****

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** | 网工二班 |
| **学 号** | 18041618 |
| **学生姓名** | 廖越强 |
| **教师姓名** | 胡伟通 |
| **完成日期** | 2020.6.17 |
| **成 绩** |  |

**作业5 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。

注意：要求1和要求2，不能按照作业4的方式来做。

1. **实验内容**

根据**实验要求中的每条规定**，进行实验。

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。

1. **程序设计思路**

要求（1）实现思路：添加路由/register，/login和ActiveAccount，并便写相应的控制器，分别用于注册，登陆和激活账号。在register控制器里使用laravel提供的Mail类来发送邮件，邮件中附带账号激活需要的验证码。

要求（2）实现思路：添加路由/forgotPassword和/resetPassword并编写控制器，忘记密码时，通过访问接口/forgotPassword来获取修改密码所需要的验证码，从邮件中取出验证码，将验证码作为参数之一发送给接口/resetPassword，即可重置密码。

要求（3）实现思路：考虑采用laravel提供的resource路由，然后在相应的控制器下编写index(),show(),destroy(),store(),update()方法即可实现。

要求（4）实验思路：根据实验所提供的资料，引入tymon/jwt-auth模块，进行相应的配置即可实现。

1. **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。**

**# <廖越强> 改成自己的名字**

1. **源代码（包含数据库表设计代码）**

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。

整个项目的源代码以附件的形式上传了，这里就放一些主要的代码

1.注册登录、激活验证、重置密码等的实现如下

    public function register(Request $request)

    {

        $data\_check = Validator::make($request->all(), [

            'name' => 'required',

            'email' => 'required|email|unique:users',

            'password' => 'required',

        ]);

        if($data\_check->fails()){

            return response()->json([

                'msg' => 'Invalid data, or the email has been registered',

                'error' =>  $data\_check->errors(),

            ], 406);

        }

        $user = new User();

        $user->name = $request->name;

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

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

        $user->auth\_code = \Str::random(24);

        $user->save();

        $this->sendMail($user);

        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',

            ], 406);

        }

        // var\_dump($input) ;

        $user = User::where(['email' => $request->email])->first();

        // echo $request->email;

        // var\_dump($user) ;

        if (!$user->available) {

            $this->sendMail($user);

            return response()->json([

                'message' => 'You need active your account first and we have sent you the Verify code. Please login again use verify\_token and value with the POST method',

            ], 400);

        }

        return response()->json([

            'success' => true,

            'token' => $jwt\_token,

        ]);

    }

    private function sendMail($user)

    {

        echo $user->email;

        Mail::raw('Welcome to webClass2020, and if this is your email please remember the Code "' . $user->auth\_code.'"And if you have not known about the website, ignore it.', function ($message) use ($user) {

            $message->to($user->email, $user->name)->subject('Welcome to webClass2020!');

        });

    }

    public function ActiveAccount(Request $request)

    {

        $auth\_code\_check = Validator::make($request->all(), [

            'auth\_code' => 'required',

        ]);

        if ($auth\_code\_check->fails()) {

            return response()->json(['message' => 'no auth\_code',], 400);

        }

        if(!(User::where(['auth\_code' => $request->auth\_code])->exists())){

            return response()->json(['message' => 'Invalid auth\_code',], 400);

        }

        User::where(['auth\_code' => $request->auth\_code])->update(['available' => true]);

        $user = User::find(3);

        return response()->json(['message' => 'Successfully activate!',], 200);

    }

    public function logout(Request $request)

    {

        $token\_check = Validator::make($request->all(), [

            'token' => 'required',

        ]);

        if($token\_check->fails()){

            return response()->json(['message'=> 'no token',], 400);

        }

        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'

            ], 400);

        }

    }

    public function forgotPassword(Request $request)

    {

        $email\_check = Validator::make($request->all(), [

            'email' => 'required|email|max:40',

        ]);

        if ($email\_check->fails()) {

            return response()->json(['message'=> 'Invalid email',], 400);

        }

        $email = $request->email;

        $user = User::where(['email' => $email])->first();

        echo $email;

        if (is\_null($user)) {

            return response()->json(['message'=> 'Email not registered',], 404);

        }

        $reset\_password\_code = \Str::random(24);

        ForgotPassword::updateOrCreate([

            'email' => $email,

            'reset\_password\_code' => $reset\_password\_code,

            'available' => false,

        ]);

        Mail::raw('You have forgot your password in WEB class, and if this is your email please remember the Code "' . $reset\_password\_code.'"And if you have not known about the website, ignore it.', function ($message) use ($user) {

            $message->to($user->email, $user->name)->subject('Reset your password!');

        });

        return response()->json(['Send success , and check your email to receive the Message. ']);

    }

    public function resetPassword(Request $request)

    {

        $data\_check = Validator::make($request->all(), [

            'reset\_password\_code' => 'required',

            'password' => 'required|min:8|max:40',

        ]);

        if ($data\_check->fails()) {

            return response()->json(['message'=> 'Invalid data',], 400);

        }

        $reset\_password\_code = $request->reset\_password\_code;

        $forgotPassword = new ForgotPassword();

        $forgotPassword = $forgotPassword->where(['reset\_password\_code' => $reset\_password\_code])->first();

        if (is\_null($forgotPassword) || $forgotPassword->available) {

            return response()->json(['The reset\_password\_code has been used.',], 403);

        }

        $forgotPassword->available = true;

        $forgotPassword->save();

        $user = new User();

        $user->where(['email' => $forgotPassword->email])->update(['password' => bcrypt($request->password)]);

        return response()->json(['Reset password successfully!',]);

    }

2.对用户物品的操作，增删改查等的实现

    public function index()

    {

        $product = new Product();

        return $product->all();

    }

    public function show($id)

    {

        $product = new Product();

        $product = $product->where(['id' => $id])->first();

        if (is\_null($product)) {

            return response()->json([

                'success' => false,

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

            ], 400);

        }

        return $product;

    }

    public function store(Request $request)

    {

        $data\_check = Validator::make($request->all(), [

            'name' => 'required',

            'price' => 'required|integer',

            'quantity' => 'required|integer'

        ]);

        if ($data\_check->fails()) {

            return response()->json(['msg' => $data\_check->errors()], 406);

        }

        $product = new Product();

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

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

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

        $product->save();

        return response()->json([

            'success' => true,

            'product' => $product

        ]);

    }

    public function destroy($id)

    {

        $product = new Product();

        $product = $product->where(['id' => $id])->first();

        if (is\_null($product)) {

            return response()->json([

                'success' => false,

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

            ], 400);

        }

       $product->delete();

        return response()->json([

            'message' => 'delete successfully'

        ]);

    }

    public function update(Request $request, $id)

    {

        // 参数验证

        $data\_check = Validator::make($request->all(), [

            'price' => 'integer',

            'quantity' => 'integer',

        ]);

        if ($data\_check->fails()) {

            return response()->json(['msg' => $data\_check->errors()], 406);

        }

        // echo $request->price;

        // echo $request->quantity;

        // var\_dump($request->all());

        // 查找物品

        $product = new Product();

        $product = $product->where(['id' => $id])->first();

        if (is\_null($product)) {

            return response()->json([

                'success' => false,

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

            ], 400);

        }

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

        // var\_dump($product) ;

        // echo $request->price;

        if ($updated) {

            return response()->json([

                'message' => 'updated successfully'

            ]);

        } else {

            return response()->json([

                'success' => false,

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

            ], 500);

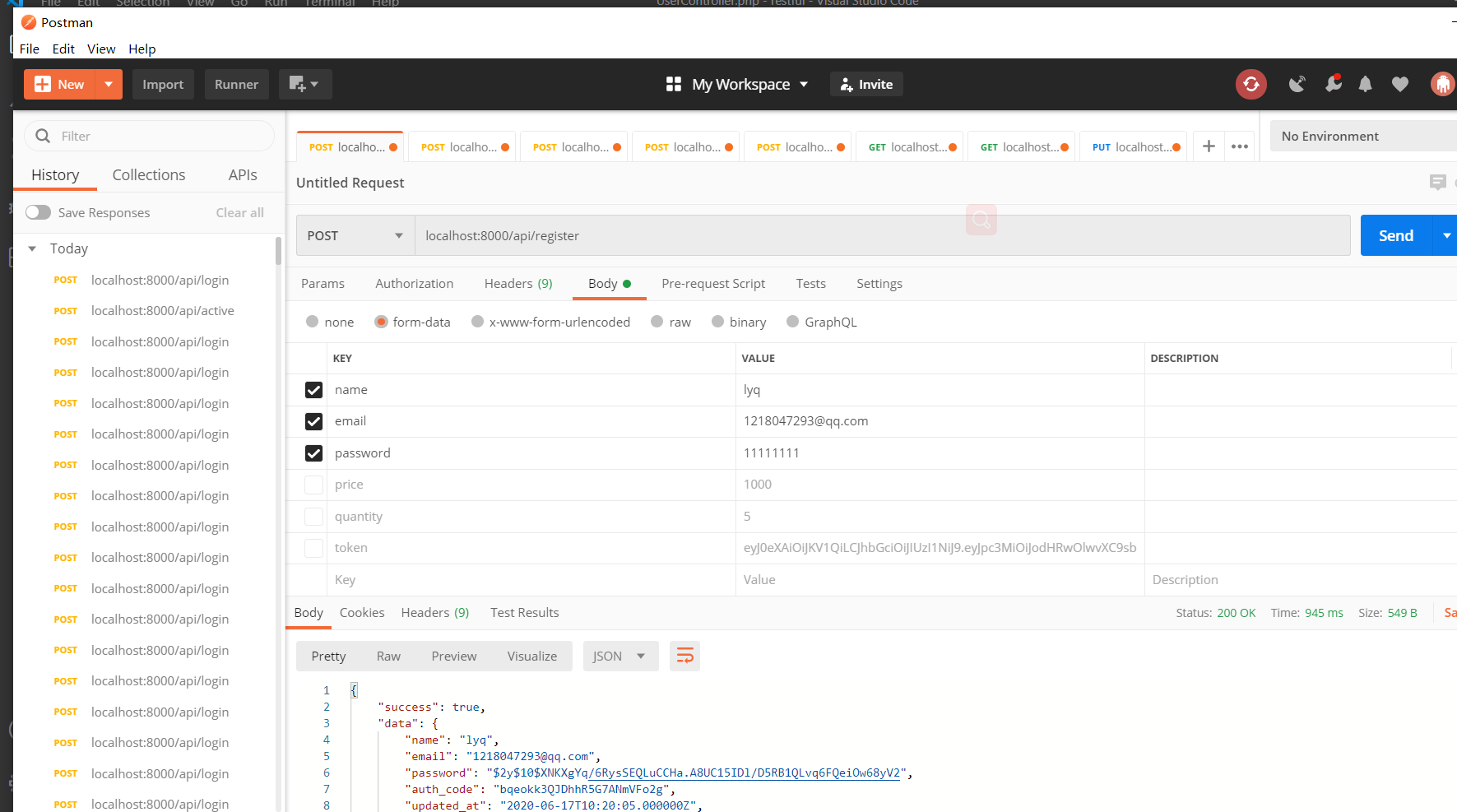
        }

    }

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**

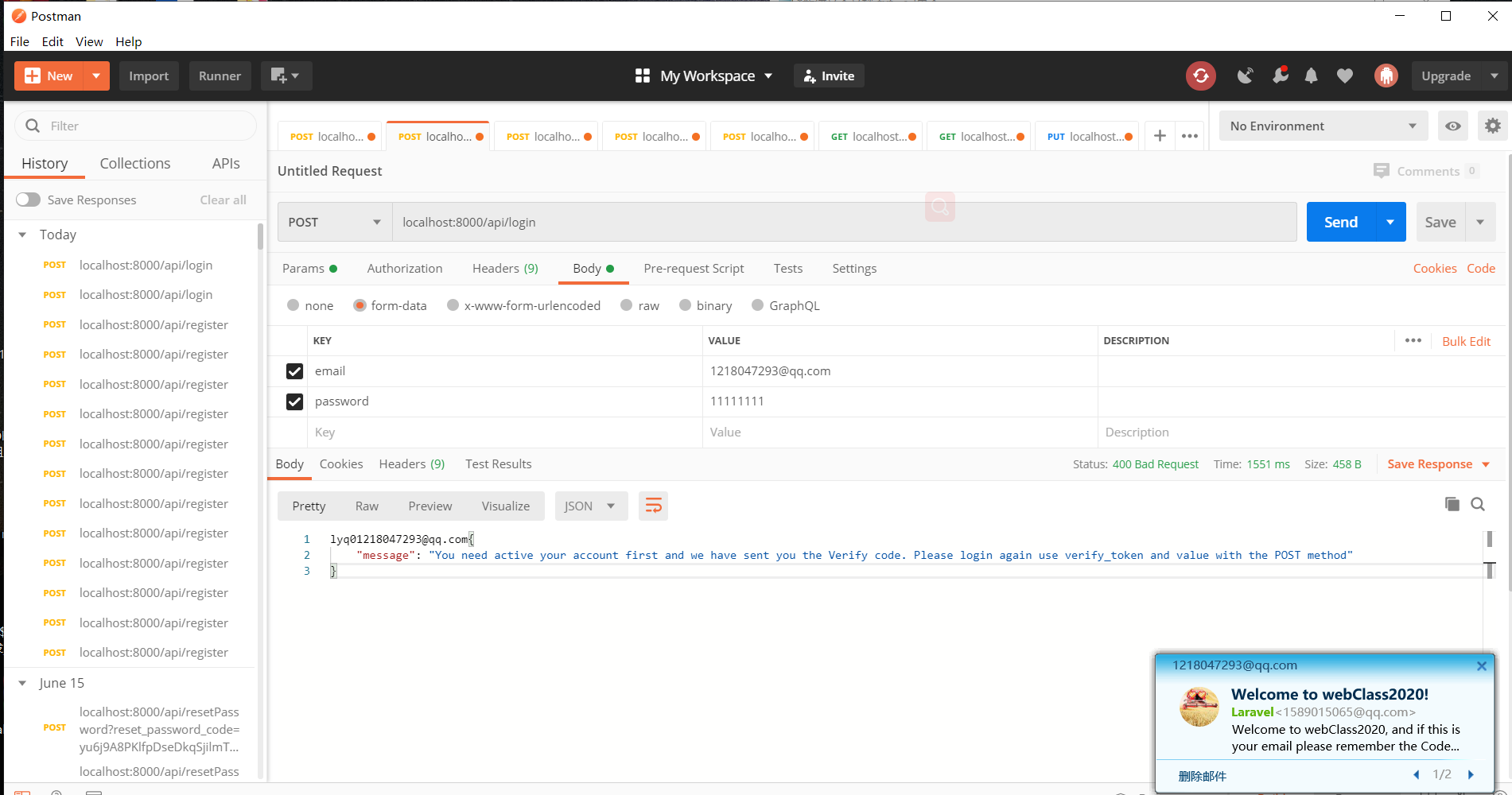
由于没有写前端页面，所以采用postman进行各个功能接口的功能测试，按以下步骤进行

1. 注册测试（/register）
2. 登陆前的邮箱验证测试（/login，/active）
3. 登陆测试（/login）
4. 对物品的增删改查测试(/product/)
5. 修改密码测试(forgetPassword, /resetPassword)
6. 登出测试(/logout)
7. **测试流程**
8. 注册测试（/register）

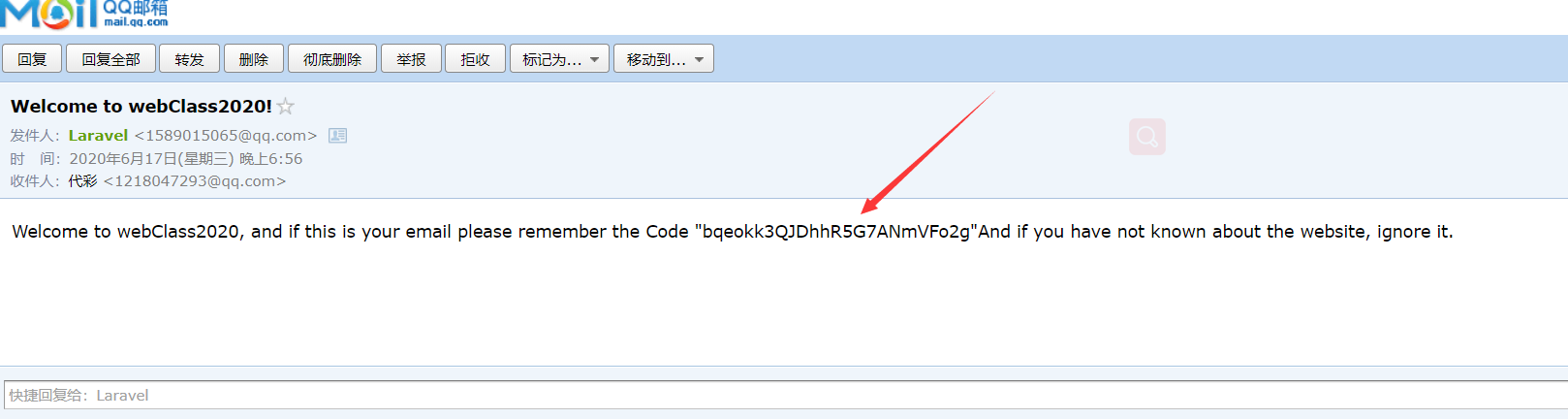


（2）登陆前的邮箱验证测试（/login, /active）

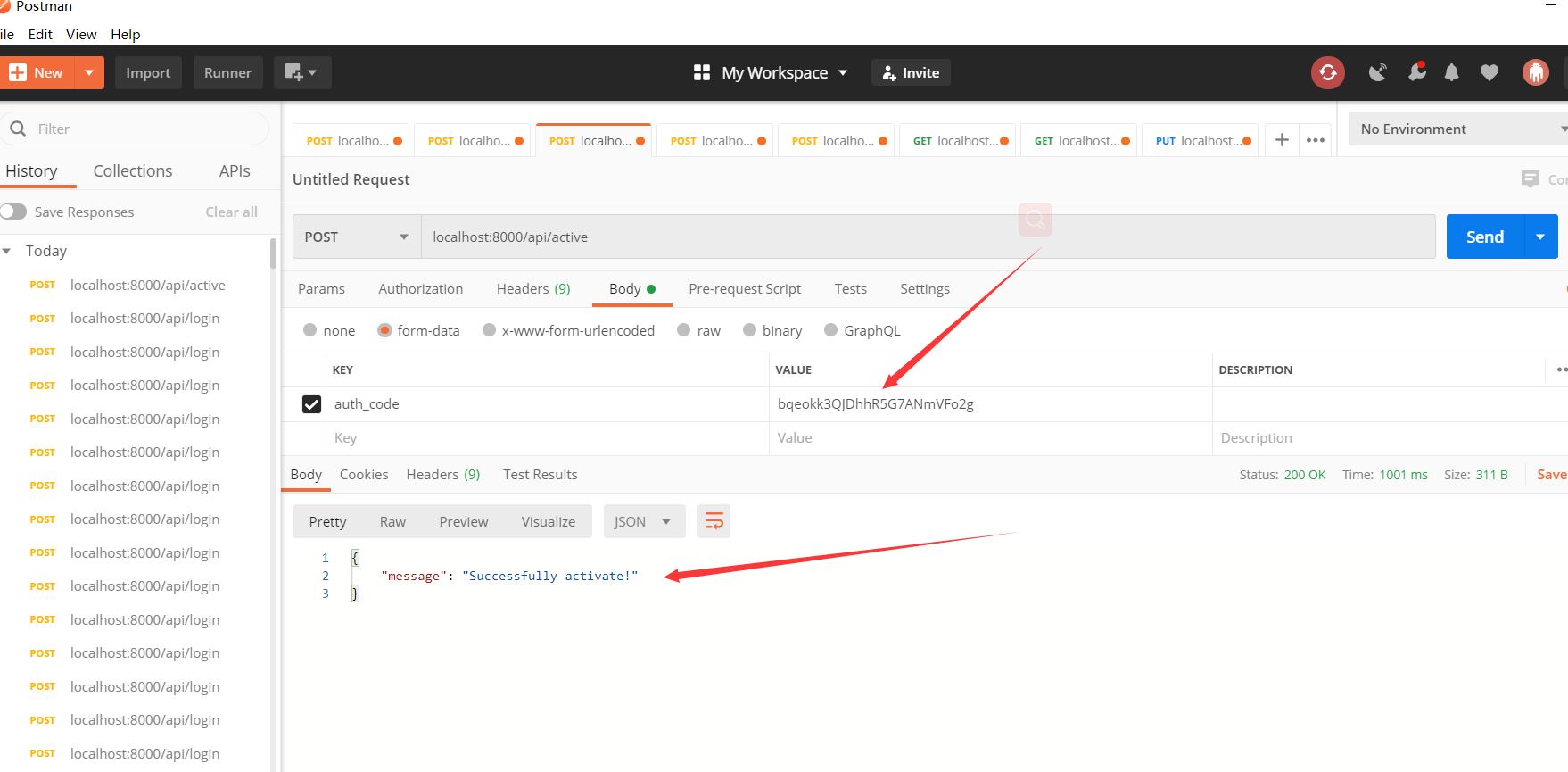
注册好后先访问登陆接口，可以看到访问后，提示要先进行邮箱验证，将邮件中的验证码附加到参数中，发送给/active激活账号接口



验证码如下

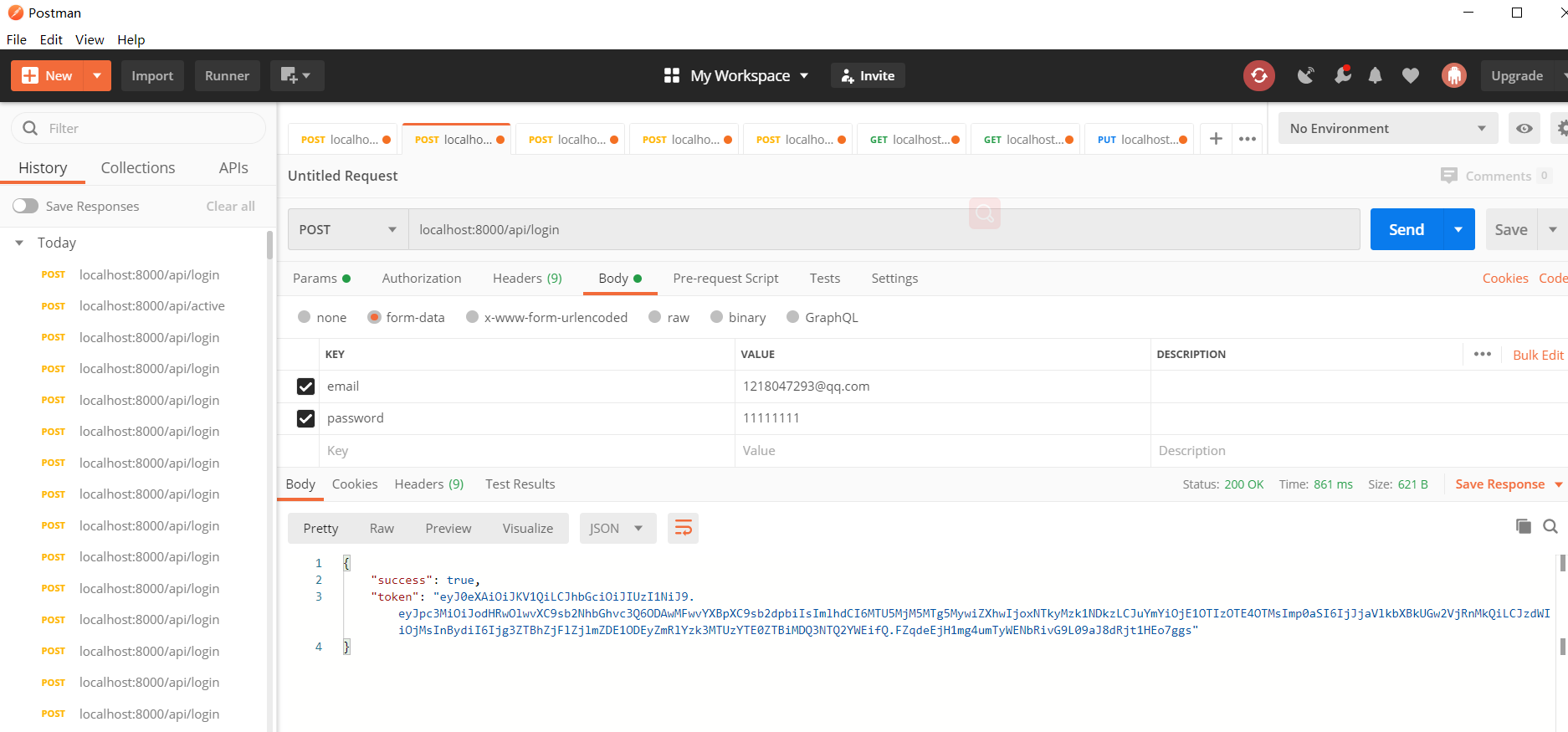


进行账号激活，成功激活如下



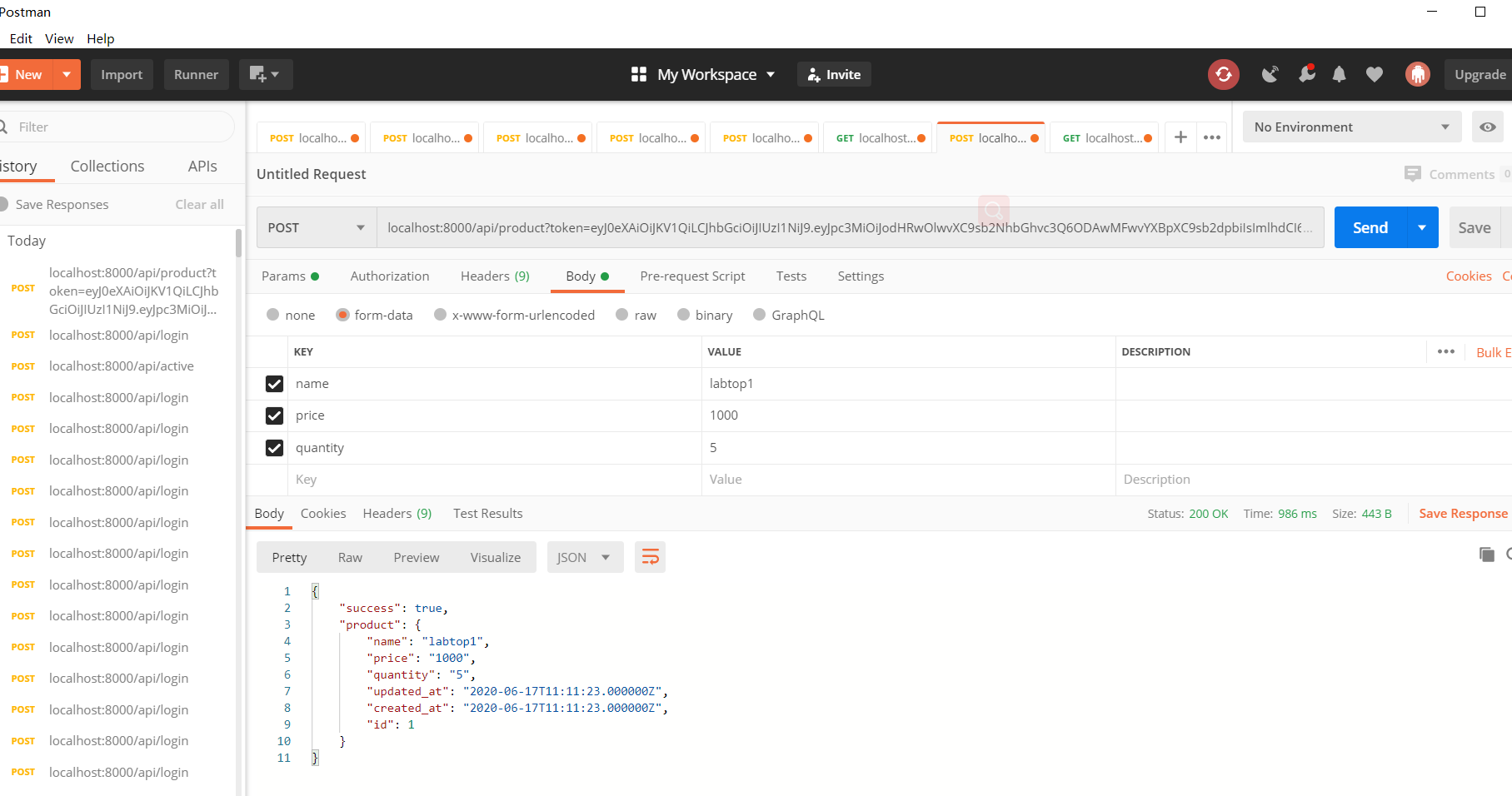
（3）登陆测试

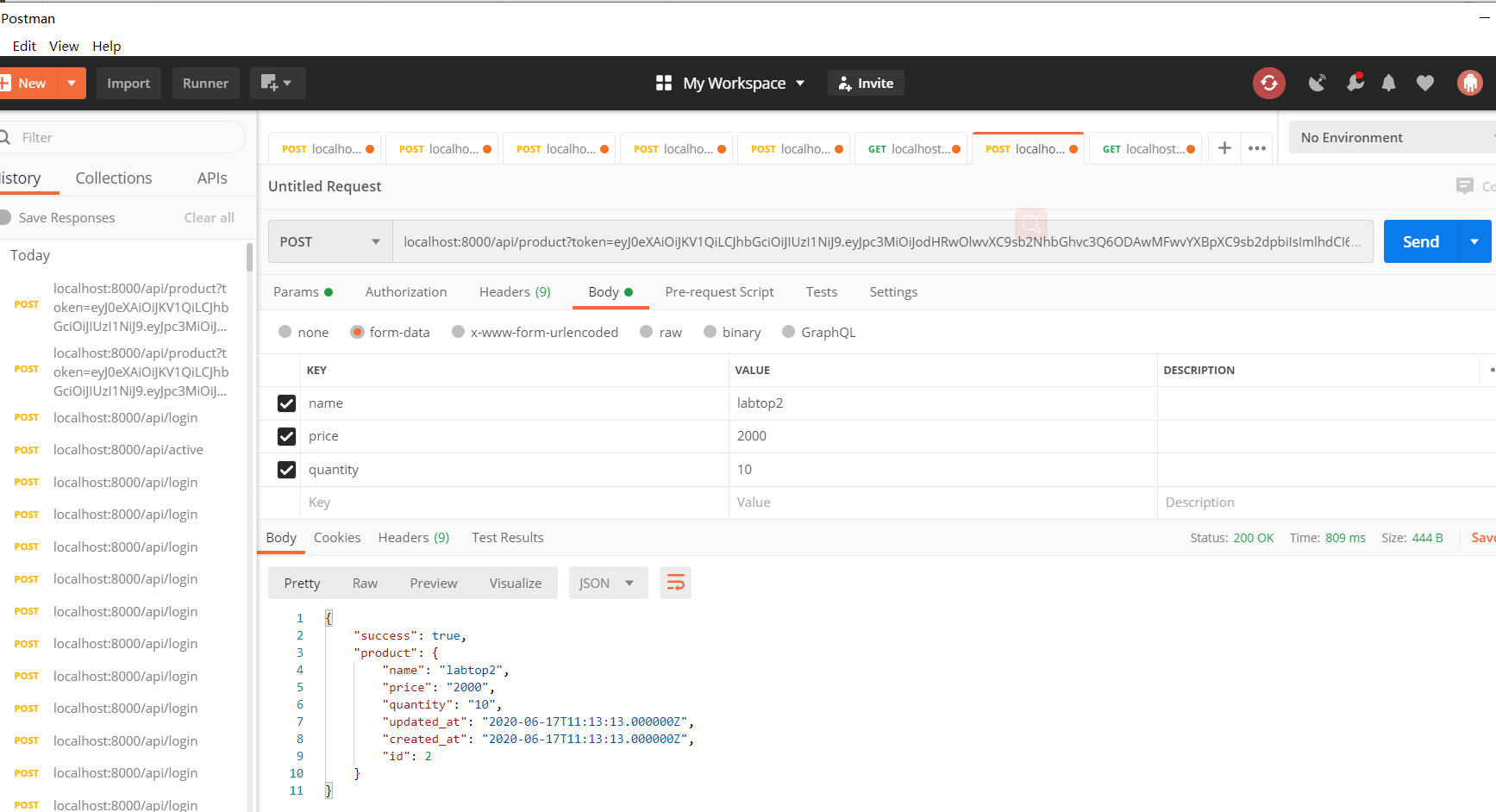
可以看到账号已经成功激活，可以正常登陆了



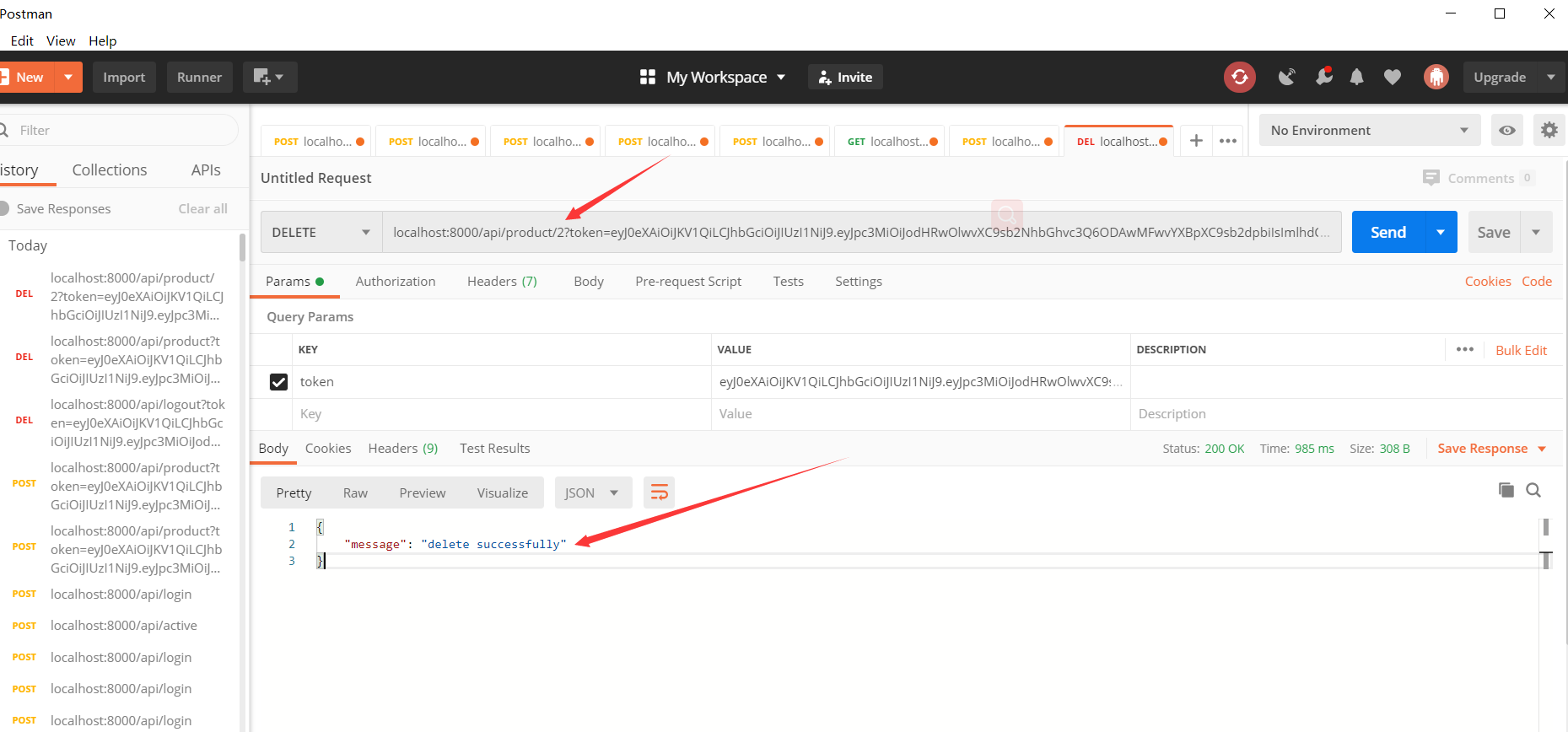
（4）对物品的增删改查测试(/product/)

增：post请求添加物品，结果如下图两个示例



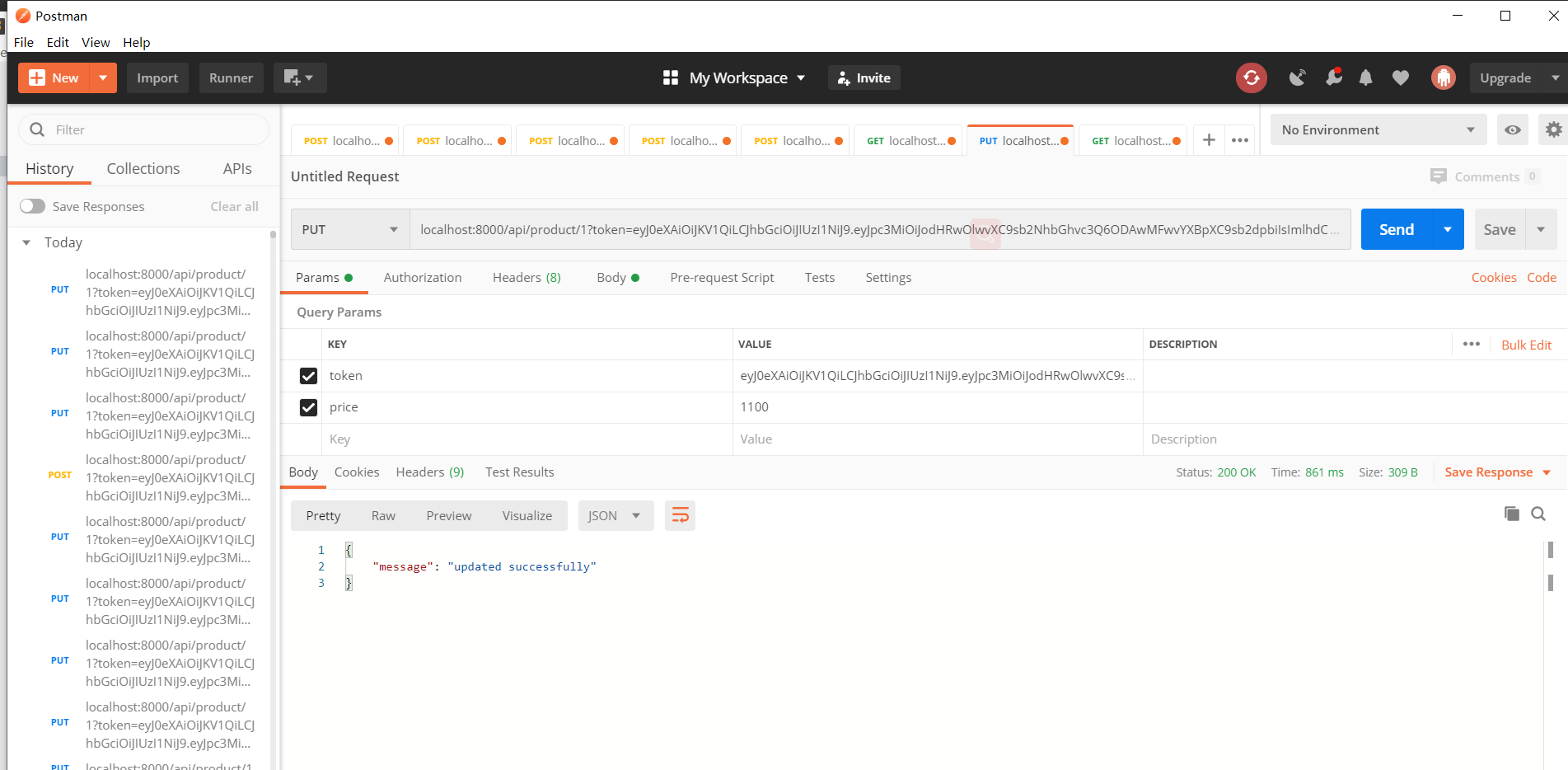


删：delete请求删除第二个产品

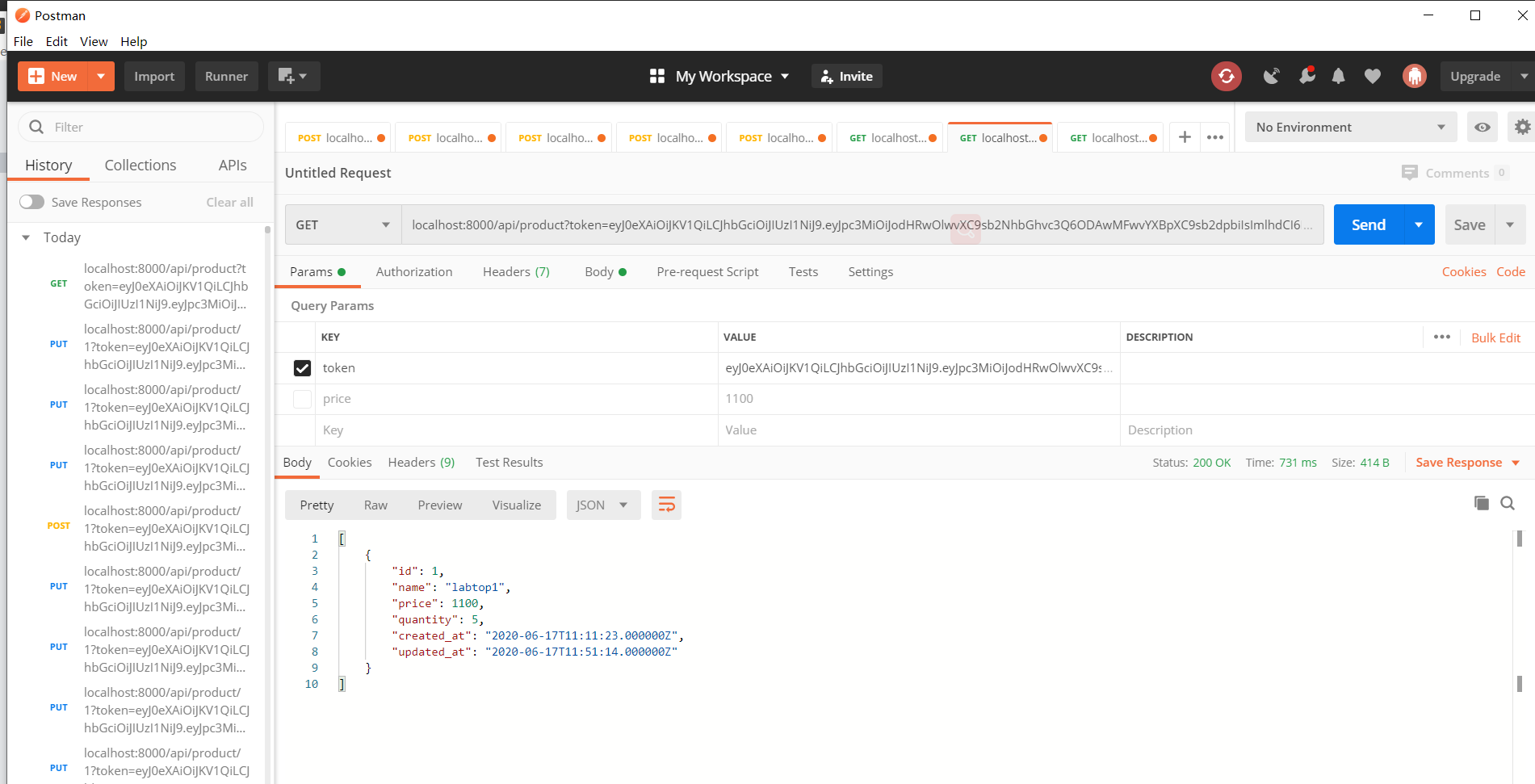


（后面的“查”会显示结果）

改：使用put请求修改物品信息，把物品1的价格改为1100

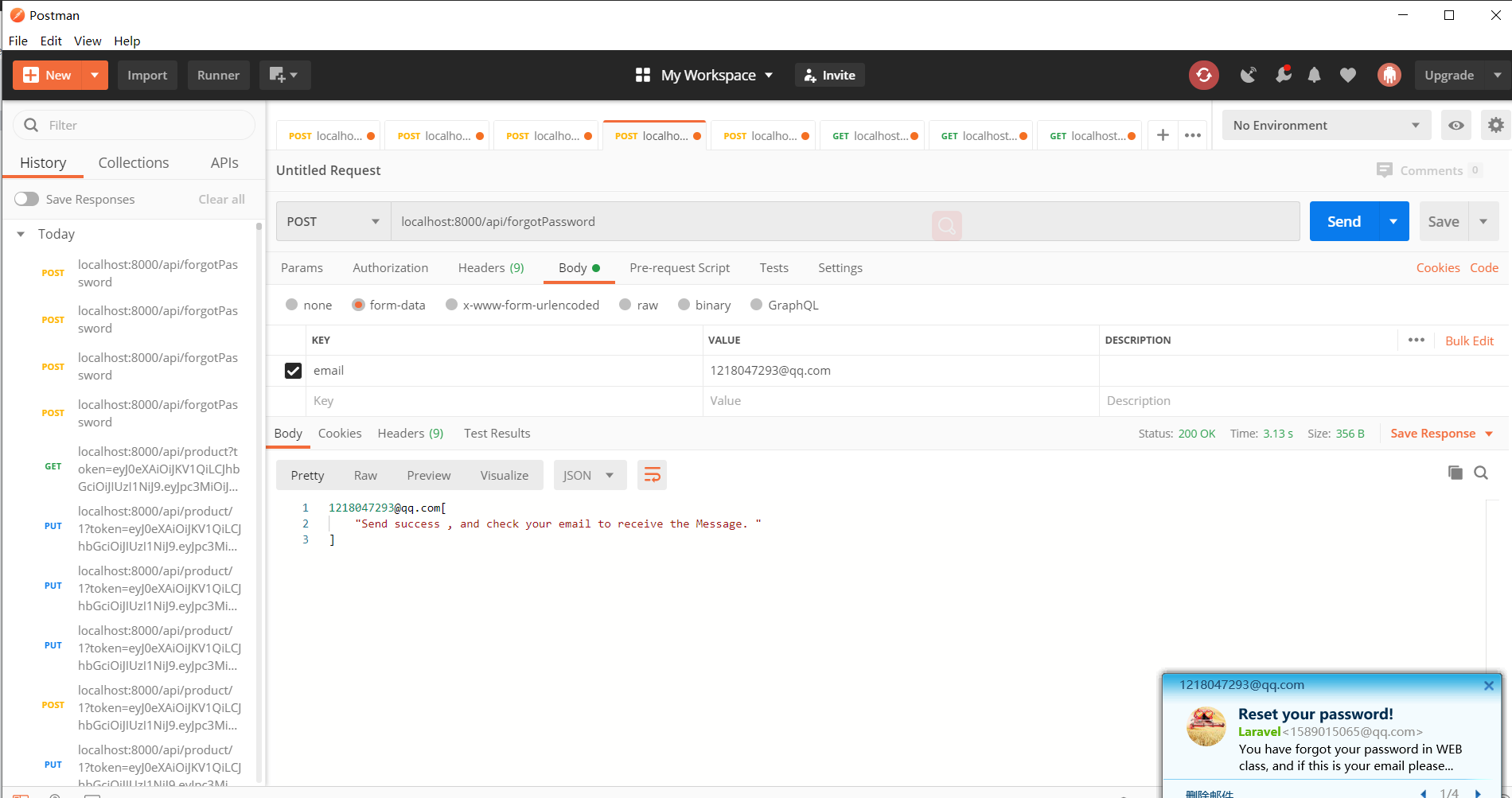


查：查看所有物品，可以看到物品2已经被删除，物品1的价格已经成功修改

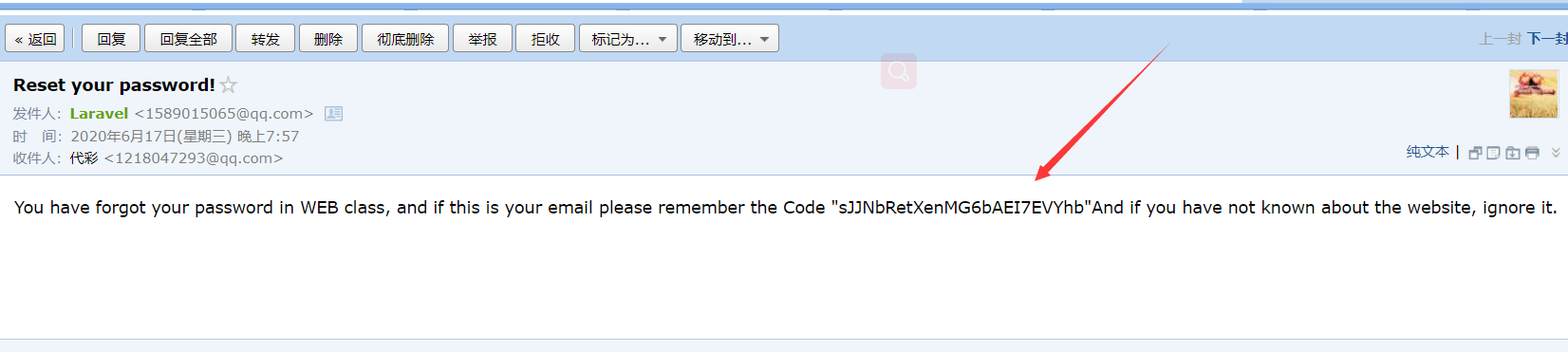


（5）修改密码测试(/forgetPassword, /resetPassword)

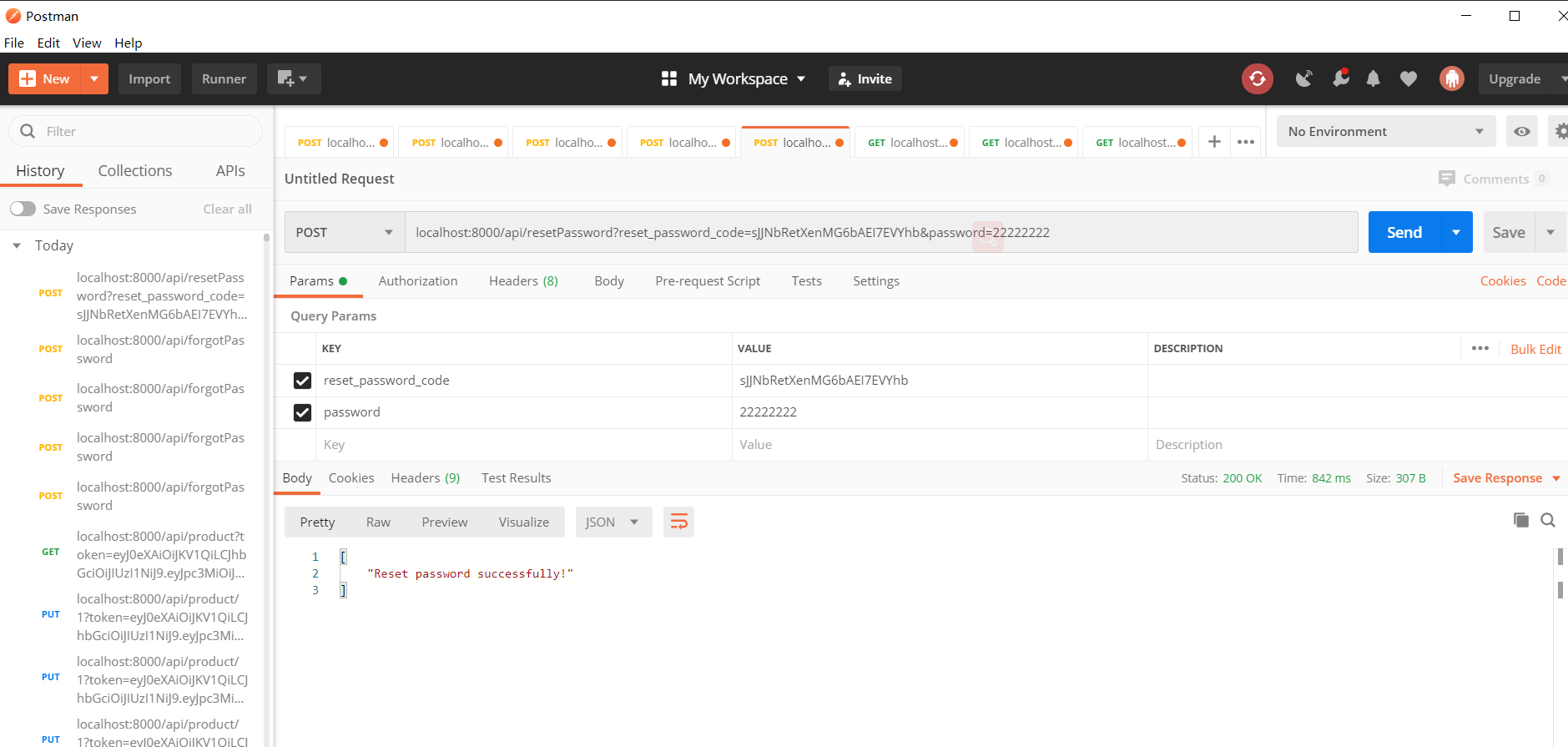
先访问/forgetPassword接口，通过邮件发送验证码



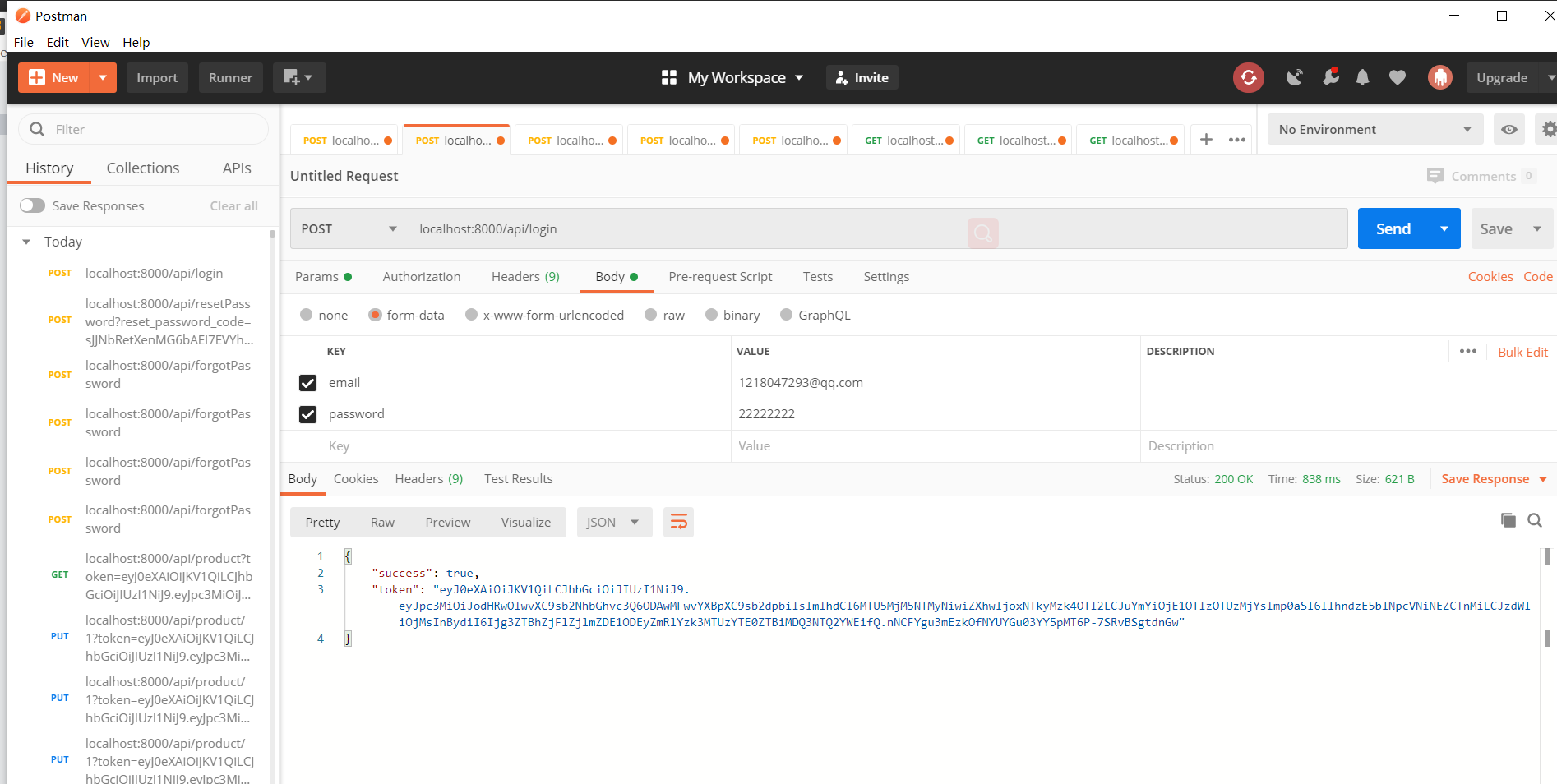
邮件验证码



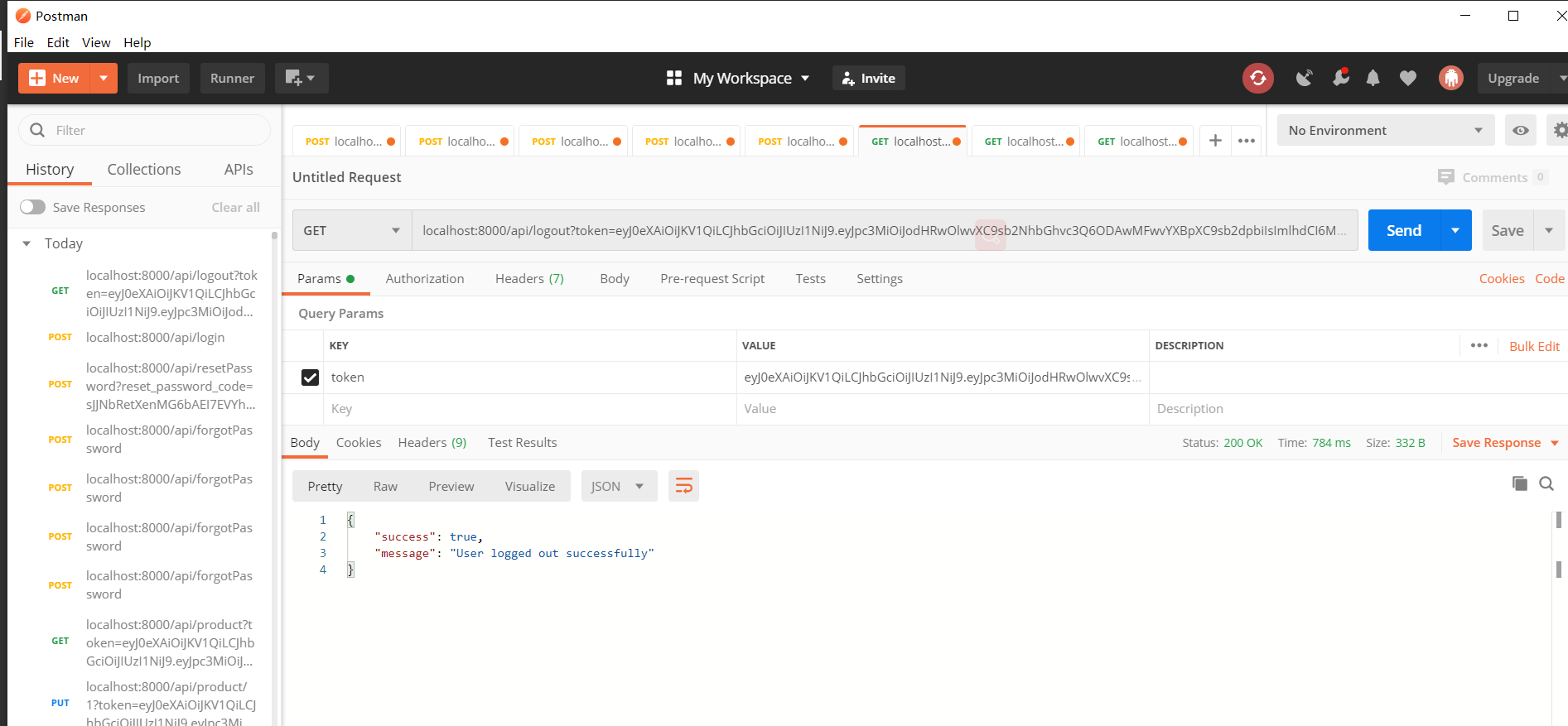
访问/resetPassword接口进行密码修改



用新密码登陆，成功登陆如下



（6）登出测试(/logout)



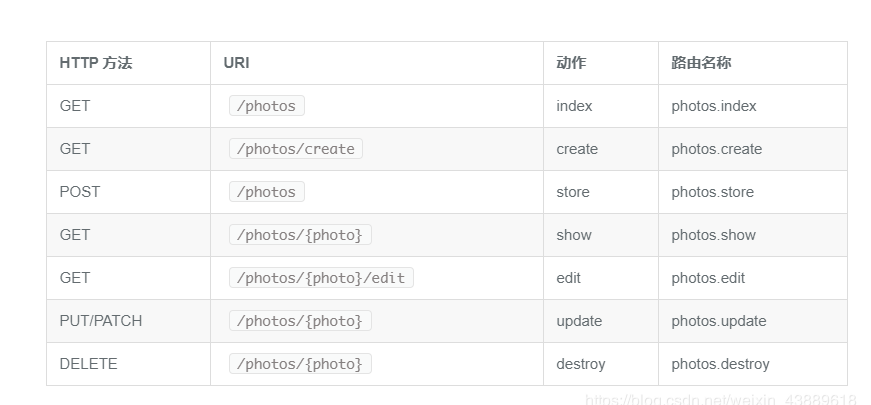
1. **实验分析总结及心得（该部分也是评分的一个重点）**

（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）

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。

这次的实验相比于实验四要难了好多好多，花了将近两个星期的时间，差点就做不出来了，但最终能够基本地实现各个功能也还是非常开心的！通过这个项目，让我更加深入地了解laravel。在模块功能实现方面，实验四使用的是内置的注册、登陆、邮箱验证等模块，使用起来非常方便，只需要简单地运行相应地命令以及修改配置文件即可使用，而实验五要求我们自己写代码来实现这些功能。亲自写各个模块之间的逻辑关系时候，才发现自己整个流程结构还不是特别的熟悉，一开始无从下手。在看了官方文档，以及CSDN上很多文章的解释后才开始有了思路。

在写控制器的时候，也想到了一些安全安全性的问题，比如用户没有输入正常的用户名、密码等，如果没有对参数进行验证，可能会遭到注入等恶意攻击，所以在把数据保存到数据库前，需要对接收到的参数使用validate模块进行检验，如果不合法则拒绝此次请求，并返回特定的信息以及状态码406（not accept）。而在写对物品增删改查的路由时，一开始我在/api下写的接口是/product/store，product/delete，product/update，product/index。但后来发现laravel框架下有一个很简洁的路由写法—Route::resource()，上述四个接口，其实在api.php文件中只需要写上/product就够了，在控制器里分别实现store（），destory（），update（），show（）和index（），在请求时分别使用post，delete，put,和get方法即可达到增删改查的目的。原来的四个路由接口现在只需要一个就可以实现，简化了路由文件里的逻辑关系。



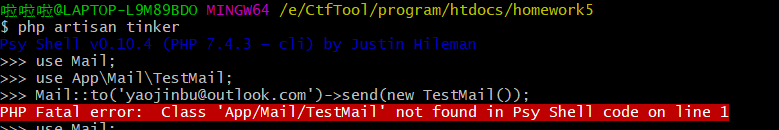
（1）一开始在使用邮件模块实现发送激活账号验证码的逻辑过程中遇到了很大的问题，在这里先感谢一波李岩同学的经验分享。刚开始我的想法是使用引入Auth下的MustVerifyEmail，但却一直不明白要怎么把这个模块添加到我的控制器里，在跟李岩交流一番后才意识到那个模块应该是给laravel内置的注册登陆模块用的，这次实验是要我们使用Mail类来实现发送邮件验证码的功能！哎，困扰我整个周末达的问题终于有了解决思路。在php中文文档网站（学院君）上找到了Mail类的使用方法，又花了一两天终于实现了发送邮箱验证码进行验证的逻辑功能。

<https://xueyuanjun.com/post/213.html>

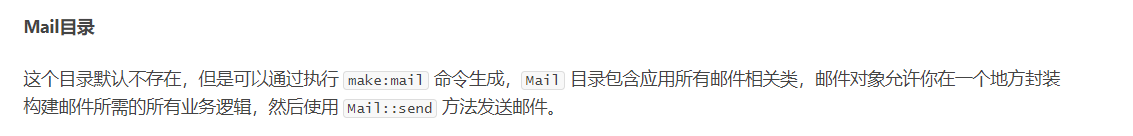
（2）对物品的增删改查。laravel这个框架提供了好几种方式来对数据库进行操作，一开始我是用最基本的sql语句进行处理，但写了好一会觉得这样子要写好多啊！于是到laravel文档里看看其他实现方式，发现了Eloquent，用它来实现增删改查，真的很简洁方便！

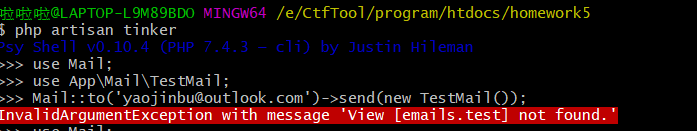
（3）邮箱验证模块在使用时也遇到了一些问题

使用tinker测试邮箱功能

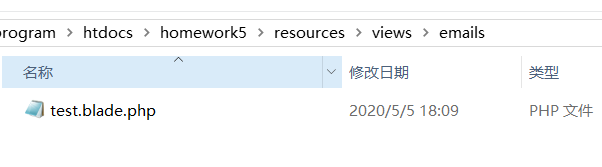


不知道为啥它一开始没有帮我生成Mail这个模块，后来又莫名其妙地有了，问题它自己就解决了。查了下文档，应该是这个原因





简单地添加一个Mail的blade网页模板就可以解决问题



成功收到邮件



（4）如果同时在数据迁移的配置函数和控制器里都加了unique限制，也会报错，提示Integrity constraint violation，只需保留其中一个限制即可解决问题





除了上面所列举的，其实还遇到了很多很多问题，只不过太过于细碎，而且也没有及时记录下来，所以在这里就只是列举了几个印象比较深的问题。