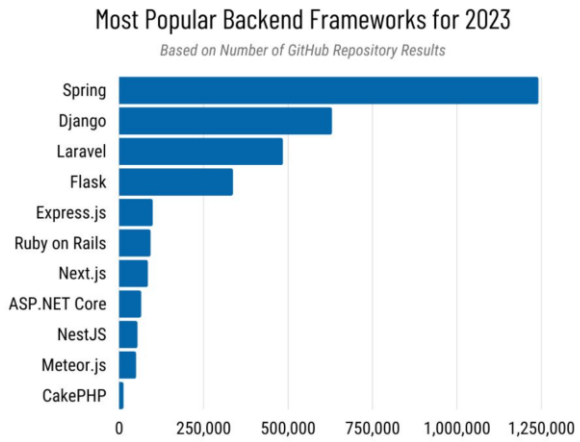
Day-1: Building News Platform with Laravel API - CRUD for Categories

Prepared by: Rana M. Fakeeh

About Laravel:



Application overview:

Our goal is to create a News Platform's CRUD APIs with the following entities:

- ***Category***: To categorize news articles.
- ***NewsArticle***: To store news articles.
- ***Image***: To associate image album with news articles.
- ***User***: To manage user accounts.

Training program:

Day-1	Setup. CRUD for Category entity.
Day-2	CRUD for NewsArticle, Image entities. Upload image files to server. One-to-Many and Many-to-Many associations.
Day-3	Recap. Filtering, Sorting, and Pagination of list results.
Day-4	Laravel Passport authentication (OAuth2 token-based) Register, Login, Logout, and Profile APIs for User entity. Public vs. Protected endpoints.
Day-5	Localization. Deployment (Docker).

Learning objectives:

- Laravel API project setup and MySQL configuration.
- Create Category table.
- Create RESTful API for CRUD operations.
- API testing (Postman client).
- Hands-on.

Steps:

1- **Tools:**
   - XAMP (PHP, Apache and MySQL).
   - Composer.
   - VS Code.
   - Postman.

2- **Project setup:**
   - Create a root directory named ***labs***.
   - Open VS Code.
   - File -> Open Folder (select folder ***labs***)
   - Terminal -> New Terminal.
   - Create Laravel project named: ***news-app-day1***
   - composer create-project laravel/laravel news-app-day1
   - Change directory to project folder
   - cd news-app-day1
   - Test the server running
   - php artisan serve

3- **Database config:**
   - Run XAMP control panel.
   - Start Apache server
   - Start MySQL server
   - Navigate to http://localhost/phpmyadmin
   - Create a new database named ***newsapp*** (no dashes or spaces allowed in identifier)
   - Update ***.env*** file with database name and save.
   - Migrate changes and built-in database schema.
   - php artisan migrate
   - Check database changes in MySQL (phpMyAdmin)

4- **Category Module (Migration -> Model -> Factory -> Seeder -> Resource -> Controller -> Route)**
   - Create Category model, migration, and factory.
   - php artisan make:model Category -mf
   - The flag -mf creates factory, and migrations files in addition to model file.
   - ***Migration***
   - Open ***database/migrations/timestamp_create_categories_table.php*** file.
   - Add fields ***name***, and ***description***.
   - $table->string('name');
   - $table->text('description')->nullable();
   - ***Model***
   - Open ***app/Models/Category.php*** file.
   - Add fields ***name***, and ***description***.
   - protected $fillable = [
   - 'name',
   - 'description'
   -];

- ***Factory***
- Open ***database/factories/CategoryFactory.php*** file.
- Create fake values for fields ***name***, and ***description***.

```
'name' => ucwords(fake()->words(rand(1, 3), true)),
'description' => fake()->text(),
```

- ***Seeder***
- Open ***database/seeders/DatabaseSeeder.php*** file.
- Populate database with 10 dummy categories

```
Category::factory(10)->create();
```

- Migrate changes to database schema.

```
php artisan migrate
```

- Run seeding

```
php artisan db:seed
```

- Check database changes in MySQL (phpMyAdmin)
- ***Resource***
- Create Category resource.

```
php artisan make:resource CategoryResource
```

- Open ***app/Http/Resources/CategoryResource.php*** file.
- Return only the category's ***id***, and ***name***.

```
[
    "id" => $this->resource->id,
    "name" => $this->resource->name,
];
```

- Create *Category* collection.

```
php artisan make:resource CategoryCollection --collection
```

- Open ***app/Http/Resources/CategoryCollection.php*** file.
- Return the objects as defined in the resource.

```
[
    'count' => $this->collection->count(),
    'categories' => $this->collection,
];
```

- ***Controller***
- Create Category controller.

```
php artisan make:controller CategoryController --api
```

- Open ***app/Http/Controllers/CategoryController.php*** file.
- Add the code snippet in (Appendix-I)
- ***Route***
- Create routes for Category's CRUD operations.
- Open ***routes/api.php*** file.
- Add either:

```
Route::resource('category', CategoryController::class)
```

- Or

```
Route::get('category', [CategoryController::class, 'index'])->name('category.index');
```

5- Testing API with Postman

- Apache server is started.
- MySQL server started.
- Migrate changes to database schema.
- php artisan migrate
- Test the server running
- php artisan serve
- Open Postman
- ***http://localhost:8000/api/category/****
- For create, use ***Body*** -> ***form-data***
- For update, use ***Body*** -> ***x-www-form-urlencoded***

Appendix-I:

CategoryController.php

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Category;
use App\Http\Resources\CategoryResource;
use App\Http\Resources\CategoryCollection;
use Illuminate\Support\Facades\Validator;
use Exception;


class CategoryController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        $categories = Category::all();

        return response()->json([
            "status" => "success",
            "error" => false,
            "data" => new CategoryCollection($categories),
        ],200);
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        $validator = Validator::make($request->all(), [
            "name" => "required|min:3|unique:categories,name",
        ]);

        if($validator->fails()) {
            return response()->json([
```

```
                "status" => "fail",
                "error" => true,
                "validation_errors" => $validator->errors()
            ]);
        }

        try {
            $category = Category::create([
                "name" => $request->name,
                "description" => $request->description
            ]);
            return response()->json([
                "status" => "success",
                "error" => false,
                "message" => "Success! category created.",
                "data" => new CategoryResource($category),
            ], 201);
        }
        catch(Exception $exception) {
            return response()->json([
                "status" => "fail",
                "error" => true,
                "message" => $exception->getMessage(),
            ], 404);
        }
    }

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

        if($category) {
            return response()->json([
                "status" => "success",
                "error" => false,
                "data" => new CategoryResource($category)
            ], 200);
        }
        return response()->json([
            "status" => "fail",
            "error" => true,
```

```
            "message" => "Failed! no category found."
        ], 404);
    }

    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $id)
    {
        $category = Category::find($id);

        if($category) {

            $validator = Validator::make($request->all(), [
                "name" => "required|min:3|unique:categories,name",
            ]);

            if($validator->fails()) {
                return response()->json([
                    "status" => "fail",
                    "error" => true,
                    "validation_errors" => $validator->errors()
                ]);
            }

            $category['name'] = $request->name;

            // if it has description
            if($request->description) {
                $category['description'] = $request->description;
            }

            $category->save();
            return response()->json([
                "status" => "success",
                "error" => false,
                "message" => "Success! category updated.",
                "data" => new CategoryResource($category)
            ], 200);
        }
        return response()->json([
            "status" => "fail",
            "error" => true,
```

```
            "message" => "Failed no category found."
        ], 404);
    }

    /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function destroy($id)
    {
        $category = Category::find($id);

        if($category) {
            $category->delete();
            return response()->json([
                "status" => "success",
                "error" => false,
                "message" => "Success! category deleted."
            ], 200);
        }
        return response()->json([
            "status" => "fail",
            "error" => true,
            "message" => "Failed no category found."
        ], 404);
    }
}
```