Step 1: Install Laravel 8

I am going to explain step by step from scratch so, we need to get fresh Laravel 8 application using bellow command, So open your terminal OR command prompt and run bellow command:

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

Step 2: Use Sanctum

```
composer require laravel/sanctum
```

After successfully install package, we need to publish configuration file with following command:

```
php artisan vendor:publish --provider="Laravel\Sanctum\SanctumServiceProvider"
```

we require to get default migration for create new sanctum tables in our database. so let's run bellow command.

```
php artisan migrate
```

Next, we need to add middleware for sanctum api, so let's add as like bellow:

app/Http/Kernel.php

```
....

'api' => [
    \Laravel\Sanctum\Http\Middleware\EnsureFrontendRequestsAreStateful::class,
    'throttle:api',
    \Illuminate\Routing\Middleware\SubstituteBindings::class,
],
```

Step 3: Sanctum Configuration

In this step, we have to configuration on three place model, service provider and auth config file. So you have to just following change on that file.

In model we added HasApiTokens class of Sanctum,

app/Models/User.php

```php
<?php

namespace App\Models;

use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable
{
    use HasFactory, Notifiable, HasApiTokens;

    /**
     * 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',
    ];
}
```

```
php artisan make:migration create_products_table
```

After this command you will find one file in following path **database/migrations** and you have to put bellow code in your migration file for create products table.

```
<?php

use Illuminate\Support\Facades\Schema;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateProductsTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('products', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->text('detail');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('products');
    }
}
```

After create migration we need to run above migration by following command:

```
php artisan migrate
```

After create "products" table you should create Product model for products, so first create file in this path app/Models/Product.php and put bellow content in item.php file:

app/Models/Product.php

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
```

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

Step 5: Create API Routes

In this step, we will create api routes. Laravel provide api.php file for write web services route. So, let's add new route on that file.

routes/api.php

```
<?php

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;

use App\Http\Controllers\API\RegisterController;
use App\Http\Controllers\API\ProductController;

/*
|--------------------------------------------------------------------------
| 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::post('register', [RegisterController::class, 'register']);
Route::post('login', [RegisterController::class, 'login']);

Route::middleware('auth:sanctum')->group( function () {
    Route::resource('products', ProductController::class);
});
```

Step 6: Create Controller Files

in next step, now we have create new controller as BaseController, ProductController and RegisterController, i created new folder "API" in Controllers folder because we will make alone APIs controller, So let's create both controller:

app/Http/Controllers/API/BaseController.php

```
<?php

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

```
{
    /**
     * success response method.
     *
     * @return \Illuminate\Http\Response
     */
    public function sendResponse($result, $message)
    {
        $response = [
            'success' => true,
            'data'    => $result,
            'message' => $message,
        ];

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

    /**
     * return error response.
     *
     * @return \Illuminate\Http\Response
     */
    public function sendError($error, $errorMessages = [], $code = 404)
    {
        $response = [
            'success' => false,
            'message' => $error,
        ];

        if(!empty($errorMessages)){
            $response['data'] = $errorMessages;
        }

        return response()->json($response, $code);
    }
}
```

app/Http/Controllers/API/RegisterController.php

```
<?php

namespace App\Http\Controllers\API;

use Illuminate\Http\Request;
use App\Http\Controllers\API\BaseController as BaseController;
use App\Models\User;
use Illuminate\Support\Facades\Auth;
use Validator;

class RegisterController extends BaseController
{
    /**
     * Register api
     *
     * @return \Illuminate\Http\Response
     */
    public function register(Request $request)
    {
```

```
        $validator = Validator::make($request->all(), [
        ]);

        if($validator->fails()){
            return $this->sendError('Validation Error.', $validator->errors());
        }

        $input = $request->all();
        $input['password'] = bcrypt($input['password']);
        $user = User::create($input);
        $success['token'] =  $user->createToken('MyApp')->plainTextToken;
        $success['name'] =  $user->name;

        return $this->sendResponse($success, 'User register successfully.');
    }

    /**
     * Login api
     *
     * @return \Illuminate\Http\Response
     */
    public function login(Request $request)
    {
        if(Auth::attempt(['email' => $request->email, 'password' => $request->password])){
            $user = Auth::user();
            $success['token'] =  $user->createToken('MyApp')->plainTextToken;
            $success['name'] =  $user->name;

            return $this->sendResponse($success, 'User login successfully.');
        }
        else{
            return $this->sendError('Unauthorised.', ['error'=>'Unauthorised']);
        }
    }
}
```

app/Http/Controllers/API/ProductController.php

```
<?php

namespace App\Http\Controllers\API;

use Illuminate\Http\Request;
use App\Http\Controllers\API\BaseController as BaseController;
use App\Models\Product;
use Validator;
use App\Http\Resources\Product as ProductResource;

class ProductController extends BaseController
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $products = Product::all();
```

```
        return $this->sendResponse(ProductResource::collection($products), 'Products retrieved successfully.');
    }


    /**
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        $input = $request->all();

        $validator = Validator::make($input, [
            'name' => 'required',
            'detail' => 'required'
        ]);

        if($validator->fails()){
            return $this->sendError('Validation Error.', $validator->errors());
        }

        $product = Product::create($input);

        return $this->sendResponse(new ProductResource($product), 'Product created successfully.');
    }

    /**
     * Display the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function show($id)
    {
        $product = Product::find($id);

        if (is_null($product)) {
            return $this->sendError('Product not found.');
        }

        return $this->sendResponse(new ProductResource($product), 'Product retrieved successfully.');
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, Product $product)
    {
        $input = $request->all();

        $validator = Validator::make($input, [
            'name' => 'required',
            'detail' => 'required'
        ]);

        if($validator->fails()){
            return $this->sendError('Validation Error.', $validator->errors());
        }

        $product->name = $input['name'];
```

```
        $product->detail = $input['detail'];

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy(Product $product)
    {
        $product->delete();

        return $this->sendResponse([], 'Product deleted successfully.');
    }
}
```

Step 7: Create Eloquent API Resources

This is a very important step of creating rest api in laravel 8. you can use eloquent api resources with api. it will helps you to make same response layout of your model object. we used in ProductController file. now we have to create it using following command:

```
php artisan make:resource Product
```

Now there created new file with new folder on following path:

app/Http/Resources/Product.php

```
<?php

namespace App\Http\Resources;

use Illuminate\Http\Resources\Json\JsonResource;

class Product extends JsonResource
{
    /**
     * Transform the resource into an array.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return array
     */
    public function toArray($request)
    {
        return [
            'id' => $this->id,
            'name' => $this->name,
            'detail' => $this->detail,
            'created_at' => $this->created_at->format('d/m/Y'),
            'updated_at' => $this->updated_at->format('d/m/Y'),
        ];
    }
}
```

Now we are ready to to run full restful api and also passport api in laravel, so let's run our example so run bellow command for quick

make sure in details api we will use following headers as listed bellow:

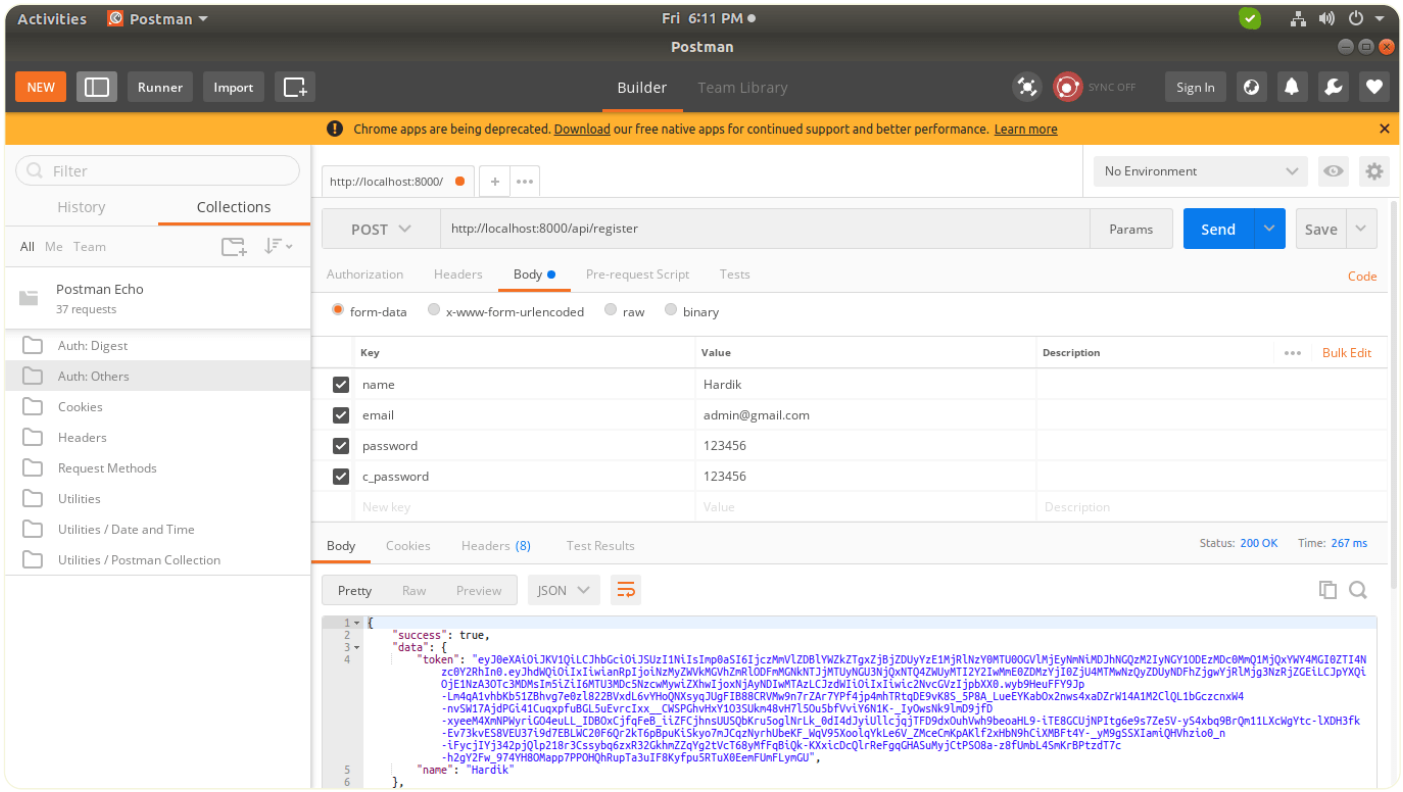
Read Also: Laravel 8 Firebase Web Push Notification Example

```
'headers' => [
    'Accept' => 'application/json',
    'Authorization' => 'Bearer '.$accessToken,
]
```

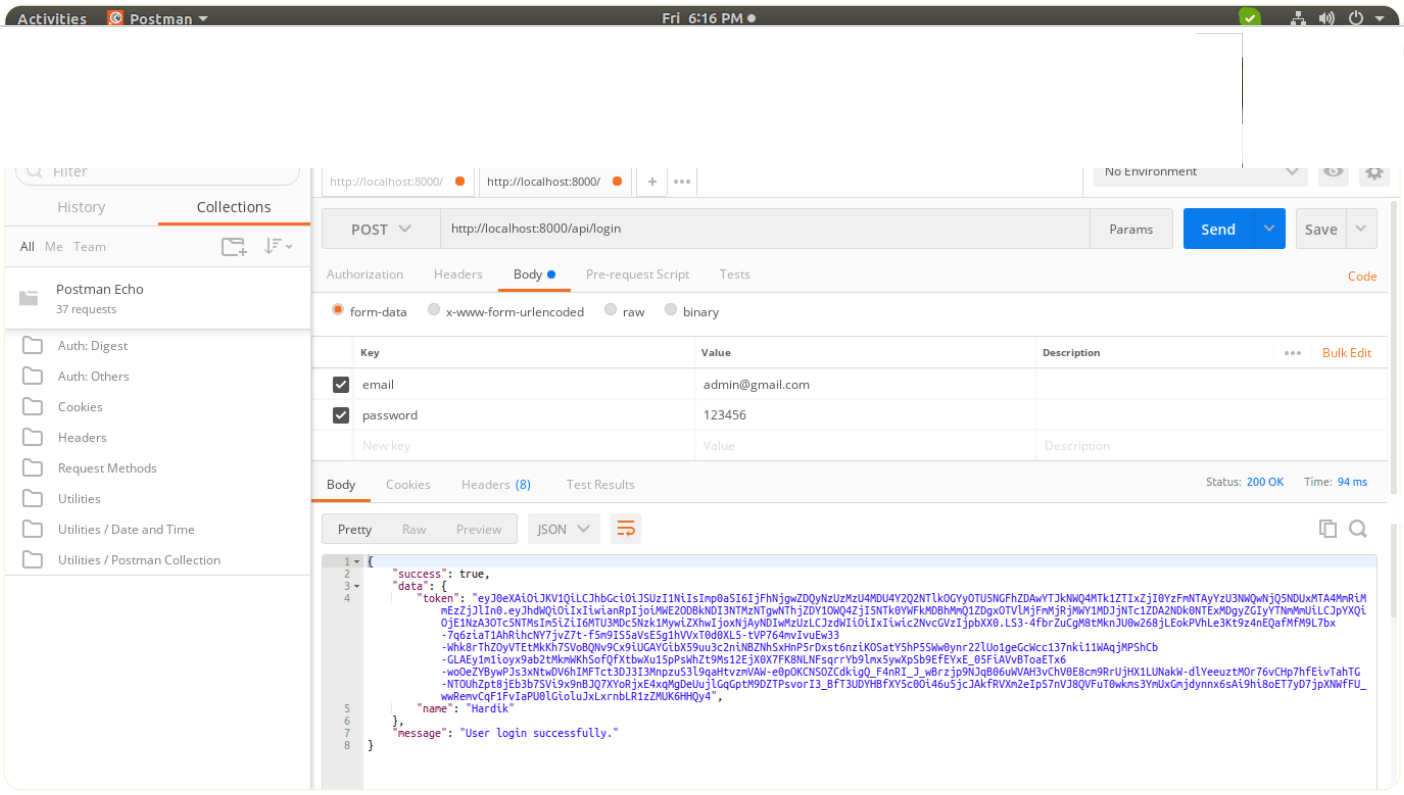
Here is Routes URL with Verb:

Now simply you can run above listed url like as bellow screen shot:

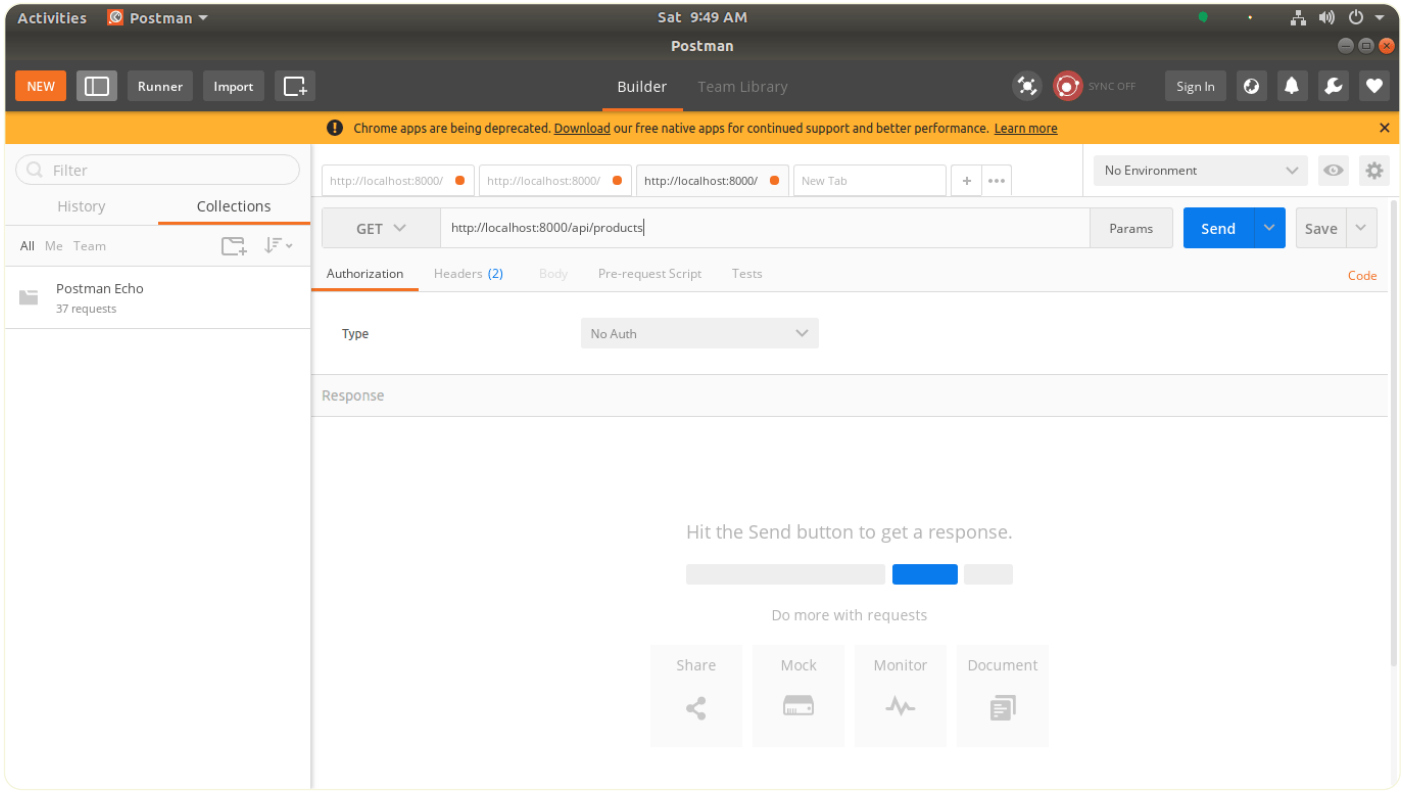
1) Register API: Verb:GET, URL:http://localhost:8000/api/register



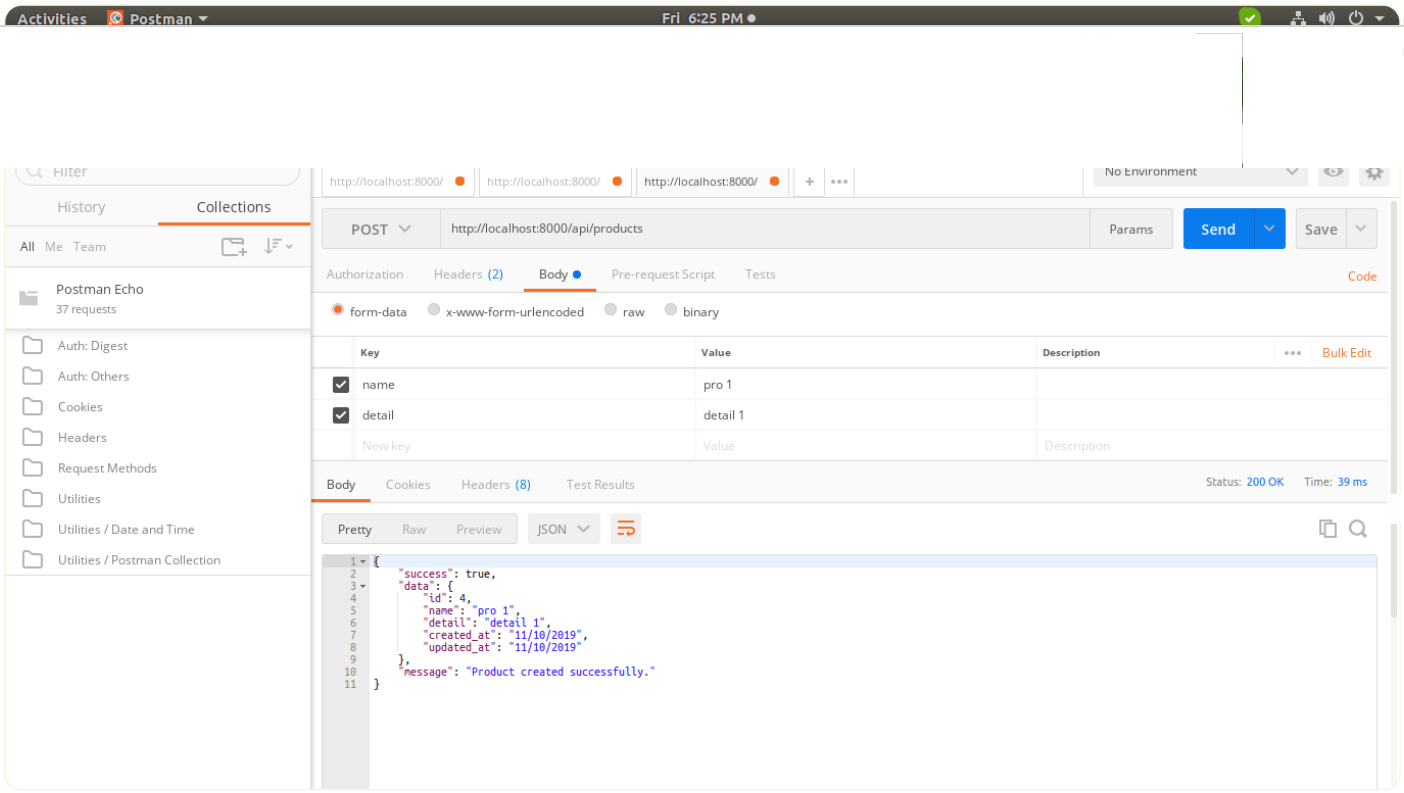
2) Login API: Verb:GET, URL:http://localhost:8000/api/login



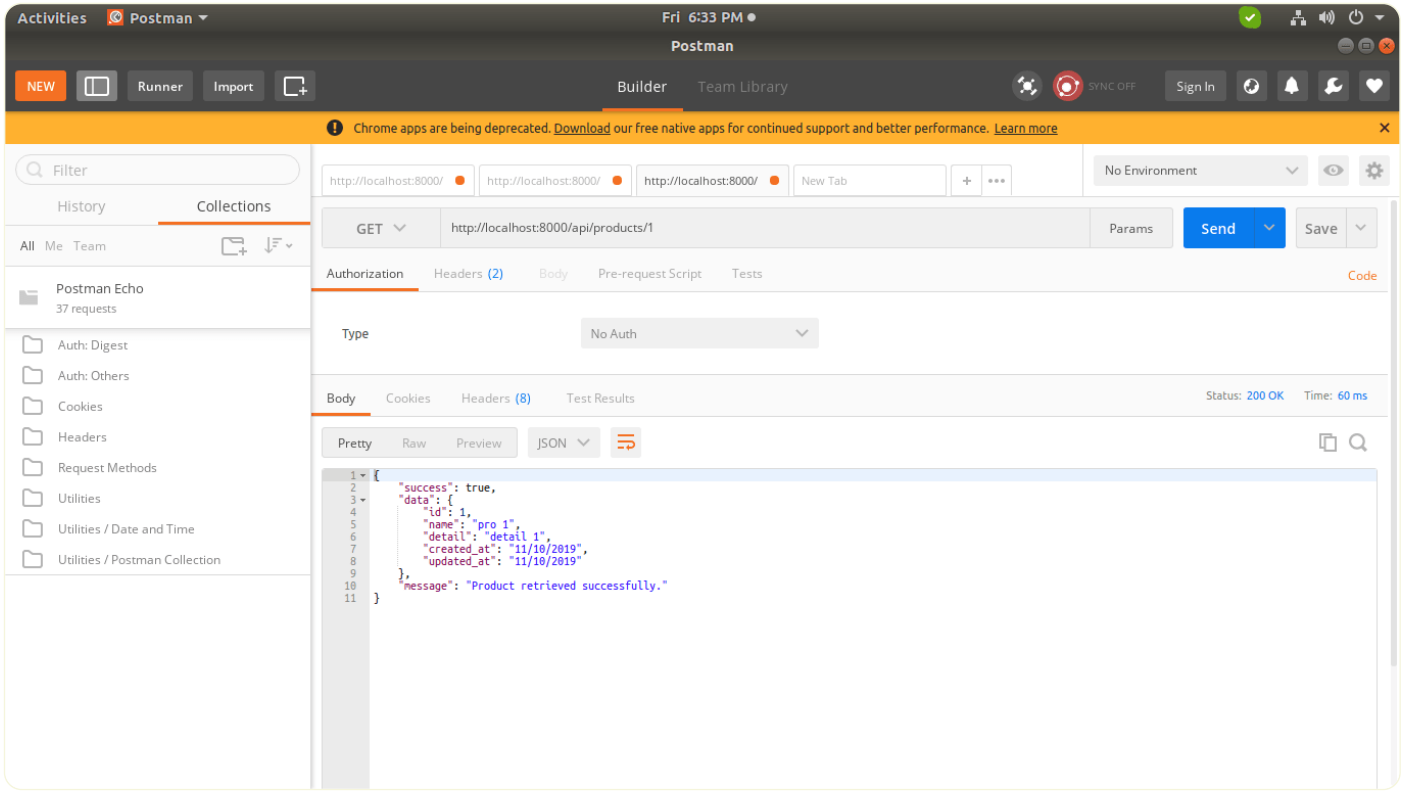
3) Product List API: Verb:GET, URL:http://localhost:8000/api/products



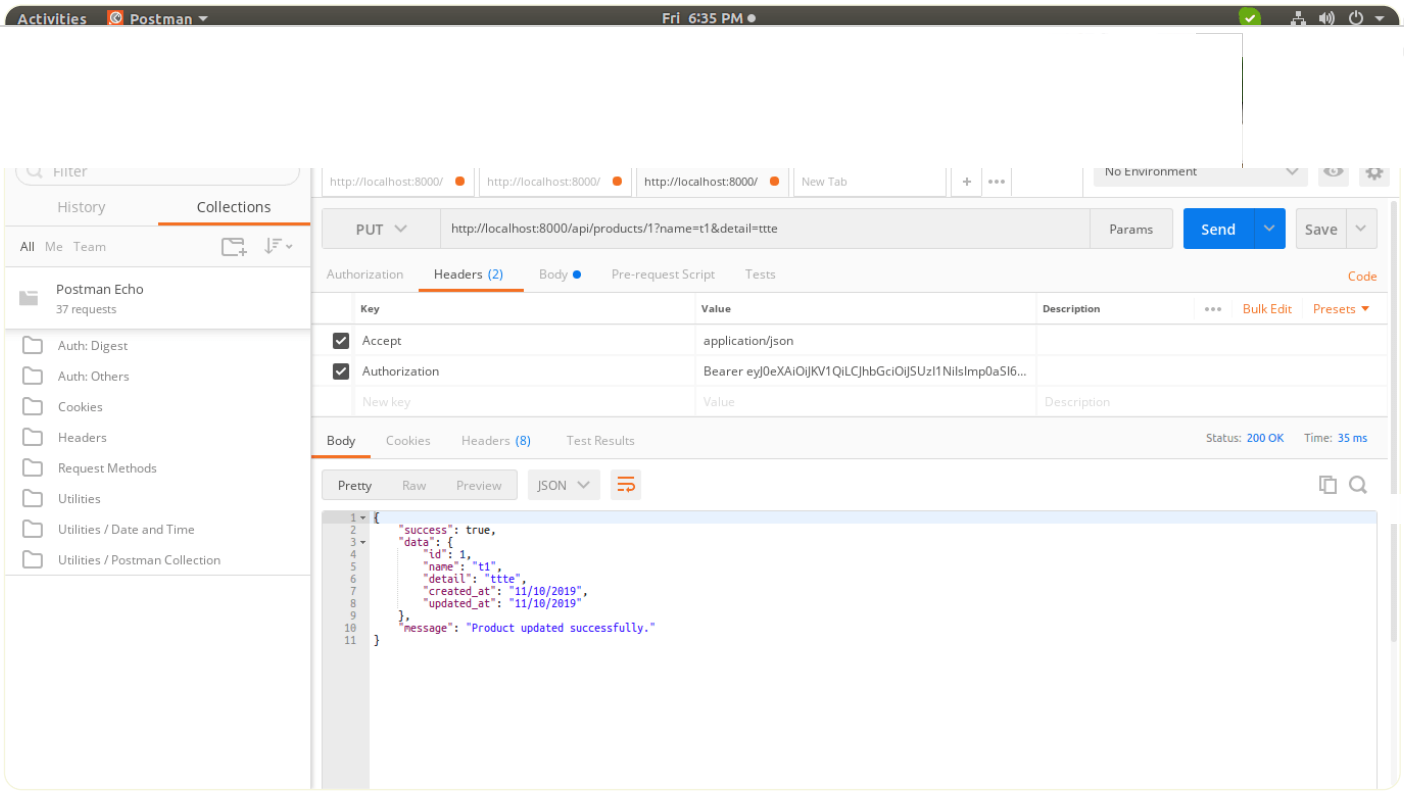
4) Product Create API: Verb:POST, URL:http://localhost:8000/api/products



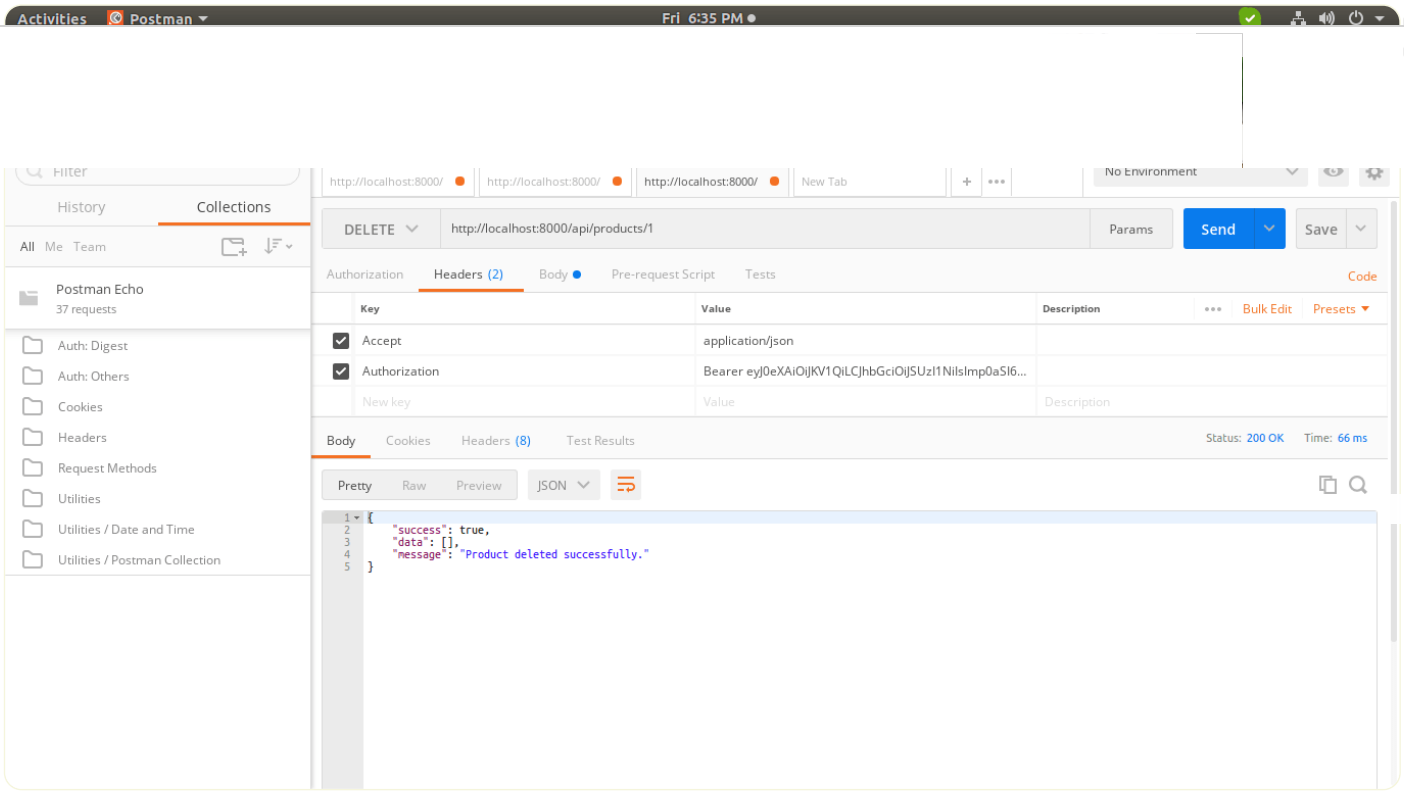
5) Product Show API: Verb:GET, URL:http://localhost:8000/api/products/{id}



6) Product Update API: Verb:PUT, URL:http://localhost:8000/api/products/{id}



7) Product Delete API: Verb:DELETE, URL:http://localhost:8000/api/products/{id}



You can download code from git: Download Code from Github

I hope it can help you...

Tags : Laravel Laravel 8

Hardik Savani

I'm a full-stack developer, entrepreneur and owner of Aatman Infotech. I live in India and I love to write tutorials and tips that can help to other artisan. I am a big fan of PHP, Laravel, Angular, Vue, Node, Javascript, JQuery, Codeigniter and Bootstrap from the early stage. I believe in Hardworking and Consistency.

Follow Me:

We are Recommending you

- Laravel 8 Stripe Payment Gateway Integration Example
- Laravel 8 Multi Auth (Authentication) Tutorial
- Laravel 8 Markdown | Laravel 8 Send Email using Markdown Example
- Drag & Drop File Uploading using Laravel 8 Dropzone JS
- Laravel 8 REST API with Passport Authentication Tutorial