# 1.环境准备

（1）安装Node.js(Node.js 是一个开源的、跨平台的 JavaScript 运行时环境。

点击链接下载安装 <https://blog.csdn.net/WHF__/article/details/129362462>

安装完成后，打开cmd，分别输入node -v 和 npm -v , 出现以下信息

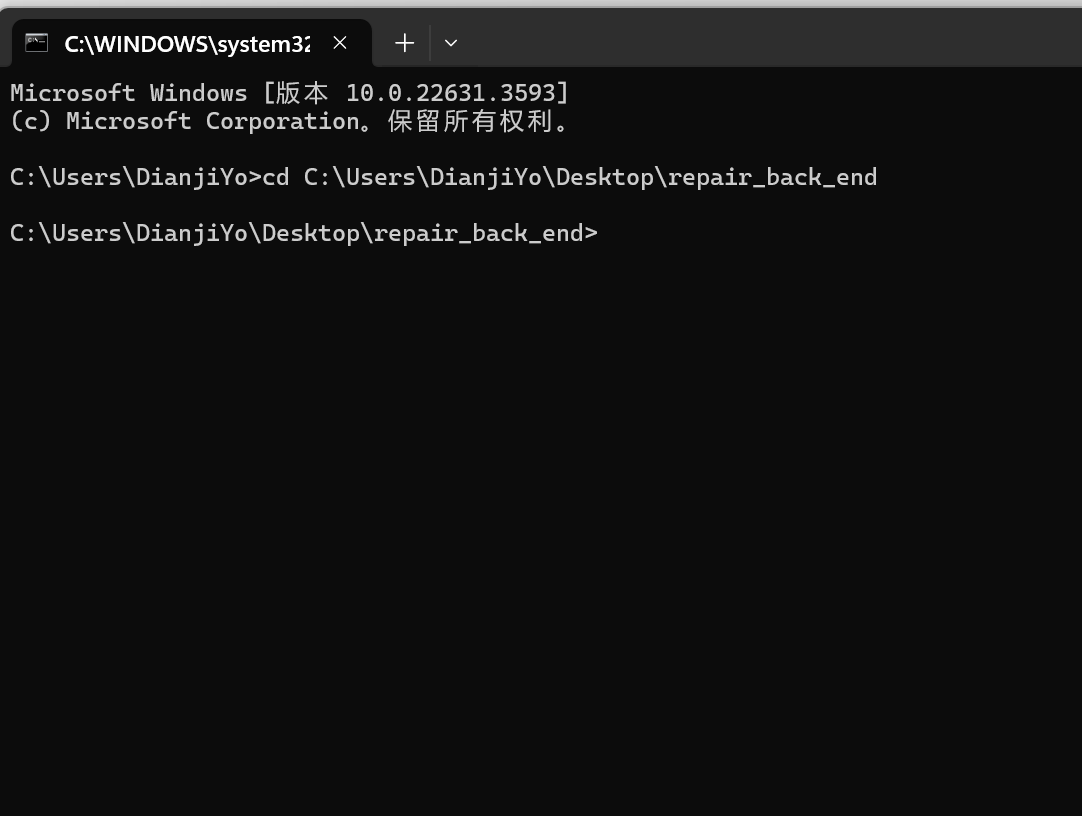


（2）安装Vue3

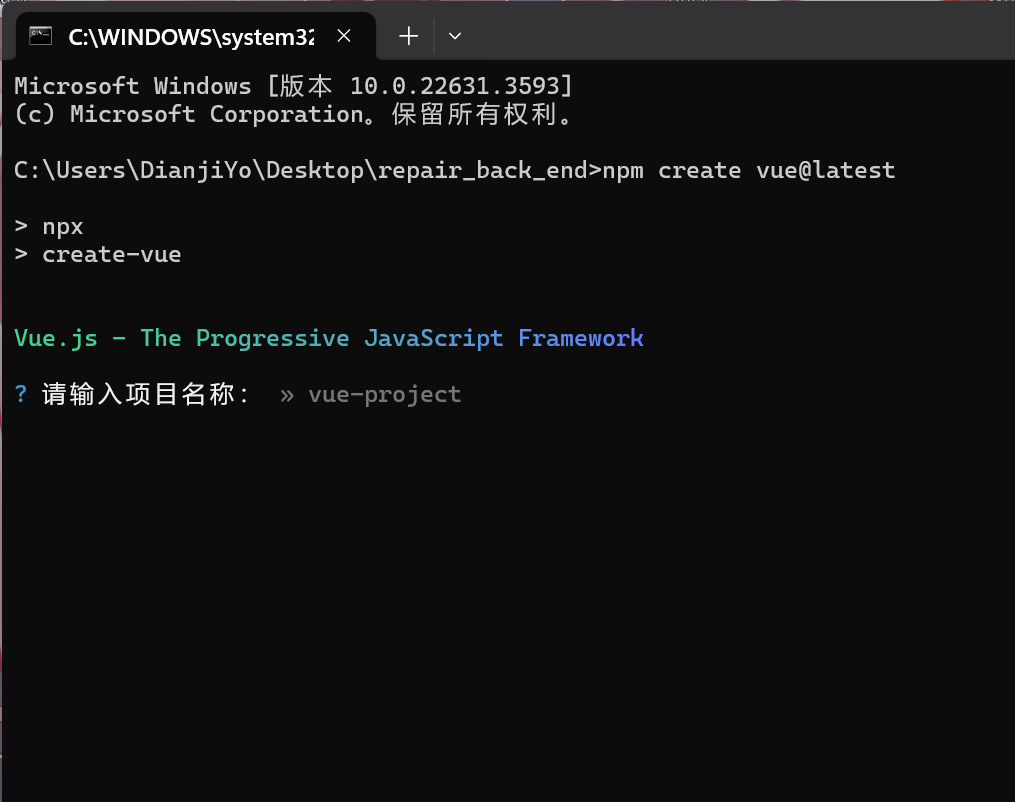
Vue3可以通过npm方式安装（NPM是包含在Node.js里的包管理工具，它具有以下功能：允许用户从NPM服务器下载别人编写的第三方包到本地使用。允许用户把自己编写的包或插件上传到NPM服务器供别人下载使用。来自各地的开源软件开发者可以使用 npm 互相分享和借鉴。包的结构使软件开发者能够轻松跟踪项目依赖项和版本。）

1. 同时按window+R键,输入cmd,打开命令提示符窗口
2. 新建一个文件夹 repair\_back\_end 做为你的vue3项目的目录
3. Cmd窗口输入 cd 文件夹路径 例如:

cd C:\Users\DianjiYo\Desktop\repair\_back\_end



1. 继续在cmd窗口输入 npm create vue@latest 然后回车

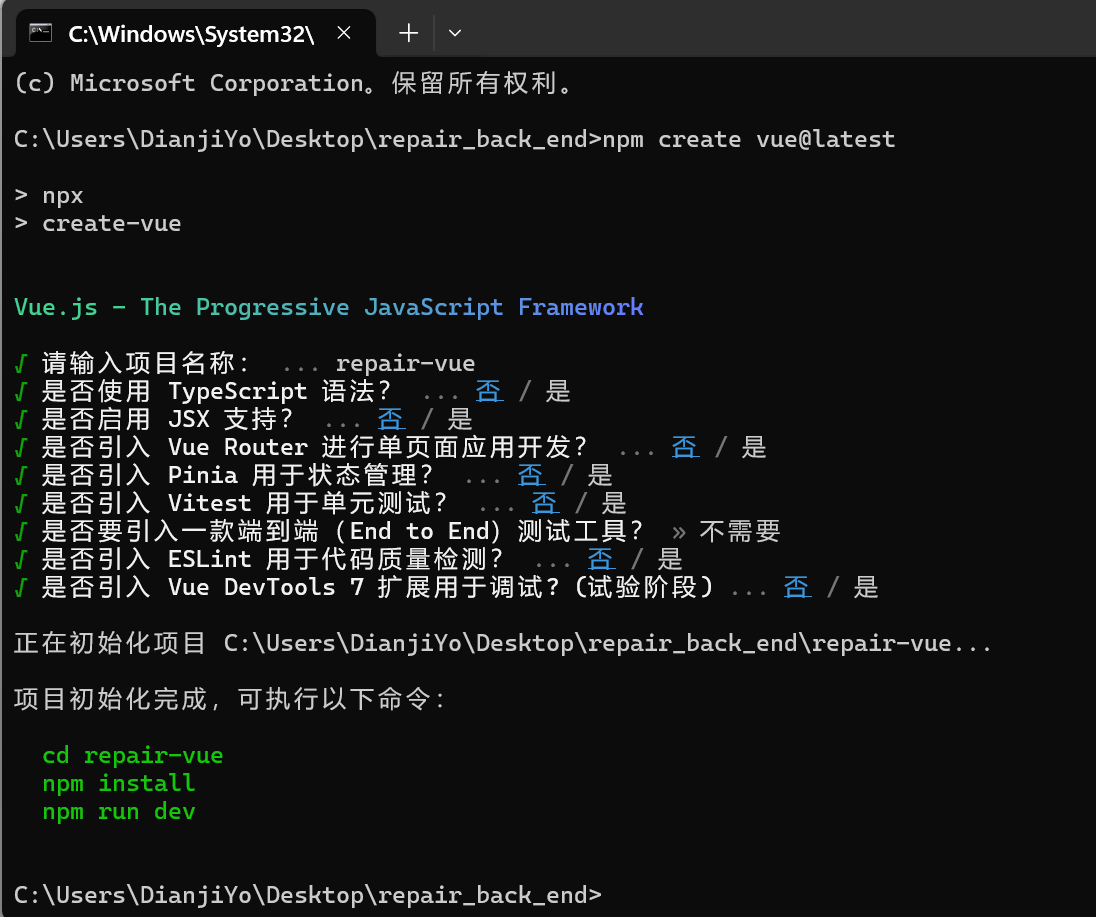


5.输入你的项目名称 例如 repair-vue 回车

 你将会看到一些诸如 TypeScript 和测试支持之类的可选功能提示：

1. √ Add TypeScript? ... No / Yes
2. √ Add JSX Support? ... No / Yes
3. √ Add Vue Router for Single Page Application development? ... No / Yes
4. √ Add Pinia for state management? ... No / Yes
5. √ Add Vitest for Unit Testing? ... No / Yes
6. √ Add an End-to-End Testing Solution? » No
7. √ Add ESLint for code quality? ... No / Yes

如果不确定是否要开启某个功能，你可以直接按下回车键选择 No。

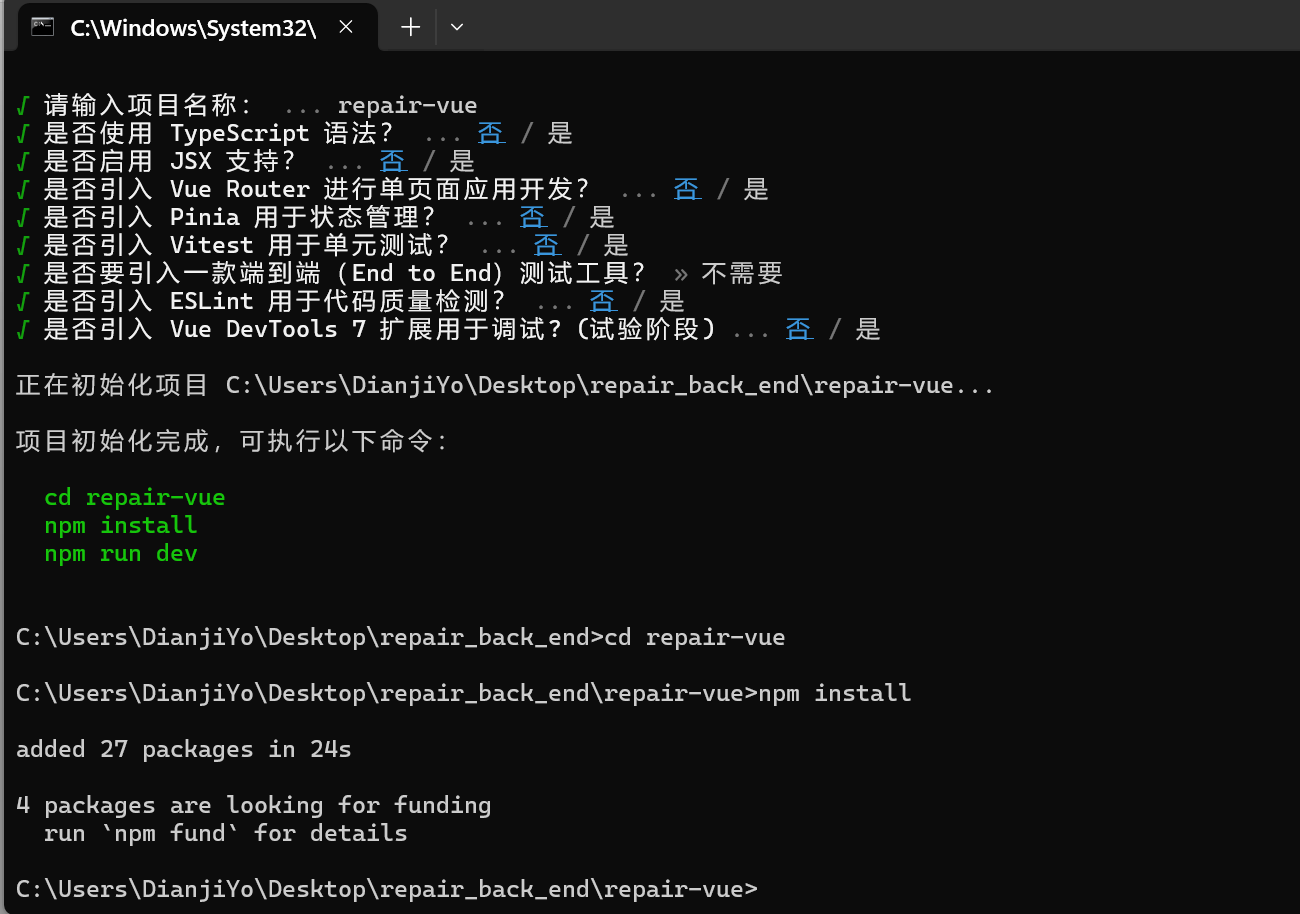


至此，项目创建完毕

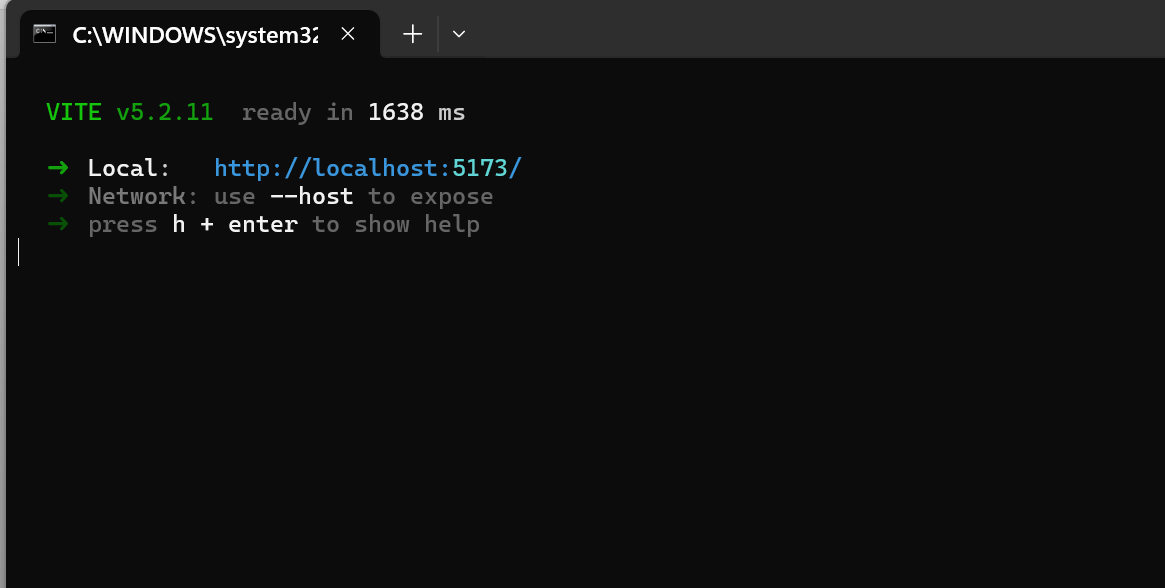
根据提示 依次输入

cd repair-vue 进入到项目目录

npm install 安装项目依赖

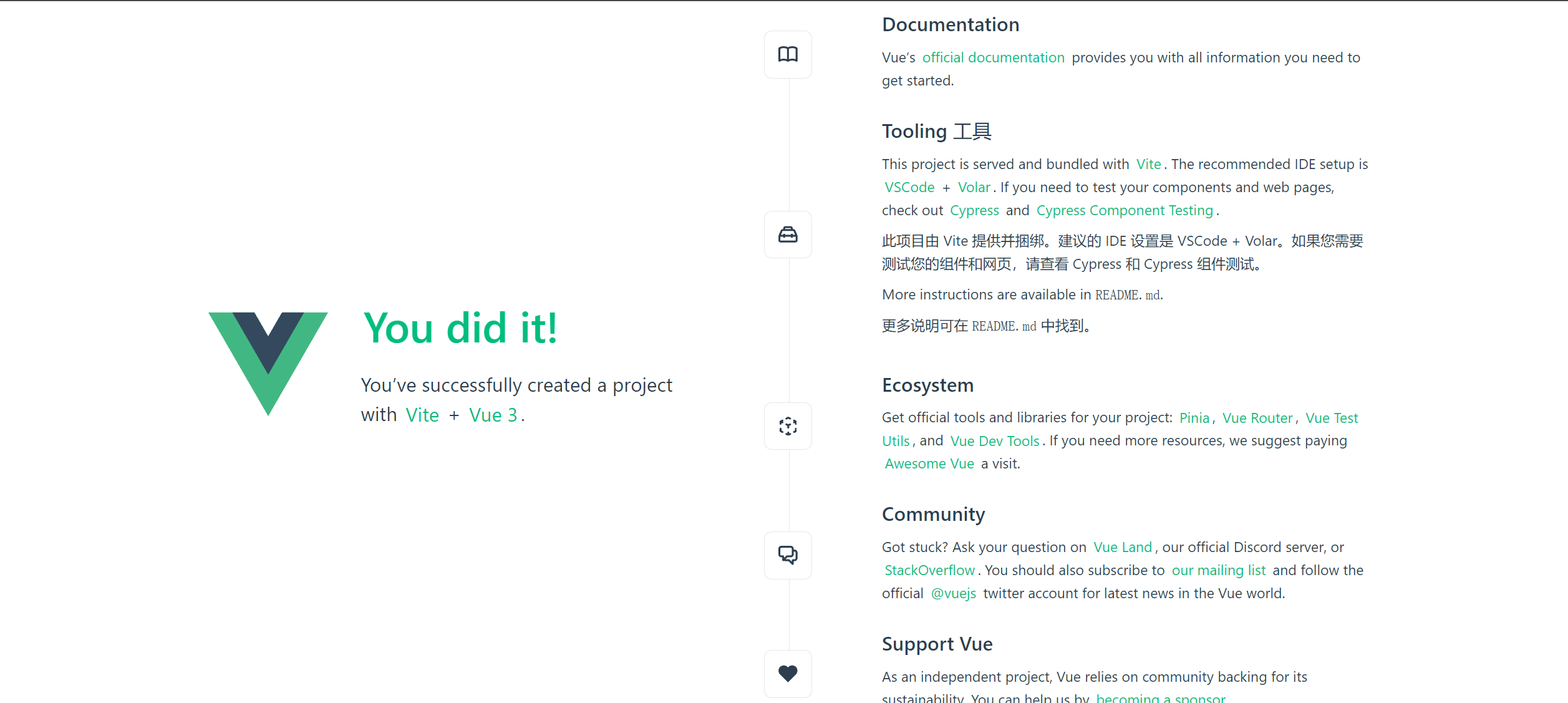


npm run dev 启动项目



我们在浏览器打开 <http://localhost:5173/>

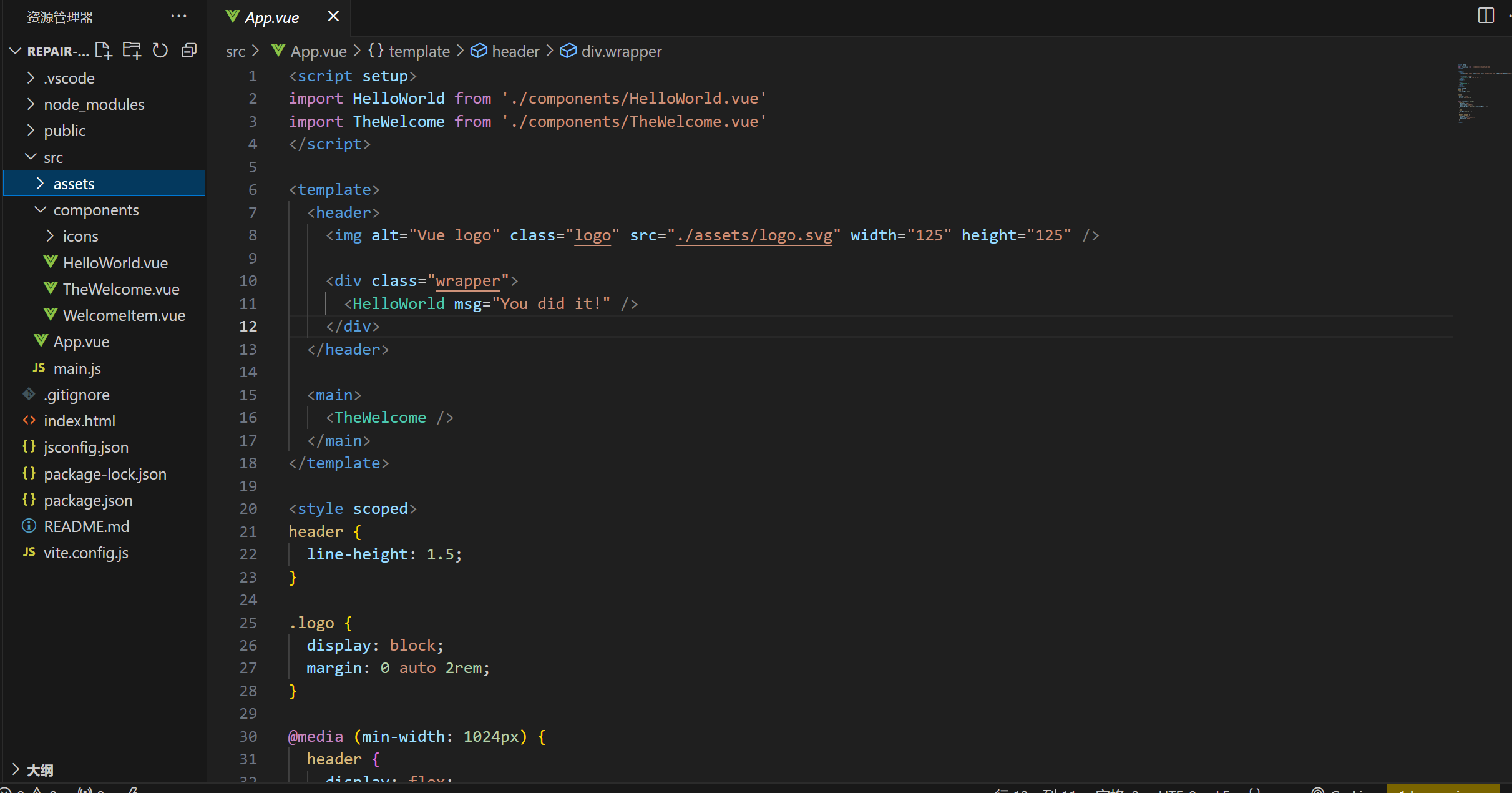
显示页面如下图：



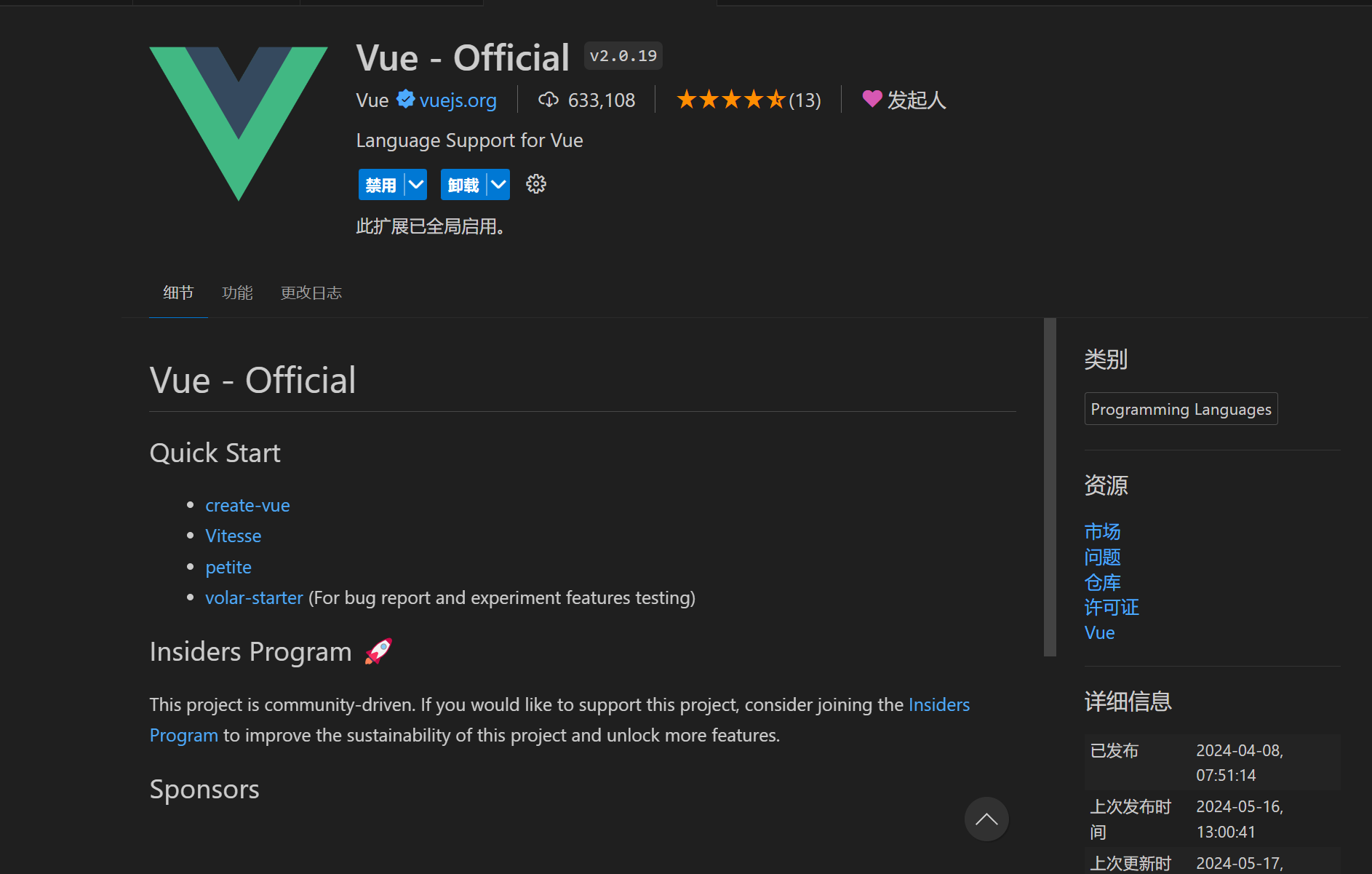
项目启动成功

# 2.用户登录

这里用编辑器打开项目文件夹（推荐VScode）

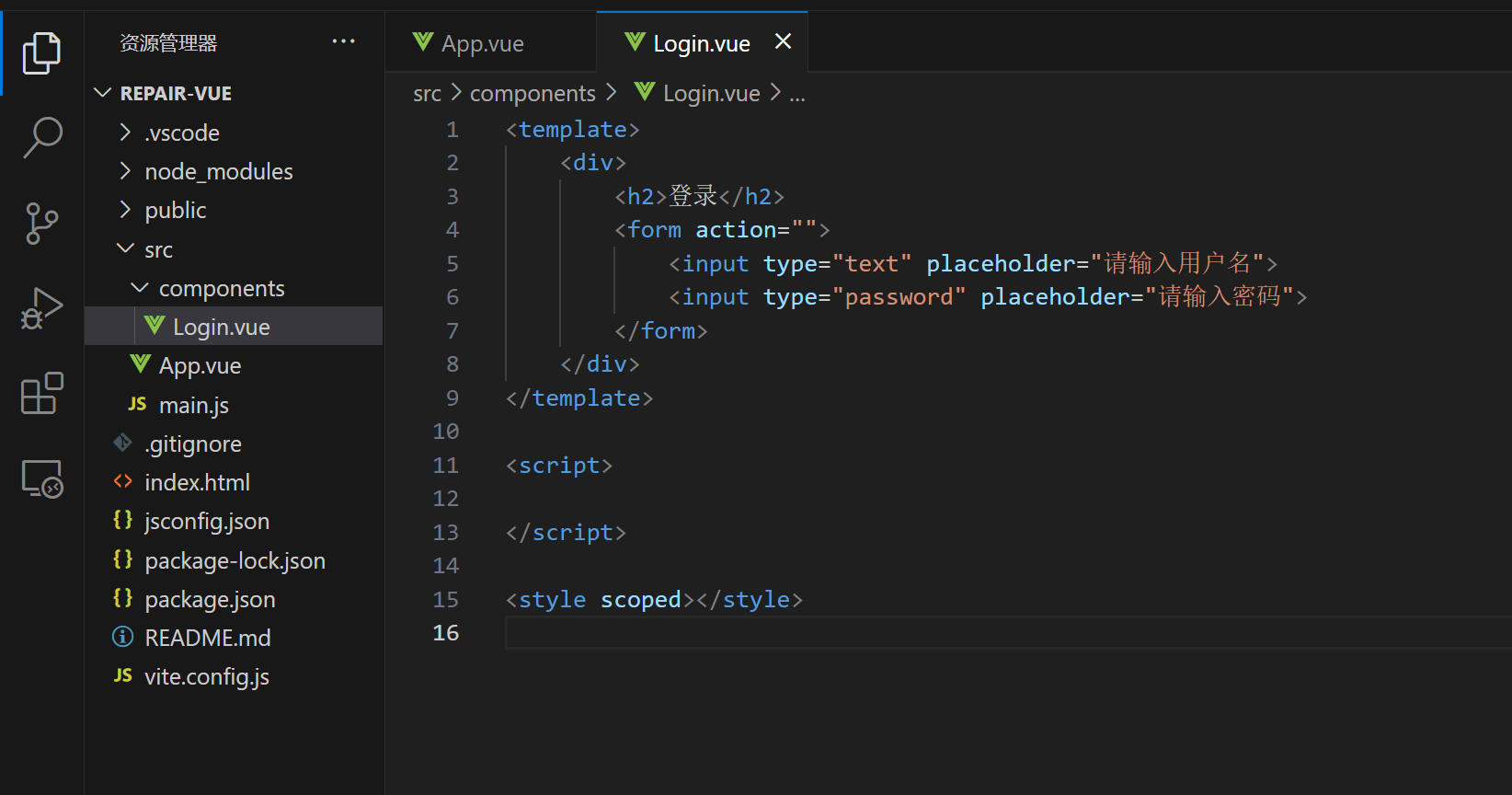


第一次打开项目可能会提示安装此插件，请同意安装，如未提示，请自行安装



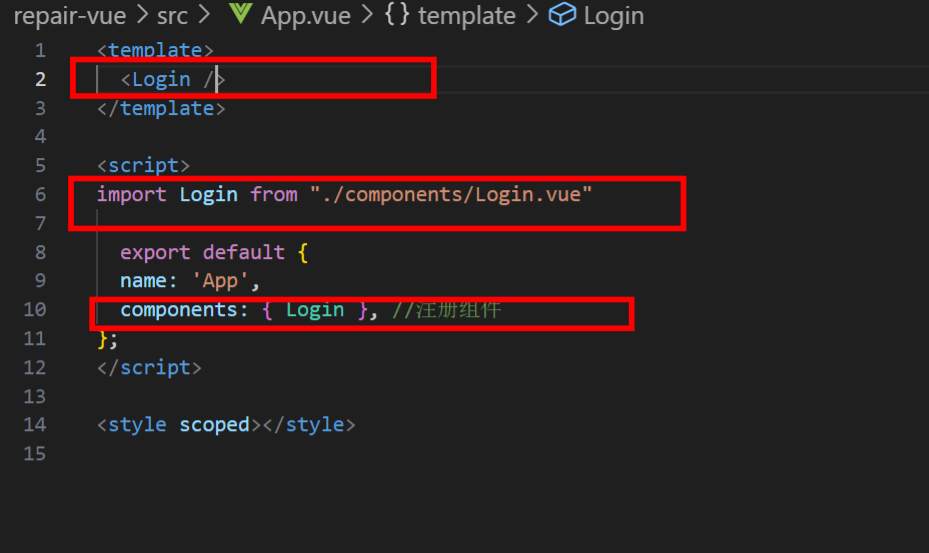
**单文件组件** (也被称为 \*.vue 文件，英文 Single-File Components，缩写为 **SFC**)。顾名思义，Vue 的单文件组件会将一个组件的逻辑 (JavaScript)，模板 (HTML) 和样式 (CSS) 封装在同一个文件里。

在components文件夹下创建Login.vue组件并编写HTML



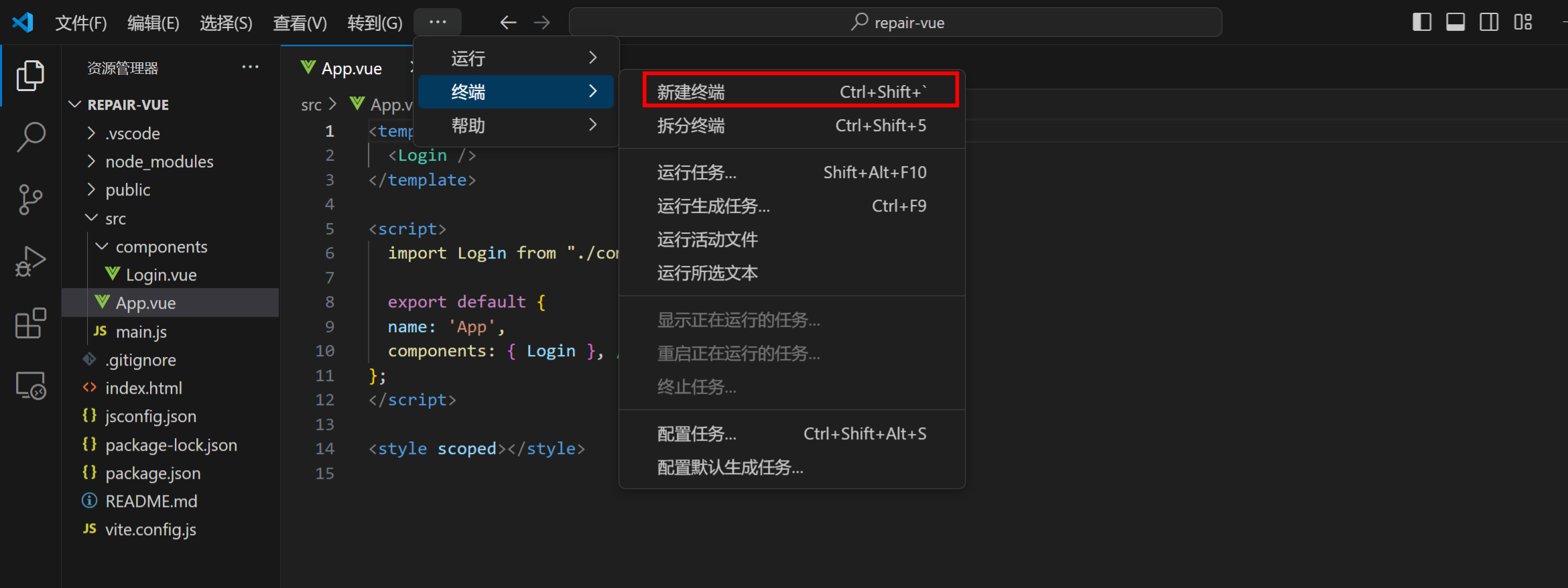
现在我们要把这个登录表单展示在页面上：

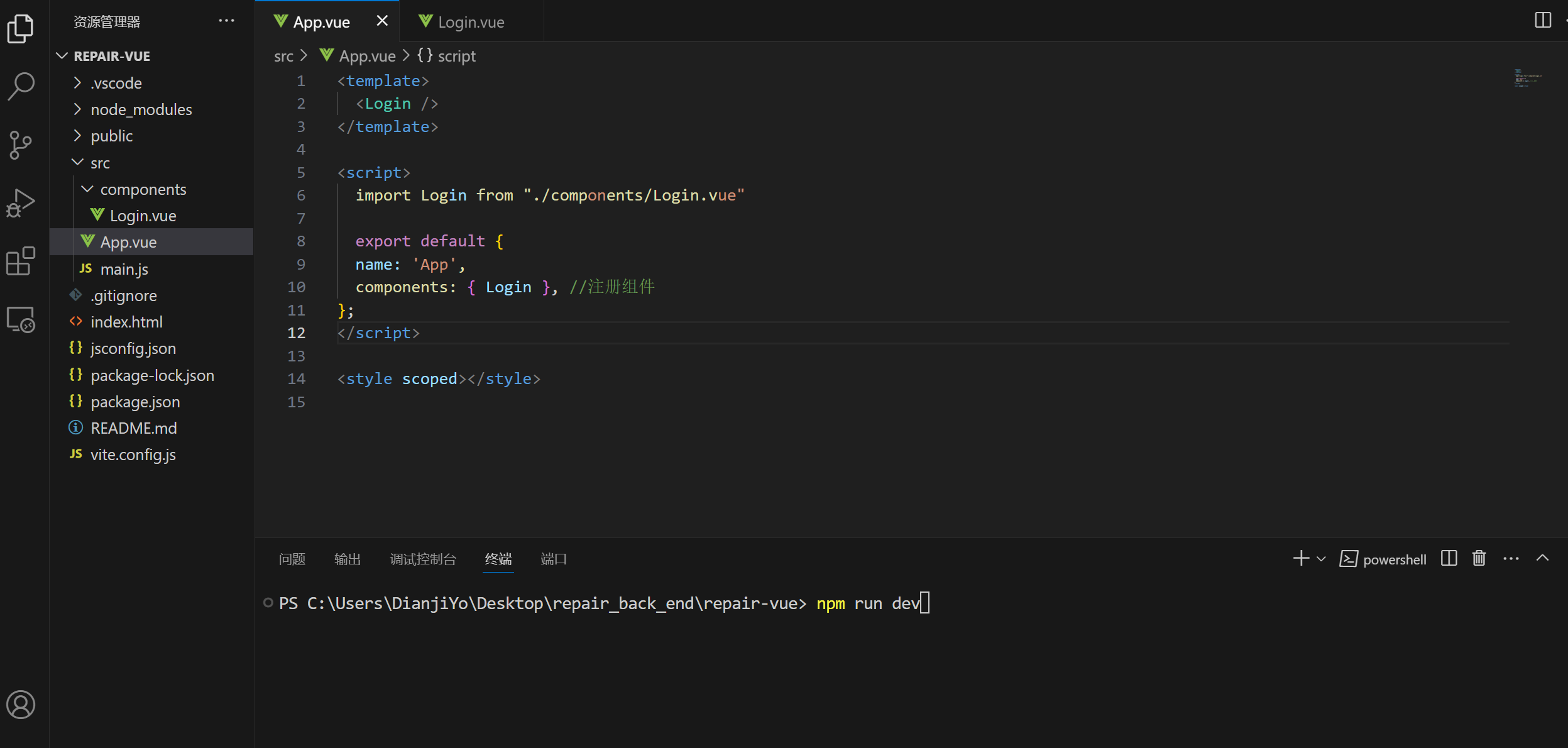
在App.vue根组件里添加如下代码

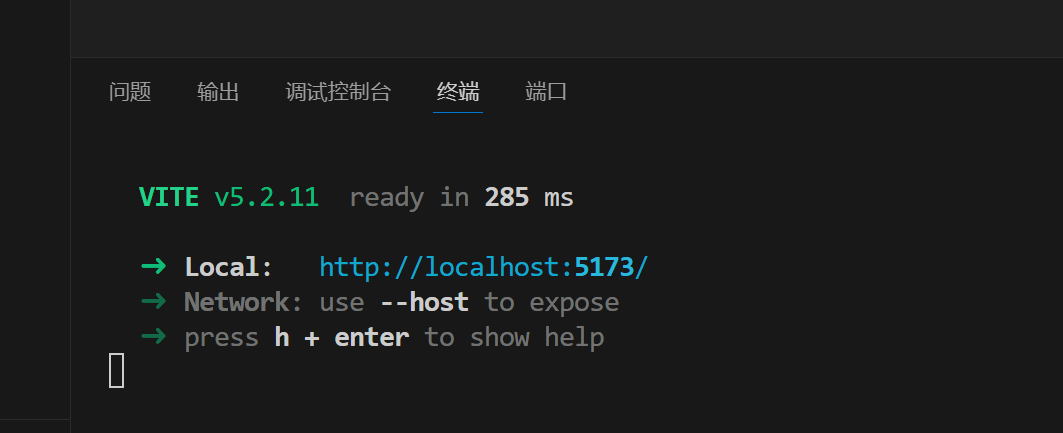


template标签里的<Login />与注册的组件名称对应

然后我们打开终端 输入 npm run dev







运行成功后浏览器访问 <http://localhost:5173/>



Login.vue 组件加载成功

前面我们简单实现了表单展示，现在我们要实现通过表单输入账号密码，发送请求，访问后端接口文件

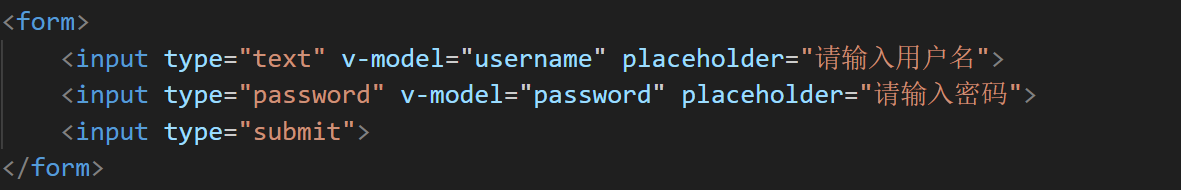
首先我们获取表单输入的数据

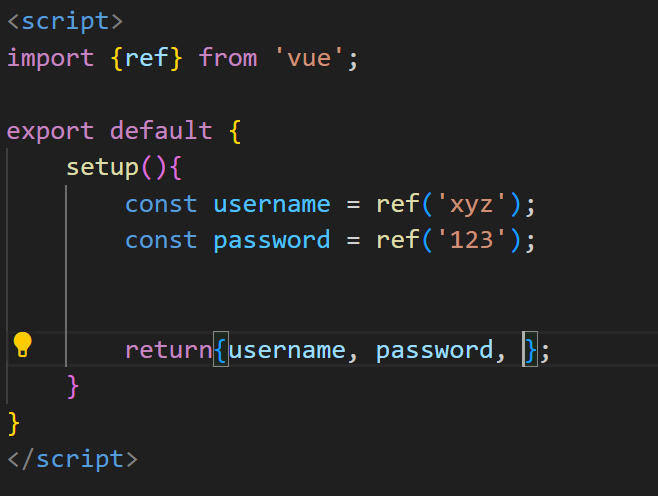
因为Vue的数据是动态更新的，所以这里要使用Vue下的一个指令v-model

v-model 是 Vue 中用于实现双向数据绑定的指令。它通常用于表单元素（如 <input>、<select>、<textarea> 等）上，可以将表单元素的值与 Vue 实例中的数据进行双向绑定。

当您在一个表单元素上使用 v-model 指令时，Vue 会自动为该表单元素创建一个 value 属性或者 checked 属性，并将其与指定的数据属性进行双向绑定。这意味着当用户在表单元素中输入内容或者做出选择时，相关的数据属性会自动更新；反之，如果您在代码中修改了相关的数据属性，表单元素也会相应地更新显示。

添加以下代码





export default 是用于导出一个模块的默认导出。当你在一个 Vue 组件中使用 export default 时，它用于导出该组件的定义，以便在其他地方导入和使用该组件。

具体来说，在一个 Vue 组件文件中，你可以将组件的定义（包括模板、脚本和样式）封装在一个单独的文件中，然后使用 export default 导出这个组件，让其他文件可以导入并在其它组件或应用中使用这个组件。

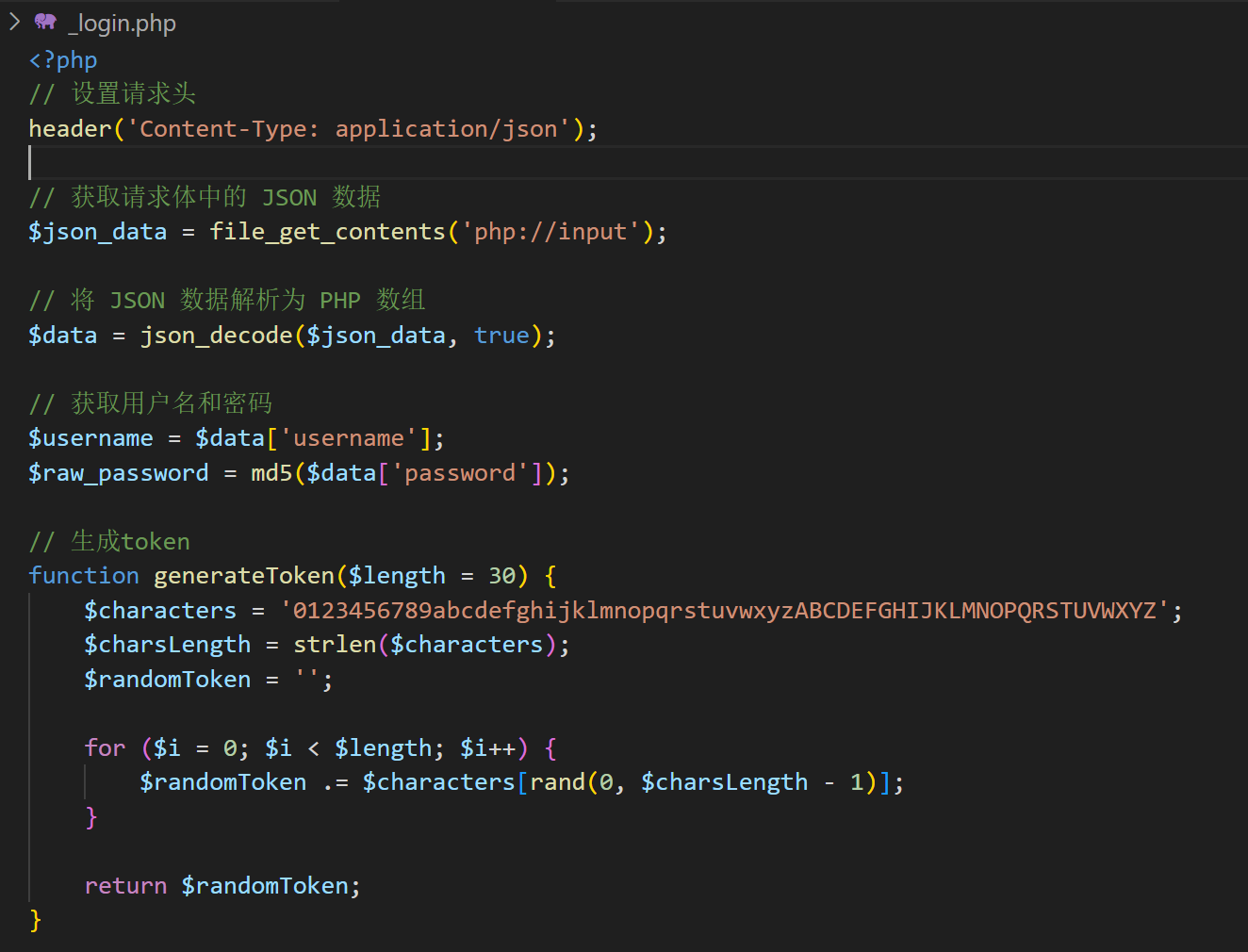
setup()函数返回一个包含响应式变量和函数的对象，这些变量和函数可以在模板中使用

这里我们就实现了数据的双向绑定



编写后端文件

在WAMP的www目录下新建admin文件夹 在里面新建\_login.php文件







登录后端编写完成，我们通过前端发送请求到后端调用登录接口

发送请求要用到axios

打开终端，输入npm install axios 回车



安装成功

添加以下代码实现发送请求



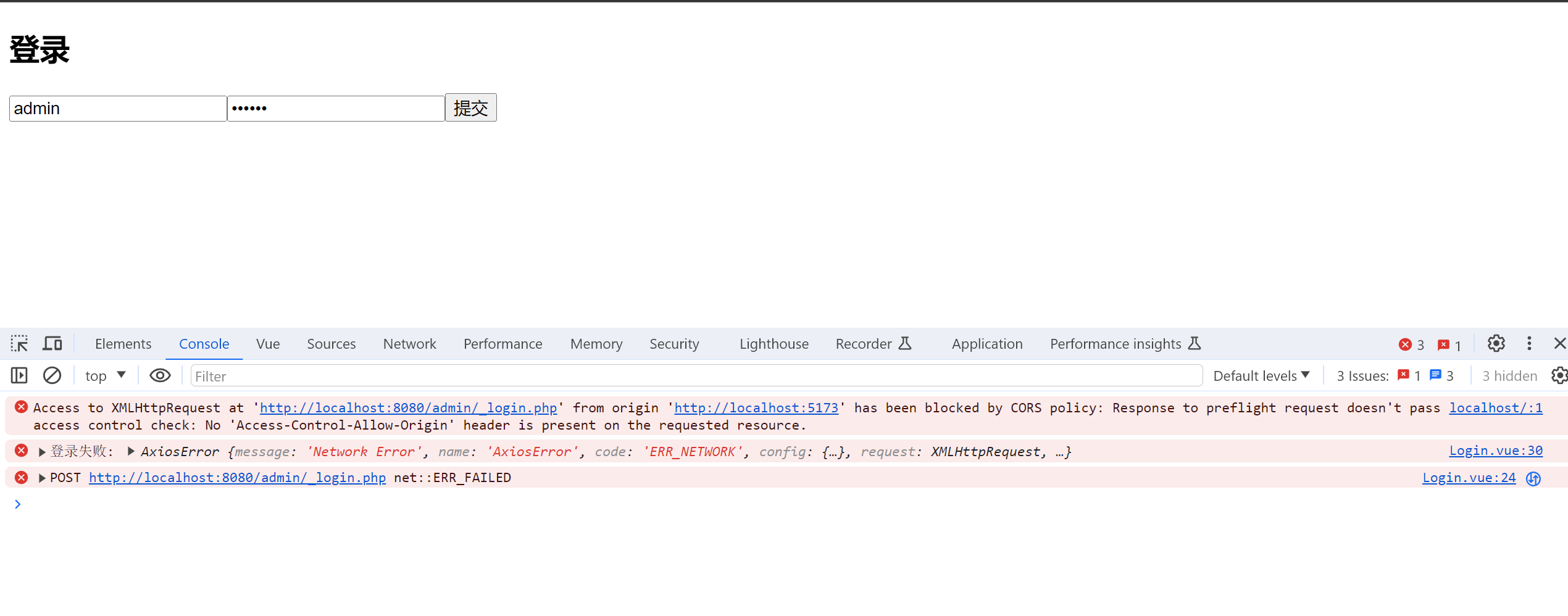
注：这里的8080改为5173

1. const login = async () => { ... };：定义了一个名为 login 的箭头函数，该函数使用了 async 关键字，表示它是一个异步函数。
2. try { ... } catch (error) { ... }：使用了异步函数的错误处理机制。在 try 代码块中，尝试执行发送登录请求到后端的操作；如果出现异常，则会跳转到 catch 代码块中处理异常。
3. const response = await axios.post('http://localhost:5173/api/admin/\_login.php', { username: username.value, password: password.value });：使用了 axios 库发送了一个 POST 请求到指定的 URL（'<http://localhost:5173/api/admin/_login.php>'），并传递了用户名和密码作为请求体数据（假设 username 和 password 是通过某种方式获取到的输入值）。

为表单提交事件绑定login函数



点击提交按钮，发送登录请求，出现以下错误

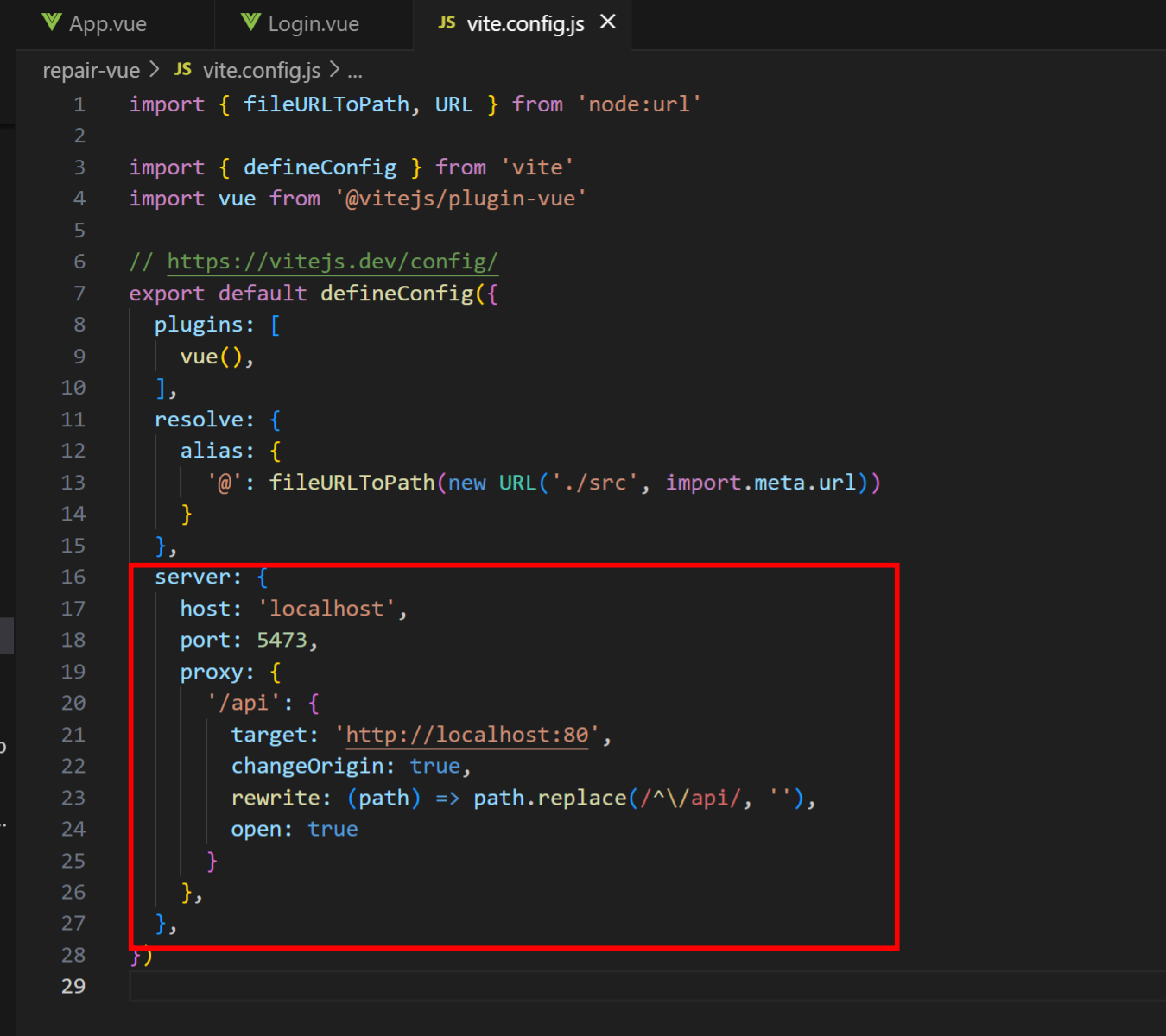


出现这个问题是因为涉及到资源跨域问题 具体可以参考这里

<https://blog.csdn.net/qq_38128179/article/details/84956552>

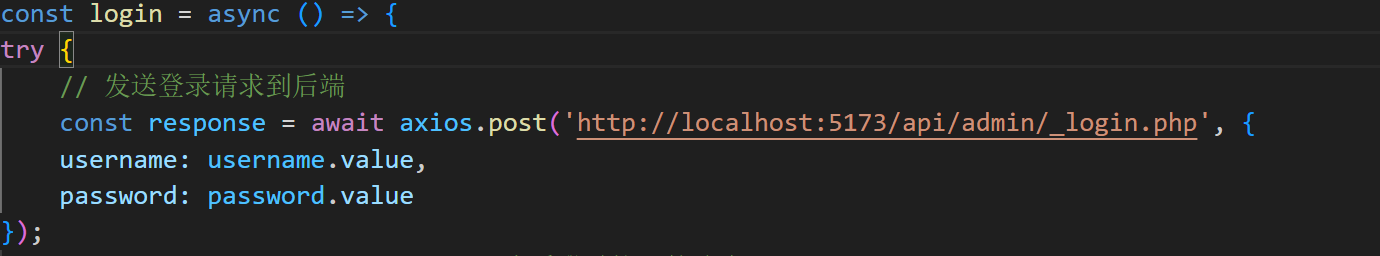
要解决跨域问题，需要做一个代理，通过虚拟的代理服务器访问不同域的网站

这里我们在项目根目录下找到vite.congig,js文件，添加以下代码



1. host: 'localhost'：指定了开发服务器的主机名为 localhost。
2. port: 5173：指定了开发服务器的端口号为 5173。
3. proxy: { '/api': { ... } }：使用了代理来处理以 /api 开头的请求。当开发服务器接收到以 /api 开头的请求时，会根据配置进行相应的转发操作。
4. target: 'http://localhost:80'：指定了代理目标的地址为 [http://localhost:80](http://localhost/)，即实际的后端服务器地址。
5. changeOrigin: true：设置为 true，表示开启跨域访问。当目标服务器不支持 CORS 跨域访问时，需要将此选项设置为 true。
6. rewrite: (path) => path.replace(/^\/api/, '')：定义了重写规则，将 /api 前缀从请求路径中移除，以便将请求发送给目标服务器时去掉 /api。
7. open: true：设置为 true，表示在服务器启动时自动打开浏览器。

修改请求链接



表单提交数据（确保wampserver已启动）



我们得到了接口返回的数据

# 3.订单列表展示

前面我们实现了前端发送登录请求到后端，接口响应数据返回前端。

登录成功后，要跳转到订单列表页面。在Vue中实现跳转需要用到路由。

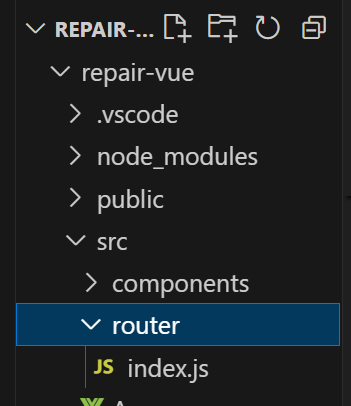
打开终端，输入npm install vue-router@next



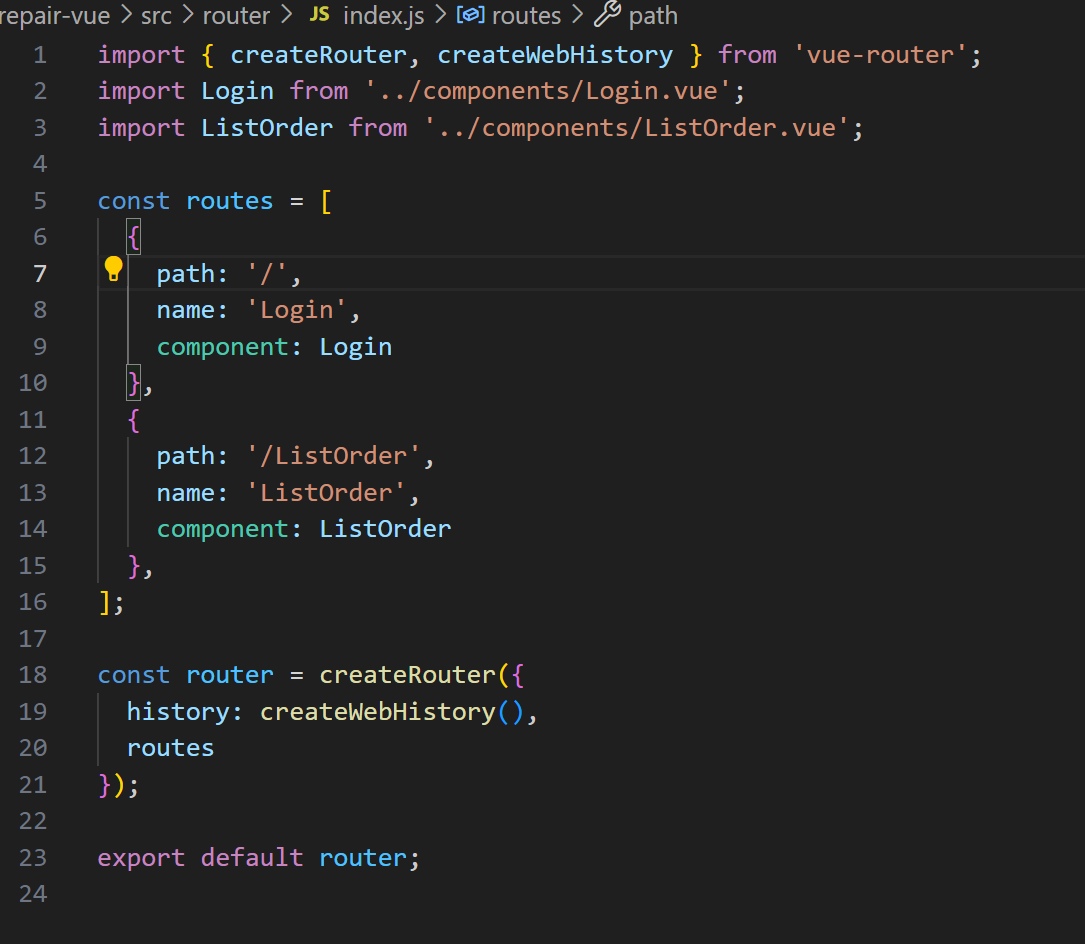
安装成功

接下来对路由进行配置

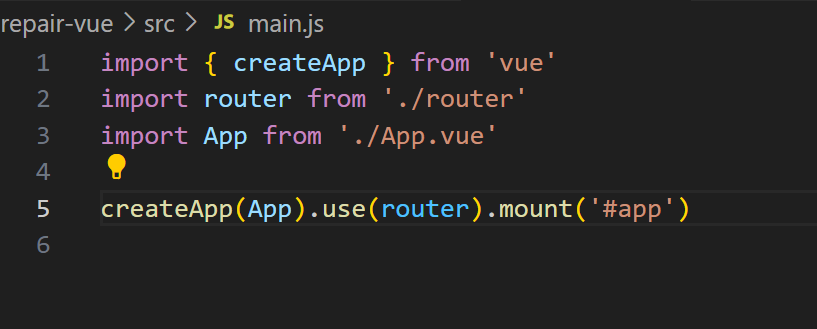
首先新建如下文件



在index.js下添加以下代码

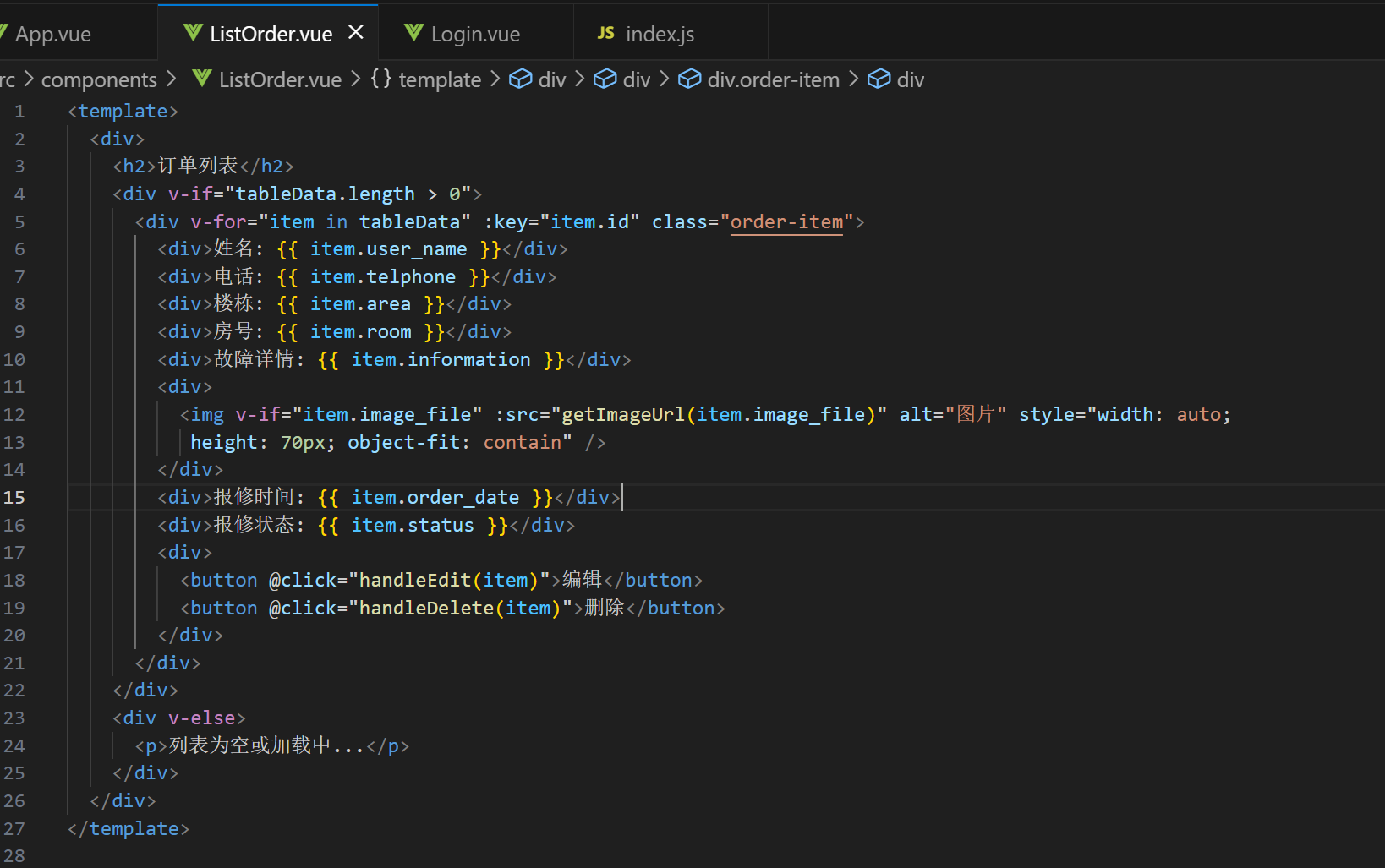


找到src文件夹下的main.js



然后编写ListOrder.vue 组件

在components文件夹下新建ListOrder.vue文件



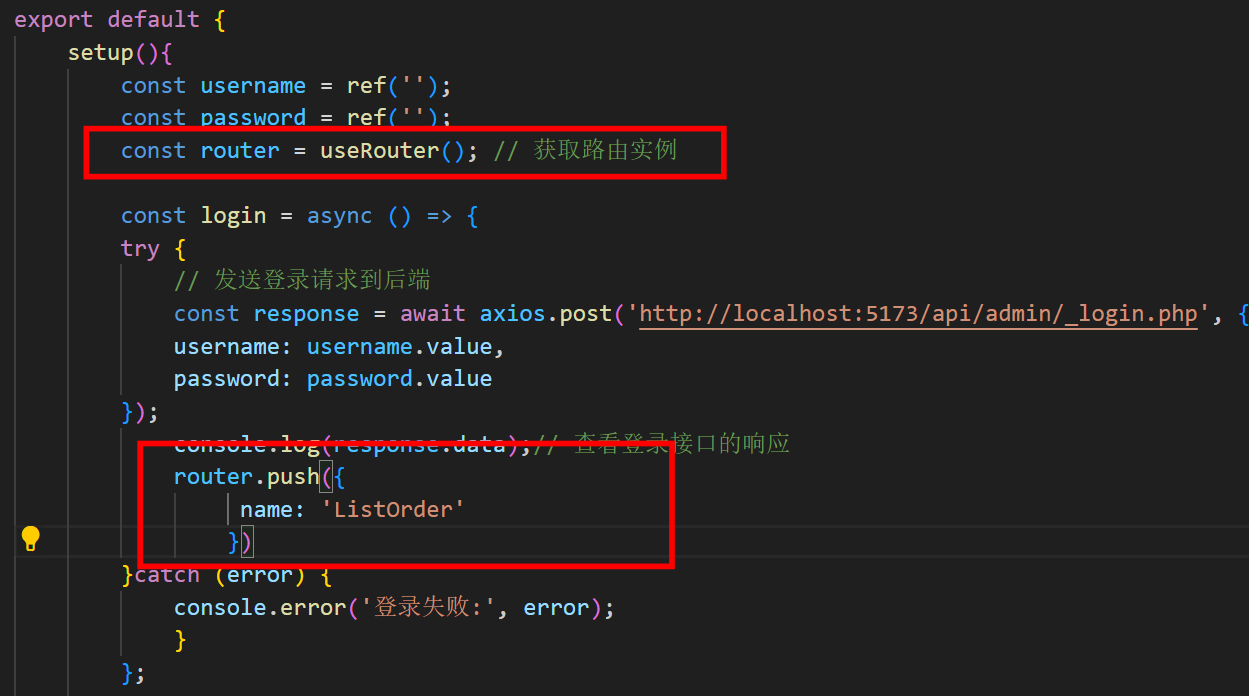




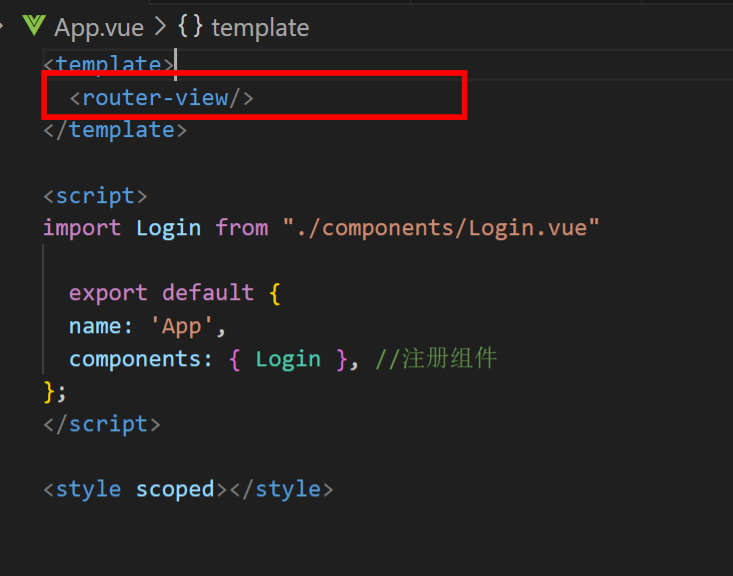
<style scoped>…</style> scoped表示改style只对当前组件有效

如要对组件添加样式，一定要带上scoped

回到Login.vue 添加以下代码



在App.vue 页面 添加以下代码



<router-view> 是 Vue Router 提供的一个组件，用于显示当前路由匹配到的组件内容

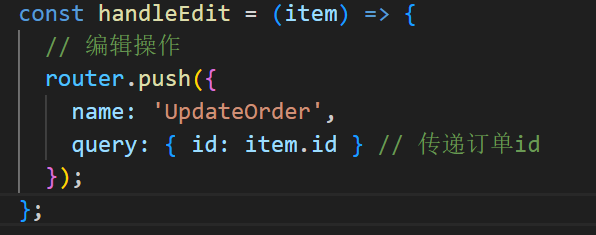
当你在 Vue 项目中使用了 Vue Router 并定义了多个路由时，<router-view> 将会根据当前 URL 的路径匹配到对应的路由配置，并渲染匹配到的组件内容。

返回浏览器，进行登录，成功跳转到订单列表页面



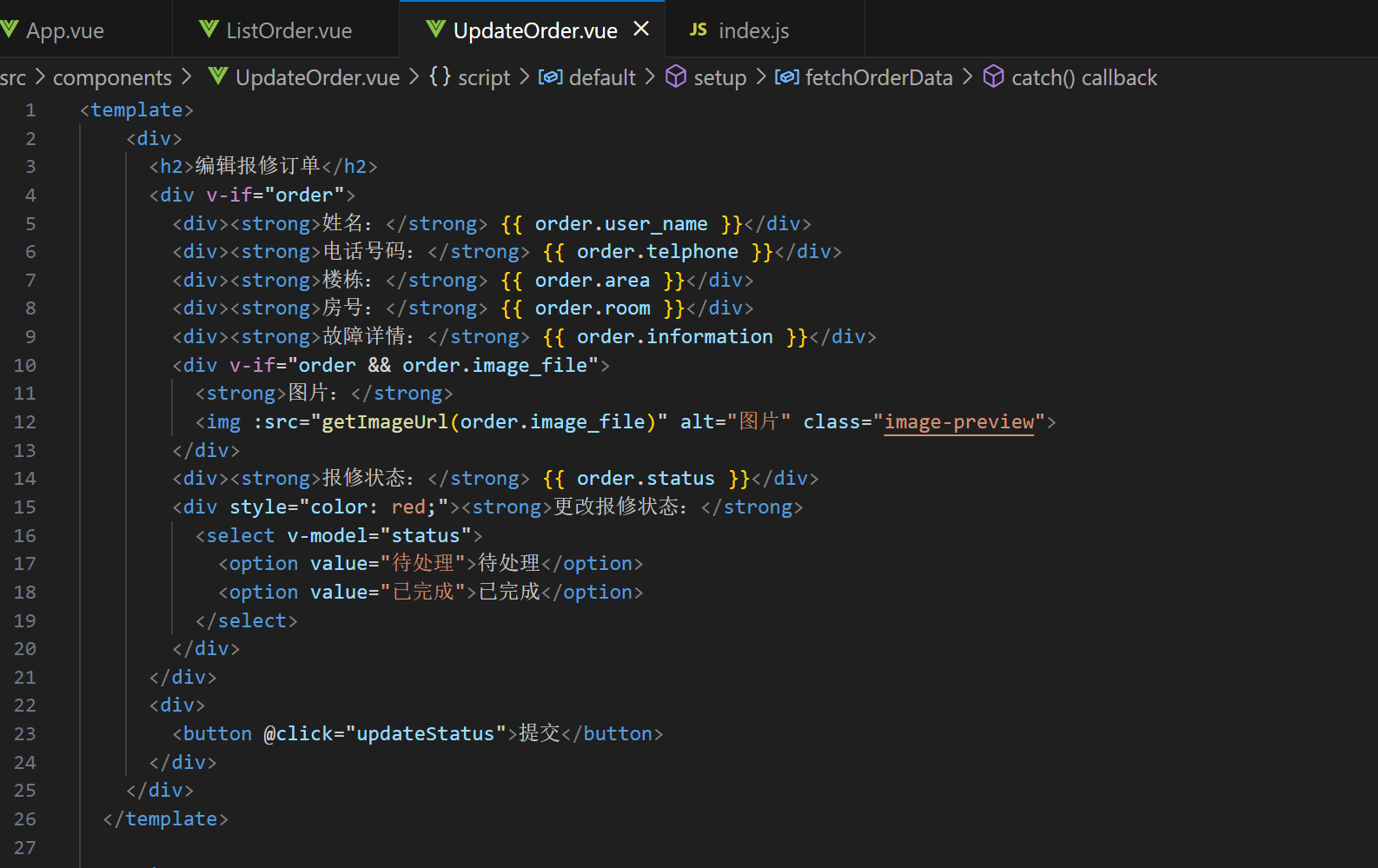
# 5.编辑订单

当点击某条订单的编辑按钮时，会传递该订单的id到UpdateOrder

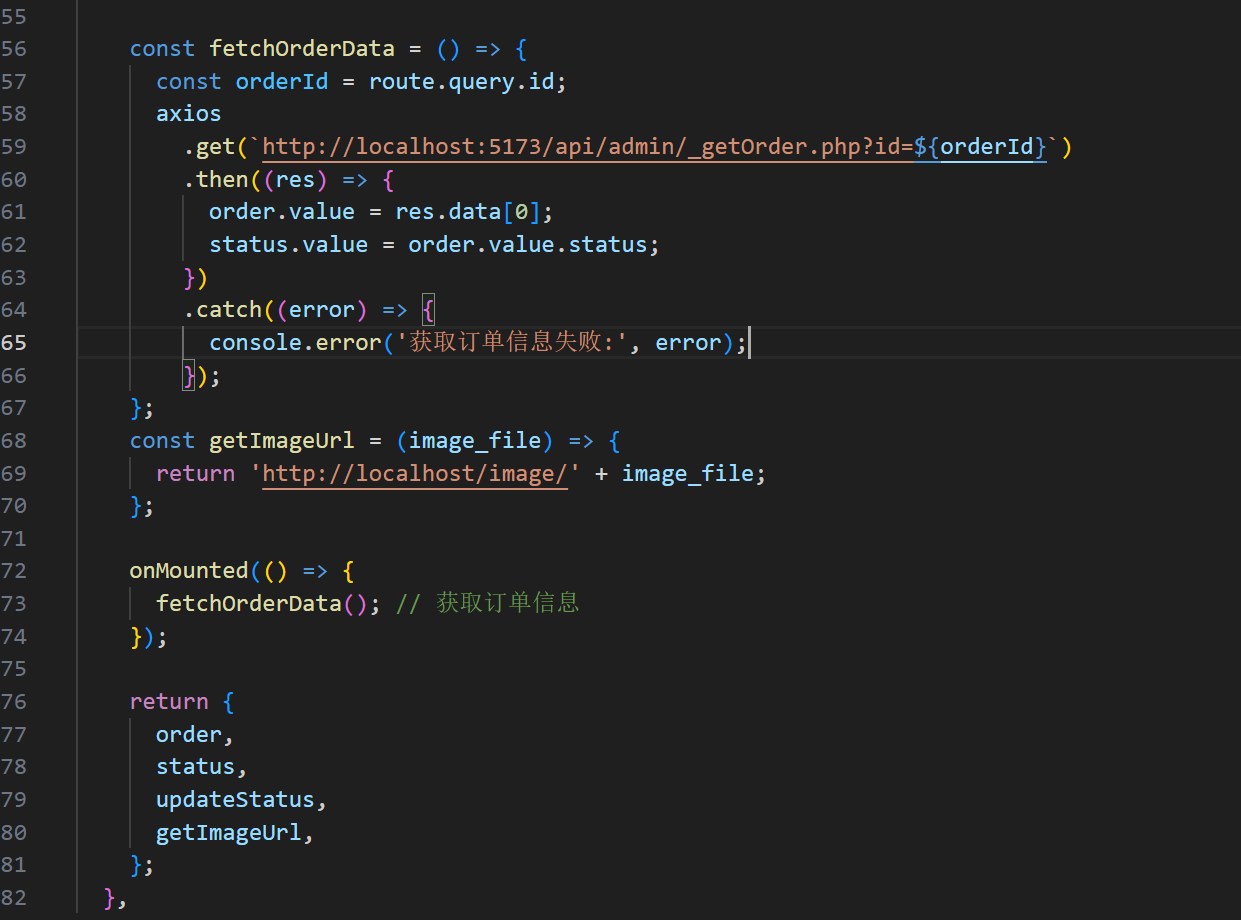


router.push 方法中传入包含 query 参数的对象，可以实现在路由跳转时传递查询参数，方便在目标组件中获取并使用这些参数。

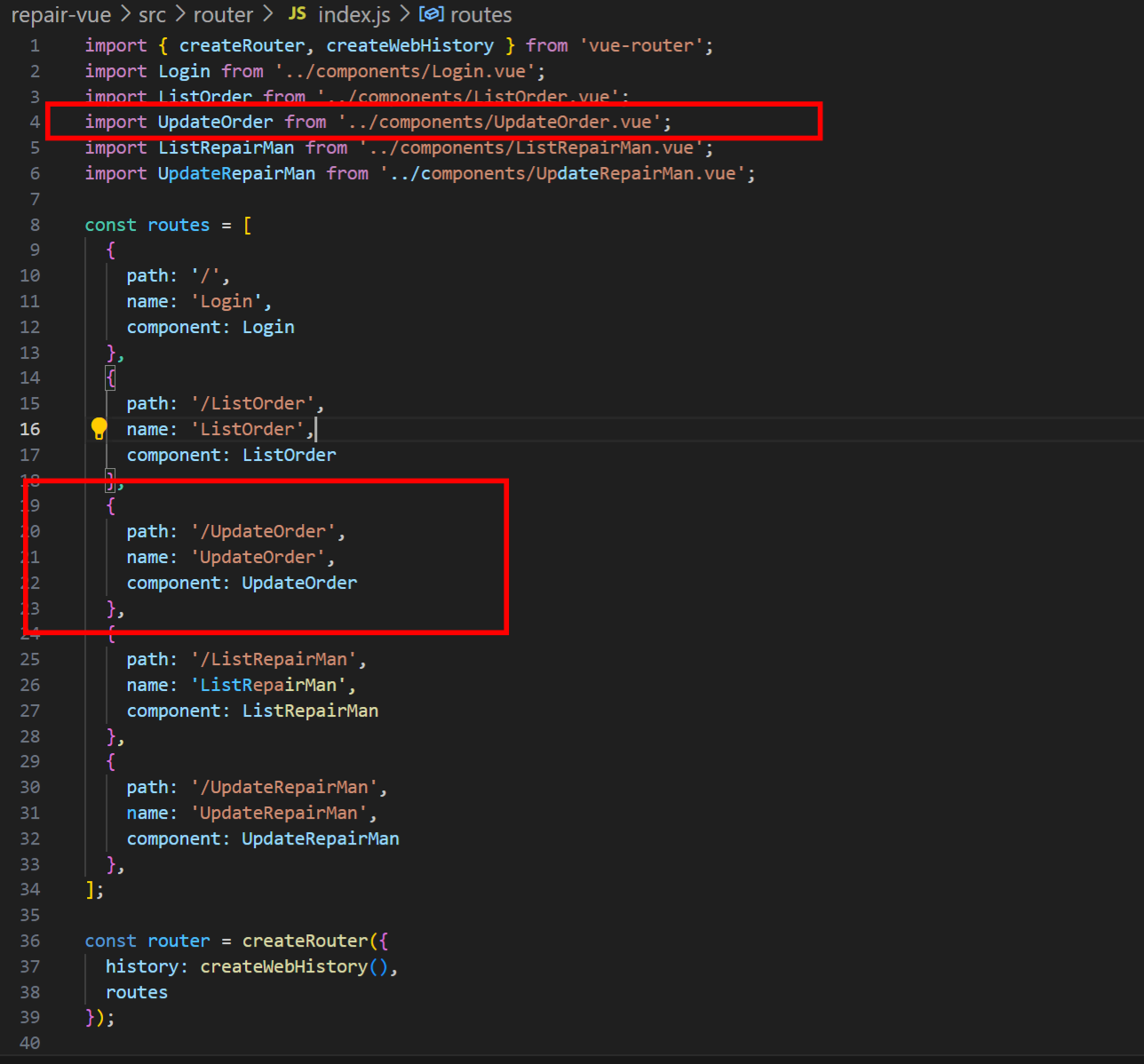
现在来编写UpdateOrder.vue 在components文件夹下新增UpdateOrder.vue



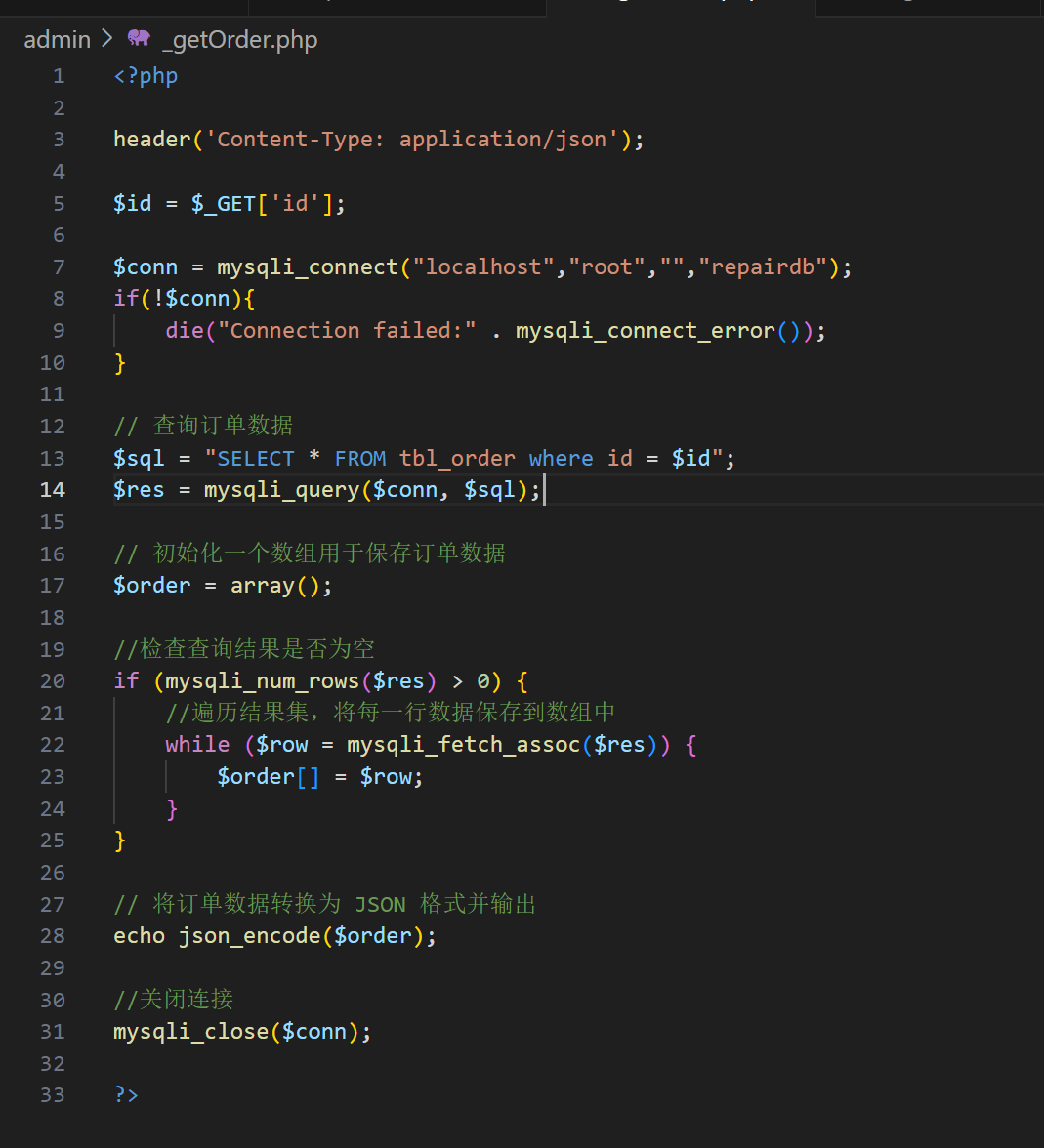




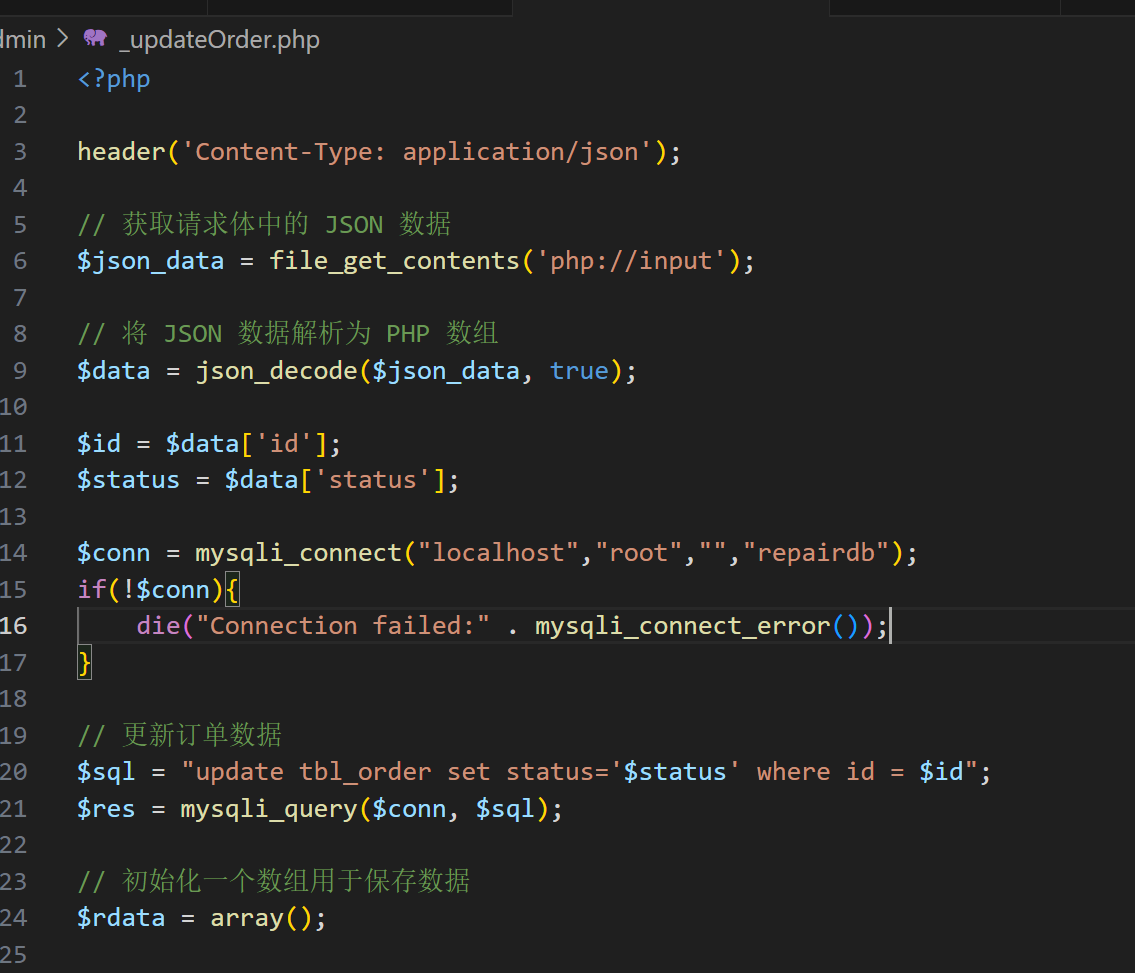
新增组件后，把组件添加到路由配置中



编写后端根据id获取订单接口



编写后端更新订单接口





点击任意订单编辑按钮，跳转到订单编辑页面，对报修状态进行修改，可以看到，修改成功





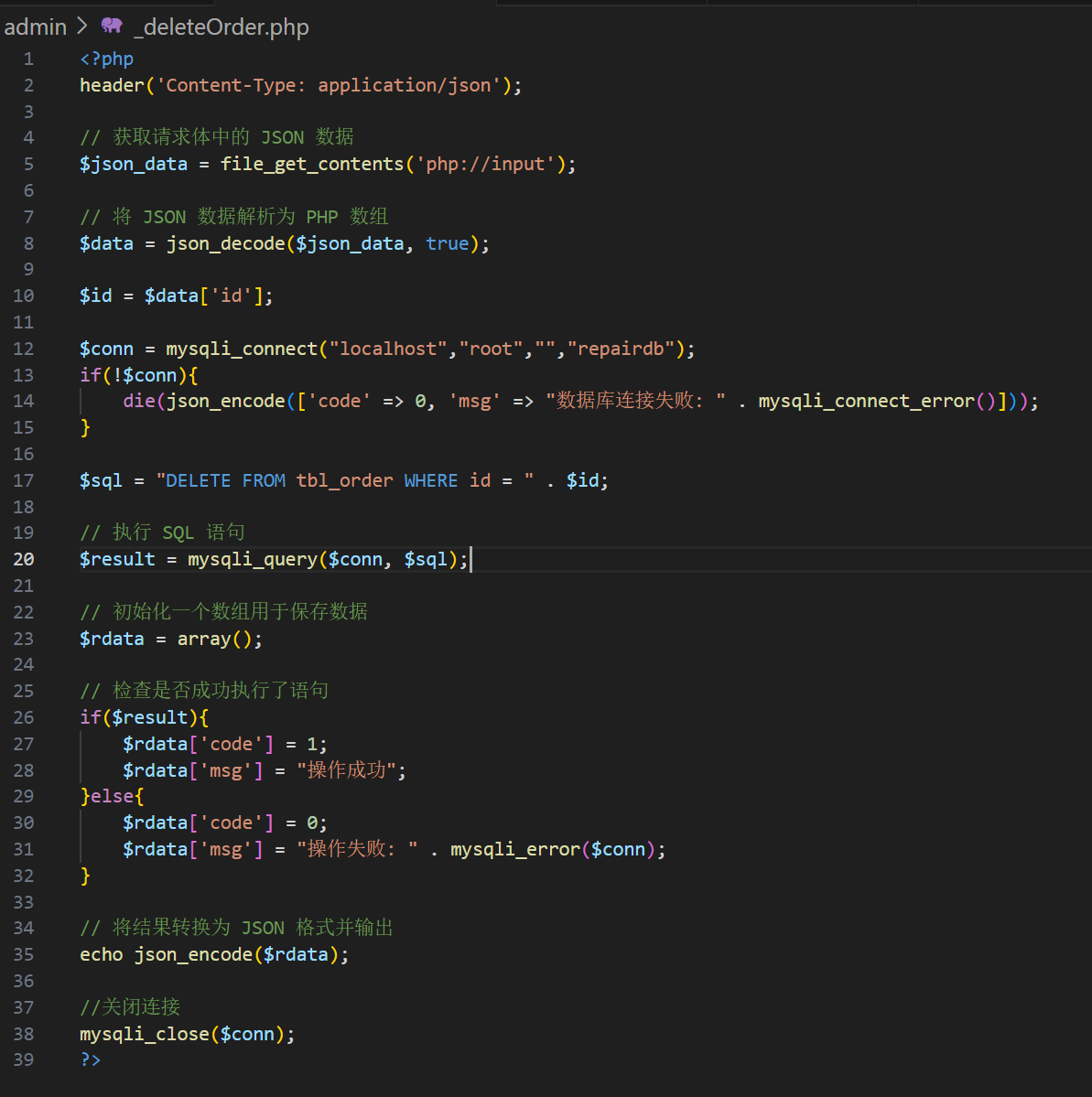
# 6.删除订单

回到ListOrder.vue



可以看到删除订单是通过发送请求调用\_deleteOrder.php实现的

编写后端删除订单接口，新建\_deleteOrder.php文件



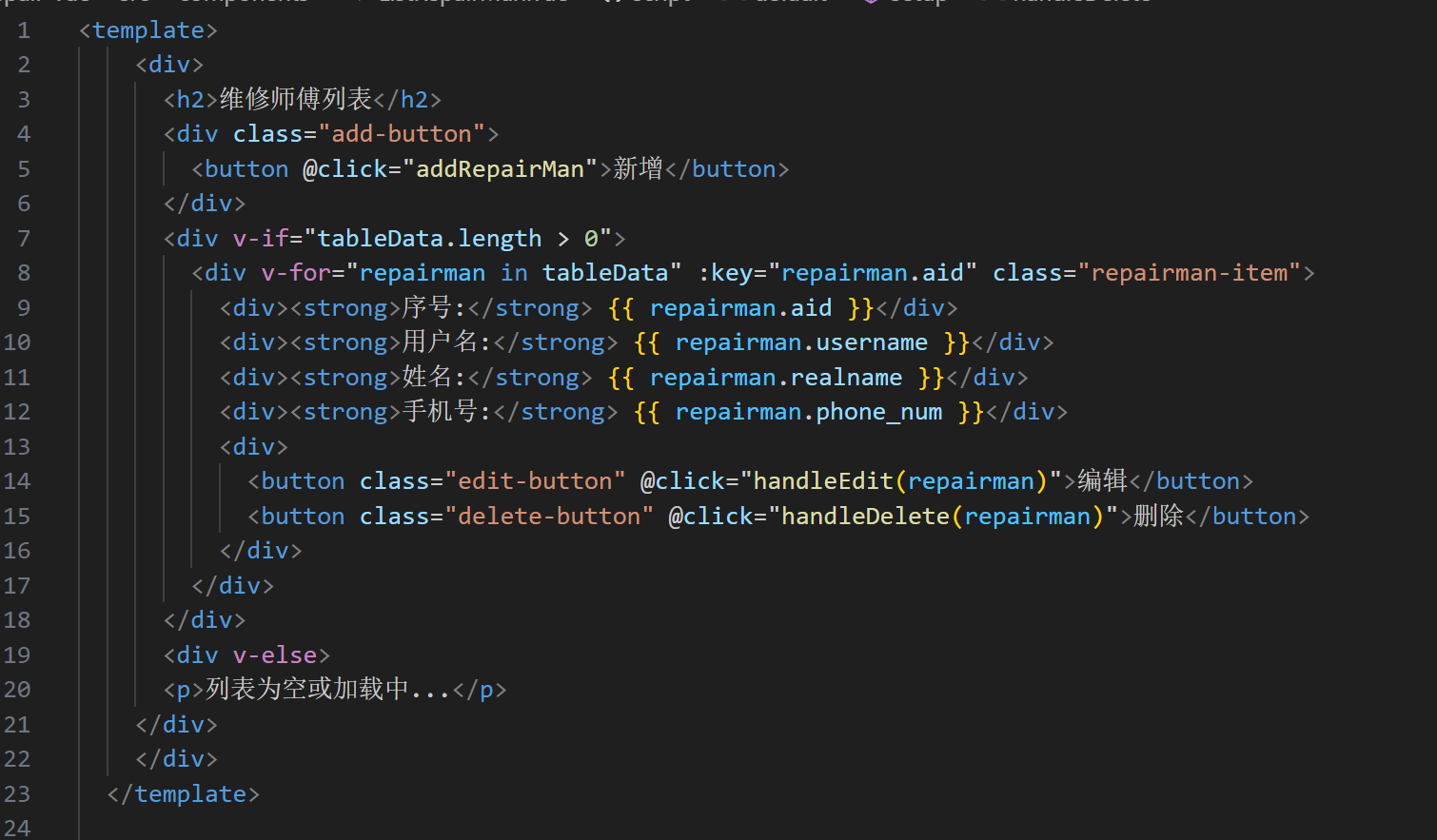
点击任一订单删除按钮，可以看到相应订单删除成功

# 7.维修师傅列表

前面我们完成了订单列表展示以及订单的修改和删除

现在我们来实现维修师傅列表展示

在components文件夹下新建ListRepairMan.vue 添加以下代码



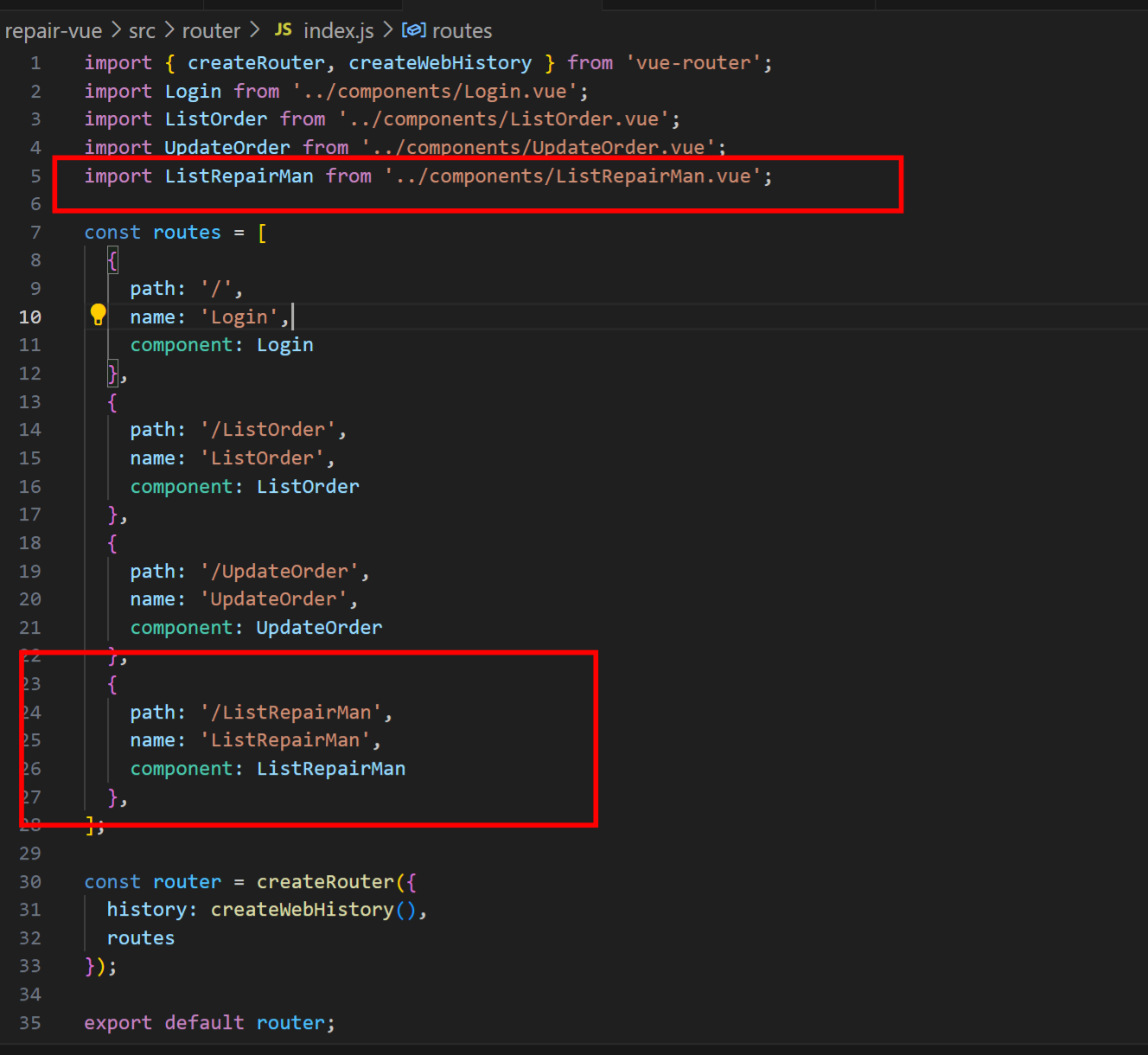




编写完成

我们发现现在并不能直接访问ListRepairMan.vue,所以我们增加一个路由导航，以便在订单列表和维修师傅之间切换

在router文件夹下的index.js里添加如下代码



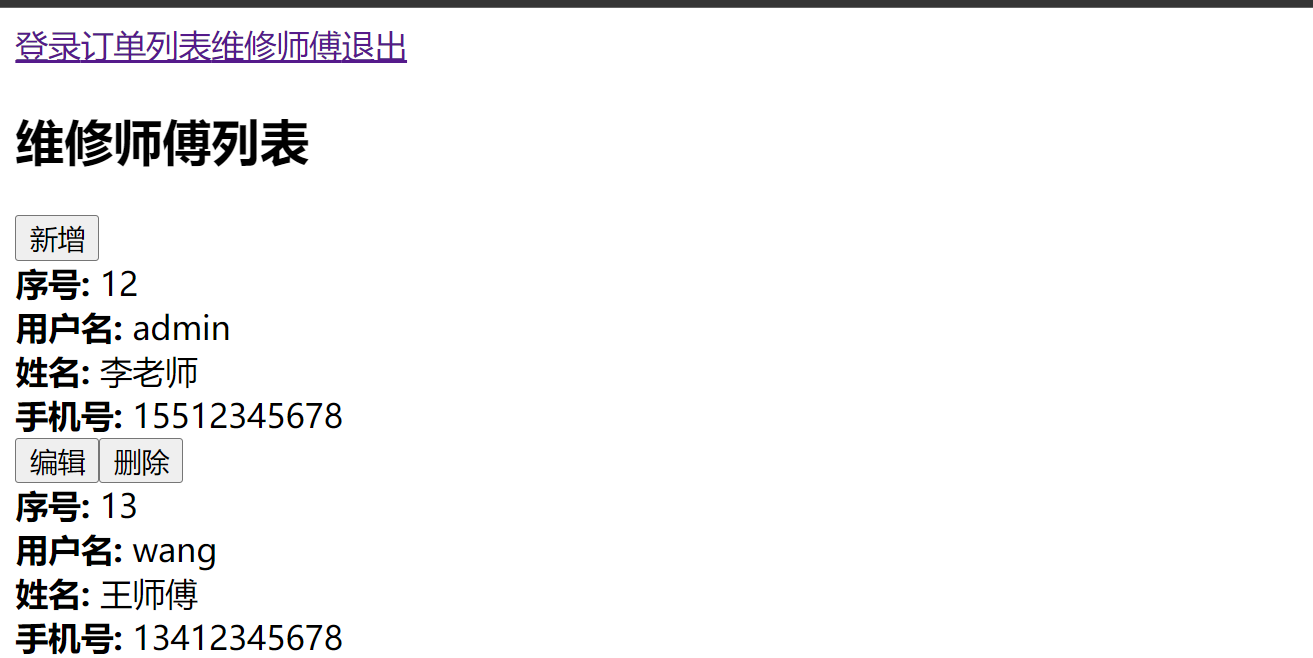
在App.vue文件夹下添加以下代码



回到浏览器，可以看到导航栏添加成功



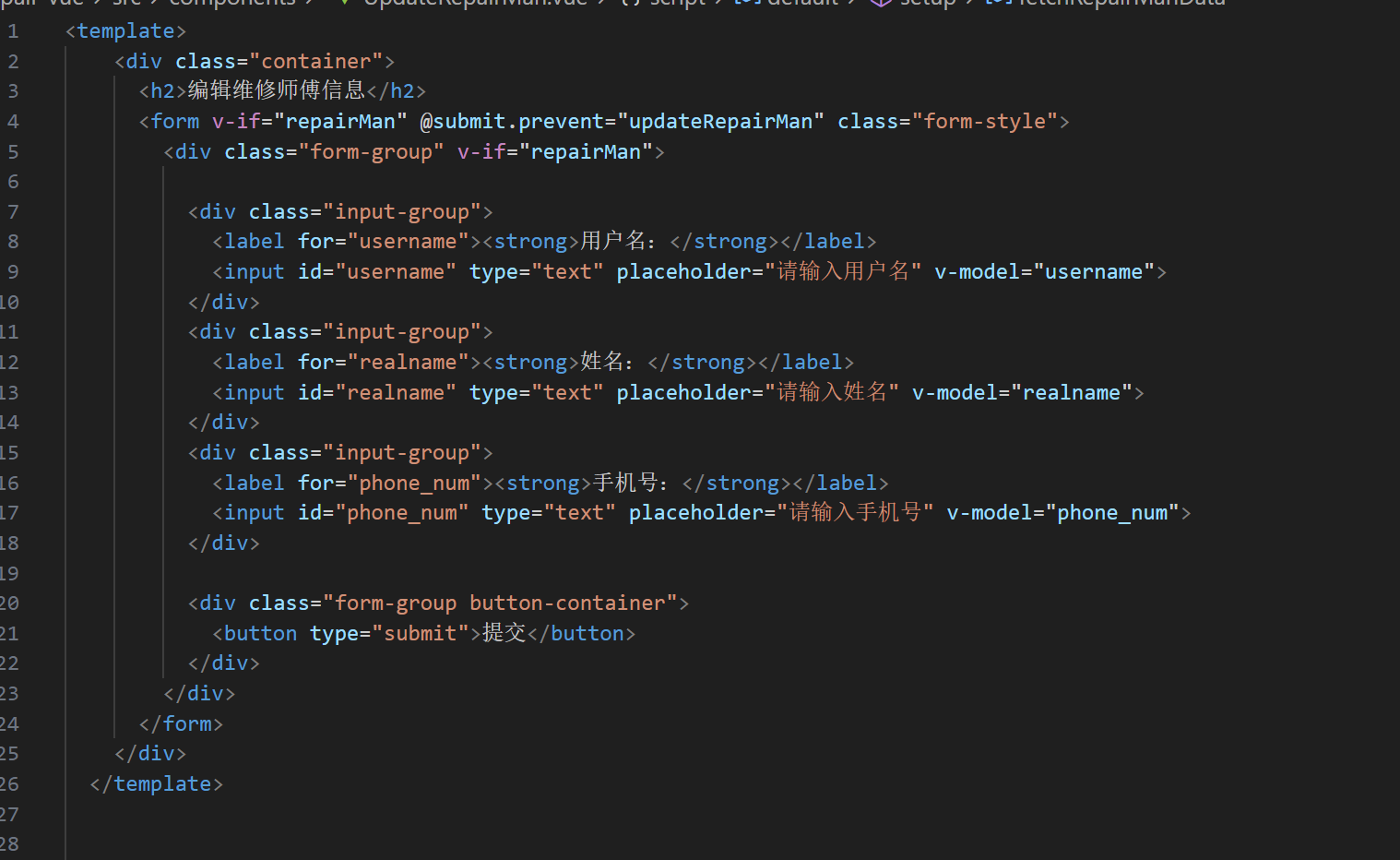
维修师傅列表页面也可正常访问



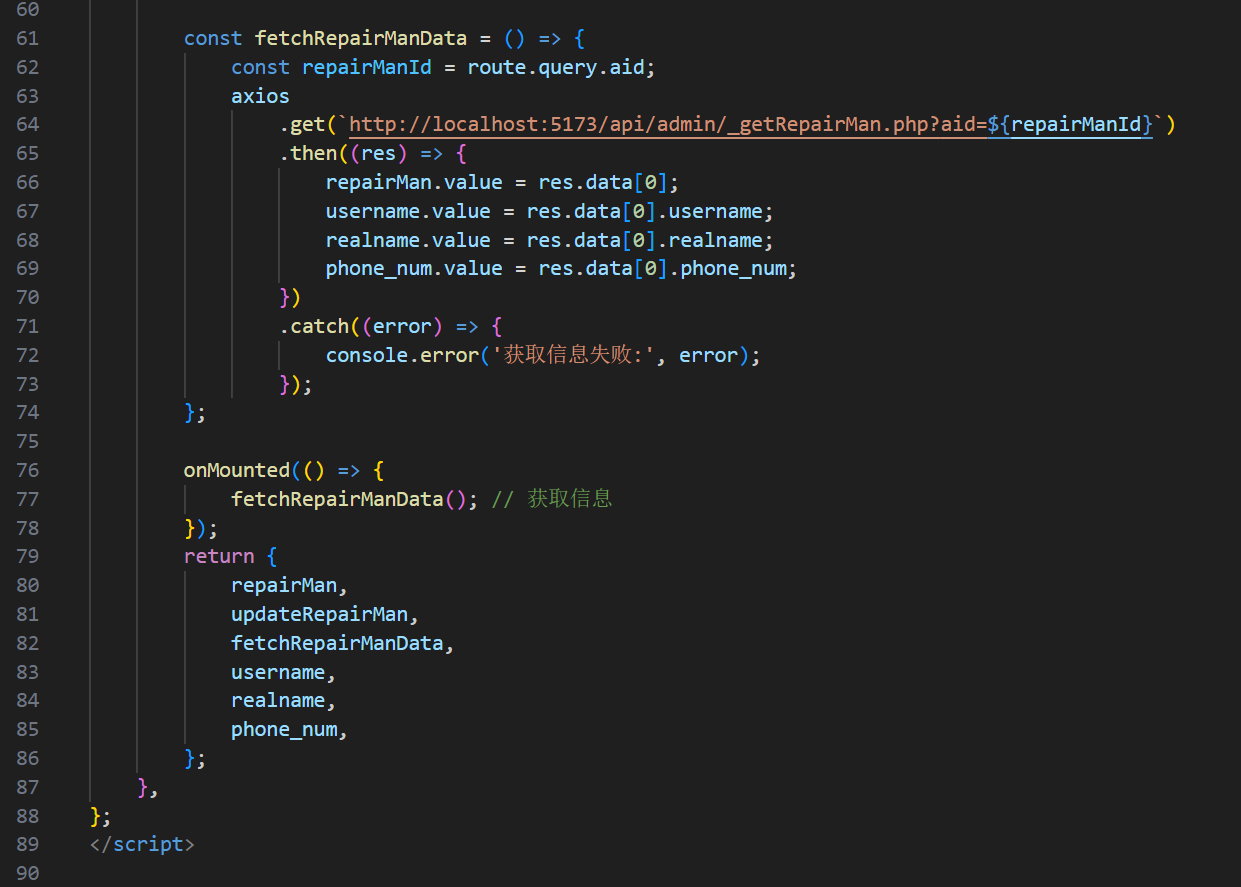
# 8.修改、删除维修师傅

现在我们来完成维修师傅的修改

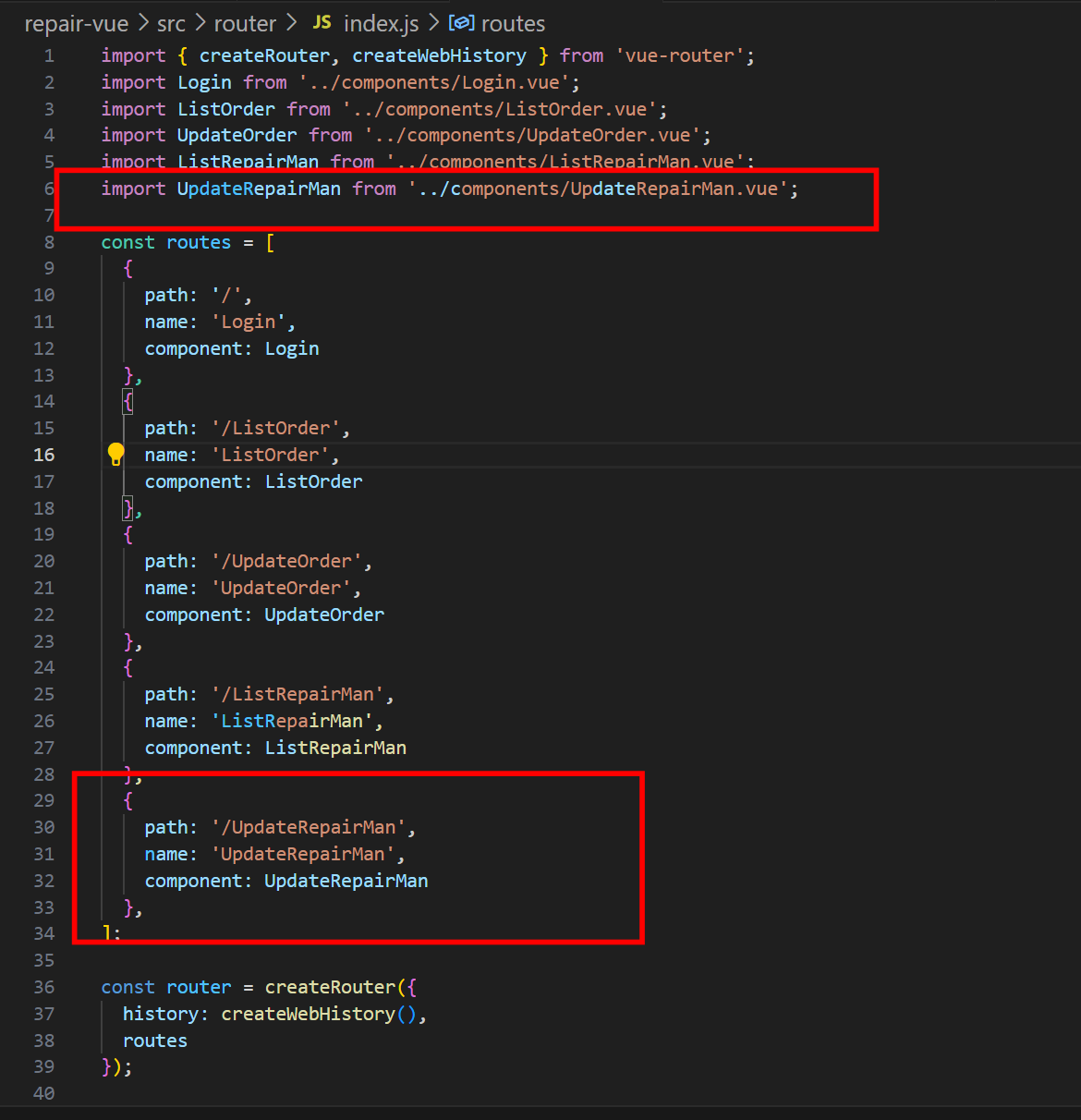
在components文件夹下新增UpdateRepairMan.vue 添加以下代码







添加相关路由

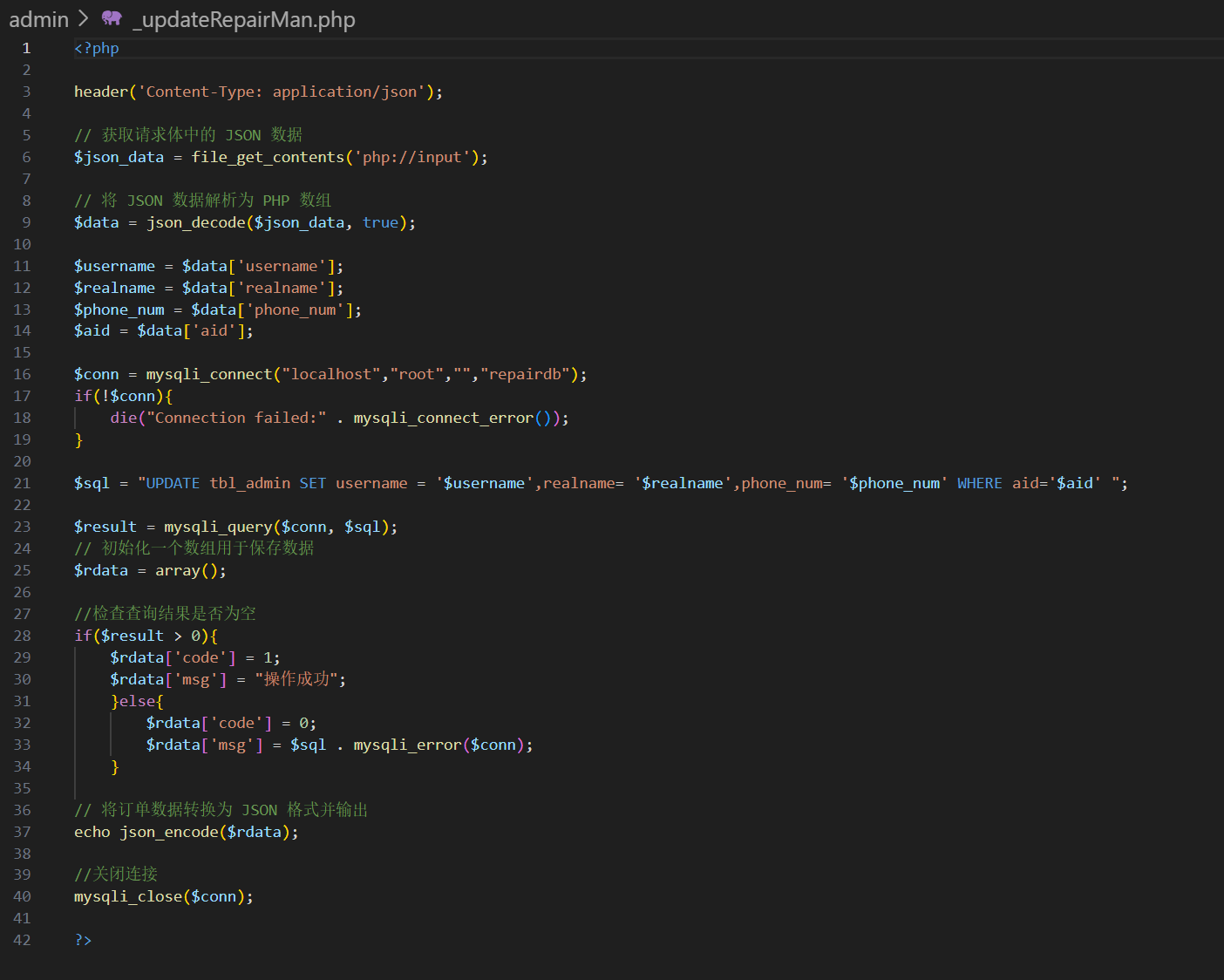


编写后端接口

根据aid获取维修师傅信息 新建\_getRepairMan.php文件



更新维修师傅接口文件，新建\_updateRepairMan.php文件

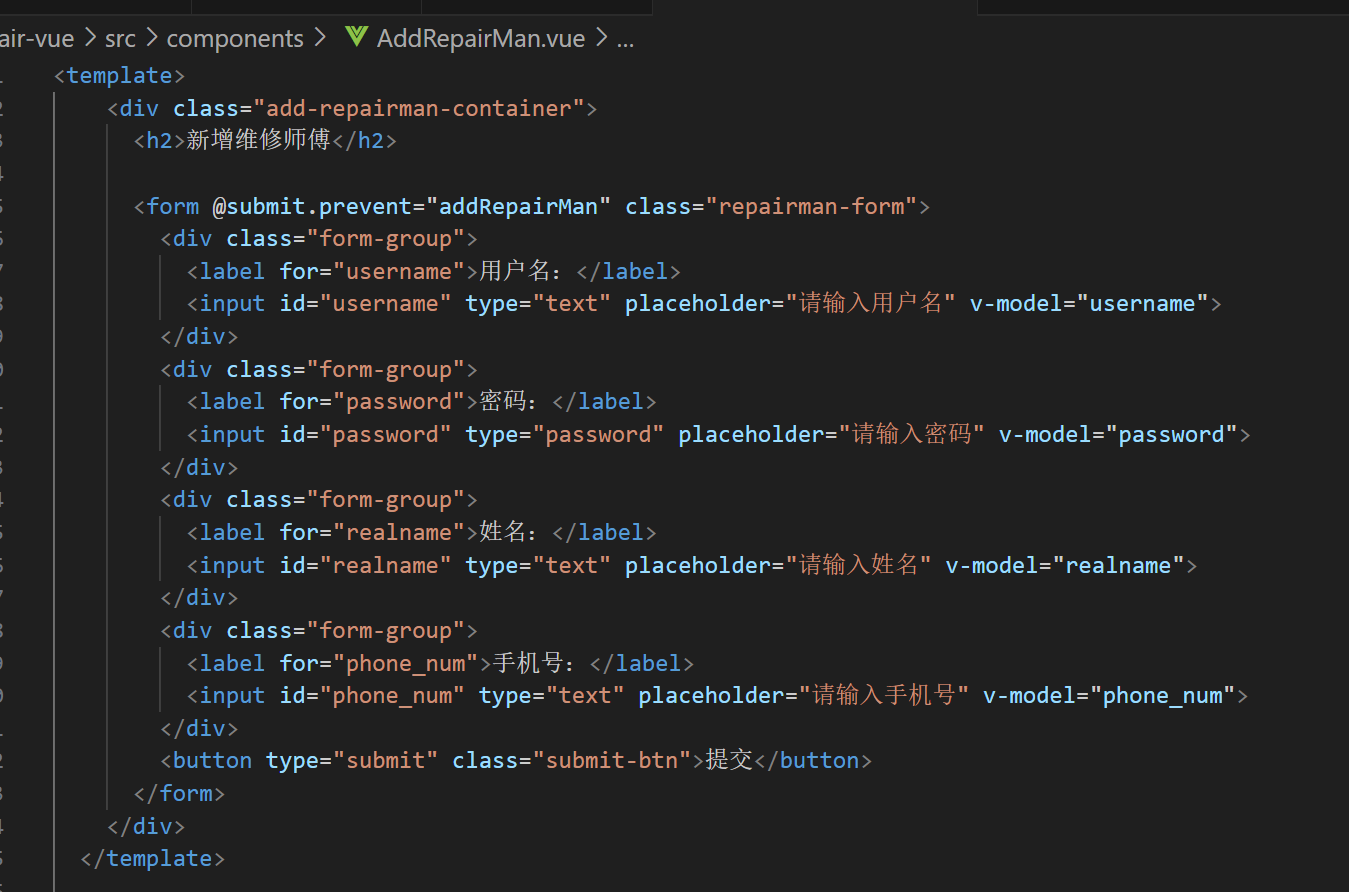


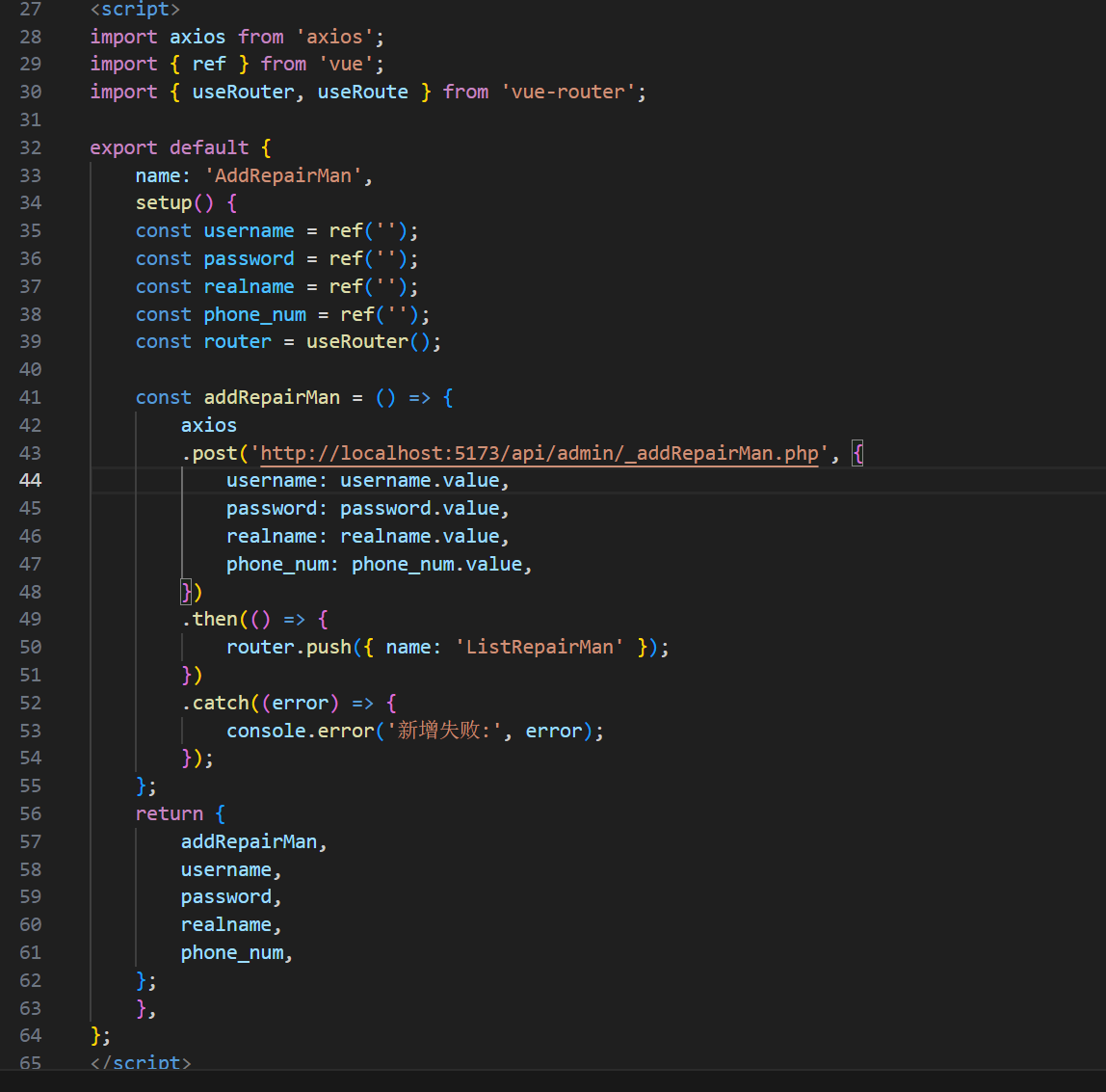
至此，维修师傅信息修改和删除完成

# 9.新增维修师傅

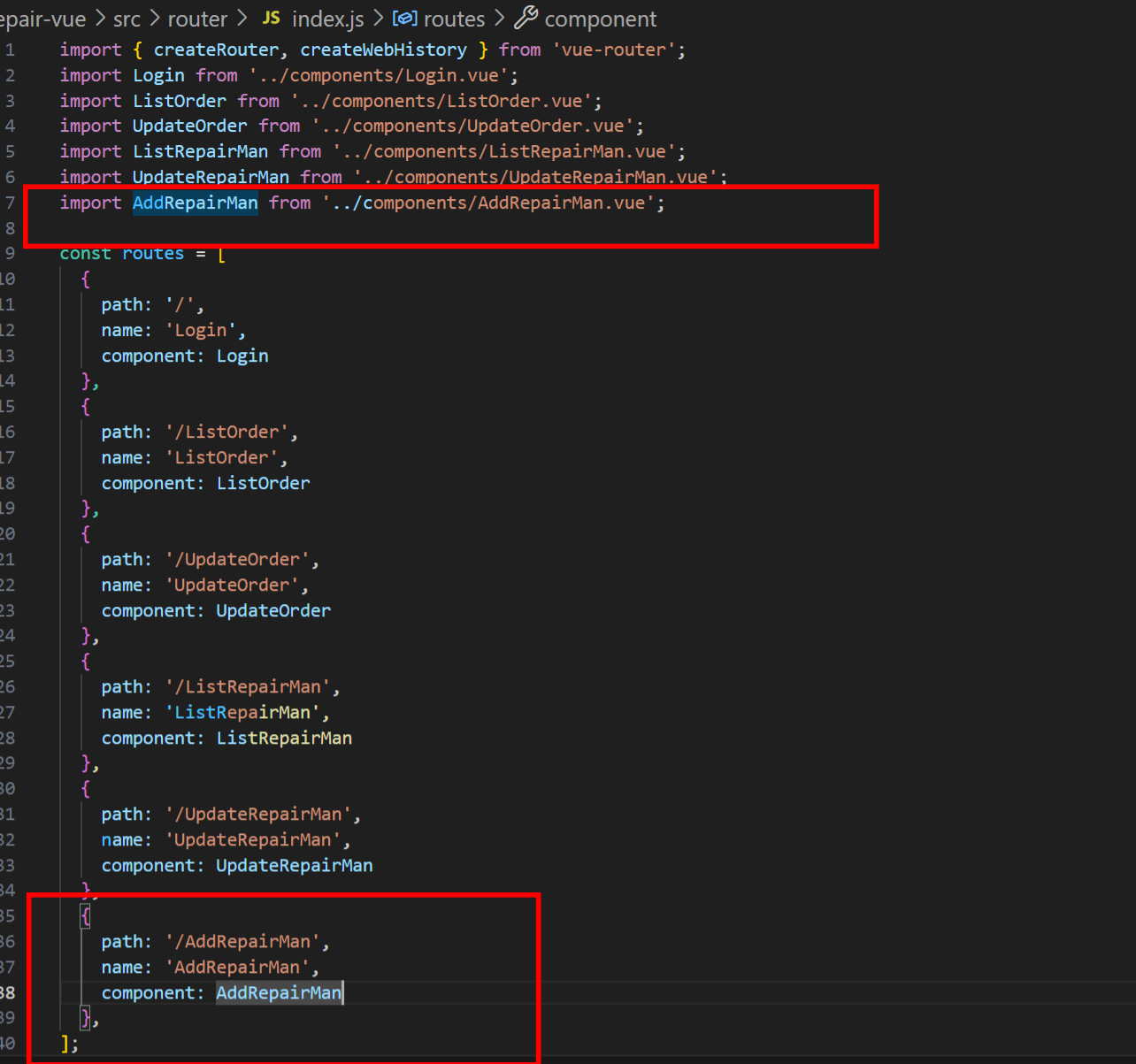
现在我们来完成维修师傅的新增

在components文件夹下新增AddRepairMan.vue 添加以下代码

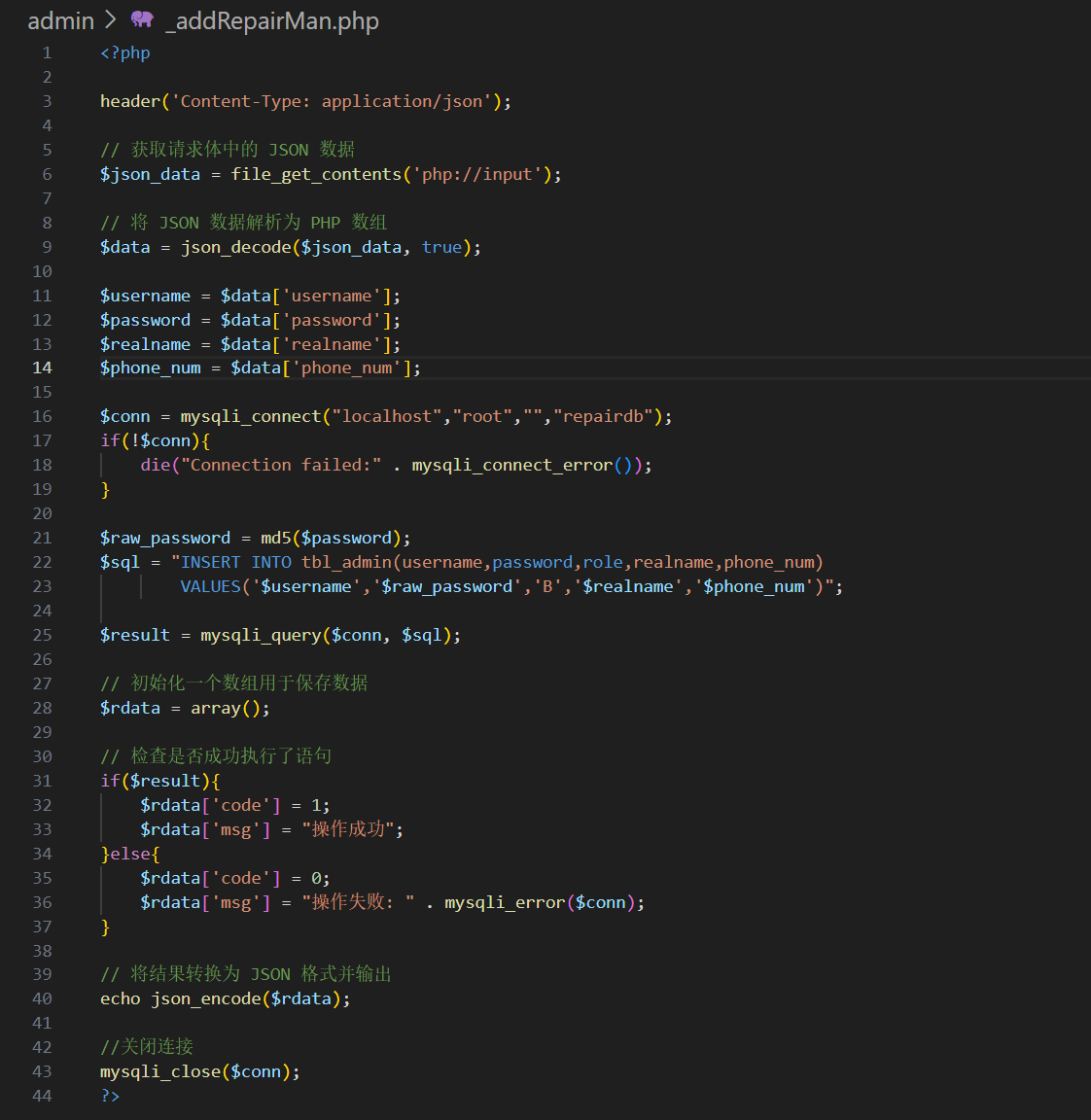




添加路由配置



编写新增维修师傅后端，新建\_addRepairMan.php文件



进行维修师傅新增



新增成功

