

# Demo Consumo API Laravel

Autor: Pedro Arriondo ([parriondo@gmail.com](mailto:parriondo@gmail.com))

## CONTENIDO

<b>Consideraciones</b>	<b>1</b>
<b>Servicio</b>	<b>2</b>
Crear el proyecto	2
Crear Modelos, Controller y Migraciones	2
Seeder	2
Seteo de los valores de la API a consultar y el json de respaldo	2
Archivo JSON de respaldo	2
DTO	2
Ruta web	2
<b>API</b>	<b>3</b>
Creación de recurso y collection	3
Manejo de errores	3
<b>Pruebas</b>	<b>4</b>
Creación	4
Aclaración	4
Pruebas Unitarias realizadas	4
Pruebas de Features realizadas	5

## Consideraciones

En el archivo .env.apichallenge se encuentran las siguientes definiciones:

API\_URL=<https://api.publicapis.org/entries>

JSON\_FILE=entries.json

El archivo JSON de respaldo debe ir en **storage/app**

## Servicio

### Crear el proyecto

```
composer create-project laravel/laravel apichallenge "^10.0"
```

### Crear Modelos, Controller y Migraciones

```
php artisan make:model Category -mcr  
php artisan make:model Entity -mcr
```

- Modificar migraciones

### Seeder

- Crear el seeder

```
php artisan make:seeder CategorySeeder
```
- Modificar archivo para sembrar las categorías 'Animals' y 'Security'
- Ejecutar seeder

```
php artisan db:seed --class=CategorySeeder
```

### Seteo de los valores de la API a consultar y el json de respaldo

**.env:**

```
API_URL=https://api.publicapis.org/entries  
JSON_FILE=entries.json
```

### Archivo JSON de respaldo

- Guardar el json file en **storage/app**

### DTO

Se utiliza un objeto de transferencia de datos para mapear la estructura de un objeto **json** recibido a objeto **entity**.

### Ruta web

.../entries

Ejemplo: <http://apichallenge.test/entries>

# API

## ***Creación de recurso y collection***

Objetivo: formatear la respuesta a la estructura solicitada

```
php artisan make:resource EntityResource  
php artisan make:resource EntityCollection
```

## ***Manejo de errores***

Para el caso puntual de no existencia de una categoría buscada, se utiliza una excepción personalizada.

```
php artisan make:exception CategoryNotFoundException
```

## Pruebas

\*A fines prácticos del ejercicio, se puede usar la misma BD (MySQL) para pruebas ó, descomentar las líneas siguientes en el archivo **phpunit.xml**:

```
<env name="DB_CONNECTION" value="sqlite"/>
<env name="DB_DATABASE" value=":memory:"/>
```

### Creación

```
php artisan make:test EntityServiceTest --unit
php artisan make:test EntityServiceTest
php artisan make:test EntityApiTest
```

- Además, se crean las factories para **entity** y **category**.

### Aclaración

Si al ejecutar una prueba, se recibe el mensaje: *"Your XML configuration validates against a deprecated schema. Migrate your XML configuration using "--migrate-configuration!"*, correr el comando:

```
php artisan test --migrate-configuration
```

### Pruebas Unitarias realizadas

Se realizan pruebas unitarias al servicio carga de datos de la BD.

#### Conexión a la API externa

- Prueba de éxito cuando `fetchApiData()` funciona

```
php artisan test --filter test_fetch_api_data_success
```

```
PASS Tests\Unit\EntityServiceTest
✓ fetch api data success

Tests:    1 passed (1 assertions)
Duration: 1.90s
```

- Prueba de éxito cuando `fetchApiData()` falla

```
php artisan test --filter test_fetch_api_data_failure
```

```
PASS Tests\Unit\EntityServiceTest
✓ fetch api data failure

Tests:    1 passed (1 assertions)
Duration: 0.82s
```

### Recuperación de datos desde el archivo backup

- Prueba de éxito cuando `getBackupData()` funciona

```
php artisan test --filter test_get_backup_data_success
```

```
PASS Tests\Unit\EntityServiceTest
✓ get backup data success

Tests: 1 passed (2 assertions)
Duration: 11.84s
```

- Prueba de éxito cuando getBackupData() falla  
php artisan test --filter test\_get\_backup\_data\_failure

```
PASS Tests\Unit\EntityServiceTest
✓ get backup data failure

Tests: 1 passed (1 assertions)
Duration: 0.78s
```

## Almacenamiento de los datos

- Prueba de éxito cuando storeData() funciona  
php artisan test --filter test\_store\_data\_success

```
PASS Tests\Unit\EntityServiceTest
✓ store data success

Tests: 1 passed (4 assertions)
Duration: 0.79s
```

- Prueba de éxito cuando storeData() recibe datos inválidos  
php artisan test --filter test\_store\_data\_invalid\_format

```
PASS Tests\Unit\EntityServiceTest
✓ store data invalid format

Tests: 1 passed (1 assertions)
Duration: 0.67s
```

- Prueba de éxito en storeData() cuando no hay categorías en BD  
php artisan test --filter test\_store\_data\_no\_categories

```
PASS Tests\Unit\EntityServiceTest
✓ store data no categories

Tests: 1 passed (1 assertions)
Duration: 0.87s
```

## Pruebas de Features realizadas

Se realizan dos sets de pruebas: Servicio y API

### Servicio

- Prueba exitosa de la llamada a **/entries** cuando la API responde y todo el proceso es exitoso  
php artisan test --filter test\_sync\_entries\_success\_from\_api

```
PASS Tests\Feature\EntityServiceTest
✓ sync entries success from api

Tests: 1 passed (3 assertions)
Duration: 0.45s
```

- Prueba exitosa de la llamada a **/entries** cuando la API falla, se usa el backup y todo el proceso es exitoso

```
php artisan test --filter test_sync_entries_success_from_backup
```

```
PASS Tests\Feature\EntityServiceTest
✓ sync entries success from backup

Tests: 1 passed (3 assertions)
Duration: 0.47s
```

- Prueba exitosa de la llamada a **/entries** cuando la API falla y falla el backup. El proceso no es exitoso

```
php artisan test --filter test_sync_entries_failure_api_and_backup
```

```
PASS Tests\Feature\EntityServiceTest
✓ sync entries failure api and backup

Tests: 1 passed (1 assertions)
Duration: 0.53s
```

- Prueba exitosa de la llamada a **/entries** cuando se obtienen datos pero falla el almacenamiento (en la prueba, la API retorna datos pero estos son incorrectos). El proceso no es exitoso

```
php artisan test --filter test_sync_entries_store_data_failure
```

```
PASS Tests\Feature\EntityServiceTest
✓ sync entries store data failure

Tests: 1 passed (1 assertions)
Duration: 0.45s
```

## API

Aquí se hacen pruebas para cuando se consulta la API con una categoría existente y luego con una no existente

```
php artisan test tests/Feature/EntityApiTest.php
```

```
PASS Tests\Feature\EntityApiTest
✓ entity endpoint returns data when category exists
✓ entity endpoint returns 404 when category does not exist

Tests: 2 passed (19 assertions)
Duration: 1.89s
```