

杭州电子科技大学

Web 应用开发 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	19061440
学生姓名	张逸轩
教师姓名	胡伟通
完成日期	
成 绩	

作业 4 RESTful API 的实现

一、实验目的和要求

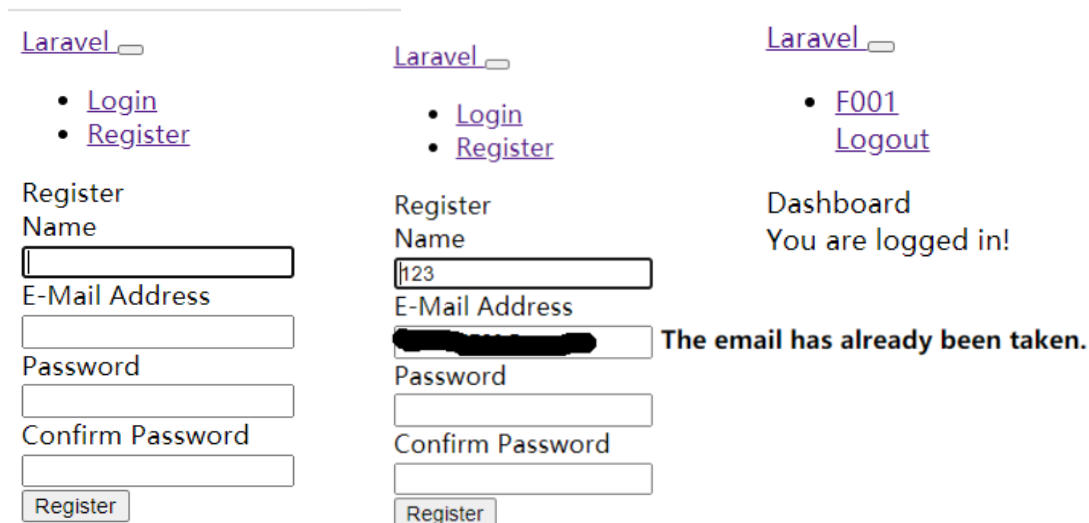
1. 实验目的：基于 Laravel 框架和 RESTful 设计准则，按照前后端分离的开发形式来实现简单的用户注册和登录。
2. 实验要求：
 - （1）实现 Email 形式的注册功能和相应的登录功能，注册部分具备邮件激活功能（使用 Laravel 的邮件发送机制或第三方组件）；
 - （2）实现忘记密码时通过重置密码邮件设置新密码（使用 Laravel 的邮件发送机制或第三方组件）；
 - （3）包含对某个物品（自己选择）的操作，以 RESTful API 风格进行；
 - （4）如果使用 JWT 认证（JSON Web Tokens），后面会酌情加分；
 - （5）不要求提供前端方面的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

注意：要求 1 和要求 2，不能按照作业 3 的方式来做。

二、实验内容

根据实验要求中的每条规定，进行实验。

- （1）实现 Email 形式的注册功能和相应的登录功能
首先需要构建相关的视图，使用 `composer require laravel/ui` 进行。
再通过脚手架命令 `php artisan ui vue --auth` 搭建相应具体的 auth
接下来在 `env` 中进行相关数据库凭证的配置即可完成登录，主要是账号密码。



The image displays three screenshots of a web application interface for user authentication.

- Left Screenshot:** Shows the registration form. It includes a "Register" link, a "Login" link, and a "Register" button. The form fields are: "Name", "E-Mail Address", "Password", and "Confirm Password".
- Middle Screenshot:** Shows the registration form with a message "The email has already been taken." indicating a duplicate email address.
- Right Screenshot:** Shows the dashboard after successful login. It includes a "Logout" link and a message "You are logged in!".

以上三张图片实现了注册及登录的相关情况。

(2) 实现忘记密码时通过重置密码邮件设置新密码

Laravel ☐

- [Login](#)
- [Register](#)

Login

E-Mail Address

Password

☐ Remember Me

[Forgot Your Password?](#)

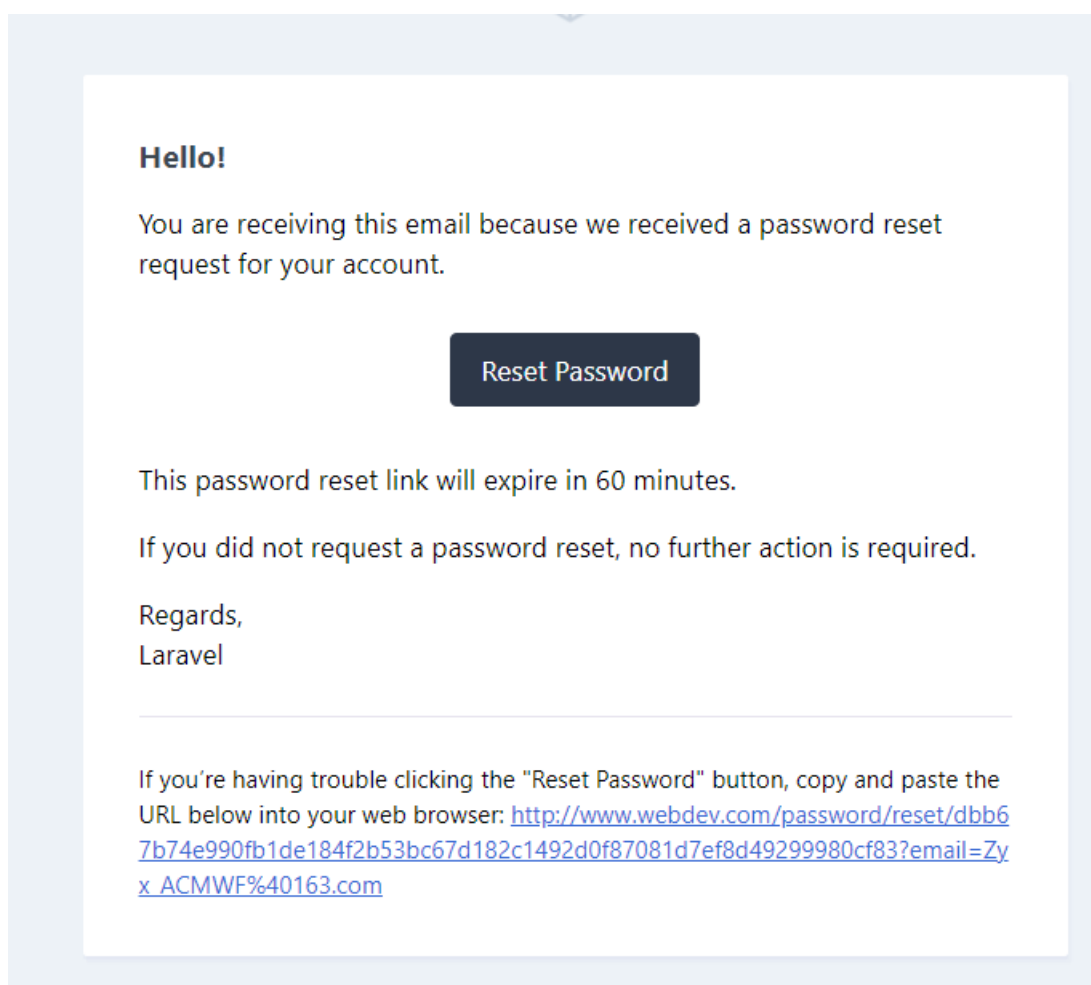
Laravel ☐

- [Login](#)
- [Register](#)

Reset Password

We have emailed your password reset link!

E-Mail Address



以上三张图，前两张为重新设置密码的操作，最后一张为邮箱中收到更改密码的邮件。可见相关操作是正确的。

直接点击 **Forgot Your Password** 即可，并且设置了相关的交互界面。

(3) 包含对某个物品（自己选择）的操作，以 **RESTful API** 风格进行，并且使用了 **JWT 认证**（**JSON Web Tokens**）；

```

D:\phpstudy_pro\WWW\my_laravel>composer create-project --prefer-dist laravel/laravel jwt
Creating a "laravel/laravel" project at "./jwt"
Installing laravel/laravel (v8.5.20)
- Downloading laravel/laravel (v8.5.20)
- Installing laravel/laravel (v8.5.20): Extracting archive
Created project in D:\phpstudy_pro\WWW\my_laravel\jwt
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 105 installs, 0 updates, 0 removals
- Locking asm89/stack-cors (v2.0.3)
- Locking brick/math (0.9.2)
- Locking doctrine/inflector (2.0.3)
- Locking doctrine/instantiator (1.4.0)
- Locking doctrine/lexer (1.2.1)
- Locking dragonmantank/cron-expression (v3.1.0)
- Locking egulias/email-validator (2.1.25)
- Locking facade/flare-client-php (1.8.1)
- Locking facade/ignition (2.10.2)
- Locking facade/ignition-contracts (1.0.2)
- Locking fakerphp/faker (v1.14.1)
- Locking fideloper/proxy (4.4.1)
- Locking filp/whoops (2.13.0)
- Locking fruitcake/laravel-cors (v2.0.4)
- Locking graham-campbell/result-type (v1.0.1)
- Locking guzzlehttp/guzzle (7.3.0)
- Locking guzzlehttp/promises (1.4.1)
- Locking guzzlehttp/psr7 (2.0.0)

```

首先新建一个文件夹，便于区分，防止不同版本不同内容的文件混淆

```

C:\Windows\system32\cmd.exe - composer require tymon/jwt-auth:dev-develop --prefer-source
Use the "composer fund" command to find out more!
> @php artisan key:generate --ansi
Application key set successfully.

D:\phpstudy_pro\WWW\my_laravel>composer require tymon/jwt-auth:dev-develop --prefer-source
./composer.json has been updated
Running composer update tymon/jwt-auth
Loading composer repositories with package information
Updating dependencies
Lock file operations: 4 installs, 0 updates, 0 removals
- Locking lcobucci/jwt (3.3.3)
- Locking namshi/jose (7.2.3)
- Locking symfony/polyfill-php56 (v1.20.0)
- Locking tymon/jwt-auth (dev-develop ab00f2d)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 4 installs, 0 updates, 0 removals
Failed to download namshi/jose from source: git was not found in your PATH, skipping so
Now trying to download from dist

```

接下来配置相关的扩展包，这里使用了 tymon/jwt-auth 的扩展包，这里可以更加方便地使用 JWT。

```

C:\Windows\system32\cmd.exe
Discovered Package: laravel/ui
Discovered Package: nesbot/carbon
Discovered Package: nunomaduro/collision
Discovered Package: tymon/jwt-auth
Package manifest generated successfully.
77 packages you are using are looking for funding.
Use the `composer fund` command to find out more!

D:\phpstudy_pro\WWW\my_laravel>
D:\phpstudy_pro\WWW\my_laravel>php artisan vendor:publish --provider="Tymon\JWTAuth\Providers\LaravelServi
Copied File [\vendor\tymon\jwt-auth\config\config.php] to [\config\jwt.php]
Publishing complete.

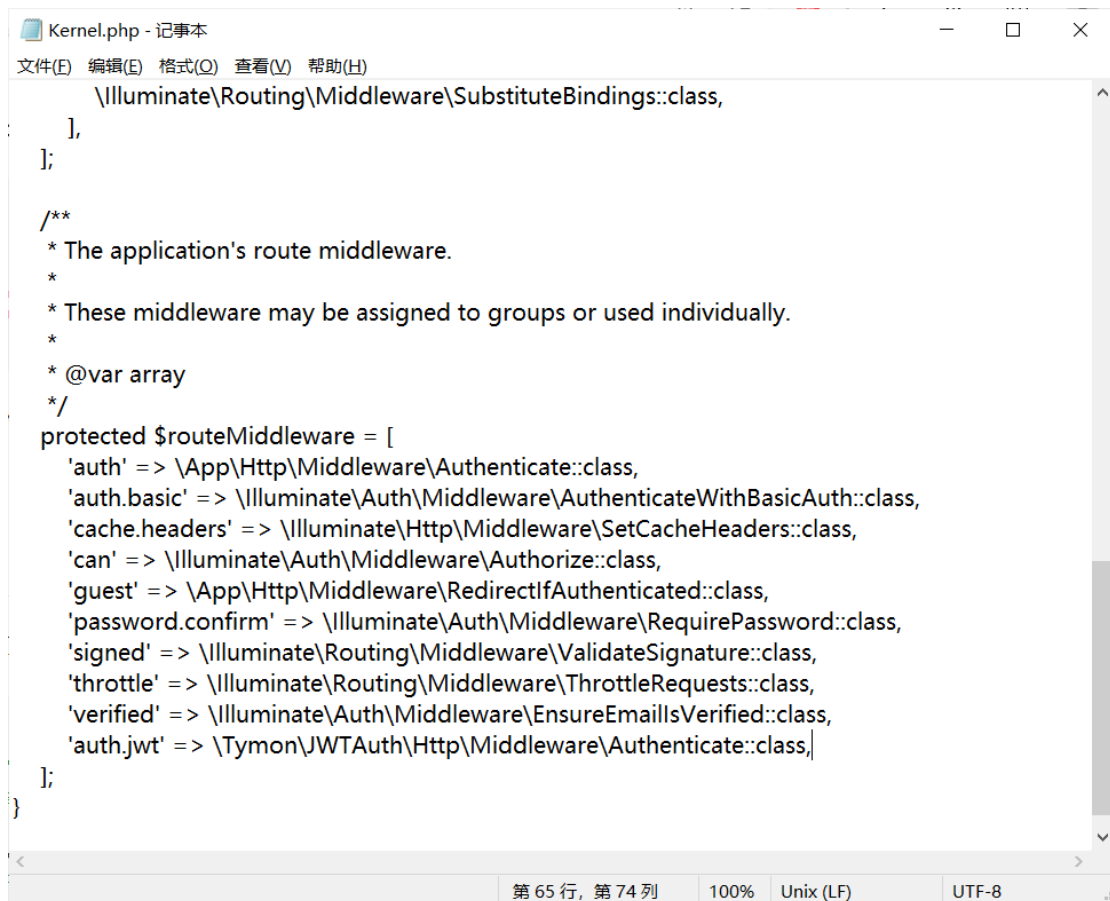
D:\phpstudy_pro\WWW\my_laravel>
D:\phpstudy_pro\WWW\my_laravel>php artisan --version
Laravel Framework 8.46.0

D:\phpstudy_pro\WWW\my_laravel>php artisan vendor:publish --provider="Tymon\JWTAuth\Providers\LaravelServi
Publishing complete.

D:\phpstudy_pro\WWW\my_laravel>
D:\phpstudy_pro\WWW\my_laravel>php artisan jwt:secret
jwt-auth secret [KP7iZJVAn6bnBNxYWC3K6FFp694BdLH2eg7Lt0YAnuQn90s7pTM7ofqf68V1Zws0] set successfully.

```

这里需要生成 JWT 密钥，进行相关地签发，其实相对于其他类型的数据而言，这里体现了 web 开发过程中安全性的考虑。



```

Kernel.php - 记事本
文件(E) 编辑(E) 格式(O) 查看(V) 帮助(H)

    \Illuminate\Routing\Middleware\SubstituteBindings::class,
],
];

/**
 * The application's route middleware.
 *
 * These middleware may be assigned to groups or used individually.
 *
 * @var array
 */
protected $routeMiddleware = [
    'auth' => \App\Http\Middleware\Authenticate::class,
    'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
    'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,
    'can' => \Illuminate\Auth\Middleware\Authorize::class,
    'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
    'password.confirm' => \Illuminate\Auth\Middleware\RequirePassword::class,
    'signed' => \Illuminate\Routing\Middleware\ValidateSignature::class,
    'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,
    'verified' => \Illuminate\Auth\Middleware\EnsureEmailsVerified::class,
    'auth.jwt' => \Tymon\JWTAuth\Http\Middleware\Authenticate::class,
];
}

```

第 65 行, 第 74 列 100% Unix (LF) UTF-8

更改 kernel 文件，实现相关路径的定义，为接下来调用 jwt 时更加方便。

```
*api.php - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

| Here is where you can register API routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| is assigned the "api" middleware group. Enjoy building your API!
|
*/

Route::middleware('auth:api')->get('/user', function (Request $request) {
    return $request->user();
});

Route::post('login', 'ApiController@login');
Route::post('register', 'ApiController@register');

Route::group(['middleware' => 'auth.jwt'], function () {
    Route::get('logout', 'ApiController@logout');

    Route::get('user', 'ApiController@getAuthUser');

    Route::get('products', 'ProductController@index');
    Route::get('products/{id}', 'ProductController@show');
    Route::post('products', 'ProductController@store');
    Route::put('products/{id}', 'ProductController@update');
});
```

修改了相关的路由设置，均在 productcontroller 和 apicontroller 下实现。

```
/**
 * Get the identifier that will be stored in the subject claim of the JWT.
 *
 * @return mixed
 */
public function getJWTIdentifier()
{
    return $this->getKey();
}

/**
 * Return a key value array, containing any custom claims to be added to the JWT.
 *
 * @return array
 */
public function getJWTCustomClaims()
{
    return [];
}
}
```

将 model 下的 user 进行更改，引入了 JWT 的相关方法。

```
D:\phpstudy_pro\WWW\my_laravel\jwt>php artisan make:controller ApiController
Controller created successfully.
```

创建新的 Apicontroller

```
ApiController.php - 记事本
文件(E) 编辑(E) 格式(O) 查看(V) 帮助(H)

{
    $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);
    }
}
```

在 ApiController 中实现 JWT 的相关功能，以上部分就完成了身份验证的部分

```
D:\phpstudy_pro\WWW\my_laravel\jwt>
D:\phpstudy_pro\WWW\my_laravel\jwt>php artisan make:model Product -mc
Model created successfully.
Created Migration: 2021_07_03_100821_create_products_table
Controller created successfully.
```

使用脚手架命令更新版本，并且创建相应的控制器（-mc 功能带有的相关方式）

更改 migration 的 up 函数为

```
public function up()
{
    Schema::create('products', function (Blueprint $table) {
        $table->increments('id');
        $table->integer('user_id');
        $table->string('name');
        $table->integer('price');
        $table->integer('quantity');
        $table->timestamps();
        $table->foreign('user_id')->references('id')->on('users')->onDelete('cascade');
    });
}
```

更改 product 的文件，设置相应的值


```
class Product extends Model
{
    use HasFactory;
    protected $fillable = [
        'name', 'price', 'quantity'
    ];
}
```

下一步需要配置.env 中的数据库凭证，主要是用户名和密码两栏。

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel
DB_USERNAME=root
DB_PASSWORD=root
```

设置好.env 里面的登录密码就能 migration 了，完成后，在 user.php 里加入对 products 的相关设置。

```
public function products()
{
    return $this->hasMany(Product::class);
}
}
```

 ProductController.php - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
<?php
```

```
namespace App\Http\Controllers;
```

```
use Illuminate\Http\Request;
use App\Product;
use Illuminate\Http\Request;
use JWTAuth;
```


之前的准备工作都已经完成了，接下来需要对 `productcontroller` 进行修改即可。首先需要增添头文件：

```
<?php

namespace App\Http\Controllers;

use App\Product;
use Illuminate\Http\Request;
use JWTAuth;

class ProductController extends Controller
{
    //
}
```

头文件添加完成后，以 Restful API 的格式实现五个功能，分别为：

- `index`, 为经过身份认证的用户获取所有产品列表
- `show`, 根据 ID 获取特定的产品
- `store`, 将新产品存储到产品列表中
- `update`, 根据 ID 更新产品详情
- `destroy`, 根据 ID 从列表中删除产品

三个 API 均有参数，另外两个没有参数。可以看到，API 部分没有 `edit view` 和 `create view` 两个方法，这与我们之前所提到的七个方法对资源进行操作是吻合的。

```
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);
    }
}
}
```

以上的文件就实现了五个方法

提示:

1. 可以参考网络资料, 但**不能照搬照抄**。
2. 作业报告的格式尽量整齐, 字体要统一, 内容要层次分明。

三、 程序设计思路

本次实验的一开始过程中, 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。

Restful 风格的要点是通过资源的角度看待问题, 实现起来的格式主要通过资源*7 种操作的方式进行, 分别是 crud (增删查改)+list+create+edit。由于存在相关共用的情况, 因此还需要加上 http 的方法用以区分, 即 GET POST PUT

DELETE。按照上述要求，以对 blog 操作为例，URL 补全后就是：

Create: POST blogs
List: GET blogs
Delete: DELETE blogs/:id
Update: PUT blogs/:id
Read: GET blogs/:id
Create: GET blogs/create
Edit: GET blogs/:id/edit

了解了上述知识即思路后，我们就可以开始编写自己的程序了。

四、 程序源代码

注意源代码要有详细的注释。同学们提交的每个程序都应该遵循 **Honor Code**(诚实代码保证) 的要求。

请大家特别注意一定要在每个程序首部的注释中加上以下保证：

```
# 我真诚地保证：
# 我自己独立地完成了整个程序从分析、设计到编码的所有工作。
# 如果在上述过程中，我遇到了什么困难而求教于人，那么，我将在程序实习报告中
# 详细地列举我所遇到的问题，以及别人给我的提示。
# 在此，我感谢 XXX, ...,XXX 对我的启发和帮助。下面的报告中，我还会具体地提到
# 他们在各个方法对我的帮助。
# 我的程序里中凡是引用到其他程序或文档之处，
# 例如教材、课堂笔记、网上的源代码以及其他参考书上的代码段，
# 我都已经在程序的注释里很清楚地注明了引用的出处。

# 我从未没抄袭过别人的程序，也没有盗用别人的程序，
# 不管是修改式的抄袭还是原封不动的抄袭。
# 我编写这个程序，从来没有想过要去破坏或妨碍其他计算机系统的正常运转。
# 张逸轩
```

1. 源代码（包含数据库表设计代码）

此处不进行复制粘贴，将会放在附件的压缩包中一并上传

注意：

- (1) 可以在此处贴关键代码，也可以放在压缩包中跟本文档一起上传。
- (2) 如果是压缩项目工程代码，建议把 vendor 目录排除掉，以减小压缩包大小。

五、 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）

1. 测试方法

使用 Postman 进行 Restful API 的测试。在网页中进行相应的操作。

2. 测试流程

使用 `php artisan serve` 命令，在命令行中监听 `localhost:8000`

```
[Sat Jul 3 18:29:24 2021] 127.0.0.1:5109 Accepted
[Sat Jul 3 18:29:24 2021] 127.0.0.1:4223 Closing
[Sat Jul 3 18:29:24 2021] 127.0.0.1:1274 Accepted
[Sat Jul 3 18:29:24 2021] 127.0.0.1:5109 [200]: GET /favicon.ico
```

因为在代码的编写中，有写当传送成功时会返回一个 200 的正确提示码，因此可以认为我们的编写正确，正常地返回了相关参数

http://localhost:8000/ + ... No Environment 👁 ⚙

▶ http://localhost:8000/api/register Examples (0) ▼

POST ▼ http://localhost:8000/api/register Params Send ▼ Save ▼

Authorization Headers Body Pre-request Script Tests Cookies Code

☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary

Key	Value
<input checked="" type="checkbox"/> name	
<input checked="" type="checkbox"/> email	
<input checked="" type="checkbox"/> password	123456789
New key	Value

通过 register 发送请求，获得 token 令牌。

Body

Cookies

Headers (9)

Test Results

Status: 200 OKTime: 1660 msSize: 627 B

PrettyRawPreviewJSON

Save Response

```
1 {  
2   "success": true,  
3   "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9  
    .eyJpc3MiOiJodHRwOiwwXC9sb2NhbGhvczQ6ODAwMFwwYXBXPC9CY9Zwdpc3RlciiIsImhldCI6MTUyNykyOTg4Nywib2ZhbmIjoxNTI0M2MtNDg  
    3LCluYmYoIE1mYy5Mjk4ODcsImp0aSI6Im3qNU8zQWdmYjZSaUxvckk4MlRCjZdwIoioisInYnBydiEiOjg3ZTBhZjZlZjZlMDZDE0DEYmZmRlYzk  
    3MTUzYTE0ZTBiMDQ3NTQZYWEifQ.xhvCe-x9rreGi7v2ngv-eyutu-QlC4lCAwFo07v9p7u"  
4 }
```

通过发送另一个请求检测 login 路由，返回 200 和令牌

http://localhost:8000/

+ ...

No Environment

Examples (0)

▶ http://localhost:8000/api/login

POST http://localhost:8000/api/login Params Send Save

Authorization Headers Body Pre-request Script Tests Cookies Code

☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary

Key	Value	... Bulk Edit
<input checked="" type="checkbox"/> email	example@example.com	
<input checked="" type="checkbox"/> password	123456789	
New key	Value	

Status: 200 OK Time: 782 ms Size: 623 B

Body Cookies Headers (9) Test Results Pretty Raw Preview JSON Save Response

```
1 {  
2   "success": true,  
3   "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.  
           .eyJ3c3MiOiJodHRwOiUvXC93sb2NhbgVhc3Q6ODAwMFVxYXBkC9sb2dpbiIsImhhbmCI6MTUyNjksImDE5NywiZmxhbjI0MzNmZmYyYmY1OjE1MjY5MScAxOTcsImp0aSI6MDY1VGRGpmU1RWMUpURFElLCJzdWI6IjoiaSInbDyddiI6Ijg3ZTBhZjFlZjlmZDE0DEYzMRIyZk3MTUzYTE0ZTBIMDQ3NTQ2YWEifQ.RYKHUC4QuOkwJw6K7e2oD7jbWfQSC6dg9D_-56fyd7E"  
4 }
```

获取用户的详细信息:

The screenshot shows a REST client interface with the following details:

- URL: `http://localhost:8000/api/user`
- Method: `GET`
- Params: `http://localhost:8000/api/user?token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRw...`
- Headers: Empty
- Status: `200 OK`, Time: `820 ms`, Size: `409 B`
- Response Body (JSON):

```
1 {
2   "user": {
3     "id": 1,
4     "name": "John Doe",
5     "email": "john.doe@example.com",
6     "created_at": "2023-01-01T00:00:00Z",
7     "updated_at": "2023-01-01T00:00:00Z"
8   }
9 }
```

以上为身份的相关测试，接下来测试产品部分，首先我们创建一个产品。

The screenshot shows a REST client interface with the following details:

- URL: `http://localhost:8000/api/products`
- Method: `POST`
- Params: Empty
- Body: `form-data` (selected), with fields: `name`, `price`, `quantity` (value: `5`), and `token` (value: `eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwOlwvXC9yb2NhbG93b3R5bGUuYm9vaWw...`)
- Status: `200 OK`, Time: `943 ms`, Size: `440 B`
- Response Body (JSON):

```
1 {
2   "success": true,
3   "product": {
4     "name": "Product Name",
5     "price": 10,
6     "quantity": 5,
7     "user_id": 1,
8     "updated_at": "2023-01-01T00:00:00Z",
9     "created_at": "2023-01-01T00:00:00Z",
10    "id": 3
11  }
12 }
```

接下来通过 `index` 的方法获取我们的产品

The screenshot shows a REST client interface with the following details:

- URL: `http://localhost:8000/api/products`
- Method: `GET`
- Params: `http://localhost:8000/api/products?token=eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwOlwvXC9yb2NhbG93b3R5bGUuYm9vaWw...`
- Headers: Empty
- Status: `200 OK`, Time: `683 ms`, Size: `328 B`
- Response Body (JSON):

```
1 [
2   {
3     "name": "Product Name",
4     "price": 10,
5     "quantity": 5
6   }
7 ]
```

可见，相关的路由均可正常工作，本次实验成功。

六、 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）

特别推荐学生写出做实验遇到的问题，以及从原理分析得到解决方案的过程。

在一开始的分析完成后，由于与实验三的需求存在重叠，因此原想法是在实验三的基础上进行一些代码的重构，例如尝试将原有的 article 文件按原有步骤命名为 articlal 文件进行区分尝试做重构，但是最终发现相关代码逻辑十分混乱，难以继续进行，因此重新新建了项目的目录，在 jwt 文件夹下继续进行相关的尝试。这启示我们在进行开发的过程中，需要有良好的项目书写规范，该新建就新建，不要依靠沉没成本做决策。

ArticalController.php - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
* Display the specified resource.
*
* @param int $id
* @return \Illuminate\Http\Response
*/
public function show($id)
{
    $susser=\app\Models\User::findOrFail($id);
    return view('home',['email'=>$susser]);
    //
}
```

Articalcontroller 与 homecontroller 乱作一团，相互引用，增添了很多不必要的麻烦。

HomeController.php - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

```
class HomeController extends Controller
{
    /**
     * Create a new controller instance.
     *
     * @return void
     */
    public function __construct()
    {
        $this->middleware('auth');
    }

    /**
     * Show the application dashboard.
     *
     * @return \Illuminate\Contracts\Support\Renderable
     */
    public function index()
    {
        return view('home');
    }
}
```

Auth

123.php

ArticalController.php

ArticleController.php

Controller.php

HomeController.php

多个版本放在一起的文件夹，也显得很凌乱。

另外，之前在学习的过程中，一直没有对 Laravel 所谓的 MVC 框架有一个更好的了解，单纯通过修改单个文件，感觉非常麻烦。在这次实验完成之后，我会尝试使用 PHPstorm 等更多更好地工具提升 web 开发的工作效率。在学习资源的寻找上，这次实验相对于实验三而言更加困难，因此需要提升我们的“搜商”，更好地解决相关问题。本次实验用时相对较长，有很大一部分原因是找不到合适的学习资料所导致的。但通过这次实验，也提高了我对于 MVC 框架的基本理解，提高了对 Web 的相关兴趣，在暑假中也同样会继续深入学习，希望未来能够取得更好地成绩。

七、 参考文献

(1)<https://zhuanlan.zhihu.com/p/97978097>

- (2)<https://learnku.com/articles/19172>
- (3)<https://laravelacademy.org/post/6171.html>
- (4)<https://laravelacademy.org/post/9153.html>
- (5)<https://www.extendfeature.com/laravel-8-target-class-seeder-does-not-exist/>
- (6)<https://www.cnblogs.com/hhyl/p/13488424.html>
- (7)<https://www.bilibili.com/video/BV1Vz4y1y7Do?>
- (8)https://blog.csdn.net/qq_30682027/article/details/78225074
- (9)<https://laravelacademy.org/post/9283>
- (10)<https://learnku.com/docs/laravel/5.5>
- (11)<https://segmentfault.com/a/1190000013474769>
- (12)https://blog.csdn.net/createn_1/article/details/81738665
- (13)<https://www.php.cn/php-weizijiaocheng-403288.html>