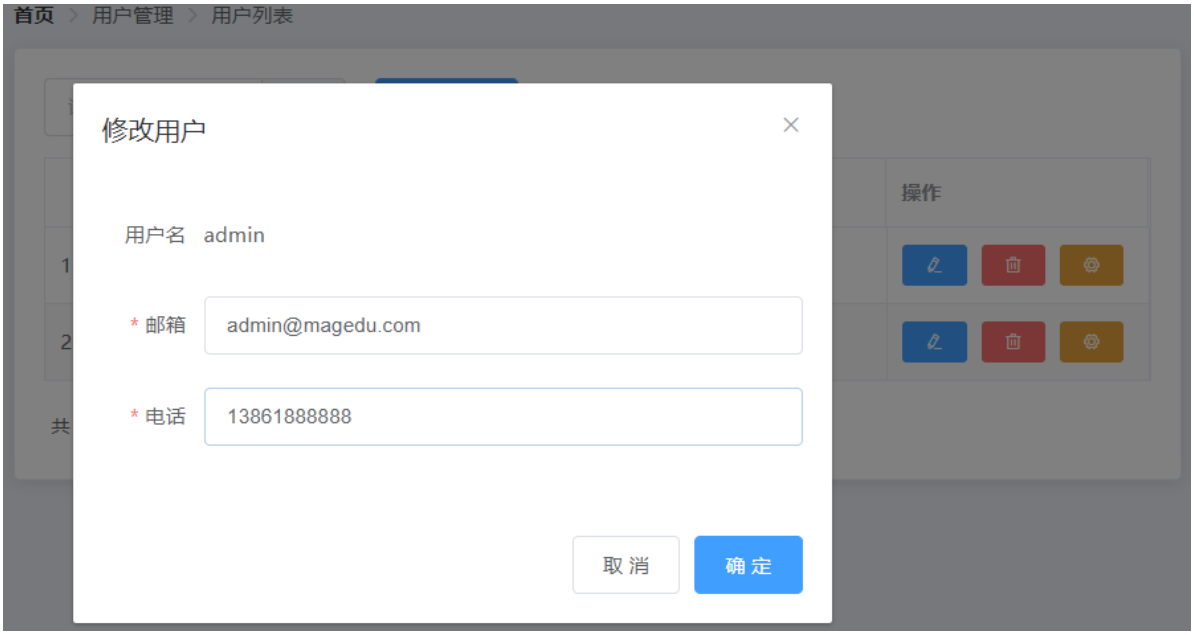
用户功能实现

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

```vue
<template>
  <div>用户列表页</div>
</template>

<script>
export default {}
</script>

<style>
</style>
```

路由配置（省略未变化代码）

```
import User from '../components/user/Users.vue'

const routes = [
```

```
  { path: '/', redirect: '/login' },
  { path: '/login', component: Login },
  {
    path: '/home',
    component: Home,
    redirect: '/welcome', // 访问/home重定向到/welcome，进入子路由
    children: [
      { path: '/welcome', component: Welcome },
      { path: '/users', component: User }
    ]
  }
]
```

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

```
import Vue from 'vue'
import {
  Form, FormItem, Input, Button, Message, Container,
  Header, Aside, Main, Menu, MenuItem, Submenu, Breadcrumb,
  BreadcrumbItem, Card, Row, Col, Table, TableColumn
} from 'element-ui'
import 'element-ui/lib/theme-chalk/index.css' // UI组件样式

Vue.use(Breadcrumb)
Vue.use(BreadcrumbItem)
Vue.use(Card)
Vue.use(Row)
Vue.use(Col)
Vue.use(Table)
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

```
/* 全局样式表 */
html, body, #app {
    height:100%;
    margin: 0;
    padding: 0;
    min-height: 200px;
}
.el-breadcrumb {
  margin-bottom: 15px;
}

.el-table {
  margin: 15px 0px;
}
```

Users.vue

```vue
<template>
  <div>
    <el-breadcrumb separator="/">
      <el-breadcrumb-item :to="{ path: '/home' }">首页</el-breadcrumb-item>
      <el-breadcrumb-item>用户管理</el-breadcrumb-item>
      <el-breadcrumb-item>用户列表</el-breadcrumb-item>
    </el-breadcrumb>
    <el-card class="box-card">
      <el-row :gutter="20">
        <el-col :span="12">
          <el-input placeholder="请输入内容" v-model="input3">
            <el-button slot="append" icon="el-icon-search"></el-button>
          </el-input>
        </el-col>
        <el-col :span="12">
          <el-button type="primary">增加用户</el-button>
        </el-col>
      </el-row>
      <el-table :data="tableData" border style="width: 100%">
        <el-table-column prop="date" label="日期" width="180"> </el-table-column>
        <el-table-column prop="name" label="姓名" width="180"> </el-table-column>
        <el-table-column prop="address" label="地址"> </el-table-column>
      </el-table>
    </el-card>
  </div>
</template>

<script>
export default {
  data() {
    return {
      tableData: [
        {
          date: '2016-05-02',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄'
        },
        {
          date: '2016-05-04',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1517 弄'
        },
        {
          date: '2016-05-01',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1519 弄'
        }
      ]
    }
  }
}
</script>
```

```
</script>

<style lang="less" scoped>
</style>
```

重建User模型类

User模型类是Django内建的，对应auth_user表

```
CREATE TABLE `auth_user` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `password` varchar(128) NOT NULL,
  `last_login` datetime(6) DEFAULT NULL,
  `is_superuser` tinyint(1) NOT NULL,
  `username` varchar(150) NOT NULL,
  `first_name` varchar(150) NOT NULL,
  `last_name` varchar(150) NOT NULL,
  `email` varchar(254) NOT NULL,
  `is_staff` tinyint(1) NOT NULL,
  `is_active` tinyint(1) NOT NULL,
  `date_joined` datetime(6) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `username` (`username`)
) 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

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'user',
]

AUTH_USER_MODEL = 'user.UserProfile'
```

user/models.py

```python
from django.db import models
from django.contrib.auth.models import AbstractUser


# 替换原有User模型类
class UserProfile(AbstractUser):
    class Meta:
        db_table = "auth_user" # 保持原来名字，方便使用
        verbose_name = "用户详细信息"
    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。

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

$ python manage.py createsuperuser
```

```
CREATE TABLE `auth_user` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `password` varchar(128) NOT NULL,
  `last_login` datetime(6) DEFAULT NULL,
  `is_superuser` tinyint(1) NOT NULL,
  `username` varchar(150) NOT NULL,
  `first_name` varchar(150) NOT NULL,
  `last_name` varchar(150) NOT NULL,
  `email` varchar(254) NOT NULL,
  `is_staff` tinyint(1) NOT NULL,
  `is_active` tinyint(1) NOT NULL,
  `date_joined` datetime(6) NOT NULL,
  `phone` varchar(32) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `username` (`username`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

序列化器

user/serializers.py

```python
from rest_framework import serializers
from .models import UserProfile


class UserSerializer(serializers.ModelSerializer):
    class Meta:
        model = UserProfile
        fields = [
            'id', 'username', 'first_name', 'last_name', 'is_superuser',
            'email', 'is_active', 'phone', 'password'
            # 'groups', 'user_permissions'
        ]
        extra_kwargs = {
```

```
            'username': {'max_length': 16, "min_length":4}, # 重新限定用户名长度
            'password' : {"write_only":True},
            'is_superuser': {'default': False},
            'is_active': {'default':False}
        }
```

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。

```
from rest_framework.viewsets import ModelViewSet
from django.contrib.auth import get_user_model
from .serializers import UserSerializer

class UserViewSet(ModelViewSet):
    queryset = get_user_model().objects.all()
    serializer_class = UserSerializer
    # permission_classes = [IsAdminUser] # 必须是管理员才能管理用户
```

user/urls.py

```
from django.urls import path
from .views import menulist_view, UserViewSet
from rest_framework.routers import SimpleRouter

router = SimpleRouter()
router.register('', UserViewSet)

urlpatterns = [
    path('menulist/', menulist_view),
] + router.urls

```

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

```
import Vue from 'vue'
import {
  Form, FormItem, Input, Button, Message, Container,
  Header, Aside, Main, Menu, MenuItem, Submenu, Breadcrumb,
  BreadcrumbItem, Card, Row, Col, Table, TableColumn,
  Dialog
} from 'element-ui'
import 'element-ui/lib/theme-chalk/index.css' // UI组件样式

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

```
<template>
  <div>
    <el-breadcrumb separator="/">
      <el-breadcrumb-item :to="{ path: '/home' }">首页</el-breadcrumb-item>
      <el-breadcrumb-item>用户管理</el-breadcrumb-item>
      <el-breadcrumb-item>用户列表</el-breadcrumb-item>
    </el-breadcrumb>
    <el-card class="box-card">
      <el-row :gutter="20">
        <el-col :span="12">
          <el-input placeholder="请输入内容" v-model="keywords">
            <el-button slot="append" icon="el-icon-search"></el-button>
          </el-input>
        </el-col>
        <el-col :span="12">
          <el-button type="primary" @click="addUserDialogVisible = true">增加用户</el-button>
        </el-col>
      </el-row>
      <el-table :data="tableData" border style="width: 100%">
        <el-table-column prop="date" label="日期" width="180"> </el-table-column>
        <el-table-column prop="name" label="姓名" width="180"> </el-table-column>
        <el-table-column prop="address" label="地址"> </el-table-column>
      </el-table>
    </el-card>
    <!-- 添加用户对话框 -->
    <el-dialog title="增加用户" :visible.sync="addUserDialogVisible" @close="resetForm('addUser')">
      <el-form :model="addUserForm" :rules="addUserRules" ref="addUser" label-width="100px">
        <el-form-item label="用户名" prop="username">
          <el-input v-model="addUserForm.username"></el-input>
        </el-form-item>
```

```
        <el-form-item label="密码" prop="password">
          <el-input type="password" v-model="addUserForm.password"></el-input>
        </el-form-item>
        <el-form-item label="确认密码" prop="checkPass">
          <el-input type="password" v-model="addUserForm.checkPass"></el-input>
        </el-form-item>
        <el-form-item label="电话" prop="phone">
          <el-input v-model="addUserForm.phone"></el-input>
        </el-form-item>
        <el-form-item label="姓" prop="last_name">
          <el-input v-model="addUserForm.last_name"></el-input>
        </el-form-item>
        <el-form-item label="名" prop="first_name">
          <el-input v-model="addUserForm.first_name"></el-input>
        </el-form-item>
        <el-form-item label="邮箱" prop="email">
          <el-input v-model="addUserForm.email"></el-input>
        </el-form-item>
      </el-form>
      <div slot="footer" class="dialog-footer">
        <el-button @click="addUserDialogVisible = false">取 消</el-button>
        <el-button type="primary" @click="addUser">确 定</el-button>
      </div>
    </el-dialog>
  </div>
</template>

<script>
export default {
  data() {
    const validatePass = (rule, value, callback) => {
      if (value !== this.addUserForm.password) {
        callback(new Error('两次输入密码不一致!'))
      } else {
        callback()
      }
    }
    return {
      keywords: '',
      // 添加用户对话框
      addUserDialogVisible: false,
      addUserForm: {
        username: '',
        password: '',
        checkPass: '',
        last_name: '',
        first_name: '',
        email: '',
        phone: ''
      },
      addUserRules: {
        username: [
          { required: true, message: '请输入登录用户名', trigger: 'blur' },
          { min: 4, max: 16, message: '长度在 4 到 16 个字符', trigger: 'blur' }
        ],
```

```
        password: [
          { required: true, message: '请输入密码', trigger: 'blur' },
          { min: 4, max: 16, message: '长度在 4 到 16 个字符', trigger:
'blur' }
        ],
        checkPass: [
          { required: true, message: '请输入密码', trigger: 'blur' },
          { validator: validatePass, trigger: 'blur' }
        ]
      },
      // 用户列表表格
      tableData: [
        {
          date: '2016-05-02',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1518 弄'
        },
        {
          date: '2016-05-04',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1517 弄'
        },
        {
          date: '2016-05-01',
          name: '王小虎',
          address: '上海市普陀区金沙江路 1519 弄'
        }
      ]
    }
  },
  methods: {
    resetForm(formName) {
      this.$refs[formName].resetFields() // 重置表单，表单必须有rules
    },
    addUser() {
      const name = 'addUser'
      this.$refs[name].validate(async (valid) => {
        if (valid) {
          console.log(valid, '+++')
          // 表单验证成功，数据post到后端/users/
          const { data: response } = await this.$http.post('users/',
this.addUserForm)
          if (response.code) {
            return this.$message.error(response.message)
          }
          this.addUserDialogVisible = false
          this.resetForm(name)
        }
      })
    }
  }
}
</script>

<style lang="less" scoped>
</style>
```

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

```
from rest_framework import serializers
from .models import UserProfile
from django.contrib.auth.models import make_password


class UserSerializer(serializers.ModelSerializer):
    class Meta:
        model = UserProfile
        fields = [
            'id', 'username', 'first_name', 'last_name', 'is_superuser',
            'email', 'is_active', 'phone', 'password'
            # 'groups', 'user_permissions'
        ]
        extra_kwargs = {
            'username': {'max_length': 16, "min_length":4}, # 重新限定用户名长度
            'password' : {"write_only":True},
            'is_superuser': {'default': False},
            'is_active': {'default':False}
        }

    def validate_password(self, value):
        if 4 <= len(value) <= 16:
            # print(value, make_password(value), '+++++++++++++++')
            return make_password(value)
        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

```
<template>
  <div>
    <el-breadcrumb separator="/">
      <el-breadcrumb-item :to="{ path: '/home' }">首页</el-breadcrumb-item>
      <el-breadcrumb-item>用户管理</el-breadcrumb-item>
      <el-breadcrumb-item>用户列表</el-breadcrumb-item>
    </el-breadcrumb>
    <el-card class="box-card">
      <el-row :gutter="20">
```

```
        <el-col :span="12">
          <el-input placeholder="请输入内容" v-model="keywords">
            <el-button slot="append" icon="el-icon-search"></el-button>
          </el-input>
        </el-col>
        <el-col :span="12">
          <el-button type="primary" @click="addUserDialogVisible = true">增加用户</el-button>
        </el-col>
      </el-row>
      <el-table :data="userList" border style="width: 100%">
        <el-table-column prop="id" label="id"> </el-table-column>
        <el-table-column prop="username" label="登录名"> </el-table-column>
        <el-table-column prop="is_active" label="激活"> </el-table-column>
        <el-table-column prop="is_superuser" label="管理员"> </el-table-column>
        <el-table-column prop="phone" label="电话"> </el-table-column>
      </el-table>
    </el-card>
    <!-- 添加用户对话框 -->
    <el-dialog title="增加用户" :visible.sync="addUserDialogVisible" @close="resetForm('addUser')">
      <el-form :model="addUserForm" :rules="addUserRules" ref="addUser" label-width="100px">
        <el-form-item label="用户名" prop="username">
          <el-input v-model="addUserForm.username"></el-input>
        </el-form-item>
        <el-form-item label="密码" prop="password">
          <el-input type="password" v-model="addUserForm.password"></el-input>
        </el-form-item>
        <el-form-item label="确认密码" prop="checkPass">
          <el-input type="password" v-model="addUserForm.checkPass"></el-input>
        </el-form-item>
        <el-form-item label="电话" prop="phone">
          <el-input v-model="addUserForm.phone"></el-input>
        </el-form-item>
        <el-form-item label="姓" prop="last_name">
          <el-input v-model="addUserForm.last_name"></el-input>
        </el-form-item>
        <el-form-item label="名" prop="first_name">
          <el-input v-model="addUserForm.first_name"></el-input>
        </el-form-item>
        <el-form-item label="邮箱" prop="email">
          <el-input v-model="addUserForm.email"></el-input>
        </el-form-item>
      </el-form>
      <div slot="footer" class="dialog-footer">
        <el-button @click="addUserDialogVisible = false">取 消</el-button>
        <el-button type="primary" @click="addUser">确 定</el-button>
      </div>
    </el-dialog>
  </div>
</template>

<script>
export default {
```

```
  created() {
    this.getUserList()
  },
  data() {
    const validatePass2 = (rule, value, callback) => {
      if (value === '') {
        callback(new Error('请再次输入密码'))
      } else if (value !== this.addUserForm.password) {
        callback(new Error('两次输入密码不一致!'))
      } else {
        callback()
      }
    }
    return {
      keywords: '',
      // 添加用户对话框
      addUserDialogVisible: false,
      addUserForm: {
        username: '',
        password: '',
        checkPass: '',
        last_name: '',
        first_name: '',
        email: '',
        phone: ''
      },
      addUserRules: {
        username: [
          { required: true, message: '请输入登录用户名', trigger: 'blur' },
          { min: 4, max: 16, message: '长度在 4 到 16 个字符', trigger:
'blur' }
        ],
        password: [
          { required: true, message: '请输入密码', trigger: 'blur' },
          { min: 4, max: 16, message: '长度在 4 到 16 个字符', trigger:
'blur' }
        ],
        checkPass: [{ validator: validatePass2, trigger: 'blur' }]
      },
      // 用户列表表格
      userList: []
    }
  },
  methods: {
    resetForm(formName) {
      this.$refs[formName].resetFields() // 重置表单，表单必须有rules
    },
    addUser() {
      const name = 'addUser'
      this.$refs[name].validate(async (valid) => {
        if (valid) {
          console.log(valid, '+++')
          // 表单验证成功，数据post到后端/users/
          const { data: response } = await this.$http.post('users/',
this.addUserForm)
          if (response.code) {
            return this.$message.error(response.message)
          }
```

```
            this.addUserDialogVisible = false
            this.resetForm(name)
            this.getUserList() // 刷新用户列表
          }
        })
    },
    async getUserList() {
      const { data: response } = await this.$http.get('users/')
      if (response.code) {
        return this.$message.error(response.message)
      }
      this.userList = response
    }
  }
}
</script>

<style lang="less" scoped>
</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

```python
from rest_framework import pagination
from rest_framework.response import Response


class PageNumberPagination(pagination.PageNumberPagination):
    def get_paginated_response(self, data):
        return Response({
            'pagination':{
                'total': self.page.paginator.count,
                'size': self.page_size,
                'page': self.page.number
            },
            'results': data,
        })
```

```python
REST_FRAMEWORK = {
    'EXCEPTION_HANDLER': 'utils.exceptions.global_exception_handler',
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'rest_framework_simplejwt.authentication.JWTAuthentication'
    ],
    'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.IsAuthenticated', # 全局，要求被认证，也就是登录成功
    ],
    'DEFAULT_PAGINATION_CLASS': 'utils.paginations.PageNumberPagination',
    'PAGE_SIZE': 20,
}
```

前台分页

组件注册

```javascript
import Vue from 'vue'
import {
  Form, FormItem, Input, Button, Message, Container,
  Header, Aside, Main, Menu, MenuItem, Submenu, Breadcrumb,
  BreadcrumbItem, Card, Row, Col, Table, TableColumn,
  Dialog, Pagination
} from 'element-ui'
import 'element-ui/lib/theme-chalk/index.css' // UI组件样式

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"`

```vue
<template>
  <div>
    <el-breadcrumb separator="/">
      <el-breadcrumb-item :to="{ path: '/home' }">首页</el-breadcrumb-item>
      <el-breadcrumb-item>用户管理</el-breadcrumb-item>
      <el-breadcrumb-item>用户列表</el-breadcrumb-item>
    </el-breadcrumb>
    <el-card class="box-card">
      <el-row :gutter="20">
        <el-col :span="12">
          <el-input placeholder="请输入内容" v-model="keywords">
            <el-button slot="append" icon="el-icon-search"></el-button>
          </el-input>
        </el-col>
        <el-col :span="12">
          <el-button type="primary" @click="addUserDialogVisible = true">增加用户</el-button>
        </el-col>
      </el-row>
```

```
      <el-table :data="userList" border style="width: 100%">
        <el-table-column prop="id" label="id"> </el-table-column>
        <el-table-column prop="username" label="登录名"> </el-table-column>
        <el-table-column prop="is_active" label="激活"> </el-table-column>
        <el-table-column prop="is_superuser" label="管理员"> </el-table-column>
        <el-table-column prop="phone" label="电话"> </el-table-column>
      </el-table>
      <el-pagination
        @size-change="handleSizeChange"
        @current-change="handleCurrentChange"
        :current-page="pagination.page"
        :page-size="pagination.size"
        layout="total, prev, pager, next, jumper"
        :total="pagination.total"
      >
      </el-pagination>
    </el-card>
    <!-- 添加用户对话框 -->
    <el-dialog title="增加用户" :visible.sync="addUserDialogVisible" @close="resetForm('addUser')">
      <el-form :model="addUserForm" :rules="addUserRules" ref="addUser" label-width="100px">
        <el-form-item label="用户名" prop="username">
          <el-input v-model="addUserForm.username"></el-input>
        </el-form-item>
        <el-form-item label="密码" prop="password">
          <el-input type="password" v-model="addUserForm.password"></el-input>
        </el-form-item>
        <el-form-item label="确认密码" prop="checkPass">
          <el-input type="password" v-model="addUserForm.checkPass"></el-input>
        </el-form-item>
        <el-form-item label="电话" prop="phone">
          <el-input v-model="addUserForm.phone"></el-input>
        </el-form-item>
        <el-form-item label="姓" prop="last_name">
          <el-input v-model="addUserForm.last_name"></el-input>
        </el-form-item>
        <el-form-item label="名" prop="first_name">
          <el-input v-model="addUserForm.first_name"></el-input>
        </el-form-item>
        <el-form-item label="邮箱" prop="email">
          <el-input v-model="addUserForm.email"></el-input>
        </el-form-item>
      </el-form>
      <div slot="footer" class="dialog-footer">
        <el-button @click="addUserDialogVisible = false">取 消</el-button>
        <el-button type="primary" @click="addUser">确 定</el-button>
      </div>
    </el-dialog>
  </div>
</template>

<script>
export default {
  created() {
```

```
    this.getUserList()
  },
  data() {
    const validatePass2 = (rule, value, callback) => {
      if (value === '') {
        callback(new Error('请再次输入密码'))
      } else if (value !== this.addUserForm.password) {
        callback(new Error('两次输入密码不一致!'))
      } else {
        callback()
      }
    }
    return {
      keywords: '',
      // 添加用户对话框
      addUserDialogVisible: false,
      addUserForm: {
        username: '',
        password: '',
        checkPass: '',
        last_name: '',
        first_name: '',
        email: '',
        phone: ''
      },
      addUserRules: {
        username: [
          { required: true, message: '请输入登录用户名', trigger: 'blur' },
          { min: 4, max: 16, message: '长度在 4 到 16 个字符', trigger:
'blur' }
        ],
        password: [
          { required: true, message: '请输入密码', trigger: 'blur' },
          { min: 4, max: 16, message: '长度在 4 到 16 个字符', trigger:
'blur' }
        ],
        checkPass: [{ validator: validatePass2, trigger: 'blur' }]
      },
      // 用户列表表格
      userList: [],
      pagination: { page: 1, size: 20, total: 0 }
    }
  },
  methods: {
    resetForm(formName) {
      this.$refs[formName].resetFields() // 重置表单，表单必须有rules
    },
    addUser() {
      const name = 'addUser'
      this.$refs[name].validate(async (valid) => {
        if (valid) {
          console.log(valid, '+++')
          // 表单验证成功，数据post到后端/users/
          const { data: response } = await this.$http.post('users/',
this.addUserForm)
          if (response.code) {
            return this.$message.error(response.message)
          }
```

```
          this.addUserDialogVisible = false
          this.resetForm(name)
          this.getUserList() // 刷新用户列表
        }
      })
    },
    async getUserList(page = 1) {
      if (!page) {
        page = 1
      }
      const { data: response } = await this.$http.get(`users/?page=${page}`)
      if (response.code) {
        return this.$message.error(response.message)
      }
      this.userList = response.results
      this.pagination = response.pagination
    },
    handleSizeChange(val) {
      console.log(`每页 ${val} 条`)
    },
    handleCurrentChange(val) {
      console.log(`当前页: ${val}`)
      this.getUserList(val)
    }
  }
}
</script>

<style lang="less" scoped>
</style>
```