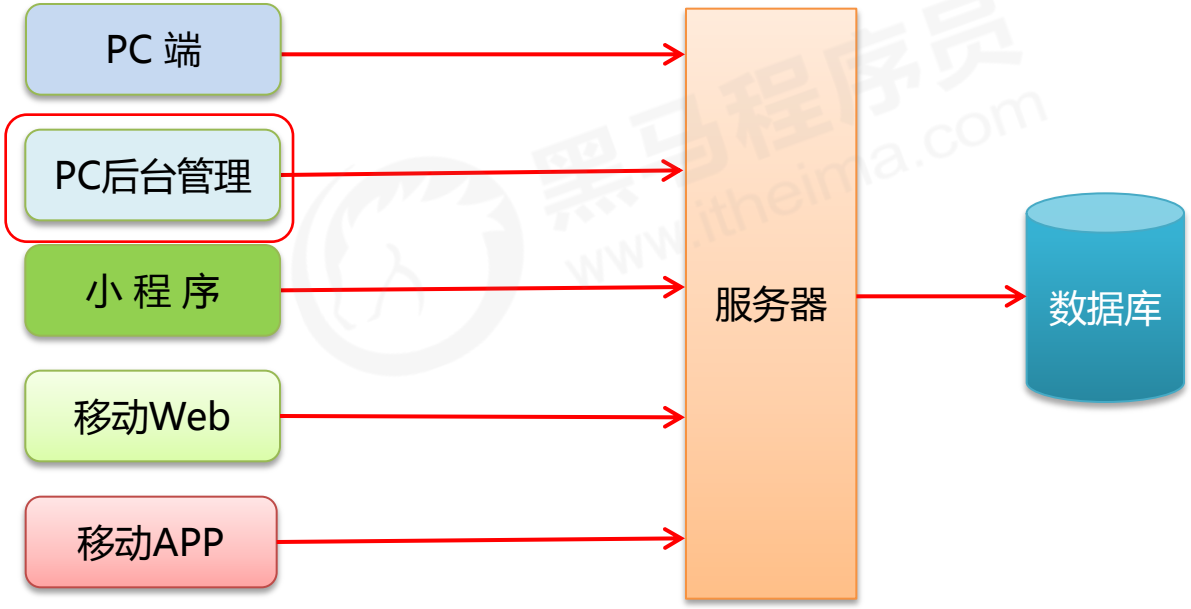

黑马程序员™
www.itheima.com
传智播客旗下
高端IT教育品牌
Vue全家桶-项目实战

目录 Contents

1. 项目概述

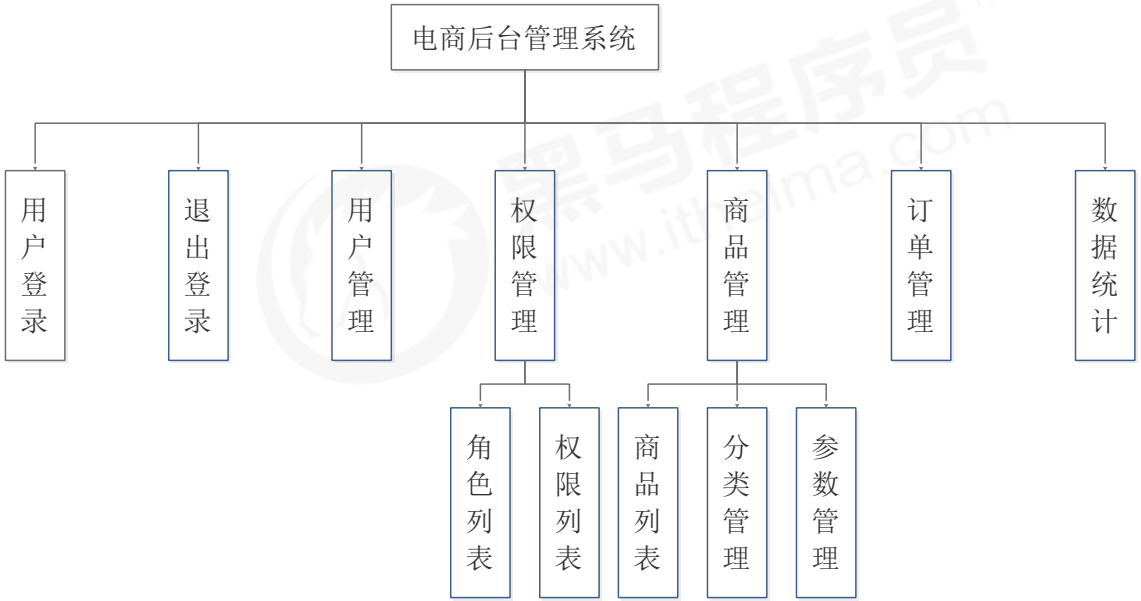
1.1 电商项目基本业务概述

根据不同的应用场景，电商系统一般都提供了 PC 端、移动 APP、移动 Web、微信小程序等多种终端访问方式。



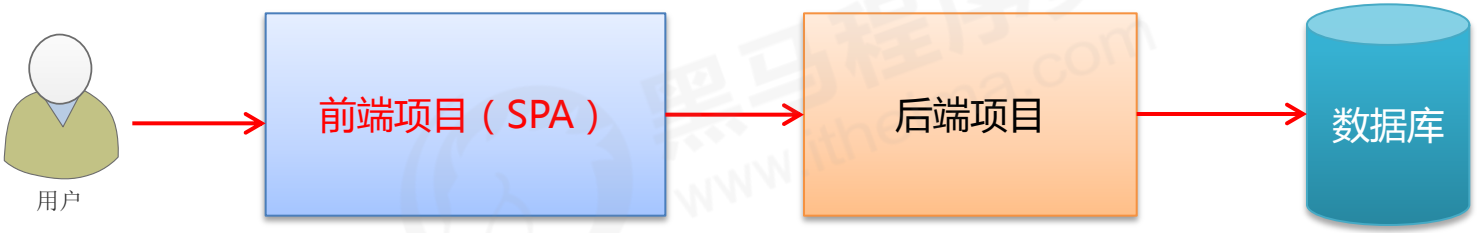
1.2 电商后台管理系统的功能

电商后台管理系统用于管理用户账号、商品分类、商品信息、订单、数据统计等业务功能。



1.3 电商后台管理系统的开发模式（前后端分离）

电商后台管理系统整体采用前后端分离的开发模式，其中前端项目是基于 Vue 技术栈的 SPA 项目。



1.4 电商后台管理系统的技术选型

1. 前端项目技术栈

- Vue
- Vue-router
- Element-UI
- Axios
- Echarts

2. 后端项目技术栈

- Node.js
- Express
- Jwt
- Mysql
- Sequelize

目录 Contents

2. 项目初始化

2.1 前端项目初始化步骤

① 安装 Vue 脚手架
② 通过 Vue 脚手架创建项目
③ 配置 Vue 路由
④ 配置 Element-UI 组件库
⑤ 配置 axios 库
⑥ 初始化 git 远程仓库
⑦ 将本地项目托管到 Github 或 码云 中

2. 项目初始化

2.2 后台项目的环境安装配置

① 安装 MySQL 数据库
② 安装 Node.js 环境
③ 配置项目相关信息
④ 启动项目
⑤ 使用 Postman 测试后台项目接口是否正常

目录 Contents

3. 登录/退出功能

3.1 登录概述

1. 登录业务流程

① 在登录页面输入用户名和密码
② 调用后台接口进行验证
③ 通过验证之后，根据后台的响应状态跳转到项目主页

2. 登录业务的相关技术点

- http 是无状态的
- 通过 cookie 在客户端记录状态
- 通过 session 在服务器端记录状态
- 通过 token 方式维持状态

3. 登录/退出功能

3.2 登录 — token 原理分析

3. 登录/退出功能

3.3 登录功能实现

1. 登录页面的布局

通过 Element-UI 组件实现布局

- el-form
- el-form-item
- el-input
- el-button
- 字体图标

3. 登录/退出功能

3.3 登录功能实现

2. 实现登录

① 通过 axios 调用登录验证接口
② 登录成功之后保持用户 token 信息
③ 跳转到项目主页

```
const {data: res } = await this.$http.post('login', this.loginForm)
if (res.meta.status !== 200)return this.$message.error('登录失败！')
// 提示登录成功
this.$message.success('登录成功！')
// 把登录成功的token保存到sessionStorage
window.sessionStorage.setItem('token', res.data.token)
// 使用编程式导航，跳转到后台主页
this.$router.push('/home')
```

3. 登录/退出功能

3.3 登录功能实现

3. 路由导航守卫控制访问权限

如果用户没有登录，但是直接通过URL访问特定页面，需要重新导航到登录页面。

```
// 为路由对象，添加 beforeEach 导航守卫
router.beforeEach((to, from, next) => {
  // 如果用户访问的登录页，直接放行
  if (to.path === '/login') return next()
  // 从 sessionStorage 中获取到 保存的 token 值
  const tokenStr = window.sessionStorage.getItem('token')
  // 没有token，强制跳转到登录页
  if (!tokenStr) return next('/login')
  next()
})
```

3. 登录/退出功能

3.3 登录功能实现

4. Vue 直接操作 DOM

- 通过 ref 标注 DOM 元素

```
// 在 DOM 元素上通过 ref 属性标注，属性名称自定义
<div ref="info">hello</div>
```

- 通过 $refs 获取 DOM 元素

```
// 通过 Vue 实例的 $refs 获取标记 ref 属性的元素
let info = this.$refs.info.innerHTML
console.log(info) // hello
```

3. 登录/退出功能

3.3 登录功能实现

5. 基于 Element-UI 进行表单验证

Element-UI表单验证规则

```
loginFormRules: {
  // 登录名称的验证规则
  username: [{ required: true, message: '请输入用户名称', trigger: 'blur' }],
  password: [{ required: true, message: '请输入用户密码', trigger: 'blur' }]
}
```

```
// 进行表单验证
this.$refs.loginFormRef.validate(async valid => {
  // 如果验证失败，直接退出后续代码的执行
  if (!valid) return
  // 验证通过后这里完成登录成功后的相关操作（保存token、跳转到主页）
})
```

3.4 退出

退出功能实现原理

基于 token 的方式实现退出比较简单，只需要销毁本地的 token 即可。这样，后续的请求就不会携带 token ，必须重新登录生成一个新的 token 之后才可以访问页面。

```
// 清空token
window.sessionStorage.clear()
// 跳转到登录页
this.$router.push('/login')
```

目录 Contents

4. 主页布局

4.1 整体布局

整体布局：先上下划分，再左右划分。

```
<el-container>
  <!-- 头部区域 -->
  <el-header></el-header>
  <el-container>
    <!-- 侧边栏区域 -->
    <el-aside></el-aside>
    <!-- 右侧主体区域 -->
    <el-main></el-main>
  </el-container>
</el-container>
```

4.2 左侧菜单布局

菜单分为二级，并且可以折叠。

```
<el-menu>
   <el-submenu>
     <!-- 这个 template 是一级菜单的内容模板 -->
     <i class="el-icon-menu"></i>
     <span>一级菜单</span>
     <!-- 在一级菜单中，可以嵌套二级菜单 -->
     <el-menu-item>
       <i class="el-icon-menu"></i>
       <span slot="title">二级菜单</span>
     </el-menu-item>
   </el-submenu>
 </el-menu>
```

4.3 通过接口获取菜单数据

通过axios请求拦截器添加token，保证拥有获取数据的权限

```
// axios请求拦截
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
})
```

4.4 动态渲染菜单数据并进行路由控制

- 通过 v-for 双层循环分别进行一级菜单和二级菜单的渲染
- 通过路由相关属性启用菜单的路由功能

```
<el-menu router>
 <el-submenu :index="item.id + ''" v-for="item in menus" :key="item.id">
   <template slot="title">
     <span>{{item.authName}}</span>
   </template>
   <el-menu-item :index="'/' + subItem.path" v-for="subItem in item.children"
      :key="subItem.id" >
      <span slot="title">{{subItem.authName}}</span>
   </el-menu-item>
 </el-submenu>
</el-menu>
```

目录 Contents

5. 用户管理

5.1 用户管理概述

通过后台管理用户的账号信息，具体包括用户信息的展示、添加、修改、删除、角色分配、账号启用/注销等功能。

- 用户信息列表展示
- 添加用户
- 修改用户
- 删除用户
- 启用或禁用用户
- 用户角色分配

5. 用户管理

5.2 用户管理-列表展示

1. 用户列表布局

- 面包屑导航 el-breadcrumb
- Element-UI 栅格系统基本使用 el-row
- 表格布局 el-table、el-pagination

5.2 用户管理-列表展示

2. 用户状态列和操作列处理

- 作用域插槽
- 接口调用

```
<template slot-scope="scope">
  <!-- 开关 -->
  <el-switch v-model="scope.row.mg_state"
     @change="stateChanged(scope.row.id, scope.row.mg_state)">
  </el-switch>
</template>
```

5.2 用户管理-列表展示

3. 表格数据填充

- 调用后台接口
- 表格数据初填充

```
const { data: res } = await this.$http.get('users', { params: this.queryInfo })
if (res.meta.status !== 200) {
  return this.$message.error('查询用户列表失败！')
}
this.total = res.data.total
this.userlist = res.data.users
```

5.2 用户管理-列表展示

4. 表格数据分页

分页组件用法：

① 当前页码：pagenum
② 每页条数：pagesize
③ 记录总数：total
④ 页码变化事件
⑤ 每页条数变化事件
⑥ 分页条菜单控制

```
<el-pagination
    @size-change="handleSizeChange"
    @current-change="handleCurrentChange"
    :current-page="queryInfo.pagenum"
    :page-sizes="[2, 3, 5, 10]"
    :page-size="queryInfo.pagesize"
    layout="total, sizes, prev, pager, next, jumper"
    :total="total">
  </el-pagination>
```

5.2 用户管理-列表展示

5. 搜索功能

将搜索关键字，作为参数添加到列表查询的参数中。

```
<el-input
    placeholder="请输入搜索的内容"
    v-model="queryInfo.query"
    clearable
    @clear="getUserList">
    <el-button slot="append"
      icon="el-icon-search"
      @click="getUserList"></el-button>
 </el-input>
```

5.3 用户管理-用户状态控制

1. 开关组件的用法
2. 接口调用更改用户的状态

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

```
async stateChanged(id, newState) {
  const { data: res } = await this.$http.put(`users/${id}/state/${newState}`)
  if (res.meta.status !== 200) {
    return this.$message.error('修改状态失败！')
  }
}
```

5.4 用户管理-添加用户

1. 添加用户表单弹窗布局

- 弹窗组件用法
- 控制弹窗显示和隐藏

```
<el-dialog title="添加用户" :visible.sync="addDialogVisible" width="50%">
   <el-form :model="addForm" label-width="70px">
    <el-form-item label="用户名" prop="username">
      <el-input v-model="addForm.username"></el-input>
    </el-form-item>
    <!-- 更多表单项 -->
  </el-form>
  <span slot="footer" class="dialog-footer">
    <el-button @click="resetAddForm">取 消</el-button>
    <el-button type="primary" @click="addUser">确 定</el-button>
  </span>
</el-dialog>
```

5.4 用户管理-添加用户

2. 表单验证

内置表单验证规则

```
<el-form :model="addForm" :rules="addFormRules" ref="addFormRef" >
  <!-- 表单 -->
</el-form>
```

```
addFormRules: {
  username: [{ required: true, message: '请输入用户名', trigger: 'blur' }],
  password: [{ required: true, message: '请输入密码', trigger: 'blur' }],
}
```

```
this.$refs.addFormRef.validate(async valid => {
  if (!valid) return
})
```

5.4 用户管理-添加用户

2. 表单验证

自定义表单验证规则

```
const checkMobile = (rule, value, cb) => {
  let reg = /^(0|86|17951)?(13[0-9]|15[012356789]|17[678]|18[0-9]|14[57])[0-9]{8}$/
  if (reg.test(value)) {
    cb()
  } else {
    cb(new Error('手机号码格式不正确'))
  }
}
```

```
mobile: [
  { required: true, message: '请输入手机号', trigger: 'blur' },
  { validator: checkMobile, trigger: 'blur' }
]
```

5.4 用户管理-添加用户

3. 表单提交

将用户信息作为参数，调用后台接口添加用户

```
this.$refs.addFormRef.validate(async valid => {
  if (!valid) return
  const { data: res } = await this.$http.post('users', this.addForm)
  if (res.meta.status !== 201) {
    return this.$message.error('添加用户失败！')
  }
  this.$message.success('添加用户成功！')
  this.addDialogVisible = false
  this.getUserList()
})
```

5.5 用户管理-编辑用户

1. 根据 ID 查询用户信息

```
<el-button type="primary" size="mini" icon="el-icon-edit"
            @click="showEditDialog(scope.row.id)"></el-button>
```

```
async showEditDialog(id) {
  const { data: res } = await this.$http.get('users/' + id)
  if (res.meta.status !== 200) {
    return this.$message.error('查询用户信息失败！')
  }
  // 把获取到的用户信息对象，保存到 编辑表单数据对象中
  this.editForm = res.data
  this.editDialogVisible = true
}
```

5.5 用户管理-编辑用户

2. 编辑提交表单

```
this.$refs.editFormRef.validate(async valid => {
  if (!valid) return
  // 发起修改的请求
  const { data: res } = await this.$http.put('users/' + this.editForm.id, {
    email: this.editForm.email,
    mobile: this.editForm.mobile
  })
  if (res.meta.status !== 200) {
    return this.$message.error('编辑用户信息失败！')
  }
  this.$message.success('编辑用户信息成功！')
  this.getUserList()
  this.editDialogVisible = false
})
```

5. 用户管理

5.6 用户管理-删除用户

```
<el-button type="danger" size="mini" icon="el-icon-delete"
           @click="remove(scope.row.id)"></el-button>
```

```
async remove(id) {
 // 询问是否要删除
 const confirmResult = await this.$confirm('此操作将永久删除该用户，是否继续?', '提示', {
   confirmButtonText: '确定',
   cancelButtonText: '取消',
   type: 'warning'
 }).catch(err => err)
 const { data: res } = await this.$http.delete('users/' + id)
 if (res.meta.status !== 200) return this.$message.error('删除用户失败！')
 this.$message.success('删除用户成功！')
 this.getUserList()
},
```

目录 Contents

传智播客旗下高端IT教育品牌