今日目标

1.实现后台首页的基本布局
2.实现左侧菜单栏
3.实现用户列表展示
4.实现添加用户

1.后台首页基本布局

打开Home.vue组件，进行布局：

```
<el-container class="home-container">
  <!-- 头部区域 -->
  <el-header>Header<el-button type="info" @click="logout"> 退出 </el-button></el-
header>
  <!-- 页面主体区域 -->
  <el-container>
    <!-- 侧边栏 -->
    <el-aside width="200px">Aside</el-aside>
    <!-- 主体结构 -->
    <el-main>Main</el-main>
  </el-container>
</el-container>
```

默认情况下，跟element-ui组件同名的类名可以帮助我们快速的给对应的组件添加样式，如：

```
.home-container {
  height: 100%;
}
.el-header{
  background-color:#373D41;
}
.el-aside{
  background-color:#333744;
}
.el-main{
  background-color:#eaedf1;
}
```

2.顶部布局，侧边栏布局

```
<template>
    <el-container class="home-container">
      <!-- 头部区域 -->
      <el-header>
        <div>
          <!-- 黑马logo -->
          <img src="../assets/heima.png" alt="">
          <!-- 顶部标题 -->
          <span>电商后台管理系统</span>
        </div>
```

```
        <el-button type="info" @click="logout"> 退出 </el-button>
      </el-header>
      <!-- 页面主体区域 -->
      <el-container>
        <!-- 侧边栏 -->
        <el-aside width="200px">
          <!-- 侧边栏菜单 -->
          <el-menu
            background-color="#333744"
            text-color="#fff"
            active-text-color="#ffd04b">
            <!-- 一级菜单 -->
            <el-submenu index="1">
              <!-- 一级菜单模板 -->
              <template slot="title">
                <!-- 图标 -->
                <i class="el-icon-location"></i>
                <!-- 文本 -->
                <span>导航一</span>
              </template>
              <!-- 二级子菜单 -->
              <el-menu-item index="1-4-1">
                <!-- 二级菜单模板 -->
                <template slot="title">
                  <!-- 图标 -->
                  <i class="el-icon-location"></i>
                  <!-- 文本 -->
                  <span>子菜单一</span>
                </template>
              </el-menu-item>
            </el-submenu>

          </el-menu>
        </el-aside>
        <!-- 主体结构 -->
        <el-main>Main</el-main>
      </el-container>
    </el-container>
</template>
```

3.axios请求拦截器

后台除了登录接口之外，都需要token权限验证，我们可以通过添加axios请求拦截器来添加token，以保证拥有获取数据的权限
在main.js中添加代码，在将axios挂载到vue原型之前添加下面的代码

```
//请求在到达服务器之前，先会调用use中的这个回调函数来添加请求头信息
axios.interceptors.request.use(config=>{
  //为请求头对象，添加token验证的Authorization字段
  config.headers.Authorization = window.sessionStorage.getItem("token")
  return config
})
```

4.请求侧边栏数据

```
<script>
export default {
  data() {
    return {
      // 左侧菜单数据
      menuList: null
    }
  },
  created() {
    // 在created阶段请求左侧菜单数据
    this.getMenuList()
  },
  methods: {
    logout() {
      window.sessionStorage.clear()
      this.$router.push('/login')
    },
    async getMenuList() {
      // 发送请求获取左侧菜单数据
      const { data: res } = await this.$http.get('menus')
      if (res.meta.status !== 200) return this.$message.error(res.meta.msg)

      this.menuList = res.data
      console.log(res)
    }
  }
}
</script>
```

通过v-for双重循环渲染左侧菜单

```
<el-menu
  background-color="#333744"
  text-color="#fff"
  active-text-color="#ffd04b">
  <!-- 一级菜单 -->
  <el-submenu :index="item.id+''" v-for="item in menuList" :key="item.id">
    <!-- 一级菜单模板 -->
    <template slot="title">
      <!-- 图标 -->
      <i class="el-icon-location"></i>
      <!-- 文本 -->
      <span>{{item.authName}}</span>
    </template>
    <!-- 二级子菜单 -->
    <el-menu-item :index="subItem.id+''" v-for="subItem in item.children"
:key="subItem.id">
      <!-- 二级菜单模板 -->
      <template slot="title">
        <!-- 图标 -->
        <i class="el-icon-location"></i>
        <!-- 文本 -->
        <span>{{subItem.authName}}</span>
      </template>
```

```
    </el-menu-item>
  </el-submenu>
</el-menu>
```

5.设置激活子菜单样式

通过更改el-menu的active-text-color属性可以设置侧边栏菜单中点击的激活项的文字颜色
通过更改菜单项模板（template）中的i标签的类名，可以将左侧菜单栏的图标进行设置，我们需要在项目中使用第三方字体图标
在数据中添加一个iconsObj：

```
iconsObj: {
        '125':'iconfont icon-user',
        '103':'iconfont icon-tijikongjian',
        '101':'iconfont icon-shangpin',
        '102':'iconfont icon-danju',
        '145':'iconfont icon-baobiao'
      }
```

然后将图标类名进行数据绑定，绑定iconsObj中的数据：

为了保持左侧菜单每次只能打开一个，显示其中的子菜单，我们可以在el-menu中添加一个属性unique-opened
或者也可以数据绑定进行设置(此时true认为是一个bool值，而不是字符串) :unique-opened="true"

6.制作侧边菜单栏的伸缩功能

在菜单栏上方添加一个div

```
        <!-- 侧边栏,宽度根据是否折叠进行设置 -->
        <el-aside :width="isCollapse ? '64px':'200px'">
          <!-- 伸缩侧边栏按钮 -->
          <div class="toggle-button" @click="toggleCollapse">|||</div>
          <!-- 侧边栏菜单，:collapse="isCollapse"（设置折叠菜单为绑定的 isCollapse 值），:collapse-transition="false"（关闭默认的折叠动画） -->
          <el-menu
          :collapse="isCollapse"
          :collapse-transition="false"
          ......
```

然后给div添加样式，给div添加事件：

|||

7.在后台首页添加子级路由

新增子级路由组件Welcome.vue
在router.js中导入子级路由组件，并设置路由规则以及子级路由的默认重定向
打开Home.vue，在main的主体结构中添加一个路由占位符

制作好了Welcome子级路由之后，我们需要将所有的侧边栏二级菜单都改造成子级路由链接
我们只需要将el-menu的router属性设置为true就可以了，此时当我们点击二级菜单的时候，就会根据菜单的index
属性进行路由跳转,如: /110,
使用index id来作为跳转的路径不合适，我们可以重新绑定index的值为 :index="'/'+subItem.path"

8.完成用户列表主体区域

新建用户列表组件 user/Users.vue
在router.js中导入子级路由组件Users.vue，并设置路由规则

当点击二级菜单的时候，被点击的二级子菜单并没有高亮，我们需要正在被使用的功能高亮显示
我们可以通过设置el-menu的default-active属性来设置当前激活菜单的index
但是default-active属性也不能写死，固定为某个菜单值
所以我们可以先给所有的二级菜单添加点击事件,并将path值作为方法的参数
@click="saveNavState('/'+subItem.path)"
在saveNavState方法中将path保存到sessionStorage中
saveNavState(path){
 //点击二级菜单的时候保存被点击的二级菜单信息
 window.sessionStorage.setItem("activePath",path);
 this.activePath = path;
}
然后在数据中添加一个activePath绑定数据，并将el-menu的default-active属性设置为activePath
最后在created中将sessionStorage中的数据赋值给activePath
this.activePath = window.sessionStorage.getItem("activePath")

9.绘制用户列表基本结构

A.使用element-ui面包屑组件完成顶部导航路径(复制面包屑代码，在element.js中导入组件Breadcrumb,BreadcrumbItem)
B.使用element-ui卡片组件完成主体表格(复制卡片组件代码，在element.js中导入组件Card)，再使用element-ui输入框完成搜索框及搜索按钮，
此时我们需要使用栅格布局来划分结构(复制卡片组件代码，在element.js中导入组件Row，Col)，然后再使用el-button制作添加用户按钮

```
<div>
    <h3>用户列表组件</h3>
    <!-- 面包屑导航 -->
    <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>
    <!-- 卡片视图区域 -->
    <el-card>
        <!-- 搜索与添加区域 -->
        <el-row :gutter="20">
            <el-col :span="7">
                <el-input placeholder="请输入内容">
                    <el-button slot="append" icon="el-icon-search"></el-button>
                </el-input>
            </el-col>
            <el-col :span="4">
                <el-button type="primary">添加用户</el-button>
            </el-col>
        </el-row>
    </el-card>
</div>
```

10.请求用户列表数据

```
<script>
export default {
  data() {
    return {
      //获取查询用户信息的参数
      queryInfo: {
        query: '',
        pagenum: 1,
        pagesize: 2
      },
      //保存请求回来的用户列表数据
      userList:[],
      total:0
    }
  },
  created() {
    this.getUserList()
  },
  methods: {
    async getUserList() {
      //发送请求获取用户列表数据
      const { res: data } = await this.$http.get('users', {
        params: this.queryInfo
      })
      //如果返回状态为异常状态则报错并返回
      if (res.meta.status !== 200)
        return this.$message.error('获取用户列表失败')
      //如果返回状态正常，将请求的数据保存在data中
      this.userList = res.data.users;
      this.total = res.data.total;
    }
  }
}
</script>
```

11.将用户列表数据展示

使用表格来展示用户列表数据，使用element-ui表格组件完成列表展示数据(复制表格代码，在element.js中导入组件Table,TableColumn)
在渲染展示状态时，会使用作用域插槽获取每一行的数据
再使用switch开关组件展示状态信息(复制开关组件代码，在element.js中导入组件Switch)
而渲染操作列时，也是使用作用域插槽来进行渲染的，
在操作列中包含了修改，删除，分配角色按钮，当我们把鼠标放到分配角色按钮上时
希望能有一些文字提示，此时我们需要使用文字提示组件(复制文字提示组件代码，在element.js中导入组件Tooltip),将分配角色按钮包含
代码结构如下：

```
<!-- 用户列表(表格)区域 -->
<el-table :data="userList" border stripe>
    <el-table-column type="index"></el-table-column>
    <el-table-column label="姓名" prop="username"></el-table-column>
    <el-table-column label="邮箱" prop="email"></el-table-column>
    <el-table-column label="电话" prop="mobile"></el-table-column>
```

```
    <el-table-column label="角色" prop="role_name"></el-table-column>
    <el-table-column label="状态">
        <template slot-scope="scope">
            <el-switch v-model="scope.row.mg_state"></el-switch>
        </template>
    </el-table-column>
    <el-table-column label="操作" width="180px">
        <template slot-scope="scope">
            <!-- 修改 -->
            <el-button type="primary" icon="el-icon-edit" size='mini'></el-
button>
            <!-- 删除 -->
            <el-button type="danger" icon="el-icon-delete" size='mini'></el-
button>
            <!-- 分配角色 -->
            <el-tooltip class="item" effect="dark" content="分配角色"
placement="top" :enterable="false">
                <el-button type="warning" icon="el-icon-setting" size='mini'>
</el-button>
            </el-tooltip>
        </template>
    </el-table-column>
</el-table>
```

12.实现用户列表分页

A.使用表格来展示用户列表数据，可以使用分页组件完成列表分页展示数据(复制分页组件代码，在element.js中导入组件Pagination)
B.更改组件中的绑定数据

```
<!-- 分页导航区域
@size-change(pagesize改变时触发)
@current-change(页码发生改变时触发)
:current-page(设置当前页码)
:page-size(设置每页的数据条数)
:total(设置总页数) -->
            <el-pagination @size-change="handleSizeChange" @current-
change="handleCurrentChange" :current-page="queryInfo.pagenum" :page-sizes="[1,
2, 5, 10]" :page-size="queryInfo.pagesize" layout="total, sizes, prev, pager,
next, jumper" :total="total">
            </el-pagination>
```

C.添加两个事件的事件处理函数@size-change，@current-change

```
handleSizeChange(newSize) {
  //pagesize改变时触发，当pagesize发生改变的时候，我们应该
  //以最新的pagesize来请求数据并展示数据
  //   console.log(newSize)
  this.queryInfo.pagesize = newSize;
  //重新按照pagesize发送请求，请求最新的数据
  this.getUserList();
},
handleCurrentChange( current ) {
  //页码发生改变时触发当current发生改变的时候，我们应该
  //以最新的current页码来请求数据并展示数据
  //   console.log(current)
```

```
  this.queryInfo.pagenum = current;
  //重新按照pagenum发送请求，请求最新的数据
  this.getUserList();
}
```

13.实现更新用户状态

当用户点击列表中的switch组件时，用户的状态应该跟随发生改变。
A.首先监听用户点击switch组件的事件，并将作用域插槽的数据当做事件参数进行传递

```
<el-switch v-model="scope.row.mg_state" @change="userStateChanged(scope.row)">
</el-switch>
```

B.在事件中发送请求完成状态的更改

```
async userStateChanged(row) {
  //发送请求进行状态修改
  const { data: res } = await this.$http.put(
    `users/${row.id}/state/${row.mg_state}`
  )
  //如果返回状态为异常状态则报错并返回
  if (res.meta.status !== 200) {
    row.mg_state = !row.mg_state
    return this.$message.error('修改状态失败')
  }
  this.$message.success('更新状态成功')
},
```

14.实现搜索功能

添加数据绑定，添加搜索按钮的点击事件(当用户点击搜索按钮的时候，调用getUserList方法根据文本框内容重新请求用户列表数据)
当我们在输入框中输入内容并点击搜索之后，会按照搜索关键字搜索，我们希望能够提供一个X删除搜索关键字并重新获取所有的用户列表数据，只需要给文本框添加clearable属性并添加clear事件，在clear事件中重新请求数据即可

```
<el-col :span="7">
    <el-input placeholder="请输入内容" v-model="queryInfo.query" clearable
@clear="getUserList">
        <el-button slot="append" icon="el-icon-search" @click="getUserList">
</el-button>
    </el-input>
</el-col>
```

15.实现添加用户

A.当我们点击添加用户按钮的时候，弹出一个对话框来实现添加用户的功能，首先我们需要复制对话框组件的代码并在element.js文件中引入Dialog组件

B.接下来我们要为“添加用户”按钮添加点击事件，在事件中将addDialogVisible设置为true，即显示对话框

C.更改Dialog组件中的内容

```
<!-- 对话框组件  :visible.sync(设置是否显示对话框) width(设置对话框的宽度)
:before-close(在对话框关闭前触发的事件) -->
<el-dialog title="添加用户" :visible.sync="addDialogVisible" width="50%">
    <!-- 对话框主体区域 -->
    <el-form :model="addForm" :rules="addFormRules" ref="addFormRef" label-
width="70px">
        <el-form-item label="用户名" prop="username">
            <el-input v-model="addForm.username"></el-input>
        </el-form-item>
        <el-form-item label="密码" prop="password">
            <el-input v-model="addForm.password"></el-input>
        </el-form-item>
        <el-form-item label="邮箱" prop="email">
            <el-input v-model="addForm.email"></el-input>
        </el-form-item>
        <el-form-item label="电话" prop="mobile">
            <el-input v-model="addForm.mobile"></el-input>
        </el-form-item>
    </el-form>
    <!-- 对话框底部区域 -->
    <span slot="footer" class="dialog-footer">
        <el-button @click="addDialogVisible = false">取 消</el-button>
        <el-button type="primary" @click="addDialogVisible = false">确 定</el-
button>
    </span>
</el-dialog>
```

D.添加数据绑定和校验规则：

```
data() {
  //验证邮箱的规则
  var checkEmail = (rule, value, cb) => {
    const regEmail = /^\w+@\w+(\.\w+)+$/
    if (regEmail.test(value)) {
      return cb()
    }
    //返回一个错误提示
    cb(new Error('请输入合法的邮箱'))
  }
  //验证手机号码的规则
  var checkMobile = (rule, value, cb) => {
    const regMobile = /^1[34578]\d{9}$/
    if (regMobile.test(value)) {
      return cb()
    }
    //返回一个错误提示
    cb(new Error('请输入合法的手机号码'))
  }
  return {
    //获取查询用户信息的参数
    queryInfo: {
      // 查询的条件
      query: '',
      // 当前的页数，即页码
      pagenum: 1,
      // 每页显示的数据条数
      pagesize: 2
```

```
    },
    //保存请求回来的用户列表数据
    userList: [],
    total: 0,
    //是否显示添加用户弹出窗
    addDialogVisible: false,
    // 添加用户的表单数据
    addForm: {
      username: '',
      password: '',
      email: '',
      mobile: ''
    },
    // 添加表单的验证规则对象
    addFormRules: {
      username: [
        { required: true, message: '请输入用户名称', trigger: 'blur' },
        {
          min: 3,
          max: 10,
          message: '用户名在3~10个字符之间',
          trigger: 'blur'
        }
      ],
      password: [
        { required: true, message: '请输入密码', trigger: 'blur' },
        {
          min: 6,
          max: 15,
          message: '用户名在6~15个字符之间',
          trigger: 'blur'
        }
      ],
      email: [
          { required: true, message: '请输入邮箱', trigger: 'blur' },
          { validator:checkEmail, message: '邮箱格式不正确，请重新输入', trigger:
'blur'}
      ],
      mobile: [
          { required: true, message: '请输入手机号码', trigger: 'blur' },
          { validator:checkMobile, message: '手机号码不正确，请重新输入', trigger:
'blur'}
      ]
    }
  }
}
```

E.当关闭对话框时，重置表单
给el-dialog添加@close事件，在事件中添加重置表单的代码

```
methods:{
  ....
  addDialogClosed(){
      //对话框关闭之后，重置表达
      this.$refs.addFormRef.resetFields();
  }
}
```

F.点击对话框中的确定按钮，发送请求完成添加用户的操作
首先给确定按钮添加点击事件，在点击事件中完成业务逻辑代码

```
methods:{
  ....
  addUser(){
      //点击确定按钮，添加新用户
      //调用validate进行表单验证
      this.$refs.addFormRef.validate( async valid => {
          if(!valid) return this.$message.error("请填写完整用户信息");
          //发送请求完成添加用户的操作
          const {data:res} = await this.$http.post("users",this.addForm)
          //判断如果添加失败，就做提示
          if (res.meta.status !== 200)
              return this.$message.error('添加用户失败')
          //添加成功的提示
          this.$message.success("添加用户成功")
          //关闭对话框
          this.addDialogVisible = false
          //重新请求最新的数据
          this.getUserList()
      })
  }
}
```