**Nhóm: 10**

**Thành viên tham gia**

**1 . Nguyễn Thanh Ngân -D17\_TH07-DH51703779**

**2. Đặng Như Vũ -D17\_TH07 –DH51702657**

**3. Trương Thế Kiệt D17-TH07--DH51703591**

**4. Nguyễn Văn Hiến D16-TH08--DH51603224**

**Bài Thu hoạch API**

**1. Các chuẩn hóa RESTful API. (Các method, cách đặt tên hàm, URL, version, ....)**

**Các method của RESTful API:**

HTTP đã định nghĩa một vài methods chỉ ra loại hành động sẽ được thực hiện trên các resource.

* GET (SELECT): Trả về một Resource hoặc một danh sách Resource.
* POST (CREATE): Tạo mới một Resource.
* PUT (UPDATE): Cập nhật thông tin cho Resource.
* DELETE (DELETE): Xoá một Resource.

**Cách đặt tên hàm:**

* **Đối với NodeJS:**

Route.get(‘/’, function (req,res,next){});

->Trả về danh sách

Exports.list= fuction(){

//trả về danh sách

}

* **Đối với Laravel:**

Cấu hình Route/api.php

Route:: get (“Userlist”,[testController::class,’Userlist’]);

Hàm trong testController.php

public function Userlist(){

return response()->json(Users::get(),200);

}

**URL:**

Trọng tâm của REST quy định cách sử dụng các HTTP method (như *GET*, *POST*, *PUT*, *DELETE*...) và cách định dạng các URL cho ứng dụng web để quản các resource. Ví dụ với một trang blog để quản lý các bài viết chúng ta có các URL đi với HTTP method như sau:

* URL tạo bài viết: http://my-blog.xyz/posts. Tương ứng với HTTP method là POST
* URL đọc bài viết với ID là 123: http://my-blog.xyz/posts/123. Tương ứng với HTTP method là GET
* URL cập nhật bài viết với ID là 123: http://my-blog.xyz/posts/123. Tương ứng với HTTP method là PUT
* URL xoá bài viết với ID là 123: http://my-blog.xyz/posts/123. Tương ứng với HTTP method là DELETE

**Version:**

Khi chúng ta xây dựng 1 appp API thì việc quản lý version là điều quan trọng. Điều đó cảng quang trọng hơn khi chúng ta public API này với khách hàng theo 1 hợp đồng. Mỗi sự thay đổi thì sẽ hình thành 1 version mới.

Về việc sử dụng version trong Rails API chúng ta cần làm 2 việc:

* Thêm 1 route constraint : việc này sẽ lựa chọn 1 phiên bản trong request headers
* Namespace controllers: có các namespace khác nhau để xử những version khác nhau.

Các API có các mô tả như sau:

- Lấy danh sách tất cả các user

- Lấy user theo id

- Thêm (lưu mới) 1 user

- Sửa (cập nhật vào db) 1 user.

- Tìm kiếm user theo tên

- Xóa user theo id

Route/Api.php

*Route*:: get("Userlist",[*testController*::class,'Userslist']);                    // Lấy danh sách User

*Route*:: get("Userlist/{id}",[*testController*::class,'UserbyID']);                //Lấy User theo ID

*Route*:: get("Userlist/search/{name}",[*testController*::class,'search']);         //tìm User theo tên

*Route*:: post("Userlist",[*testController*::class,'adUser']);                      //Thêm User

*Route*:: put("Userlist/{id}",[*testController*::class,'Update']);                  // Update User

*Route*:: delete("Userlist/{id}",[*testController*::class,'delete']);               //Xóa User theo ID

Testcontronller.php

    public function Userslist() //Lấy danh sách User

    {

        return response()->json(*Users*::get(),200);

    }

    public function UserbyID($id) //Lấy User theo Id

    {

        return response()->json(*Users*::find($id),200);

    }

    public function adUser(*Request* $request) //Thêm User

    {

        $user = new *Users*;

        $user->name = $request->name;

        $user->email = $request->email;

        $user->password = $request->password;

        $user->Phone = $request->Phone;

        $result= $user->save();

        return response()->json($result,201);

    }

    public function Update($id,*Request* $request){ //update User

        $user = *Users*::Find($id);

        $user->name=isset($request->name)?$request->name:$user->name;

        $user->email=isset($request->email)?$request->email:$user->email;

        $user->password=isset($request->password)?$request->password:$user->password;

        $user->Phone=isset($request->Phone)?$request->Phone:$user->Phone;

        $result= $user->save();

        return response()->json($result,200);

    }

    public function delete($id){ //Xóa User

        $user = *Users*::Find($id);

        $result =$user->delete();

        return response()->json($result,201);

    }

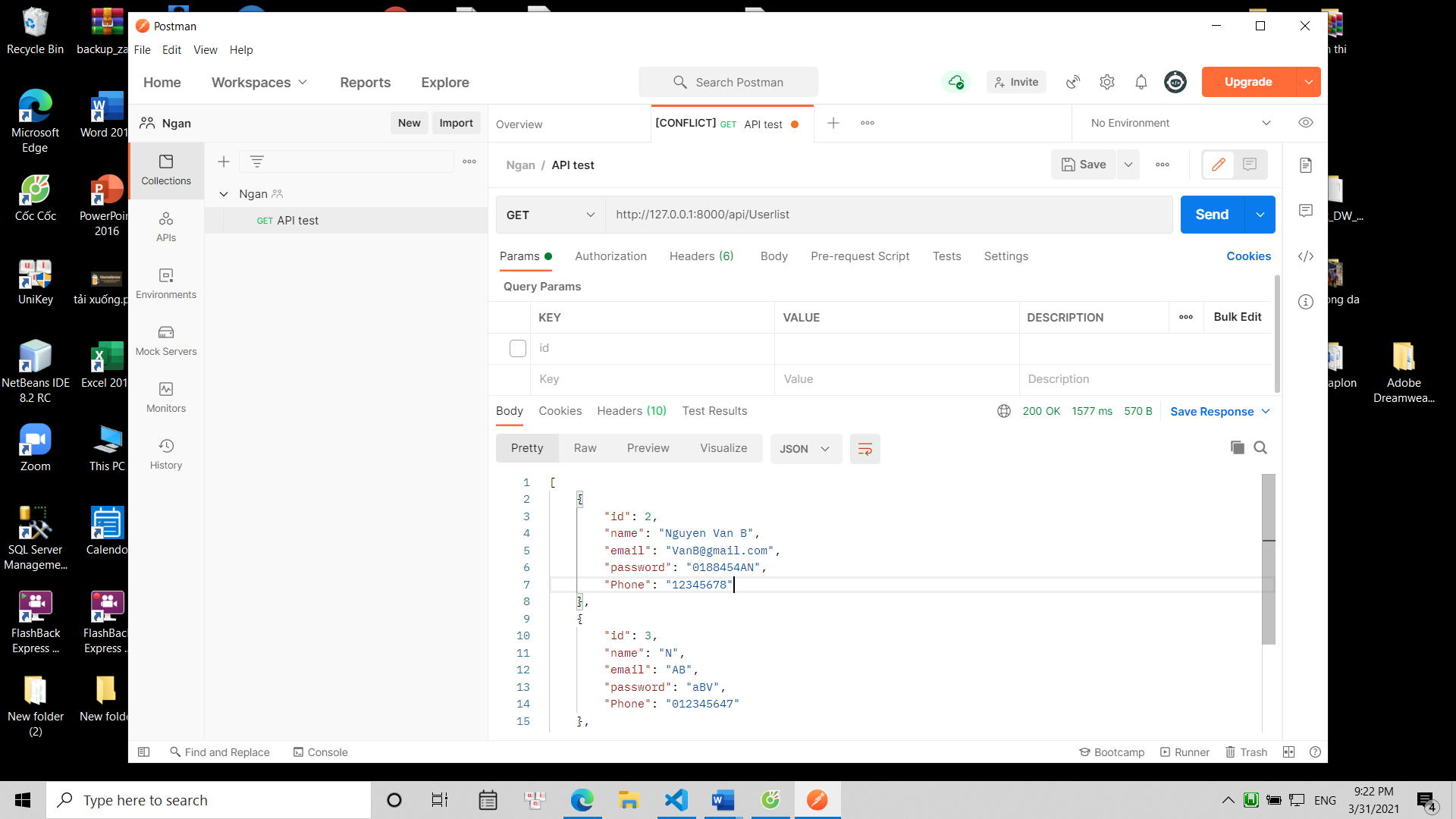
    public function Search($name){ // Tìm kiếm User

        return *Users*::where("name","Like","%".$name."%")->get();

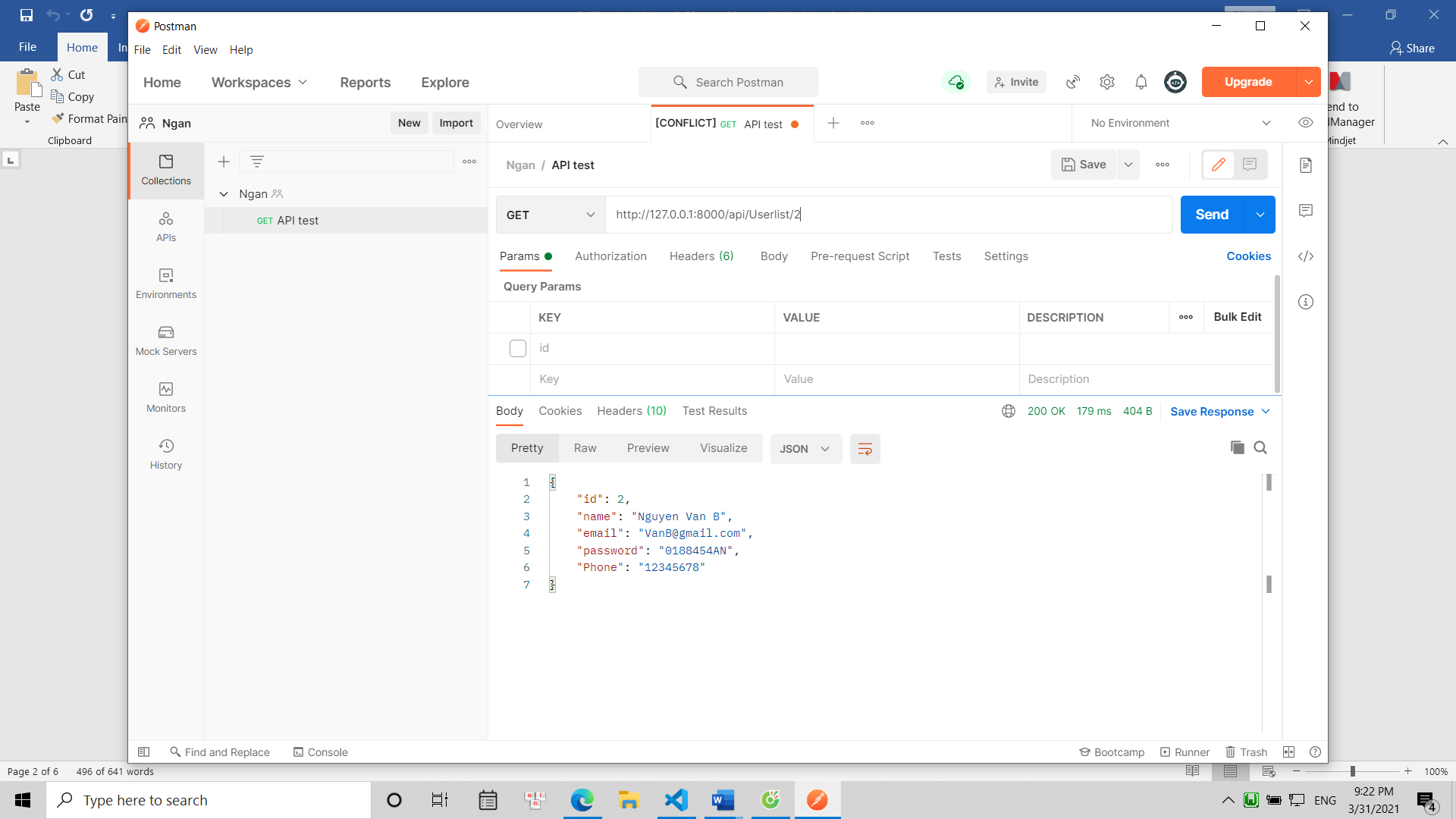
    }

Kết quả

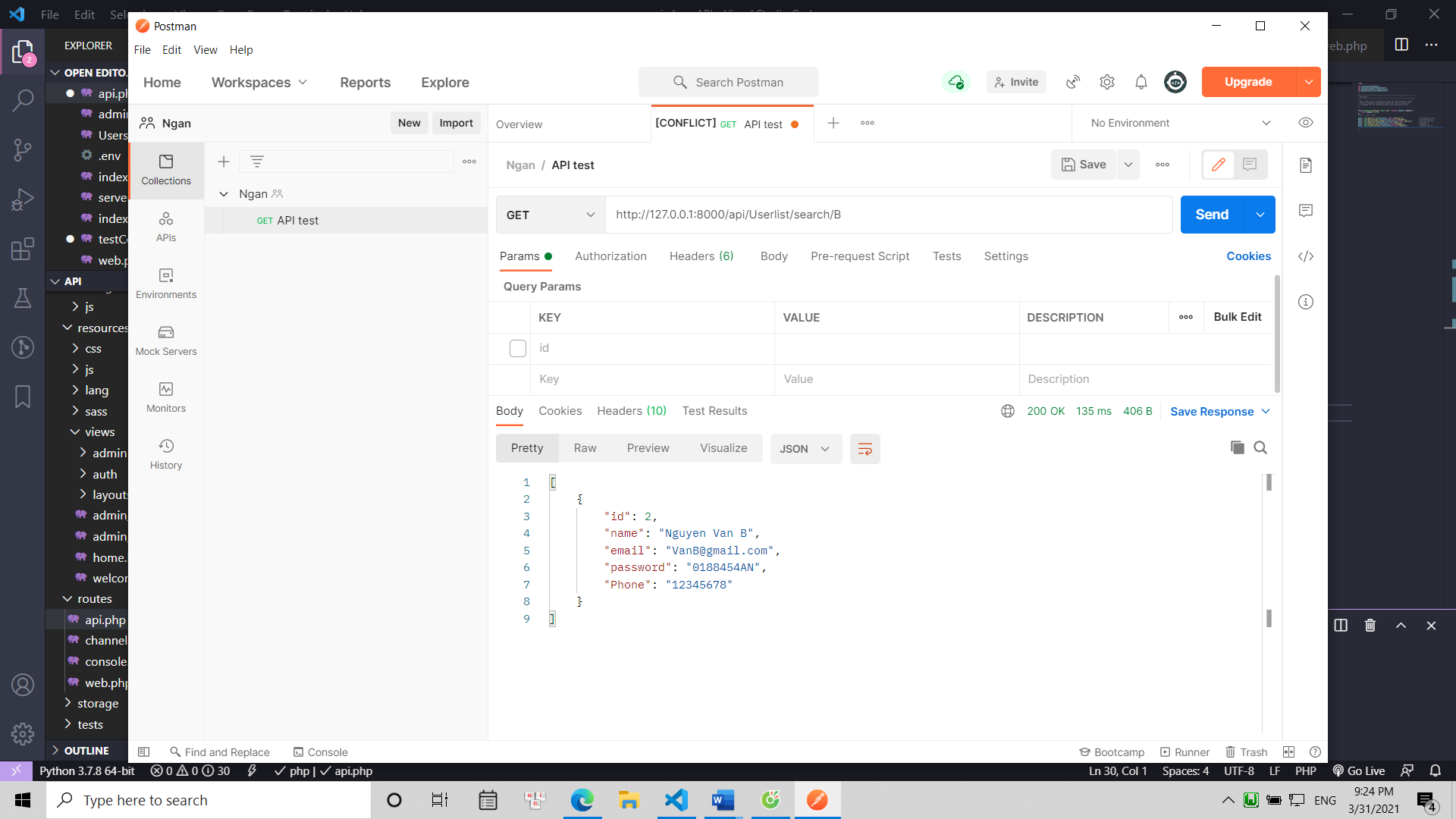
Hiện danh sách



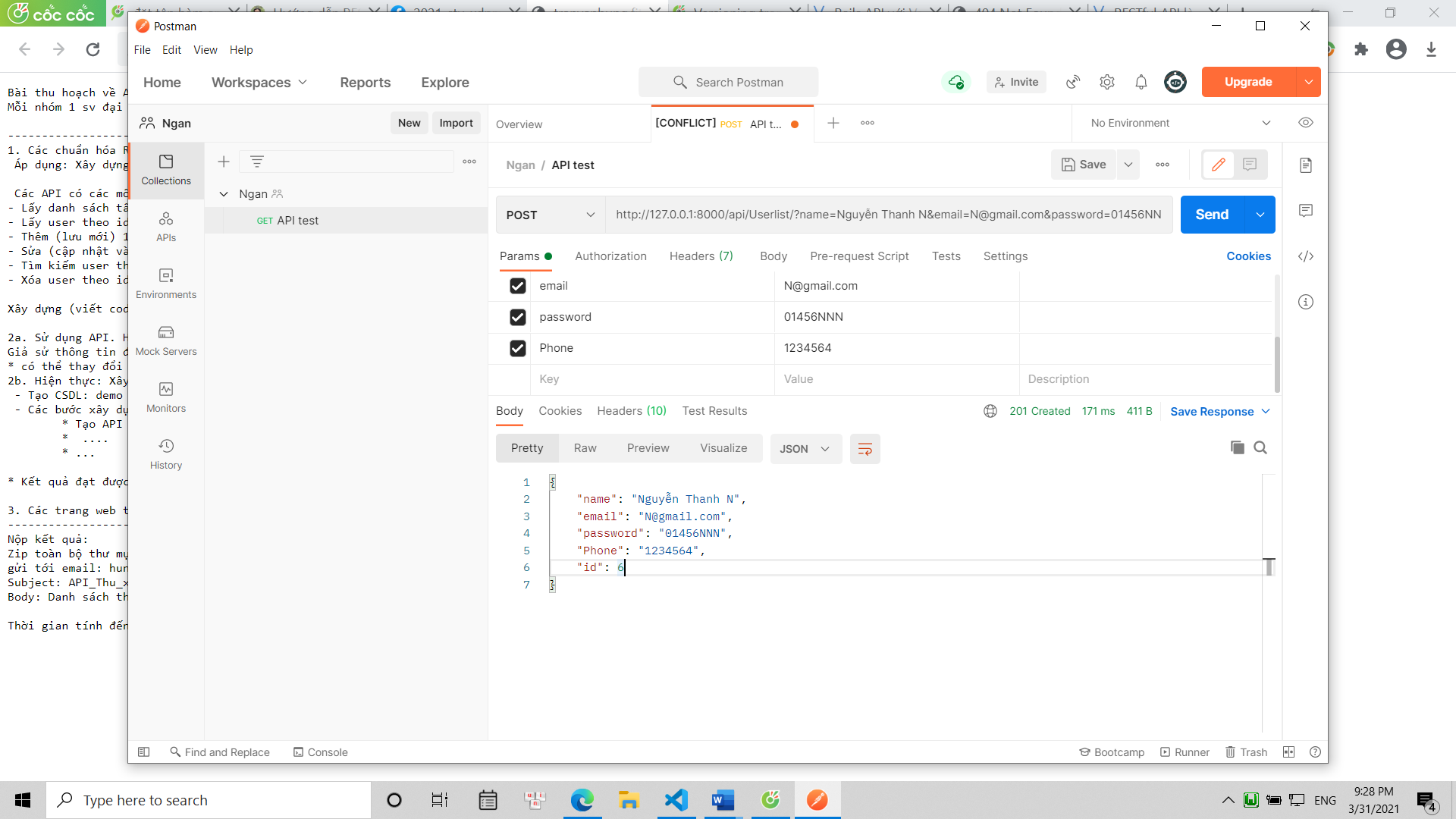
Tìm theo Id



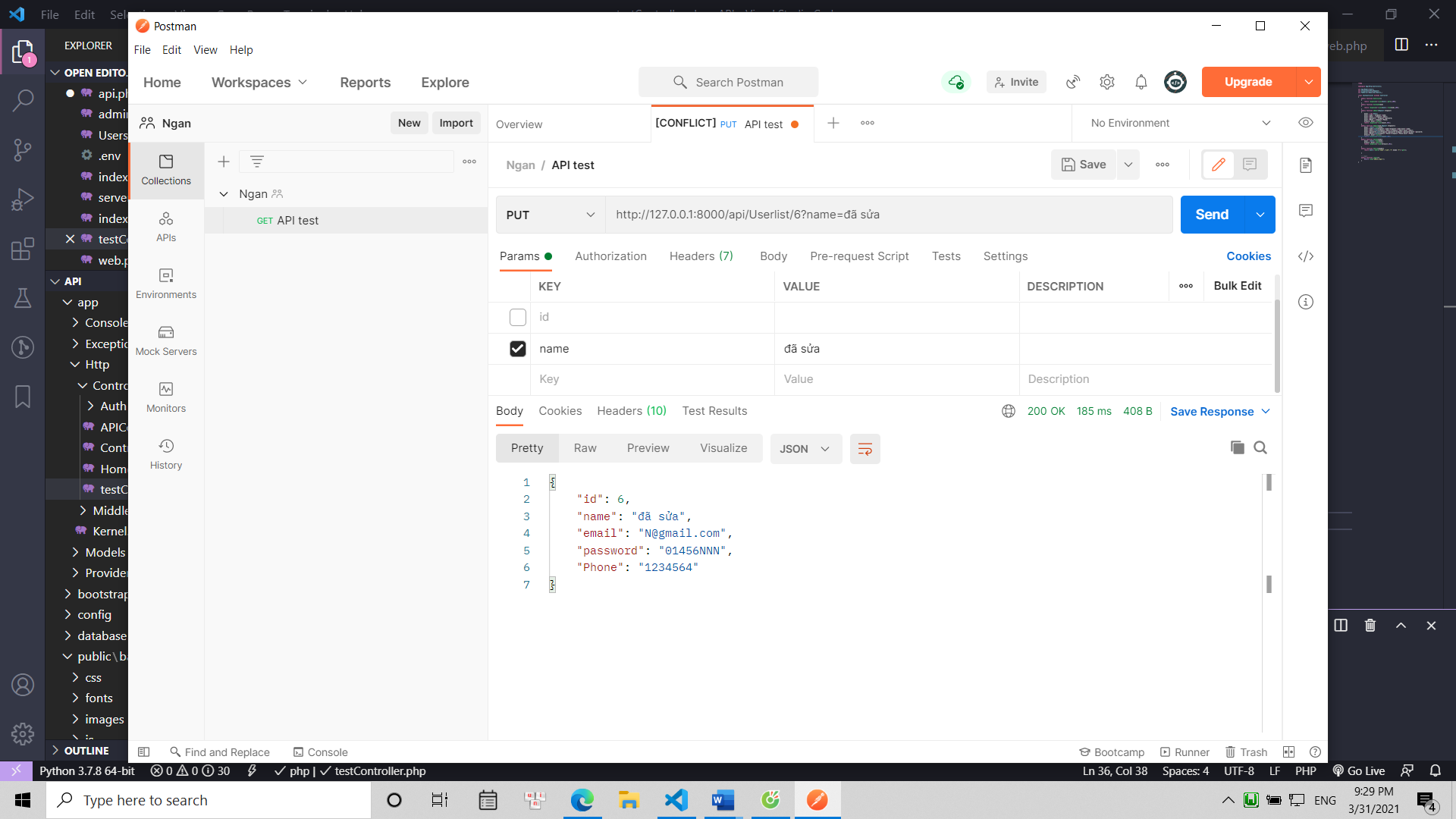
Tìm theo Tên



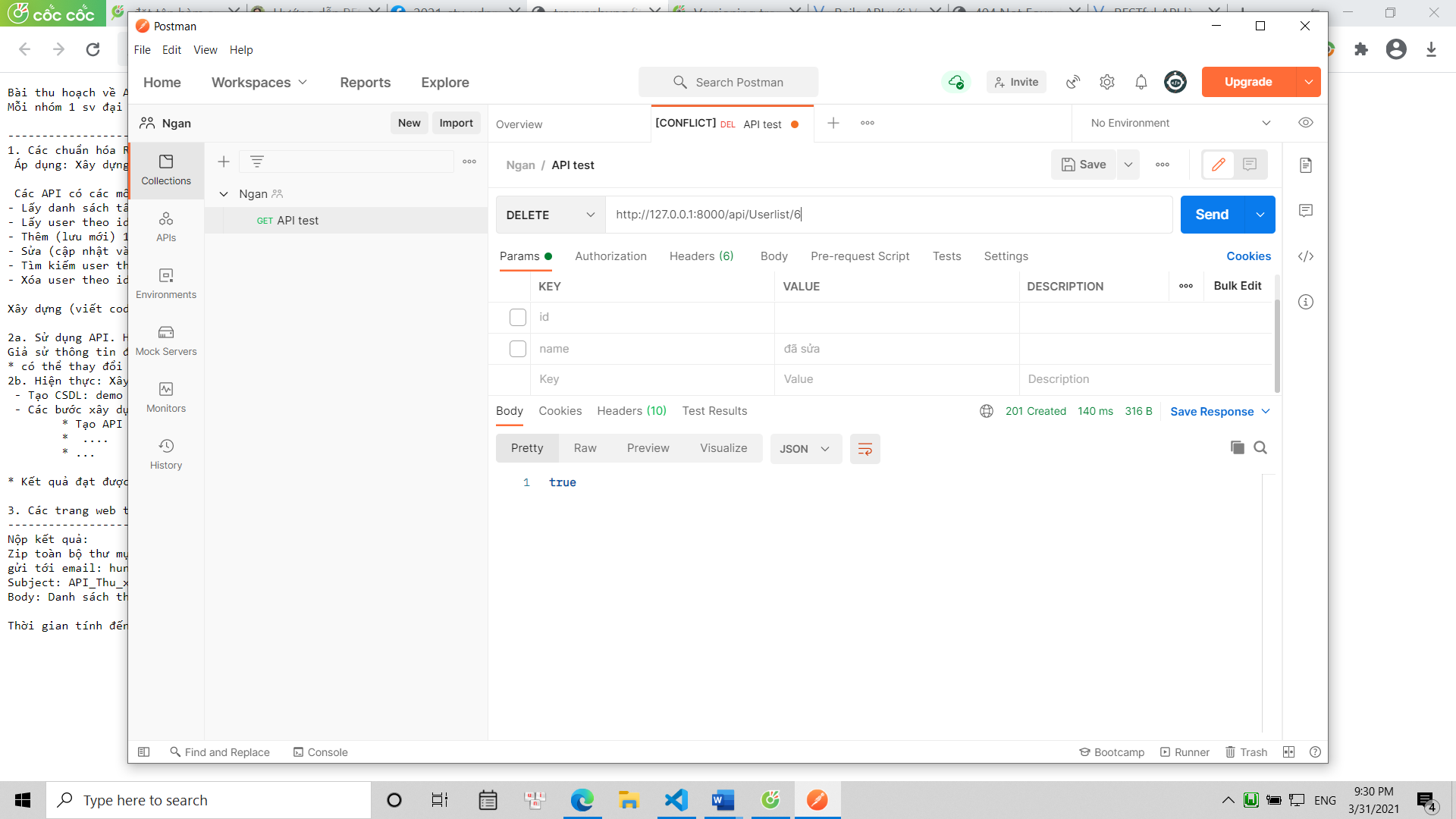
Thêm



Sửa



Xóa



**Câu 2a**

Sử dụng xác thực OAuth tài khoản Google Plus

Dịch vụ này cho phép người dùng cung cấp các thông tin xác thực, như để đăng nhập vào website của bạn chẳng hạn. Người dùng sẽ được chuyển hưởng đến dịch vụ của Google và Google sẽ cung cấp lại các thông tin cần thiết để xác thực như ID, username, email ... tùy thuộc vào sự cho phép của người dùng ... (thông tin và quyền hạn do bạn thiết lập - thường thì với login thì bạn chỉ cần xin email của người dùng là đủ).

Để sử dụng được API của Google trước tiên bạn cần có một dự án (ứng dụng) tại **Google API**:

**2b**

Lấy thông tin người dùng từ google sử dụng Laravel 8

Bước 1

Tạo laravel 8 Auth sử dụng jetstream

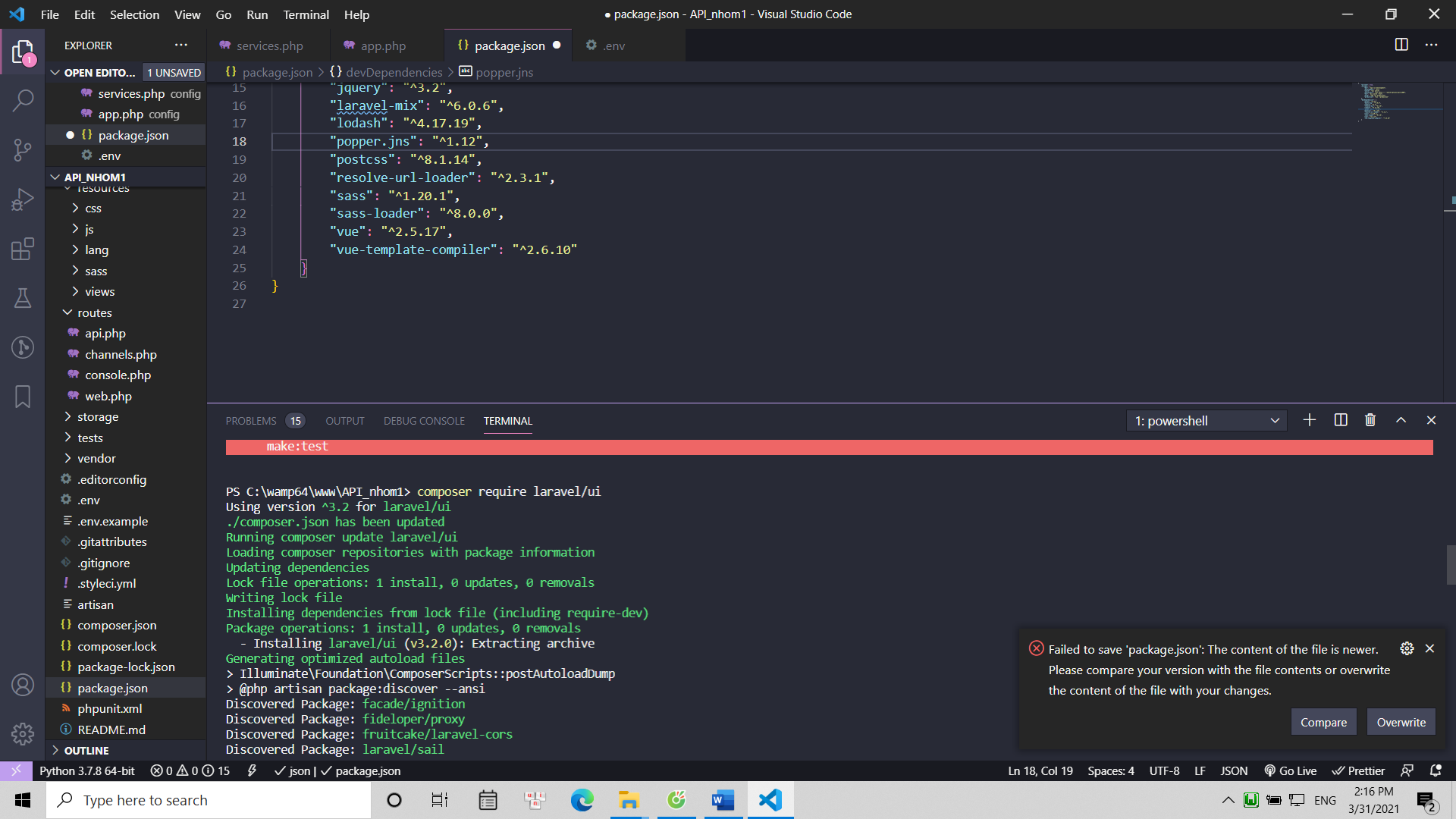
1.composer **require** laravel/jetstream

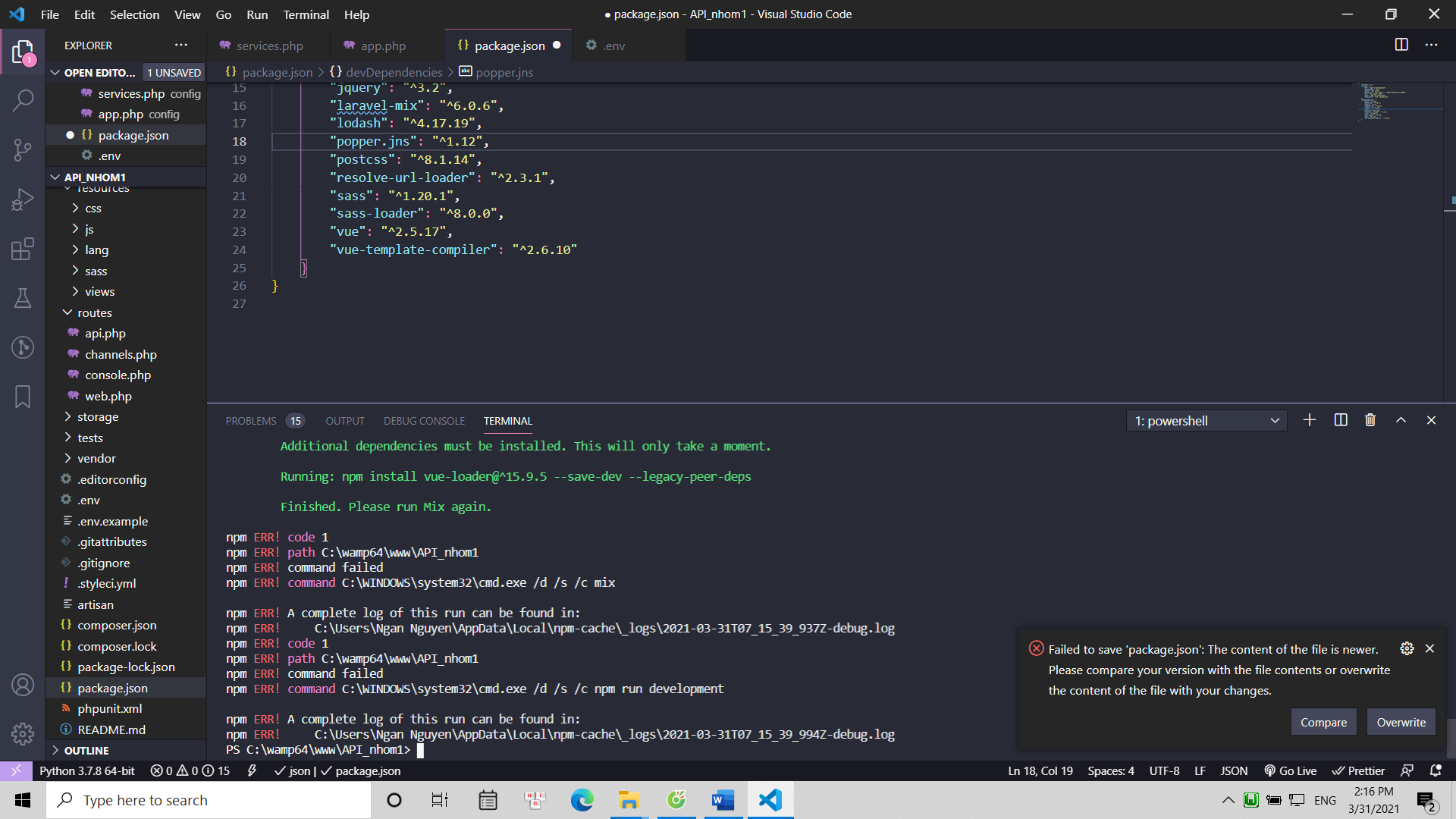
2.php artisan jetstream:install livewire

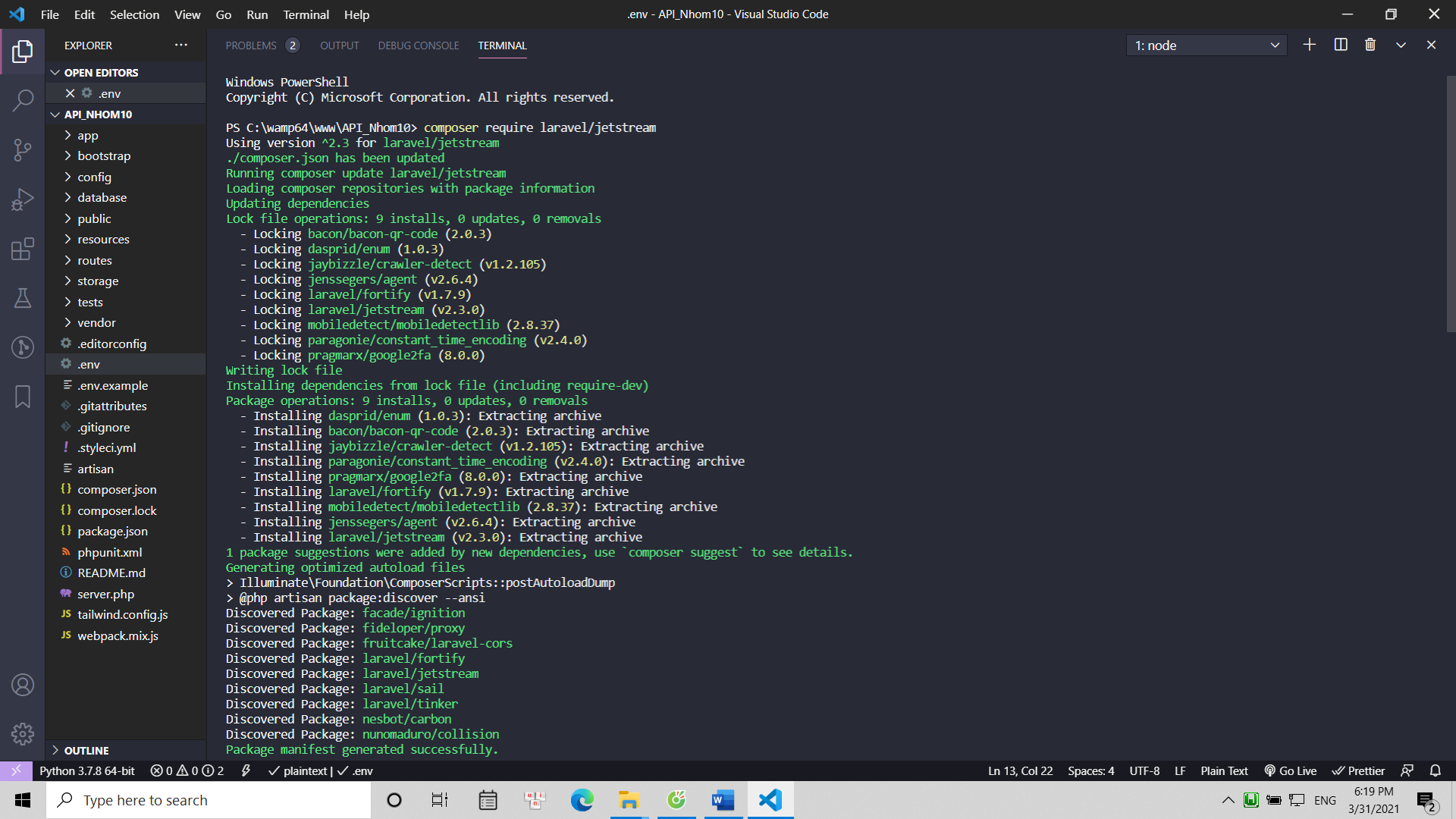
3.npm install

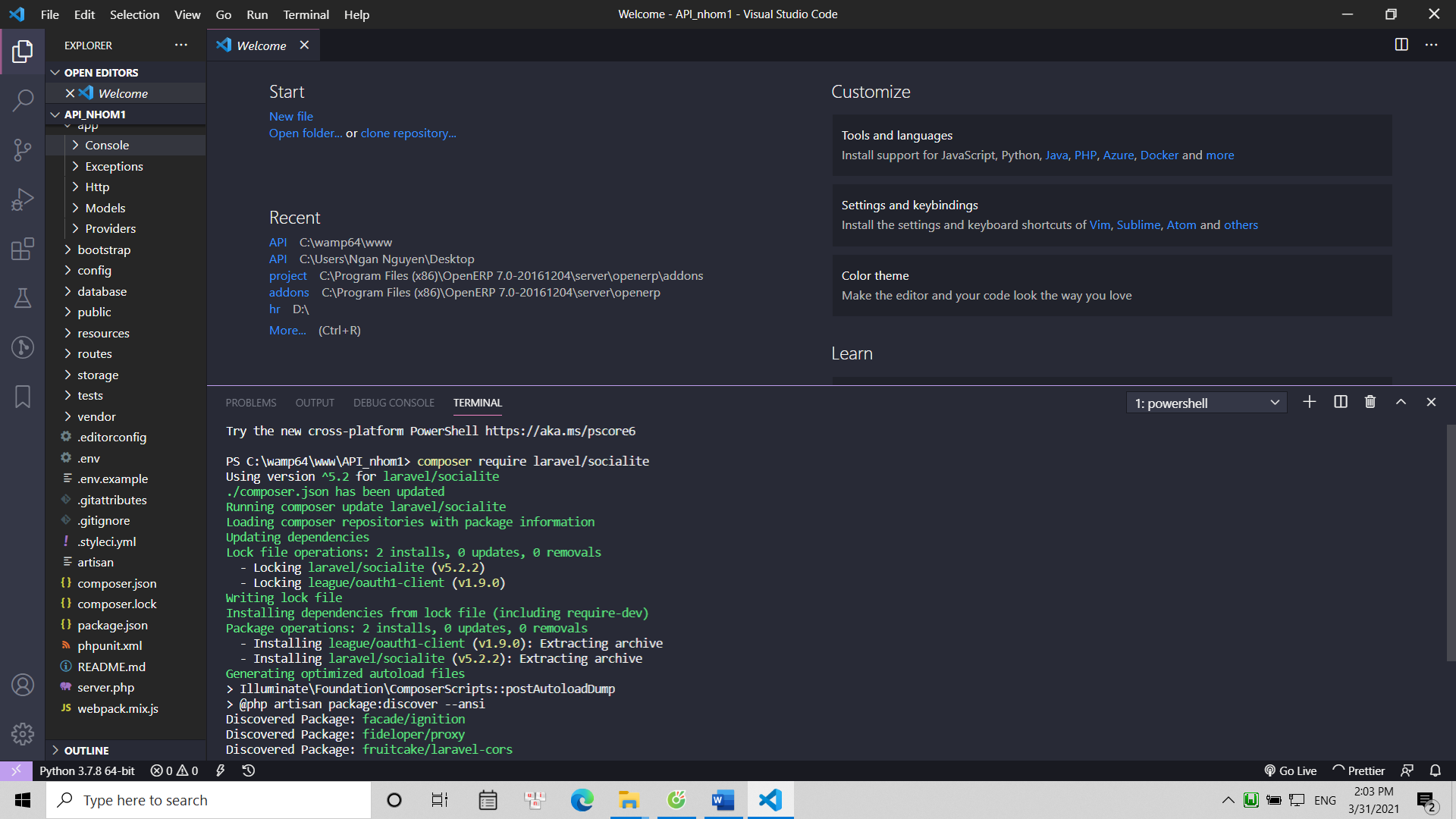
4.npm run dev

5.php artisan migrate

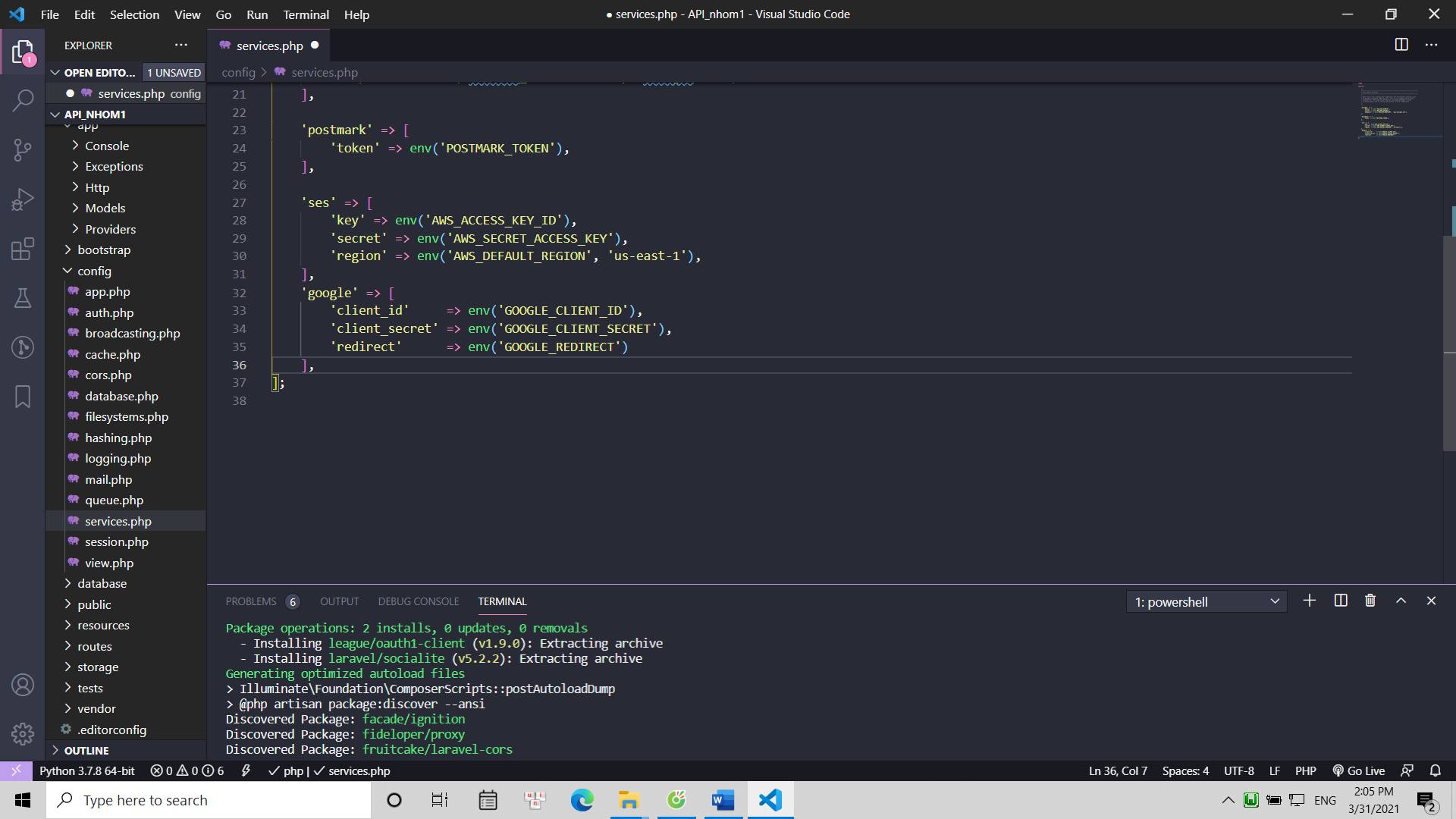






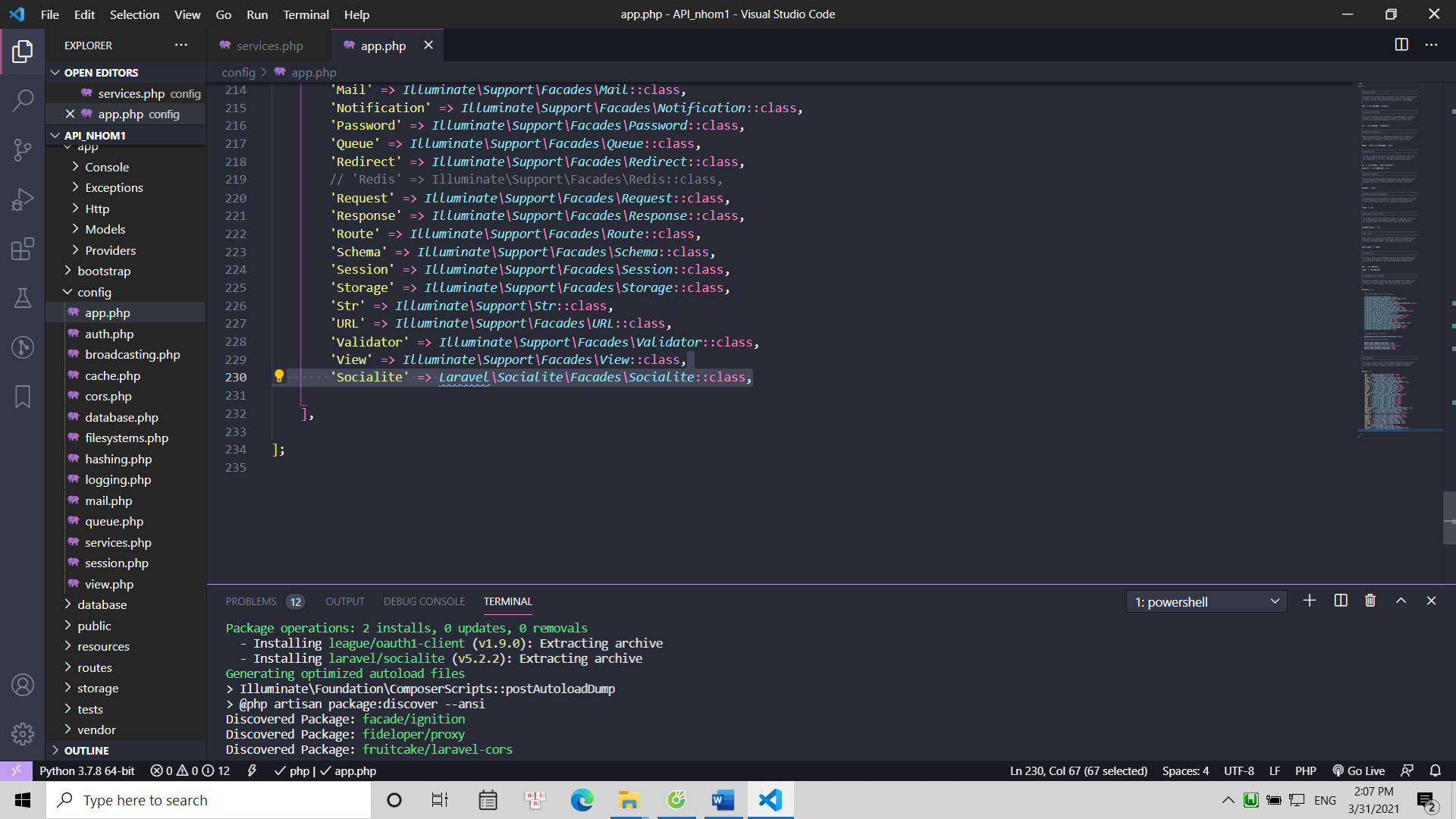


Bước 2 Cấu hình service.php



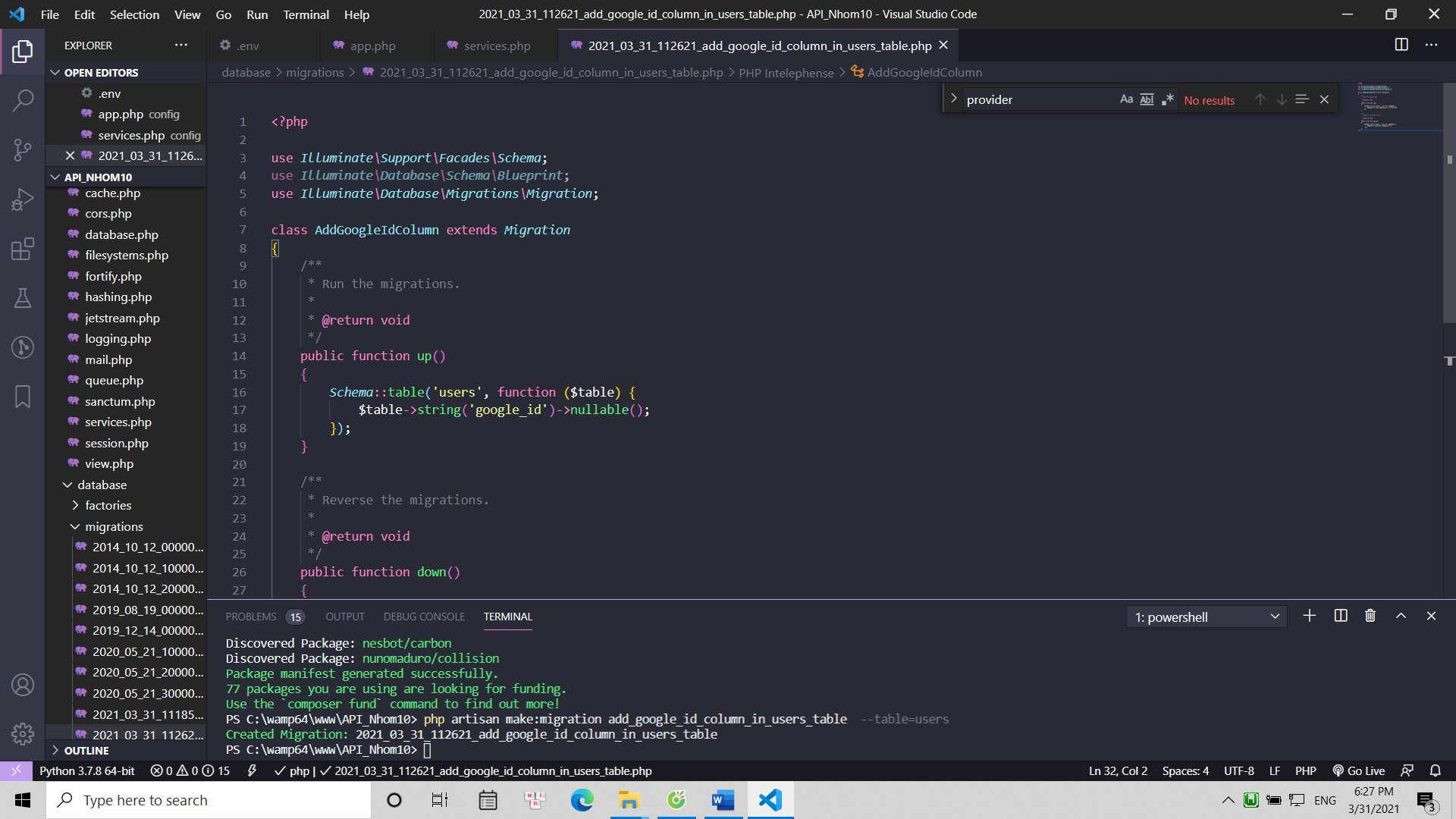
Bước 3 :Thêm đường dẫn vào app.php



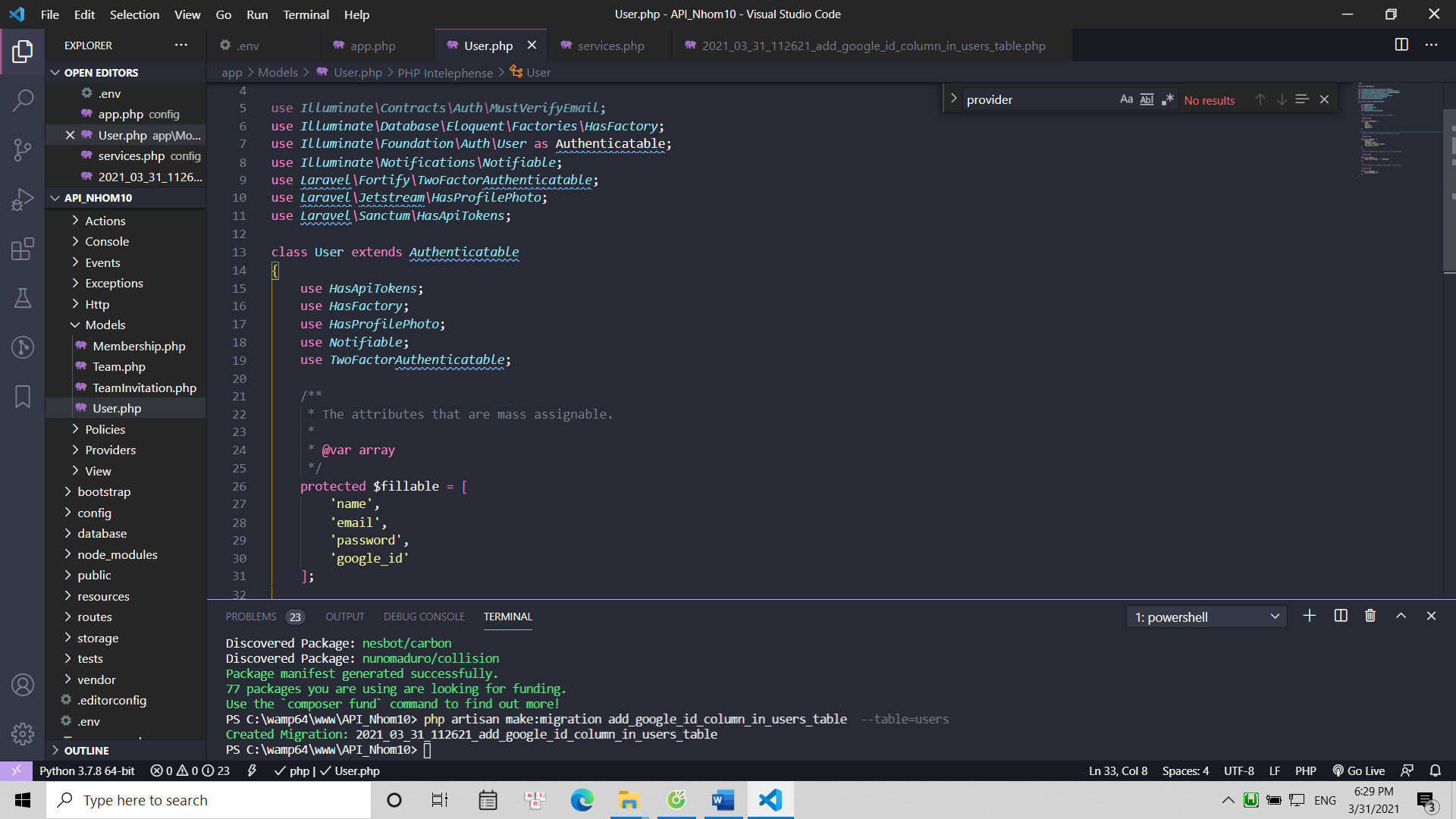


## Bước 4 :Tạo migrate User

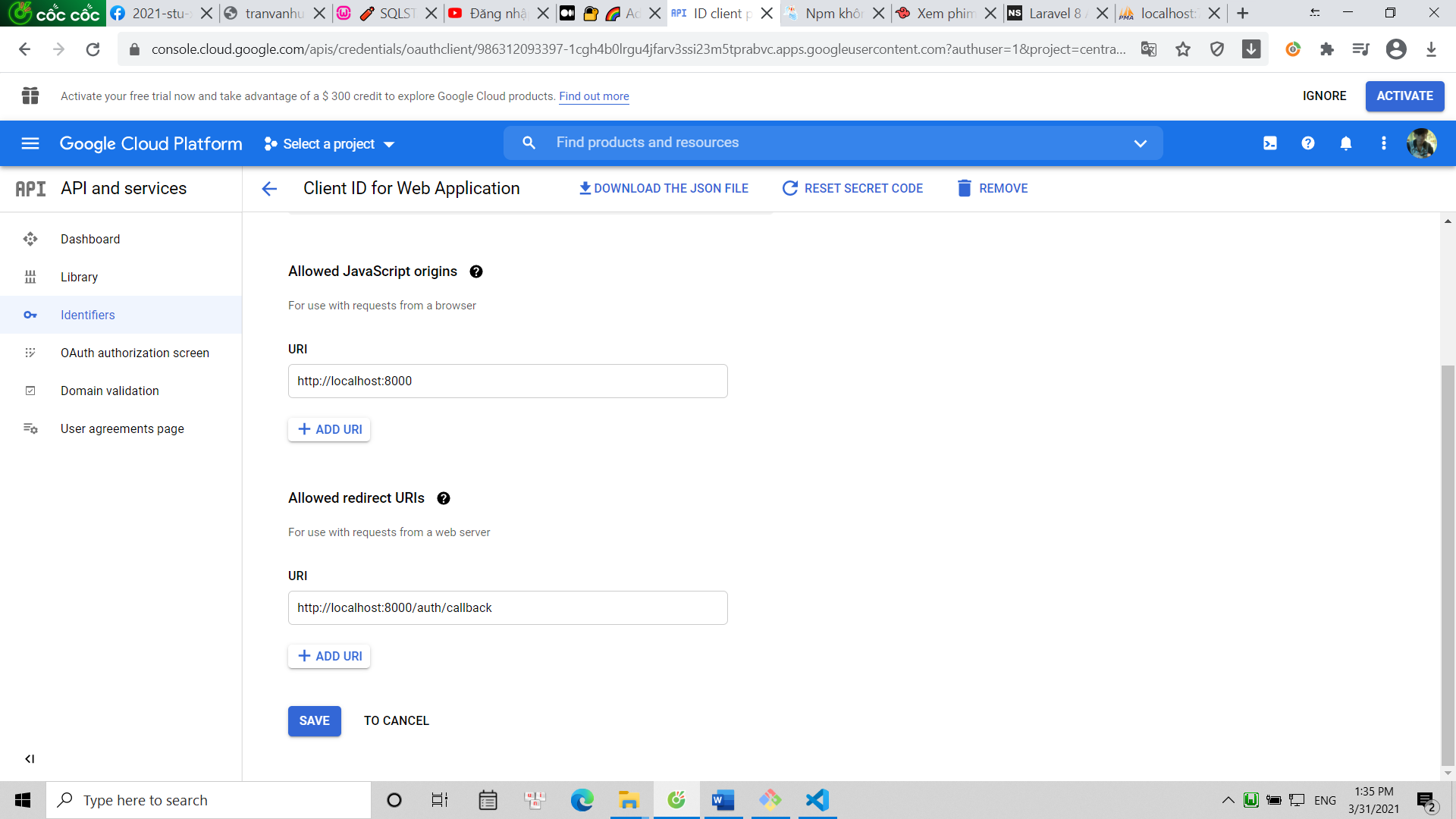
php artisan make:migration add\_google\_id\_column\_in\_users\_table --table=users



Bước 5 :Thiết kế models User.php



Bước 6:Cấu hình Google API



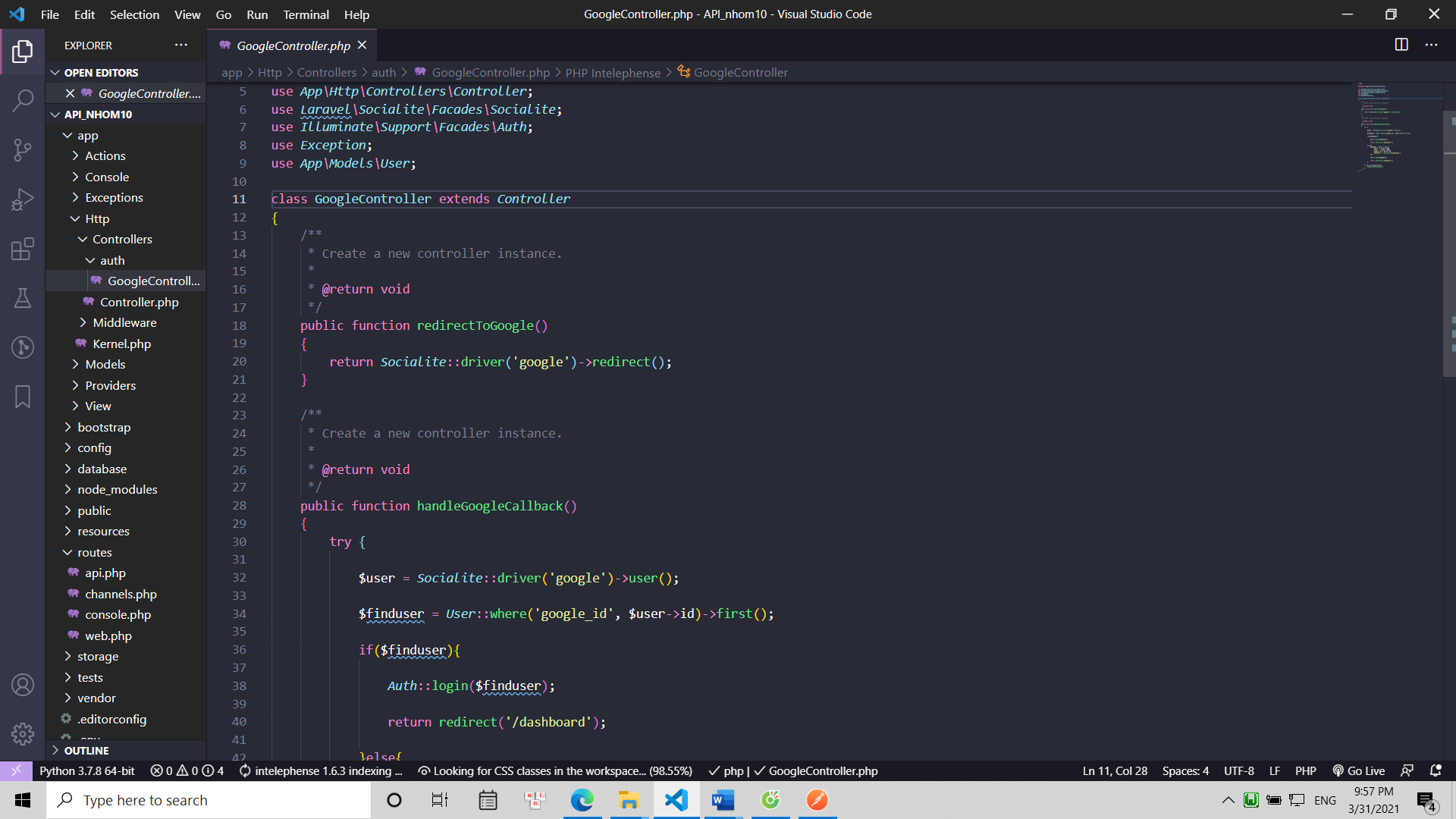
Bước 7 :Tạo routes

use *App\Http\Controllers\Auth\GoogleController*;

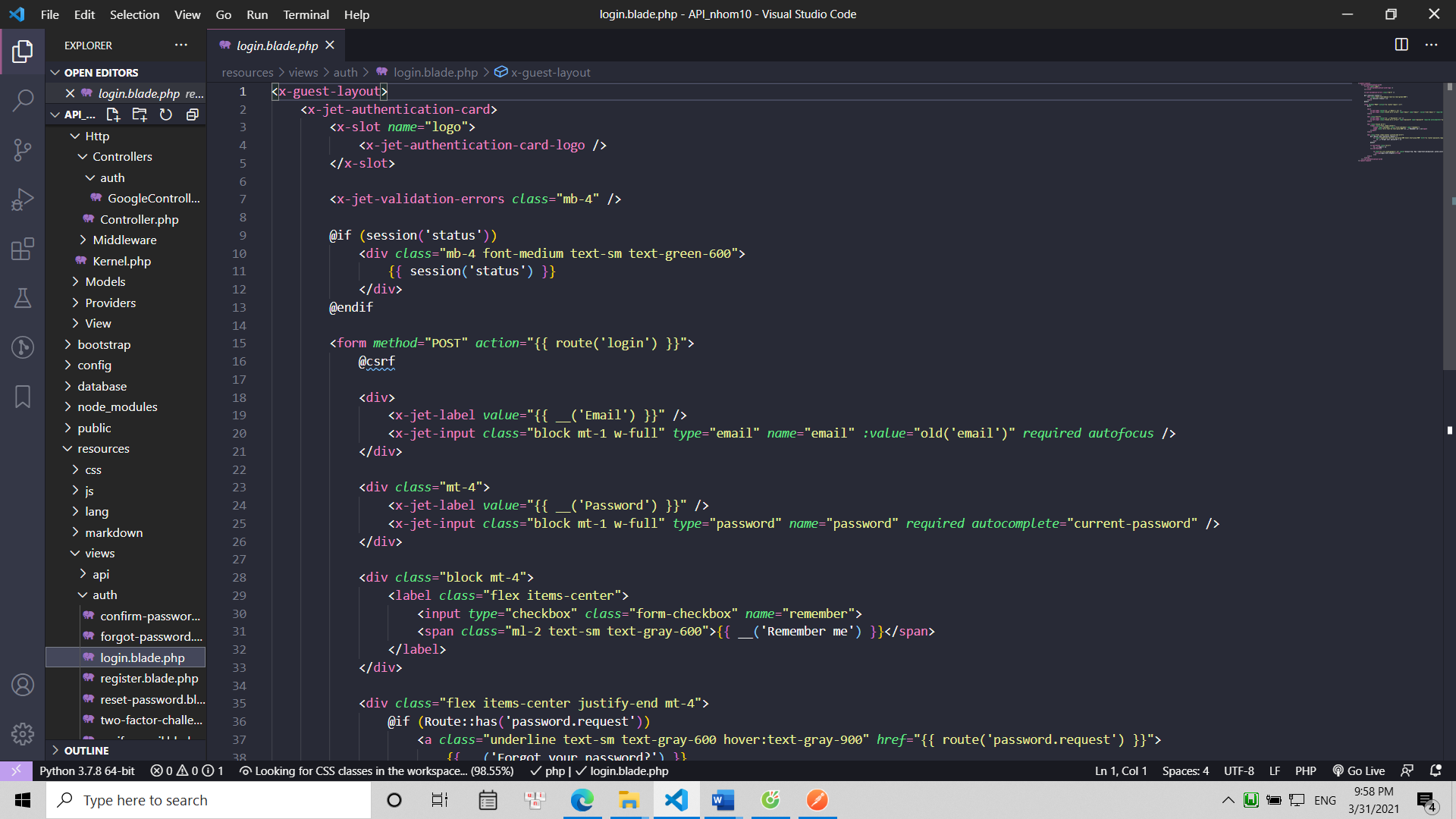
*Route*::get('auth/google', [*GoogleController*::class, 'redirectToGoogle']);

*Route*::get('auth/google/callback', [*GoogleController*::class, 'handleGoogleCallback']);

Bước 8 : tạo Controller



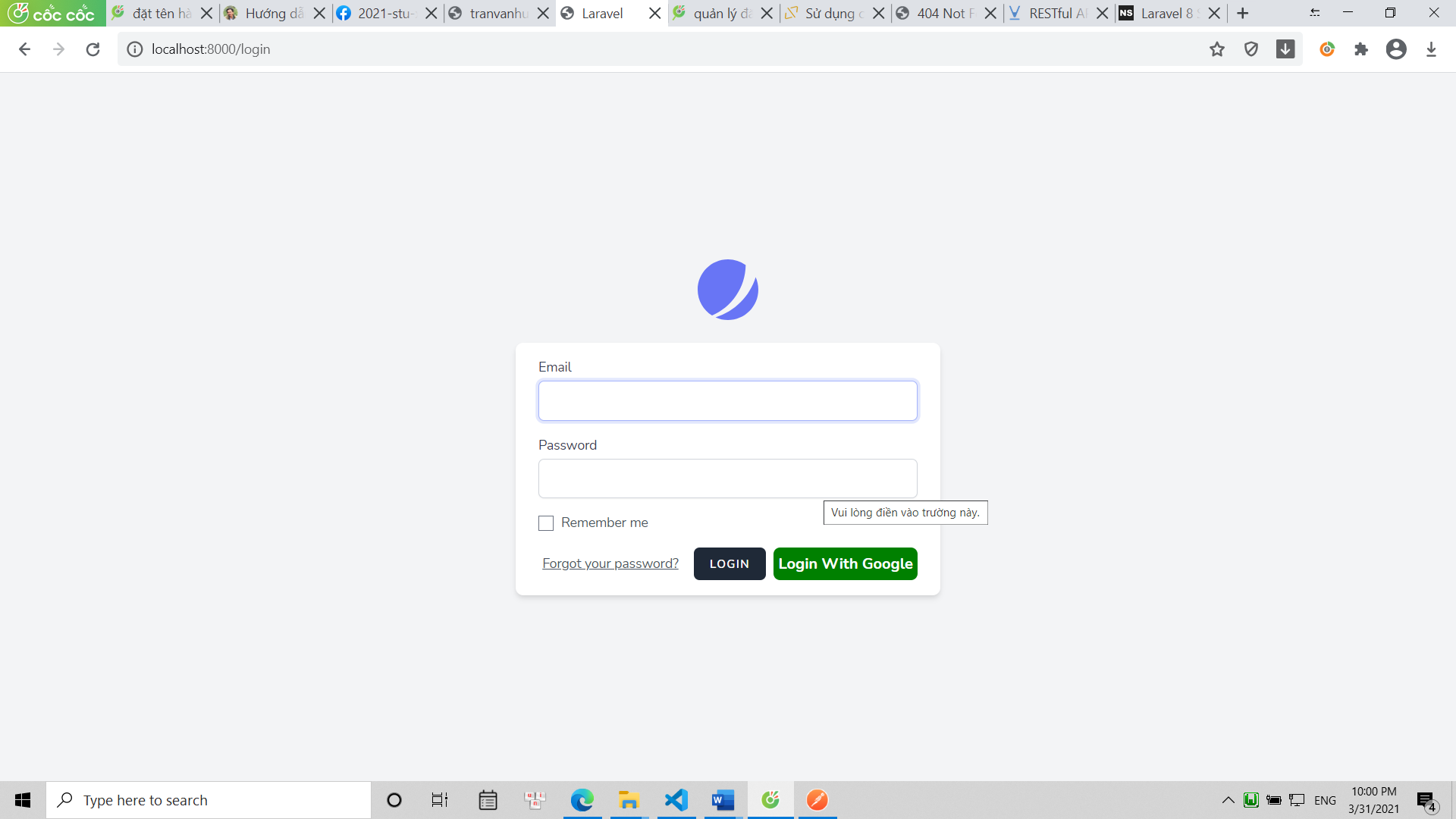
Bước 9 : update Login.blade.php

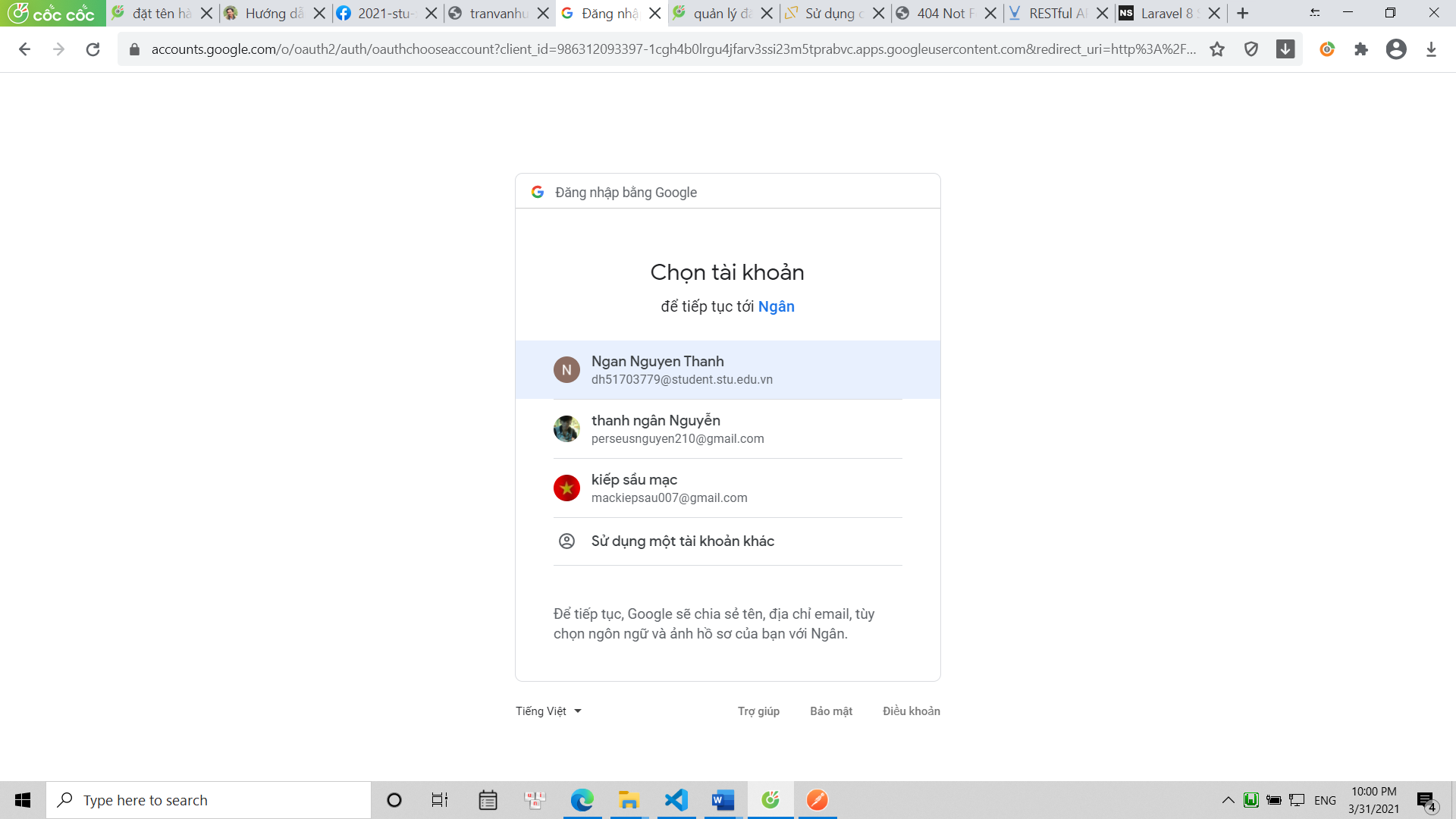


Bước 10 : Chạy

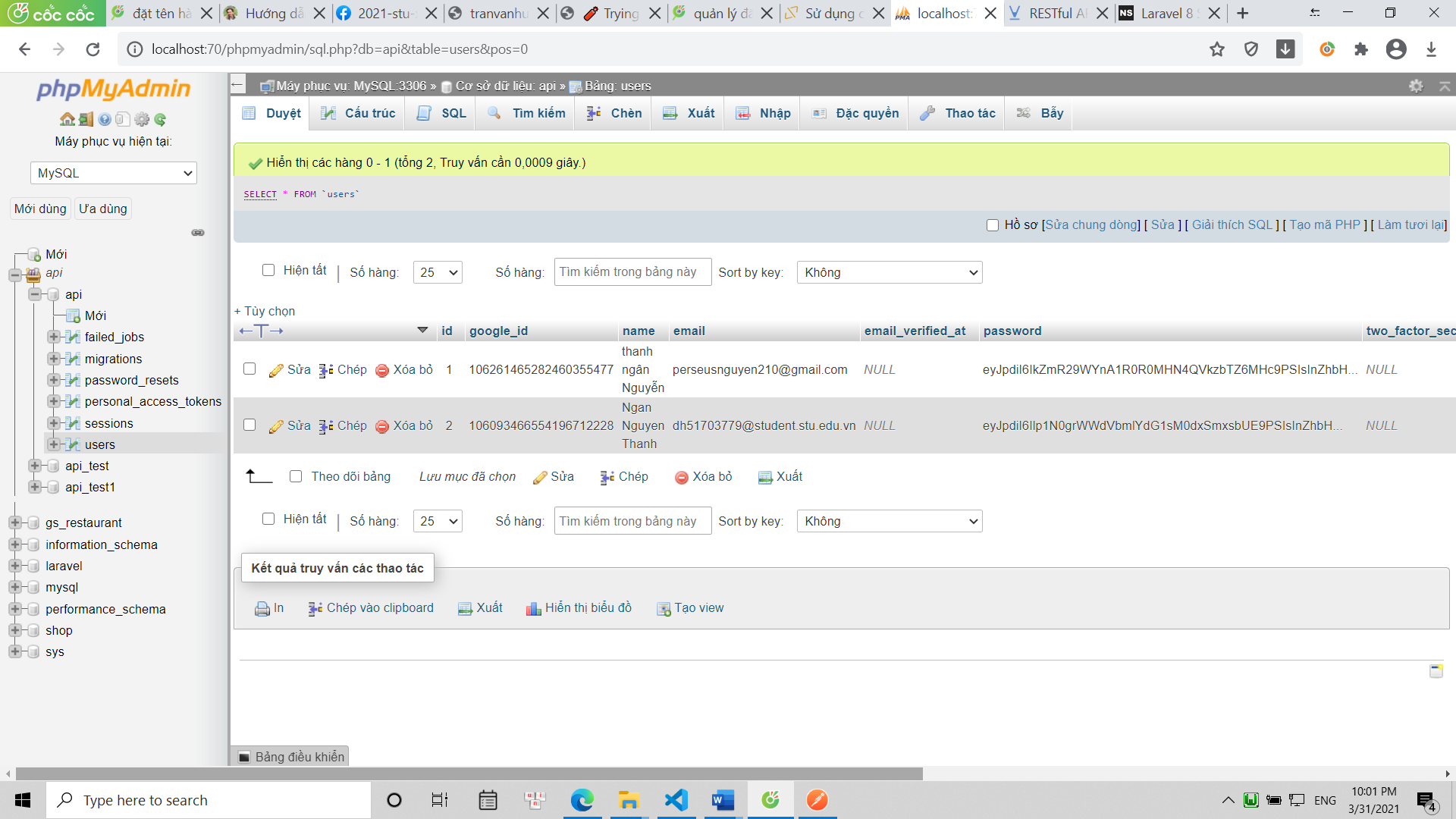
Php artisan serve

Bước 11 : Đăng nhập





Kết quả



**3. Các trang web trong https://redcircleland.com/, trang nào không đạt chuẩn SEO? vì sao?**

Các trang web này không đạt tiêu chuẩn SEO trong <https://redcircleland.com/>:

- <https://redcircleland.com/chi-tiet-bds/110>

Không hiển thị lên google khi tìm kiếm với tên<https://redcircleland.com/chi-tiet-bds/110>

- <https://redcircleland.com/chi-tiet-bds/88>

Không hiển thị lên google khi tìm kiếm với tên

Kết quả trang ko thân thiện với thiết bị di động <https://search.google.com/test/mobile-friendly>

- https://redcircleland.com/chi-tiet-bds/108

Không hiển thị lên google khi tìm kiếm với tên <https://redcircleland.com/chi-tiet-bds/108>

Kết quả trang ko thân thiện với thiết bị di động <https://search.google.com/test/mobile-friendly>

- https://redcircleland.com/chi-tiet-bds/107

Do:

Không hiển thị lên google khi tìm kiếm với tên: https://redcircleland.com/chi-tiet-bds/107

<https://redcircleland.com/chi-tiet-bds/104>

Không hiển thị lên google khi tìm kiếm

<https://redcircleland.com/chi-tiet-bds/103>

Không hiển thị lên google khi tìm kiếm