

Cola-Ui 前端文档

序

项目地址

- 后端地址: <https://gitee.com/xiaolifeizei/cola-admin>
- 前端地址: <https://gitee.com/xiaolifeizei/cola-ui>

在线演示

- 演示地址: <http://www.cola-admin.vip>
- 默认用户: admin
- 默认密码: 123123

前端介绍

Cola-Ui基于Vue-Element-Admin框架开发, 参考了Eladmin-Web的部分代码, 项目地址如下

Vue-Element-Admin: <https://gitee.com/panjiachen/vue-element-admin>

Eladmin-web: <https://gitee.com/elunez/eladmin-web>

感谢两个优秀的开源前端框架, 大佬V5!

目录结构

```
1  cola-ui
2  |—build      # 构建配置目录
3  |—mock       # mock数据
4  |—public     # 静态资源
5  |—src
6  |   |—api      # 与后端交互的接口
7  |   |—components # 公共组件
8  |   |—layout    # 页面布局
9  |   |—libs      # 自定义UI
10 |   |—router     # 路由
11 |   |—store      # vuex
12 |   |—styles     # 自定义样式
13 |   |—utils      # axios封装、鉴权封装等
14 |   |—views      # 页面
15 |—nginx.conf    # nginx配置文件
```

环境要求

基础开发环境

- NodeJs v16+
- npm v8+

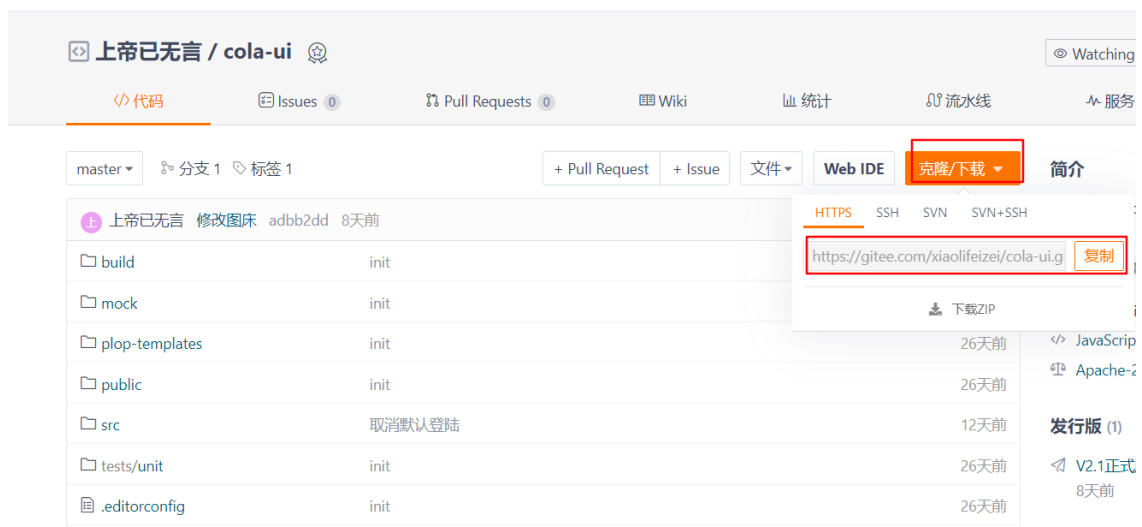
推荐IDE

- WebStorm

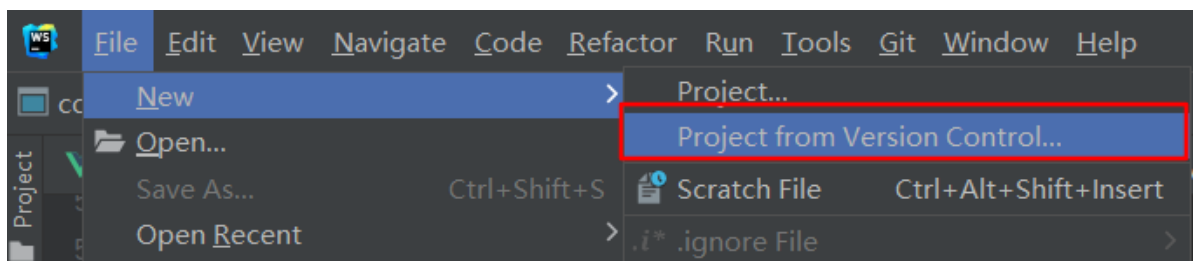
环境准备

导入项目

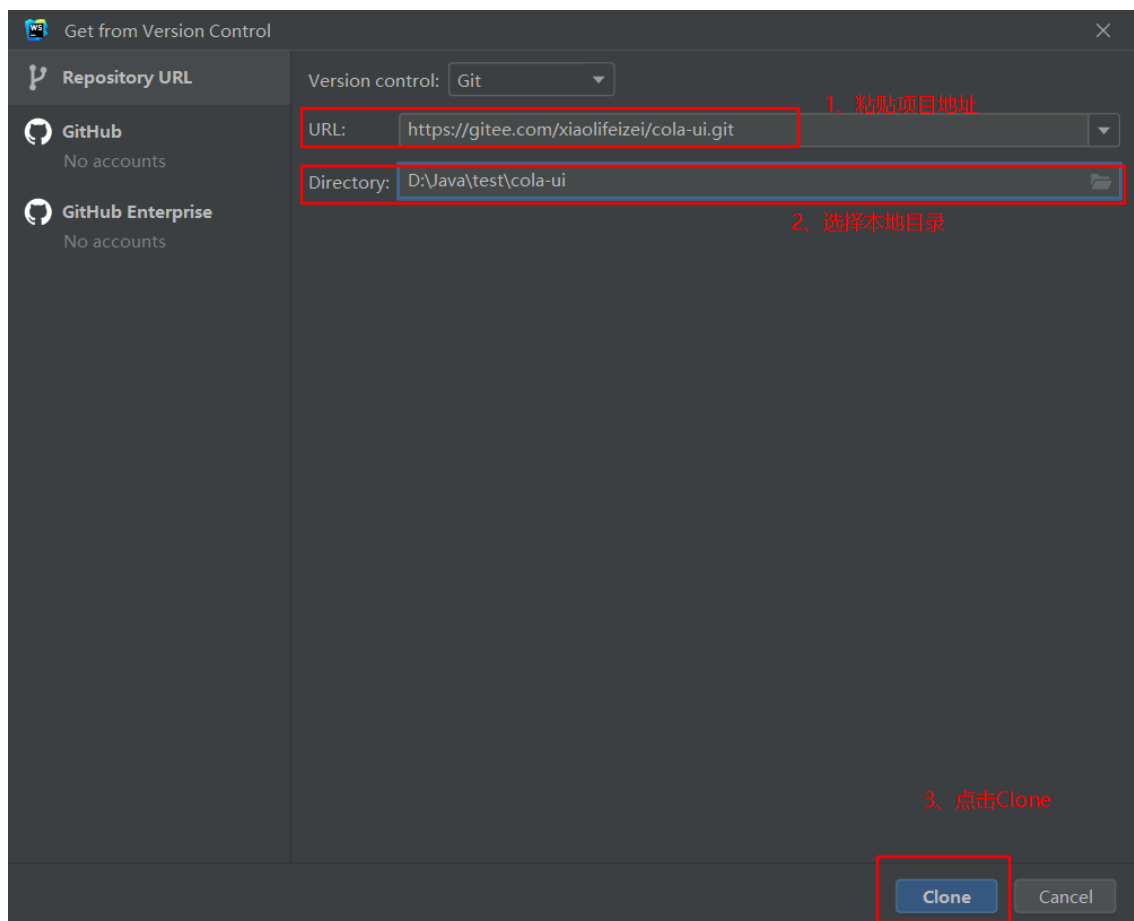
1. 进入cola-ui项目首页<https://gitee.com/xiaolifeize/cola-ui>
2. 复制项目地址



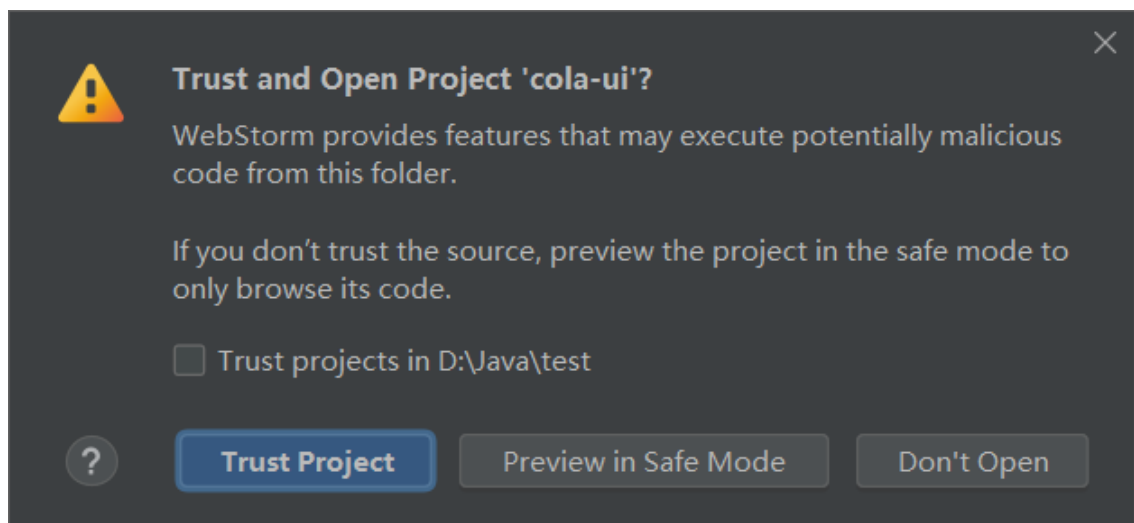
3. 打开WebStorm，依次点击File->New->Project from Version Control...



4. 在弹出的对话框中粘贴复制的项目地址

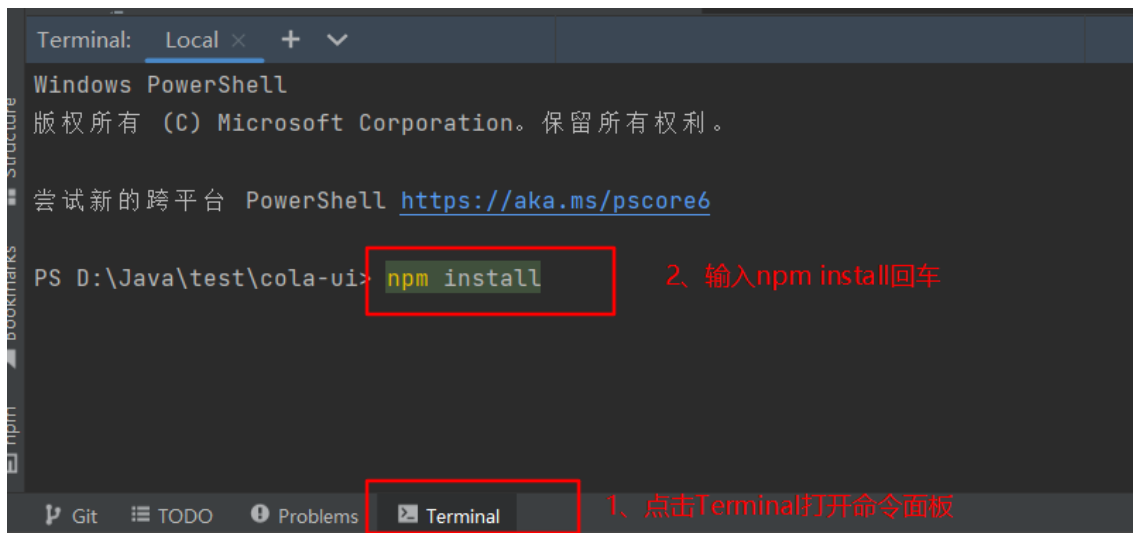


5. IDE可能会弹出对话框，点击“Trust Project”



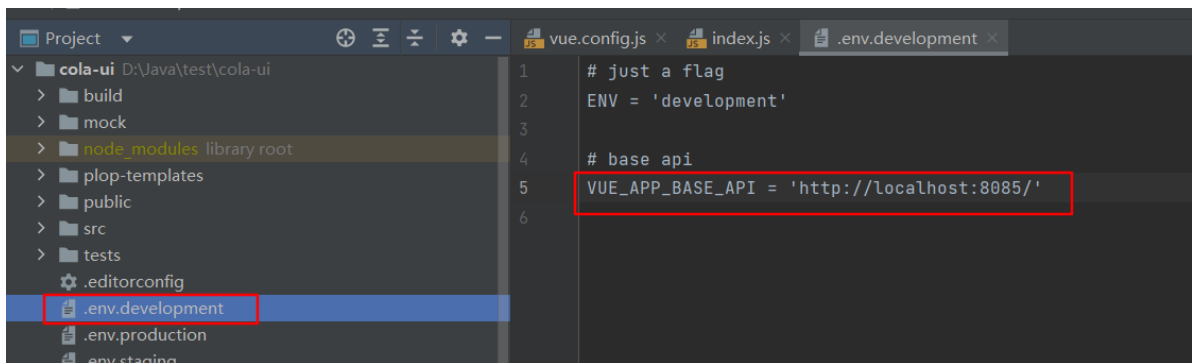
6. 等待代码下载完成

7. 安装依赖

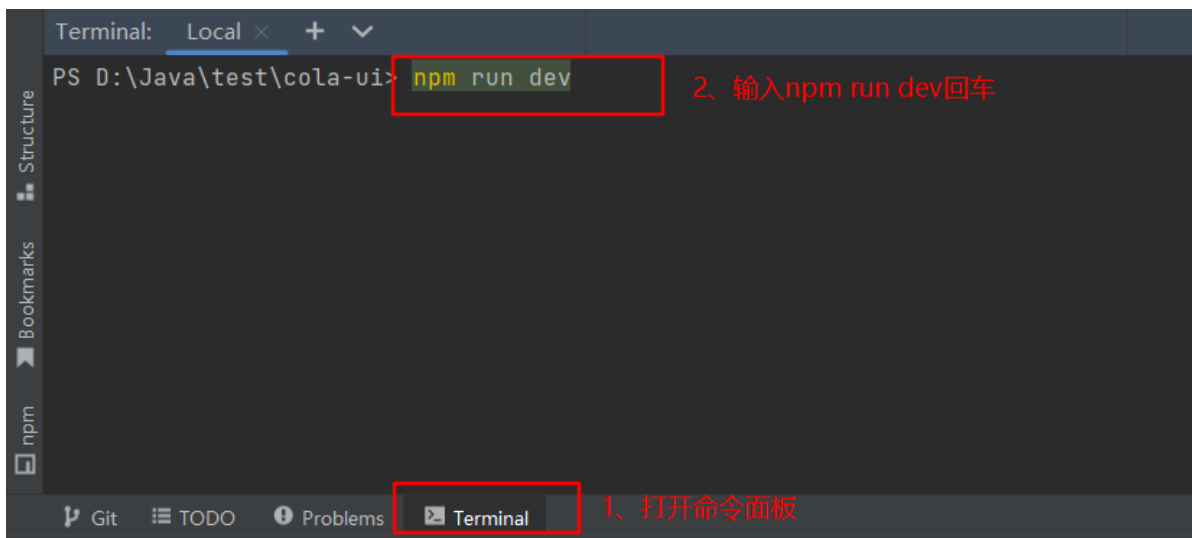


运行项目

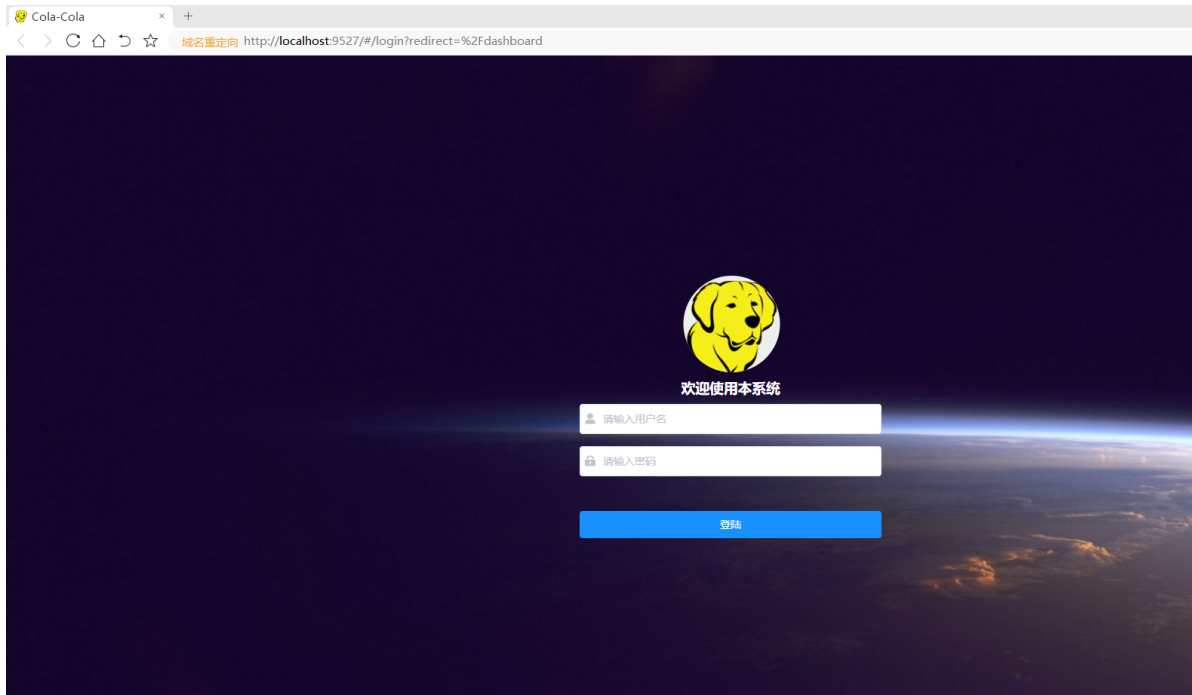
1. 运行前请先启动cola-admin后台服务，请在.env.development文件中修改后台地址及端口，默认为8085端口



2. 输入命令npm run dev启动服务



3. 启动成功后会自动打开浏览器




开发初探

第一个CURD

以后端项目中的学生管理为例，后端具体内容请参考文档：<https://gitee.com/xiaolifeize/cola-admin/wikis/%E5%90%8E%E7%AB%AF%E6%96%87%E6%A1%A3>

添加菜单

进入系统后点击“系统后台管理”--“菜单管理”，再点击新增按钮

 Cola-Admin

首页

订单管理

基础信息管理

系统权限管理

系统运维管理

系统后台管理

通知公告

字典管理

菜单管理

用户管理

机构管理

导航 / 系统后台管理 / 菜单管理

首页 菜单管理 x

请输入菜单名称

请输入菜单编码






请选择菜单类型

查询

新增

修改

删除

	名称	图标	编码	url
<input type="checkbox"/>	> 订单管理		order	/order
<input type="checkbox"/>	> 基础信息管理		basics	/basics
<input type="checkbox"/>	> 系统权限管理		permission	/permission
<input type="checkbox"/>	> 系统运维管理		operation	/operation
<input type="checkbox"/>	> 系统后台管理		system	/system

填写菜单内容后点击确定按钮完成添加

添加菜单



父节点

根节点

* 菜单名称

学生信息管理

图标

el-icon-s-flag

* 菜单编码

studentInfo

* 组件

Layout

* URL

/studentInfo

菜单类型

菜单

打开类型

内置

是否隐藏

否

排序

-

0

+

第一级菜单必填Layout

取消

确定

添加子菜单，在列表选中刚添加的学生信息管理菜单项，再点击新增按钮

请输入菜单名称

请输入菜单编码

请选择菜单类型

查询

重置

+ 新增

修改

删除

	名称	图标	编码	url	组件	菜单
<input type="checkbox"/>	> 订单管理		order	/order	Layout	菜单
<input checked="" type="checkbox"/>	学生信息管理		studentInfo	/studentInfo	Layout	菜单
<input type="checkbox"/>	> 基础信息管理		basics	/basics	Layout	菜单
<input type="checkbox"/>	> 系统权限管理		permission	/permission	Layout	菜单
<input type="checkbox"/>	> 系统运维管理		operation	/operation	Layout	菜单
<input type="checkbox"/>	> 系统后台管理		system	/system	Layout	菜单

填写子菜单信息

添加菜单

×

父节点

学生信息管理

* 菜单名称

学生管理

图标

el-icon-s-flag

* 菜单编码

studentInfoStudent

上级编码

本级编码

* 组件

student_info_student

上级编码加本级编码，单词前用下滑线

* URL

/studentInfo/student

上级url

本级url

菜单类型

菜单

打开类型

内置

是否隐藏

否

排序

-

0

+

取消

确定

最终效果如下

请输入菜单名称

请输入菜单编码

请选择菜单类型

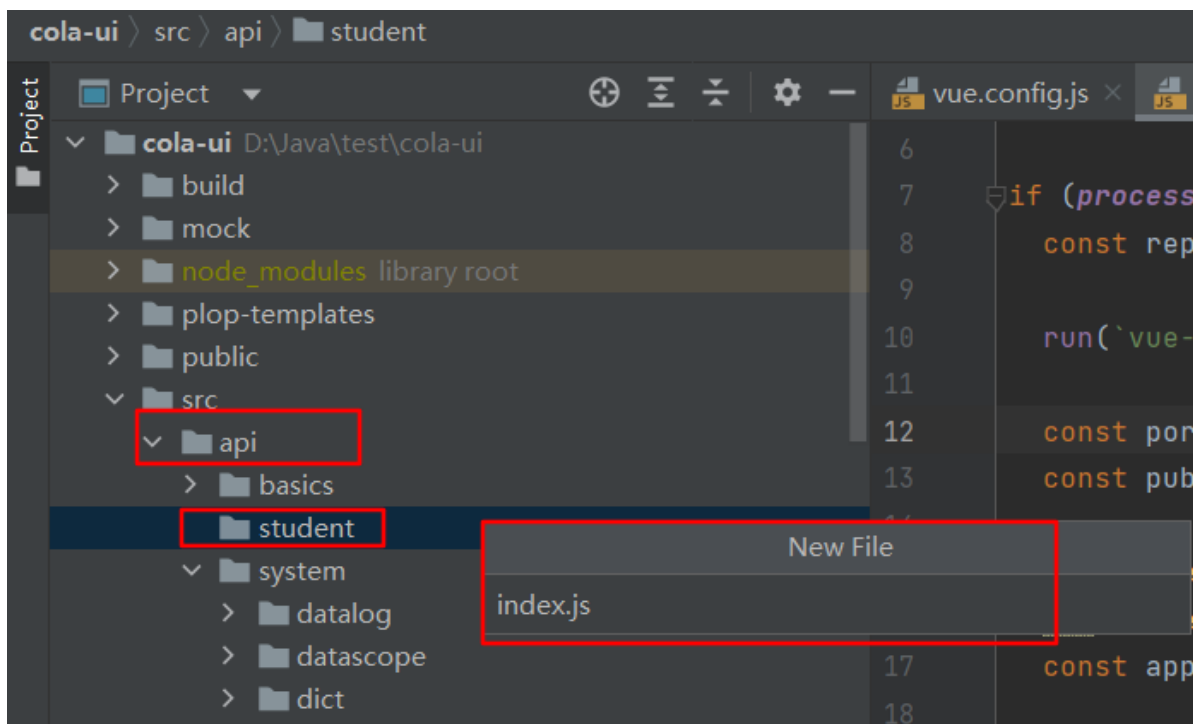
查询

重置

	名称	图标	编码	url	组件	菜单类型	打开方式	是否隐藏	排序
<input type="checkbox"/>	> 订单管理		order	/order	Layout	菜单	内置	否	0
<input type="checkbox"/>	▼ 学生信息管理		studentInfo	/studentInfo	Layout	菜单	内置	否	0
<input type="checkbox"/>	学生管理		studentInfoStudent	/studentInfo/student	student_info_student	菜单	内置	否	0
<input type="checkbox"/>	> 基础信息管理		basics	/basics	Layout	菜单	内置	否	10

编写API

在src\api目录下新建student目录，并添加index.js文件



打开src\api\student\index.js

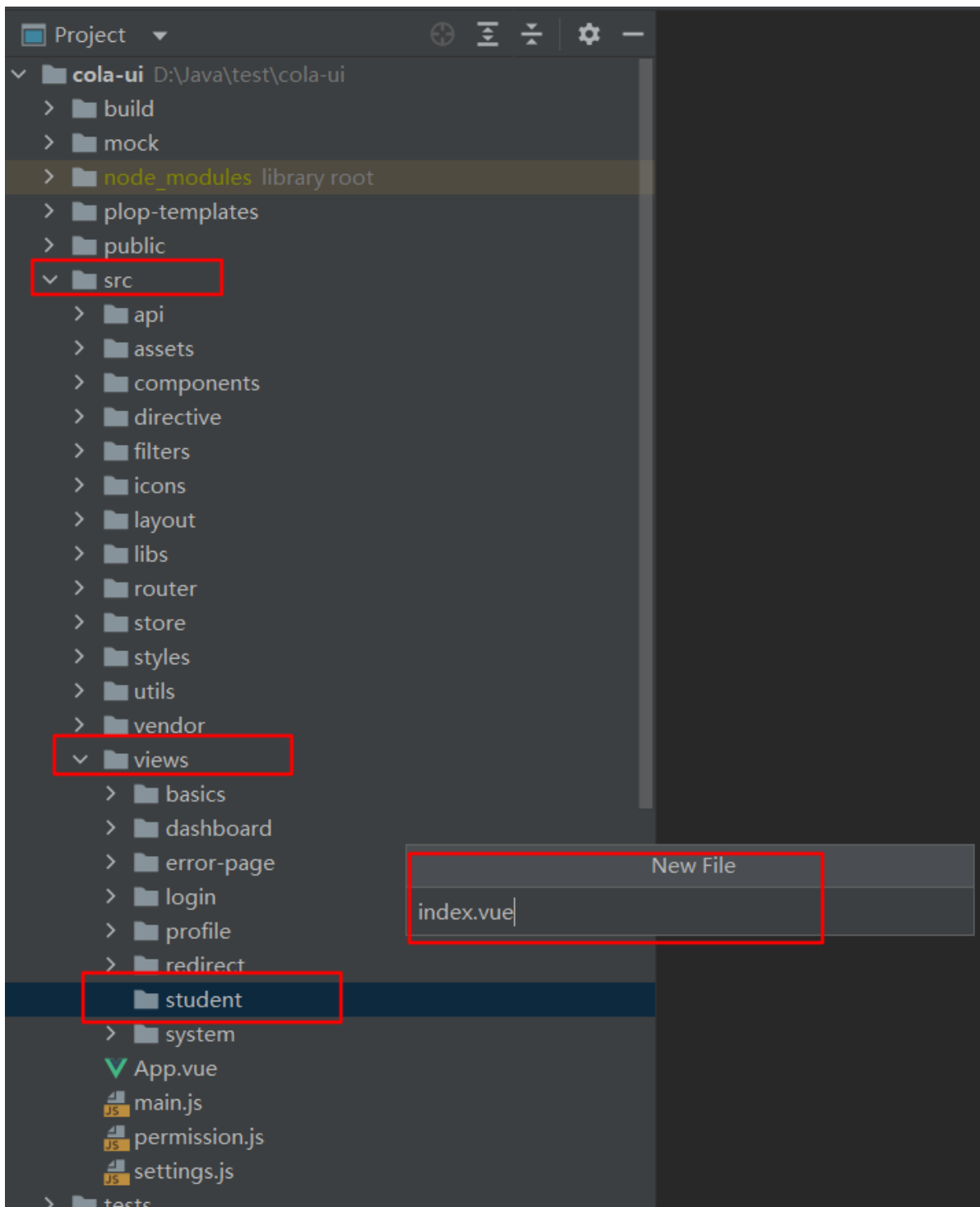
```
1  import request from '@utils/request'
2
3  /**
4   * 添加学生
5   * @param data
6   * @returns {*}
7   */
8  export function addStudent(data) {
9    return request({
10      url: '/student/addStudent',
11      method: 'post',
12      data
13    })
14  }
15
16  /**
17   * 修改学生信息
18   * @param data
19   * @returns {*}
20   */
21  export function editStudent(data) {
22    return request({
23      url: '/student/editStudent',
24      method: 'post',
25      data
26    })
27  }
28
29  /**
30   * 删除学生信息
31   * @param data
32   * @returns {*}
33   */
34  export function deleteStudent(data) {
35    return request({
```



```
36     url: '/student/deleteStudent',
37     method: 'post',
38     data
39   })
40 }
41
42 /**
43  * 分页查询
44  * @param data
45  * @returns {*}
46  */
47 export function getStudentPage(data) {
48   return request({
49     url: '/student/getStudentPage',
50     method: 'post',
51     data
52   })
53 }
54
```

编写主页面

打开src\views目录，新建student目录，然后再新建index.vue



index.vue为主页面，主要功能是查询列表的展示以及查询、修改、删除等组件的容器。复制如下代码到index.vue中

```
1 <template>
2   <!-- 主页面容器-->
3   <div class="app-container">
4     <!-- 查询容器-->
5     <table-query></table-query>
6     <!-- 表格工具条 -->
7     <table-header></table-header>
8     <!-- 表格 -->
9     <el-table
10      ref="indexTable"
11      v-loading="tableLoading"
12      :data="list"
13      fit
```

```

14     border
15     highlight-current-row
16     style="width: 100%"
17     class="indexTable"
18     size="mini"
19     @selection-change="handleSelectionChange"
20   >
21     <el-table-column align="center" type="selection" width="60" />
22     <!-- 显示字段 -->
23     <el-table-column label="id" prop="id" width="80" />
24     <el-table-column label="姓名" prop="name" width="200" />
25     <el-table-column label="年龄" prop="age" width="80" />
26     <el-table-column label="性别" prop="sex" width="80" />
27     <el-table-column label="地址" prop="address" />
28     <el-table-column label="创建时间" prop="createTime" width="150px" />
29     <el-table-column label="操作" width="120px" align="center">
30       <template v-slot="scope">
31         <div style="display: flex;">
32           <!-- 修改按钮 -->
33           <el-button type="primary" icon="el-icon-edit" size="mini"
@click="toEdit(scope.row)" />
34           <!-- 删除按钮 -->
35           <el-button type="danger" icon="el-icon-delete" size="mini"
@click="toDelete(scope.row)" />
36         </div>
37       </template>
38     </el-table-column>
39   </el-table>
40 </div>
41 </template>
42
43 <script>
44
45 import { getStudentPage } from '@/api/student'
46
47 export default {
48   name: 'Student',
49   data() {
50     return {
51       tableLoading: false,
52       list: []
53     }
54   },
55   mounted() {
56     // 增加刷新方法
57     this.$refs.indexTable.refresh = this.getTableData
58     // 页面加载完成后自动查询数据
59     this.getTableData()
60   },
61   methods: {
62     // 控制只能单选
63     handleSelectionChange(val) {
64       if (val.length > 1) {
65         this.$refs.indexTable.clearSelection()
66         this.$refs.indexTable.toggleRowSelection(val.pop())
67       }
68     },
69     getTableData() {

```

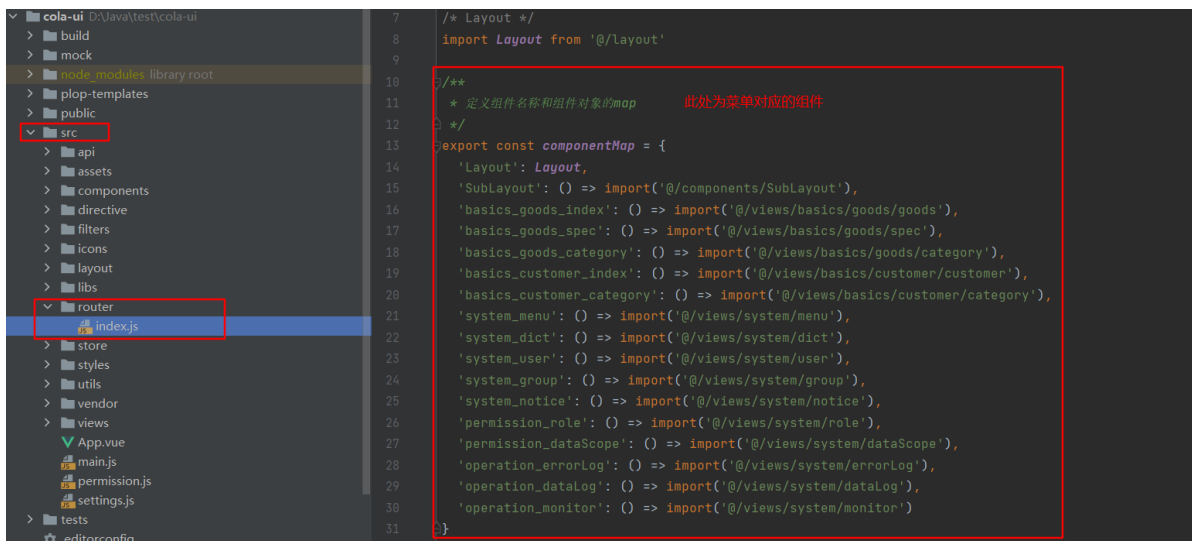
```

70     this.tableLoading = true
71     getStudentPage().then(response => {
72         if (response.data.page) {
73             this.list = response.data.page.records ?
response.data.page.records : []
74         }
75         this.tableLoading = false
76     }).catch(() => {
77         this.tableLoading = false
78     })
79 },
80 toEdit(row) {
81 },
82 toDelete(row) {
83 }
84 }
85 }
86 </script>
87
88 <style scoped>
89 /* 取消表头上的选择框 */
90 .indexTable /deep/ .el-table-column--selection.is-leaf .el-checkbox {
91     display: none;
92 }
93 </style>
94

```

添加路由

打开src\router\index.js



添加学生管理路由映射

```
export const componentMap = {
  'Layout': Layout,
  'SubLayout': () => import('@components/SubLayout'),
  'basics_goods_index': () => import('@views/basics/goods/goods'),
  'basics_goods_spec': () => import('@views/basics/goods/spec'),
  'basics_goods_category': () => import('@views/basics/goods/category'),
  'basics_customer_index': () => import('@views/basics/customer/customer'),
  'basics_customer_category': () => import('@views/basics/customer/category'),
  'system_menu': () => import('@views/system/menu'),
  'system_dict': () => import('@views/system/dict'),
  'system_user': () => import('@views/system/user'),
  'system_group': () => import('@views/system/group'),
  'system_notice': () => import('@views/system/notice'),
  'permission_role': () => import('@views/system/role'),
  'permission_dataScope': () => import('@views/system/dataScope'),
  'operation_errorLog': () => import('@views/system/errorLog'),
  'operation_dataLog': () => import('@views/system/dataLog'),
  'operation_monitor': () => import('@views/system/monitor'),
  'student_info_student': () => import('@views/student')
```

刷新页面后可以看到新增加的菜单，效果如下

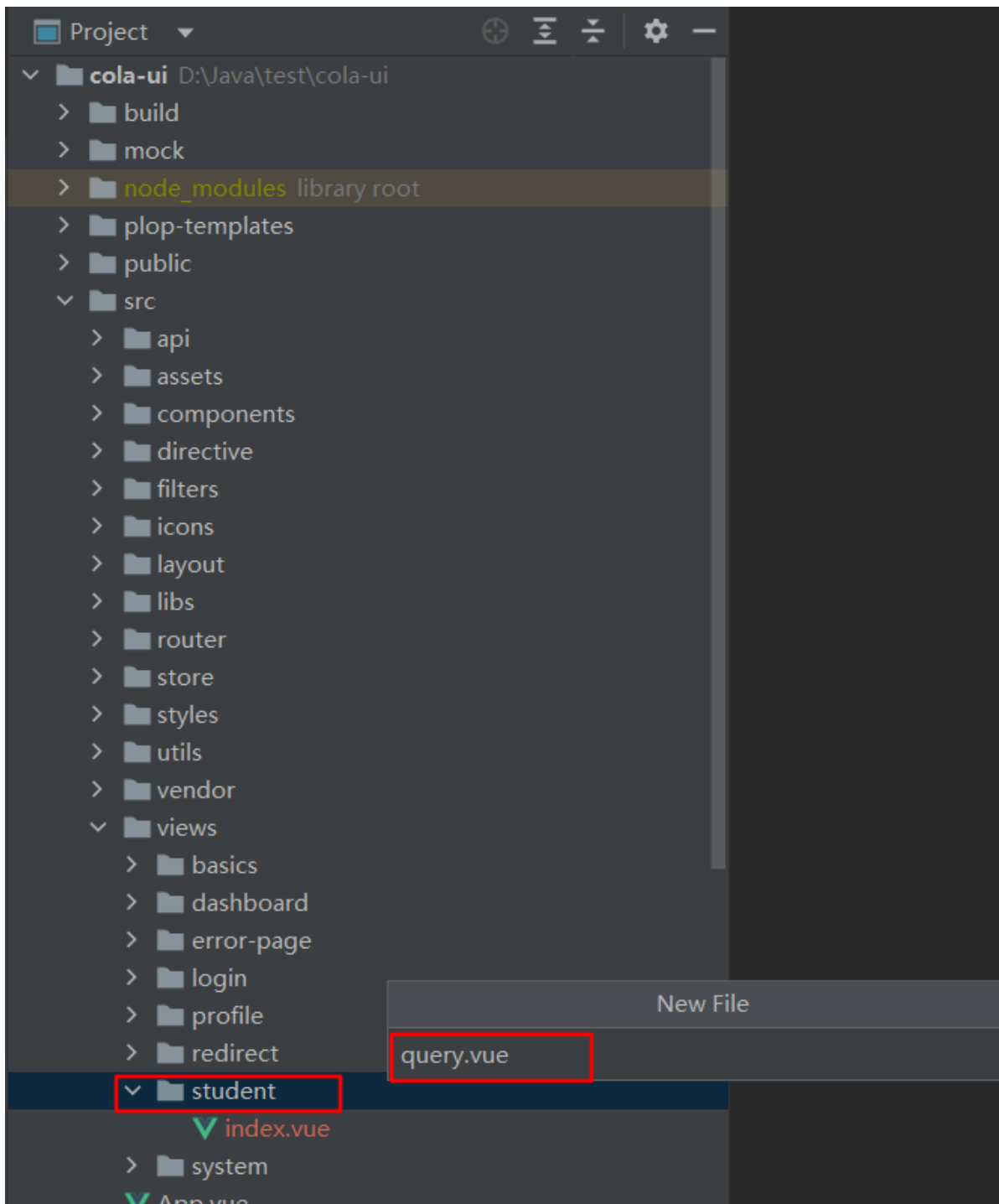


The screenshot shows the 'Cola-Admin' web application. The sidebar on the left contains navigation links: 首页 (Home), 订单管理 (Order Management), 学生信息管理 (Student Information Management), 学生管理 (Student Management), 基础信息管理 (Basic Information Management), and 系统权限管理 (System Permissions Management). The main content area shows the 'Student Management' page with a table of students.

	id	姓名	年龄	性别	地址	创建时间	操作
<input type="checkbox"/>	2	张三	16	1		2022-07-22 15:38:15	编辑 删除
<input type="checkbox"/>	3	王五	19	0		2022-07-22 15:38:15	编辑 删除
<input type="checkbox"/>	4	赵六	20	1		2022-07-22 15:38:16	编辑 删除
<input type="checkbox"/>	5	张生	19	0		2022-07-22 17:31:11	编辑 删除

添加查询条件

1. 在src\views\student目录下新建query.vue



2. 编写查询条件代码

```
1 <template>
2   <!-- 查询条件容器 -->
3   <div class="query-content">
4     <!-- 表单 -->
5     <el-form ref="queryForm" :model="query.data" :inline="true">
6       <!-- 查询条件 -->
7       <el-form-item prop="name">
8         <el-input v-model="query.data.name" :clearable="true"
placeholder="请输入学生姓名" />
9       </el-form-item>
10      <!-- 查询条件 -->
11      <el-form-item prop="sex">
12        <el-select v-model="query.data.sex" :clearable="true"
placeholder="请选择性别">
13          <el-option label="女" :value="0" />
```

```
14     <el-option label="男" :value="1" />
15   </el-select>
16 </el-form-item>
17   <!-- 查询按钮 -->
18   <el-button type="primary" icon="el-icon-search" @click="doQuery"> 查
19 询 </el-button>
20   <el-button type="success" icon="el-icon-refresh-right"
21 @click="doReset"> 重 置 </el-button>
22 </el-form>
23 </div>
24 </template>
25
26 <script>
27 export default {
28   name: 'StudentQuery',
29   props: {
30     // 主页面传过来的查询对象
31     query: {
32       type: Object,
33       require: true,
34       default() {
35         return {
36           data: {
37             name: '',
38             sex: ''
39           },
40           currentPage: 1
41         }
42       }
43     },
44     data() {
45       return {
46         // 主页面table对象
47         table: {}
48       }
49     },
50     mounted() {
51       // 获取父页面table对象
52       this.table = this.$parent.$parent.$children.find(vc => vc.tableId)
53     },
54     methods: {
55       doQuery() {
56         // 调用父页面刷新方法执行查询操作
57         this.table.refresh()
58       },
59       doReset() {
60         // 重置时清空表单信息
61         this.$refs.queryForm.resetFields()
62         // 重置时重新刷新
63         this.table.refresh()
64       }
65     }
66   }
67 }
68 </script>
69
70 <style scoped>
```

```
70 | </style>
71 |
```

3. 在主页面中添加查询条件组件和查询条件对象

```
5 | import { getStudentPage } from '@api/student'
6 | import Query from './query' 引入查询条件组件
7 |
8 | export default {
9 |   name: 'Student',
10 |   components: { Query }, 引入查询组件
11 |   data() {
12 |     return {
13 |       tableLoading: false,
14 |       list: [],
15 |       query: { 查询条件对象
16 |         conditions: [ 表示data中的name字段使用like查询
17 |           { name: 'name', keyword: 'like' }
18 |         ],
19 |         data: {
20 |           name: '',
21 |           sex: '' | 默认是=号查询
22 |         },
23 |         total: 0, 总条数
24 |         currentPage: 0, 当前第几页
25 |         pageSize: 20 每页条数
26 |       } 分页属性
27 |     }
28 |   },
29 | }
```

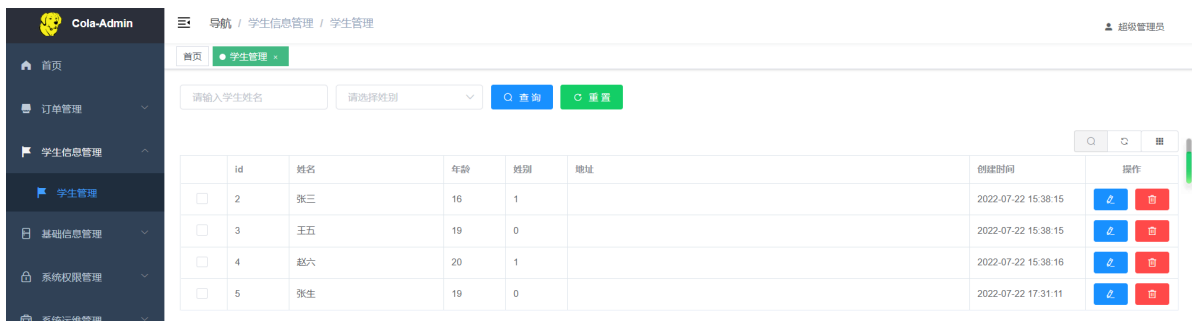
4. 页面中添加引用

```
✓ student\index.vue x
1 | <template>
2 |   <!-- 主页面容器-->
3 |   <div class="app-container">  引用查询组件
4 |     <!-- 查询容器-->
5 |     <table-query> 传递查询条件
6 |       <query :query.sync="query" />
7 |     </table-query>
8 |     <!-- 表格工具条容器 -->
9 |     <table-header></table-header>
10 |    <!-- 表格 -->
11 |    <el-table
```

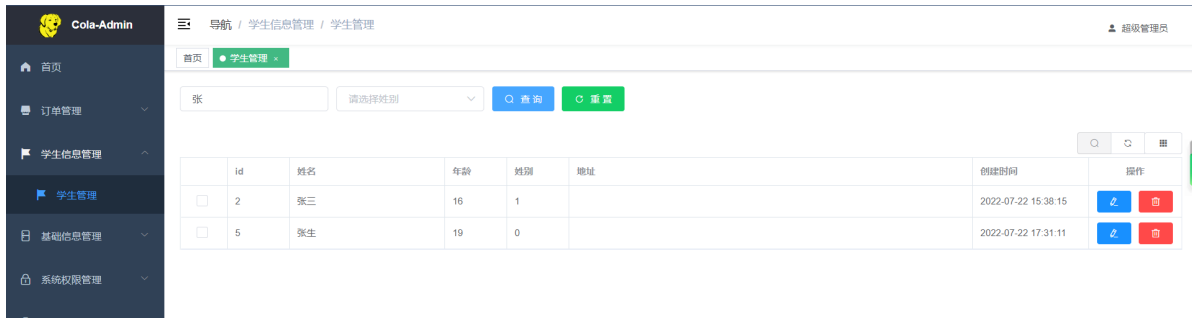
5. 修改查询方法，将查询条件对象传入

```
getTableData() {
  this.tableLoading = true
  getStudentPage(this.query).then(response => {
    if (response.data.page) {
      this.list = response.data.page.records ? response.data.page.records : []
    }
    this.tableLoading = false
  }).catch(() => {
    this.tableLoading = false
  })
}
```

6. 最终效果

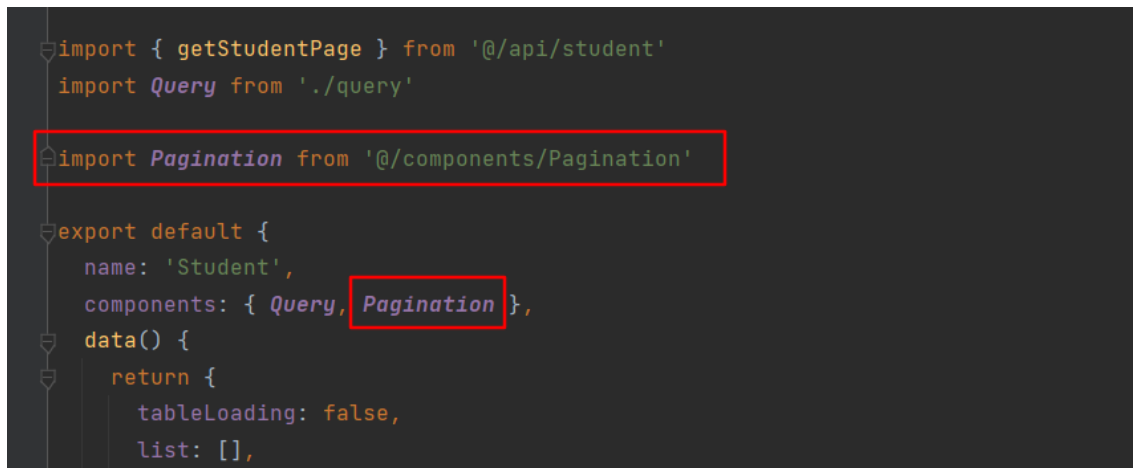


7. 测试查询

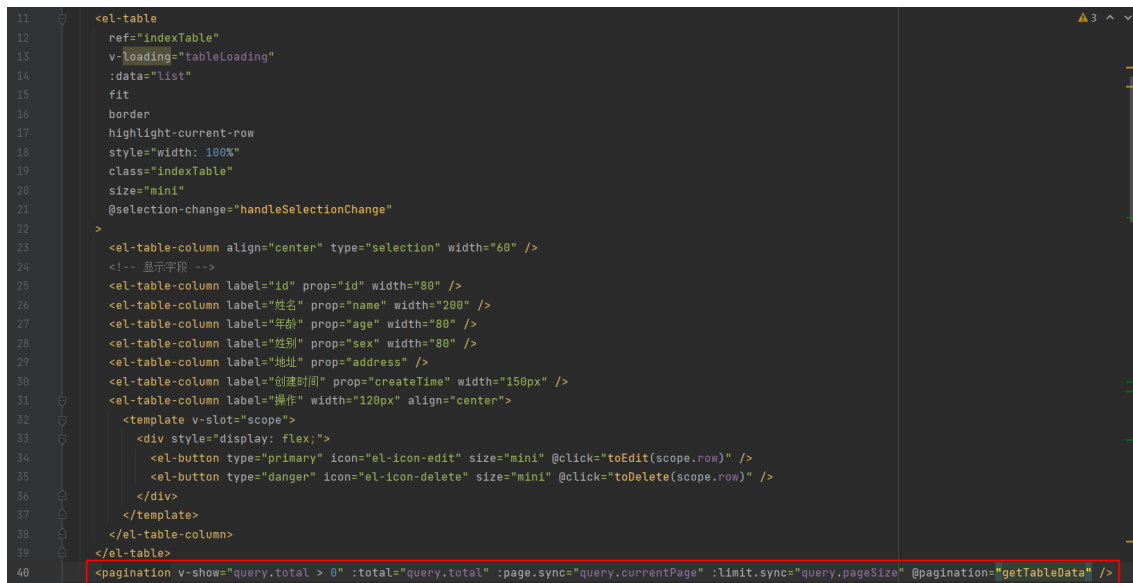


添加分页

1. 引入分页组件



2. 添加分页组件



3. 修改查询方法

```
getTableData() {
  this.tableLoading = true
  getStudentPage(this.query).then(response => {
    if (response.data.page) {
      this.list = response.data.page.records ? response.data.page.records : []
      this.query.total = response.data.page.total
    } else {
      this.query.total = 0
    }
    this.tableLoading = false
  }).catch(() => {
    this.tableLoading = false
    this.query.total = 0
  })
},
},
```

4. 最终效果

Cola-Admin

首页

订单管理

学生信息管理

学生管理

基础信息管理

系统权限管理

系统运维管理

导航 / 学生信息管理 / 学生管理

高级管理员

学生管理

请输入学生姓名

请选择性别

查询

重置

	id	姓名	年龄	性别	地址	创建时间	操作
<input type="checkbox"/>	2	张三	16	1		2022-07-22 15:38:15	<div>编辑删除</div>
<input type="checkbox"/>	3	王五	19	0		2022-07-22 15:38:15	<div>编辑删除</div>
<input type="checkbox"/>	4	赵六	20	1		2022-07-22 15:38:16	<div>编辑删除</div>
<input type="checkbox"/>	5	张生	19	0		2022-07-22 17:31:11	<div>编辑删除</div>

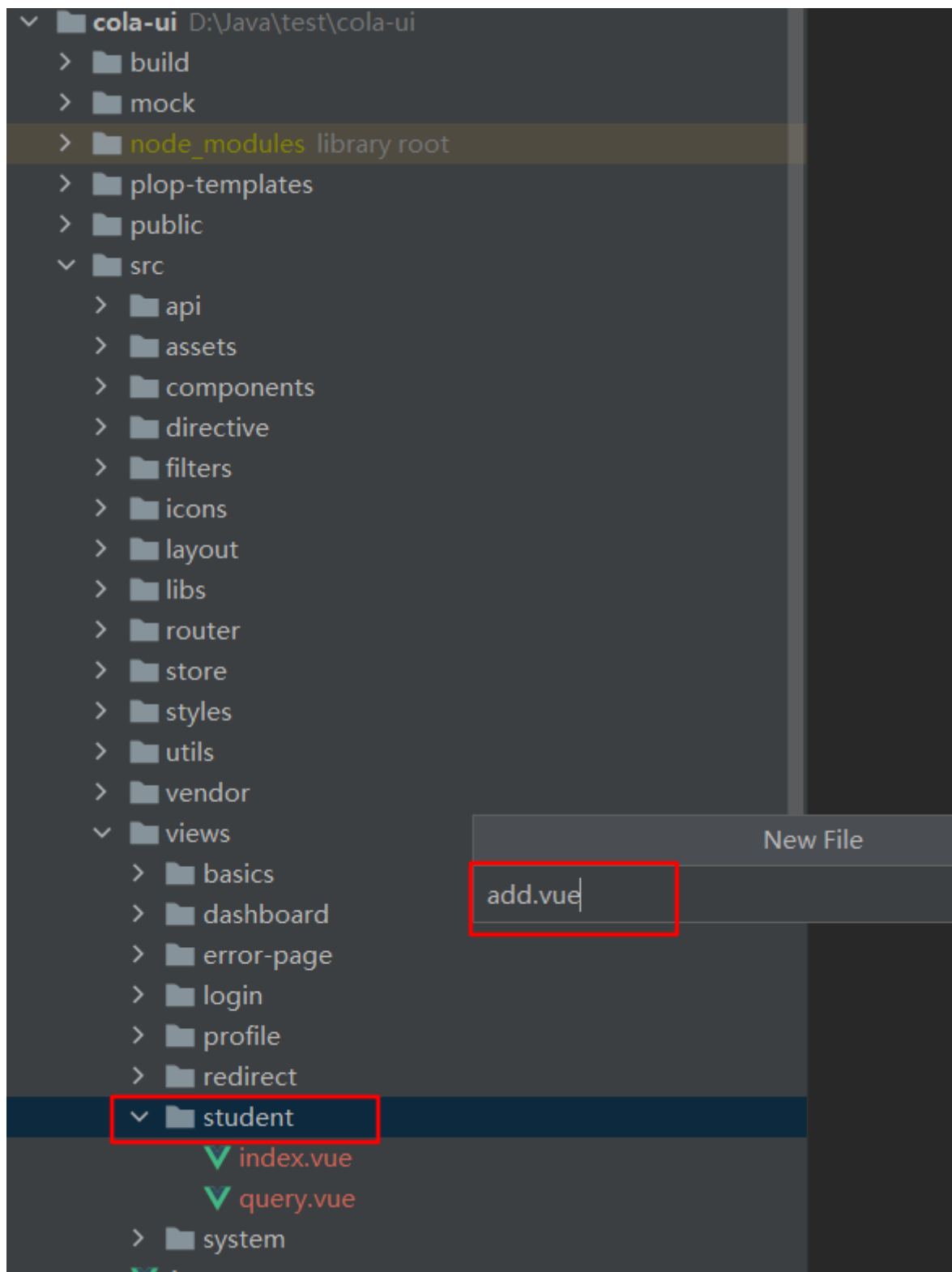
共 4 条

<1>

20条/页

新增

1. 创建新增页面



2. 编写新增代码

```
1 <template>
2   <!-- 顶层容器 -->
3   <div>
4     <!-- 新增按钮 -->
5     <el-button
6       class="filter-item"
7       size="mini"
8       type="primary"
9       icon="el-icon-plus"
10      @click="showAdd"
11    >
```

```

12     新增
13     </el-button>
14     <!-- 新增对话框 -->
15     <el-dialog :append-to-body="true" title="添加学生"
:visible.sync="dialogShow" width="600px" @close="cancelAdd">
16         <!-- form表单 -->
17         <el-form ref="addForm" :rules="rules" :model="form" label-
width="80px">
18             <el-form-item label="姓名" prop="name">
19                 <el-input v-model="form.name" autocomplete="off"
maxlength="20" />
20             </el-form-item>
21             <el-form-item label="年龄" prop="age">
22                 <el-input-number v-model.number="form.age" controls-
position="right" step-strictly style="width: 140px" />
23             </el-form-item>
24             <el-form-item label="性别" prop="sex">
25                 <el-select v-model="form.sex" placeholder="请选择性别"
style="width: 140px">
26                     <el-option label="女" :value="0" />
27                     <el-option label="男" :value="1" />
28                 </el-select>
29             </el-form-item>
30             <el-form-item label="地址" prop="address">
31                 <el-input v-model="form.address" autocomplete="off"
maxlength="100" />
32             </el-form-item>
33         </el-form>
34         <div slot="footer" class="dialog-footer">
35             <el-button @click="dialogShow = false">取 消</el-button>
36             <el-button type="primary" :loading="addLoading"
@click="doAdd">确 定</el-button>
37         </div>
38     </el-dialog>
39 </div>
40 </template>
41
42 <script>
43 import { addStudent } from '@api/student'
44
45 export default {
46     name: 'StudentAdd',
47     data() {
48         return {
49             table: {},
50             dialogShow: false,
51             addLoading: false,
52             form: {
53                 name: '',
54                 age: 11,
55                 sex: 0,
56                 address: ''
57             },
58             rules: {
59                 name: [{ required: true, message: '请输入学生姓名', trigger:
'blur' }],
60                 age: [{ required: true, message: '年龄不能为空' }, { type:
'number', message: '年龄必须为数字值' }]

```

```

61     }
62   }
63 },
64   mounted() {
65     this.table = this.$parent.$parent.$children.find(vc => vc &&
vc.tableId)
66   },
67   methods: {
68     showAdd() {
69       this.dialogShow = true
70     },
71     cancelAdd() {
72       this.$refs.addForm.resetFields()
73     },
74     doAdd() {
75       this.addLoading = true
76       this.$refs.addForm.validate(valid => {
77         if (valid) {
78           addStudent(this.form).then(response => {
79             this.$refs.addForm.resetFields()
80             this.dialogShow = false
81             this.addLoading = false
82             this.table.refresh()
83             this.$notify.success({
84               title: '成功',
85               message: '添加成功'
86             })
87           }).catch(() => {
88             this.addLoading = false
89           })
90         }
91         this.addLoading = false
92       })
93     }
94   }
95 }
96 </script>
97
98 <style scoped>
99
100 </style>

```

3. 引入新增组件

在index.vue中引入组件

```
student\index.vue
40     </el-table-column>
41   </el-table>
42   <pagination v-show="query.total > 0" :total="query.total" :page.sync="query.currentPage"
43 </div>
44 </template>
45
46 <script>
47
48 import { getStudentPage } from '@api/student'
49 import Query from './query'
50 import StudentAdd from './add'
51
52 import Pagination from '@components/Pagination'
53
54 export default {
55   name: 'Student',
56   components: { Query, Pagination, StudentAdd },
57   data() {
58     return {
59       tableLoading: false,
60       list: [],
```

4. 添加新增组件

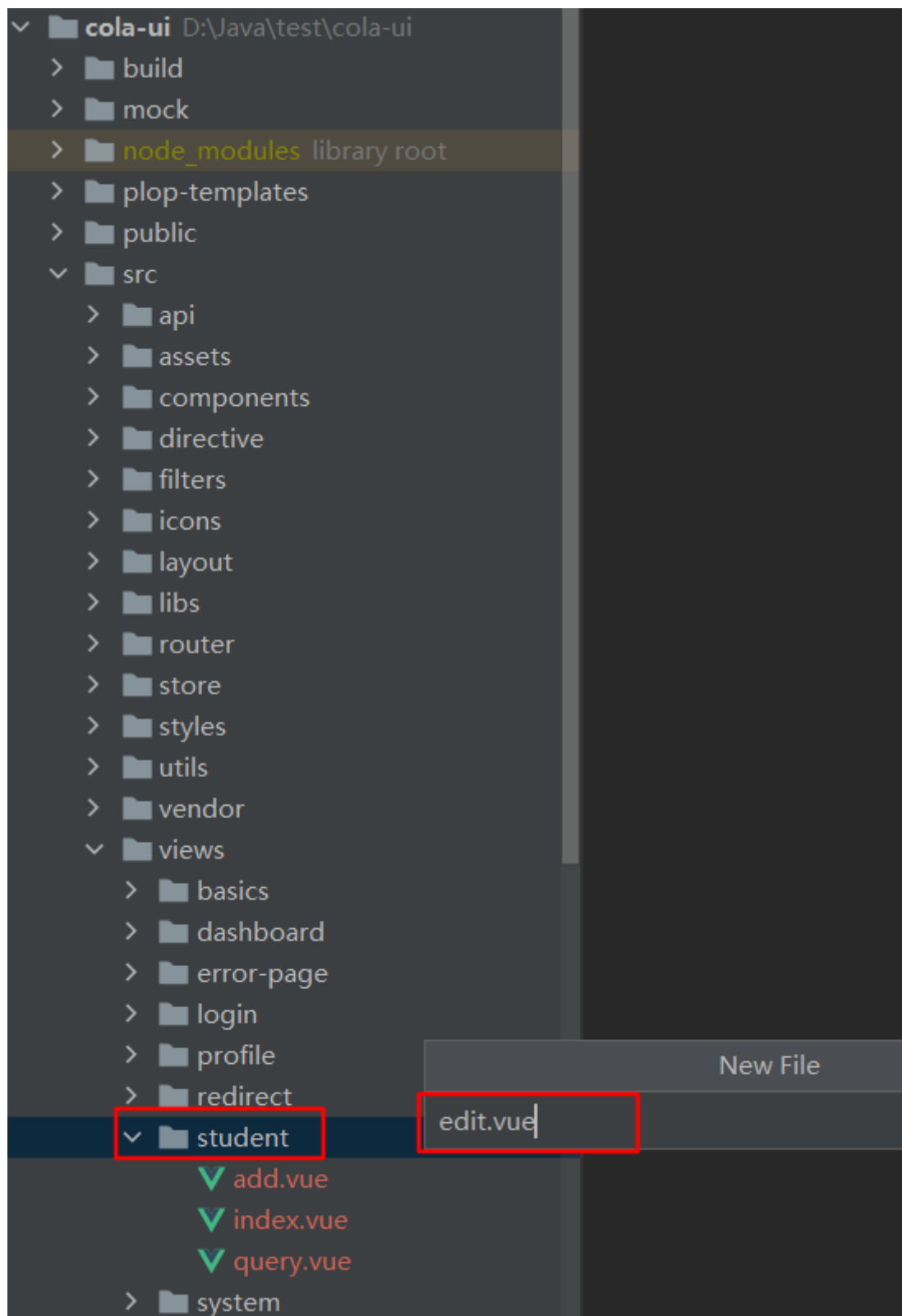
```
student\index.vue
1 <template>
2   <!-- 主页面容器-->
3   <div class="app-container">
4     <!-- 查询容器-->
5     <table-query>
6       <query :query.sync="query" />
7     </table-query>
8     <!-- 表格工具条容器 -->
9     <table-header>
10      <student-add />
11    </table-header>
12    <!-- 表格 -->
13    <el-table
```

5. 最终效果



修改

1. 新建修改页面



2. 编写修改页面代码

```
1 <template>
2   <!-- 顶层容器 -->
3   <div>
4     <!-- 修改按钮 -->
5     <el-button
6       class="filter-item"
7       size="mini"
8       type="success"
9       icon="el-icon-edit"
10      :disabled="editDisabled"
11      @click="showEdit(table.selection[0])"
```

```

12     >
13     修改
14   </el-button>
15   <!-- 弹出修改对话框 -->
16   <el-dialog :append-to-body="true" title="修改学生"
:visible.sync="dialogShow" width="600px" @close="cancelEdit">
17     <!-- 表单 -->
18     <el-form ref="editForm" :rules="rules" :model="form" label-
width="80px">
19       <el-form-item label="姓名" prop="name">
20         <el-input v-model="form.name" autocomplete="off"
maxlength="20" />
21       </el-form-item>
22       <el-form-item label="年龄" prop="age">
23         <el-input-number v-model.number="form.age" controls-
position="right" step-strictly style="width: 140px" />
24       </el-form-item>
25       <el-form-item label="性别" prop="sex">
26         <el-select v-model="form.sex" placeholder="请选择性别"
style="width: 140px">
27           <el-option label="女" :value="0" />
28           <el-option label="男" :value="1" />
29         </el-select>
30       </el-form-item>
31       <el-form-item label="地址" prop="address">
32         <el-input v-model="form.address" autocomplete="off"
maxlength="100" />
33       </el-form-item>
34     </el-form>
35     <div slot="footer" class="dialog-footer">
36       <el-button @click="dialogShow = false">取 消</el-button>
37       <el-button type="primary" :loading="editLoading"
@click="doEdit">修 改</el-button>
38     </div>
39   </el-dialog>
40 </div>
41 </template>
42
43 <script>
44 import { editStudent } from '@api/student'
45
46 export default {
47   name: 'StudentEdit',
48   data() {
49     return {
50       table: {},
51       editDisabled: true,
52       dialogShow: false,
53       editLoading: false,
54       form: {
55         name: '',
56         age: 0,
57         sex: 0,
58         address: ''
59       },
60       rules: {
61         name: [{ required: true, message: '请输入学生姓名', trigger:
'blur' }],

```



```

62         age: [{ required: true, message: '年龄不能为空' }, { type:
'number', message: '年龄必须为数字值' }]
63     }
64 }
65 },
66 watch: {
67     'table.selection'() {
68         // 监控是否选中记录, 如果选中则允许点击
69         this.editDisabled = this.table.selection.length !== 1
70     }
71 },
72 mounted() {
73     // 获取index.vue中的table对象
74     this.table = this.$parent.$parent.$children.find(vc => vc &&
vc.tableId)
75 },
76 methods: {
77     cancelEdit() {
78         this.dialogShow = false
79         this.$refs.editForm.resetFields()
80     },
81     showEdit(row) {
82         // 将选择的记录赋值给当前表单
83         this.form = { ...row }
84         // 显示修改对话框
85         this.dialogShow = true
86     },
87     doEdit() {
88         this.editLoading = true
89         this.$refs.editForm.validate(valid => {
90             if (valid) {
91                 // 修改数据
92                 editStudent(this.form).then(() => {
93                     this.$refs.editForm.resetFields()
94                     this.editLoading = false
95                     this.dialogShow = false
96                     // 刷新主页面列表
97                     this.table.refresh()
98                     // 提示信息
99                     this.$notify.success({
100                         title: '成功',
101                         message: '修改成功'
102                     })
103                 }).catch(() => {
104                     this.editLoading = false
105                 })
106             } else {
107                 this.editLoading = false
108             }
109         })
110     }
111 }
112 }
113 </script>
114
115 <style scoped>
116
117 </style>

```

3. index.vue中引用修改页面

```
43 <pagination v-show="query.total > 0" :total="query.total" :page.sync="query.current
44 </div>
45 </template>
46
47 <script>
48
49 import { getStudentPage } from '@api/student'
50 import Query from './query'
51 import StudentAdd from './add'
52 import StudentEdit from './edit'
53
54 import Pagination from '@components/Pagination'
55
56 export default {
57   name: 'Student',
58   components: { Query, Pagination, StudentAdd, StudentEdit },
59   data() {
60     return {
61       tableLoading: false,
62       list: [],
```

4. 添加修改页面

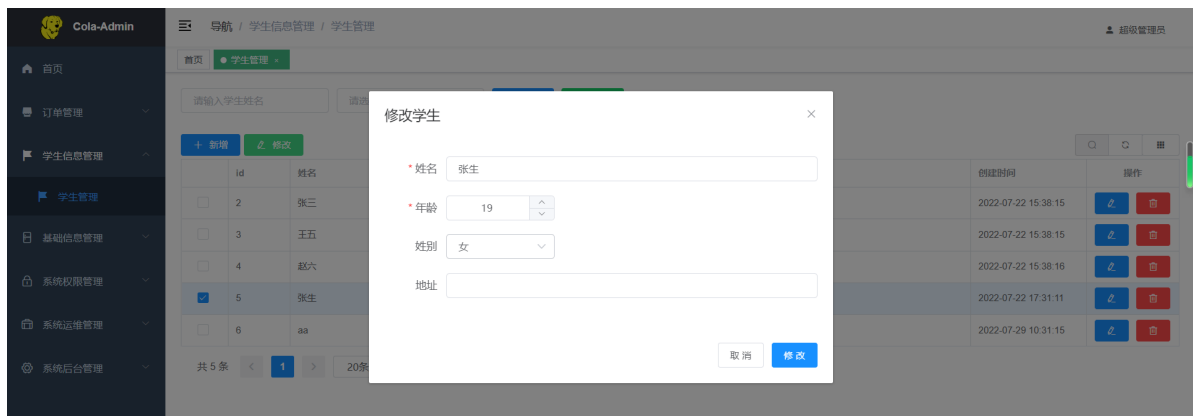
```
1 <template>
2   <!-- 主页面容器-->
3   <div class="app-container">
4     <!-- 查询容器-->
5     <table-query>
6       <query :query.sync="query" />
7     </table-query>
8     <!-- 表格工具条容器 -->
9     <table-header>
10      <student-add />
11      <student-edit ref="btnEdit" />
12    </table-header>
13    <!-- 表格 -->
14    <el-table
```

ref="btnEdit"用于每一行的修改

5. 修改index.vue中的代码

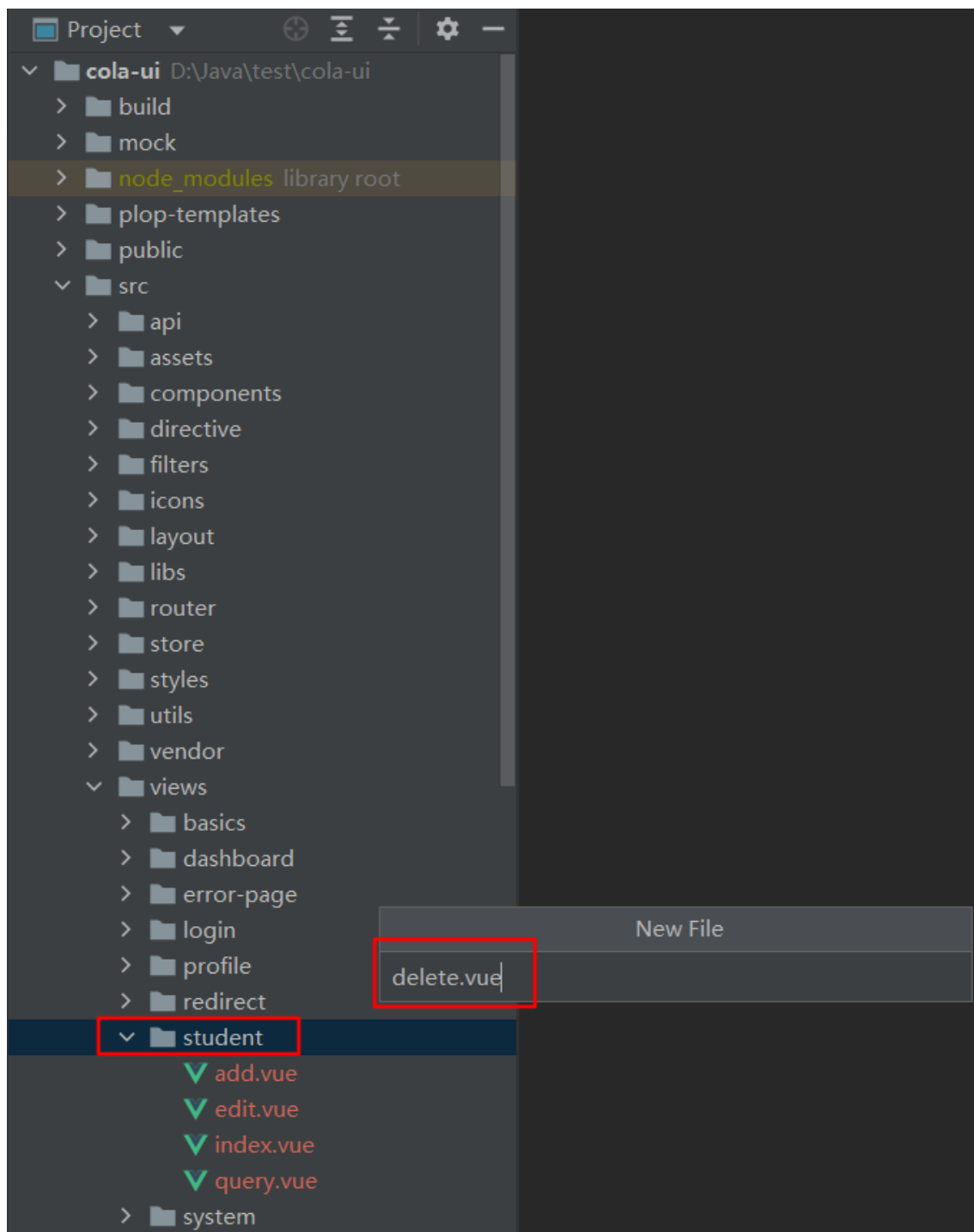
```
edit.vue x index.vue x
90 },
91 getTableData() {
92   this.tableLoading = true
93   getStudentPage(this.query).then(response => {
94     if (response.data.page) {
95       this.list = response.data.page.records ? response.data.page.records : []
96       this.query.total = response.data.page.total
97     } else {
98       this.query.total = 0
99     }
100    this.tableLoading = false
101  }).catch(() => {
102    this.tableLoading = false
103    this.query.total = 0
104  })
105 },
106 toEdit(row) {
107   this.$refs.btnEdit.showEdit(row)
108 },
109 toDelete(row) {}
110 }
111 }
112 }
```

6. 最终效果



删除

1. 新建删除页面



2. 编写删除代码

```
1 <template>
2   <div>
3     <el-button
4       class="filter-item"
5       size="mini"
6       type="danger"
7       icon="el-icon-delete"
8       :disabled="delDisabled"
9       @click="doDelete(table.selection[0])"
10    >
11      删除
12    </el-button>
13  </div>
14 </template>
15
```

```

16 <script>
17 import { deleteStudent } from '@/api/student'
18
19 export default {
20   name: 'StudentDelete',
21   data() {
22     return {
23       table: {},
24       delDisabled: true
25     }
26   },
27   watch: {
28     'table.selection'() {
29       this.delDisabled = this.table.selection.length <= 0
30     }
31   },
32   mounted() {
33     this.table = this.$parent.$parent.$children.find(vc => vc &&
vc.tableId)
34   },
35   methods: {
36     doDelete(row) {
37       if (!row) {
38         this.$alert('您未选择记录', '删除失败', {
39           confirmButtonText: '确定'
40         })
41         return
42       }
43       this.$confirm('此操作将永久删除数据，是否继续?', '提示', {
44         confirmButtonText: '确定',
45         cancelButtonText: '取消',
46         type: 'warning'
47       }).then(() => {
48         deleteStudent(row).then(response => {
49           this.$notify.success({
50             title: '成功',
51             message: '删除成功'
52           })
53           this.table.refresh()
54         })
55       })
56     }
57   }
58 }
59 </script>
60
61 <style scoped>
62
63 </style>

```

3. index.vue中引入删除组件

```

40 |         </div>
41 |       </template>
42 |     </el-table-column>
43 |   </el-table>
44 |   <pagination v-show="query.total > 0" :total="query.total" :page.sync="query.currentPage" />
45 | </div>
46 | </template>
47 |
48 | <script>
49 |
50 | import { getStudentPage } from '@api/student'
51 | import Query from './query'
52 | import StudentAdd from './add'
53 | import StudentEdit from './edit'
54 | import StudentDelete from './delete'
55 |
56 | import Pagination from '@components/Pagination'
57 |
58 | export default {
59 |   name: 'Student',
60 |   components: { Query, Pagination, StudentAdd, StudentEdit, StudentDelete },
61 |   data() {
62 |     return {

```

4. 添加删除组件

```

1 | <template>
2 |   <!-- 主页面容器 -->
3 |   <div class="app-container">
4 |     <!-- 查询容器 -->
5 |     <table-query>
6 |       <query :query.sync="query" />
7 |     </table-query>
8 |     <!-- 表格工具条容器 -->
9 |     <table-header>
10 |       <student-add />
11 |       <student-edit ref="btnEdit" />
12 |       <student-delete ref="btnDel" />
13 |     </table-header>
14 |     <!-- 表格 -->
15 |     <el-table
16 |       ref="indexTable"

```

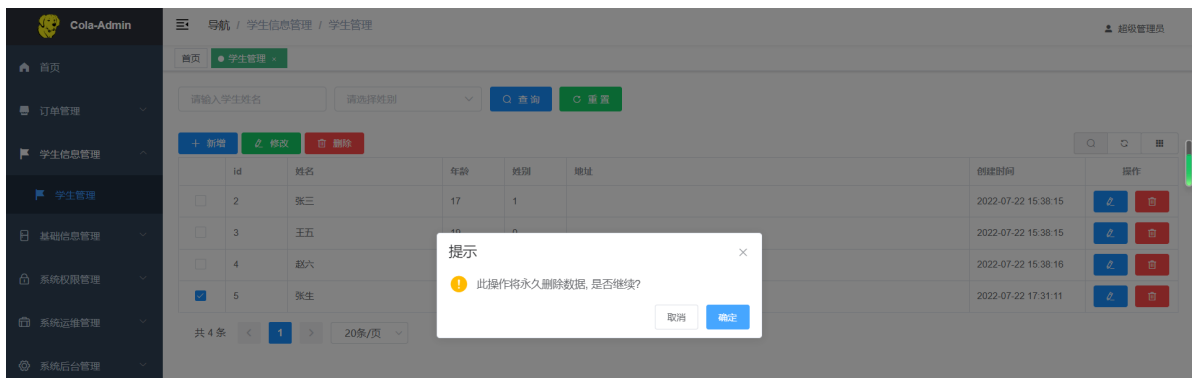
5. 修改index.vue中的代码

```

100 |       this.query.total = 0
101 |     }
102 |     this.tableLoading = false
103 |   }).catch(() => {
104 |     this.tableLoading = false
105 |     this.query.total = 0
106 |   })
107 | },
108 | toEdit(row) {
109 |   this.$refs.btnEdit.showEdit(row)
110 | },
111 | toDelete(row) {
112 |   this.$refs.btnDel.doDelete(row)
113 | }
114 | }
115 | }
116 | </script>

```

6. 最终效果



使用字典

增删查改功能已经都实现了，再来看列表中的数据，可以看到性别字段还是显示0、1，一般这种字段是使用字典功能来实现的。

1. 添加字典数据

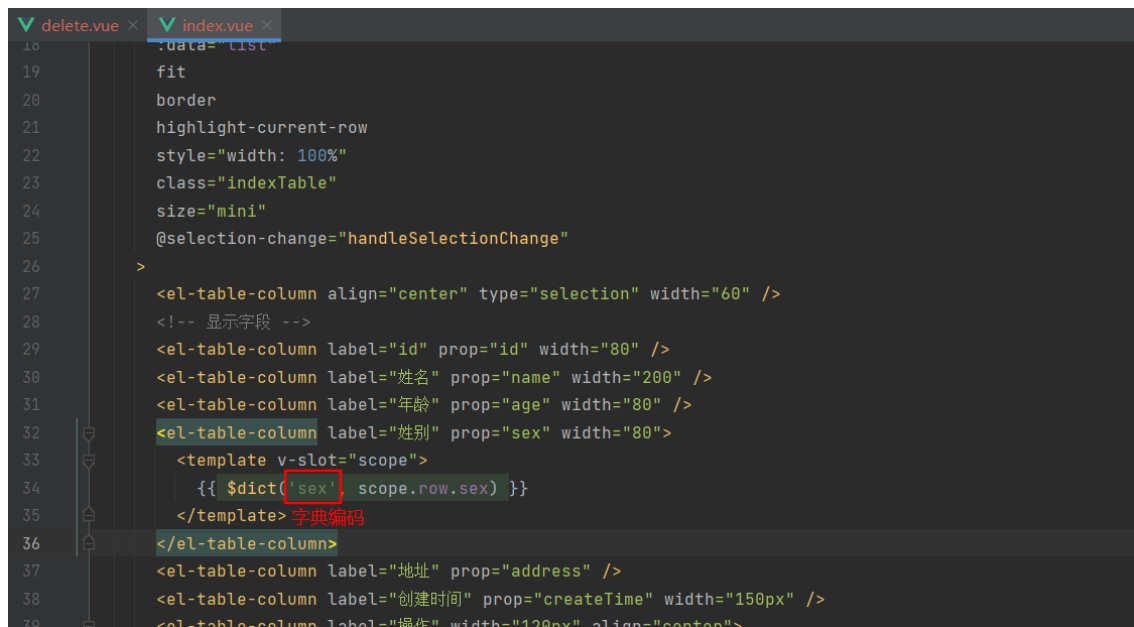
点击系统后台管理->字典管理页面，添加字典数据，系统已经存在性别字典，这里不再重复添加



2. 列表中使用字典

修改index.vue页面

```
1 <el-table-column label="性别" prop="sex" width="80">
2   <template v-slot="scope">
3     {{ $dict('sex', scope.row.sex) }}
4   </template>
5 </el-table-column>
```



3. 添加和修改页面中使用字典

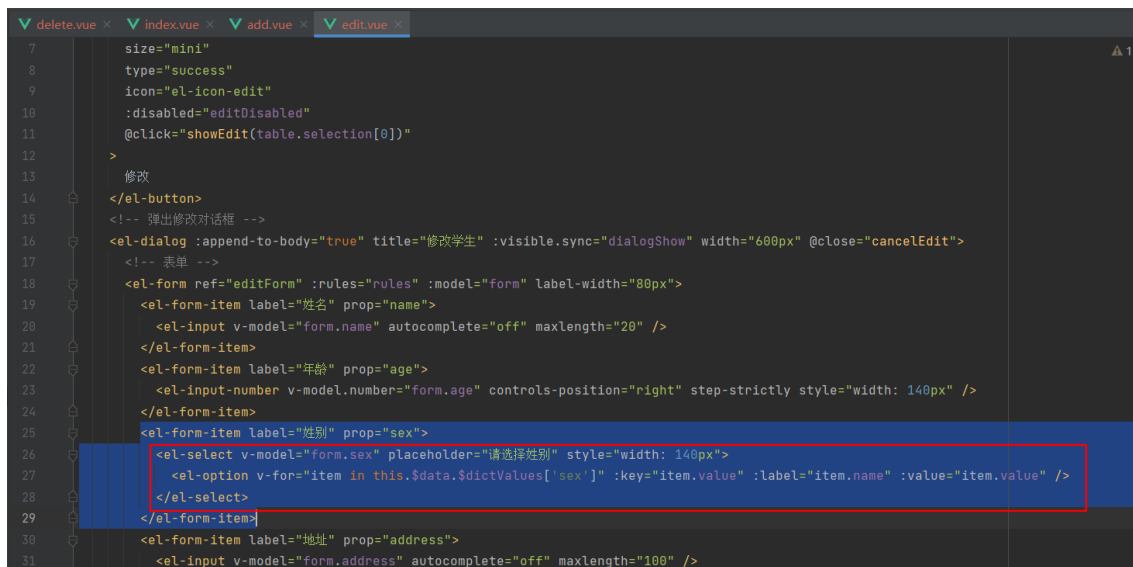
```
1 <el-form-item label="性别" prop="sex">
2   <el-select v-model="form.sex" placeholder="请选择性别" style="width:
140px">
3     <el-option v-for="item in this.$data.$dictValues['sex']"
:key="item.value" :label="item.name" :value="item.value" />
4   </el-select>
5 </el-form-item>
```

添加页面



```
13 </el-button>
14 <!-- 新增对话框 -->
15 <el-dialog :append-to-body="true" title="添加学生" :visible.sync="dialogShow" width="600px" @close="cancelAdd">
16   <!-- form表单 -->
17   <el-form ref="addForm" :rules="rules" :model="form" label-width="80px">
18     <el-form-item label="姓名" prop="name">
19       <el-input v-model="form.name" autocomplete="off" maxlength="20" />
20     </el-form-item>
21     <el-form-item label="年龄" prop="age">
22       <el-input-number v-model.number="form.age" controls-position="right" step-strictly style="width: 140px" />
23     </el-form-item>
24     <el-form-item label="性别" prop="sex">
25       <el-select v-model="form.sex" placeholder="请选择性别" style="width: 140px">
26         <el-option v-for="item in this.$data.$dictValues['sex']" :key="item.value" :label="item.name" :value="item.value" />
27       </el-select>
28     </el-form-item>
29     <el-form-item label="地址" prop="address">
30       <el-input v-model="form.address" autocomplete="off" maxlength="100" />
31     </el-form-item>
32   </el-form>
33 </el-dialog>
```

修改页面



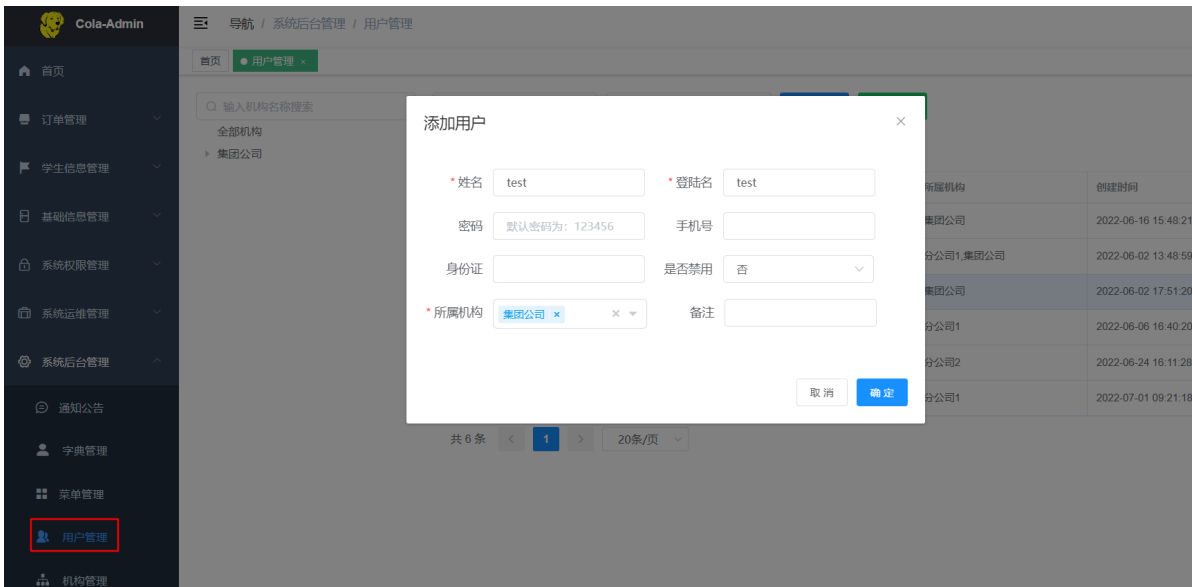
```
7 size="mini"
8 type="success"
9 icon="el-icon-edit"
10 :disabled="editDisabled"
11 @click="showEdit(table.selection[0])"
12 >
13 修改
14 </el-button>
15 <!-- 弹出修改对话框 -->
16 <el-dialog :append-to-body="true" title="修改学生" :visible.sync="dialogShow" width="600px" @close="cancelEdit">
17   <!-- 表单 -->
18   <el-form ref="editForm" :rules="rules" :model="form" label-width="80px">
19     <el-form-item label="姓名" prop="name">
20       <el-input v-model="form.name" autocomplete="off" maxlength="20" />
21     </el-form-item>
22     <el-form-item label="年龄" prop="age">
23       <el-input-number v-model.number="form.age" controls-position="right" step-strictly style="width: 140px" />
24     </el-form-item>
25     <el-form-item label="性别" prop="sex">
26       <el-select v-model="form.sex" placeholder="请选择性别" style="width: 140px">
27         <el-option v-for="item in this.$data.$dictValues['sex']" :key="item.value" :label="item.name" :value="item.value" />
28       </el-select>
29     </el-form-item>
30     <el-form-item label="地址" prop="address">
31       <el-input v-model="form.address" autocomplete="off" maxlength="100" />
32     </el-form-item>
33   </el-form>
34 </el-dialog>
```

开发进阶

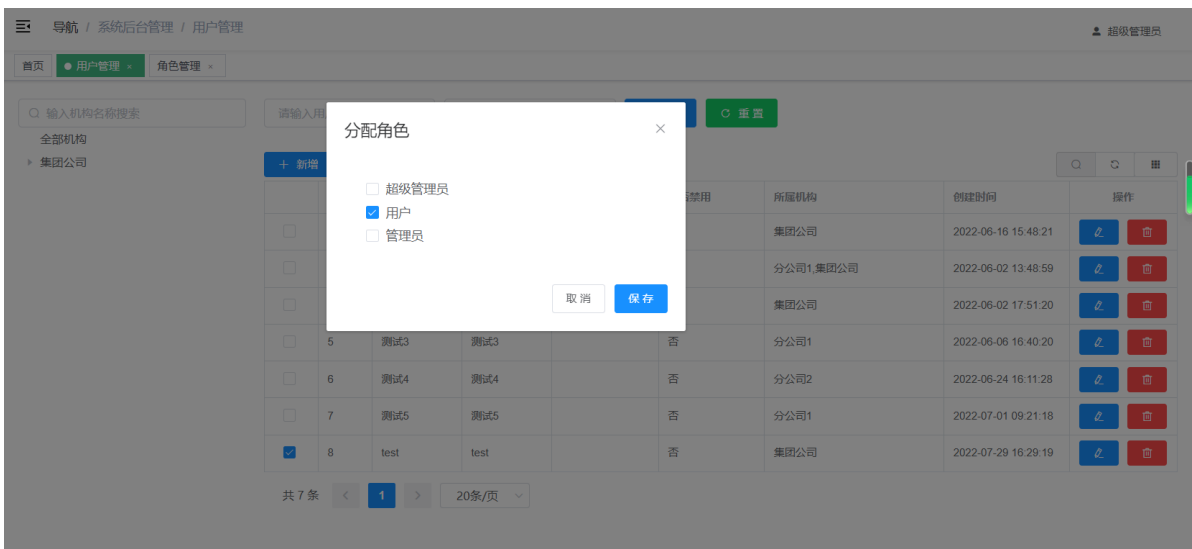
按钮权限

添加用户

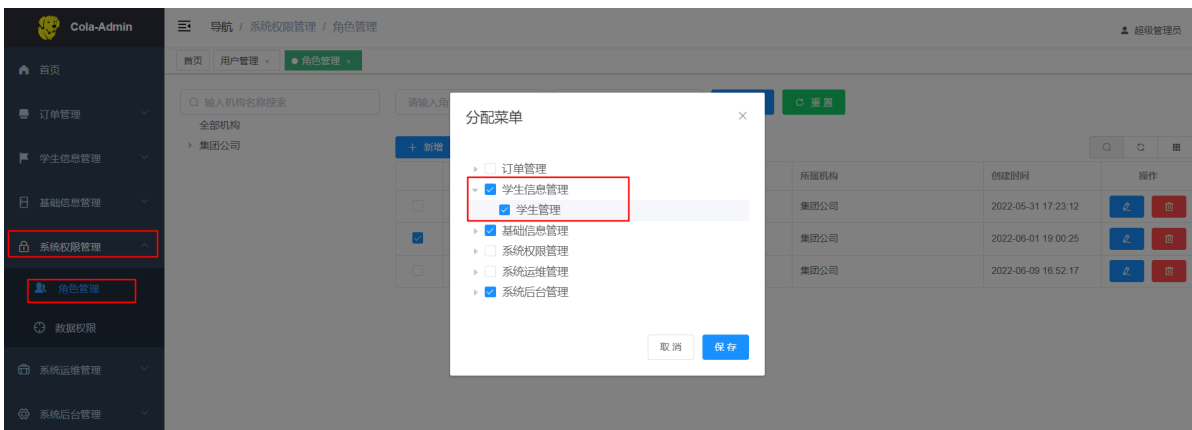
cola-ui中超级管理员是无视权限的，所以需要先添加一个测试用户test，密码默认为：123456



给用户分配角色

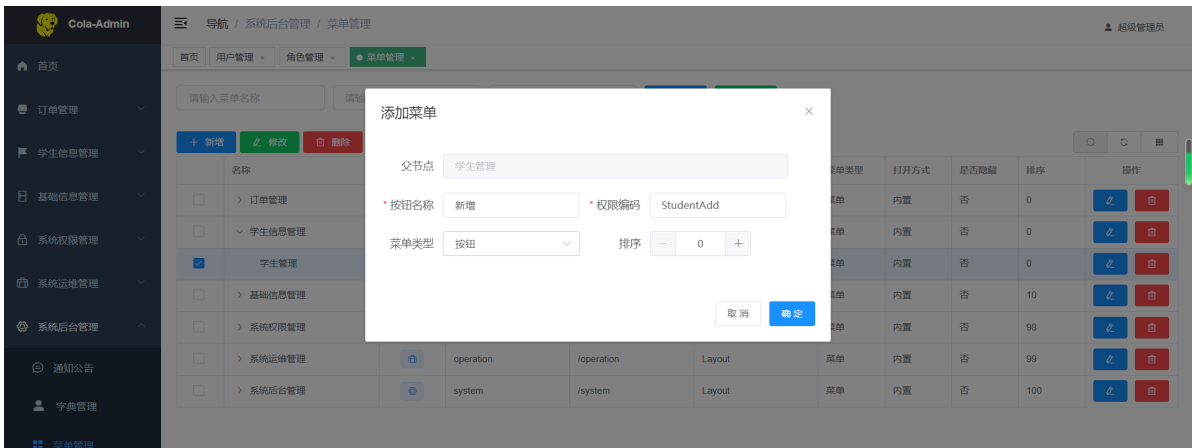


给角色分配菜单，进入系统权限管理->角色管理，选中用户角色，点击分配菜单，将学生信息管理->学生管理菜单分配给用户角色

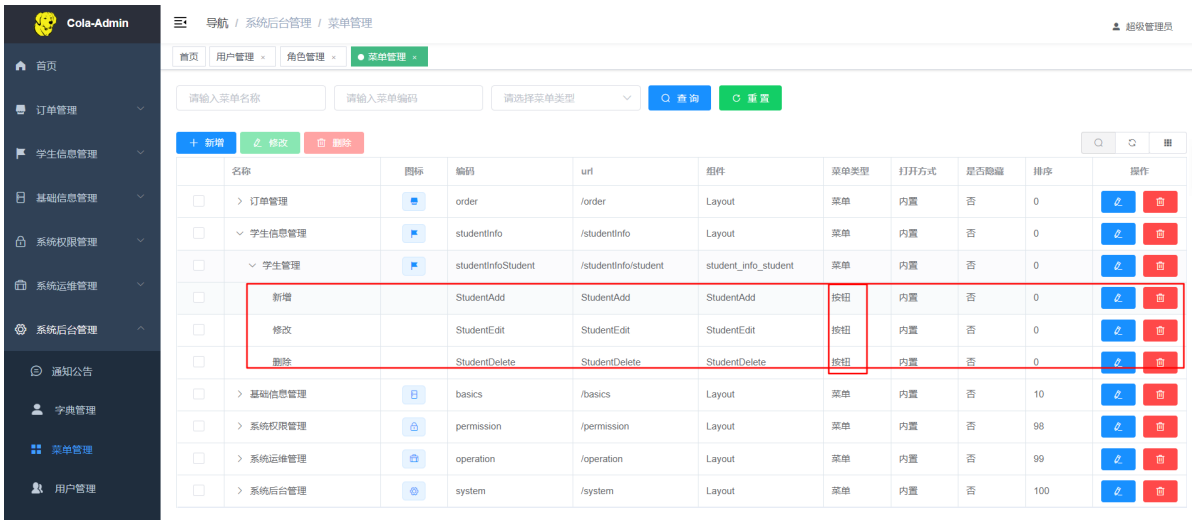


添加按钮

进入菜单管理，给学生管理菜单添加按钮



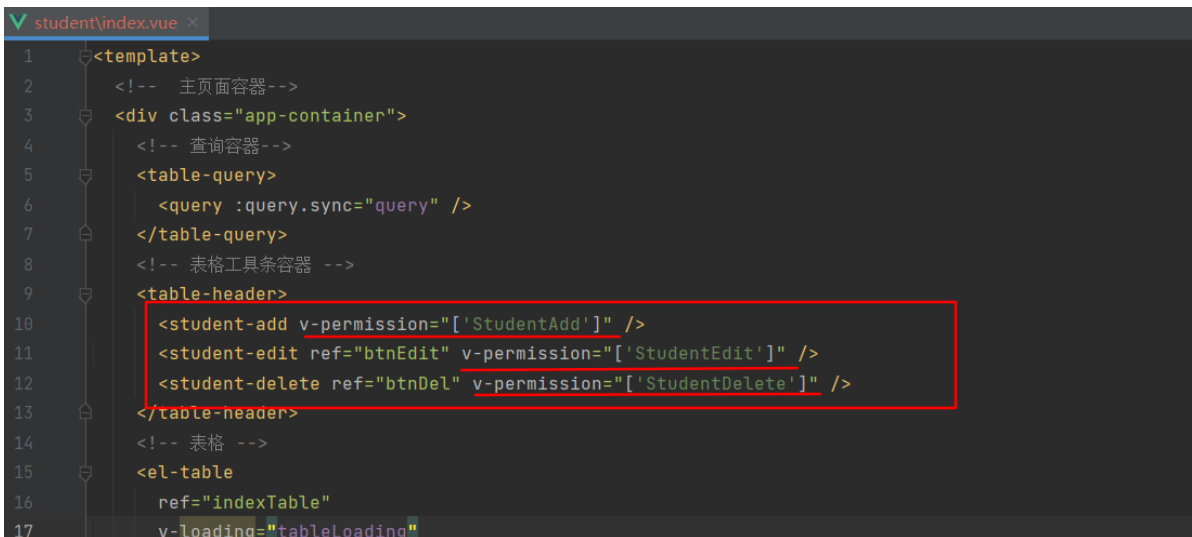
添加后如下图



修改代码

打开src\views\index.vue

修改表格顶部操作按钮

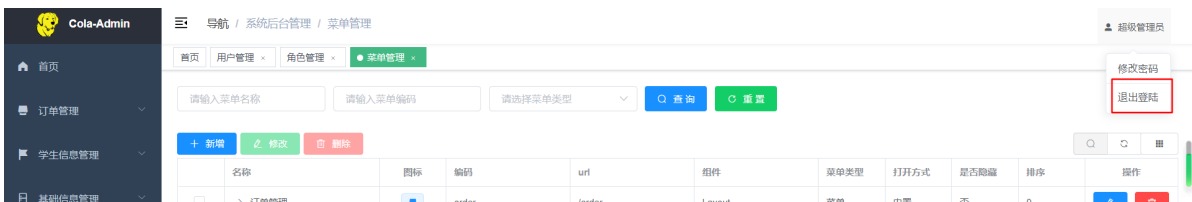


修改表格行内操作按钮

```
28 <!-- 显示字段 -->
29 <el-table-column label="id" prop="id" width="80" />
30 <el-table-column label="姓名" prop="name" width="200" />
31 <el-table-column label="年龄" prop="age" width="80" />
32 <el-table-column label="性别" prop="sex" width="80">
33   <template v-slot="scope">
34     {{ $dict('sex', scope.row.sex) }}
35   </template>
36 </el-table-column>
37 <el-table-column label="地址" prop="address" />
38 <el-table-column label="创建时间" prop="createTime" width="150px" />
39 <el-table-column label="操作" width="120px" align="center">
40   <template v-slot="scope">
41     <div style="display: flex;">
42       <el-button v-permission="['StudentEdit']" type="primary" icon="el-icon-edit" size="mini" @click="toEdit(scope.row)" />
43       <el-button v-permission="['StudentDelete']" type="danger" icon="el-icon-delete" size="mini" @click="toDelete(scope.row)" />
44     </div>
45   </template>
46 </el-table-column>
47 </el-table>
```

使用测试用户登陆

退出当前用户

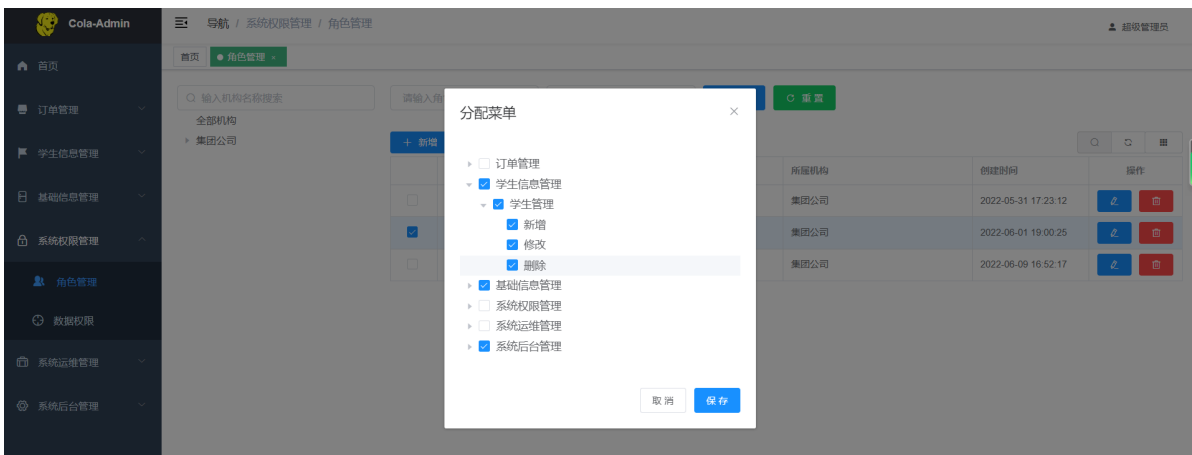


使用测试用户登陆



分配按钮权限

退出登陆，使用admin用户重新登陆，并给用户这个角色分配上按钮的权限



再次测试

退出登陆，使用test用户重新登陆，可以看到按钮可以显示了

Cola-Admin

首页

学生信息管理

学生管理

基础信息管理

系统后台管理

导航 / 学生信息管理 / 学生管理

test

学生管理

请输入学生姓名

请选择性别

查询

重置

+ 新增

✎ 修改

🗑 删除

	id	姓名	年龄	性别	地址	创建时间	操作
<input type="checkbox"/>	2	张三	17	男		2022-07-22 15:38:15	<div><div>✎</div><div>🗑</div></div>
<input type="checkbox"/>	3	王五	19	女		2022-07-22 15:38:15	<div><div>✎</div><div>🗑</div></div>
<input type="checkbox"/>	4	赵六	20	男		2022-07-22 15:38:16	<div><div>✎</div><div>🗑</div></div>
<input type="checkbox"/>	5	张生	19	女		2022-07-22 17:31:11	<div><div>✎</div><div>🗑</div></div>

共 4 条

<

1

>

20条/页

表格列动态显示