.log(valid, '+++')
123            // 表单验证成功, 数据post到后端/users/
124            const { data: response } = await this.$http.post('users/',
125              this.addUserForm)
126            if (response.code) {
127              return this.$message.error(response.message)
128            }
129          }
130        })
131      }
132    }
133  }
134 }

```

```
127         this.addUserDialogvisible = false
128         this.resetForm(name)
129         this.getUserList() // 刷新用户列表
130     }
131 })
132 },
133 async getUserList(page = 1) {
134     if (!page) {
135         page = 1
136     }
137     const { data: response } = await this.$http.get(`users/?
page=${page}`)
138     if (response.code) {
139         return this.$message.error(response.message)
140     }
141     this.userList = response.results
142     this.pagination = response.pagination
143 },
144 handleSizeChange(val) {
145     console.log(`每页 ${val} 条`)
146 },
147 handleCurrentChange(val) {
148     console.log(`当前页: ${val}`)
149     this.getUserList(val)
150 }
151 }
152 }
153 </script>
154
155 <style lang="less" scoped>
156 </style>
```