FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate

THỰC HỌC – THỰC NGHIỆP

FRONT-END FRAMEWORK

VUE ROUTER - AUTHENTICATE

http://www.poly.edu.vn

Mục tiêu

- Nắm vững cách thiết lập và cấu hình Vue Router trong ứng dụng Vue.js
- Hiểu về cách sử dụng route cơ bản, nested routes, và dynamic routing.
- Nắm được cách bảo Vệ Route Bằng Navigation Guards
- Nắm được cách xử lý lỗi xác thực

NỘI DUNG

Phần 1

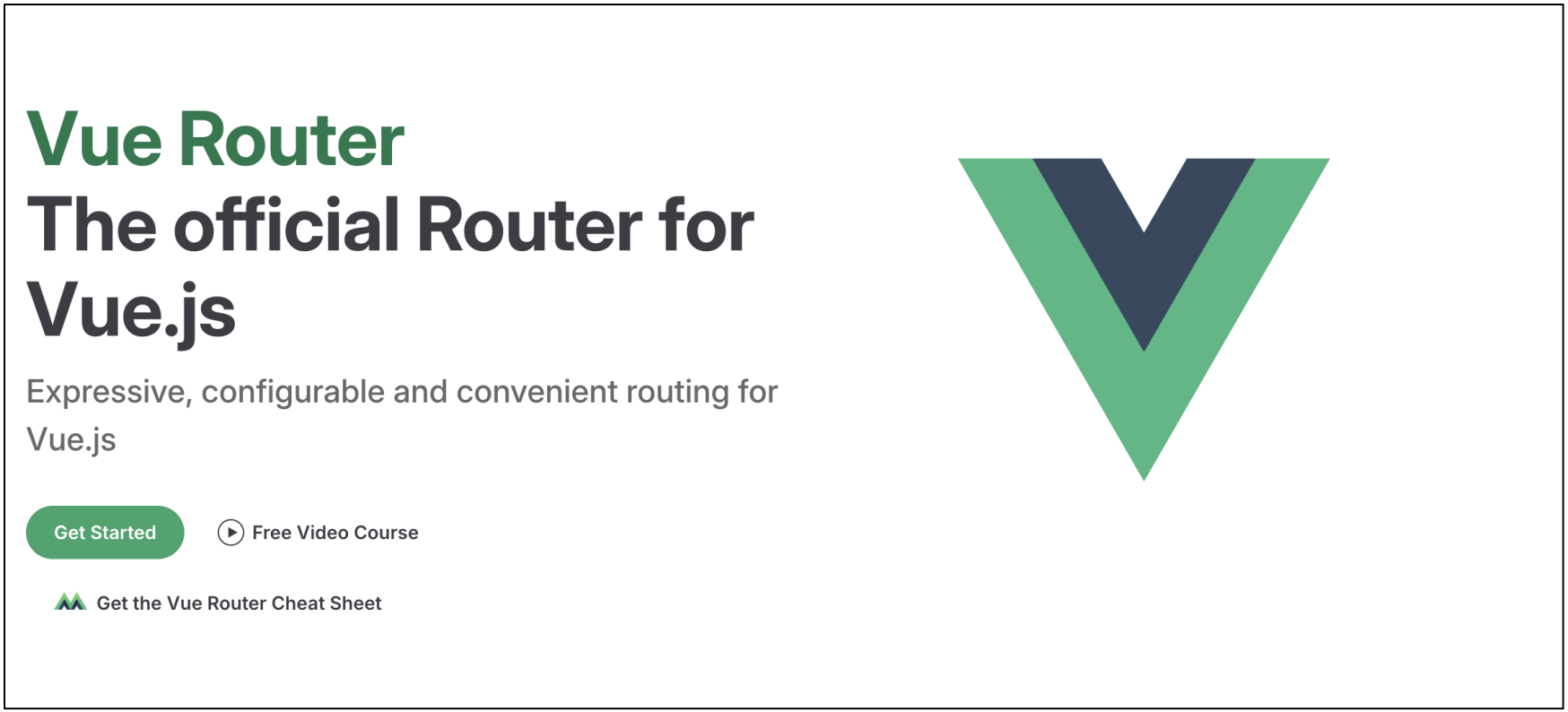
- Tổng quan về Vue Router
- Cài đặt và sử dụng Vue Router
- Tạo Instance Router
- Dynamic Route Matching với Params
- Name Routes
- Nested Routes

Phần 2

- Tổng quan về Authenticate
- Authenticate với Session Based
- Authenticate với Token Based
- Sử dụng Authenticate với Vue

PHẦN 1
TỔNG QUAN VỀ VUE ROUTER

TỔNG QUAN VUE ROUTER



Vue Router là router chính thức cho Vue.js. Nó tích hợp sâu với lõi của Vue.js, giúp việc xây dựng các ứng dụng một trang (Single Page Applications - SPAs) với Vue.js trở nên dễ dàng hơn bao giờ hết.

TỔNG QUAN VUE ROUTER

- Một số tính năng nổi bật bao gồm:
  - **Mapping nested routes** (Định tuyến lồng ghép)
  - **Dynamic Routing** (Định tuyến động)
  - **Modular, component-based router configuration** (Cấu hình router dựa trên component theo mô-đun)
  - **Route params**, **query, wildcards** (Tham số route, truy vấn, ký tự đại diện)
  - **View transition effects powered by Vue.js' transition system** (Hiệu ứng
  - ...

CÀI ĐẶT

Sử dụng npm

```
npm install vue-router@4
```

```
"dependencies": {
  "axios": "^1.7.7",
  "bootstrap": "^5.3.3",
  "vue": "^3.4.31",
  "vue-router": "^4.4.5"
},
```

package.json

Sử dụng link CDN:

```
https://unpkg.com/vue-router@4.0.15/dist/vue-router.global.js
```

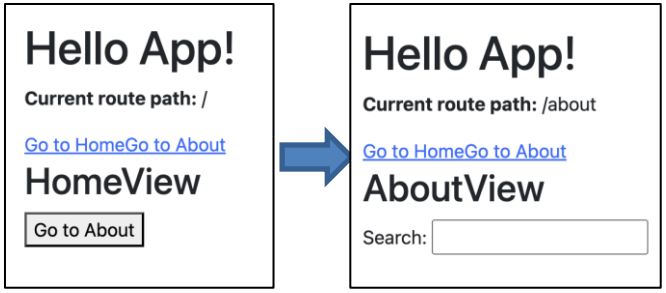
Muốn sử dụng Vue Router bạn có thể thông qua một đối tượng toàn cục VueRouter, ví dụ: VueRouter.createRouter(...).

SỬ DỤNG VUE ROUTER

Để hiểu Vue Router, chúng ta hãy xem xét ví dụ sau:

app.vue

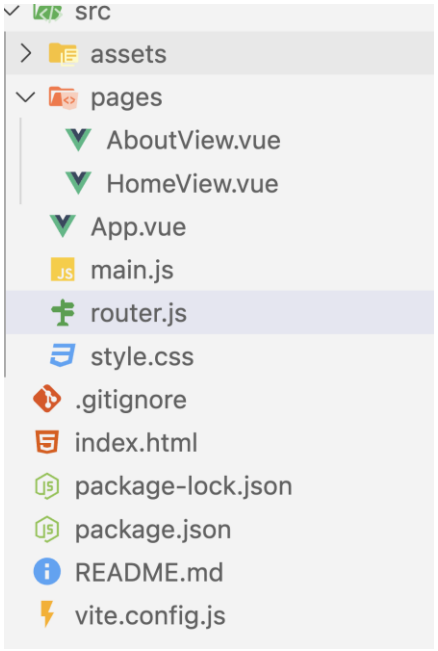
```
<template>
        <h1>Hello App!</h1>
        <p><strong>Current route path:</strong> {{$route.fullPath}}</p>
        <nav>
        <RouterLink to="/">Go to Home</RouterLink>
        <RouterLink to="/about">Go to About</RouterLink>
        </nav>
        <main>
                <RouterView />
        </main>
</template>
```



- **RouterLink** : thay thế cho thẻ <a>, cho phép thay đổi URL mà không cần tải lại trang.
- **RouterView:** dùng để xác định vị trí mà component tương ứng với route hiện tại sẽ được hiển thị trong ứng dụng Vue

TẠO INSTANCE ROUTER

Để tạo instance router, ta sử dụng hàm createRouter():



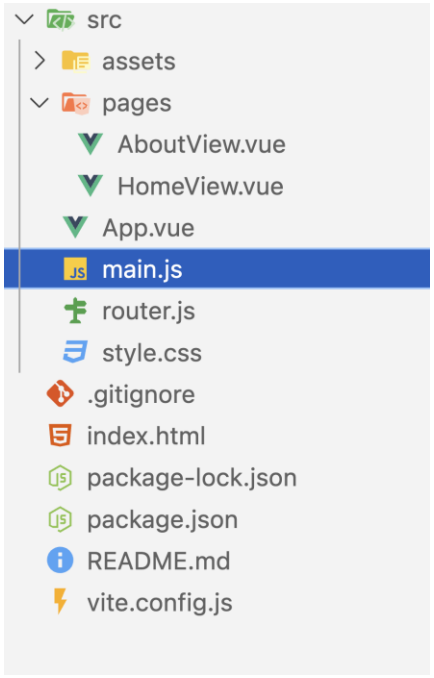
router.js

```
import { createMemoryHistory, createRouter } from"vue-router";
import HomeView from"./pages/HomeView.vue";
import AboutView from"./pages/AboutView.vue";
const routes = [
 { path:"/", component:HomeView },
 { path:"/about", component:AboutView },
];
const router = createRouter({
            history:createMemoryHistory(),
            routes,
});
export default router;
```

- Tùy chọn **routes** định nghĩa các route, ánh xạ các đường dẫn URL với các component tương ứng.
- Tùy chọn **history** xác định cách thức các route được ánh xạ lên URL.

ĐĂNG KÝ PLUGIN ROUTER

Sau khi tạo **instance router**, chúng ta cần đăng ký nó dưới dạng một plugin:



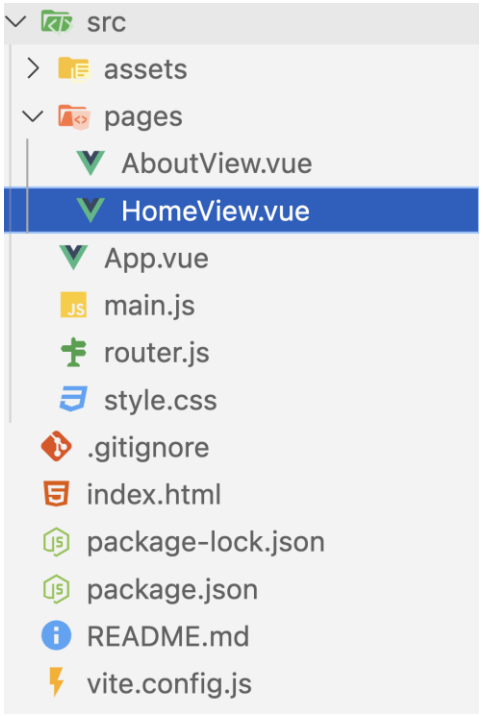
router.js

```
import { createApp } from"vue";
import router from"./router";
import App from"./App.vue";
createApp(App).use(router).mount("#app");
```

Hàm **use()** phải được gọi trước khi gọi **mount()** để đảm bảo router được khởi tạo đúng cách.

TRUY CẬP ROUTER

- Vue Router cung cấp những cách thức để truy cập và quản lý router và route trong ứng dụng của bạn. Dưới đây là hướng dẫn về cách sử dụng hàm chính: **useRouter**() .
- Template này cho phép người dùng điều hướng đến trang "About" khi nhấn vào nút.



HomeView.vue

```
<template>
<h2>HomeView</h2>
<button @click="goToAbout">Go to About</button>
</template>
<script setup>
import { useRouter } from"vue-router";
// Khởi tạo router
const router = useRouter();
// Hàm chuyển hướng đến trang About
const goToAbout = () => {
        router.push("/about");
};
</script>
```

DYNAMIC ROUTE MATCHING VỚI PARAMS

Cách định nghĩa một Dynamic Route

- Dynamic Route Matching cho phép bạn định nghĩa các route động trong Vue Router bằng cách sử dụng các tham số (params) để xác định URL động.
- Ví dụ cơ bản: Giả sử bạn muốn tạo một trang chi tiết sản phẩm, trong đó mỗi sản phẩm có một ID khác nhau.

```
const routes = [{ path:"/products/:id", component:ProductDetail }];
```

*Ở đây, **:id** là một route param động. Vue Router sẽ tự động khớp URL có dạng /products/:id với bất kỳ giá trị nào được truyền vào phần :id của URL.*

DYNAMIC ROUTE MATCHING VỚI PARAMS

Truy cập vào Params

- Trong component, bạn có thể truy cập các tham số cụ thể trên URL dựa vào hàm **useRoute()**.
- Ví dụ:

ProductDetail.vue

```
<template>
        <div>
        <h1>Product ID: {{productId}}</h1>
        </div>
</template>
<script setup>
import { useRoute } from 'vue-router';
// Truy cập vào route params thông qua useRoute
const route = useRoute();
const productId = route.params.id;
</script>
```

Kết hợp Route Params trong Links

- Khi tạo liên kết với RouterLink, bạn có thể dễ dàng thêm tham số vào URL bằng cách sử dụng cú pháp **:to="{ path: '/products/' + id }".**
- Ví dụ:

```
<RouterLink :to="{ path:'/products/' + product.id }">View Product</RouterLink>
```

Named Routes

- **Named Routes** cho phép bạn gán tên cho mỗi route khi tạo ra, giúp việc điều hướng dễ dàng và linh hoạt hơn mà không cần sử dụng đường dẫn tĩnh.
- **Cách tạo Named Route:**

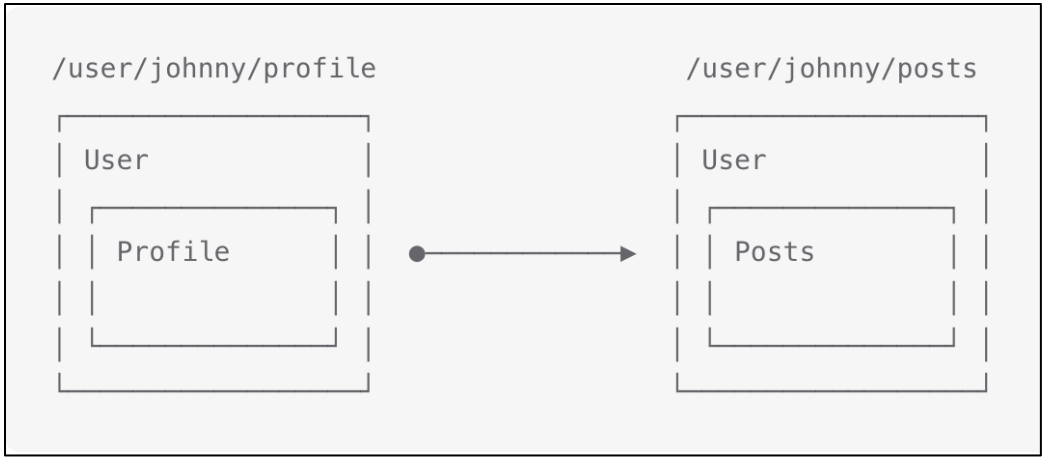
```
const routes = [
{
        path:"/user/:username",
        name:"profile",
        component:User,
},
];
```

- **Sử dụng Named Route với RouterLink:** Khi đã định nghĩa tên cho route, bạn có thể sử dụng nó thay vì đường dẫn khi truyền vào thuộc tính to của <router-link>:

```
<router-link :to="{ name:'profile', params: { username:'erina' } }">
 User profile
</router-link>
```

NESTED ROUTES

- **Nested Routes** cho phép bạn xây dựng cấu trúc URL tương ứng với cấu trúc thành phần lồng nhau trong giao diện người dùng.
- Giả sử bạn có một ứng dụng mà người dùng có thể truy cập trang cá nhân của họ với các phần con như "**Profile**" và "**Posts**". URL có thể là **/user/john/profile** hoặc **/user/john/posts.**



NESTED ROUTES

router.js

```
import User from "./App.vue";
import UserProfile from "./UserProfile.vue";
import UserPosts from "./UserPosts.vue";
import { createRouter, createWebHistory } from "vue-router";
constroutes = [
{
        path:"/user/:id",
        component:User, // Thành phần cha
        children: [
        {
                path:"profile",
                component:UserProfile, // Thành phần con
        },
        {
                path:"posts",
                component:UserPosts, // Thành phần con
        },
        ],
},
];
const router = createRouter({
        history:createWebHistory(),
        routes,
});
export default router;
```

UserProfile.vue

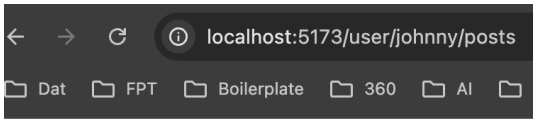
```
<template>
        <div>
        <h3>User Profile</h3>
        </div>
</template>
```

UserPosts.vue

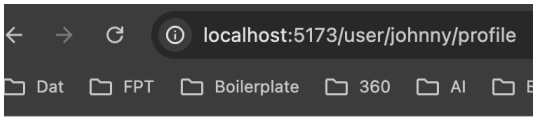
```
<template>
        <div>
        <h3>User Profile</h3>
        </div>
</template>
```

App.vue

```
<template>
<div class="user">
<h2>User {{$route.params.id}}</h2>
<!-- Đây là nơi route con sẽ được render -->
<router-view/>
</div>
</template>
```



User Posts

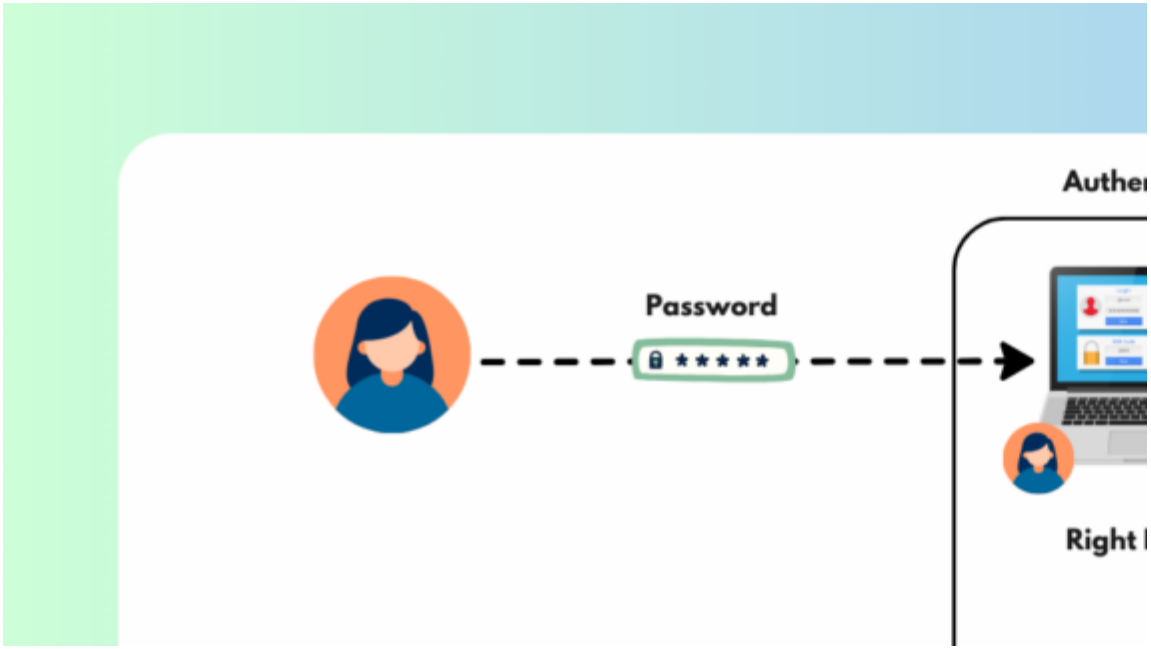


User Profile

Thực hiện lại ví dụ ở các bài vừa học

PHẦN 2: AUTHENTICATE

AUTHENTICATE (XÁC THỰC) LÀ GÌ?

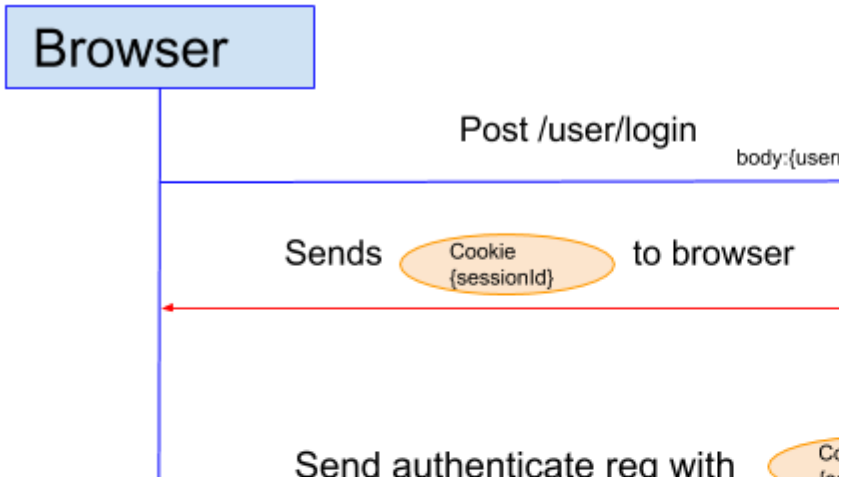


Xác thực (Authentication) là quá trình xác minh danh tính của một người dùng hoặc một hệ thống. Đây là bước quan trọng trong việc bảo vệ dữ liệu và thông tin nhạy cảm, đảm bảo rằng chỉ những người dùng được phép mới có thể truy cập vào tài nguyên hoặc chức năng của một ứng dụng.

AUTHENTICATE VỚI SESSION BASED

Cơ chế authencation với **session based** hoặc **cookie based**

1. **Đăng Nhập**: Người dùng gửi thông tin đăng nhập (username và password) đến server.
2. **Tạo Session**: Server xác thực và tạo session mới, lưu trữ thông tin (như ID người dùng).
3. **Gửi Cookie**: Server gửi cookie chứa Session ID cho client.
4. **Gửi Yêu Cầu**: Client gửi yêu cầu đến server, cookie tự động được kèm theo.
5. **Kiểm Tra Session**: Server kiểm tra Session ID trong cookie để xác định quyền truy cập.
6. **Đăng Xuất**: Khi người dùng đăng xuất, server xóa session và cookie.

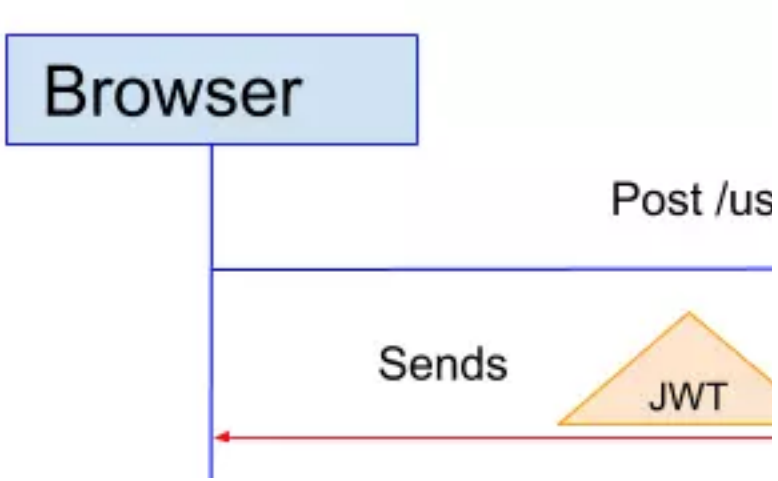


AUTHENTICATE VỚI TOKEN BASED

Authencation trong Vue hoạt động như thế nào

Quy trình hoạt động:

1. **Đăng Nhập**: Gửi thông tin đến server.
2. **Tạo Token**: Server tạo và gửi token.
3. **Lưu Token**: Client lưu token.
4. **Gửi Yêu Cầu**: Client gửi yêu cầu kèm toke
5. **Kiểm Tra Token**: Server kiểm tra token.
6. **Đăng Xuất**: Client xóa token.



AUTHENTICATE TRONG VUE

Bước 1: Cấu hình Router

Tạo file router.js để định nghĩa các route trong ứng dụng.

router.js

```
import { createRouter, createWebHistory } from"vue-router";
importHomefrom"./components/Home.vue";
importLoginfrom"./components/Login.vue";
importDashboardfrom"./components/Dashboard.vue";
constroutes = [
 { path:"/", component:Home },
 { path:"/login", component:Login },
 {
        path:"/dashboard", component:Dashboard, meta: { requiresAuth:true } },
];
constrouter = createRouter({
        history:createWebHistory(),
        routes,
});
exportdefaultrouter;
```

AUTHENTICATE TRONG VUE

Bước 2: Tạo hàm để kiểm tra xem người dùng đã đăng nhập hay chưa.

```
exportfunctionisAuthenticated() {
return !!localStorage.getItem("token");
// Kiểm tra xem token có trong localStorage không
}
```

Bước 3: Middleware Kiểm Tra Xác Thực

Thêm middleware để kiểm tra xem người dùng đã xác thực hay chưa trước khi truy cập các route yêu cầu.

middleware.js

```
import { isAuthenticated } from"./auth";
// Hàm kiểm tra xác thực
router.beforeEach((to, from, next) => {
if (to.meta.requiresAuth && !isAuthenticated()) {
next("/login");
 } else {
next();
 }
});
```

Code đầy đủ cho file router.js

```
import { createRouter, createWebHistory } from"vue-router";
import Home from"./components/Home.vue";
import Login from"./components/Login.vue";
import Dashboard from"./components/Dashboard.vue";
import { isAuthenticated } from"./auth";
const routes = [
 { path:"/", component:Home },
 { path:"/login", component:Login },
 { path:"/dashboard", component:Dashboard, meta: { requiresAuth:true } },
];
const router = createRouter({
        history:createWebHistory(),
        routes,
});
router.beforeEach((to, from, next) => {
if (to.meta.requiresAuth && !isAuthenticated()) {
        next("/login");
         } else {
                next();
         }
});
exportdefaultrouter;
```

router.js

AUTHENTICATE TRONG VUE

Bước 5: Tạo Component Đăng Nhập đặt tên là **Login.Vue**

```
<template>
<form @submit.prevent="login">
        <input
                v-model="username"
                placeholder="Username"/>
        <input
                v-model="password"
                type="password"
                placeholder="Password"/>
        <button type="submit">Login</button>
</form>
</template>
```

```
<script setup>
import { ref } from "vue";
import { useRouter } from "vue-router";
import axios from "axios";
const username = ref("");
const password = ref("");
const router = useRouter();
constlogin = async () => {
if (!username.value || !password.value) {
        alert("Vui lòng điền đầy đủ thông tin.");
        return;
}
try {
const response = await axios.post("/api/login", {
        username:username.value,
        password:password.value,
});
localStorage.setItem("token", response.data.token);
        // Lưu token vào localStorage
        router.push("/dashboard");
        // Chuyển đến dashboard sau khi đăng nhập thành công
} catch (error) {
        console.error("Login failed:", error);
        alert("Đăng nhập không thành công. Vui lòng kiểm tra lại thông tin.");
}
};
</script>
```

Bước 6: Đăng xuất

Tạo hàm để xóa token và chuyển hướng về trang đăng nhập.

```
exportfunctionisAuthenticated() {
        return !!localStorage.getItem("token");
        // Kiểm tra xem token có trong localStorage không
}
exportfunctionlogout() {
        localStorage.removeItem("token");
        router.push("/login");
}
```

AUTHENTICATE TRONG VUE

- Xác thực trong Vue 3 thường liên quan đến việc sử dụng Vue Router để kiểm soát quyền truy cập, kết hợp với một API backend để xác thực người dùng và quản lý phiên làm việc.
- Bằng cách sử dụng localStorage để lưu trữ token, ứng dụng có thể duy trì trạng thái xác thực qua các lần tải lại trang.

Tóm Tắt lại các bước:

1. Thêm Vue Router: Quản lý các route trong ứng dụng.
2. Middleware: Kiểm tra xác thực trước khi vào các route yêu cầu.
3. Component Đăng Nhập: Xử lý xác thực và lưu token.
4. Quản lý Phiên: Sử dụng localStorage để lưu trữ thông tin xác thực.

THỰC HIỆN ĐĂNG KÝ VÀ ĐĂNG NHẬP
ĐĂNG NHẬP VÀ ĐĂNG XUẤT TỰ ĐỘNG

SUMARRY

- ☑ Phần 1
  - ❖ Tổng quan về Vue Router
  - ❖ Cài đặt và sử dụng Vue Router
  - ❖ Tạo Instance Router
  - ❖ Dynamic Route Matching với Params
  - ❖ Name Routes
  - ❖ Nested Routes
- ☑ Phần 2
  - ❖ Tổng quan về Authenticate
  - ❖ Authenticate với Session Based
  - ❖ Authenticate với Token Based
  - ❖ Sử dụng Authenticate với Vue


thank
you!