Restful API with Laravel 9

Multiple Authentication Guards & Sanctum

Objective

- 1. What is Sanctum?
- 2. Configuring expiration time
- 3. Creating Member migration
- 4. Setting up the Model
- 5. Defining the Guard
- 6. Creating Member Controller
- 7. Creating Register Function
- 8. Creating Login Function
- 9. Creating Profile Function
- 10. Adding Routes
- 11. Testing

1. What is Sanctum?

- Sanctum provides a featherweight authentication system for SPAs (single page applications), mobile applications, and simple, token based APIs.
- Sanctum allows each user of your application to generate multiple API tokens for their account
- These tokens may be granted abilities / scopes which specify which actions the tokens are allowed to perform.
- Url: https://laravel.com/docs/9.x/sanctum

2. Configuring expiration time

- By default, Sanctum tokens never expire and may only be invalidated by revoking the token.
- If you would like to configure an expiration time, defined in "\config\sanctum.php" configuration file.

```
_____
Expiration Minutes
           _____
This value controls the number of minutes until an issued token will be
considered expired. If this value is null, personal access tokens do
not expire. This won't tweak the lifetime of first-party sessions.
'expiration' => 24 * 60,
```

3. Creating "Member" Migration

• To make the "members table", run the following command:

php artisan make:migration create_members_table

- The file will be created in "database/migrations" folder (https://laravel.com/docs/9.x/migrations)
- Add the highlighted codes at next slide, then migrate the database

php artisan migrate

```
public function up()
    Schema::create('members', function (Blueprint $table) {
        $table->id();
        $table->string('lastName');
        $table->string('firstName');
        $table->string('email')->unique();
        $table->string('phone')->nullable();
        $table->string('username')->unique();
        $table->string('password');
        $table->timestamps();
    });
public function down()
    Schema::dropIfExists('members');
```

4. Setting up the Model

• To make the model for the admins, run the following command:

php artisan make:model Member

• Open the Member model in app/Models/Member.php and add the following:

```
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;
class Member extends Authenticatable
    use HasApiTokens, HasFactory, Notifiable;
    protected $guard = 'member';
    protected $table = "members";
    protected $hidden = [
        'password', 'remember_token',
```

5. Defining the Guard

 Open config/auth.php and add a new guard edit as follows:

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

6. Creating Member Controller

• In terminal, type:

php artisan make:controller MembersController

• The file will be created in "app/Http/Controllers" folder

```
<?php
    namespace App\Http\Controllers;
    use Illuminate\Http\Request;
6
    class UsersController extends Controller
```

7. Creating Register Function

```
class MembersController extends Controller
    private $rules = [
        'lastName' => 'required',
        'firstName' => 'required',
        'username' => 'required',
        'password' => 'required',
        'email' => 'required|email'
    private $messages = [
        'lastName.required' => 'Last name is empty',
        'firstName.required' => 'First name is empty',
        'username.required' => 'Username is empty',
        'password.required' => 'Password is empty',
        'email.email' => 'Invalid email format'
    ];
```

```
public function register(Request $req) {
    $validator = Validator::make($req->all(), $this->rules, $this->messages);
    if ($validator->fails()) {
        return BaseResponse::error(400, $validator->messages()->toJson());
    } else {
        try {
            $member = new Member();
            $member->lastName = $req->lastName;
            $member->firstName = $req->firstName;
            $member->email = $req->email;
            $member->phone = $req->phone;
            $member->username = $req->username;
            $member->password = bcrypt($req->password);
            $member->save();
            return BaseResponse::withData($member);
        } catch (\Exception $e) {
            return BaseResponse::error(500, $e->getMessage());
```

8. Creating Login Function

```
public function login(Request $req) {
    $data = ['username' => $req->username, 'password' => $req->password];
    if (auth('member')->attempt($data)) {
        $user = auth('member')->user(); // get member info
        $user->tokens()->delete(); // delete all previous tokens
        $token = $user->createToken('ApiToken')->plainTextToken;
        $authToken = explode('|', $token)[1]; // get token string only
        $data = [
            'id' => $user->id,
            'username' => $user->username,
            'fullName' => $user->name,
            'token' => $authToken
        return BaseResponse::withData($data);
    } else {
        return BaseResponse::error(1, "Wrong username or password");
```

9. Creating Profile Function

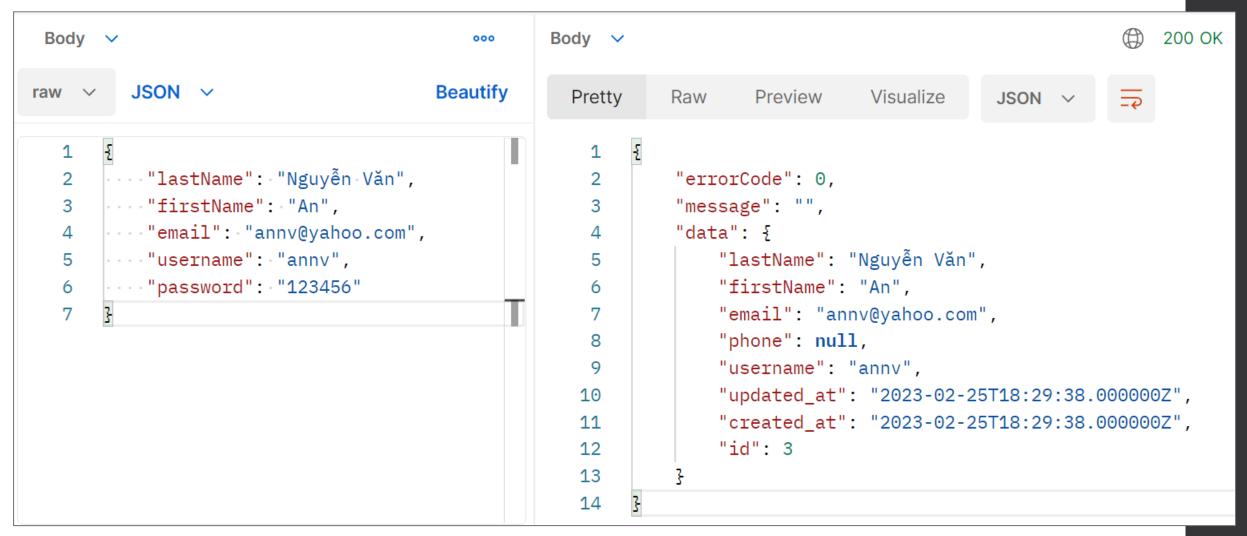
```
public function register(Request $req) { ...
public function login(Request $req) { ...
public function profile(Request $request) {
    $member = $request->user();
    return BaseResponse::withData($member);
```

10. Adding Routes

```
Route::get('login', function(){
    $response = ['errorCode' => 401, 'message' => 'Unauthenticated'];
    return response()->json($response, 401);
})->name('login');
Route::post('/login',[UsersController::class, 'login']);
Route::group(['middleware' => 'auth:api'], function () {
    Route::get('/employees/{id?}', [EmployeesController::class, 'index']);
    Route::post('/employees',[EmployeesController::class, 'create']);
    Route::put('/employees/{id}',[EmployeesController::class, 'update']);
    Route::delete('/employees/{id}',[EmployeesController::class, 'delete']);
});
Route::post('/member/register',[MembersController::class, 'register']);
Route::post('/member/login',[MembersController::class, 'login']);
Route::group(['prefix' => 'member', 'middleware' => 'auth:sanctum'], function () {
   Route::get('/profile', [MembersController::class, 'profile']);
});
```

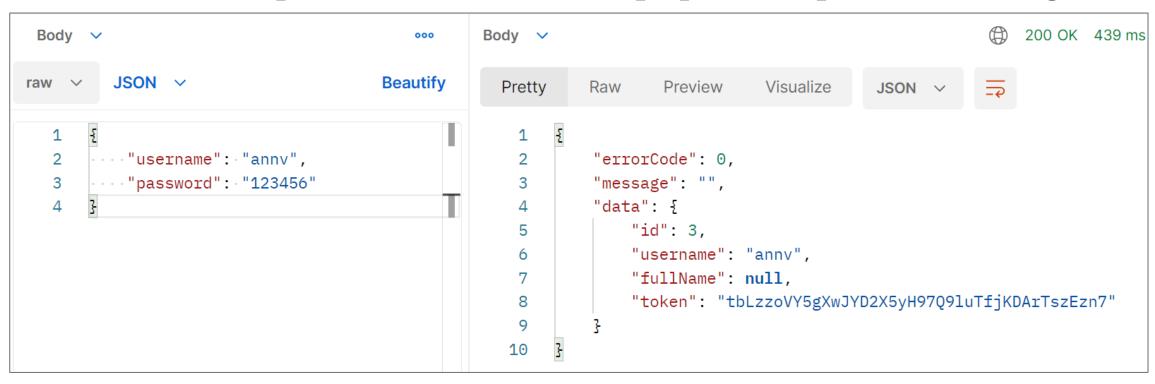
Testing Register API of Member

• POST: http://localhost/laravelapi/public/api/member/register



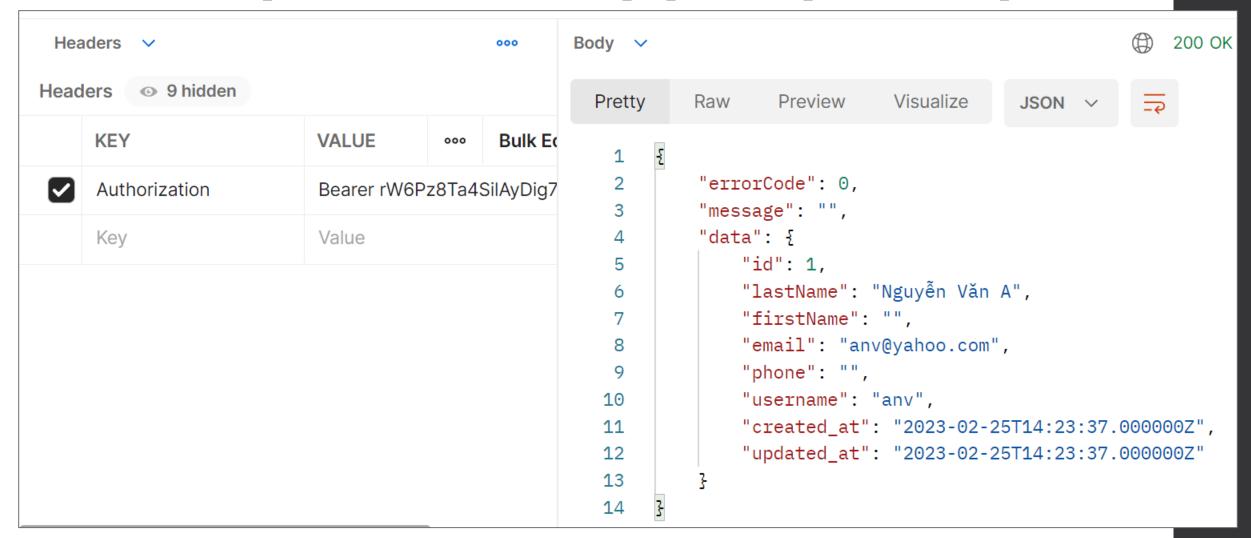
Testing Login API of Member

• POST: http://localhost/laravelapi/public/api/member/login



Testing protected API

• GET: http://localhost/laravelapi/public/api/member/profile



• GET: http://localhost/laravelapi/public/api/member/profile

