****

Web应用与开发

实 验 报 告

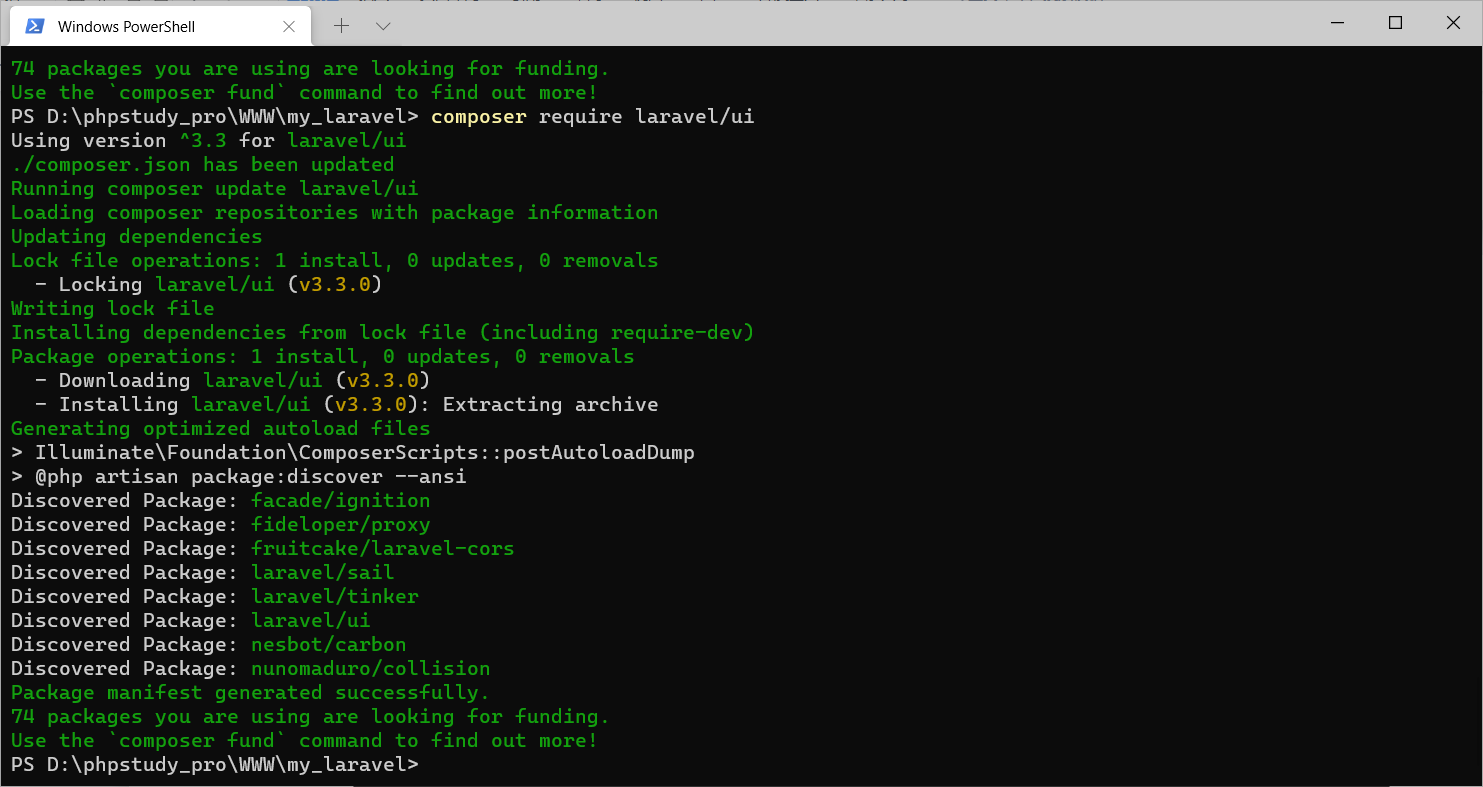
|  |  |
| --- | --- |
| **学 院** | 网络空间安全学院 |
| **专 业** | 网络工程 |
| **班 级** | 19272401 |
| **学 号** | 19270811 |
| **学生姓名** | Hqz |
| **教师姓名** | 胡伟通 |
| **完成日期** | 2021-6月 |
| **成 绩** |  |

**作业4 RESTful API的实现**

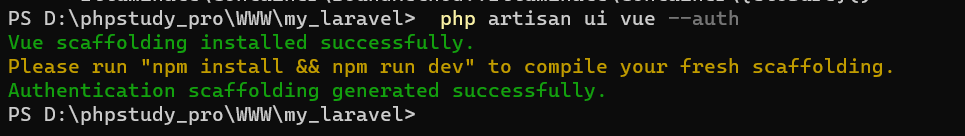
1. **实验目的和要求**
2. 实验目的：基于Laravel框架和RESTful设计准则，按照前后端分离的开发形式来实现简单的用户注册和登录。
3. **实验要求**：（1）实现Email形式的注册功能和相应的登录功能，注册部分具备邮件激活功能（使用Laravel的邮件发送机制或第三方组件）；（2）实现忘记密码时通过重置密码邮件设置新密码（使用Laravel的邮件发送机制或第三方组件）；（3）包含对某个物品（自己选择）的操作，以RESTful API风格进行；（4）如果使用JWT认证（JSON Web Tokens），后面会酌情加分；（5）不要求提供前端方面的实现，即不需要做网站或APP，此外，对API的实际使用在Postman上进行，并进行截图。
4. **实验内容**
5. 实现Email形式的注册功能和相应的登录功能，注册部分具备邮件激活功能（使用Laravel的邮件发送机制或第三方组件）

基础构建，生成需要的路由和视图

打开终端输入：composer require laravel/ui 成功后如下图

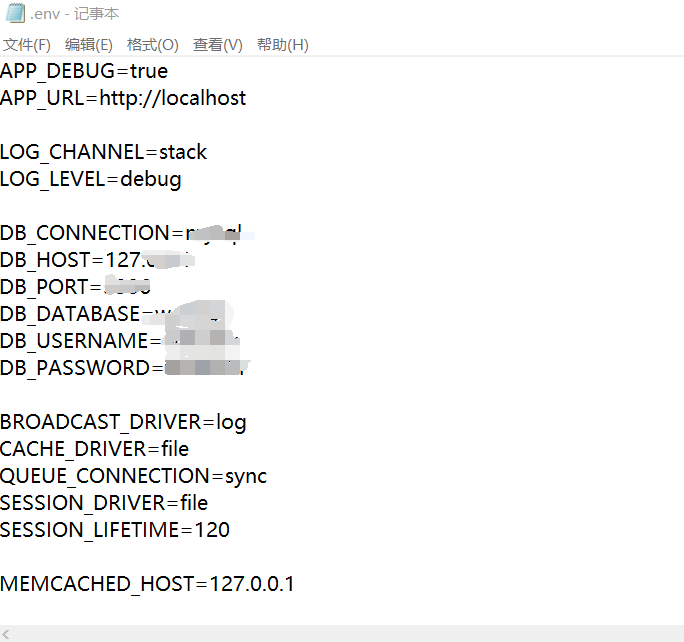


再输入 php artisan ui vue --auth成功



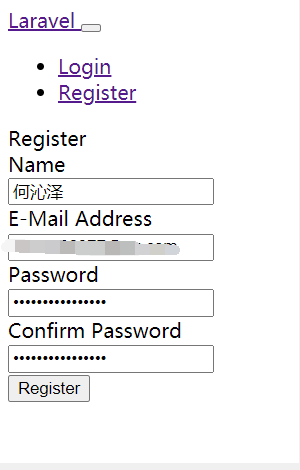
此时进行数据库的配置

在.env中进行配置



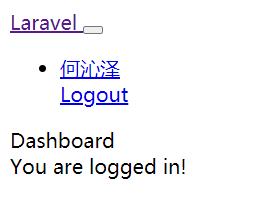
命令行生成larave登录认证需要的表数据库里面会有

尝试登录和注册功能





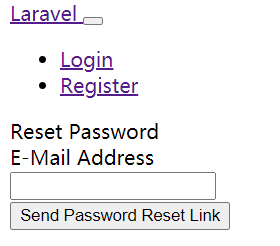
此时注册成功



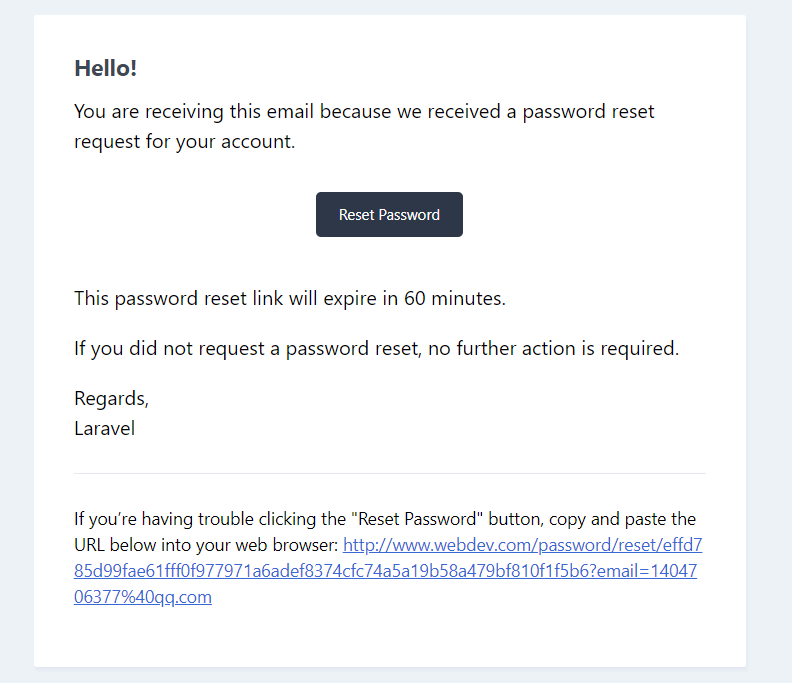


1. 实现忘记密码时通过重置密码邮件设置新密码（使用Laravel的邮件发送机制或第三方组件）

因为重置密码的路由和视图在laravel进行php artisan make:auth时已经自动生成了，所以现在可以直接通过/password/reset或者点击登录页面的For got your password链接进入重置密码页面



QQ邮箱收到邮件

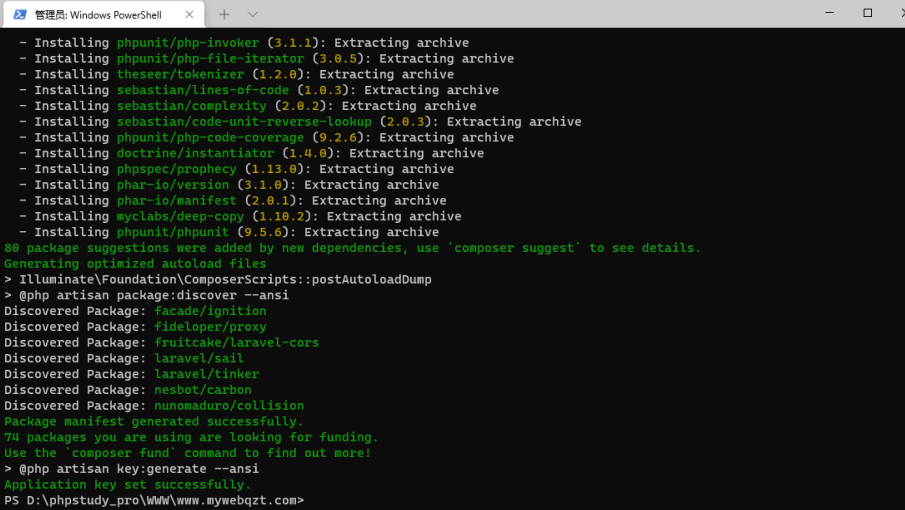


点击进入 修改密码

新的构建

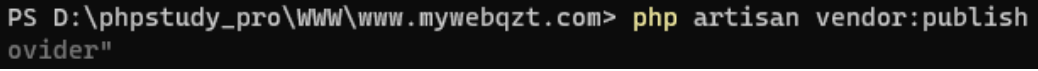
1.首先通过 composer create-project --prefer-dist laravel/laravel jwt 命令，创建新的

Laravel 项目，这个命名会在 jwt 目录下创建一个新的 laravel 项目

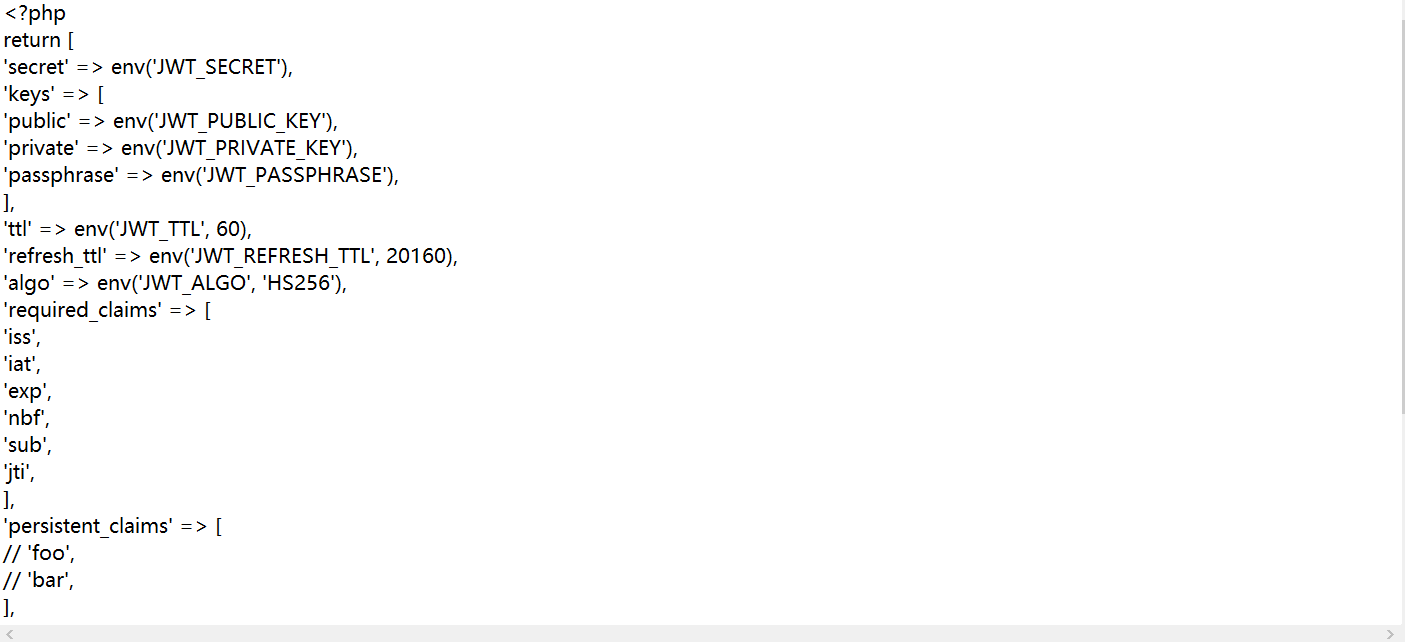


2.安装 tymon/jwt-auth 扩展包

运行一下命令获取 dev-develop 版本的 JWT 包 3.发布配置文件



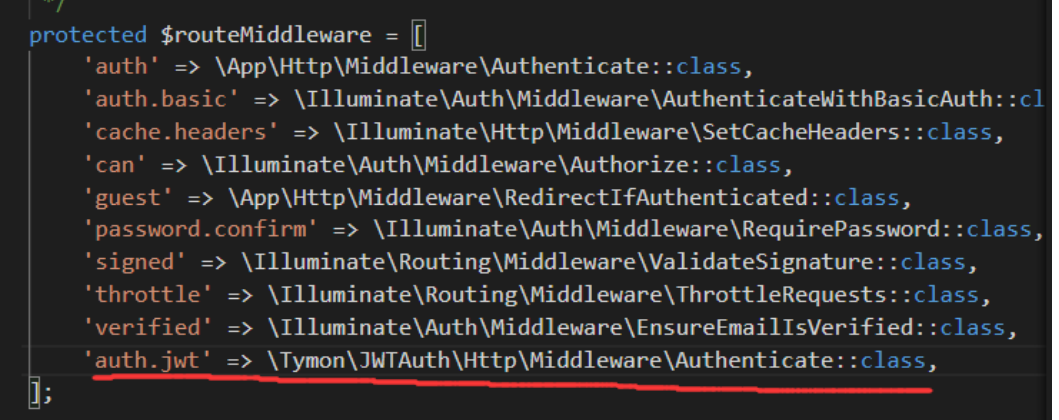
该命令会生成 config/jwt.php 配置文件



4.通过运行下面的命令来生成密钥以便于签发令牌

php artisan jwt:secret

1. 在 app/Http/Kernel.php 中注册 auth.jwt 中间件，在图中所示位置添加代码



1. 设置路由为 routes/api.php 添加以下代码

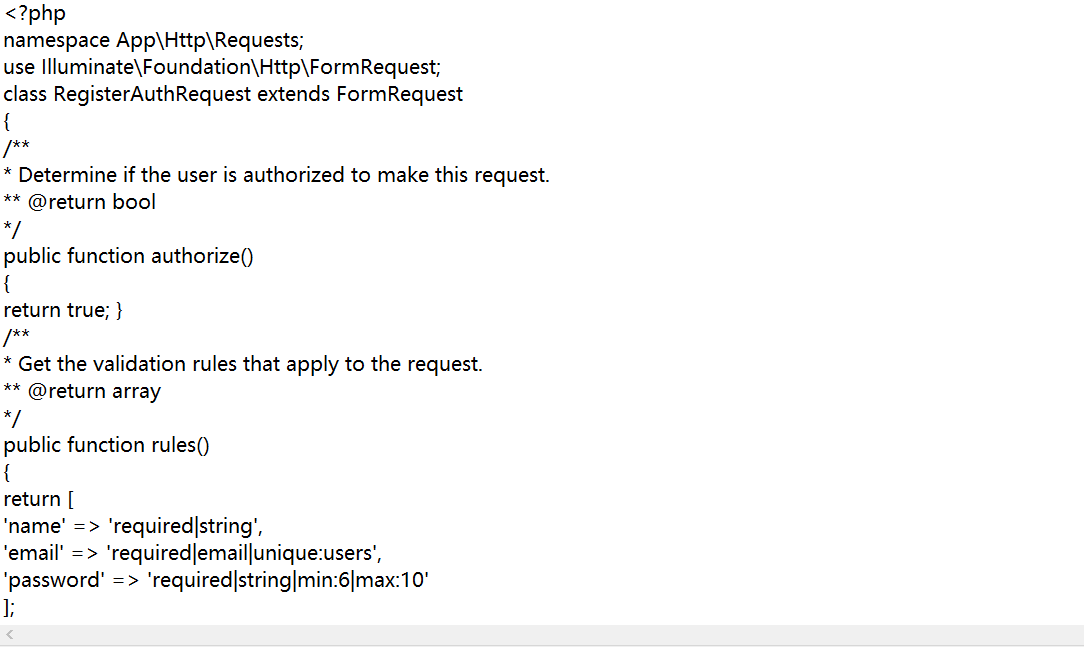


7.创建一个表单请求来验证数据，通过以下命令创建 RegisterAuthRequest 的表单请求

php artisan make:request RegisterAuthRequest

这段代码将在 app/Http/Requests 目录下创建 RegisterAuthRequest.php 文件。修改

这个文件如下



8.运行以下命令创建一个新的 ApiController

php artisan make:controller ApiController

9. 这段代码会在 app/Http/Controllers 目录下创建 ApiController.php 文件。修改以

下代码如下



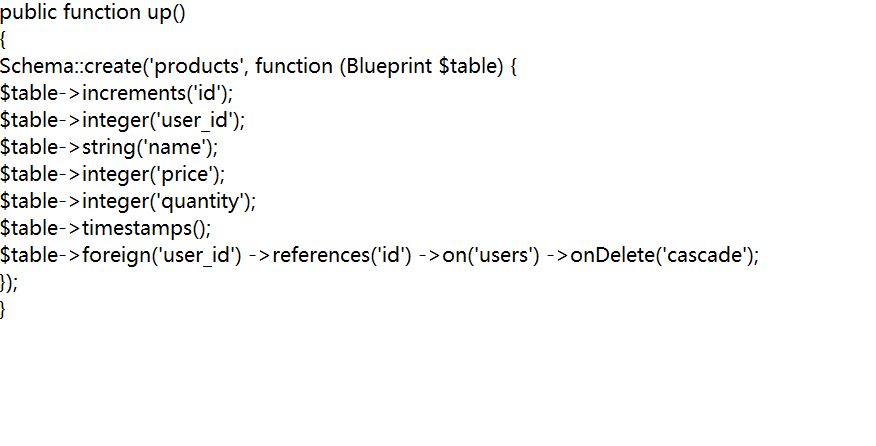
10.创建产品部分，需要 Product 模型，控制器与迁移文件。运行如下命令创建 Product

模型，控制器与迁移文件。

php artisan make:model Product -mc

这段代码会在 database/migrations 目 录 下 创 建 一 个 新 的 数 据 库 迁 移 文 件

create\_products\_table.php，现在更改以下里面的 up 方法。



11.在 Product 模型中添加 fillable 属性。在 app 目录下打开 Product.php 文件并添加属

性。

class Product extends Model

{

protected $fillable = [

'name', 'price', 'quantity'

];

}

12.添加 5 个方法

① index, 为经过身份认证的用户获取所有产品列表

② show, 根据 ID 获取特定的产品

③ store, 将新产品存储到产品列表中

④ update, 根据 ID 更新产品详情

⑤ destroy, 根据 ID 从列表中删除产品

添加一个构造函数来获取经过身份认证的用户，并将其保存在 user 属性中。

protected $user;

public function \_\_construct()

{

$this->user = JWTAuth::parseToken()->authenticate();

}

添加 index 方法

public function index()

{

return $this->user

->products()

->get(['name', 'price', 'quantity'])

->toArray();}

添加 show 方法

public function show($id)

{

$product = $this->user->products()->find($id);

if (!$product) {

return response()->json([

'success' => false,

'message' => 'Sorry, product with id ' . $id . ' cannot be

found'

], 400);

}

return $product;

}

添加 store 方法

public function store(Request $request)

{

$this->validate($request, [

'name' => 'required',

'price' => 'required|integer',

'quantity' => 'required|integer'

]);

$product = new Product();

$product->name = $request->name;

$product->price = $request->price;

$product->quantity = $request->quantity;

if ($this->user->products()->save($product))

return response()->json([

'success' => true,

'product' => $product

]);

else

return response()->json([

'success' => false,

'message' => 'Sorry, product could not be added'

], 500);

}

添加 update 方法

public function update(Request $request, $id){

$product = $this->user->products()->find($id);

if (!$product) {

return response()->json([

'success' => false,

'message' => 'Sorry, product with id ' . $id . ' cannot be

found'

], 400);

}

$updated = $product->fill($request->all())

->save();

if ($updated) {

return response()->json([

'success' => true

]);

} else {

return response()->json([

'success' => false,

'message' => 'Sorry, product could not be updated'

], 500);

}

}

添加 destroy 方法

public function destroy($id)

{

$product = $this->user->products()->find($id);

if (!$product) {

return response()->json([

'success' => false,

'message' => 'Sorry, product with id ' . $id . ' cannot be

found'

], 400);

}

if ($product->delete()) {

return response()->json([

'success' => true

]);

} else {

return response()->json(['success' => false,

'message' => 'Product could not be deleted'

], 500);

}

}

1. **程序设计思路**
2. **程序源代码**
3. **源代码（包含数据库表设计代码）**

9.<?php

namespace App\Http\Controllers;

use App\Http\Requests\RegisterAuthRequest;

use App\User;

use Illuminate\Http\Request;

use JWTAuth;

use Tymon\JWTAuth\Exceptions\JWTException;

class ApiController extends Controller

{

public $loginAfterSignUp = true;public function register(RegisterAuthRequest $request)

{

$user = new User();

$user->name = $request->name;

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

$user->password = bcrypt($request->password);

$user->save();

if ($this->loginAfterSignUp) {

return $this->login($request);

}

return response()->json([

'success' => true,

'data' => $user

], 200);

}

public function login(Request $request)

{

$input = $request->only('email', 'password');

$jwt\_token = null;

if (!$jwt\_token = JWTAuth::attempt($input)) {

return response()->json([

'success' => false,

'message' => 'Invalid Email or Password',

], 401);

}

return response()->json([

'success' => true,

'token' => $jwt\_token,

]);

}

public function logout(Request $request)

{

$this->validate($request, [

'token' => 'required'

]);

try {JWTAuth::invalidate($request->token);

return response()->json([

'success' => true,

'message' => 'User logged out successfully'

]);

} catch (JWTException $exception) {

return response()->json([

'success' => false,

'message' => 'Sorry, the user cannot be logged out'

], 500);

}

}

public function getAuthUser(Request $request)

{

$this->validate($request, [

'token' => 'required'

]);

$user = JWTAuth::authenticate($request->token);

return response()->json(['user' => $user]);

}

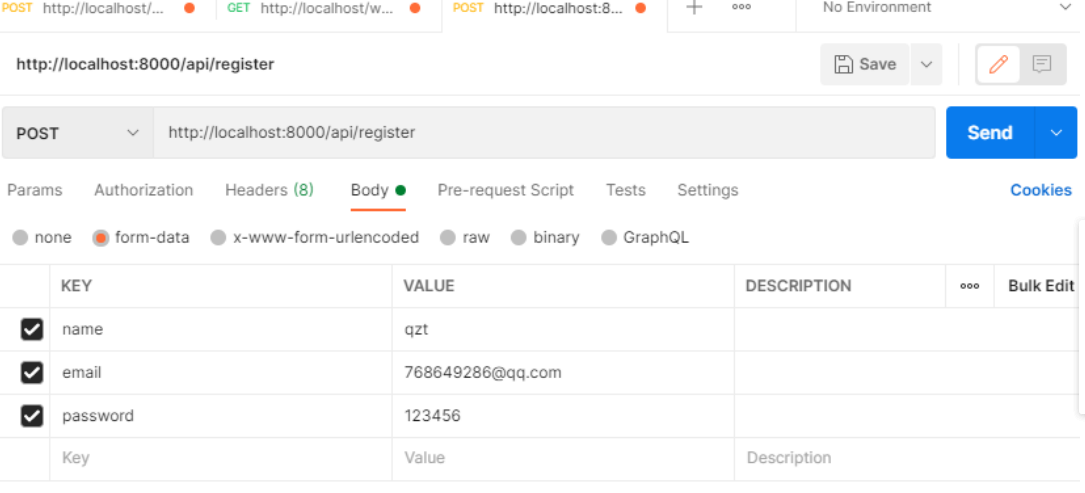
}

1. **程序测试方法及测试结果记录（不能光截图，要有相应的文字说明）**
2. **测试方法**

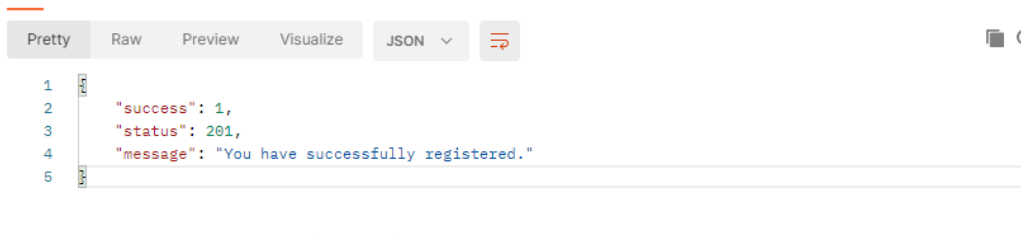
**应用PHP加POSTMAN**

1. **测试流程**

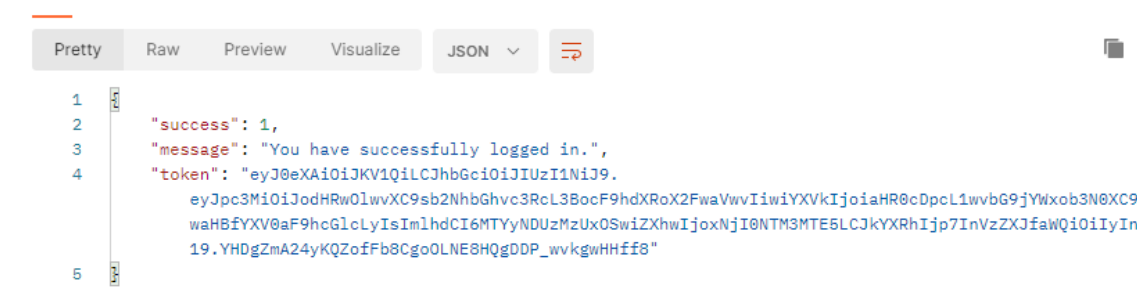
打开 postman 开始测试



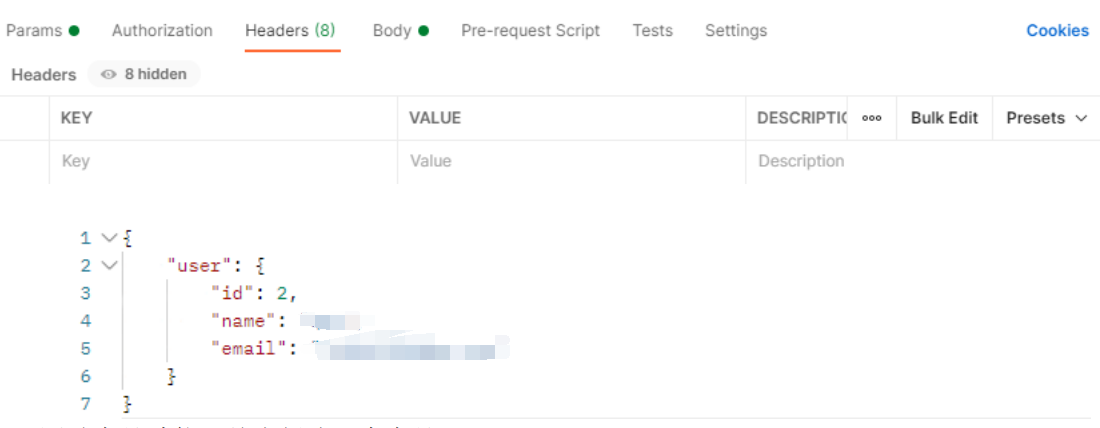
填入姓名，邮件，密码，然后点击 send ，显示注册成功 \



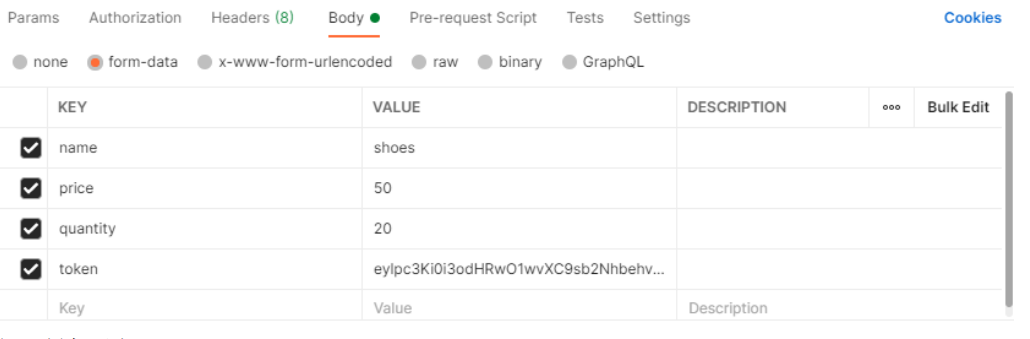
测试登录 login 路由，输入邮箱和密码



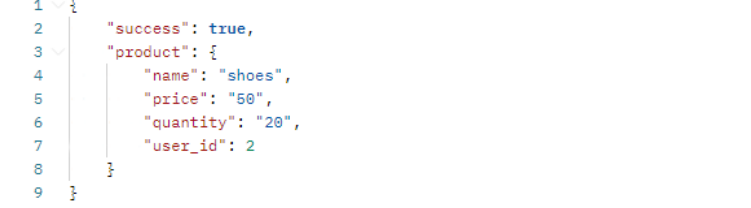
然后获取详细信息



然后测试产品功能



成功



用index 方法获取成功



1. **实验分析总结及心得（该部分也是评分的一个重点）**

（结合所学知识对实验过程中观察到的实验结果进行分析总结，以便加深对知识的理解，并总结通过实验学到的知识或技术）

本实验有点类似实验三的，在做完实验三之后，对实验的前面部分理解更加深入。在本次实验一开始的过程中，尝试按照老师文档中附带的相关链接进行操作，但是在使用Seed随机生成数据的过程中，产生了相关的问题，无法正常继续。在重新深入学习Laravel及相关知识后，发现该步骤是可以略过的。因为前两个要求与实验3的要求一致，因此这里仅将路由配置中的代码更改为Restful API风格即可，再加上登陆后对相关信息修改的操作，同样按照Restful API风格进行。同时也体会到PHP语言是一门在当下确实有用的语言，在暑假的空闲时间内我也会抽出时间继续学习。