REST API with Laravel 5.8 using Laravel Passport

Sadam Bapunawar
Jun 30, 2019 · 2 min read

In this article, I will show how we can create REST API using laravel passport package by developing four API signup, login, logout, and getuser.

Laravel makes API authentication using Laravel Passport, which provides a full OAuth2 server implementation for your Laravel application in a matter of minutes.

Step 1: Install Laravel by issuing the Composer create-project command in your terminal.

we can get more details about server requirement and laravel installation on laravel documentation.

```
composer create-project --prefer-dist laravel/laravel blog
```

download laravel hosted with ❤ by **GitHub** **view raw**

Step 2: Install Passport via the Composer package manage.

```
composer require laravel/passport
```

Install laravel Passport hosted with ❤ by **GitHub** **view raw**

Step 3: Run Migration command.

The Passport service provider registers its own database migration directory with the framework, so you should migrate your database after installing the package. The Passport migrations will create the tables your application needs to store clients and access tokens.

```
php artisan migrate
```

artisan migrate hosted with ❤ by **GitHub** **view raw**

Step 4: Run Passport install command.

This command will create the encryption keys needed to generate secure access tokens.

```
php artisan passport:install
```

passport install hosted with ❤ by **GitHub** **view raw**

Step 5: Passport Configuration.

Next we have to configure passport by changing 3 files.

1. add the Laravel\Passport\HasApiTokens trait to your App/User model.

app/user.php

```
<?php

namespace App;

use Laravel\Passport\HasApiTokens;
use Illuminate\Notifications\Notifiable;
use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Foundation\Auth\User as Authenticatable;

class User extends Authenticatable
{
    use HasApiTokens, Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'name', 'email', 'password',
    ];

    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
        'password', 'remember_token',
    ];

    /**
     * The attributes that should be cast to native types.
     *
     * @var array
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
    ];
}
```

has api tokens hosted with ❤ by **GitHub** **view raw**

2 . Next, you should call the Passport::routes() method within the boot method of your AuthServiceProvider.

app/Providers/AuthServiceProvider.php

```
<?php

namespace App\Providers;
use Laravel\Passport\Passport;
use Illuminate\Support\Facades\Gate;
use Illuminate\Foundation\Support\Providers\AuthServiceProvider as ServiceProvider;

class AuthServiceProvider extends ServiceProvider
{
    /**
     * The policy mappings for the application.
     *
     * @var array
     */
    protected $policies = [
        // 'App\Model' => 'App\Policies\ModelPolicy',
    ];

    /**
     * Register any authentication / authorization services.
     *
     * @return void
     */
    public function boot()
    {
        $this->registerPolicies();

        //
        Passport::routes();
    }
}
```

passport routes hosted with ❤ by **GitHub** **view raw**

3 . Finally, in your config/auth.php configuration file, you should set the driver option of the api authentication guard to passport.

config/auth.php

```
<?php

return [

    'defaults' => [
        'guard' => 'web',
        'passwords' => 'users',
    ],

    'guards' => [
        'web' => [
            'driver' => 'session',
            'provider' => 'users',
        ],

        'api' => [
            'driver' => 'passport',
            'provider' => 'users',
            'hash' => false,
        ],
    ],

    'providers' => [
        'users' => [
            'driver' => 'eloquent',
            'model' => App\User::class,
        ],

        // 'users' => [
        //     'driver' => 'database',
        //     'table' => 'users',
        // ],
    ],



    'passwords' => [
        'users' => [
            'provider' => 'users',
            'table' => 'password_resets',
            'expire' => 60,
        ],
    ],

];
```

config auth hosted with ❤ by **GitHub** **view raw**

Step 6: Create API Routes

routes/api.php

```
<?php

use Illuminate\Http\Request;

/*
|--------------------------------------------------------------------------
| API Routes
|--------------------------------------------------------------------------
|
| 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::group([ 'prefix' => 'auth'], function (){
```

```
Route::group(['prefix' => 'auth'], function (){
    Route::group(['middleware' => ['guest:api']], function () {
        Route::post('login', 'API\AuthController@login');
        Route::post('signup', 'API\AuthController@signup');
    });
    Route::group(['middleware' => 'auth:api'], function() {
        Route::get('logout', 'API\AuthController@logout');
        Route::get('getuser', 'API\AuthController@getUser');
    });
});
```

laravel rest api hosted with ❤ by **GitHub** **view raw**

Step 7: Create a Common Response Controller to return response.

app/Http/Controllers/API/ResponseController.php

```
<?php

namespace App\Http\Controllers\API;


use Illuminate\Http\Request;
use App\Http\Controllers\Controller as Controller;
use Illuminate\Support\Facades\Auth;
use App\User;

class ResponseController extends Controller
{
    public function sendResponse($response)
    {
        return response()->json($response, 200);
    }


    public function sendError($error, $code = 404)
    {
        $response = [
            'error' => $error,
        ];
        return response()->json($response, $code);
    }
}
```

response controller hosted with ❤ by **GitHub** **view raw**

Step 8: Create a Auth Controller.

app/Http/Controllers/API/AuthController.php

```php
<?php

namespace App\Http\Controllers\API;


use Illuminate\Http\Request;
use App\Http\Controllers\API\ResponseController as ResponseController;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Auth;
use App\User;
use Validator;

class AuthController extends ResponseController
{
    //create user
    public function signup(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'name' => 'required|string|',
            'email' => 'required|string|email|unique:users',
            'password' => 'required',
            'confirm_password' => 'required|same:password'
        ]);

        if($validator->fails()){
            return $this->sendError($validator->errors());
        }

        $input = $request->all();
        $input['password'] = bcrypt($input['password']);
        $user = User::create($input);
        if($user){
            $success['token'] =  $user->createToken('token')->accessToken;
            $success['message'] = "Registration successfull..";
            return $this->sendResponse($success);
        }
        else{
            $error = "Sorry! Registration is not successfull.";
            return $this->sendError($error, 401);
        }

    }

    //login
    public function login(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'email' => 'required|string|email',
            'password' => 'required'
        ]);

        if($validator->fails()){
            return $this->sendError($validator->errors());
        }

        $credentials = request(['email', 'password']);
        if(!Auth::attempt($credentials)){
            $error = "Unauthorized";
            return $this->sendError($error, 401);
        }
        $user = $request->user();
        $success['token'] =  $user->createToken('token')->accessToken;
        return $this->sendResponse($success);
    }

    //logout
    public function logout(Request $request)
    {

        $isUser = $request->user()->token()->revoke();
        if($isUser){
            $success['message'] = "Successfully logged out.";
            return $this->sendResponse($success);
        }
        else{
            $error = "Something went wrong.";
            return $this->sendResponse($error);
        }


    }

    //getuser
    public function getUser(Request $request)
    {
        //$id = $request->user()->id;
        $user = $request->user();
        if($user){
            return $this->sendResponse($user);
        }
        else{
            $error = "user not found";
```

```
                return $this->sendResponse($error);
            }
        }
    }
```

9. Now run the local development server.

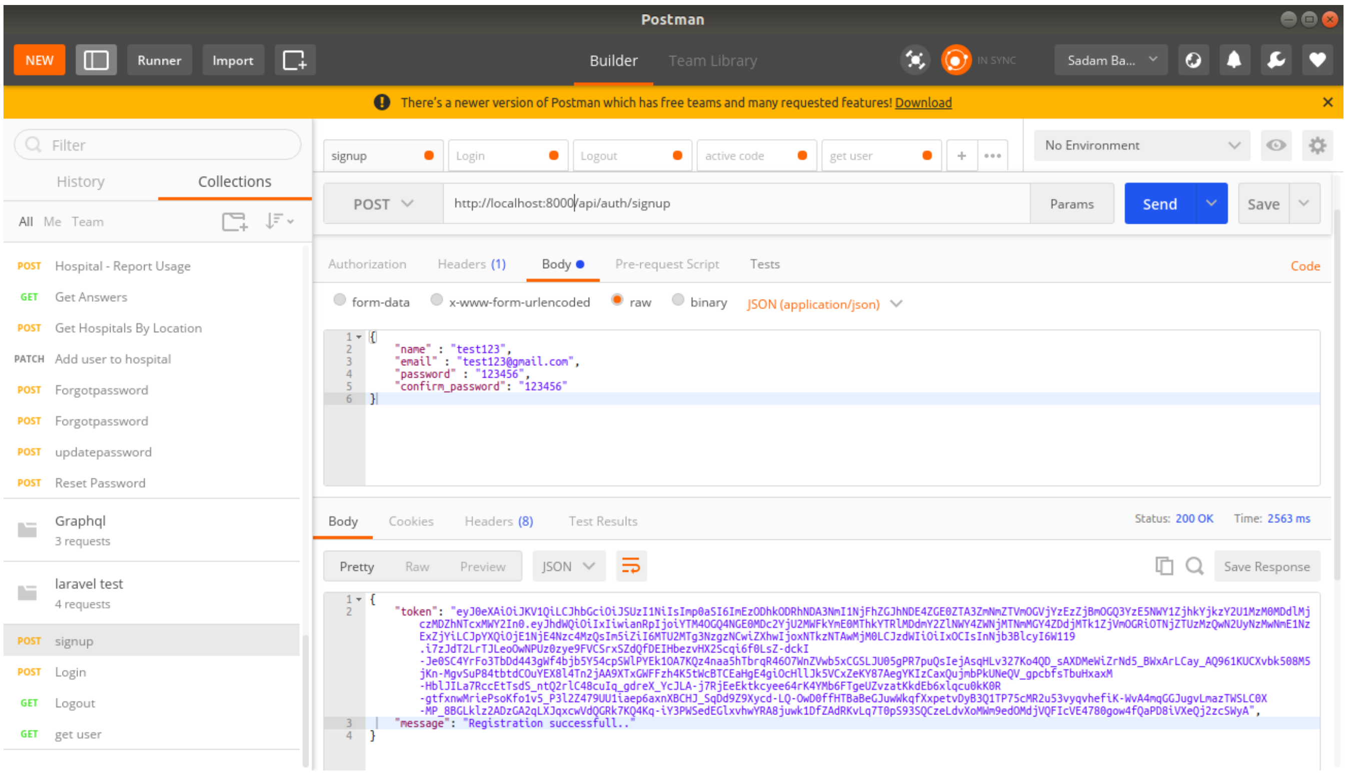
This command will start a development server at localhost:8000.

```
php artisan serve
```

php artisan serve hosted with ❤ by **GitHub** **view raw**

Test : we can use postman tool test the API.

1. signup



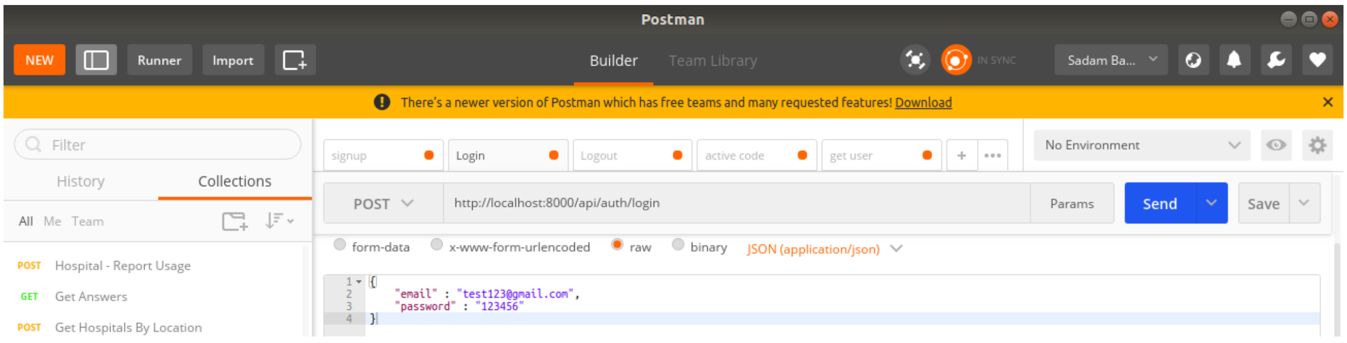
POST http://localhost:8000/api/auth/signup

```
{
    "name" : "test123",
    "email" : "test123@gmail.com",
    "password" : "123456",
    "confirm_password": "123456"
}
```

Status: 200 OK Time: 2563 ms

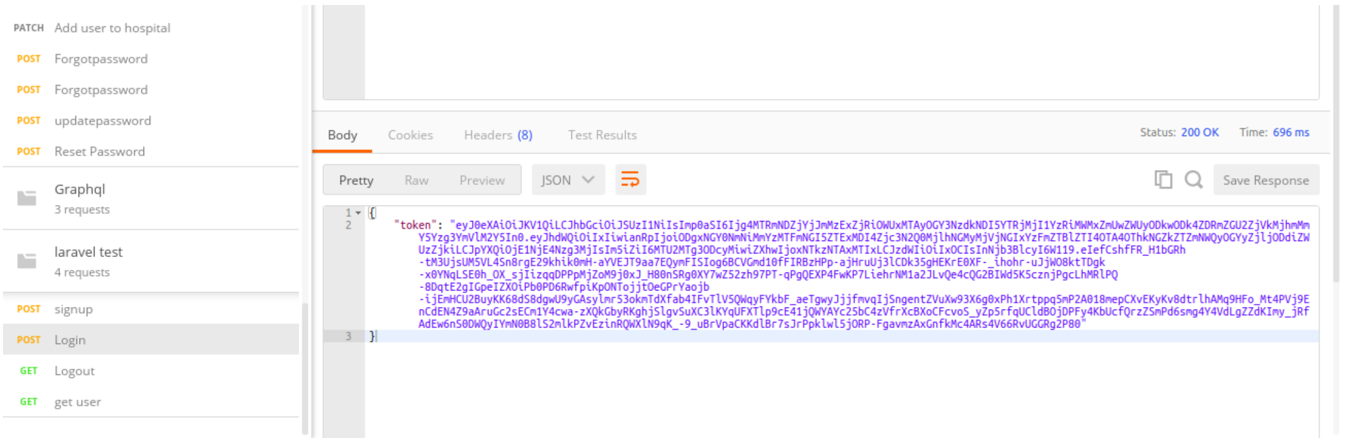
```
{
    "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImp0aSI6ImEzODhkODRhNDA3NmI1NjFhZGJhNDE4ZGE0ZTA3ZmNmZTVmOGVjYzEzZjBmOGQ3YzE5NWY1ZjhkYjkzY2U1MzM0MDdlMj
        czMDZhNTcxMWY2In0.eyJhdWQiOiIxIiwianRpIjoiYTM4OGQ4NGE0MDc2YjU2MWFkYmE0MThkYTRlMDdmY2ZlNWY4ZWNjMTNmMGY4ZDdjMTk1ZjVmOGRiOTNjZTUzMzQwN2UyNzMwNmE1Nz
        ExZjYiLCJpYXQiOjE1NjE4Nzc4MzQsIm5iZiI6MTU2MTg3NzgzNCwiZXhwIjoxNTkzNTAwMjM0LCJzdWIiOiIxOCIsInNjb3BlcyI6W119
        .i7zJdT2LrTJLeoOwNPUz0zye9FVCSrxSZdQfDEIHbezvHX2Scqi6f0LsZ-dckI
        -Je0SC4YrFo3TbDd443gWf4bjb5Y54cpSWlPYEk1OA7KQz4naa5hTbrqR4607WnZVwb5xCGSLJU05gPR7puQsIejAsqHLv327Ko4QD_sAXDMeWiZrNd5_BWxArLCay_AQ961KUCXvbk508M5
        jKn-MgvSuP84tbtdCOuYEX8l4Tn2jAA9XTxGWFFzh4K5tWcBTCEaHgE4giOcHllJk5VCxZeKY87AegYKIzCaxQujmbPkUNeQV_gpcbfsTbuHxaxM
        -HblJILa7RccEtTsdS_ntQ2rlC48cuIq_gdreX_YcJLA-j7RjEeEktkcyee64rK4YMb6FTgeUZvzatKkdEb6xlqcu0kK0R
        -gtfxnwMriePsoKfo1v5_P3l2Z479UU1iaep6axnXBCHJ_SqDd9Z9Xycd-LQ-OwD0ffHTBaBeGJuwWkqfXxpetvDyB3Q1TP75cMR2u53vyqvhefiK-WvA4mqGGJugvLmazTWSLC0X
        -MP_8BGLklz2ADzGA2qLXJqxcwVdQGRk7KQ4Kq-iY3PWSedEGlxvhwYRA8juwk1DfZAdRKvLq7T0pS93SQCzeLdvXoMWm9edOMdjVQFIcVE478Ogow4fQaPD8iVXeQj2zcSWyA",
    "message": "Registration successfull.."
}
```

2. Login



POST http://localhost:8000/api/auth/login

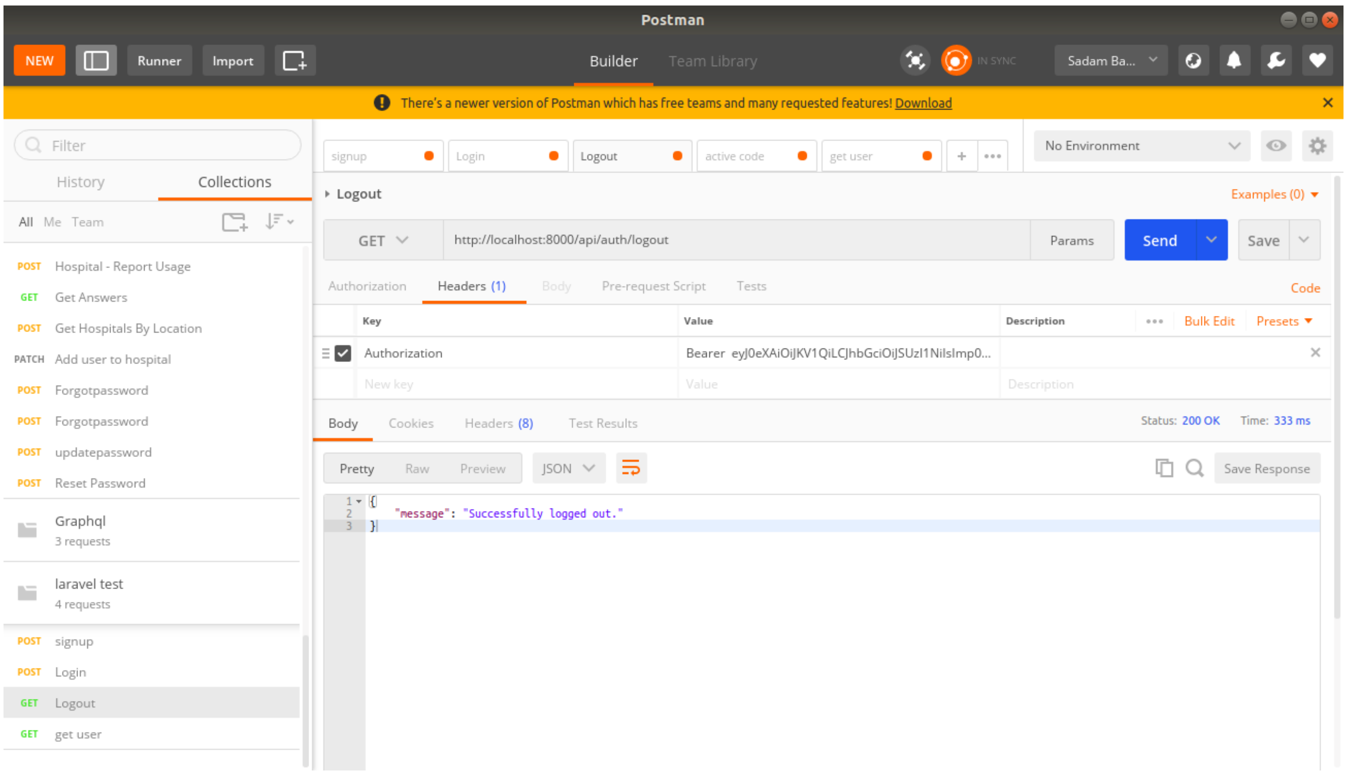
```
{
    "email" : "test123@gmail.com",
    "password" : "123456"
}
```



Status: 200 OK Time: 696 ms

```
{
    "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImp0aSI6Ijg4MTRmNDZjYjJmMzExZjRiOWUxMTAyOGY3NzdkNDI5YTRjMjI1YzRiMWMxZmUwZWUyODkwODk4ZDRmZGU2ZjVkMjhmMm
        Y5Yzg3YmVlM2Y5In0.eyJhdWQiOiIxIiwianRpIjoiODgxNGY0NmNiMmYzMTFmNGI5ZTExMDI4Zjc3N2Q0MjlhNGMyMjVjNGIxYzFmZTBlZTI4OTA4OThkNGZkZTZmNWQyOGYyZjljODdiZW
        UzZjkiLCJpYXQiOjE1NjE4Nzg3MjIsIm5iZiI6MTU2MTg3ODcyMiwiZXhwIjoxNTkzNTAxMTIxLCJzdWIiOiIxOCIsInNjb3BlcyI6W119.eIefCshfFR_H1bGRh
        -tM3UjsUM5VL4Sn8rgE29khik0mH-aYVEJT9aa7EQymFISIog6BCVGmd10fFIRBzHPp-ajHruUj3lCDk35gHEKrE0XF-_ihohr-uJjWO8ktTDgk
        -x0YNqLSE0h_OX_sjIizqqDPPpMjZoM9j0xJ_H80nSRg0XY7wZ52zh97PT-qPgQEXP4FwKP7LiehrNM1a2JLvQe4cQG2BIWd5K5csznjPgcLhMRlPQ
        -8DqtE2gIGpeIZXOiPb0PD6RwfpiKpONTojjtOeGPrYaojb
        -ijEmHCU2BuyKK68dS8dgwU9yGAsylmr53okmTdXfab4IFvTlV5QWqyFYkbF_aeTgwyJjjfmvqIjSngentZVuXw93X6g0xPh1Xrtppq5mP2A018mepCXvEKyKv8dtrlhAMq9HFo_Mt4PVj9E
        nCdEN4Z9aAruGc2sECm1Y4cwa-zXQkGbyRKghjSlgvSuXC3lKYqUFXTlp9cE41jQWYAYc25bC4zVfrXcBXoCFcvoS_yZp5rfqUCldBOjDPFy4KbUcfQrzZSmPd6smg4Y4VdLgZZdKImy_jRf
        AdEw6nS0DWQyIYmN0B8lS2mlkPZvEzinRQWXlN9qK_-9_uBrVpaCKKdlBr7sJrPpklwl5jORP-FgavmzAxGnfkMc4ARs4V66RvUGGRg2P80"
}
```

3. Logout



Logout

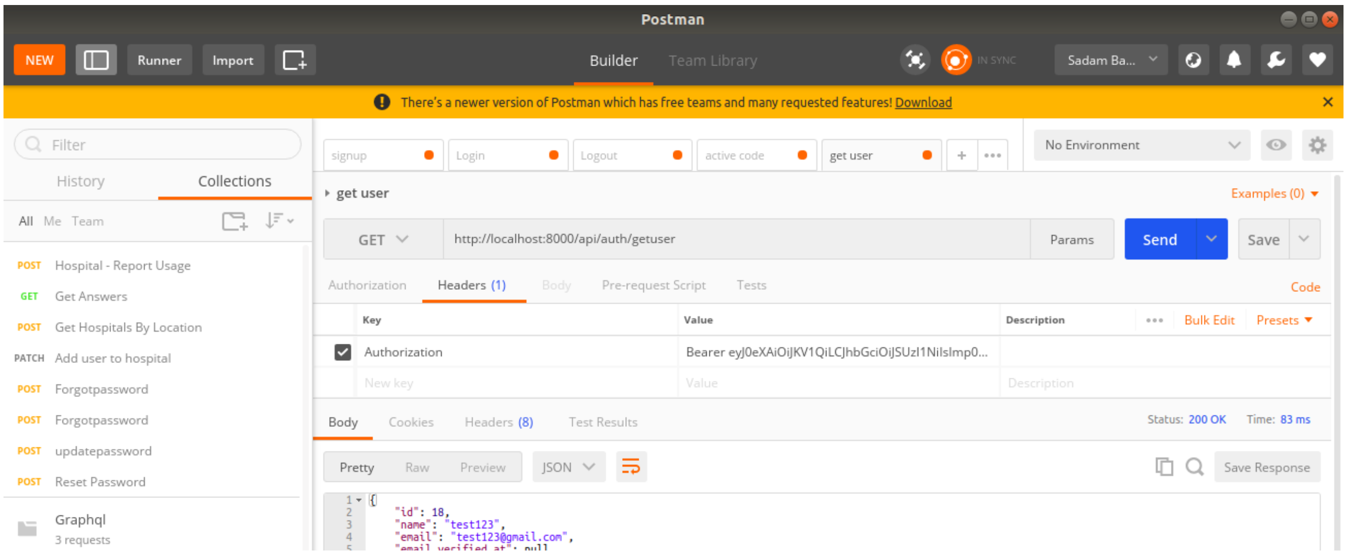
GET http://localhost:8000/api/auth/logout

Key	Value	Description
✓ Authorization	Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImp0...	
New key	Value	Description

Status: 200 OK Time: 333 ms

```
{
    "message": "Successfully logged out."
}
```

4. get user


Postman
NEW Runner Import
Builder Team Library
IN SYNC
Sadam Ba...
There's a newer version of Postman which has free teams and many requested features! Download
Filter
History Collections
All Me Team
POST Hospital - Report Usage
GET Get Answers
POST Get Hospitals By Location
PATCH Add user to hospital
POST Forgotpassword
POST Forgotpassword
POST updatepassword
POST Reset Password
Graphql
3 requests
signup Login Logout active code get user
No Environment
Examples (0)
Params Send Save
Authorization Headers (1) Body Pre-request Script Tests Code
Bulk Edit Presets
Body Cookies Headers (8) Test Results
Pretty Raw Preview JSON
Save Response

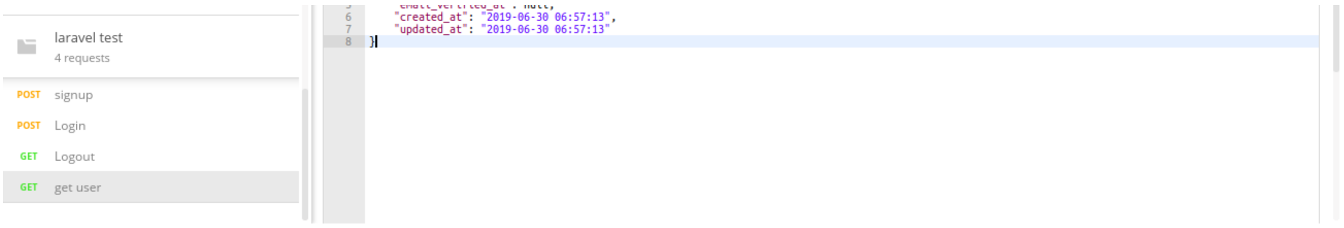
get user

GET http://localhost:8000/api/auth/getuser

Key	Value	Description
✓ Authorization	Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImp0...	
New key	Value	Description

Status: 200 OK Time: 83 ms

```
{
    "id": 18,
    "name": "test123",
    "email": "test123@gmail.com",
    "email_verified_at": null
```



```
    "email_verified_at": null,
    "created_at": "2019-06-30 06:57:13",
    "updated_at": "2019-06-30 06:57:13"
}
```

Github

Laravel PHP Passport Postman

About Help Legal