Challenge Laravel Dropea

Autor: Pedro Arriondo (parriondo@gmail.com)

CONTENIDO

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 categorias `'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
```

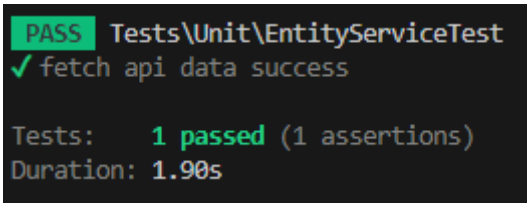
Pruebas Unitarias realizadas

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

Conexión a la API externa

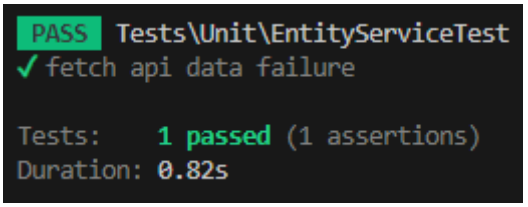
- Prueba de éxito cuando fetchApiData() funciona

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



- Prueba de éxito cuando fetchApiData() falla

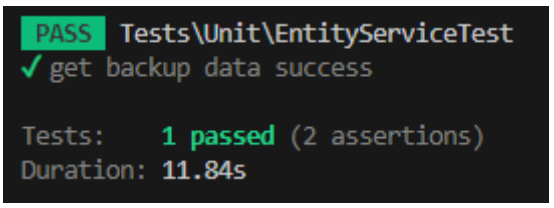
```
php artisan test --filter test_fetch_api_data_failure
```



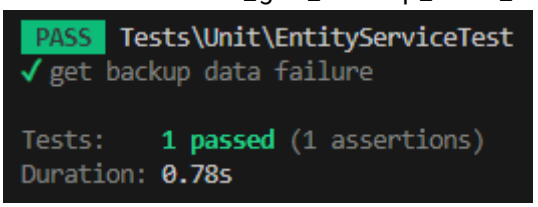
Recuperación de datos desde el archivo backup

- Prueba de éxito cuando getBackupData() funciona

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

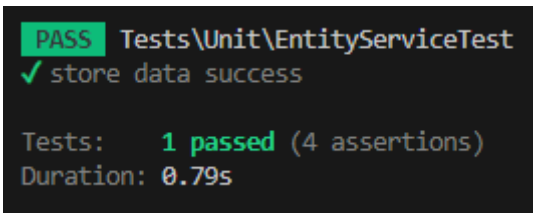


- Prueba de éxito cuando getBackupData() falla
  `php artisan test --filter test_get_backup_data_failure`

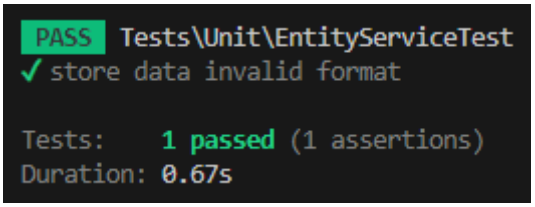


Almacenamiento de los datos

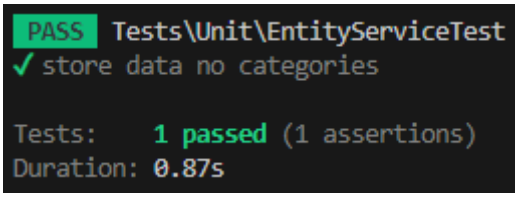
- Prueba de éxito cuando storeData() funciona
  `php artisan test --filter test_store_data_success`



- Prueba de éxito cuando storeData() recibe datos inválidos
  `php artisan test --filter test_store_data_invalid_format`



- Prueba de éxito en storeData() cuando no hay categorías en BD
  `php artisan test --filter test_store_data_no_categories`

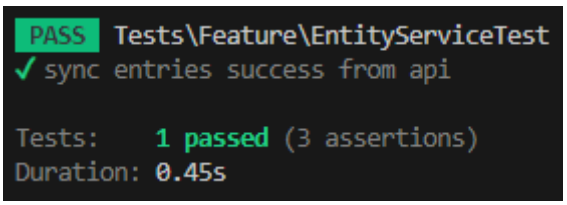


Pruebas de Features realizadas

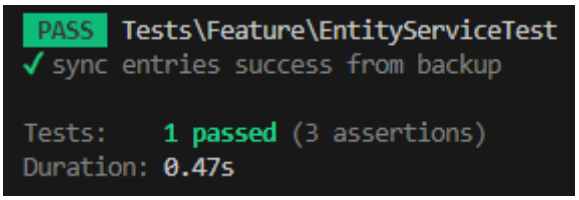
Se realizan dos sets de pruebas: Servicio y API

Servicio

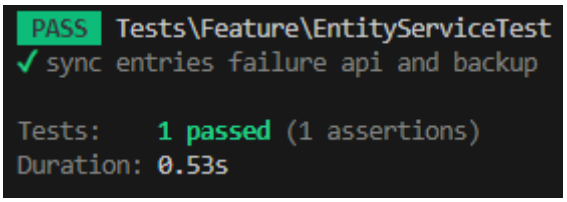
- Prueba exitosa de la llamada a **/entries** cuando la API responde y todo el proceso es exitoso
  `php artisan test --filter test_sync_entries_success_from_api`



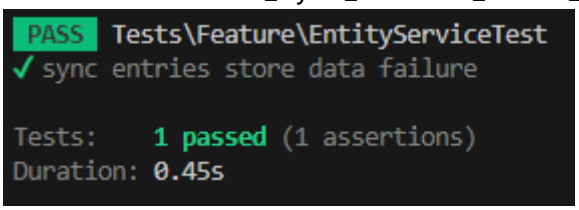
- Prueba exitosa de la llamada a **/entries** cuando la API falla, se usa el backup y todo el proceso es exitoso
  `php artisan test --filter test_sync_entries_success_from_backup`



- 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`



- 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`



API

Aqui se hacen pruebas para cuando se consulta la API con una categoria existente y luego con una no existente

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