

Gobierno de APIs. (API Owner)

Herramientas de testing

Índice del capítulo

- ▶ Postman.
- ▶ SOAP UI.

Postman

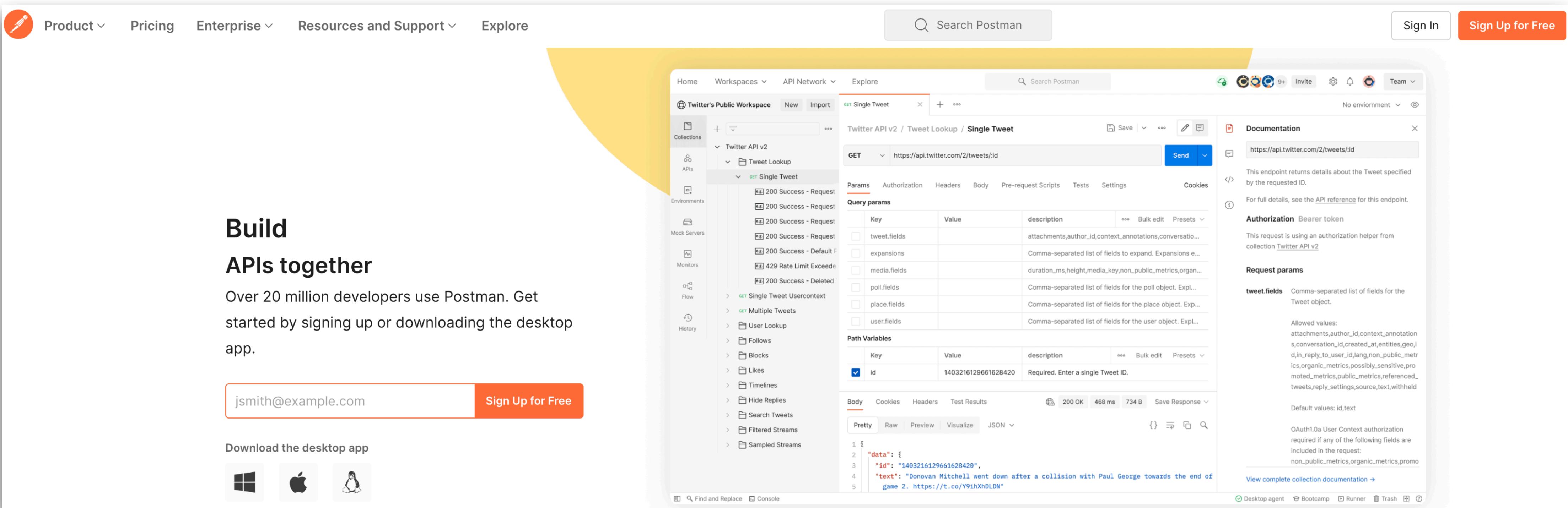
- ▶ De un vistazo.
- ▶ Peticiones y colecciones.
- ▶ Entornos.
- ▶ Trabajando con APIs.
- ▶ Tests.
- ▶ Servidores Mock.

De un vistazo

- ▶ Introducción.
- ▶ Descarga e instalación.
- ▶ La pantalla principal.
- ▶ Organizando el trabajo en workspaces.

Introducción

- ▶ **Postman es una plataforma para construir y usar APIs.**
- ▶ Postman **simplifica** cada paso del **ciclo de vida de la API** y agiliza la **colaboración** para que pueda crear mejores API, más rápido.
- ▶ <https://www.postman.com/>



The screenshot shows the Postman website interface. On the left, there's a sidebar with navigation links: Product, Pricing, Enterprise, Resources and Support, and Explore. A yellow callout points from the 'Explore' link to the main workspace area. The main workspace shows a collection named 'Twitter's Public Workspace' containing a 'Twitter API v2' collection with a 'Tweet Lookup' folder and a 'Single Tweet' endpoint. The 'Single Tweet' endpoint is selected, showing a GET request to 'https://api.twitter.com/2/tweets/:id'. The 'Query params' section includes 'tweet.fields' and 'expansions'. The 'Path Variables' section has a checked 'id' variable with the value '1403216129661628420'. The 'Body' tab shows a JSON response with fields 'data', 'id', and 'text'. To the right, there's a 'Documentation' panel with the URL 'https://api.twitter.com/2/tweets/:id' and a detailed description of the endpoint. Below the documentation are sections for 'Authorization', 'Request params', and 'OAuth1.0a User Context authorization'. At the bottom, there are links for 'View complete collection documentation' and file management options like 'Desktop agent', 'Bootcamp', 'Runner', 'Trash', and 'Edit'.

Introducción

- ▶ <https://www.postman.com/pricing/>

Postman API Platform Plans and Pricing

Free	Basic	Professional	Enterprise
<p>Start designing, developing, and testing APIs.</p> <p>\$ 0</p> <p>Sign Up</p> <p>All core tooling and collaboration for up to 3 users</p>	<p>Collaborate with your team to design, develop, and test APIs faster.</p> <p>\$ 12</p> <p>Per user/month, billed annually \$15 Per user/month, billed monthly</p> <p>Buy Now</p> <p>Everything in Free, plus:</p> <ul style="list-style-type: none"> ✓ Unlimited collaboration for plan members ✓ Collection recovery for 30 days ✓ 1 custom domain ✓ 10x calls to Postman API ✓ 10 integrations 	<p>Centrally manage the entire API workflow.</p> <p>\$ 29</p> <p>Per user/month, billed annually \$36 Per user/month, billed monthly</p> <p>Buy Now</p> <p>Everything in Basic, plus:</p> <ul style="list-style-type: none"> ✓ Native Git Support ✓ Single Sign-On Google Workspaces ✓ Collection recovery for 90 days ✓ Basic roles and permissions ✓ Private workspaces ✓ Static IP addresses for API testing 	<p>Securely manage, organize, and accelerate API-first development at scale.</p> <p>\$ 99</p> <p>Per user/month, billed annually</p> <p>Contact Sales</p> <p>Everything in Professional, plus:</p> <ul style="list-style-type: none"> ✓ Identity and access management, including user governance ✓ Automated provisioning and deprovisioning via SCIM ✓ SSO Identity Management & SAML ✓ Reporting and analytics ✓ Deployment control

Descarga e instalación

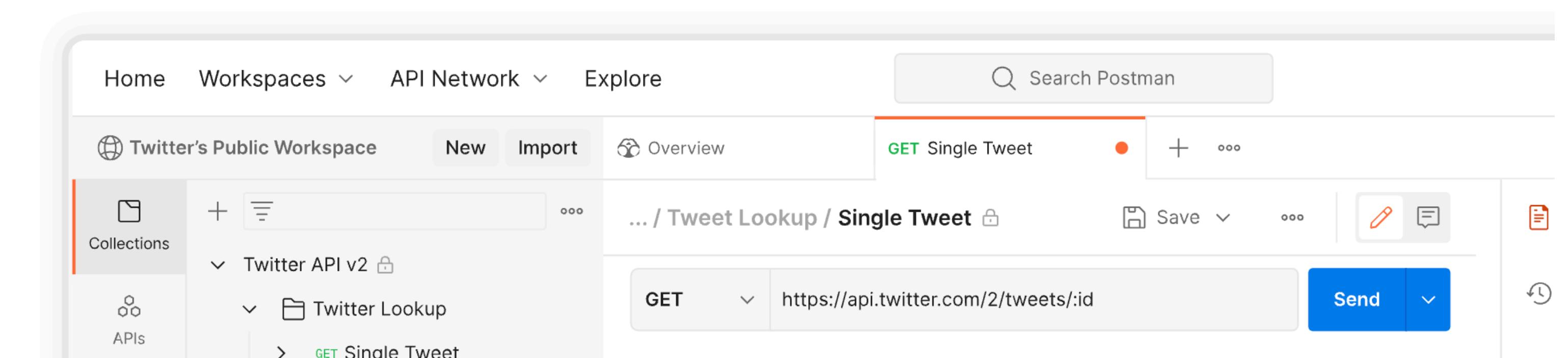
- ▶ <https://www.postman.com/downloads/>

Download Postman

Download the app to get started using the Postman API Platform today. Or, if you prefer a browser experience, you can try the web version of Postman.

The Postman app

Download the app to get started with the Postman API Platform.

[Mac Intel Chip](#)[Mac Apple Chip](#)

La pantalla principal

The screenshot shows the main interface of the Postman application. At the top, there is a navigation bar with icons for Home, Workspaces, API Network, and Explore, along with a search bar and user account options. The main content area features a greeting message "Rise and shine, jsalinas76!" followed by a reminder to "Pick up where you left off." A central modal window titled "Take a shortcut to sending requests" shows a screenshot of the Postman interface with a "Send a request" button. To the right, there is a "Product Updates" section with a link to the VS Code Waitlist and a "Sign Up" button. Below that is an "Activity Feed" section with a placeholder message about team activity. On the left side, there is a sidebar with a team collaboration illustration and sections for "Postman works best with teams", "Workspaces", "API governance", "API security", "Integrations", and "Reports". A "Create Team" button is also present in this sidebar.

Rise and shine, jsalinas76!

Pick up where you left off.

Take a shortcut to sending requests

Send an HTTP request to any endpoint.

Send a request

Product Updates

Join the VS Code Waitlist NEW

Sign up for early access to try out the new Visual Studio Code extension that brings Postman features to your...

Sign Up

Activity Feed

Your team's activity will show up here

Get started by inviting people to your team.

Create Team

Postman works best with teams

Collaborate in real-time and establish a single source of truth for all API workflows.

Create Team

Workspaces >

API governance >

API security >

Integrations >

Reports >

Recently visited workspaces

Curso Postman

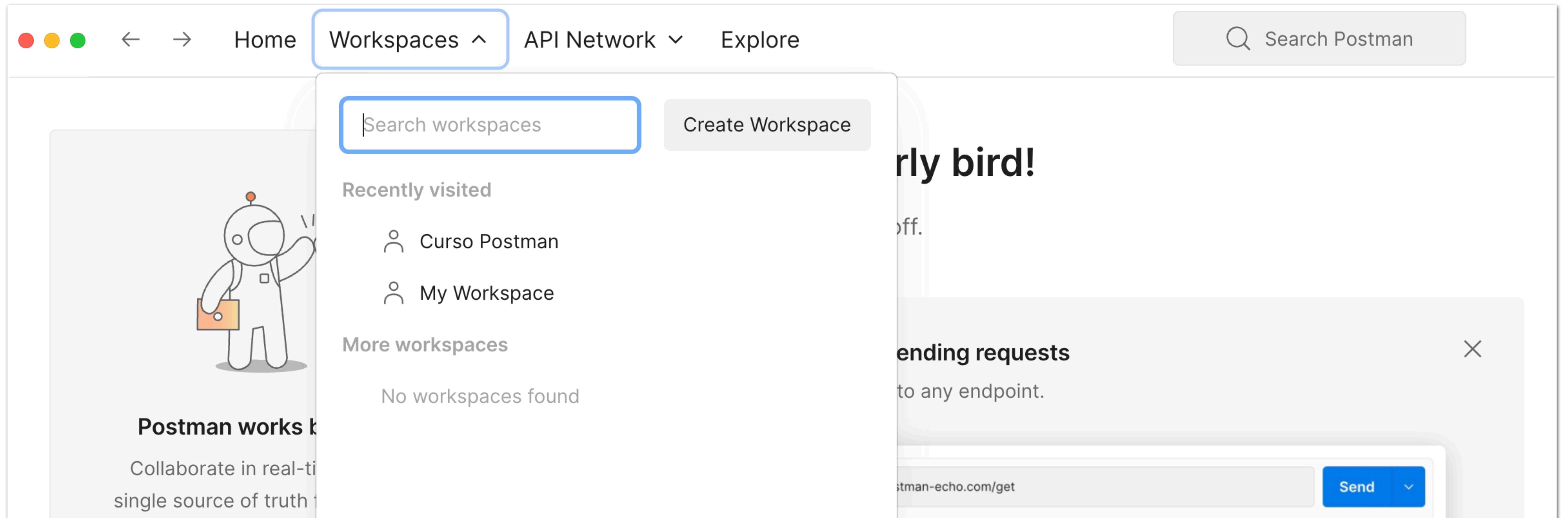
My Workspace

La pantalla principal

- ▶ Desde la pantalla principal podemos acceder a las secciones:
 - ▶ Home.
 - ▶ Workspaces.
 - ▶ API Networks.
 - ▶ Explore.
- ▶ La sección que nos interesa en **Workspaces**.

Organizando el trabajo en workspaces

- El espacio de trabajo nos permite **organizarnos** adecuadamente:



Organizando el trabajo en workspaces

- ▶ Para crear un workspace:

Create workspace

Name

Summary

Add a brief summary about this workspace.

Visibility

Determines who can access this workspace.

Personal
Only you can access

Private
Only invited team members can access

Team
All team members can access

Partner NEW
Only invited partners and team members can access

Public
Everyone can view

Create Workspace **Cancel**

Organizando el trabajo en workspaces

- ▶ Una vez creado nos encontramos con algo tal que así:

The screenshot shows the Postman application interface for a "Espacio de trabajo personalizado" (Personal Workspace). The left sidebar includes links for Collections, APIs, Environments, Mock Servers, Monitors, Flows, and History. A prominent feature is the "Create a collection for your requests" section, which explains that collections group related requests and can set common authorization, tests, scripts, and variables. Below this is a "Create Collection" button. The main workspace area is titled "Overview" and displays the title "Espacio de trabajo personalizado". It includes fields for adding a brief summary and a detailed description. An "Activity" section shows a recent creation by "jsalinas76" just now. On the right, a sidebar titled "In this workspace" lists Requests (0), Collections (0), APIs (0), Environments (0), Mock Servers (0), and Monitors (0).

Organizando el trabajo en workspaces

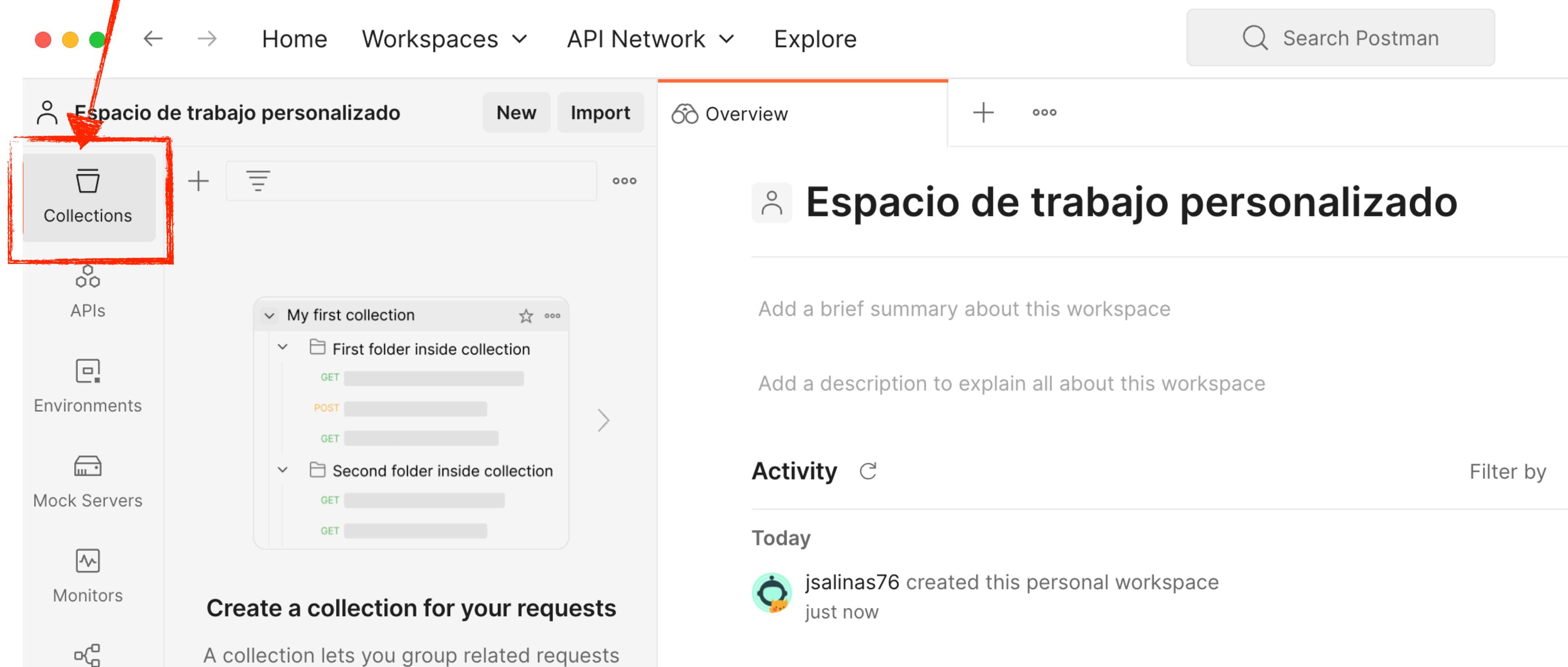
- ▶ Dentro de nuestro espacio de trabajo se podrán gestionar:
 - ▶ Colecciones.
 - ▶ APIs.
 - ▶ Entornos.
 - ▶ Servidores Mock.
 - ▶ Monitorización.
 - ▶ ...

Peticiones y colecciones

- ▶ Introducción.
- ▶ Creando una colección.
- ▶ Creando una petición.
- ▶ Creando ejemplos para una petición.
- ▶ Definiendo parámetros de consulta.
- ▶ Definiendo parámetros de url.
- ▶ Definiendo el cuerpo de la petición.
- ▶ Trabajando con autorización.

Introducción

- Postman nos permite crear peticiones REST de forma simple. La forma de organizarlas será usando **colecciones**.
- Dentro de las colecciones podremos guardarlas en carpetas.



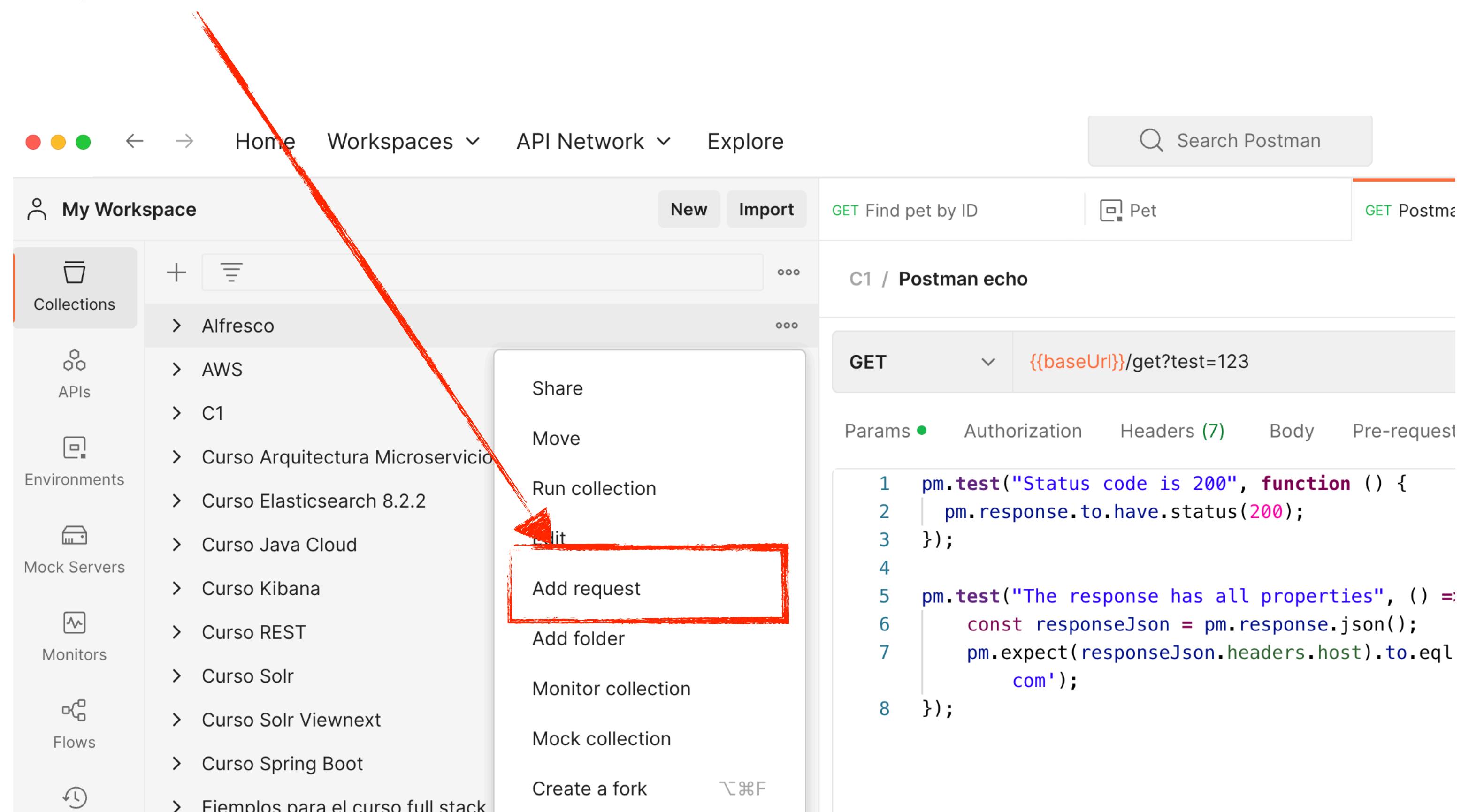
Creando una colección

- Se hace click en el botón + y se rellena el formulario:

The screenshot shows the Postman application interface. A red arrow points from the top-left towards the '+ New Collection' button in the main toolbar. The left sidebar lists 'Collections', 'APIs', 'Environments', 'Mock Servers', and 'Monitors'. The main workspace is titled 'New Collection' under 'Overview'. It contains tabs for 'Authorization', 'Pre-request Script', 'Tests', 'Variables', and 'Runs'. The 'Authorization' tab is selected, showing 'Type: No Auth'. A note below states, 'This collection does not use any authorization. Learn more about [authorization](#)'. To the right, there's a 'Documentation' section with a placeholder 'Add collection description...' and three circular icons.

Creando una petición

- Una vez creada una colección, la herramienta nos ofrece operaciones sobre ellas. Entre otras, la creación de una petición:



Creando una petición

The screenshot shows the Postman application interface for creating a new API request. The top navigation bar includes a 'New Request' button, a 'Save' icon, and other global settings. The main request configuration area is set to 'GET' method and has a placeholder 'Enter request URL'. Below this, the 'Params' tab is selected, showing a table for 'Query Params' with columns: KEY, VALUE, DESCRIPTION, and Bulk Edit. A single row is present with 'Key' and 'Value' fields. To the right of the table are icons for 'Cookies', 'Header Editor', 'Flashlight', and 'Info'. At the bottom of the request section is a note 'h 8.2.2'. The bottom half of the screen is a large, empty 'Response' area.

Creando ejemplos para una petición

- Podemos personalizar una petición aprovechando el concepto de **ejemplo**:

The screenshot shows the Elasticsearch interface. On the left, there's a sidebar with a tree view of a course structure:

- Curso Elasticsearch 8.2.2
 - Conceptos fundamentales
 - Indexando datos
 - Trabajando con índices

In the main area, there's a search bar with "PUT Crear índice usuario" and a dropdown menu with the following options:

- Open in Tab
- Add example
- View Documentation
- Rename ⌘E
- Duplicate ⌘D
- Paste ⌘V
- Delete ⌘X

A red box and arrow highlight the "Add example" option. To the right, the "Response" section displays the following JSON code:

```
1 {  
2   "settings": {  
3     "number_of_shards": 1,  
4     "number_of_replicas": 2  
5   }  
6 }
```

At the bottom right, there's a small illustration of a rocket launching.

Definiendo parámetros de consulta

- La sección **Params** nos permite definir **Query Params**:

The screenshot shows the Postman interface for a PUT request to `https://localhost:9200/usuario`. The 'Params' tab is selected, indicated by an orange underline. Below it, the 'Query Params' section is visible, featuring a table with columns: KEY, VALUE, and DESCRIPTION. A single row is present in the table, showing 'Key' in the KEY column and 'Value' in the VALUE column.

KEY	VALUE	DESCRIPTION
Key	Value	Description

Definiendo parámetros de url

- Si modificamos la **url** de la petición utilizando la **sintaxis :nombre** nos encontramos con la opción de los **parámetros de url**:

The screenshot shows the Postman interface for a PUT request to `https://localhost:9200/usuario/:userId`. The 'Params' tab is selected, showing a table for 'Query Params' with one entry: 'Key' and 'Value'. Below this, the 'Path Variables' section shows a table with one entry: 'userId' and 'Value'.

KEY	VALUE	DESCRIPTION
Key	Value	Description

KEY	VALUE	DESCRIPTION
userId	Value	Description

Definiendo el cuerpo de la petición

- La sección Body me permite definir este concepto:

The screenshot shows the Postman interface with a PUT request to `https://localhost:9200/usuario`. The 'Body' tab is active, displaying a JSON configuration for Elasticsearch settings:

```
1 {  
2   "settings": {  
3     "number_of_shards": 1,  
4     "number_of_replicas": 2  
5   }  
6 }
```

Trabajando con autorización

The screenshot shows the Postman interface for a PUT request to `https://localhost:9200/usuario`. The 'Authorization' tab is selected. A dropdown menu is open, showing options like 'Inherit auth from ...', 'No Auth', 'API Key', 'Bearer Token', 'JWT Bearer', 'Basic Auth', 'Digest Auth', 'OAuth 1.0', 'OAuth 2.0', and 'Hawk Authentication'. A note at the bottom right says 'This request is using Basic Auth from collection [Curso Elasticsearch 8.2.2](#)'.

PUT https://localhost:9200/usuario

Params Authorization Headers (10) Body • Pre-request Script Tests Settings

Type Inherit auth fr...

Inherit auth from ...

No Auth

API Key

Bearer Token

JWT Bearer

Basic Auth

Digest Auth

OAuth 1.0

OAuth 2.0

Hawk Authentication...

This request is using Basic Auth from collection [Curso Elasticsearch 8.2.2](#).

Entornos

- ▶ Introducción.
- ▶ Definiendo variables.
- ▶ Usando variables.
- ▶ Variables a nivel de colección.
- ▶ El alcance de una variable.
- ▶ Aplicando entornos sobre nuestras colecciones.

Introducción

- ▶ Los entornos son una forma de **parametrizar** nuestro trabajo definiendo **variables**.
- ▶ Estas variables podrán ser utilizadas en nuestras peticiones flexibilizando así todo lo que hacemos.
- ▶ Podemos definir tantos entornos como necesitemos y aplicarlos con facilidad.

The screenshot shows the Postman application interface. On the left, there's a sidebar with icons for Collections, APIs, Environments (which is highlighted with a red box and has a red arrow pointing to it), Mock Servers, Monitors, Flows, and History. The main workspace is titled 'Base' and contains a table of environment variables. The table has columns for VARIABLE, TYPE, INITIAL VALUE, CURRENT VALUE, and actions like Persist All and Reset All. One variable, 'baseUrl', is listed with a checked checkbox, a 'default' type, an initial value of 'https://postman-echo.com', and a current value of 'https://postman-echo.com'. A search bar at the top says 'Filter variables'.

	VARIABLE	TYPE ⓘ	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	...	Persist All	Reset All
<input checked="" type="checkbox"/>	baseUrl	default	https://postman-echo.com	https://postman-echo.com	...		
	Add a new variable						

Definiendo variables

- Se define el nombre, tipo y valores:

The screenshot shows the Postman environment variables interface. At the top, there's a header with 'Base' and a dropdown menu. Below the header is a toolbar with 'Fork' (0), 'Save', 'Share', and other options. A search bar labeled 'Filter variables' is present. The main area is a table with the following columns: VARIABLE, TYPE ⓘ, INITIAL VALUE ⓘ, CURRENT VALUE ⓘ, and three additional columns for 'Persist All' and 'Reset All'. One row is shown for 'baseUrl' with a checked checkbox in the first column. The 'INITIAL VALUE' is 'https://postman-echo.com' and the 'CURRENT VALUE' is also 'https://postman-echo.com'. A button 'Add a new variable' is located at the bottom of the table.

	VARIABLE	TYPE ⓘ	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ	...	Persist All	Reset All
<input checked="" type="checkbox"/>	baseUrl	default	https://postman-echo.com	https://postman-echo.com			
Add a new variable							

Usando variables

- Para usar una variable debemos aplicar la siguiente sintaxis:

```
 {{NOMBRE_VARIABLE}}
```

- Ejemplo:

The screenshot shows the Postman interface with a red annotation highlighting the use of variables in the URL and headers.

URL: {{baseUrl}}/pet/:petId

Authorization: Type: API Key, Key: api_key, Value: {{apiKey}}, Add to: Header

Variables a nivel de colección

- ▶ También pueden definirse a **este nivel**:

The screenshot shows the Postman collection editor interface. At the top, there are tabs for 'Base' (selected), 'Swagger Petstore - Open...', '+', '...', and 'Base'. Below the tabs, the title 'Swagger Petstore - Op... / Swagger Petstore - Op...' is displayed, along with 'Fork | 0' and '...' buttons. The main navigation bar includes 'Authorization', 'Pre-request Script', 'Tests', 'Variables' (which is underlined and highlighted with a red arrow), and 'Runs'. A note below the navigation states: 'These variables are specific to this collection and its requests. Learn more about [collection variables](#)'.

A red box highlights the 'Variables' section, which contains a 'Filter variables' input field and a table. The table has columns: VARIABLE, INITIAL VA..., CURRENT, ..., Persist All, and Reset All. It shows one row for 'baseUrl' with the value 'https://petstor...'. There is also a link 'Add a new var...'. At the bottom of the table are 'Persist All' and 'Reset All' buttons.

To the right of the collection editor is a 'Documentation' panel. It starts with a heading 'Documentation' and a note: 'This is a sample Pet Store Server based on the OpenAPI 3.0 specification. You can find out more about Swagger at <https://swagger.io>. In the third iteration of the pet store, we've switched to the design first approach!'. It continues with instructions: 'You can now help us improve the API whether it's by making changes to the definition itself or to the code. That way, with time, we can improve the API in general, and expose some of the new features in OAS3.' Below this, there is a note: 'If you're looking for the Swagger 2.0/OAS 2.0 version of Petstore, then click [here](#). Alternatively, you can load via the Edit > Load Petstore OAS 2.0 menu option!'. At the very bottom of the documentation panel, there is a link 'Some useful links'.

El alcance de una variable

- La realidad es que podemos definir variables en diferentes ámbitos. Si colocamos el puntero sobre una variable la herramienta nos informa de estas opciones:

The screenshot shows the Postman application interface. A tooltip is displayed over a placeholder in the URL bar: `http://jsonplaceholder.typicode.com/users/{{userId}}`. The tooltip title is "Set as new variable". It contains fields for "Name" (set to "userId") and "Value" (empty). Below these is a "Scope" dropdown labeled "Select A Scope" with three options: "Global" (selected, indicated by a blue border), "Active Environment: Base" (green border), and "Collection: JSON Api" (orange border).

Aplicando entornos sobre nuestras colecciones

- ▶ Sobre una colección en general o sobre una petición de forma específica podemos determinar cuál es el entorno con el que estamos trabajando en cada momento.
- ▶ En la esquina superior derecha de la pantalla tenemos un **desplegable** que nos permite controlar cuál se está aplicando:

The screenshot shows the Postman interface with a red arrow pointing from the top right towards a dropdown menu. The main window displays a collection named "Swagger Petstore - OpenAPI 3.0". The "Authorization" tab is selected, showing "No Auth" as the type. A note at the bottom states: "This collection does not use any authorization. Learn more about [authorization](#)". To the right, a "Documentation" panel is open, featuring a "Base" section with the text: "This is a sample Pet Store Server based on the OpenAPI 3.0 specification. You can find out more about Swagger at <https://swagger.io>. In the third iteration of the pet store, we've switched to the design first approach! You can now help us improve the API whether it's by making changes to the definition itself or to the code. That way, with time, we can improve the API in general, and expose some of the new features in OAS3." Below this, there is a note for users looking for the Swagger 2.0/OAS 2.0 version: "If you're looking for the Swagger 2.0/OAS 2.0 version of Petstore, then click [here](#). Alternatively, you can load via the [Edit > Load Petstore OAS 2.0](#) menu option!" The "Documentation" panel has a red border around its title and content area.

Trabajando con APIs

- ▶ Creando la definición de una API.
- ▶ Creando una colección a partir de la definición.
- ▶ Moviendo la colección al área de colecciones.

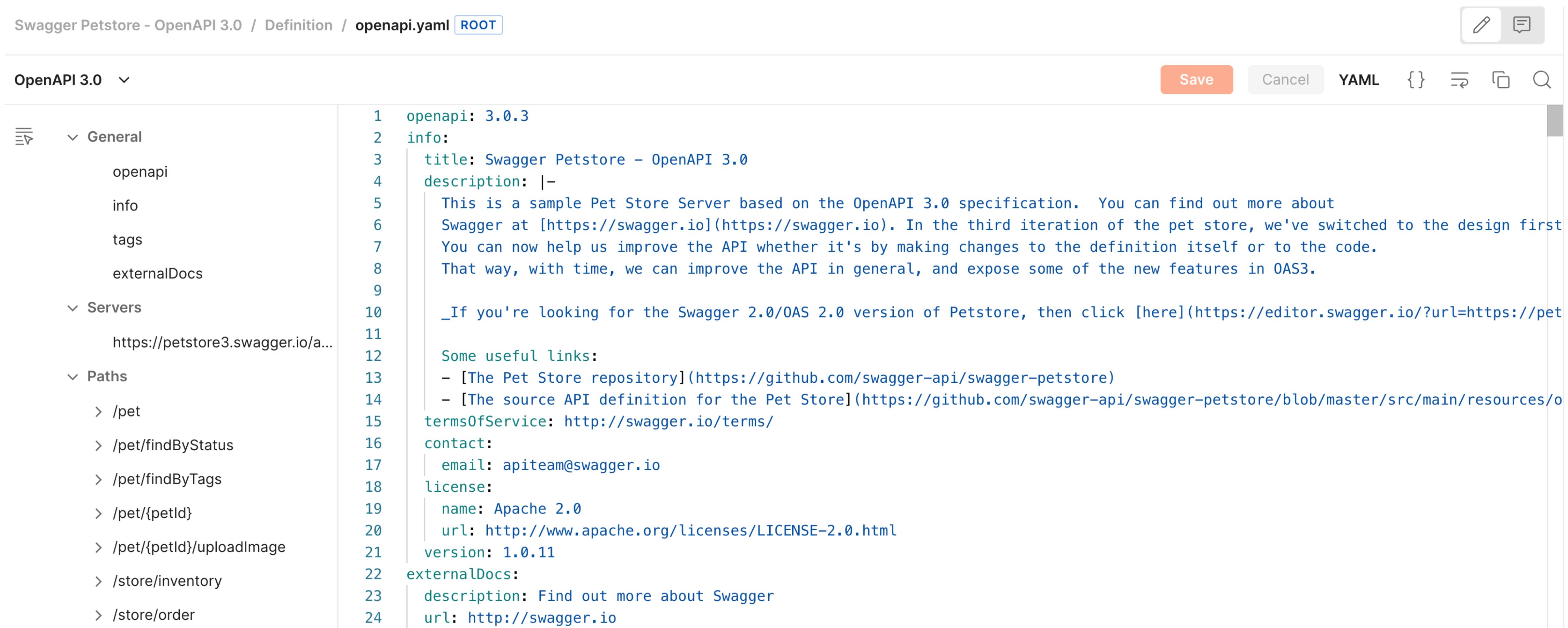
Creando la definición de una API

The screenshot shows a user interface for creating an API definition. At the top, it says "New API". On the right, there are buttons for "Share", "Watch" (with 0 notifications), and more options. The main area is divided into several sections:

- About this API**: Fields for "Add summary..." and "Add description...".
- Connect repository**: A section with a "Connect" button.
- Collections**: A section with a "+ Add" button and the message "No collections added yet".
- Definition**: A section with a "+ Add" button and the message "Add files to describe your service and setup the API definition."
- Test and Automation →**
- API Performance →**
- Deployments →**
- Publish your API to consumers**: A section with a "Publish API" button.
- Editors**

Creando la definición de una API

- El editor nos ofrece una interfaz limpia y sencilla:



The screenshot shows the Swagger Petstore OpenAPI 3.0 definition editor interface. The title bar reads "Swagger Petstore - OpenAPI 3.0 / Definition / openapi.yaml ROOT". The top right features standard file operations: Save, Cancel, YAML, {}, ⌂, ⌂, and ⌂. On the left, a sidebar lists sections: General, Servers, and Paths. Under General, there are sub-sections for openapi, info, tags, and externalDocs. Under Servers, there is a single entry: https://petstore3.swagger.io/. Under Paths, there are several entries: /pet, /pet/findByStatus, /pet/findByTags, /pet/{petId}, /pet/{petId}/uploadImage, /store/inventory, and /store/order. The main content area displays the YAML code for the API definition, starting with "openapi: 3.0.3" and including sections for info, servers, paths, and externalDocs.

```

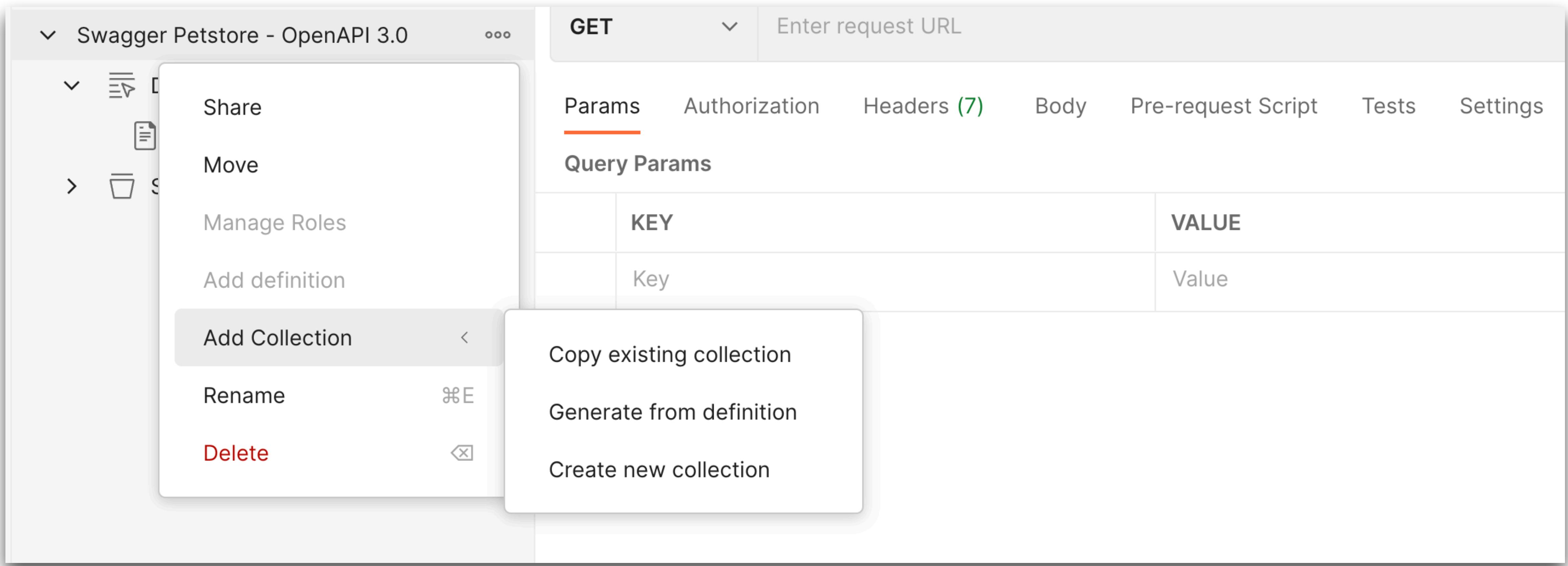
openapi: 3.0.3
info:
  title: Swagger Petstore – OpenAPI 3.0
  description: |
    This is a sample Pet Store Server based on the OpenAPI 3.0 specification. You can find out more about Swagger at [https://swagger.io](https://swagger.io). In the third iteration of the pet store, we've switched to the design first. You can now help us improve the API whether it's by making changes to the definition itself or to the code. That way, with time, we can improve the API in general, and expose some of the new features in OAS3.

  _If you're looking for the Swagger 2.0/OAS 2.0 version of Petstore, then click [here](https://editor.swagger.io/?url=https://petstore.swagger.io/v2/swagger.yaml)

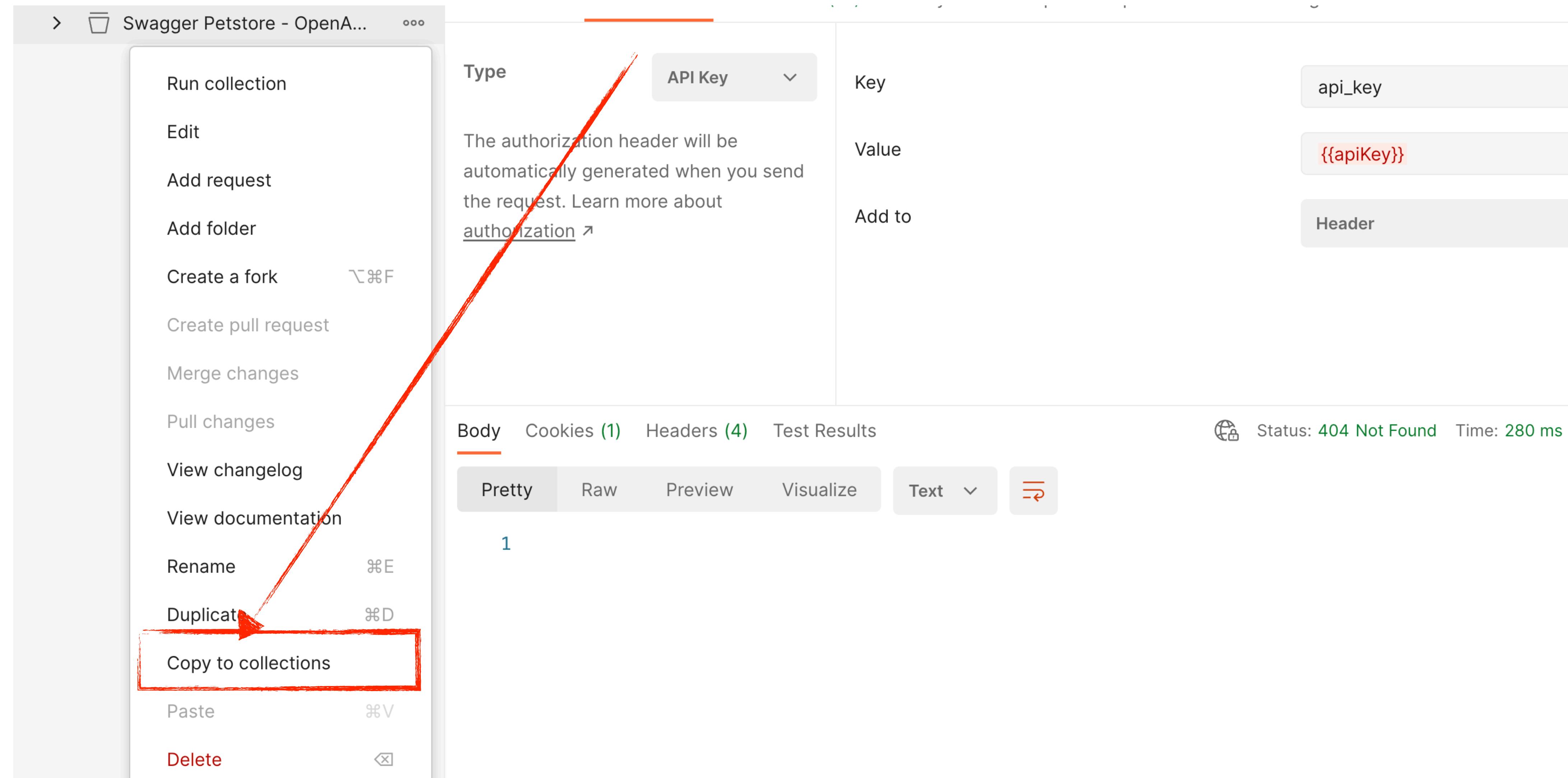
  Some useful links:
  - [The Pet Store repository](https://github.com/swagger-api/swagger-petstore)
  - [The source API definition for the Pet Store](https://github.com/swagger-api/swagger-petstore/blob/master/src/main/resources/openapi.yaml)
  termsOfService: http://swagger.io/terms/
  contact:
    email: apiteam@swagger.io
  license:
    name: Apache 2.0
    url: http://www.apache.org/licenses/LICENSE-2.0.html
  version: 1.0.11
  externalDocs:
    description: Find out more about Swagger
    url: http://swagger.io
  
```

Creando una colección a partir de la definición

- Una vez creada la definición podemos generar una colección:



Moviendo la colección al área de colecciones



Tests

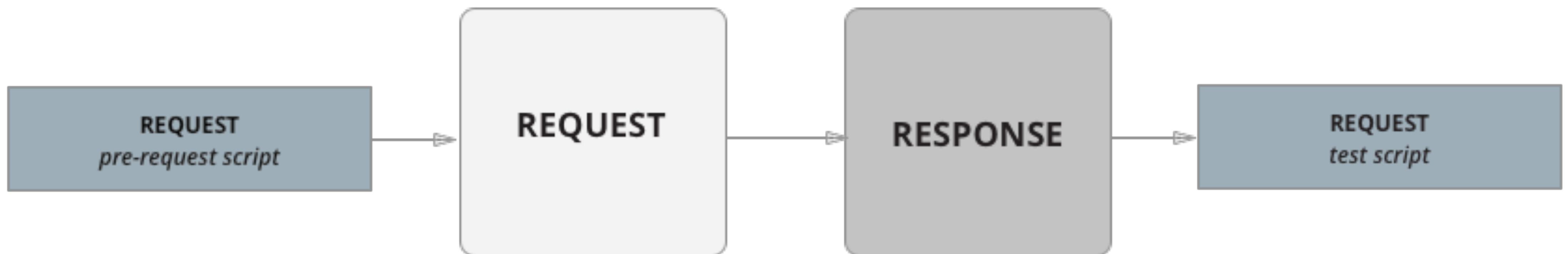
- ▶ Introducción.
- ▶ Desde la interfaz.
- ▶ Creando un test.
- ▶ La API de Postman.
- ▶ Automatizando la ejecución con Newman.
- ▶ Data-Driven Testing.

Introducción

- ▶ Cuando **construyamos APIs** una de las cosas que tenemos que entender es que **cada API o servicio** que codifiquemos es un activo más de software que hay que gestionar.
- ▶ Es por ello que como buen software, cuando construyamos el API habrá que **asociarle tests** de usuario.
- ▶ En este caso vamos a ver **cómo nos podemos apoyar en Postman** para construir nuestros tests de los APIs.
- ▶ Postman cuenta con un **entorno de ejecución Node.js** que nos permite ejecutar comportamientos dinámicos en las colecciones y peticiones. Será este entorno de ejecución Node.js el que utilizaremos para crear tests en Postman.

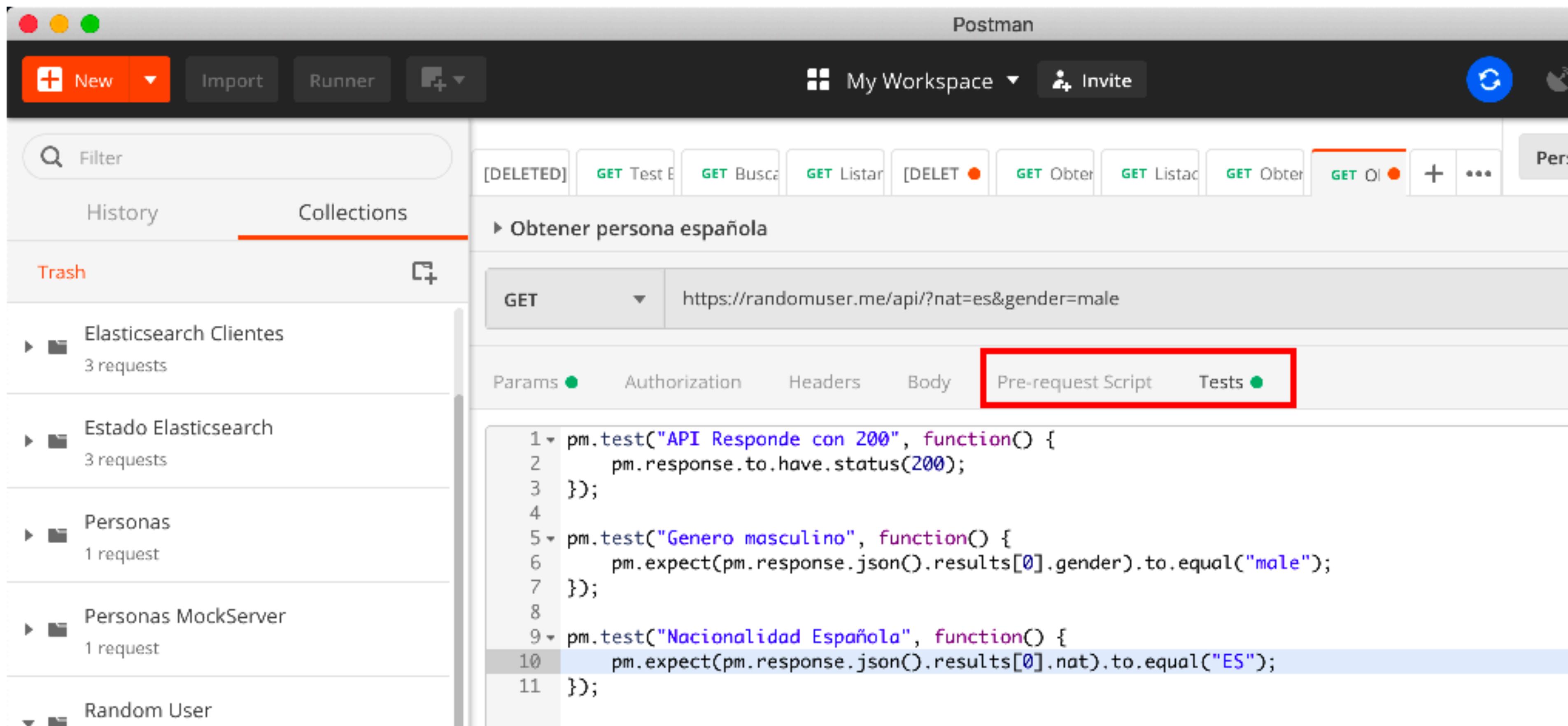
Introducción

- Postman nos ofrece **dos puntos en los cuales podemos preparar nuestros tests** sobre las APIs. En primer lugar nos posibilita hacer **pre-test scripting**, es decir, código que nos sirva para poder preparar la invocación al API, y una vez ejecutado el API podemos hacer ya los tests con Postman.



Desde la interfaz

- Para poder codificar nuestros tests **simplemente** tendremos que ir a las pestañas **pre-request script** y **tests**.



The screenshot shows the Postman application interface. On the left, there's a sidebar with tabs for History, Collections, and Trash. Under Collections, there are several items: Elasticsearch Clientes (3 requests), Estado Elasticsearch (3 requests), Personas (1 request), Personas MockServer (1 request), and Random User. The main area displays a collection titled "Obtener persona española". A GET request is listed with the URL `https://randomuser.me/api/?nat=es&gender=male`. Below the request, there are tabs for Params, Authorization, Headers, Body, Pre-request Script, and Tests. The "Pre-request Script" tab is currently selected and contains the following code:

```
1 - pm.test("API Responde con 200", function() {  
2     pm.response.to.have.status(200);  
3 };  
4  
5 - pm.test("Genero masculino", function() {  
6     pm.expect(pm.response.json().results[0].gender).to.equal("male");  
7 };  
8  
9 - pm.test("Nacionalidad Española", function() {  
10    pm.expect(pm.response.json().results[0].nat).to.equal("ES");  
11});
```

Creando un test

- ▶ Un test en Postman tiene **tres partes**:
 - ▶ **Descripción** del Test: un texto que describe la finalidad del test.
 - ▶ **Código** asociado al Test: código que utiliza el api de Postman pm para poder codificar los tests unitarios.
 - ▶ **Resultado** de la ejecución del Test: información sobre los tests que se han ejecutado correctamente, incorrectamente o que se han obviado.

La API de Postman

- ▶ Para codificar los tests con Postman deberemos de conocer un poco el API que nos ofrecen para poder construir los tests.
- ▶ Cada uno de los tests se ejecuta con **el objeto pm** y en concreto **con el método .test()**. Así tendremos la siguiente estructura por cada uno de los test:

```
pm.test("Descripción Funcionalidad a Probar", function(){  
    // Código que valida la prueba del test  
});
```

La API de Postman

- Para poder acceder al contenido de la respuesta de las invocaciones tenemos el objeto **pm.response** y su método **.json()** que nos permitirá acceder a los elementos de la respuesta en JSON.

```
pm.response.json();
```

- Si nuestra estructura es:

```
{  
  "nombre": "Víctor",  
  "telefono": 913456762  
}
```

- Podremos acceder al nombre escribiendo:

```
nombre = pm.response.json().nombre;
```

La API de Postman

- El último método que tenemos que controlar es el que nos permite realizar una comprobación de contenido, este es **pm.expect**:

```
pm.expect(valor).to.equal(valor);
```

- Así, siguiendo el ejemplo anterior podríamos comprobar que la respuesta viene con un nombre que es "Víctor" de la siguiente manera:

```
pm.expect(nombre).to.equal("Víctor");
```

- El test completo podría quedar de la siguiente forma:

```
pm.test("Nombre es Víctor", function(){  
    nombre = pm.response.json().nombre;  
    pm.expect(nombre).to.equal("Víctor");  
});
```

Automatizando la ejecución con Newman

- ▶ Instalación de Newman.
- ▶ Ejecución de Newman.
- ▶ Usando entornos en Newman.

Instalación de Newman

- ▶ Trabajando sobre nuestro manual de editorial:
 - ▶ Capítulo 8. Running API Tests in CI with Newman.
 - ▶ Sección. Installing Newman.

Installing Newman

Newman is built into **Node.js**. Node.js is a JavaScript runtime environment. JavaScript generally runs in web browsers, but Node.js provides an environment for running JavaScript outside a web browser. With Node.js, you can run JavaScript code on servers or even directly on your own computer. This lets developers write applications directly in JavaScript, which is exactly what Newman is. One other major feature of Node.js is the package manager. The **Node.js package manager**, or **npm**, makes it easy for programmers to publish and share their Node.js code. This package manager makes installing and setting up things such as Newman dead simple, but in order to use it, you will need to install Node.js.

Ejecución de Newman

- ▶ Trabajando sobre nuestro manual de editorial:
 - ▶ Capítulo 8. Running API Tests in CI with Newman.
 - ▶ Sección. Running Newman.

Running Newman

Now that you have Newman installed, let's run a command in it to make sure everything is working correctly. The easiest way to do that is by running a collection in Postman, but first, you will need a collection to run. You can create a simple collection in Postman and then export it by following these steps:

1. In Postman, create a new collection called **Newman Test** and add a request to the collection called **Test GET**.
2. Set the request URL to `https://postman-echo.com/get?test=true`.
3. Save the request and then click on the **View more actions** menu beside the

Usando entornos en Newman

- ▶ Trabajando sobre nuestro manual de editorial:
 - ▶ Capítulo 8. Running API Tests in CI with Newman.
 - ▶ Sección. Using environments in Newman.

Using environments in Newman

Often, a collection will have variables defined in an environment. Requests in that collection will need those variables when they are run, but the values don't get exported with the collection. To see this in practice, you can modify the **Test GET** request in the Newman Test collection that you made earlier by following these steps:

1. Create a new environment called Newman Test Env.
2. Modify the URL field of the **Test GET** request to set the `https://postman-echo.com` part to a variable called `baseUrl`.

Save that variable in the environment that you just created

Data-Driven Testing

- ▶ Podemos cargar datos sobre las variables de nuestros tests:

The screenshot shows the 'Choose how to run your collection' section of the Postman Runner. On the left, under 'RUN ORDER', a single API endpoint 'GET Users' is selected. On the right, the 'Choose how to run your collection' section is displayed with the following settings:

- Run manually** (radio button selected): Run this collection in the Collection Runner.
- Schedule runs**: Periodically run collection at a specified time on the Postman Cloud.
- Automate runs via CLI**: Configure CLI command to run on your build pipeline.

Run configuration

Iterations: Set to 1.

Delay: Set to 0 ms.

Data: A 'Select File' button.

Advanced settings: An expandable section.

Run JSON Api: A large orange button at the bottom.

SOAP UI

- ▶ Introducción a SOAP UI.
- ▶ Proyectos y definiciones de suites.

Introducción a SOAP UI

- ▶ Introducción.
- ▶ Instalación de la herramienta.
- ▶ Pruebas que se pueden realizar con SoapUI.
- ▶ Tecnologías que soporta SoapUI.
- ▶ Automatización con SoapUI.
- ▶ Ecosistema alrededor de SoapUI.

Introducción

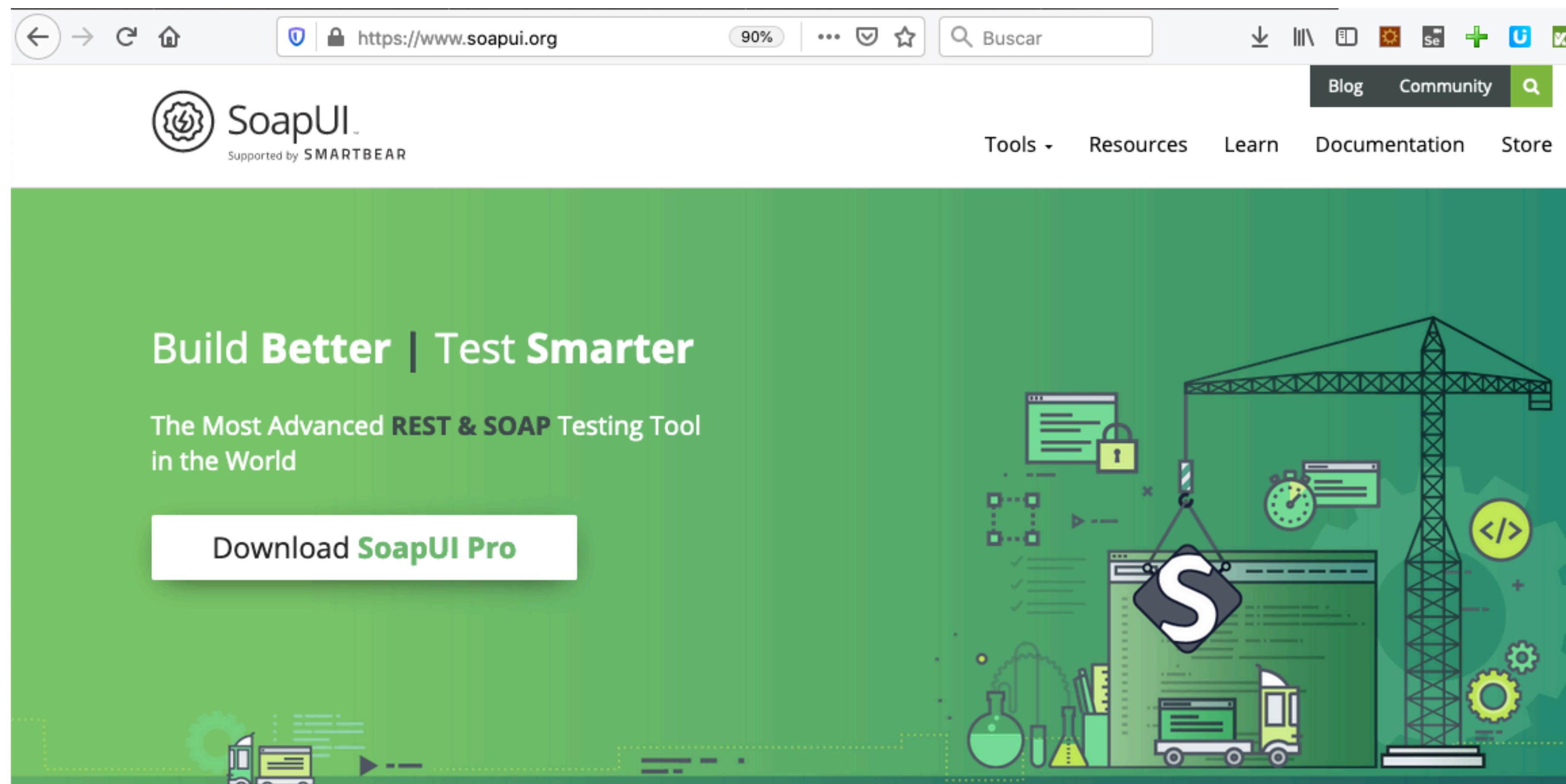
- ▶ En el ecosistema de tecnologías y software empresarial de la actualidad, ninguna aplicación puede funcionar aislada, la posibilidad de comunicarse con otras aplicaciones internas e, inclusive, externas de la organización es obligatoria.
- ▶ El mecanismo de mayor adopción utilizado para estos intercambios es el de los Webservices (Servicios Web). Prácticamente se ha convertido en un **estándar de consideración obligatoria** al diseñar y desarrollar aplicaciones empresariales.
- ▶ SoapUI es la **herramienta de mayor difusión probar webservices** en arquitecturas orientadas a servicios (SOA) y Representational State Transfers (REST).
- ▶ Fue lanzada en 2005 como herramienta de software libre (Open Source) y desde entonces ha sido descargada más de 2 millones de veces.

Introducción

- ▶ Con SoapUI puedes probar:
 - ▶ Webservices en los protocolos SOAP, REST, JMS y AMF,
 - ▶ Además, puedes realizar llamadas HTTP(S) para aplicaciones web y JDBC para bases de datos.

Introducción

- La url <https://www.soapui.org/> nos lleva a la web del producto en su versión open source:



Introducción

- En la web encontramos documentación muy útil y, entre otras cosas, una comparativa de la versión open source y la versión profesional:

Features	Open Source	Professional
API Functional Testing	Create simple, ad hoc API functional tests	 Create & automate advanced API tests, with real world data <input checked="" type="checkbox"/>
API Performance Testing	Quick API load testing for beginners	 Powerful API performance testing in minutes <input checked="" type="checkbox"/>
API Mocking and Virts	Simulate static API behavior during testing	 Create dynamic API snapshots to enable parallel testing & development <input checked="" type="checkbox"/>
API Security Testing	Assess API vulnerabilities and validate security	 Simulate common API attacks with pre-built security scans <input checked="" type="checkbox"/>

Instalación de la herramienta

- ▶ Descarga de la herramienta.
- ▶ Requisitos del sistema.
- ▶ Procedimiento de instalación.
- ▶ La aplicación de un vistazo.

Descarga de la herramienta

- Para ello primero tenemos que **descargarla**, la siguiente url nos lo permite:
 - <https://www.soapui.org/downloads/soapui.html>

Professional Tools

Downloads

SoapUI Open Source

SoapUI Pro

LoadUI Pro

Latest Releases

Maintenance Builds

Getting Started with SoapUI

Documentation

Learn

Store

Resources

Download the Most Advanced API Testing Tool on the Market

With an improved interface and feature set, you can immediately switch to SoapUI Pro and pick up right where you left off in SoapUI. It's as seamless as it can get!

SoapUI Pro

Get the most advanced functional testing tool for REST and SOAP APIs.

[Download SoapUI Pro](#)

[Learn More](#)

SoapUI Open Source

Get the open source version of the most widely used API testing tool in the world. Read about the latest release [here](#).

[Download SoapUI Open Source](#)

[Alternate Platforms](#)



Descarga de la herramienta

- ▶ Esta disponible para diferentes plataformas:

Professional Tools

Downloads

- SoapUI Open Source
- SoapUI Pro
- LoadUI Pro
- Latest Releases**
 - Latest Release
 - Release Notes
 - Release History
- Maintenance Builds

Getting Started with SoapUI

Documentation

Learn

Store

Resources

Latest Release of SoapUI

Share this article: [Twitter](#) [Facebook](#) [Google+](#) [LinkedIn](#) [Email](#)

Latest **SoapUI Open Source** Downloads (Version 5.5.0)

 Linux

Linux Installer (64-bit)

Download 

 Windows

Windows Installer (64-bit)

Download 

 Mac OS

Mac OS X Installer (64-bit)

Download 

Requisitos del sistema

- ▶ La siguiente url detalla esta información:
 - ▶ <https://www.soapui.org/open-source/system-requirements.html>
- ▶ Procesadores con 32-bit o 64-bit con 1GHz o superior.
- ▶ 512MB de RAM.
- ▶ Alrededor de 200MB de disco duro para la instalación (SoapUI y HermesJMS).
- ▶ Java 7 o superior.

Procedimiento de instalación

- ▶ La siguiente url describe el procedimiento de instalación en los diferentes sistemas operativos:
 - ▶ <https://www.soapui.org/getting-started/installing-soapui.html>

Installing SoapUI

Share this article: [!\[\]\(4d7832f32da92348575447a4aae88cd6_img.jpg\)](#) [!\[\]\(fc7e370f8a58401c31dfa0cc8c0e3085_img.jpg\)](#) [!\[\]\(f3fba8e1b1aae2296a356b71b867bb24_img.jpg\)](#) [!\[\]\(82714054fd3e8d25b757fe0b3cb5e40e_img.jpg\)](#) [!\[\]\(db3674719270ee749465c8f2209740ae_img.jpg\)](#) [!\[\]\(96f6efb355f2c2a120bbbbb8d4f66ffa_img.jpg\)](#)

SoapUI Installation Guides

SoapUI is cross-platform, and can be used on either Windows, Mac or Linux/Unix, according to your needs and preferences. See the appropriate article for your environment:



Linux



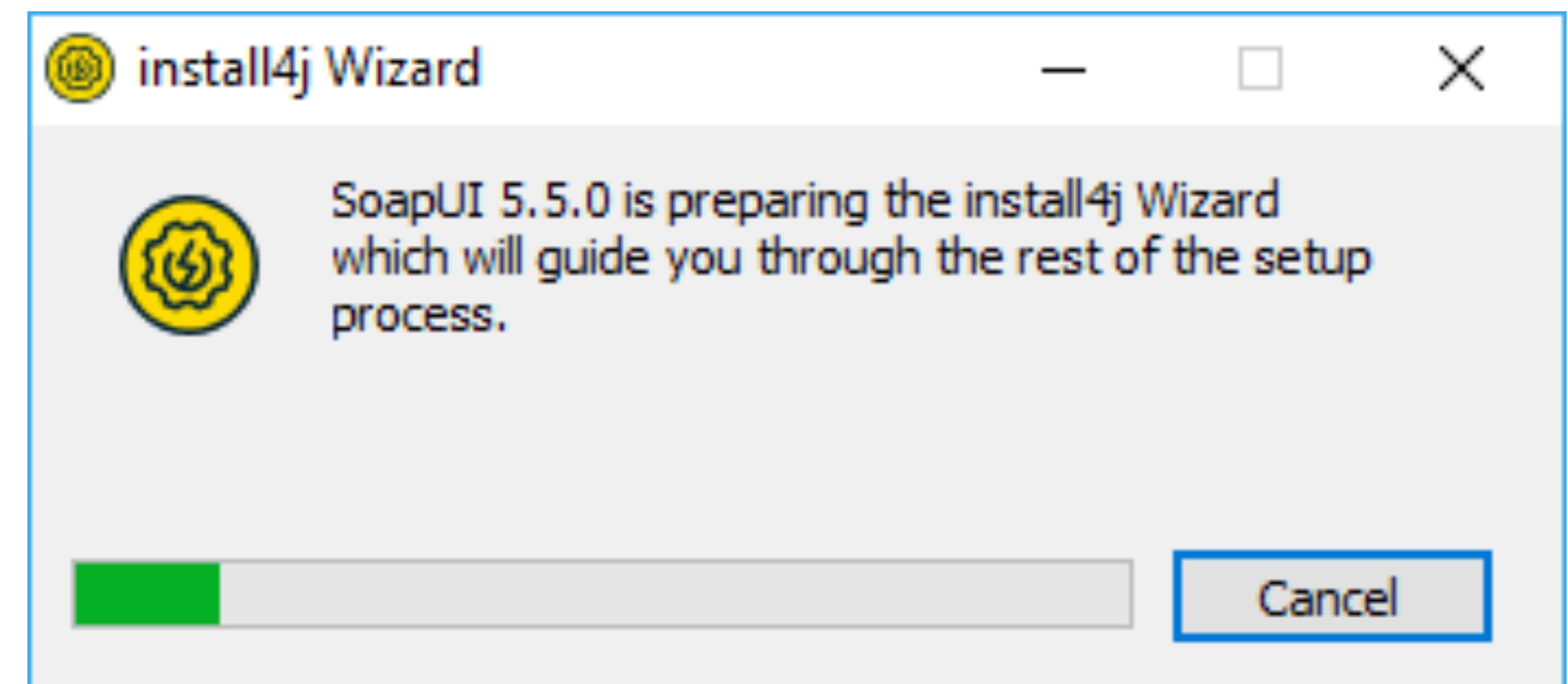
Windows



Mac OS

Procedimiento de instalación

- ▶ Lo describimos para Windows:
 - ▶ Doble click sobre el ejecutable:



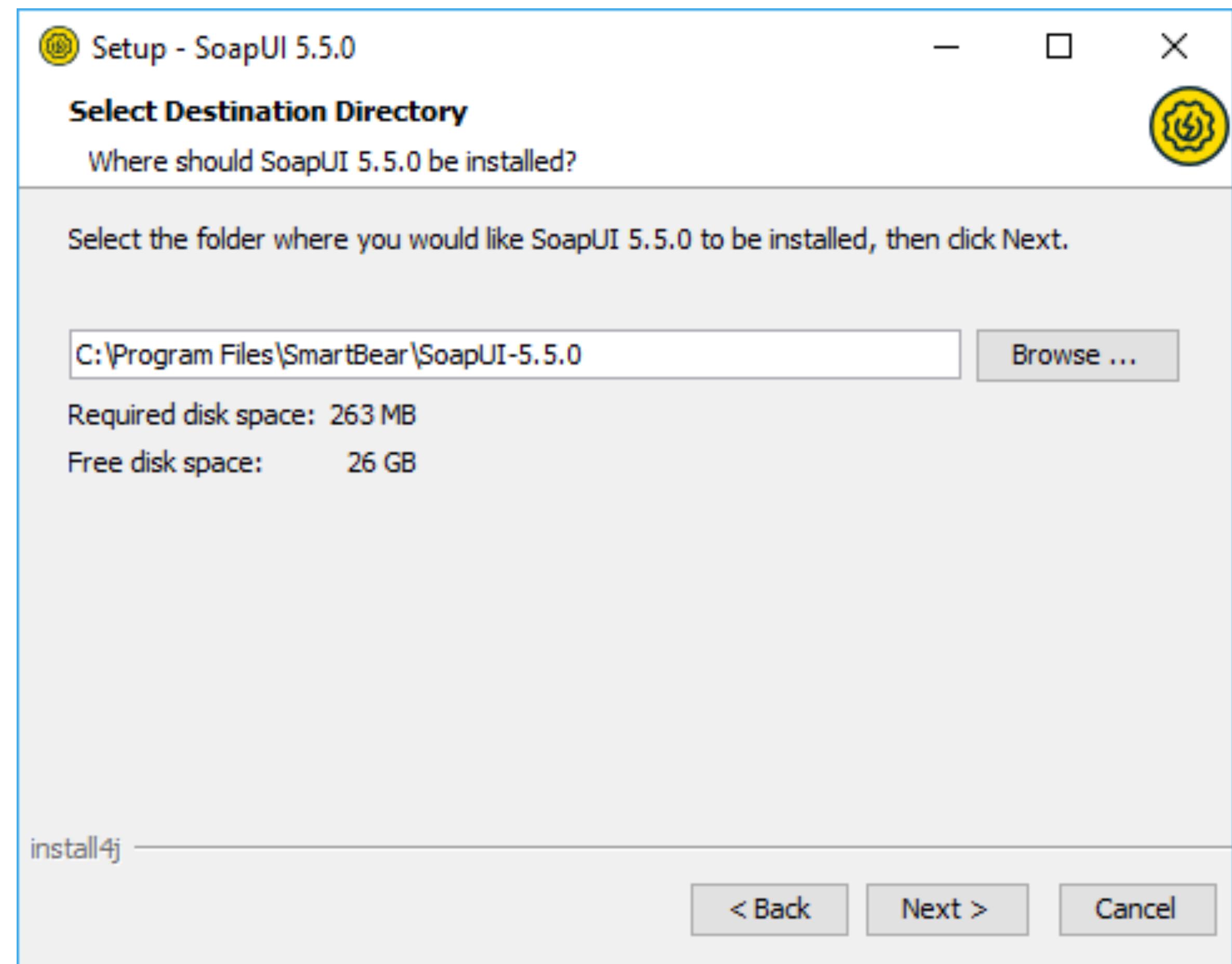
Procedimiento de instalación

- Aparece el asistente de instalación y se hace click en siguiente:



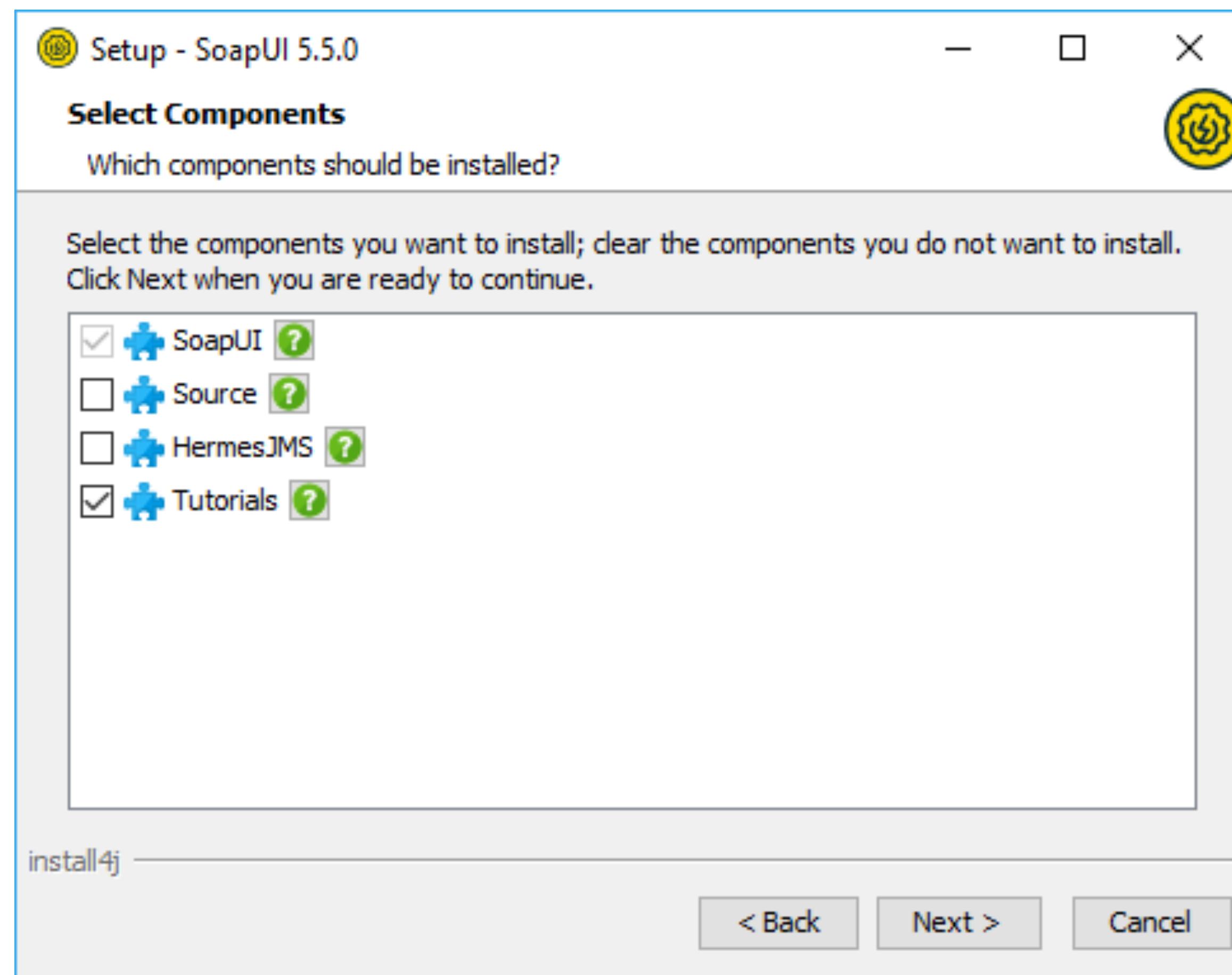
Procedimiento de instalación

- Se selecciona la carpeta de destino y se hace click en siguiente:



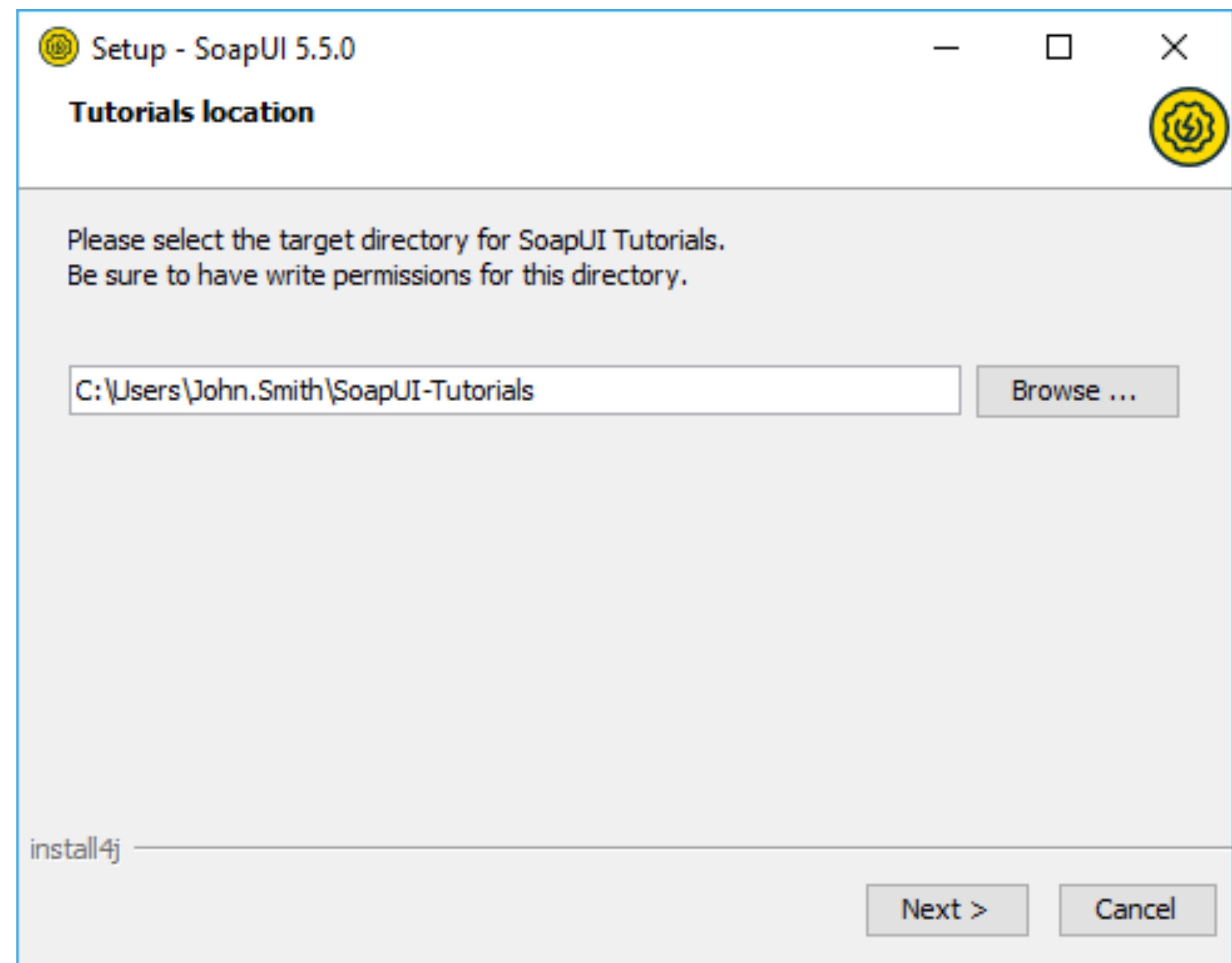
Procedimiento de instalación

- Se seleccionan los elementos que queremos instalar y se hace click en siguiente:



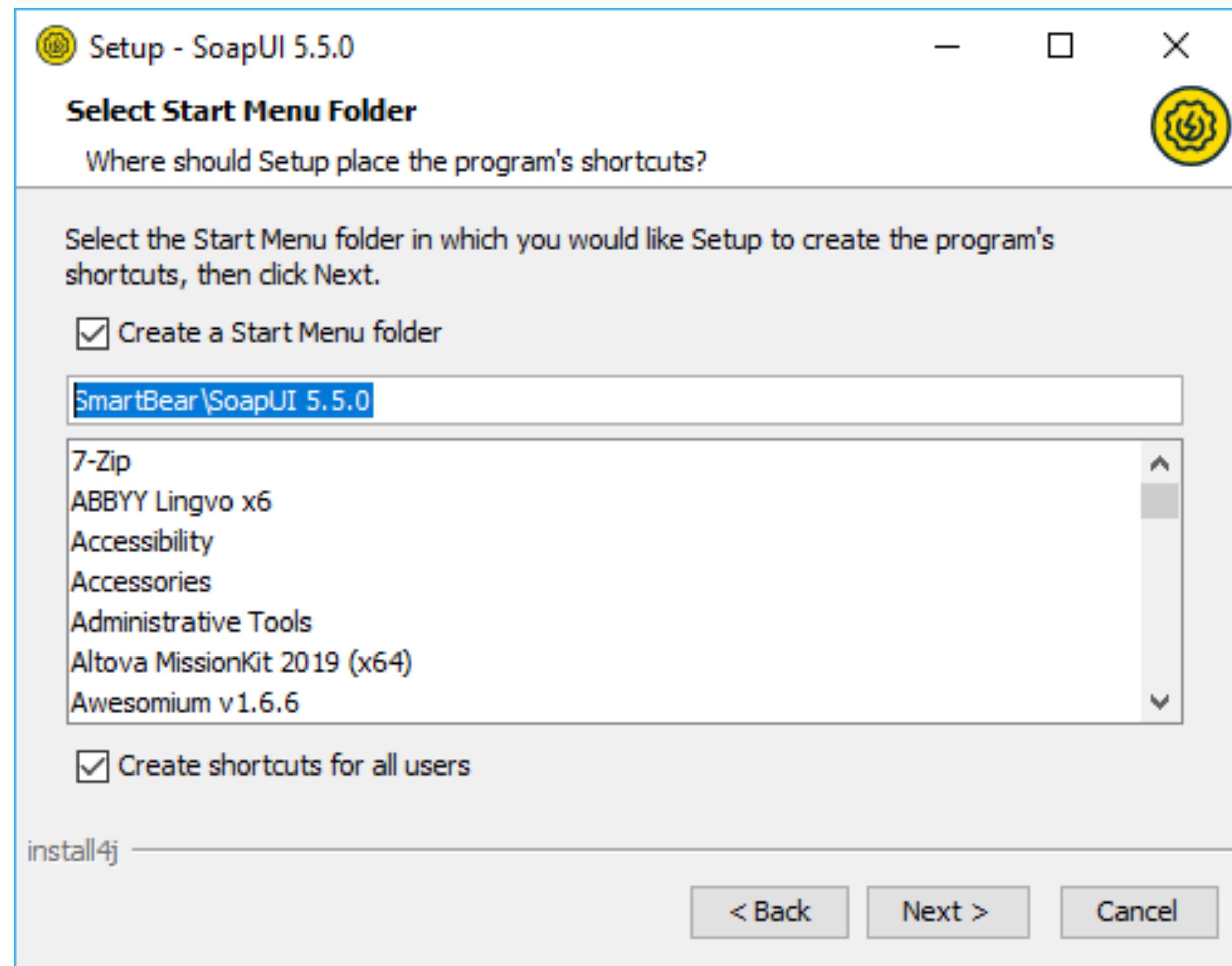
Procedimiento de instalación

- Se selecciona la ruta de los tutoriales si se ha seleccionado esa opción:



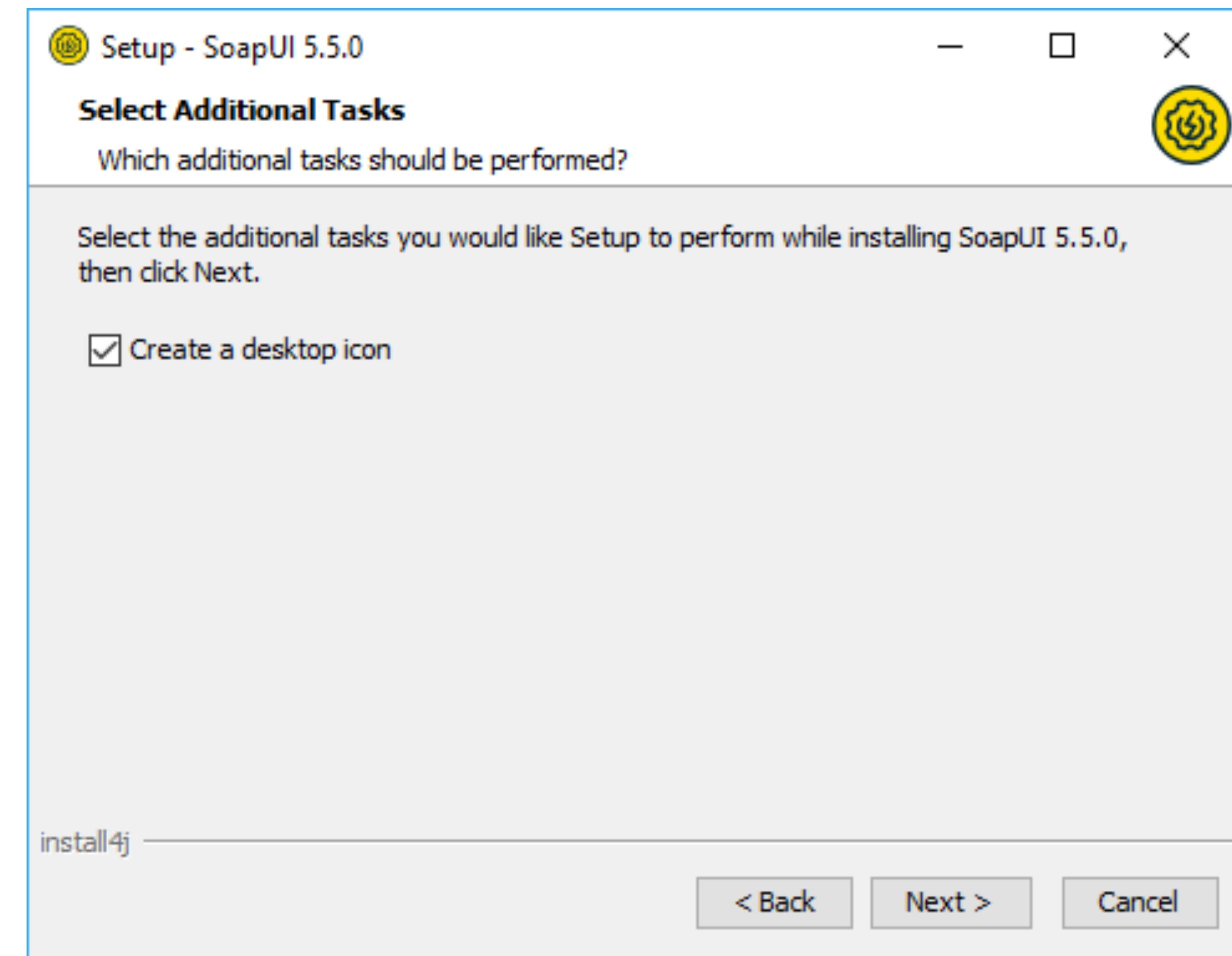
Procedimiento de instalación

- ▶ Se selecciona el menú de inicio:



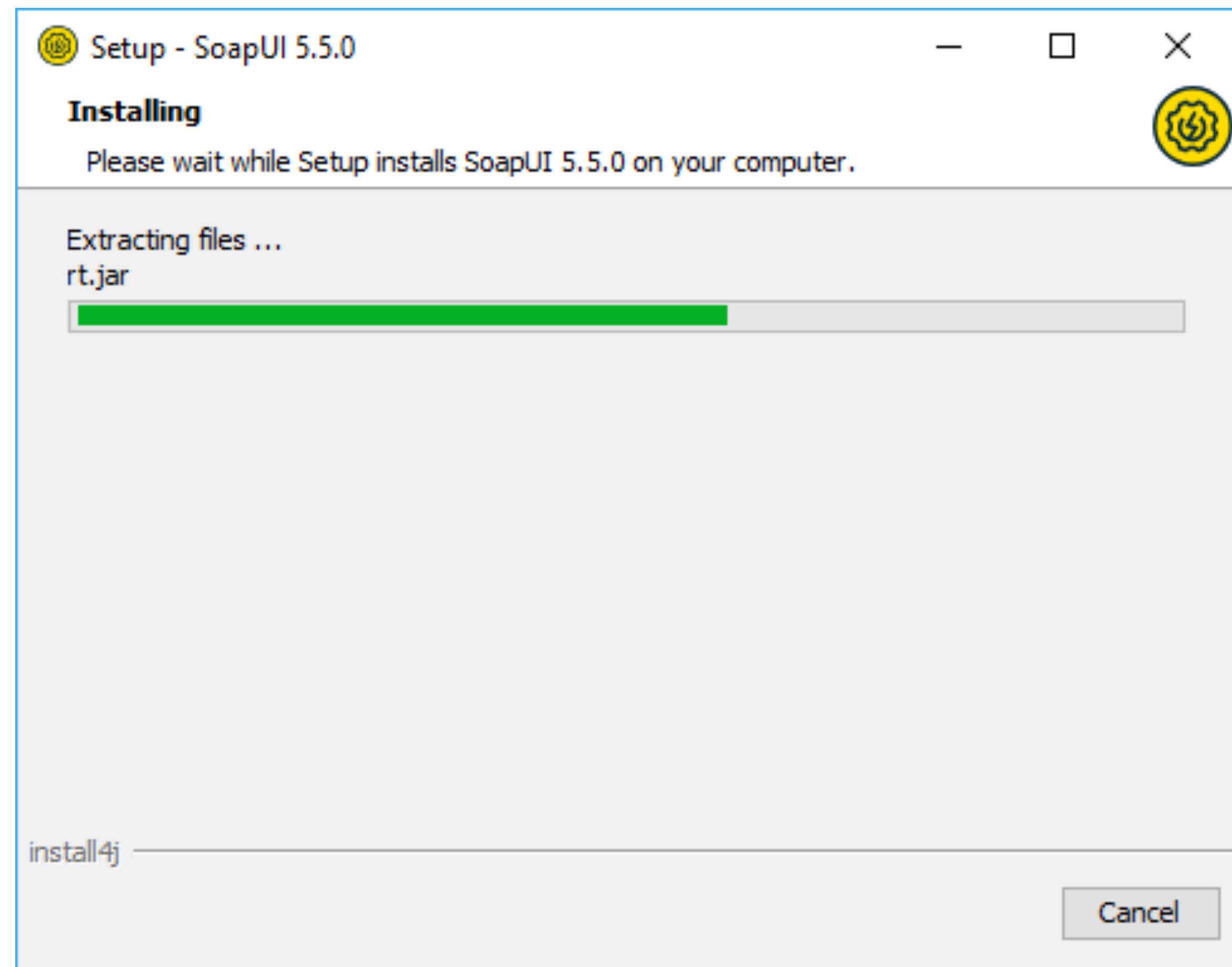
Procedimiento de instalación

- Se decide si queremos incluir un icono en el escritorio:



Procedimiento de instalación

- Una vez realizada esta operación, se inicia el proceso de instalación:



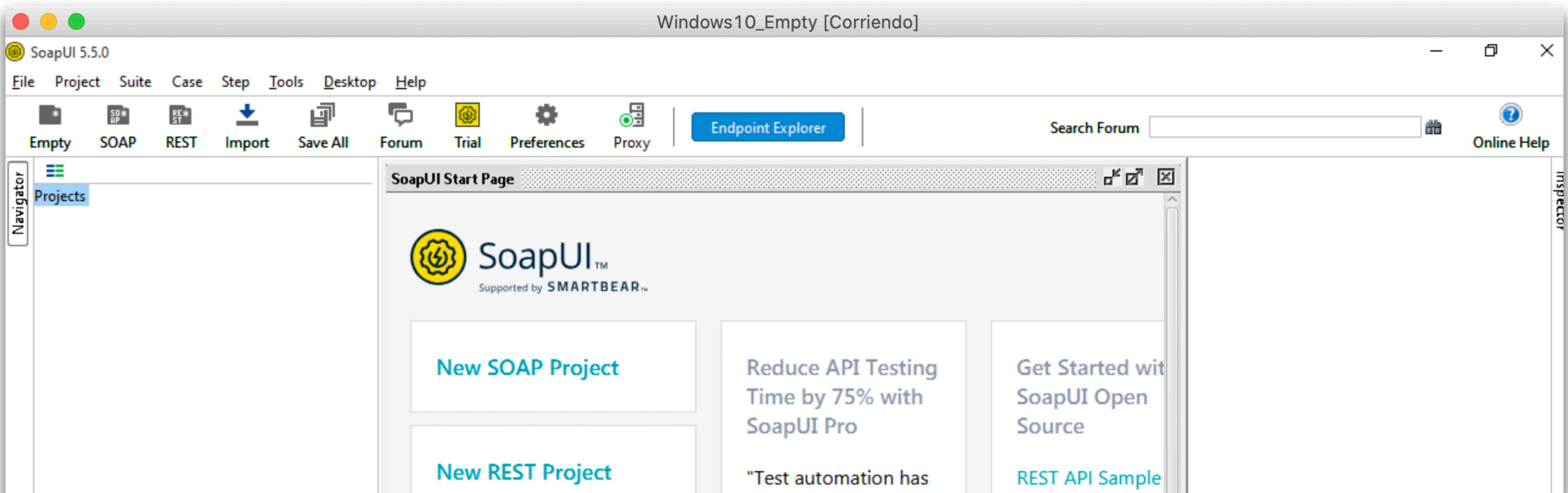
Procedimiento de instalación

- Una vez finalizada la instalación, aparece la siguiente pantalla:

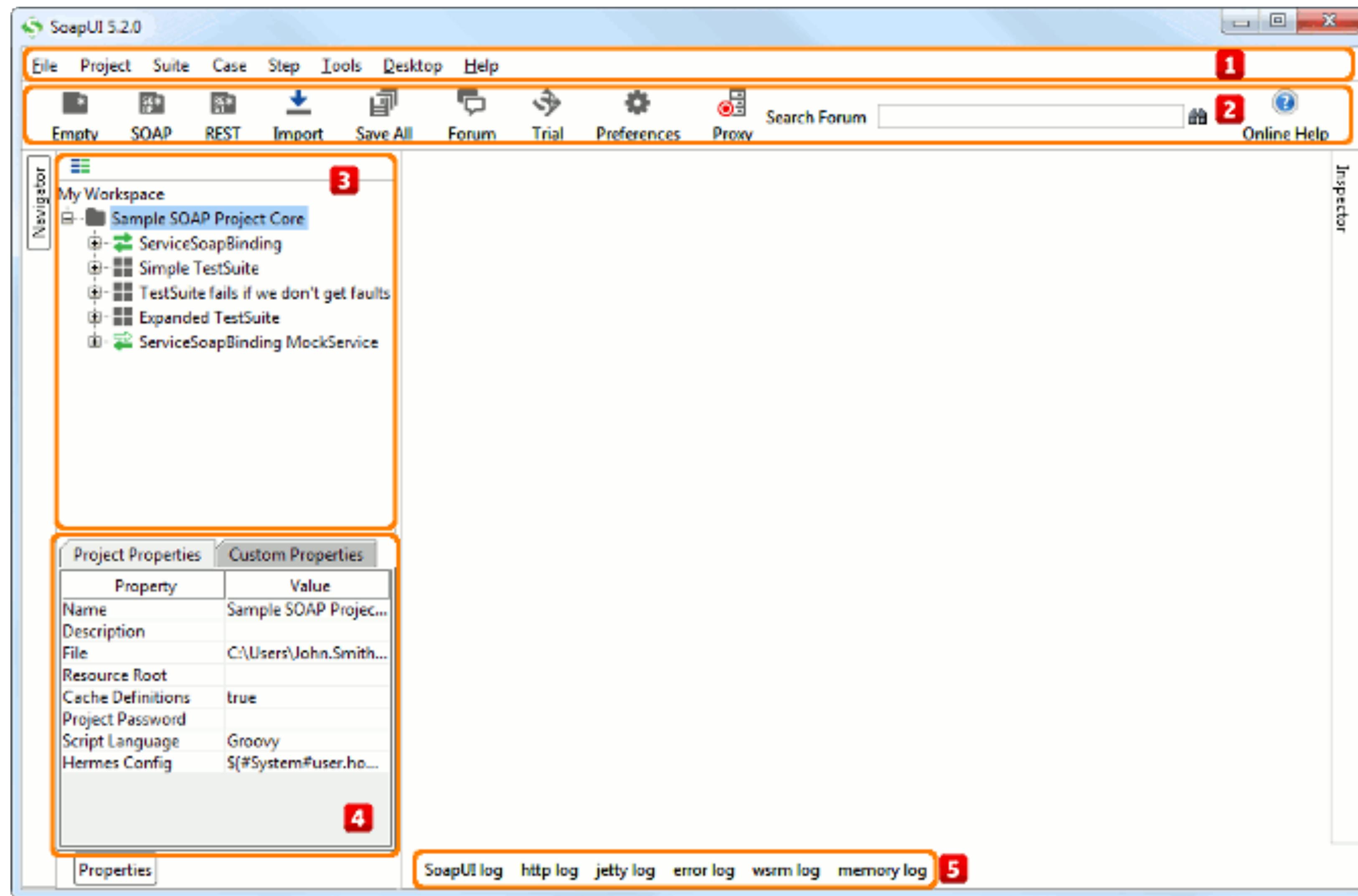


Procedimiento de instalación

- Si decidimos que la aplicación arranque nos encontraremos con la interface de la aplicación:



La aplicación de un vistazo



La aplicación de un vistazo

- ▶ Toolbar principal -> 1
- ▶ Toolbar en formato icono -> 2
- ▶ Panel de navegación -> 3
- ▶ Panel de propiedades -> 4
- ▶ Inspeccionador de los -> 5

Pruebas que se pueden realizar con SoapUI

- ▶ Introducción.
- ▶ Pruebas funcionales.
- ▶ Simulación de servicios.
- ▶ Pruebas de seguridad.
- ▶ Pruebas de carga.

Pruebas que se pueden realizar con SoapUI

- ▶ SoapUI permite la creación y ejecución automatizada de distintos tipos de pruebas de software, incluyendo:
 - ▶ Testing funcional,
 - ▶ Pruebas de regresión,
 - ▶ Simulación de servicios (Mocking),
 - ▶ Pruebas de carga y
 - ▶ Pruebas de seguridad.

Pruebas funcionales

- ▶ Las pruebas funcionales están orientadas a validar y mejorar la calidad de los webservices y aplicaciones. Con SoapUI puedes:
 - ▶ Crear y ejecutar escenarios de pruebas complejos con una interfaz gráfica Drag and Drop.
 - ▶ No necesitas tener conocimientos de programación para definir casos de prueba en SoapUI.
 - ▶ Crear Suites de pruebas y añadir casos de prueba para mejor organización.
 - ▶ Definir escenarios de pruebas complejos, por ejemplo probar un procedimiento, mensajería empresarial y captura de tráfico al mismo tiempo.
 - ▶ Si estás trabajando pruebas en distintos ambientes (ambiente de desarrollo, ambiente de pruebas o producción), puedes establecer las configuraciones de cada ambiente y luego cambiar entre ellos rápidamente.
 - ▶ Permite la ejecución asíncrona de las pruebas.

Simulación de servicios

- ▶ SoapUI permite crear pruebas de webservices antes que estos sean implementados, muy útil para probar aplicaciones que invocan a webservices que no están en el alcance o programas implementados alrededor de webservices.
- ▶ Con SoapUI se puede realizar:
 - ▶ Simulación de comportamientos complejos, pudiendo configurar las respuestas predeterminadas en distintos escenarios.
 - ▶ Creación automática de webservices a partir del WSDL.
 - ▶ Posibilidad de crear Mocks de total cumplimiento con los estándares WSDL, SOAP o HTTP.
 - ▶ Posibilidad de desplegar el MockService a Apache, Tomcat, GlassFish o cualquier otro.

Simulación de servicios

- ▶ (cont.)
 - ▶ Simulación de APIs tanto en SOAP como en REST.
 - ▶ Mocking de contenido estático, Scripting, Soporte a SSL y más.
 - ▶ Obtener estadísticas de uso de los WSDL simulados a nivel de proyecto, caso de pruebas y set de pruebas.

Pruebas de seguridad

- ▶ SoapUI incluye un complemento de pruebas para identificar problemas de seguridad, información que puedes usar para proteger tus webservices y sitios web contra las vulnerabilidades más comunes.

Pruebas de seguridad

- ▶ Con SoapUI puedes:
 - ▶ Enviar SQL maliciosos a tus bases de datos para asegurarte que no son vulnerables a inyecciones de SQL.
 - ▶ Realizar un escaneo contra bombas XML, para validar si tus sistemas son vulnerables a volcados de pila (stack overflow) usando documentos de gran tamaño.
 - ▶ Validar si los webservices exponen los parámetros que utiliza y sus mensajes.
 - ▶ Enviar textos al azar para intentar provocar errores, volcados del buffer, trazas de pila o para encontrar vulnerabilidades de string.
 - ▶ Realizar escaneo de datos borde para determinar si el sistema se comporta erráticamente o muestra información no deseada.
 - ▶ Otros escaneos como de archivo adjunto malicioso, script custom, inyección de XPath, datos inválidos y XML malformados.

Pruebas de carga

- ▶ Con las pruebas de carga (también conocida como pruebas de estrés) puedes validar si tus webservices o interfaces de programa pueden manejar múltiples usuarios concurrentemente.

Pruebas de carga

- ▶ Con SoapUI es posible:
 - ▶ Crear conjuntos de pruebas de carga con interfaz drag and drop.
 - ▶ Integrar con LoadUI (módulo específico para pruebas de carga de estrés).
 - ▶ Recibir Feedback en tiempo real del estatus y resultado de las pruebas.
 - ▶ Crear pruebas de carga a partir de las pruebas funcionales existentes.
 - ▶ Usar distintas estrategias de pruebas carga predefinidas tales como simple, tasa fija, tasa variable, para probar el desempeño bajo diferentes condiciones.

Tecnologías que soporta SoapUI

- ▶ Una de las ventajas de SoapUI es que soporta los principales estándares para webservices e interfaces API existentes en la actualidad, lo cual es muy útil pues permite trabajar con amplia variedad de aplicaciones.
- ▶ SoapUI maneja las siguientes tecnologías:
 - ▶ SOAP / WSDL: Permite importar y validar servicios web basados en WSDL y SOAP. Con el uso del editor no se necesitar ser un técnico, QA y el área de negocio también se pueden involucrar.
 - ▶ REST: Permite agregar recursos REST o métodos custom a los servicios, para luego validar sus respuestas y más.
 - ▶ Web y HTTP(S): Automatización de la creación de escenarios típicos de uso, por medio de la grabación del uso de la aplicación web (Recording) a medida que la utilizas. Luego permite repetir esas acciones, construyendo así pruebas funcionales automatizadas.

Tecnologías que soporta SoapUI

- ▶ (cont.):
 - ▶ AMF: Creación de escenarios para probar Rich Internet Applications (RIAs), identificar cuellos de botella y entonar antes de su salida a producción.
 - ▶ JDBC: Por medio de interfaz intuitiva te permite la exploración y análisis de las bases de datos. Pruebas y verificación de entradas y resultados de cualquier base de datos JDBC, simultáneamente con tus pruebas de servicios web y aplicaciones.
 - ▶ JMS: SoapUI está totalmente integrado con HermesJMS, lo cual te permite enviar y recibir mensajes de texto y binario a una amplia variedad de proveedores JMS.

Automatización con SoapUI

- ▶ Otra de las funcionalidades de SoapUI es la posibilidad de integrarlo prácticamente con cualquier programador de tareas o como parte del proceso de compilación.
- ▶ Inclusive es posible personalizar la ejecución en estos casos omitiendo parámetros, controlando que pruebas ejecutar y cuáles no, que producir como resultado, entre otros.
- ▶ La ejecución de pruebas en SoapUI se puede integrar con las siguientes aplicaciones:

Automatización con SoapUI

- ▶ Maven: Se pueden ejecutar las pruebas y mocks de SoapUI en un entorno Maven, Software para compilación automatizada de bases de código. Inclusive, puedes seleccionar los TestSuites o TestCases que quieras ejecutar, sin necesidad de ejecutar el proyecto completo.
- ▶ Jenkins: SoapUI también facilita la integración continua con Hudson (Herramienta de integración continua de código desarrollada en Java). Como resultado, las pruebas SoapUI se pueden integrar al proceso de construcción Hudson, sin tener que interrumpir el desarrollo de software y facilitando la identificación de los errores.
- ▶ Banboo: SoapUI también permite integrar y ejecutar las pruebas en un Build de Bamboo, servidor de integración Atlassian.

Automatización con SoapUI

- JUnit: Por medio del uso del TestRunner (Inicio de la línea de comando desde SoapUI), se puede ejecutar y controlar la ejecución de pruebas SoapUI desde JUnit.
- ANT: También puedes ejecutar y personalizar las pruebas en un entorno de compilación automatizada ANT, usando las herramientas de línea de comando de SoapUI.
- Otros sistemas: Desde SoapUI se pueden ejecutar pruebas funcionales, de carga y simulaciones de servicios (MockServices) desde cualquier herramienta.

Ecosistema alrededor de SoapUI

- ▶ SoapUI está construida totalmente en la plataforma Java, utilizando Swing para la interfaz gráfica con el usuario. SoapUI es una herramienta multiplataforma.
- ▶ Como herramienta de código abierto, SoapUI posee una gran comunidad y socios a su alrededor. Los usuarios de SoapUI pueden desarrollar sus propias funcionalidades como Plugins de SoapUI.
- ▶ SoapUI cuenta con amplia variedad de plugins desarrollados por terceros, como por ejemplo Agiletestware, SOA DataPro, TestMaker y Pegamento.
- ▶ Si el usuario opta por la versión comercial de SoapUI (SoapUI Pro) además se puede contar con el soporte del equipo de SmartBear.

Ecosistema alrededor de SoapUI

- ▶ SoapUI puede instalarse para trabajar conjuntamente con ambientes de desarrollo integrados (IDE) como:
 - ▶ IntelliJ IDEA: Posibilita el acceder a todas las funcionalidades de SoapUI desde el entorno IntelliJ IDEA, creando un único entorno para el desarrollo y pruebas de tus webservices.
 - ▶ NetBeans: Integración de las funcionalidades de SoapUI al entorno NetBeans por medio de plugins, logrando también la combinación de desarrollo y pruebas de webservices en el mismo entorno.
 - ▶ Eclipse: La tecnología SoapUI Nature, brinda el acceso a SoapUI desde un proyecto Java en Eclipse.

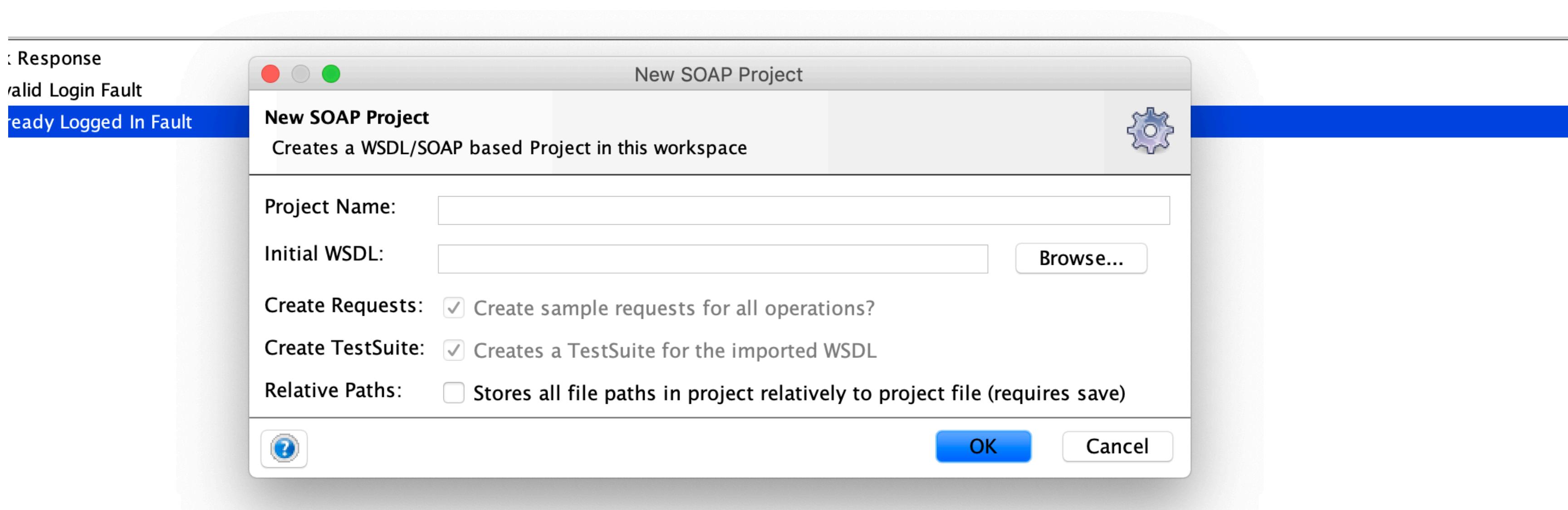
Proyectos y definiciones de suites

- ▶ Proyecto SOAP.
- ▶ Proyecto REST.

Proyecto SOAP

- ▶ Creación de un proyecto SOAP.
- ▶ Inspección del proyecto.
- ▶ Ejecución de peticiones.
- ▶ Definición de MockServices.

Creación de un proyecto SOAP



- ▶ Debemos incluir el fichero wsdl que define el servicio.
- ▶ El asistente nos permite crear peticiones y conjuntos de tests.

Inspección del proyecto

The screenshot shows the SoapUI interface for a service named "SampleServiceSoapBinding". The "Overview" tab is selected. The "WSDL Definition" section shows the following details:

WSDL URL	file:///Users/ematiz/Documents/PROYECTOS/2019/00_CURSOS_IC_AVANTE/doc/manual/soapUI/SoapUI-Tutorials/WSDL-WADL/sample-service.wsdl
Namespace	http://www.soapui.org/sample/
Binding	SampleServiceSoapBinding
SOAP Version	SOAP 1.1
Style	RPC
WS-A version	NONE

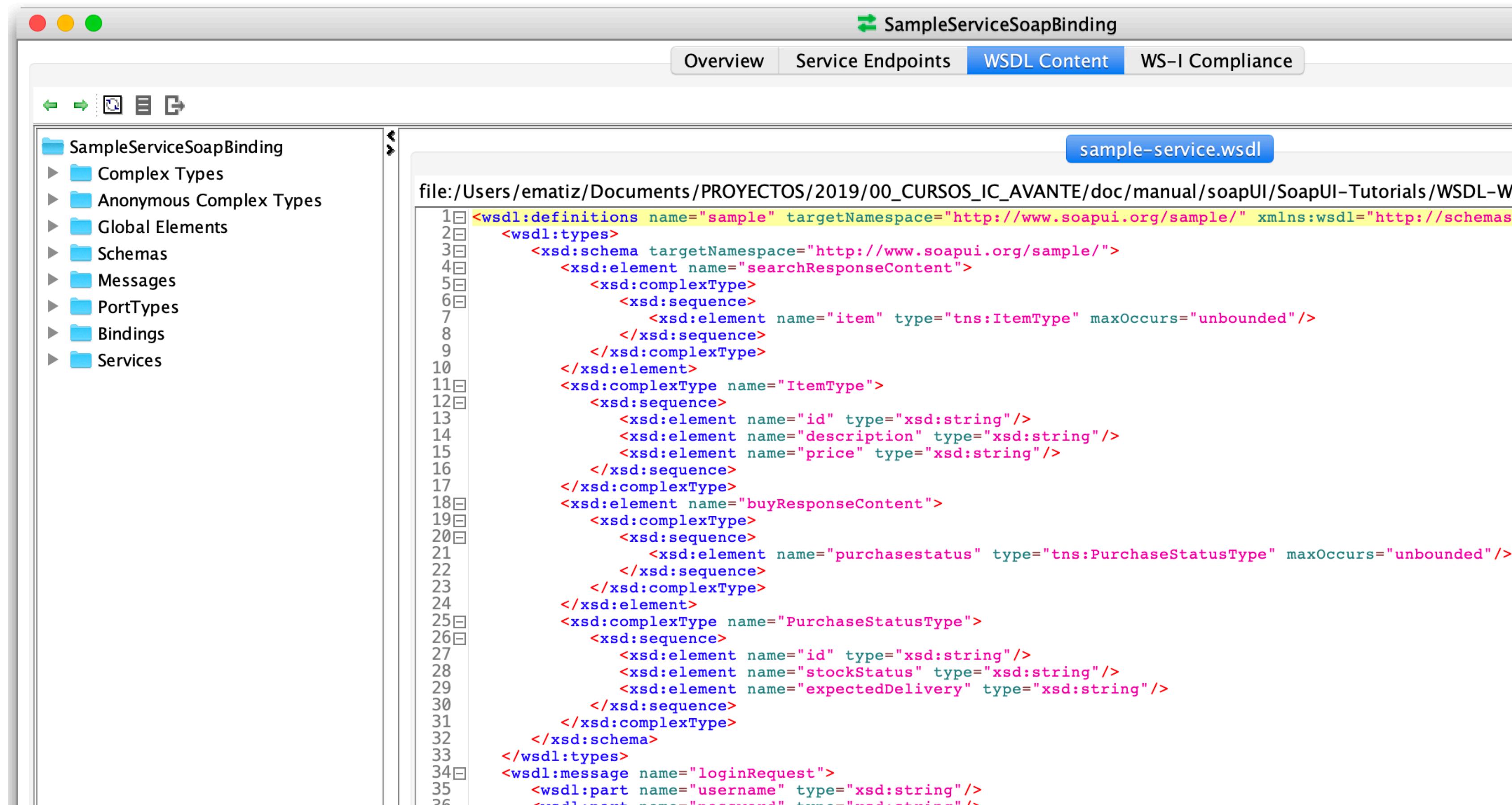
The "Definition Parts" section shows:

sample-service.wsdl	file:///Users/ematiz/Documents/PROYECTOS/2019/00_CURSOS_IC_AVANTE/doc/manual/soapUI/SoapUI-Tutorials/WSDL-WADL/sample-service.wsdl
---------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

The "Operations" section contains a table:

Name	Use	One-...	Action
buy	Literal	false	http://www.soapui.org/sample/buy
login	Literal	false	http://www.soapui.org/sample/login
logout	Literal	false	http://www.soapui.org/sample/logout
search	Literal	false	http://www.soapui.org/sample/search

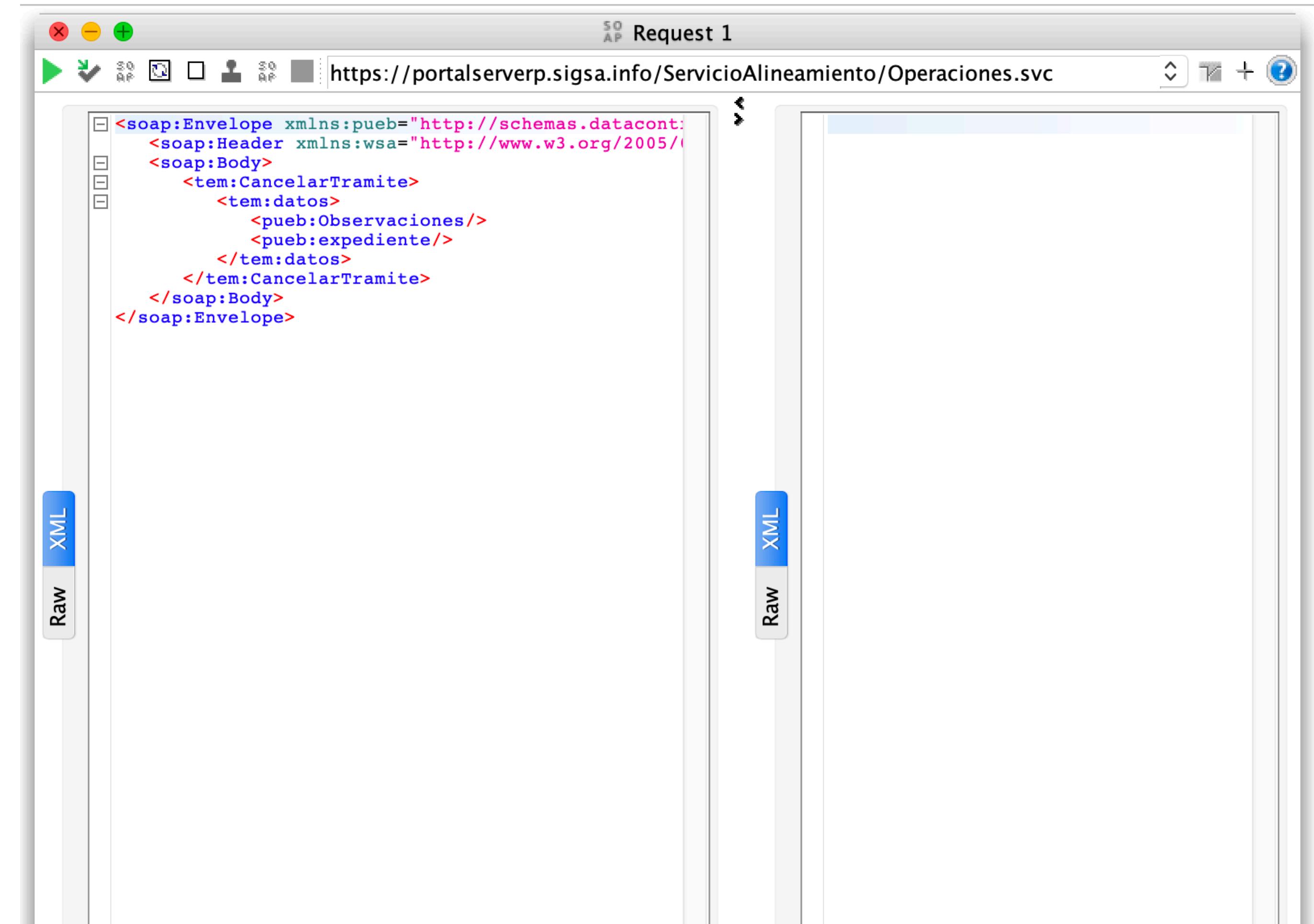
Inspección del proyecto



The screenshot shows the SoapUI application interface. The title bar reads "SampleServiceSoapBinding". The top menu bar has tabs: Overview, Service Endpoints, WSDL Content (which is selected and highlighted in blue), and WS-I Compliance. On the left, there's a tree view of the project structure under "SampleServiceSoapBinding", including Complex Types, Anonymous Complex Types, Global Elements, Schemas, Messages, PortTypes, Bindings, and Services. The main pane displays the WSDL XML code for "sample-service.wsdl". The XML code defines a service named "sample" with various complex types, global elements, and messages, including "loginRequest" and "searchResponseContent". Lines 1 through 36 are visible in the code editor.

```
file:/Users/ematiz/Documents/PROYECTOS/2019/00_CURSOS_IC_AVANTE/doc/manual/soapUI/SoapUI-Tutorials/WSDL-W...
1<wsdl:definitions name="sample" targetNamespace="http://www.soapui.org/sample/" xmlns:wsdl="http://schemas...
2  <wsdl:types>
3    <xsd:schema targetNamespace="http://www.soapui.org/sample/">
4      <xsd:element name="searchResponseContent">
5        <xsd:complexType>
6          <xsd:sequence>
7            <xsd:element name="item" type="tns:ItemType" maxOccurs="unbounded"/>
8          </xsd:sequence>
9        </xsd:complexType>
10       </xsd:element>
11       <xsd:complexType name="ItemType">
12         <xsd:sequence>
13           <xsd:element name="id" type="xsd:string"/>
14           <xsd:element name="description" type="xsd:string"/>
15           <xsd:element name="price" type="xsd:string"/>
16         </xsd:sequence>
17       </xsd:complexType>
18       <xsd:element name="buyResponseContent">
19         <xsd:complexType>
20           <xsd:sequence>
21             <xsd:element name="purchasestatus" type="tns:PurchaseStatusType" maxOccurs="unbounded"/>
22           </xsd:sequence>
23         </xsd:complexType>
24       </xsd:element>
25       <xsd:complexType name="PurchaseStatusType">
26         <xsd:sequence>
27           <xsd:element name="id" type="xsd:string"/>
28           <xsd:element name="stockStatus" type="xsd:string"/>
29           <xsd:element name="expectedDelivery" type="xsd:string"/>
30         </xsd:sequence>
31       </xsd:complexType>
32     </xsd:schema>
33   </wsdl:types>
34   <wsdl:message name="loginRequest">
35     <wsdl:part name="username" type="xsd:string"/>
36     <wsdl:part name="password" type="xsd:string"/>
```

Ejecución de peticiones

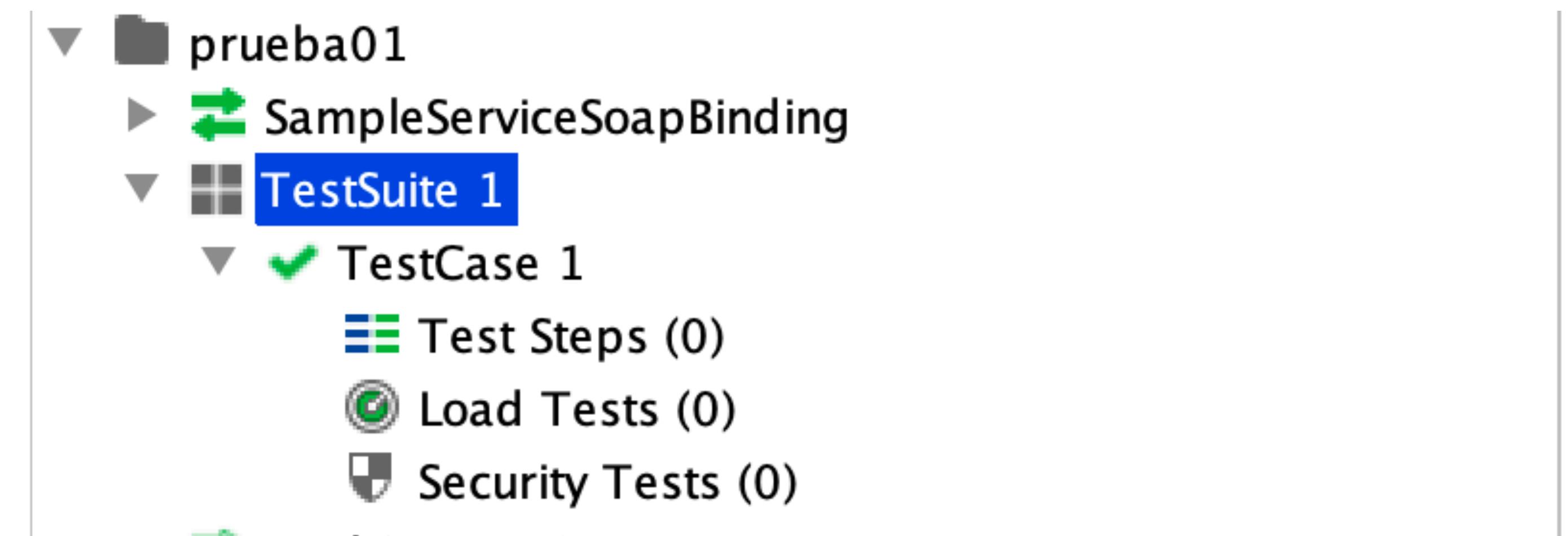


Trabajando con tests

- ▶ Introducción.
- ▶ Pasos dentro de un test.

Introducción

- ▶ Organizamos nuestros tests dentro de suites:



Pasos dentro de un test

- ▶ Peticiones de todo tipo.
- ▶ Properties.
- ▶ Properties transfer.

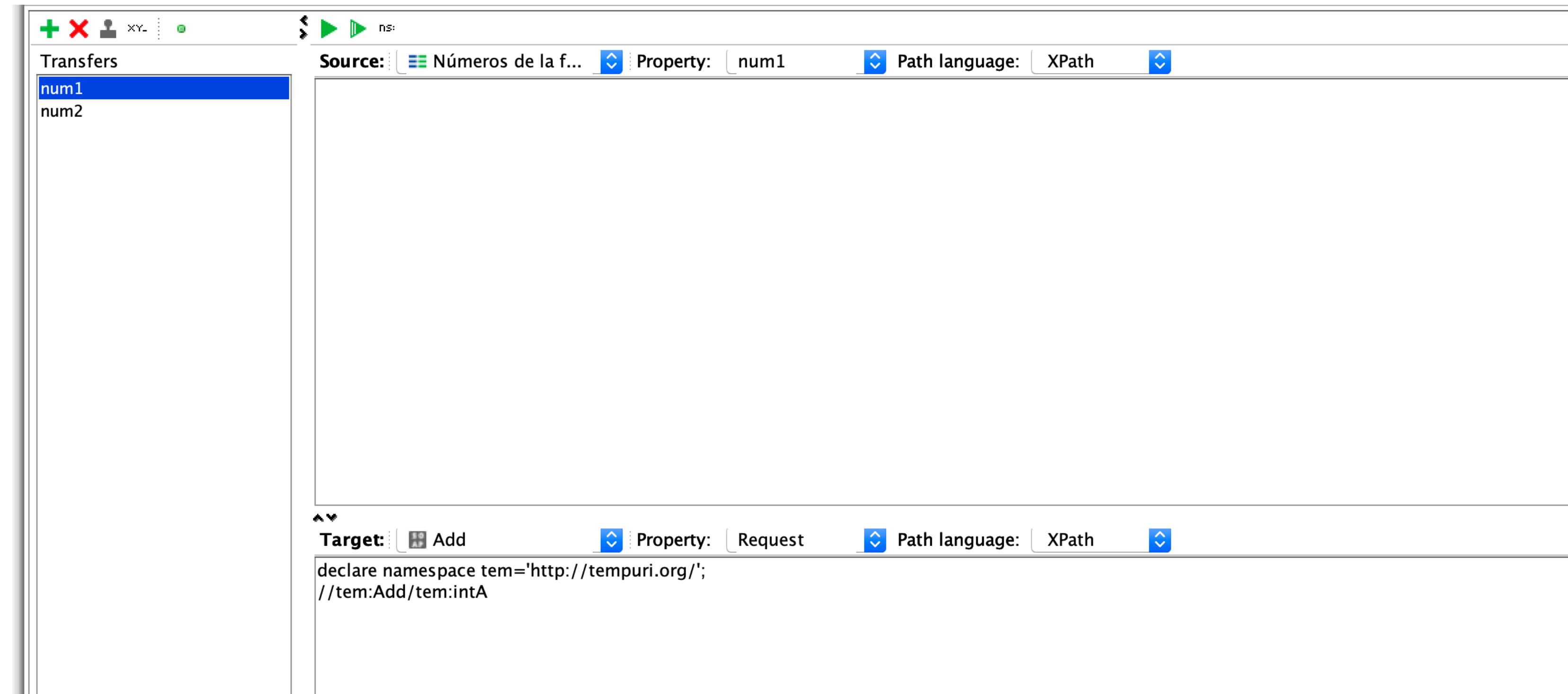
Properties

The screenshot shows a software interface for managing test properties. At the top, there are tabs: 'Mocks' (selected), 'Source', 'Run', 'Property' (highlighted in blue), 'Response', and 'Path lang'. On the left, there are two items: 'ado1' and 'ado2', each with a red, yellow, and green status indicator. Below these are three colored icons: red, yellow, and green. The main area contains a title 'declare namespace tem='http://tempuri.org/';' and a subtitle 'Números de la formula'. A toolbar below the subtitle includes icons for adding (+), deleting (X), up/down (arrow), copy (copy), paste (paste), and save (Save). There is also a 'Load from:' field and a 'Save' button. A table lists properties:

Name	Value
num1	2
num2	5
num3	7
res	49

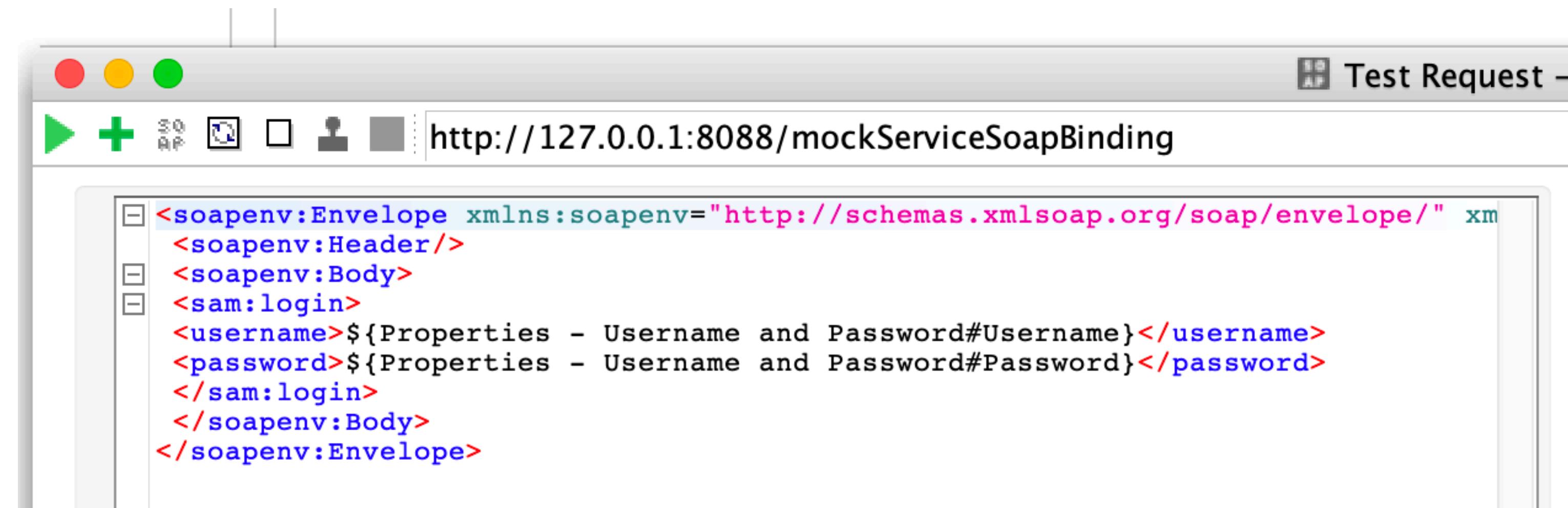
At the bottom of the window, there is a checkbox labeled 'Fail transfer on error' with a checked state.

Properties transfer



Expansión de propiedades

- ▶ Podemos utilizar las propiedades siguiendo otra estrategia:
 - ▶ \${NOMBRE_PROPERTY#nombrePropiedad}



The screenshot shows the SoapUI interface with a 'Test Request' window. The URL is set to `http://127.0.0.1:8088/mockServiceSoapBinding`. The message body contains a SOAP envelope with a login request. The 'username' and 'password' fields are populated with properties from the context:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:am="http://www.myservice.com/AM" xmlns:sam="http://www.myservice.com/SAM">
<soapenv:Header/>
<soapenv:Body>
<sam:login>
<username>${Properties - Username and Password#Username}</username>
<password>${Properties - Username and Password#Password}</password>
</sam:login>
</soapenv:Body>
</soapenv:Envelope>
```

Expansión de propiedades

- ## ► Otro ejemplo:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:sam="http://  
    <soapenv:Header/>  
    <soapenv:Body>  
        <sam:logout>  
            <sessionid>${Test Request - login#Response#/sam:loginResponse/sessionid}</sessionid>  
        </sam:logout>  
    </soapenv:Body>  
</soapenv:Envelope>
```

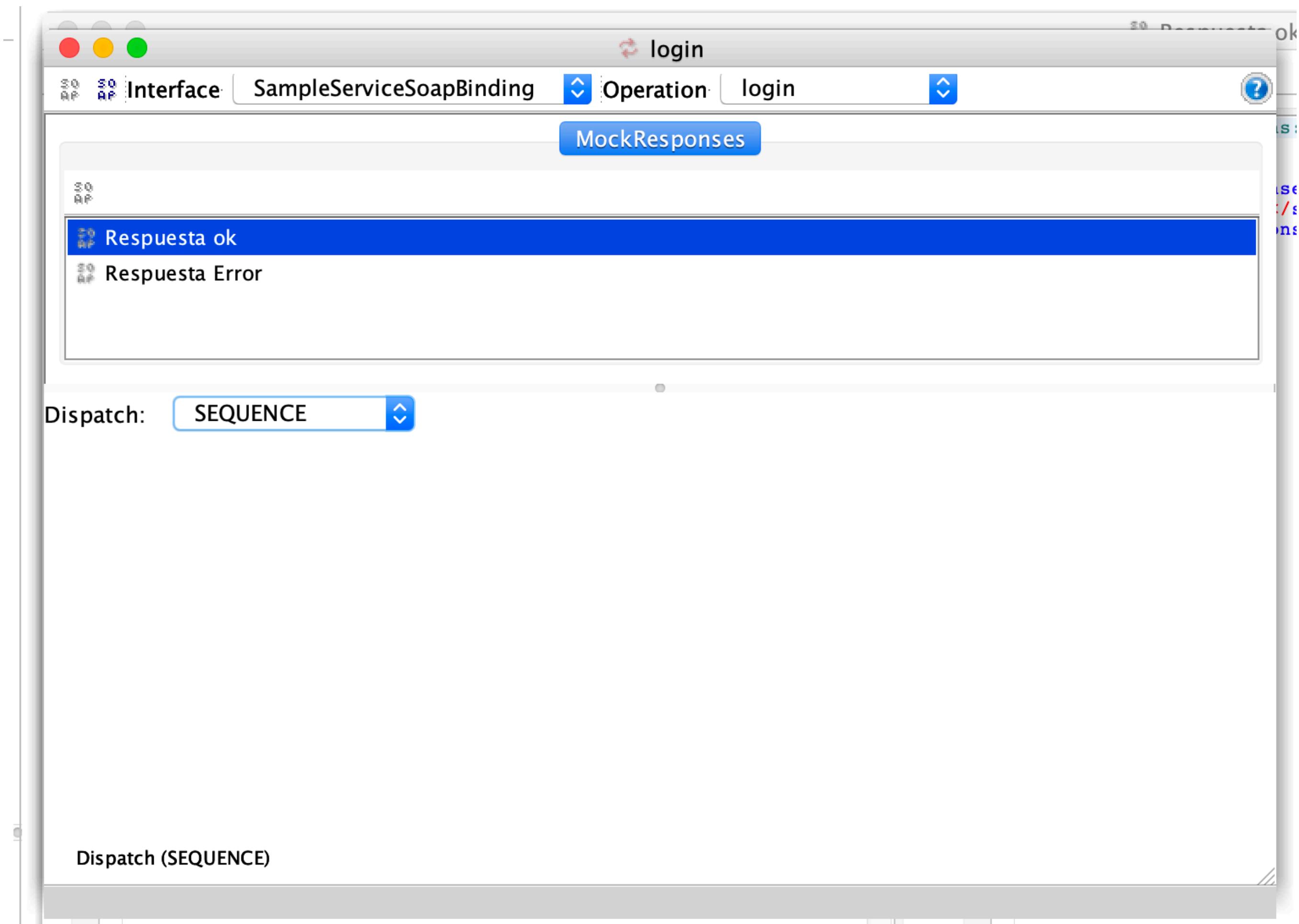
MockServices

- ▶ Introducción.
- ▶ Tipos de redireccionamientos.
- ▶ Trabajando con Scripts en Groovy.

Introducción

- Podemos definir servicios mock sobre cualquiera de las operaciones con las que vamos a trabajar.
- Se selecciona el servicio y con el botón derecho Add to Mockservice.
- Esto crea un servicio que será capaz de generar tantas respuestas como necesitemos para emular el comportamiento del servicio web real.

Introducción



Tipos de redirecccionamientos

- ▶ SEQUENCE: Este es el estilo de envío predeterminado. Con este estilo, al enviar solicitudes a MockOperation, MockResponses se seleccionan iterativamente una tras otra tal como aparecen en la lista MockResponses.
- ▶ SCRIPT: Este estilo ofrece la capacidad de controlar las respuestas en base a las secuencias de comandos Groovy. El MockResponse está determinado por la ejecución del script especificado.
- ▶ RANDOM: se selecciona la respuesta de forma aleatoria.
- ▶ QUERY_MATCH: Devuelve la respuesta evaluando expresiones XPATH.

Trabajando con Scripts en Groovy

- ▶ Tenemos acceso a un conjunto de variables implícitas:
 - ▶ log.
 - ▶ context: interface PropertyExpansionContext.
 - ▶ requestContext.
 - ▶ mockRequest.
 - ▶ mockOperations.
- ▶ Tenemos acceso a la API Java de SoapUI.

Trabajando con Scripts en Groovy

- ▶ <https://www.soapui.org/apidocs/index.html>

OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP
PREV NEXT FRAMES NO FRAMES ALL CLASSES

SoapUI 5.4.0 API

Packages	Package	Description
	com.eviware.soapui	SoapUI root package
	com.eviware.soapui.actions	Core soapui actions
	com.eviware.soapui.analytics	
	com.eviware.soapui.analytics.providers	
	com.eviware.soapui.autoupdate	

Trabajando con Scripts en Groovy

- ▶ Es muy habitual empezar nuestros scripts tal que así:

```
def groovyUtils =  
    new com.eviware.soapui.support.GroovyUtils(context)  
  
def holder =  
    groovyUtils.getXmlHolder(mockRequest.requestContent)
```

- ▶ Holder es de tipo XmlHolder y nos permite extraer información de la petición.

```
def username = holder.getNodeValue( "//username" )
```

Trabajando con Scripts en Groovy

- El valor de retorno es el nombre de la respuesta que va a generar:

```
if( context.hasProperty( sum ) )
    return "Already Logged In Fault"

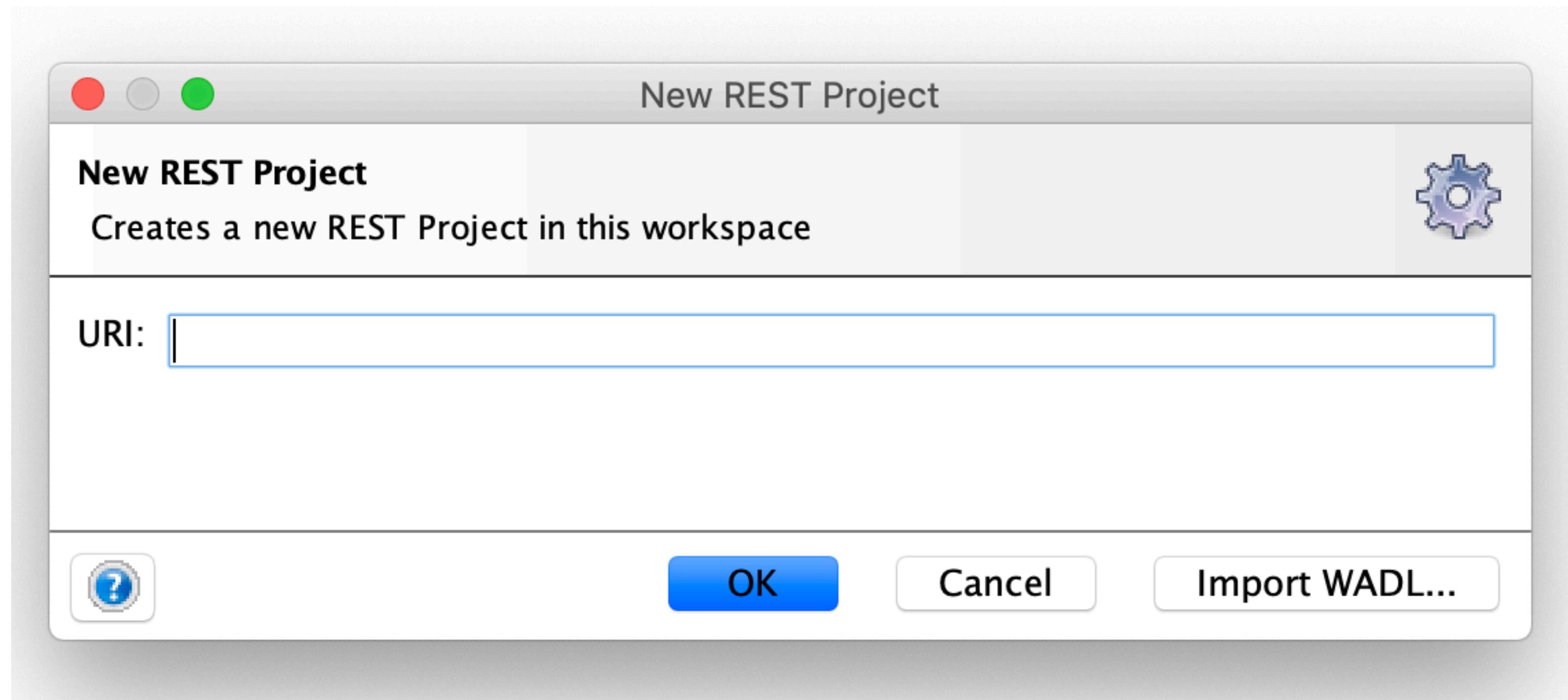
// seems ok
def sessionid = String.valueOf( Math.random() ).substring( 2 )

context.setProperty( sum, sessionid )
context.setProperty( sessionid, sum )

requestContext.sessionid = sessionid
return "Ok Response"
```

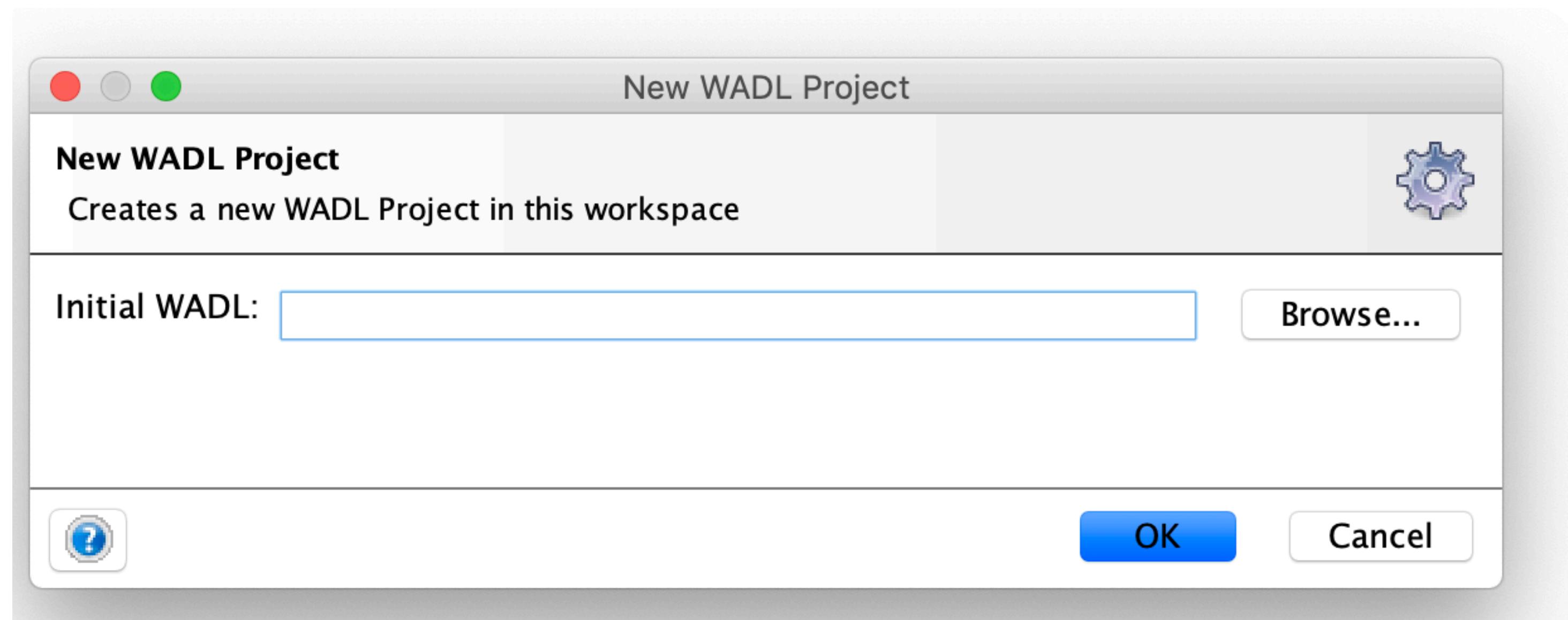
Proyecto REST

- ▶ Crear proyecto REST:



Proyecto REST

- ▶ Podemos crearlo a partir de un fichero WADL:



Proyecto REST

- Una vez creado el proyecto, se generan peticiones de prueba y, de la misma forma que trabajamos para proyectos SOAP, podemos definir peticiones, servicio mock y tests:

