

EJERCICIO 2 API

16 DE OCTUBRE DEL 2022

ELABORADO POR:

MELISSA GERALDINE JIMÉNEZ CALLISAYA

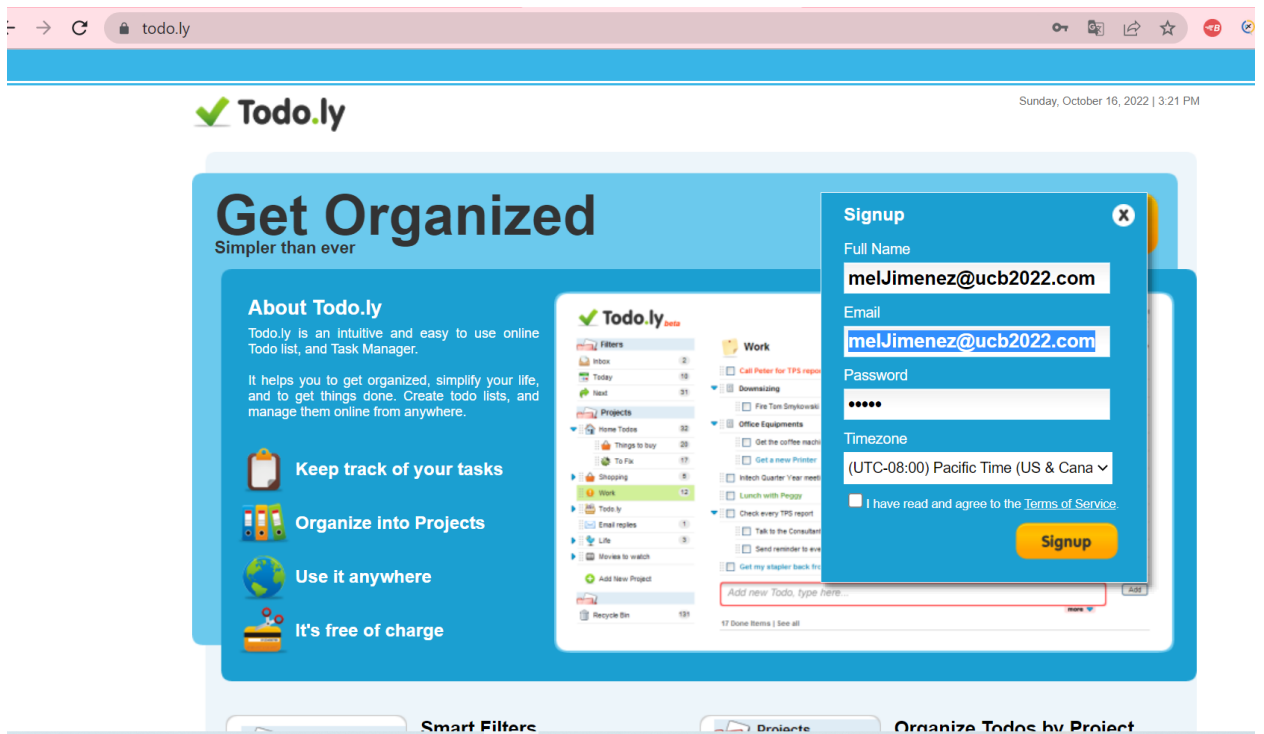
INSTRUCCIONES

Automatización del API de <http://todo.ly/apiwiki/?items>, realizando la creación, lectura, actualización y eliminación utilizando RESTASSURED.

PASOS EJECUTADOS

1. Conocer la aplicación

Creamos una cuenta



user: melJimenez@ucb2022.com

password: 12345

Revisamos la documentación

POST. A DELETE request is also accepted for methods that destroy data. API Methods that require a particular HTTP method will return an error if you do not make your request with the correct method. HTTP Response Codes are meaningful.

The API is a RESTful resource

REST (Representational State Transfer) is a programming concept in which you build your web application to only process basic CRUD (Create Read Update and Delete) operations. The API presently supports the following data formats: XML, JSON.

To try the API for the first time we suggest to use [curl](#)

Here are some examples to use curl:

- Check if you are authenticated: `curl https://todo.ly/api/authentication/IsAuthenticated.xml`
- Get your UserObject in XML format: `curl -u username:password https://todo.ly/api/user.xml`
- Get all your projects in JSON format: `curl -u username:password https://todo.ly/api/projects.json`

Objects of the API

Todo.ly API has 4 main Objects.

- [UserObject](#) that stores data about the user itself
- [ProjectObject](#) holding information about the projects
- [FilterObject](#) that is very similar to the ProjectObject
- [ItemObject](#) that represents the Tasks in Todo.ly

UserObject Class

- Fields
 - AddItemMoreExpanded
 - DefaultProjectId
 - EditDueDateMoreExpanded
 - Email

APIError Class

- Fields
 - ErrorCode
 - ErrorMessage

ItemObject Class

- Fields
 - Checked
 - Children
 - Collapsed
 - Content

Projects API Methods

- Delete User
- Get All Projects
- Create New Project
- Get Project By Id
- Update Project By Id
- Delete Project By Id
- Get Items of a Project
- Get Done Items of a Project

Filters API Methods

- Get List of Filters
- Get Filter By Id
- Get Items of a Filter
- Get Done Items of a Filter

Items API Methods

- Get All Items
- Create new Item
- Get Item By Id
- Update Item By Id
- Delete Item By Id
- Get Root Item By child Id
- Get Done Root Item By child Id

Icons API Methods

- Get Icons By Id

De la documentación se puede deducir que es necesario un proyecto para asociar el ítem que se vaya a crear, leer, actualizar y eliminar.

a. Creación de proyecto

URL:

<https://todo.ly/api/projects.format>

MÉTODO:

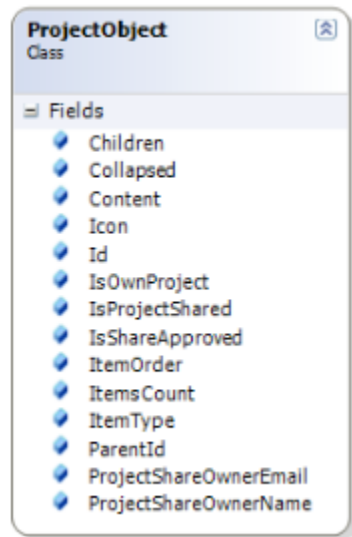
POST

PARÁMETROS:

Se requiere un ProjectObject como parámetro de entrada, aunque no se requieren todos los campos solo: Content

JSON:

```
{  
  "Content" : "Melissa",  
  "Icon" : "2"  
}
```

**b. Creación de ítem****URL:**

<https://todo.ly/api/items.format>

MÉTODO:

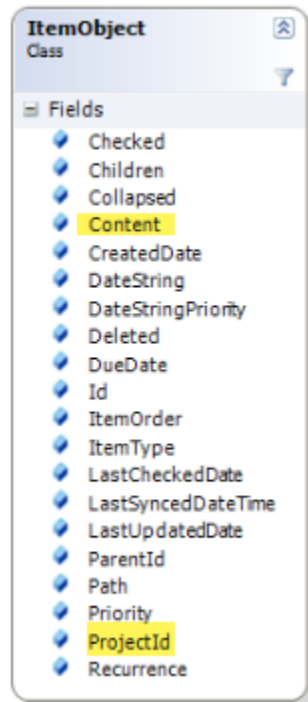
POST

PARÁMETROS:

Se requiere un ItemObject como parámetro de entrada, aunque no se requieren todos los campos solo: Content

JSON:

```
{  
  "Content" : "GoToTheVetForPetsVaccines",  
  "ProjectId" : "" <- Acá irá el ID del proyecto  
}
```



c. Lectura de ítem

URL:

[https://todo.ly/api/items/\[id\].format](https://todo.ly/api/items/[id].format)

MÉTODO:

GET

PARÁMETROS:

Ninguno

d. Actualización de ítem

URL:

[https://todo.ly/api/items/\[id\].format](https://todo.ly/api/items/[id].format)

MÉTODO:

PUT

PARÁMETROS:

Se requiere un ItemObject como parámetro de entrada, aunque no se requieren todos los campos, los campos enviados serán los actualizados.

JSON:

```
{  
    "Content" : "vaccinesForMyPets"  
    "ProjectId" : "" <- Acá irá el ID del proyecto  
}
```

e. Eliminación de ítem

URL:

[https://todo.ly/api/items/\[id\].format](https://todo.ly/api/items/[id].format)

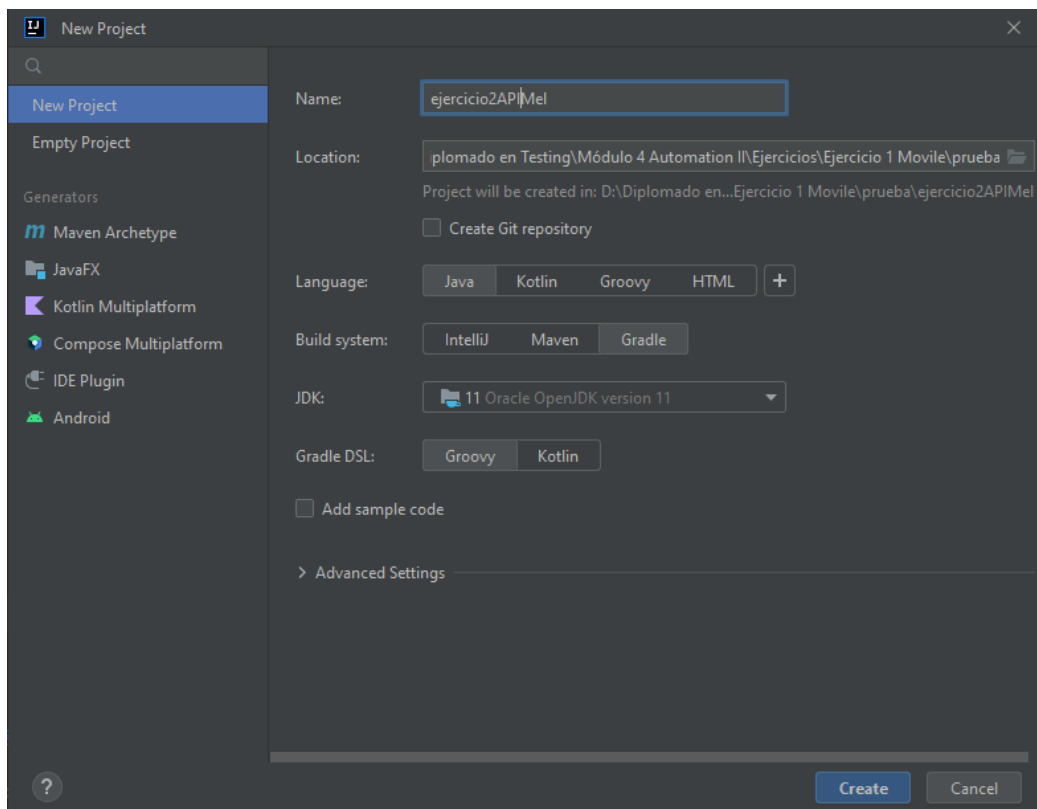
MÉTODO:

DELETE

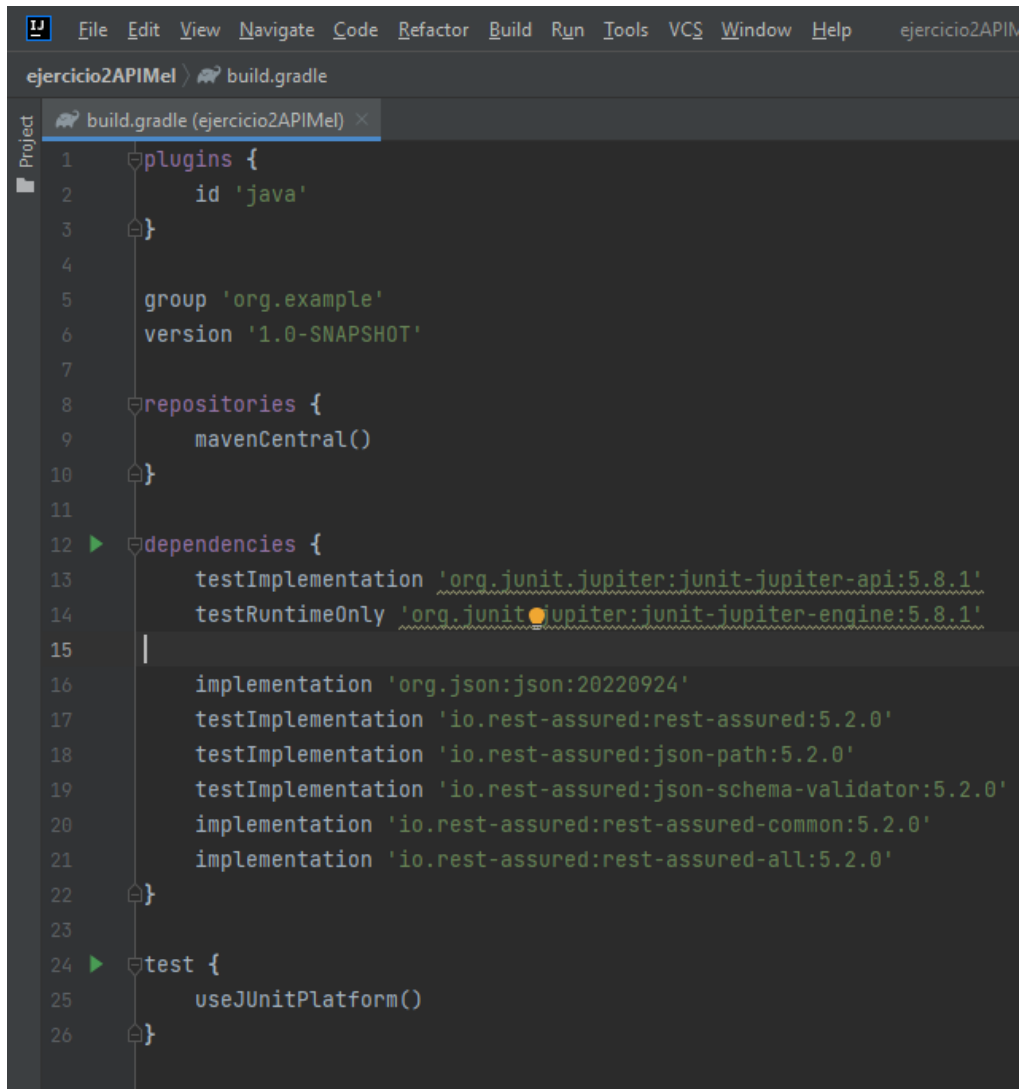
PARÁMETROS:

Ninguno

2. Creación del proyecto. Crear un nuevo proyecto IntelliJ IDEA.



3. **Dependencias.** Se adicionan las dependencias



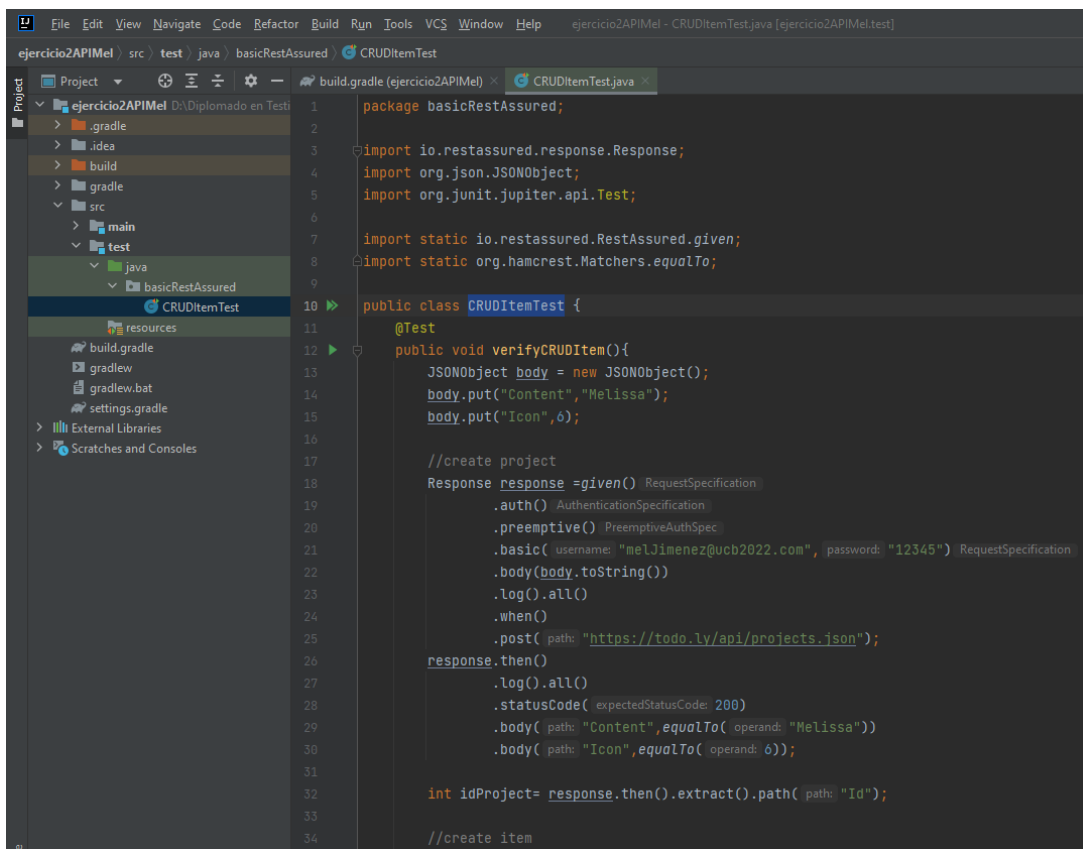
The screenshot shows an IDE window with the title 'ejercicio2APIMel'. The menu bar includes File, Edit, View, Navigate, Code, Refactor, Build, Run, Tools, VCS, Window, and Help. The project name 'ejercicio2APIMel' is visible in the top bar. The main editor displays the 'build.gradle' file for the project. The code is as follows:

```
1 plugins {  
2     id 'java'  
3 }  
4  
5 group 'org.example'  
6 version '1.0-SNAPSHOT'  
7  
8 repositories {  
9     mavenCentral()  
10 }  
11  
12 dependencies {  
13     testImplementation 'org.junit.jupiter:junit-jupiter-api:5.8.1'  
14     testRuntimeOnly 'org.junit.jupiter:junit-jupiter-engine:5.8.1'  
15  
16     implementation 'org.json:json:20220924'  
17     testImplementation 'io.rest-assured:rest-assured:5.2.0'  
18     testImplementation 'io.rest-assured:json-path:5.2.0'  
19     testImplementation 'io.rest-assured:json-schema-validator:5.2.0'  
20     implementation 'io.rest-assured:rest-assured-common:5.2.0'  
21     implementation 'io.rest-assured:rest-assured-all:5.2.0'  
22 }  
23  
24 test {  
25     useJUnitPlatform()  
26 }
```

4. Probar consumo de la API desde una clase simple

CRUDItemTest

- a. Create project
- b. Create Item
- c. Get Item
- d. Update Item
- e. Delete Item



```
package basicRestAssured;

import io.restassured.response.Response;
import org.json.JSONObject;
import org.junit.jupiter.api.Test;

import static io.restassured.RestAssured.given;
import static org.hamcrest.Matchers.equalTo;

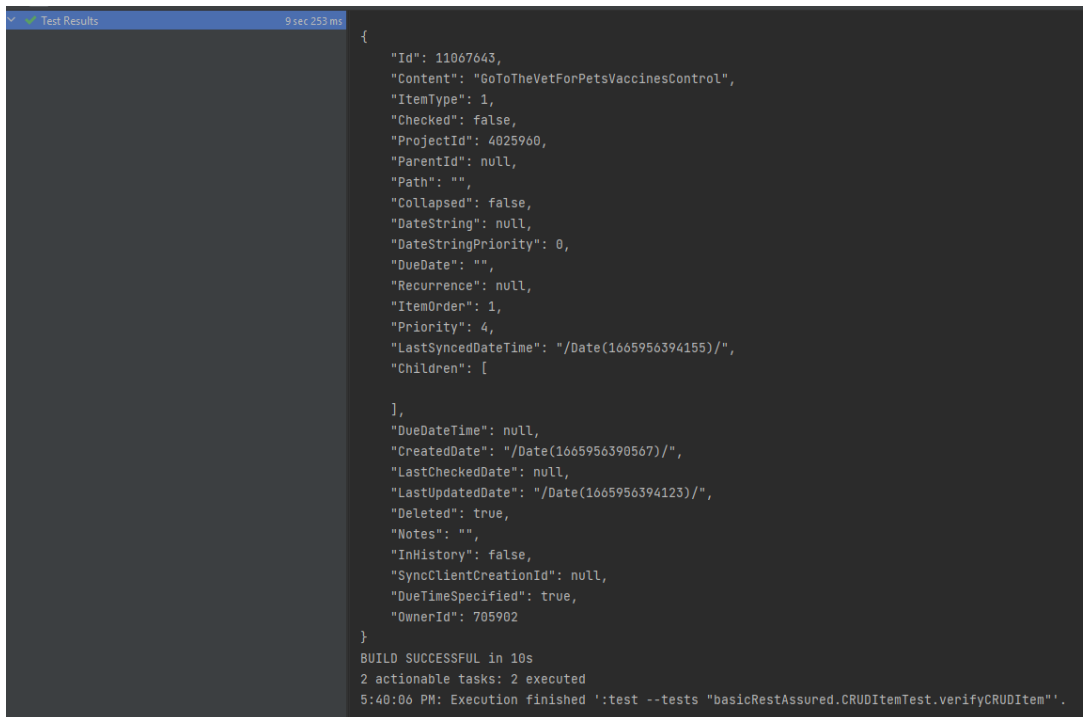
public class CRUDItemTest {

    @Test
    public void verifyCRUDItem(){
        JSONObject body = new JSONObject();
        body.put("Content", "Melissa");
        body.put("Icon", 6);

        //create project
        Response response = given() RequestSpecification
            .auth() AuthenticationSpecification
            .preemptive() PreemptiveAuthSpec
            .basic( username: "melJimenez@ucb2022.com", password: "12345") RequestSpecification
            .body(body.toString())
            .log().all()
            .when()
            .post( path: "https://todo.ly/api/projects.json");
        response.then()
            .log().all()
            .statusCode( expectedStatusCode: 200)
            .body( path: "Content", equalTo( operand: "Melissa"))
            .body( path: "Icon", equalTo( operand: 6));

        int idProject= response.then().extract().path( path: "Id");

        //create item
    }
}
```



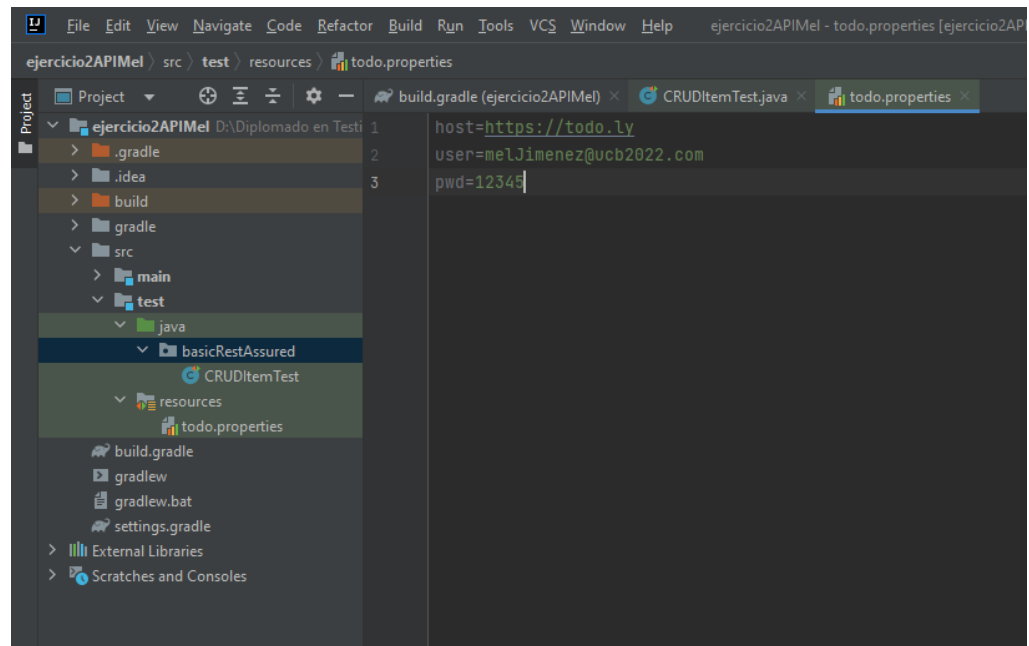
```
{
  "Id": 11067643,
  "Content": "GoToTheVetForPetsVaccinesControl",
  "ItemType": 1,
  "Checked": false,
  "ProjectId": 4025960,
  "ParentId": null,
  "Path": "",
  "Collapsed": false,
  "DateString": null,
  "DateStringPriority": 0,
  "DueDate": "",
  "Recurrence": null,
  "ItemOrder": 1,
  "Priority": 4,
  "LastSyncedDateTime": "/Date(1665956394155)/",
  "Children": [

  ],
  "DueDateTime": null,
  "CreatedDate": "/Date(1665956390567)/",
  "LastCheckedDate": null,
  "LastUpdatedDate": "/Date(1665956394123)/",
  "Deleted": true,
  "Notes": "",
  "InHistory": false,
  "SyncClientCreationId": null,
  "DueTimeSpecified": true,
  "OwnerId": 705902
}

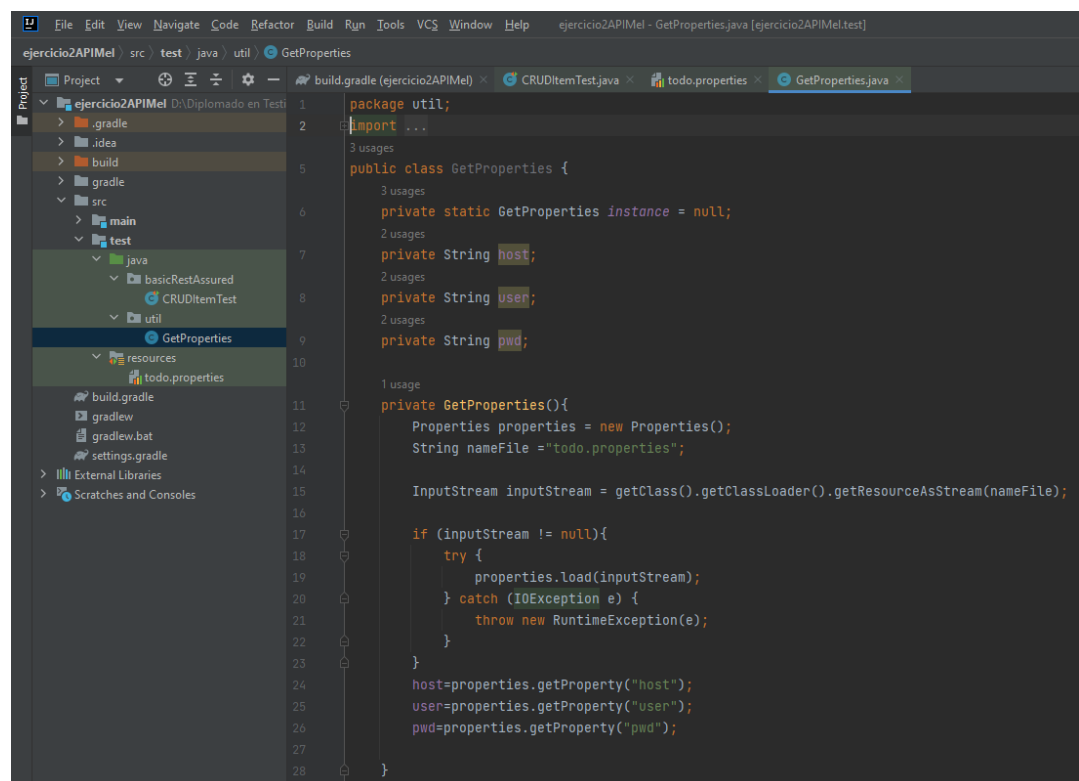
BUILD SUCCESSFUL in 10s
2 actionable tasks: 2 executed
5:40:06 PM: Execution finished ':test --tests "basicRestAssured.CRUDItemTest.verifyCRUDItem".'
```

5. **Limpiar del proyecto.** Una vez verificado el consumo de API limpiamos el proyecto.

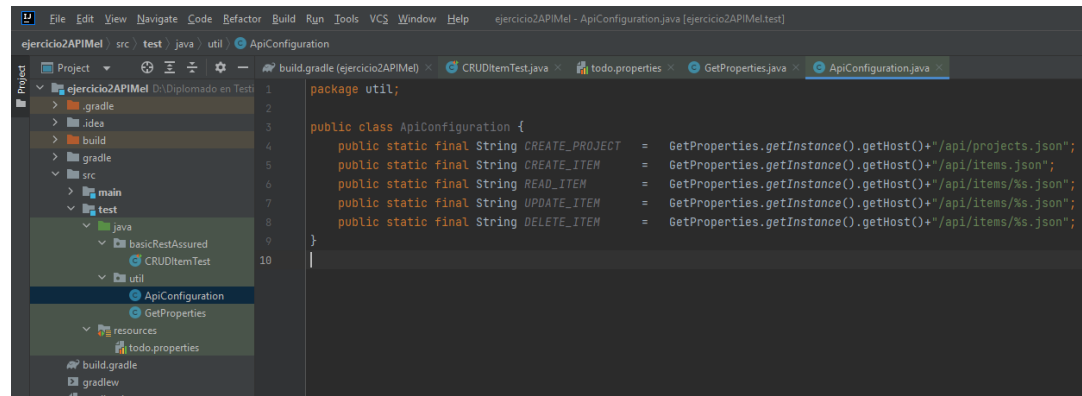
- a. Creamos un archivo properties con los datos generales de hot, password y user.



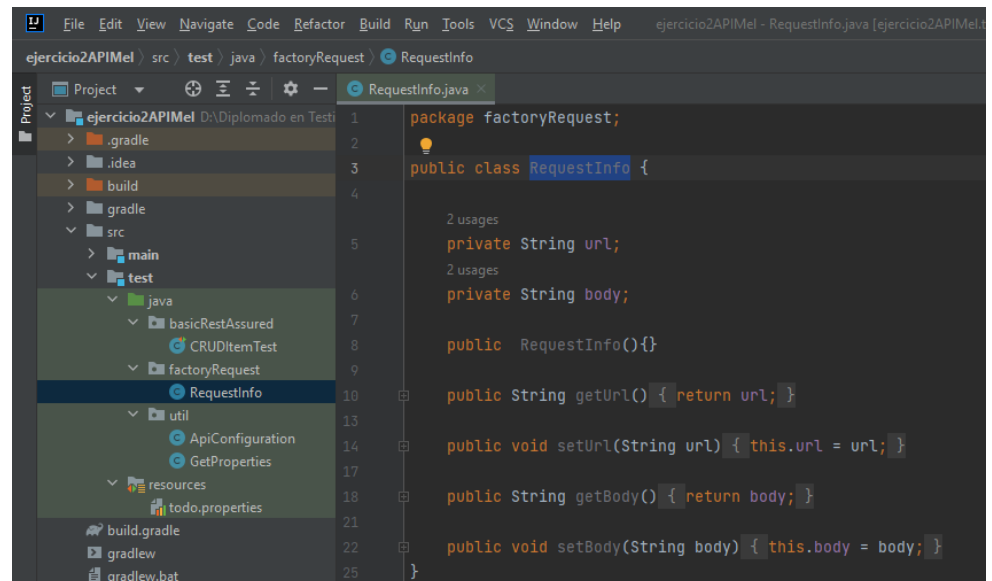
- b. Creamos una clase para recuperar las properties



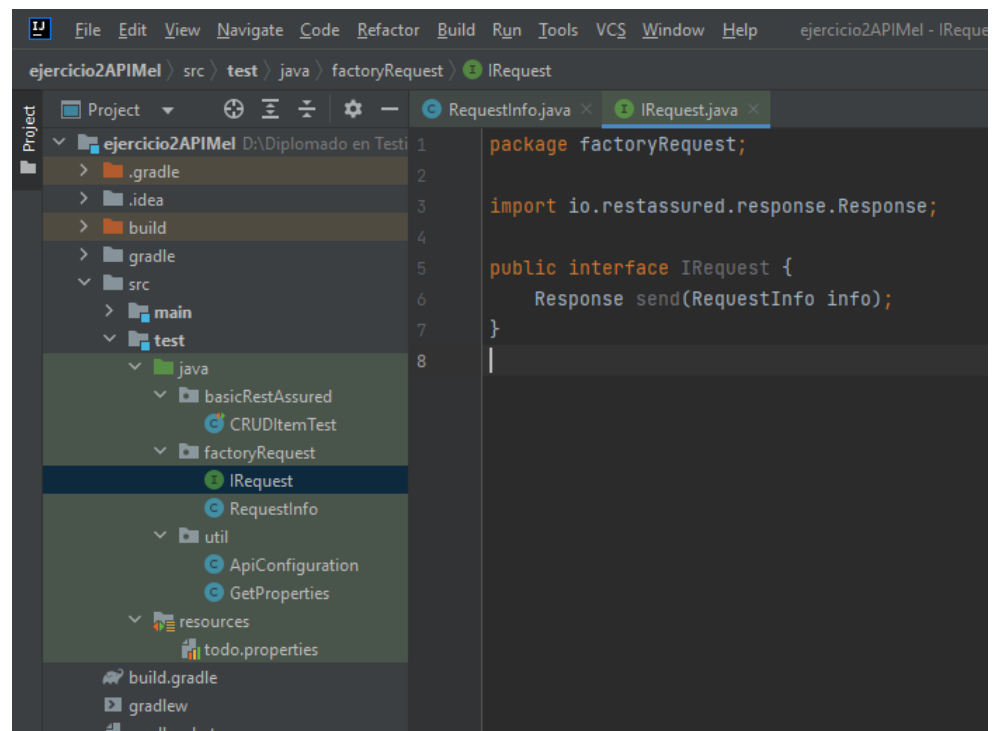
- c. Creamos una clase con las configuraciones del API



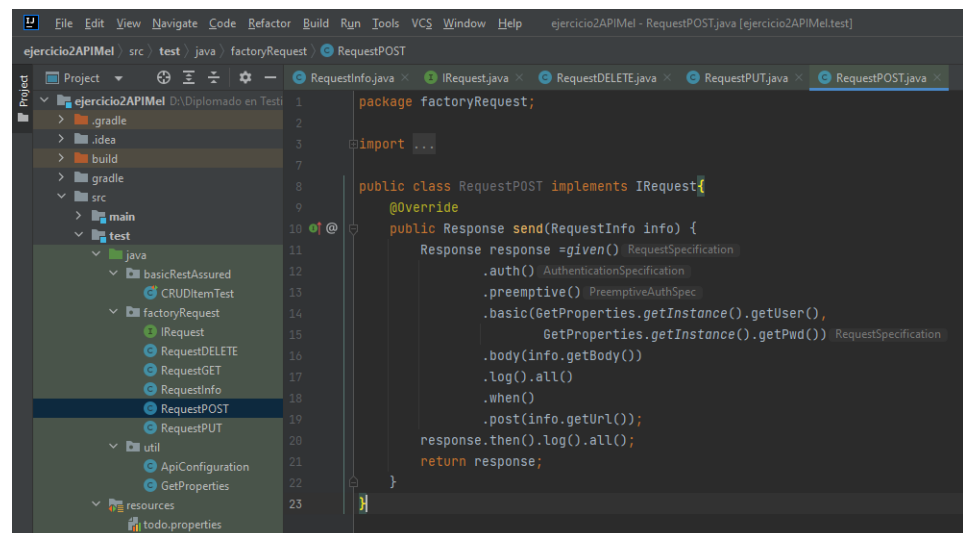
- d. Creamos el paquete factoryRequest con las clases para crear los request
 - i. Requestinfo - Clase con los atributos de un request

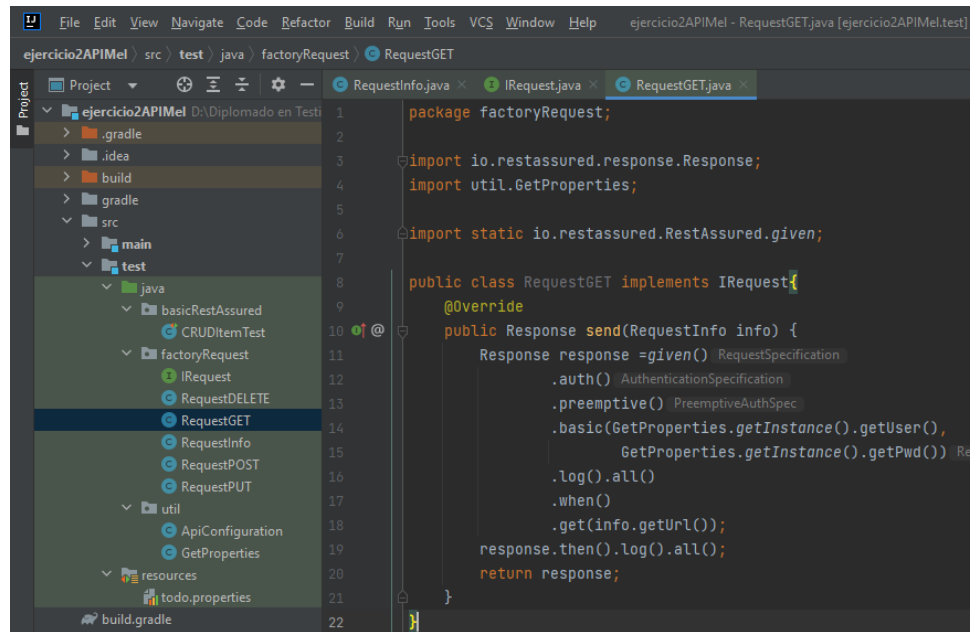


- ii. Interface para el factory



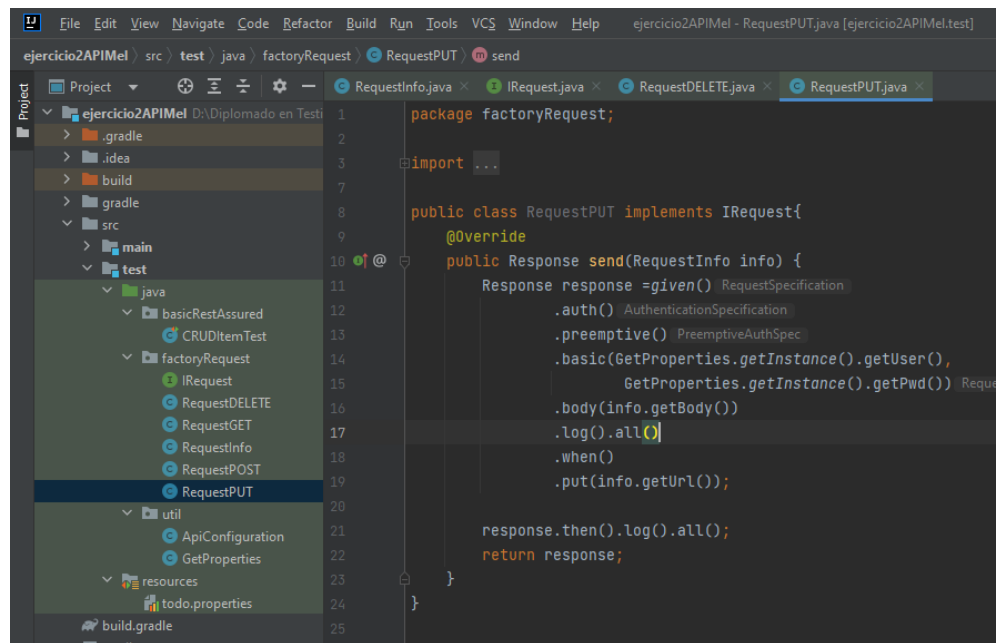
iii. POST, GET. PUT, DELETE





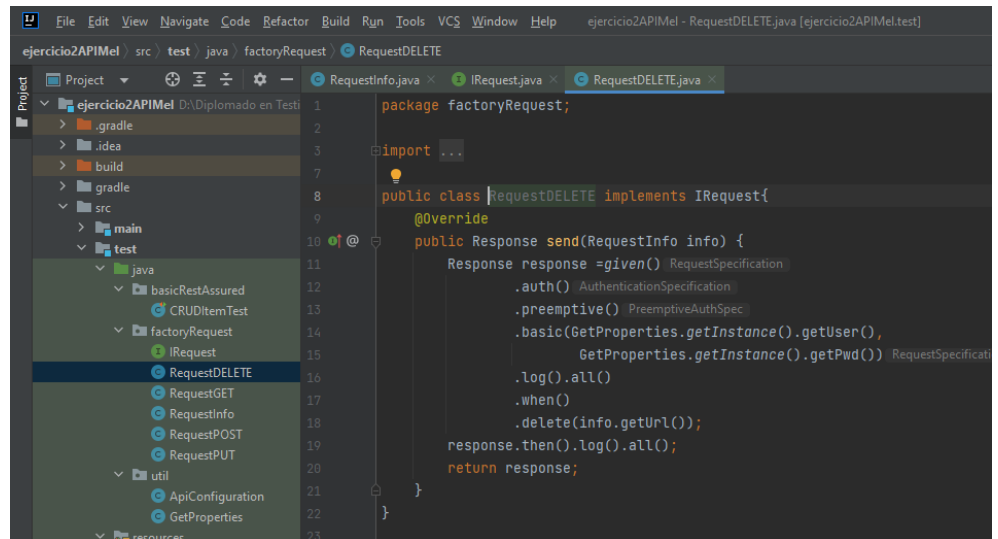
The screenshot shows an IDE with the project 'ejercicio2APIMel' open. The left sidebar displays the project structure, with the 'test' directory expanded to show the 'factoryRequest' package. The 'RequestGET.java' file is selected. The main editor displays the code for 'RequestGET.java', which implements the 'IRequest' interface. The code includes imports for 'io.restassured.response.Response' and 'util.GetProperties', and a static import for 'io.restassured.RestAssured.given()'. The 'send' method is annotated with '@Override' and uses 'given()' to configure the request, followed by 'auth()', 'preemptive()', 'basic()', 'log().all()', and 'when()' to execute the request. The response is then logged and returned.

```
1 package factoryRequest;
2
3 import io.restassured.response.Response;
4 import util.GetProperties;
5
6 import static io.restassured.RestAssured.given;
7
8 public class RequestGET implements IRequest{
9     @Override
10    public Response send(RequestInfo info) {
11        Response response =given() RequestSpecification
12            .auth() AuthenticationSpecification
13            .preemptive() PreemptiveAuthSpec
14            .basic(GetProperties.getInstance().getUser(),
15                GetProperties.getInstance().getPwd()) Re
16            .log().all()
17            .when()
18            .get(info.getUrl());
19        response.then().log().all();
20        return response;
21    }
22 }
```



The screenshot shows the same IDE with the project 'ejercicio2APIMel'. The left sidebar shows the 'test' directory expanded to the 'factoryRequest' package, with 'RequestPUT.java' selected. The main editor displays the code for 'RequestPUT.java', which implements the 'IRequest' interface. The code includes an import for 'io.restassured.response.Response'. The 'send' method is annotated with '@Override' and uses 'given()' to configure the request, followed by 'auth()', 'preemptive()', 'basic()', 'body()', 'log().all()', and 'when()' to execute the request. The response is then logged and returned.

```
1 package factoryRequest;
2
3 import ...
4
5 public class RequestPUT implements IRequest{
6     @Override
7     public Response send(RequestInfo info) {
8         Response response =given() RequestSpecification
9             .auth() AuthenticationSpecification
10             .preemptive() PreemptiveAuthSpec
11             .basic(GetProperties.getInstance().getUser(),
12                 GetProperties.getInstance().getPwd()) Reque
13             .body(info.getBody())
14             .log().all()
15             .when()
16             .put(info.getUrl());
17
18         response.then().log().all();
19         return response;
20     }
21 }
22
23
24
25 }
```



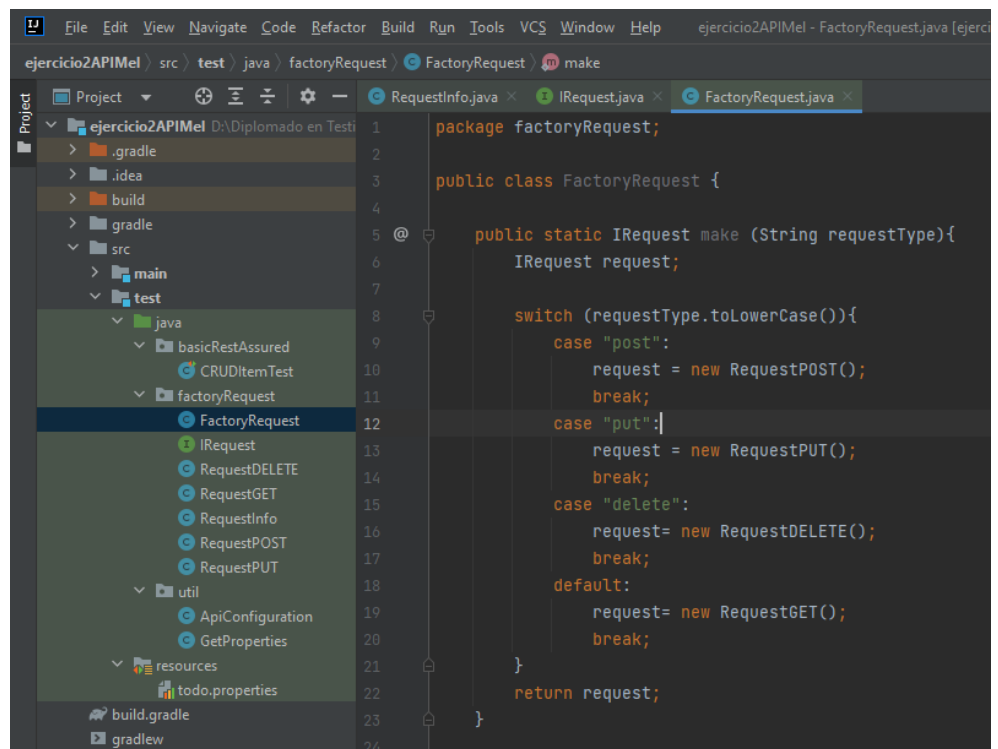
```
package factoryRequest;

import ...

public class RequestDELETE implements IRequest {

    @Override
    public Response send(RequestInfo info) {
        Response response = given() RequestSpecification
            .auth() AuthenticationSpecification
            .preemptive() PreemptiveAuthSpec
            .basic(GetProperties.getInstance().getUser(),
                GetProperties.getInstance().getPwd()) RequestSpecifi
            .log().all()
            .when()
            .delete(info.getUrl());
        response.then().log().all();
        return response;
    }
}
```

iv. FactoryRequest



```
package factoryRequest;

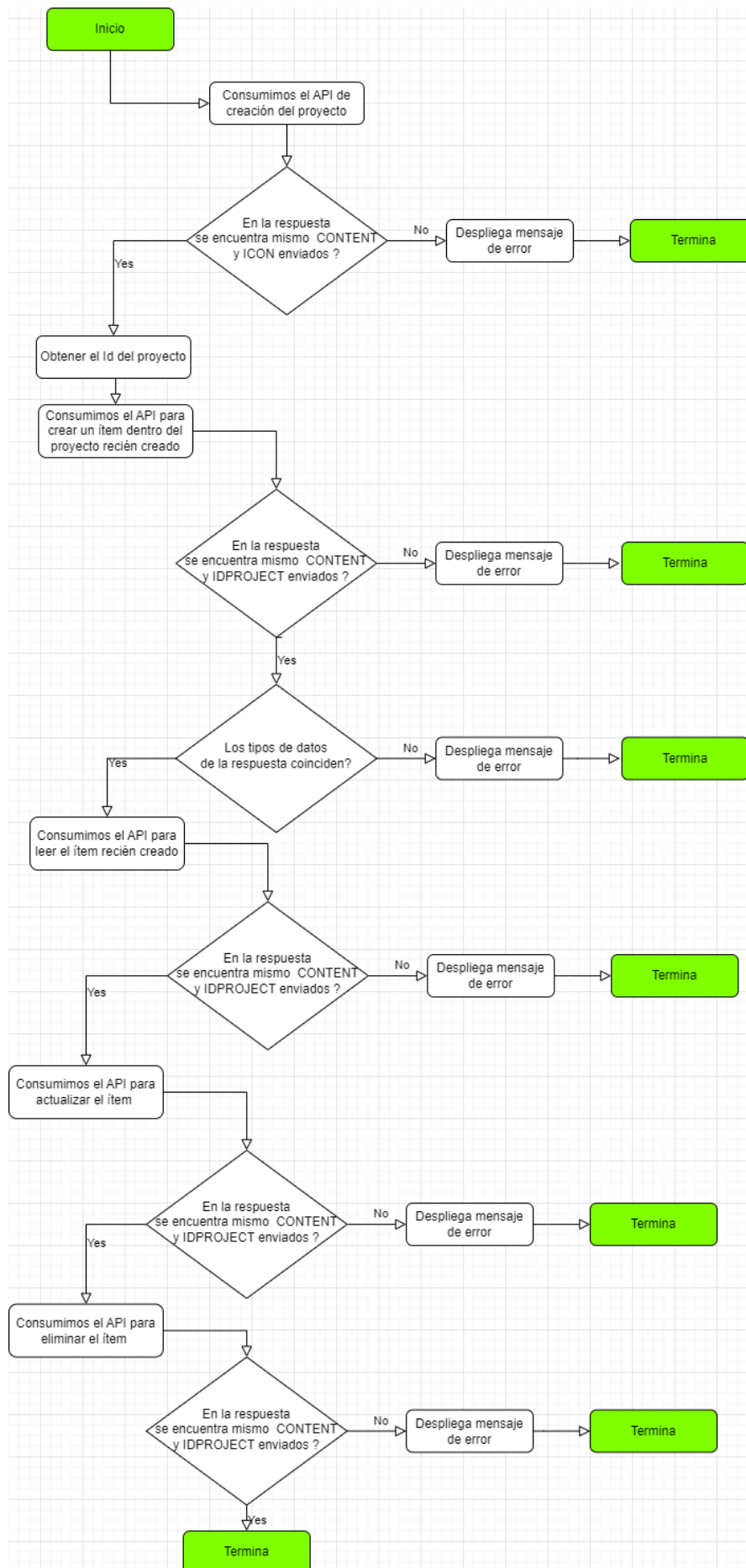
public class FactoryRequest {

    @
    public static IRequest make (String requestType){
        IRequest request;

        switch (requestType.toLowerCase()){
            case "post":
                request = new RequestPOST();
                break;
            case "put":
                request = new RequestPUT();
                break;
            case "delete":
                request = new RequestDELETE();
                break;
            default:
                request = new RequestGET();
                break;
        }
        return request;
    }
}
```

6. Creación de Test

Flujo



Creamos el archivo para la validación schemaCreateItemResponse.json

The screenshot shows the Liquid Technologies website. The header includes the company name 'liquid .XML the smart way!', navigation links (Products, Pricing, Download, Support, Company), and user options (Sign In / Register, Shopping Cart). The main heading is 'Free Online JSON to JSON Schema Converter'. Below it, a subheading says 'Uses the sample JSON document to infer a JSON schema.' A large blue banner promotes the 'Complete XML Toolkit' and 'FREE Community Edition', with a '2024 New Release' badge. The 'Sample JSON Document' section contains a JSON object with fields like 'DueDateTime', 'CreatedDate', 'LastCheckedDate', 'LastUpdatedDate', 'Deleted', 'Notes', 'InHistory', 'SyncClientCreationId', 'DueTimeSpecified', and 'OwnerId'. Below the JSON is a 'Generate Schema' button. A sidebar on the left lists various tools and tutorials. At the bottom, there is a cookie notice and a file explorer showing 'FujoEjercicio2-Pag....png'.

sales@liquid-technologies.com Sign In / Register Shopping Cart

liquid .XML the smart way! Products Pricing Download Support Company

Free Online JSON to JSON Schema Converter

Uses the sample JSON document to infer a JSON schema.

Complete XML Toolkit
XSD | JSON | XSLT | WSDL | DTD | XML Mapping

FREE Community Edition
2024 New Release

Access the online tools directly from your desktop.
Download Free Liquid Studio Community Edition Now!

Sample JSON Document

```
18 },
19
20 "DueDateTime": null,
21 "CreatedDate": "/Date(1665963084903)/",
22 "LastCheckedDate": null,
23 "LastUpdatedDate": "/Date(1665963084903)/",
24 "Deleted": false,
25 "Notes": "",
26 "InHistory": false,
27 "SyncClientCreationId": null,
28 "DueTimeSpecified": true,
29 "OwnerId": 705902
30 }
```

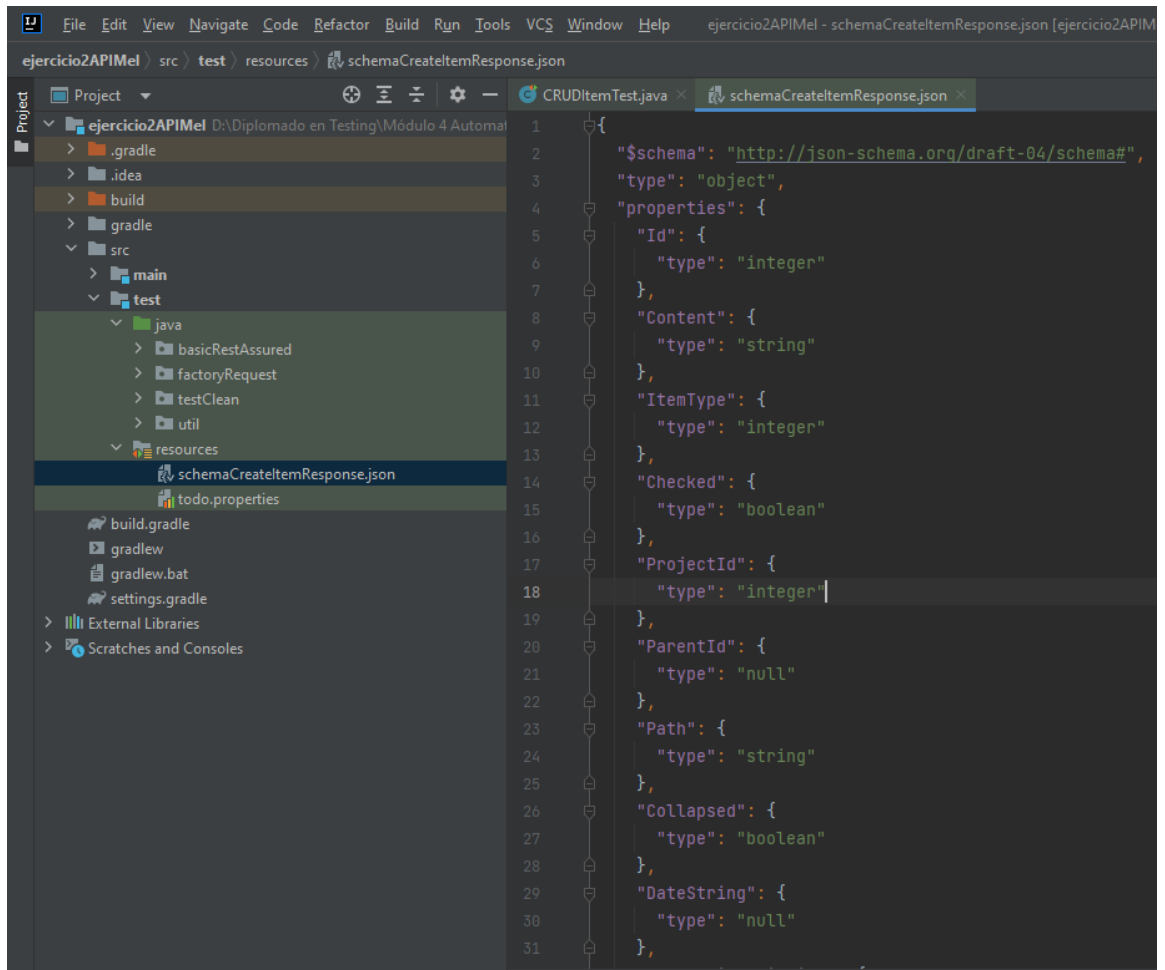
Options

✓ No soy un robot reCAPTCHA

Generate Schema

Liquid Technologies Web Site uses cookies. [Learn more](#) [Close](#)

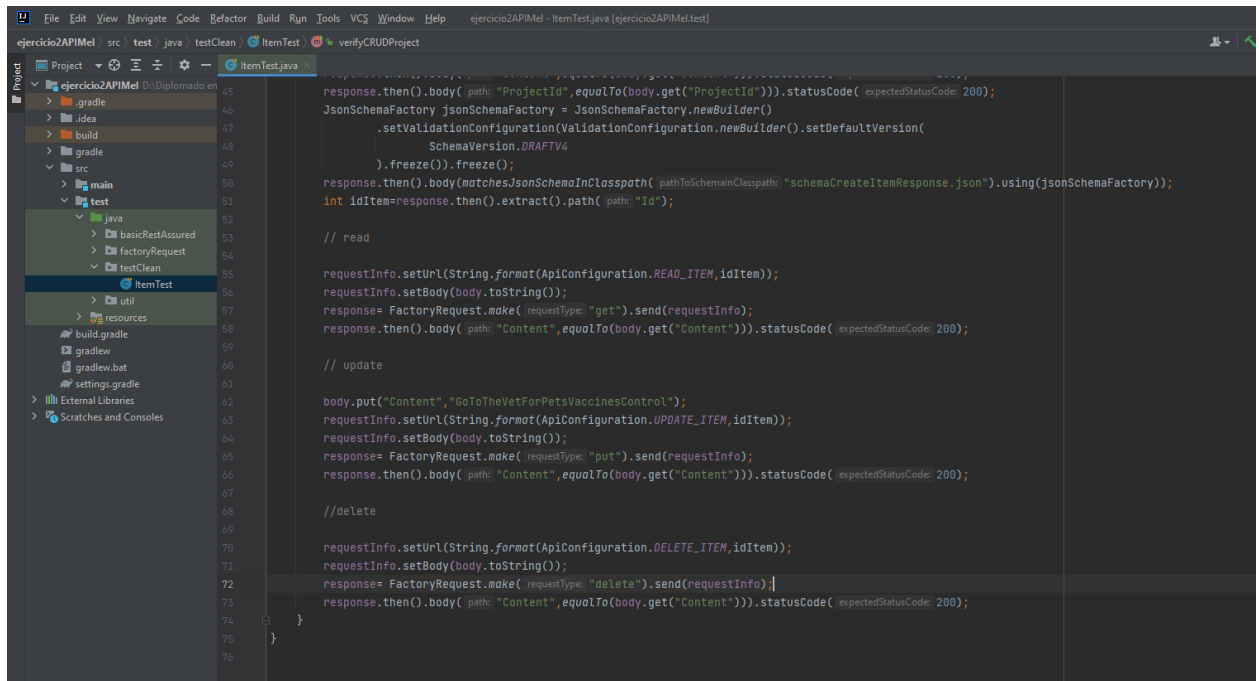
FujoEjercicio2-Pag....png [Mostrar todo](#) [X](#)



El test limpio


```
1 package testClean;
2
3 import com.github.fge.jsonschema.SchemaVersion;
4 import com.github.fge.jsonschema.cfg.ValidationConfiguration;
5 import com.github.fge.jsonschema.main.JsonSchemaFactory;
6 import factoryRequest.FactoryRequest;
7 import factoryRequest.RequestInfo;
8 import io.restassured.response.Response;
9 import org.json.JSONObject;
10 import org.junit.jupiter.api.Test;
11 import util.ApiConfiguration;
12
13 import static io.restassured.module.jsv.JsonSchemaValidator.matchesJsonSchemaInClasspath;
14 import static org.hamcrest.Matchers.equalTo;
15
16 public class ItemTest {
17     Response response;
18     JSONObject body= new JSONObject();
19     RequestInfo requestInfo = new RequestInfo();
20
21     @Test
22     public void verifyCRUDProject(){
23         //create project
24
25         body.put("Content","Melissa");
26         body.put("Icon",6);
27         requestInfo.setUrl(ApiConfiguration.CREATE_PROJECT);
28         requestInfo.setBody(body.toString());
29
30         response= FactoryRequest.make( requestType: "post").send(requestInfo);
31         response.then().body( path: "Content",equalTo(body.get("Content"))).statusCode( expectedStatusCode: 200);
32         response.then().body( path: "Icon",equalTo(body.get("Icon"))).statusCode( expectedStatusCode: 200);
33         int idProject=response.then().extract().path( path: "Id");
34     }
```

```
34 //create item
35
36 body = new JSONObject();
37 body.put("Content","GoToTheVetForPetsVaccines");
38 body.put("ProjectId",idProject);
39 requestInfo.setUrl(ApiConfiguration.CREATE_ITEM);
40 requestInfo.setBody(body.toString());
41
42 response= FactoryRequest.make( requestType: "post").send(requestInfo);
43 response.then().body( path: "Content",equalTo(body.get("Content"))).statusCode( expectedStatusCode: 200);
44 response.then().body( path: "ProjectId",equalTo(body.get("ProjectId"))).statusCode( expectedStatusCode: 200);
45 JsonSchemaFactory jsonSchemaFactory = JsonSchemaFactory.newBuilder()
46     .setValidationConfiguration(ValidationConfiguration.newBuilder().setDefaultVersion(
47         SchemaVersion.DRAFT4
48     ).freeze()).freeze();
49 response.then().body(matchesJsonSchemaInClasspath( pathToSchemaInClasspath: "schemaCreateItemResponse.json").using(jsonSchemaFactory));
50 int idItem=response.then().extract().path( path: "Id");
51 }
```



```
ejercicio2APIMel - ItemTest.java [ejercicio2APIMel.test]
ejercicio2APIMel src / test / java / testClean ItemTest verifyCRUDProject
Project
  ejercicio2APIMel
    .gradle
    .idea
    .build
    gradle
    src
    test
      java
        basicRestAssured
        factoryRequest
        testClean
          ItemTest
      resources
      build.gradle
      gradlew
      gradlew.bat
      settings.gradle
    External Libraries
    Scratches and Consoles

response.then().body( path: "ProjectId", equalTo(body.get("ProjectId"))).statusCode( expectedStatusCode: 200);
JsonSchemaFactory jsonSchemaFactory = JsonSchemaFactory.newBuilder()
    .setValidationConfiguration(ValidationConfiguration.newBuilder().setDefaultVersion(
        SchemaVersion.DRAFTV4
    ).freeze()).freeze();
response.then().body(matchesJsonSchemaInClasspath( pathToSchemaInClasspath: "schemaCreateItemResponse.json").using(jsonSchemaFactory));
int idItem=response.then().extract().path( path: "Id");

// read

requestInfo.setUrl(String.format(ApiConfiguration.READ_ITEM,idItem));
requestInfo.setBody(body.toString());
response= FactoryRequest.make( requestType: "get").send(requestInfo);
response.then().body( path: "Content", equalTo(body.get("Content"))).statusCode( expectedStatusCode: 200);

// update

body.put("Content","GoToTheVetForPetsVaccinesControl");
requestInfo.setUrl(String.format(ApiConfiguration.UPDATE_ITEM,idItem));
requestInfo.setBody(body.toString());
response= FactoryRequest.make( requestType: "put").send(requestInfo);
response.then().body( path: "Content", equalTo(body.get("Content"))).statusCode( expectedStatusCode: 200);

//delete

requestInfo.setUrl(String.format(ApiConfiguration.DELETE_ITEM,idItem));
requestInfo.setBody(body.toString());
response= FactoryRequest.make( requestType: "delete").send(requestInfo);
response.then().body( path: "Content", equalTo(body.get("Content"))).statusCode( expectedStatusCode: 200);
}
```

CÓDIGO

El proyecto se encuentra versionado en el siguiente

enlace:<https://github.com/melJimenez/Mod4Ejercicio2MelJimenez.git>

RESULTADO DE EJECUCIÓN

La ejecución se captura en video compartido en el siguiente enlace:

<https://github.com/melJimenez/Mod4Ejercicio2MelJimenez/blob/main/videoEjercicio2.mp4>