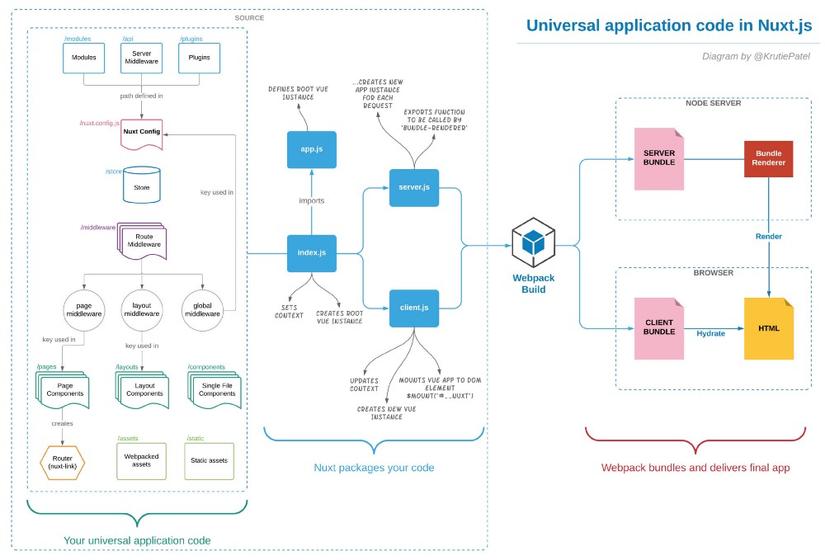
# Các thư mục và cấu trúc

## 1. NuxtJs là gì?

Nuxt là một framwork được phát triển dựa trên Vue.js để tạo các ứng dụng web tốt hơn. Nó dựa trên các thư viện chính thức của Vue.js (vue, vue-router và vuex) và các công cụ phát triển mạnh mẽ (webpack, Babel và PostCSS).

Mục đích chính của nó là làm cho sự phát triển web trở nên mạnh mẽ và tạo ra một quy trình phát triển hợp lý hơn dành cho các Developers. Về bản chất, nó hỗ trợ các mục tiêu khác nhau (máy chủ, không máy chủ hoặc tĩnh) và kết xuất phía máy chủ có thể chuyển đổi được, đặc biệt có thể mở rộng với hệ sinh thái module mạnh mẽ.



## Cấu trúc thư mục trong NuxtJs

Cấu trúc ứng dụng Nuxt.js mặc định nhằm cung cấp một điểm khởi đầu tuyệt vời cho cả các ứng dụng nhỏ và lớn. Tất nhiên, bạn có thể tự do sắp xếp ứng dụng của mình theo cách bạn muốn.

### Assets

Thư mục assets, đúng như với tên của nó. Assets chứa các nội dung un-compiled của bạn, chẳng hạn như JavaScript, SCSS và images. Nuxt.js sử dụng webpack để xử lý và compile các nội dung này.

Nuxt sẽ config webpack để xử lý các file URL ASSET bằng file-loader and url-loader. Lợi ích của file-loader là đảm bảo băm các file tốt hơn sẽ có lợi cho tổng thể trong quá trình sản xuất, trong khi url-loader sẽ hỗ trợ giảm các yêu cầu của HTTP đối với các file URL ASSET không vượt quá một ngưỡng nhất định bằng cách đặt chúng dưới dạng URL base-64.

### Static

Đối với các nội dung sẽ không thay đổi, bạn có thể đặt chúng vào thư mục static và webpack sẽ bỏ qua thư mục static và sẽ không xử lý bất cứ thứ gì trong đó (ví dụ: file favicon).

Trong code của bạn, sau đó bạn có thể tham chiếu các file từ root (/):

Xem sự khác biệt về cú pháp truy cập giữa thư mục static và thư mục assets:

<!-- How to access static image from static directory -->

<img src="/image.jpg"/>

<!-- How to access webpacked image from assets directory -->

<img src="~/assets/image.jpg"/>

### Pages

Có thể nói đây là một trong những thư mục quan trọng nhất của Nuxt, vì nó sẽ tự động tạo các route bất kỳ theo từng file .vue trong đó. Thư mục pages chứa các Views và Routers cho ứng dụng của bạn. Nó sẽ mô tả tất cả những gì bạn cần để cấu hình dữ liệu và view cho một route cụ thể trong ứng dụng của bạn App Template, Layouts, Pages and HTML Head).

Tất cả các file .vue trong thư mục này đều cung cấp các tùy chọn như **asyncData, layout, middleware, auth, fetch, head, validate, scrollToTop** là tất cả những gì bạn cần để cấu hình dữ liệu và view cho bất kỳ route nào.

Đặc biệt, mặc định Nuxt CLI tạo ra file pages/index.vue và nó đóng vai trò là trang chủ của ứng dụng (trang index). Với sức mạnh này thì nó có thể tạo ra các URL thân thiện với SEO.

### Middleware

Middleware về cơ bản, nó chứa các hàm JavaScript tùy chỉnh chạy ngay trước khi một trang hoặc nhóm trang được hiển thị. Ví dụ: Hãy tưởng tượng bạn muốn kiểm tra xem người dùng có thông tin đăng nhập phù hợp để truy cập một trang hay không. Trong trường hợp này, bạn có thể có thể tạo một file có tên middleware/auth.js chứa đoạn code bên dưới.

export default function({ store }) {

// some code to check user authentication

}

Tuy nhiên, nếu chỉ tạo một chức năng tùy chỉnh trong thư mục middleware sẽ không có tác dụng gì. Bạn phải cho Nuxt.js biết nơi mà bạn muốn áp dụng nó, nghĩa là trên tất cả các trang hoặc một vài trang được chọn hoặc một trang duy nhất.

Từ đây, bạn có thể sử dụng chúng theo những cách sau:

* Global middleware - (Thông qua file cấu hình Nuxt và ảnh hưởng đến tất cả các route)

// nuxt.config.js

export default {

router: {

middleware: 'auth'

}

}

* Layout middleware (Thông qua layouts và ảnh hưởng đến nhóm các route tương ứng, tức là các trang sử dụng layouts đó chỉ được xem và truy cập bởi người dùng đã xác thực)

// layouts/default.vue

export default {

middleware: 'authenticated-basic-plan-user'

}

* Page middleware (Thông qua từng page và ảnh hưởng trên route gọi đến page đó)

// pages/index.vue

export default {

middleware: 'subscribed'

}

### Components

Các components là các bit code độc lập và có thể tái sử dụng. Để nhập một component được lưu dưới dạng BarChart (tức là components/BarChart.vue) và sau đó gọi nó bằng tên BarChart trong một page (tức là pages/posts.vue), bạn viết như này:

import BarChart from "~/components/BarChart";

export default {

components: {

BarChart

},

}

Lưu ý rằng các thành phần trong thư mục này không có quyền truy cập vào asyncData.

### Layouts

Layouts được sử dụng để thay đổi giao diện ứng dụng của bạn. Một ứng dụng có thể có nhiều layouts ví dụ: admin layout, guest layout và registered clients layout. Các bố cục này sẽ được sử dụng lại trong các trang khác nhau để xử lý giao diện của chúng (sidebar, menu, footer, v.v.). Trong khi cài đặt, Nuxt CLI cung cấp mặc định layouts/default.vue layout và được sử dụng trong tất cả các trang.

Mỗi dự án được tạo Nuxt có một file default.vue trong thư mục layouts, với cấu trúc mẫu tối thiểu sau:

<template>

<nuxt/>

</template>

Thành phần </nuxt> rất quan trọng vì nó hiển thị các thành phần trang, tức là các file .vue của bạn từ thư mục pages.

Tất nhiên, bạn cũng có thể tạo layout tùy chỉnh của riêng mình, bao gồm cả trang lỗi. Do đó, để sử dụng một instance layout có tên là admin (layouts/admin.vue) tại một page post (pages/posts.vue) , bạn sẽ viết:

<template>

<!-- Your template -->

</template>

<script>

export default {

layout: 'admin'

// page component definitions

}

</script>

### Plugins

Trong bất kỳ dự án Vue thông thường nào, bạn có thể đăng ký global các thư viện Vue trong file main.js. Tuy nhiên, file này không tồn tại trong ứng dụng Nuxt.js và do đó thư mục plugins sẽ làm điều này.

Ví dụ: Vue plugin vue-notifications, cho phép bạn hiển thị thông báo trong ứng dụng của mình. Sau khi cài đặt qua npm hoặc yarn, bạn tạo file plugins/vue-notification.js chứa đoạn code bên dưới:

import Vue from 'vue'

import VueNotifications from 'vue-notifications'

Vue.use(VueNotifications)

Nhìn cú pháp quen chưa kìa  Sau đó, bạn phải tìm cách thông báo cho Nuxt.js rằng bạn đã cài đặt xong rồi và muốn sử dụng plugin này, bằng cách chỉnh sửa file nuxt.config.js. Dấu ~ có chức năng giống như ký tự @, nghĩa là nó tham chiếu đến thư mục gốc:

export default {

plugins: ['~/plugins/vue-notifications']

}

Vậy là xong rồi đó, về cơ bản thì bạn muốn dùng plugin nào thì cũng sẽ làm như bên trên vậy đó.

### Store

Nuxt được đóng gói sẵn với Vuex, nhưng nó sẽ không được kích hoạt trừ khi bạn tạo một store Vuex trong thư mục store. Đây là thư mục rất đặc biệt cho bất kỳ data-driven project nào. Đây là nơi bạn có thể tạo một kho lưu trữ dữ liệu, cũng như xác định hành động nuxtServerInit. Đây cũng là vòng đời đầu tiên!

const createStore = () => {

return new Vuex.Store({

...

})

}

Khi người dùng truy cập ban đầu vào ứng dụng của bạn, điều này giúp fill/update dữ liệu vào store. Nó cho phép bạn sử dụng modules, state, getters, mutations and actions giống như tiêu chuẩn trên ứng dụng Vue.js và cũng duy trì trạng thái dữ liệu của bạn trong suốt ứng dụng.

### Nuxt Configuration File

File nuxt.config.js chứa cấu hình tùy chỉnh Nuxt của bạn và cho phép bạn config cấu hình ứng dụng của mình, những cấu hình này bao gồm head title và associated styles và scripts, middlewares, plugins, authentication, modules và thậm chí cả các API.

CHÚ Ý: Các example trên đều là của nuxt 2. Nuxt 3 cú pháp có khác. Em không tìm thấy tài liệu Tiếng Việt. Ae có thể tham khảo nuxt 3 ở đây

https://nuxt.com/docs/guide/directory-structure/nuxt

# Hướng dẫn coding

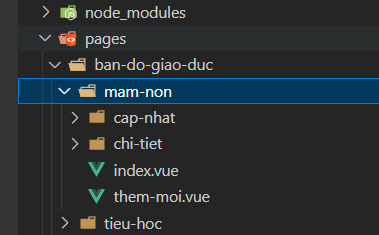
# Để tạo mới 1 module bất kỳ ta làm các bước sau:

Bước 1: Vào thư mục constants mở file menu.js. Thêm 1 “node” trong mảng.

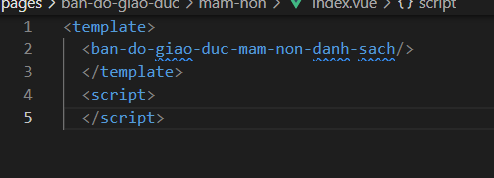
Các params: title: Tiêu đề hiển thị trên menu, url: link router, name: Tên menu (chưa dùng tới), icon: icon hiển thị. ( icon dùng ảnh svg).

Bước 2: Tạo file trong thư mục pages. Tương ứng với url mình đã tạo trong menu.js

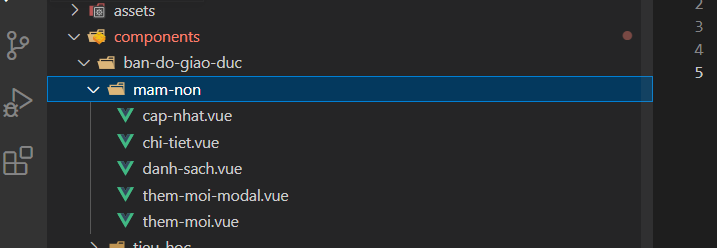
Ví dụ với url: ban-do-giao-duc/mam-non => Tạo thư mục: ban-do-giao-duc/mam-non/index.vue



Chú ý: Đây là trang điều hướng thôi. Không viết logic trong này

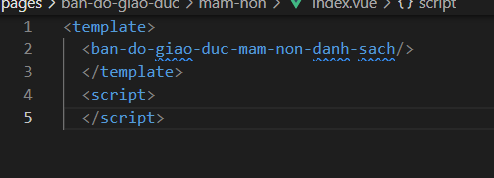


Bước 3: Tạo các file trong thư mục component để chứa các logic.

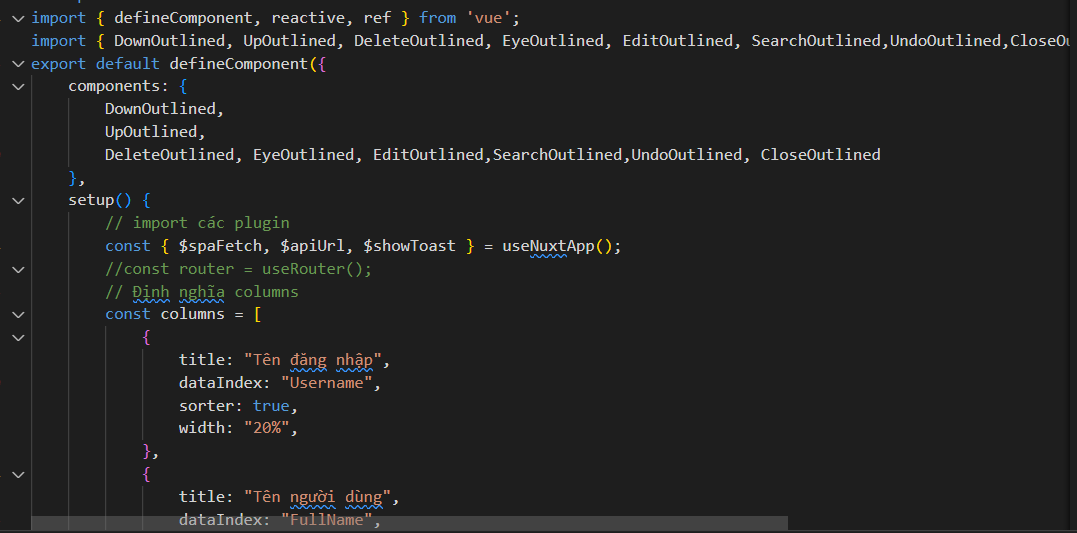


Mình dùng thằng nuxt nên component sẽ được tự động import vào nếu viết trong thư mục component.

Để gọi chỉ cần viết giống 1 module: tên đường dẫn + tên file.



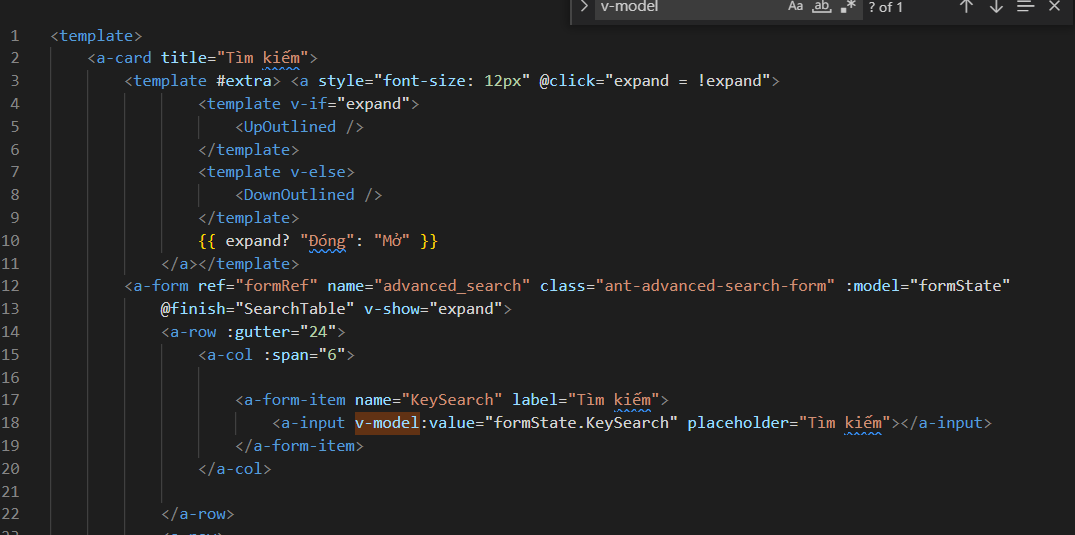
Coding bên trong: Mình dùng vue 3 nên sẽ viết kiểu omposition API



Tìm hiểu thêm tại đây

<https://viblo.asia/p/tim-hieu-vue-composition-api-phan-1-eW65GL7PKDO>

còn phần template vẫn thế. UI framework thì mình sử dụng các control của VueAntDesgin.



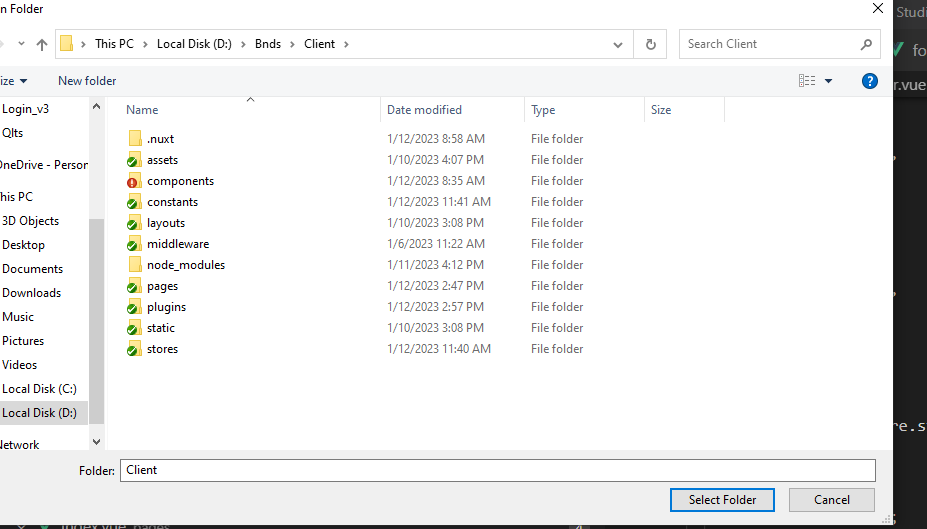
Tìm hiểu thêm tại đây:

<https://antdv.com/components/overview>

# Hướng dẫn build project Yêu cầu: Cài đặt nodejs, yarn (có thể không cần, như recommend vì ngon hơn npm ☺) ), cài thêm GIT bash chạy console cho ngon lành

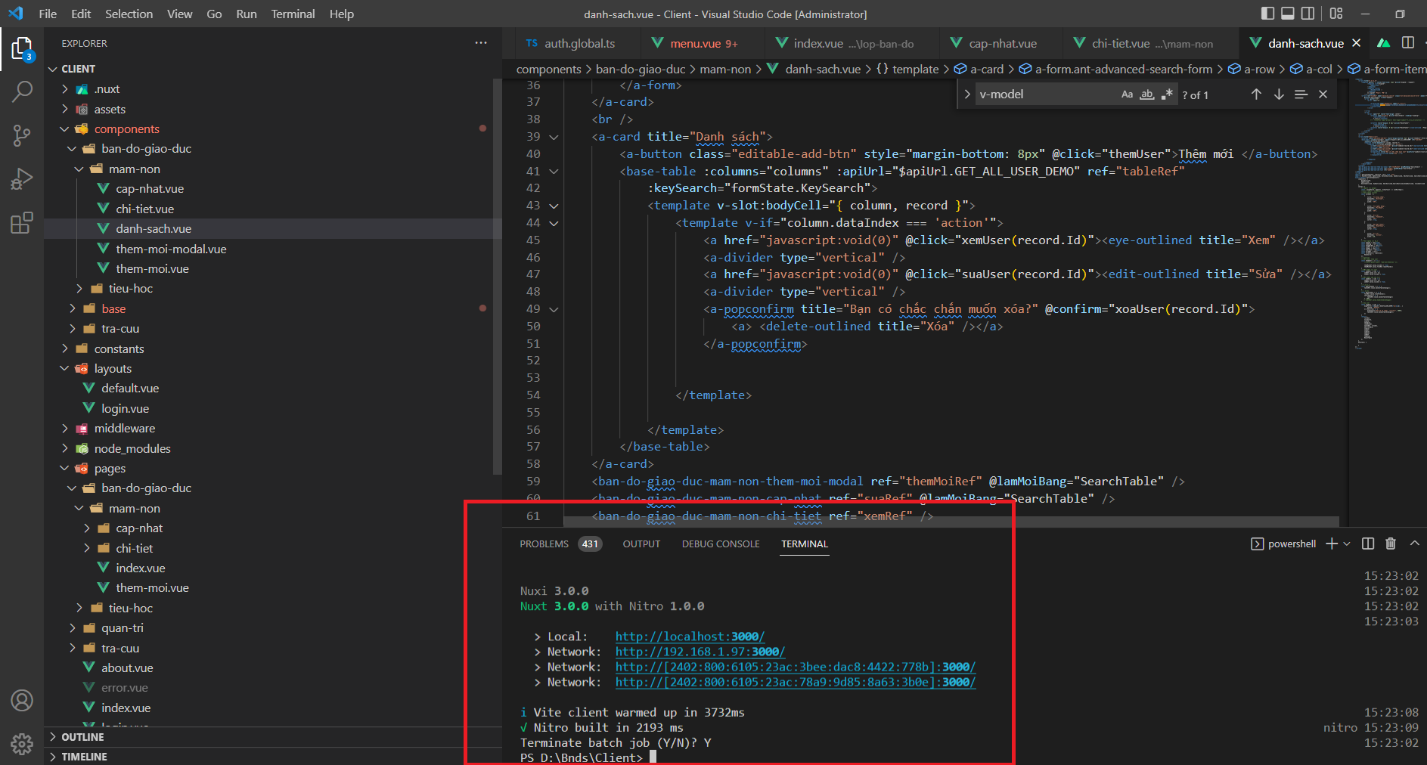
Sau khi get code thì bật visual studio hoặc visual code: (ở đây mình dùng visual code)

Mở thư mục code client:



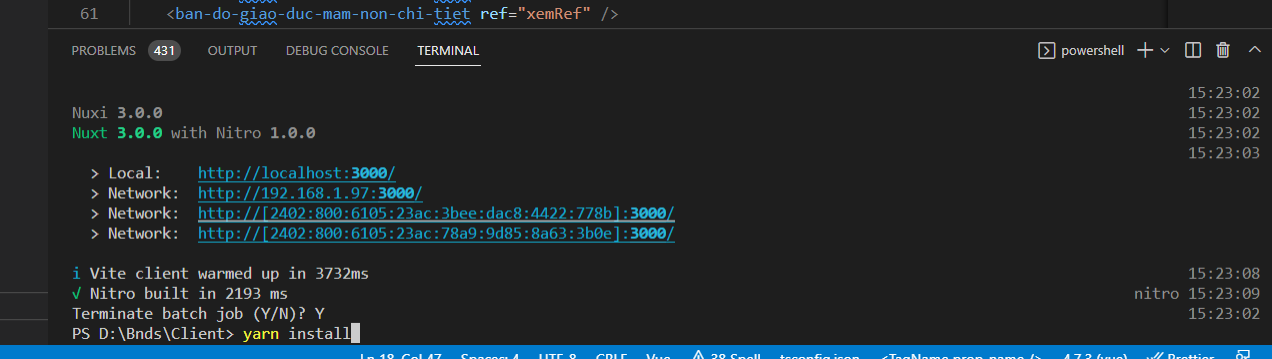
Nhấn select folder.

Sau đó bật cửa sổ TERMINAL( bấm Ctrl ` )



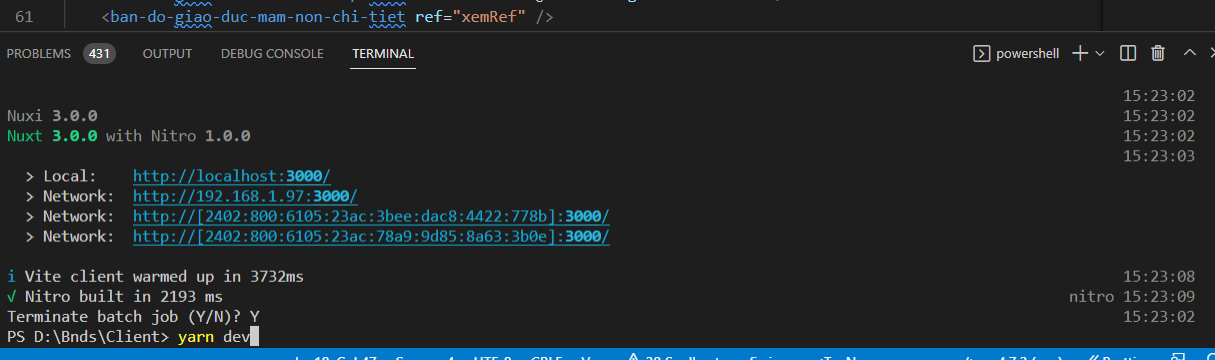
Chạy install package:

Gõ trên cửa sổ TERMINAL: yarn install (hoặc npm install)



Chạy project:

Gõ trên cửa sổ TERMINAL: yarn dev (hoặc npm run dev)



Sau đó console sẽ hiện các link như trên. Click vào link để chạy client

# Coding convention

1. **Chung**
   1. Sử dụng từ khóa TODO với các phần code cần xem lại, điều chỉnh thêm, sửa lại sau… (các việc sẽ làm)
   2. Nghĩ đến người khác khi code
2. **Quy chuẩn code phía server**
   1. Đặt tên
      1. Tất cả tên biến đều phải theo chuẩn camelCase
      2. Tất cả hằng số (constant) đều phải viết upper case
      3. Tất cả tên thư mục, tên file đặt snake case (trừ trường hợp đặc biệt)
   2. Comment
      1. Tất cả method phải có comment, sử dụng ///

Nội dung nêu rõ method dùng để làm gì

* + 1. Các thuộc tính của Entity nếu đặc biệt đều cần có comment, sử dụng ///
  1. Try catch và log lỗi
     1. Các method ở BLL bắt buộc phải try catch, log error
     2. Các method ở controller nếu có thể phát sinh lỗi mà các tầng dưới chưa bắt thì phải thêm try catch và log lỗi (nếu chỉ gọi BLL mà BLL đã bắt lỗi thì controller không cần nữa)

1. **Quy chuẩn code phía client**
   1. Đặt tên
      1. Tất cả tên biến đều phải theo chuẩn camelCase
      2. Tên các biến, url và thư mục các thứ nên để tiếng việt.
      3. Tất cả tên thư mục, tên file đều phải theo chuẩn lower case, ngăn cách bởi dấu – nếu dài hơn 1 từ (mam-non, tieu-hoc)