Pruebas de servicios web con rest assured y cucumber

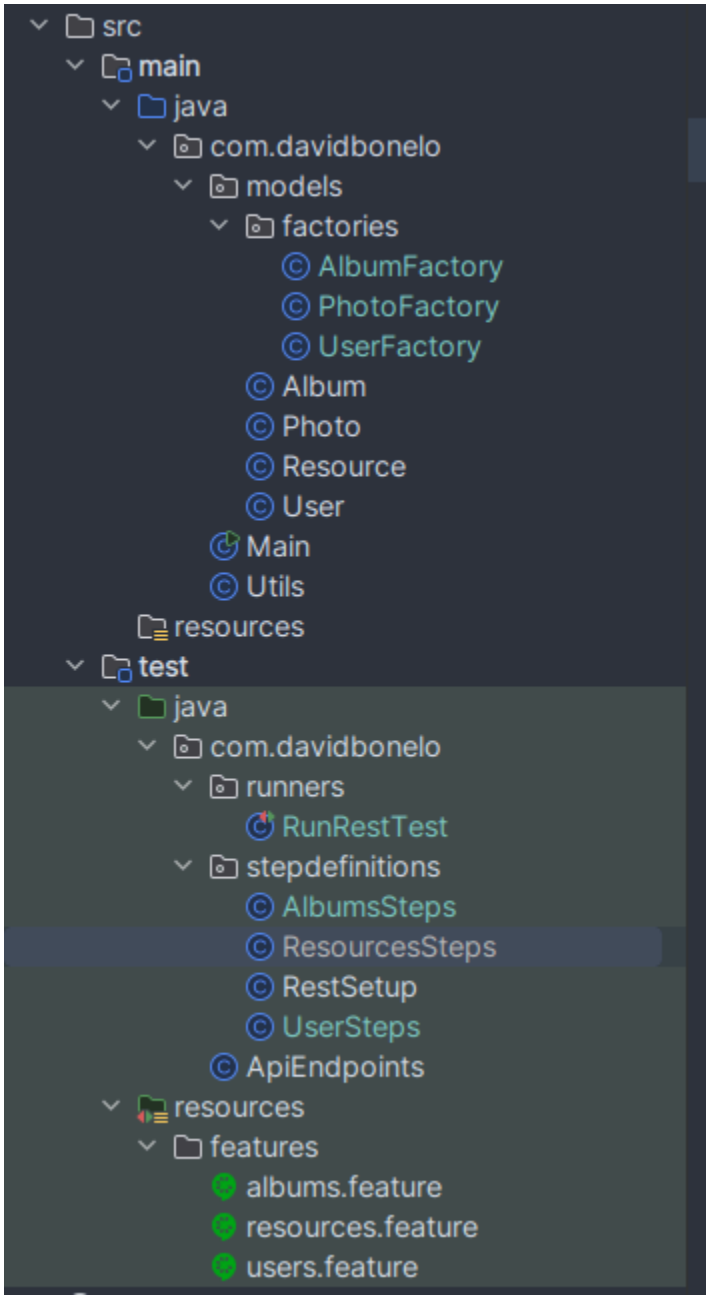
Por: Jesús David Bonelo Cuellar

> En el repositorio se encuentran adjuntos los reportes tipo web interactivos generados por cucumber de forma local.

Los endpoint de las APIs utilizados para realizar las pruebas son los siguientes:

- https://reqres.in/api/users/:id **GET**
- https://reqres.in/api/unknown/:id **GET**
- https://reqres.in/api/login **POST**
- https://jsonplaceholder.typicode.com/albums **POST** **DELETE**
- https://jsonplaceholder.typicode.com/users/:id/albums **GET**
- https://jsonplaceholder.typicode.com/photos **POST** **DELETE**

Estructura del proyecto:



Realicé modelos para definir la estructura de los objetos que se envían y reciben desde los servicios, también utilicé el patrón Factory para separar las distintas estrategias de creación de los objetos (ej: con faker o desde una jsonString)

Configuración

Reestructuré el archivo build.gradle siguiendo la documentación oficial de cucumber, para ejecutarlo con Junit 5 en lugar del 4, y generar reportes locales.
`(build.gradle` adjunto en el repositorio)

El runner requirió cierta modificación

```
@Suite
@IncludeEngines("cucumber")
@SelectPackages("features")
@ConfigurationParameter(key = PLUGIN_PROPERTY_NAME, value = "pretty")
@ConfigurationParameter(key = GLUE_PROPERTY_NAME, value = "com.davidbonelo.stepdefinitions")
//@IncludeTags("Test1")
public class RunRestTest {
}
```

En el README.md del proyecto agregué instrucciones para generar los reportes de forma local.

Para la configuración de las url base utilicé la funcionalidad de hooks de cucumber aplicándolos con las etiquetas
https://cucumber.io/docs/cucumber/api/?lang=java#hooks

```java
public class RestSetup {

    @BeforeAll
    public static void setup() {
        // log all the data from the sent requests and received responses
        RestAssured.filters(new RequestLoggingFilter(), new ResponseLoggingFilter());
    }

    @Before("@ReqRes")
    public static void setReqResUrl() {
        RestAssured.baseURI = BASE_URL;
        RestAssured.basePath = BASE_PATH;
    }

    @Before("@jsonPlaceholder")
    public static void setJsonPlaceholderUrl() {
        RestAssured.baseURI = PLACEHOLDER_BASE;
        RestAssured.basePath = "";
    }
}
```

El método de setup se ejecuta una vez antes de todos los escenarios y configura filtros de rest assured para imprimir en la consola toda la información de las peticiones enviadas y las respuestas recibidas; De esta forma me permite activar o desactivar todos los logs en una única parte del código

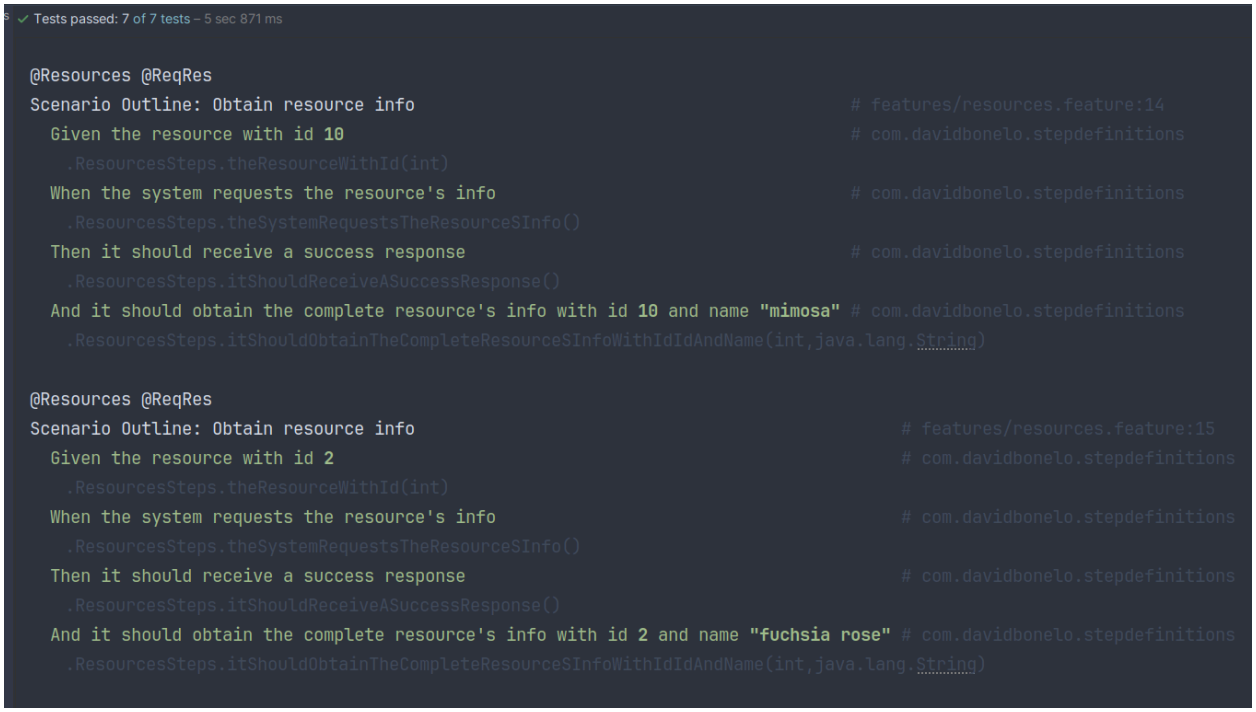
Definí las URIs de los endpoints como constantes en una clase dedicada.

```java
public class ApiEndpoints {
      public static final String BASE_URL = "https://reqres.in";
      public static final String BASE_PATH = "/api";
      public static final String LOGIN_ENDPOINT = "/login";
      public static final String USERS_ENDPOINT = "/users";
      public static final String RESOURCES_ENDPOINT = "/unknown";
      public static final String PLACEHOLDER_BASE =
"https://jsonplaceholder.typicode.com";
      public static final String ALBUMS_ENDPOINT = "/albums";
      public static final String PHOTOS_ENDPOINT = "/photos";
}
```

Escenarios de prueba con evidencias

```
@Resources @ReqRes
Feature: Resources management via REST API
  As an integration system I
  WANT to manage the information about the resources in the service
  SO THAT I can keep the information up to date and give it to the
users

  Scenario Outline: Obtain resource info
      Given the resource with id <id>
      When the system requests the resource's info
      Then it should receive a success response
      And it should obtain the complete resource's info with id <id>
and name "<name>"
      Examples:
      | id | name      |
      | 10 | mimosa    |
      | 2  | fuchsia rose |
```




```
@Users @ReqRes
Feature: Users

  Scenario: Obtain user profile info
      Given A user is logged in
      When he requests its profile data
      Then he should see his correct information
```

Este caso de prueba lo utilicé para practicar simulando autenticación recibiendo un token y enviándolo en otra petición como parte del header

```
@Given("A user is logged in")
public void aUserIsLoggedIn() {
      //...
      String response = request.post(LOGIN_ENDPOINT).asString();
      token = JsonPath.from(response).getString("token");
}

@When("he requests its profile data")
public void heRequestsItsProfileData() {
      RequestSpecification request =
```

```
given().contentType(ContentType.JSON);
      request.header("Authorization", "Bearer " + token);
      response = request.when().get(USERS_ENDPOINT + "/" +
user.getId());
}
```

```
@Users @ReqRes
Scenario: Obtain user profile info          # features/users.feature:4
  Given A user is logged in                 # com.davidbonelo.stepdefinitions.UserSteps.aUserIsLoggedIn()
  When he requests its profile data         # com.davidbonelo.stepdefinitions.UserSteps.heRequestsItsProfileData()
  Then he should see his correct information # com.davidbonelo.stepdefinitions.UserSteps.heShouldSeeHisCorrectInformation()
```

```
@Albums @jsonPlaceholder
Feature: Albums

  Background: User's albums
      Given A list of albums from a user

  Scenario: Create a new empty album
      When the user creates a new album
      Then he should get a successful response
      And he should get the id assigned to that album

  Scenario: Delete a user album
      When the user deletes one of the albums
      Then he should get a successful response
      And he shouldn't see that album anymore in his list

  Scenario: Add a photo to a album
      When the user adds a photo to one album
      Then he should get a successful response
      And he should get the id assigned to the new photo

  Scenario: Delete a photo
      When the user removes a photo from one album
      Then he should get a successful response
```

Las aserciones para validar que un elemento fue creado correctamentes las realizo utilizando las `ValidatableResponse` de RestAssured y los matchers de Hamcrest

```java
@And("he should get the id assigned to the new photo")
public void heShouldGetTheIdAssignedToTheNewPhoto() {
      response.then()
                  .body("albumId",
equalTo(newPhoto.getAlbum().getId()))
                  .body("title", equalTo(newPhoto.getTitle()))
                  .body("url", equalTo(newPhoto.getUrl()))
                  .body("thumbnailUrl",
equalTo(newPhoto.getThumbnailUrl()))
                  .body("id", notNullValue(Integer.class));
}
```

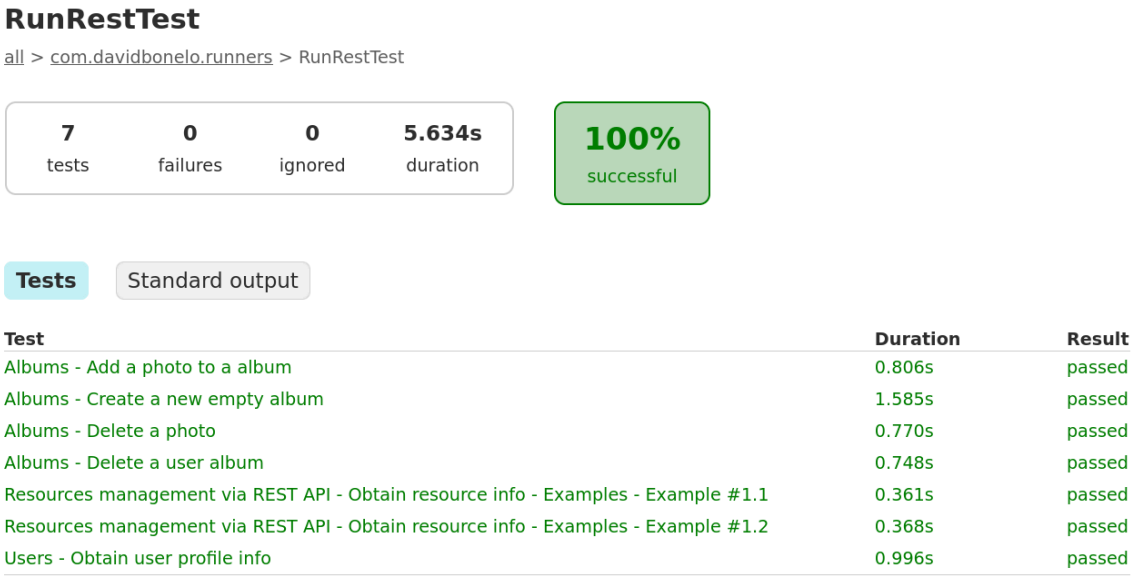
```
@Albums @jsonPlaceholder
Scenario: Create a new empty album                # features/albums.feature:7
  Given A list of albums from a user              # com.davidbonelo.stepdefinitions.AlbumsSteps.aListOfAlbumsFromAUser()
  When the user creates a new album               # com.davidbonelo.stepdefinitions.AlbumsSteps.theUserCreatesANewAlbum()
  Then he should get a successful response        # com.davidbonelo.stepdefinitions.AlbumsSteps
    .heShouldGetASuccessfulResponse()
  And he should get the id assigned to that album # com.davidbonelo.stepdefinitions.AlbumsSteps
    .heShouldGetTheIdAssignedToThatAlbum()

@Albums @jsonPlaceholder
Scenario: Delete a user album                      # features/albums.feature:12
  Given A list of albums from a user               # com.davidbonelo.stepdefinitions.AlbumsSteps.aListOfAlbumsFromAUser()
  When the user deletes one of the albums          # com.davidbonelo.stepdefinitions.AlbumsSteps
    .theUserDeletesOneOfTheAlbums()
  Then he should get a successful response         # com.davidbonelo.stepdefinitions.AlbumsSteps
    .heShouldGetASuccessfulResponse()
  And he shouldn't see that album anymore in his list # com.davidbonelo.stepdefinitions.AlbumsSteps
    .heShouldnTSeeThatAlbumAnymoreInHisList()

@Albums @jsonPlaceholder
Scenario: Add a photo to a album                  # features/albums.feature:17
  Given A list of albums from a user              # com.davidbonelo.stepdefinitions.AlbumsSteps.aListOfAlbumsFromAUser()
  When the user adds a photo to one album         # com.davidbonelo.stepdefinitions.AlbumsSteps
    .theUserAddsAPhotoToOneAlbum()
  Then he should get a successful response        # com.davidbonelo.stepdefinitions.AlbumsSteps
    .heShouldGetASuccessfulResponse()
  And he should get the id assigned to the new photo # com.davidbonelo.stepdefinitions.AlbumsSteps
    .heShouldGetTheIdAssignedToTheNewPhoto()
```

```
@Albums @jsonPlaceholder
Scenario: Delete a photo                    # features/albums.feature:22
  Given A list of albums from a user        # com.davidbonelo.stepdefinitions.AlbumsSteps.aListOfAlbumsFromAUser()
  When the user removes a photo from one album # com.davidbonelo.stepdefinitions.AlbumsSteps.theUserRemovesAPhotoFromOneAlbum()
  Then he should get a successful response  # com.davidbonelo.stepdefinitions.AlbumsSteps.heShouldGetASuccessfulResponse()
```

Reporte Cucumber-Junit5



RunRestTest

all > com.davidbonelo.runners > RunRestTest

7	0	0	5.634s
tests	failures	ignored	duration

100% successful

Tests | Standard output

Test	Duration	Result
Albums - Add a photo to a album	0.806s	passed
Albums - Create a new empty album	1.585s	passed
Albums - Delete a photo	0.770s	passed
Albums - Delete a user album	0.748s	passed
Resources management via REST API - Obtain resource info - Examples - Example #1.1	0.361s	passed
Resources management via REST API - Obtain resource info - Examples - Example #1.2	0.368s	passed
Users - Obtain user profile info	0.996s	passed

En la pestaña `Standard output` se encuentran los logs de toda la información de las peticiones y respuestas de las APIs

```
Request method: POST
Request URI:    https://jsonplaceholder.typicode.com/albums
Proxy:                  <none>
Request params: <none>
Query params:   <none>
Form params:    <none>
Path params:    <none>
Headers:                Accept=*/*
                        Content-Type=application/json
Cookies:                <none>
Multiparts:             <none>
Body:
{
    "title": "Bea Lowe",
    "userId": 3
}
```

```
}
HTTP/1.1 201 Created
Date: Tue, 07 May 2024 17:24:13 GMT
Content-Type: application/json; charset=utf-8
Content-Length: 53
Connection: keep-alive
Report-To: {"group":"heroku-nel","max_age":3600,"endpoints":[{
Reporting-Endpoints: heroku-nel=https://nel.heroku.com/reports
Nel: {"report_to":"heroku-nel","max_age":3600,"success_fractio
X-Powered-By: Express
X-Ratelimit-Limit: 1000
X-Ratelimit-Remaining: 999
X-Ratelimit-Reset: 1715102713
Vary: Origin, X-HTTP-Method-Override, Accept-Encoding
Access-Control-Allow-Credentials: true
Cache-Control: no-cache
Pragma: no-cache
Expires: -1
Access-Control-Expose-Headers: Location
Location: https://jsonplaceholder.typicode.com/albums/101
X-Content-Type-Options: nosniff
Etag: W/"35-Yrp4P2a8bLJ2ZZOySBGjW2+y08U"
Via: 1.1 vegur
CF-Cache-Status: DYNAMIC
Server: cloudflare
CF-RAY: 8802e6ffaae3a55e-MIA
alt-svc: h3=":443"; ma=86400

{
    "title": "Bea Lowe",
    "userId": 3,
    "id": 101
}
```