Laravel

geek wannabe

November 2025

1 Project Key Terms

1. **Sanctrum** A lightweight authentication system for APIs in Laravel.
2. **CORS** A security rule in browsers that blocks websites from making requests to different domains. Meaning that a server from port 5500 will be able to send request to a server in the ort 8000.
3. **fetch()** a JS method to send http request.
4. **Middleware** A filter that runs before a request reaches the controller.
5. **Controller** The "brain" that sits between the URL and the database.It receives the request, talks to the database (or other services), and sends back a JSON answer.
6. **Bearer Token** A way to send the token in the request header.
7. **PHPUnit** Automated testing for Laravel.
8. **Seeder** A script to insert test data into database.
9. **Migration** A version-controlled database schema.
10. **OpenAPI / Swagger** Interactive documentation for your API.

2 Project setup

Here is the project structure

```
    tp4/
|-- app/
|   |-- Http/
|   |   |-- Controllers/
|   |   |   |-- AuthController.php
|   |   |-- Middleware/
|   |   |   |-- CheckRole.php
|   |-- Models/
|       |-- User.php
|-- database/
|   |-- migrations/
|   |   |--2014_10_12_000000_create_users_table.php
|   |-- seeders/
|   |   |-- UserSeeder.php
|-- routes/
|   |-- api.php
|-- config/
|   |-- cors.php
|   |-- sanctum.php
|-- tests/
|   |-- Feature/
|   |   |-- AuthTest.php
|-- .env
|-- .env.example
|--README.md
```

2.1 Setting up Laravel

Initializes a fresh Laravel installation with proper security configuration.

```
composer create-project laravel/laravel tp4
cd tp4
php artisan key:generate
```

The configuration of the .env file is to Configure cross-origin authentication between Laravel backend (port 8000) and frontend (port 5500).

```
APP_URL=http://localhost:8000
SANCTUM_STATEFUL_DOMAINS=localhost:5500
SESSION_DOMAIN=localhost
```

2.2 Install and Publish Sanctum

```
composer require laravel/sanctum
php artisan vendor:publish --provider="Laravel\Sanctum\SanctumServiceProvider"
php artisan migrate
```

- Installs Sanctum package
- Publishes Sanctum migration and config files
- Runs migrations to create personal access tokens table

When Encountering the problem with the execution of the php artisan migrate . Here are the different steps for an ubuntu/debian user.

1. **Install apache2**

   ```
   sudo apt install apache2 -y
   sudo systemctl start apache2
   sudo systemctl enable apache2
   ```

2. Checking the php version

   ```
   php -v
   ```

3. Installation of php mysql

   ```
   sudo apt-get install php8.1-mysql
   ```

4. Enabling the extension pdo_mysql

   ```
   sudo phpenmod pdo_mysql
   ```

5. Restart PHP

   ```
   sudo service apache2 restart
   ```

6. VERIFY IT WORKS

   ```
   php -m | grep -i mysql

   # If everything is ok we will get
   mysqli
   mysqlnd
   pdo_mysql

   php -r "echo extension_loaded('pdo_mysql') ? 'OK' : 'ERROR';"
   # If extension is well loaded
   OK
   ```

7. Create Database

```
# Entering root password
mysql -u root -p

# Then run
CREATE DATABASE laravel;
EXIT;
```

8. Updating the .env

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

9. Clear Config Cache

```
php artisan config:clear
```

10. Run Migration

```
php artisan migrate

#And get something like this
2014_10_12_000000_create_users_table ..... 4,129ms DONE
2014_10_12_100000_create_password_reset_tokens_table ..... 1,424ms
 2019_08_19_000000_create_failed_jobs_table ..... 5,481ms DONE
2019_12_14_000001_create_personal_access_tokens_table .....7,601ms DONE
```

2.3 Enable CORS (config/cors.php)

```
'paths' => ['api/*', 'sanctum/csrf-cookie'],
'allowed_methods' => ['*'],
'allowed_origins' => ['http://localhost:5500'],
'allowed_headers' => ['*'],
'supports_credentials' => true,
```

- **Paths:** Protects API routes and CSRF cookie endpoint
- **Origins:** Allows requests from frontend on port 5500
- **Credentials:** Enables cookies/session authentication
- **Methods/Headers:** Permits all HTTP methods and headers

2.4 Adding role to Users Table

Migration: database/migrations/..._create_users_table.php

```
$table->enum('role', ['USER', 'ADMIN', 'SUPERADMIN'])->default('USER');
# And Run
php artisan migrate
```

2.5 User Model (app/Models/User.php)

Let's check the code in this part.

1. **Traits:** HasApiTokens (Sanctum), Notifiable
2. **Fields:** Basic auth fields + role for permissions
3. **Methods:** isAdmin() checks if user has admin role

2.6 Create Auth Controller

```
php artisan make:controller AuthController
```

Look for app/Http/Controllers/AuthController.php to check the code.

1. **Public:** POST /register, POST /login
2. **Protected** (requires auth token):
   - POST /logout - destroys token
   - GET /user - returns current user

Structure: Versioned API (v1) with token-based authentication guard.

2.7 Routes (routes/api.php)

```
php artisan make:controller AuthController
```

Look for app/Http/Controllers/AuthController.php to check the code.

1. **Public:** POST /register, POST /login
2. **Protected** (requires auth token):
   - POST /logout - destroys token
   - GET /user - returns current user

2.8 Role Middleware

```
php artisan make:middleware CheckRole
```

Look for app/Http/Middleware/CheckRole.php to check the code.

1. **Parameters:** Accepts roles (e.g., ADMIN, SUPERADMIN)
2. Logic: Checks if authenticated user's role matches allowed roles Checks if authenticated user's role matches allowed roles
3. **Response:** Returns 403 Forbidden if role doesn't match

Purpose: Protects routes by user roles.
Register in app/Http/Kernel.php : **Apply to routes like -¿middleware('role:ADMIN')**

2.9 Seed Test Users

```
php artisan make:seeder UserSeeder
```

Look for database/seeders/UserSeeder.php to check the code.

1. **Admin:** admin@shopcart.com with ADMIN role
2. **Client:** client@shopcart.com with USER role

Purpose: Populates database with sample users for testing roles.
The verification in the database :

```
mysql -u root -p
USE laravel;  -- or your DB name
DESCRIBE users;
```

2.10 PHPUnit Test

PHPUnit tests are automated checks that prove your Laravel API works correctly — without you having to click buttons or use Postman every time.

1. **No bugs:** Automatically tests every endpoint
2. **Fast feedback** Run php artisan test -¿ know in 2 seconds if something breaks
3. **Team safety:** When D adds a product, G's login still works
4. **Confidence:** You can deploy without fear
5. **Documentation:** Tests show how the API should behave

```
php artisan make:test AuthTest --unit
```

The test code is in the file tests/Feature/AuthTest.php

2.11 Launching the server

```
php artisan serve
```

3 Testing API

3.1 Terminal

3.1.1 Register (POST /api/v1/register)

```
    curl -X POST http://localhost:8000/api/v1/register \
  -H "Content-Type: application/json" \
  -d '{
    "name": "Test User",
    "email": "test@shopcart.com",
    "password": "password123",
    "password_confirmation": "password123"
  }'
```

If ok Here is the Result :

```
    {
  "message": "Registered",
  "token": "1|random123...",
  "user": {
    "id": 3,
    "name": "Test User",
    "email": "test@shopcart.com",
    "role": "USER"
  }
}
```

3.1.2 Login (POST /api/v1/login)

```
curl -X POST http://localhost:8000/api/v1/login \
  -H "Content-Type: application/json" \
  -d '{
    "email": "admin@shopcart.com",
    "password": "password123"
  }'
```

Click **"Send"** .
Expected Response (200):

```
{
  "message": "Login successful",
  "token": "3|abc123def456ghi789...",
  "user": {
    "id": 1,
    "name": "Admin User",
    "email": "admin@shopcart.com",
    "role": "ADMIN"
```

```
  }
}
```

3.1.3 Get Current User (GET /api/v1/user) – Protected

```
curl -X GET http://localhost:8000/api/v1/user \
  -H "Authorization: Bearer 5|XsWfJjOHY6yJ1TZYLDC1jJK1jEq1etyI5cqJjdrv537a43c8"
  -H "Content-Type: application/json"
```

Click **"Send"** .
Expected Response (200):

```
{
"id":1,
"name":"AdminUser",
"email":"admin@shopcart.com",
"email_verified_at":null,
"created_at":"2025-11-09T19:35:19.000000Z",
"updated_at":"2025-11-09T19:35:19.000000Z",
"role":"ADMIN"
}
```

3.1.4 Logout (POST /api/v1/logout) – Protected

Follow : **Headers**

```
curl -X POST http://localhost:8000/api/v1/logout \
  -H "Authorization: Bearer 3|AbCdEfGhIjKlMnOpQrStUvWxYz1234567890" \
  -H "Content-Type: application/json"
```

Click **"Send"** .
Expected Response (200):

```
{
  "message": "Logged out"
}
```

3.2 Postman

1. Open Postman
2. Click "New" -¿ "Collection"
3. Name it: TP4 - Auth API

Now Add Request such as :

3.2.1 Register (POST /api/v1/register)

- **Method** POST
- **FieldValueMethodPOST** URLhttp://localhost:8000/api/v1/register

Follow : **Body =- raw =- JSON**

```
    {
  "name": "Alice",
  "email": "alice@shopcart.com",
  "password": "password123",
  "password_confirmation": "password123"
}
```

Click **"Send"** .
Expected Response (201):

```
 {
  "message": "Registered",
  "token": "1|abc123...",
  "user": {
    "id": 3,
    "name": "Alice",
    "email": "alice@shopcart.com",
    "role": "USER"
  }
}
```

3.2.2 Login (POST /api/v1/login)

- **Method** POST
- **FieldValueMethodPOST** URLhttp://localhost:8000/api/v1/login

Follow : **Body =- raw =- JSON**

```
 {
  "email": "admin@shopcart.com",
  "password": "password123"
}
```

Click **"Send"** .
Expected Response (200):

```
{
    "message": "Login successful",
    "token": "4|sqrbKyunR6nNh8s7gXnPFYXgTMvwSUvuYN2hQX7w56428d82",
    "user": {
        "id": 1,
```

```
        "name": "Admin User",
        "email": "admin@shopcart.com",
        "role": "ADMIN"
    }
}
```

3.2.3 Get Current User (GET /api/v1/user) – Protected

- **Method** GET
- **FieldValueMethodPOST** http://localhost:8000/api/v1/user

Follow : **Headers**

Key	Value
Authorization	‘Bearer 2
Content–Type	application/json

Click **"Send"** .
Expected Response (200):

```
{
  "id": 1,
  "name": "Admin User",
  "email": "admin@shopcart.com",
  "role": "ADMIN"
}
```

3.2.4 Logout (POST /api/v1/logout) – Protected

- **Method** POST
- **FieldValueMethodPOST** http://localhost:8000/api/v1/logout

Follow : **Headers**

Key	Value
Authorization	‘Bearer 2

Click **"Send"** .
Expected Response (200):

```
{
  "message": "Logged out"
}
```