Postman

Julian F. Latorre

Índice

1. Uso Avanzado de Postman

1.1. Collections en Postman

Las Collections en Postman son grupos organizados de solicitudes API que pueden guardarse y compartirse. Son fundamentales para la gestión eficiente de proyectos API.

1.1.1. Creación de una Collection

1. En Postman, haz clic en "New» Çollection".
2. Nombra tu collection, por ejemplo, ÇRUD API Users".
3. Opcionalmente, añade una descripción y elige un color para la collection.

1.1.2. Organización de Solicitudes

1. Dentro de la collection, crea carpetas para cada operación CRUD.
2. Arrastra y suelta las solicitudes existentes en las carpetas correspondientes.
3. Utiliza la función de "Duplicate"para crear variantes de solicitudes similares.

1.1.3. Ejecución de Collection

1. Haz clic en los tres puntos junto al nombre de la collection.
2. Selecciona Run collection".
3. Configura el orden de ejecución, el número de iteraciones y los retrasos.
4. Ejecuta la collection y analiza los resultados.

1.1.4. Compartir Collections

1. Haz clic en "Share.[en] la vista de la collection.
2. Genera un link para compartir o exporta la collection como un archivo JSON.
3. Los miembros del equipo pueden importar la collection compartida.

1.2. Environments en Postman

Los Environments son conjuntos de variables que permiten ejecutar colecciones en diferentes contextos (por ejemplo, desarrollo, pruebas, producción).

1.2.1. Creación de un Environment

1. Haz clic en el icono de engranaje y selecciona .ªdd".
2. Nombra el environment, por ejemplo, ÇRUD API Local".
3. Añade variables como:
   - `baseUrl`: `http://localhost:3000`
   - `apiVersion`: `v1`

1.2.2. Uso de Variables de Environment

1. En las solicitudes, usa la sintaxis {{variableName}}.
2. Por ejemplo: `GET {{baseUrl}}/{{apiVersion}}/users`
3. Cambia entre environments para probar en diferentes entornos.

1.2.3. Variables Globales vs de Environment

1. Las variables globales están disponibles en todos los environments.
2. Las variables de environment son específicas del environment activo.
3. Las variables de environment tienen prioridad sobre las globales.

1.2.4. Gestión de Claves

1. Usa variables de environment para almacenar tokens y claves API.
2. Marca estas variables como "secret"para mayor seguridad.
3. Evita compartir environments que contengan información sensible.

1.3. API Documentation en Postman

Postman permite generar documentación automática para tus APIs, facilitando su comprensión y uso.

1.3.1. Generación de Documentación

1. Selecciona la collection para la que deseas generar documentación.
2. Haz clic en "View documentation.ᵉⁿ el menú de la collection.
3. Postman generará automáticamente la documentación basada en tus solicitudes.

1.3.2. Personalización de la Documentación

1. Añade descripciones detalladas a cada solicitud y parámetro.
2. Utiliza la pestaña .ExamplesenEn cada solicitud para proporcionar ejemplos de respuestas.
3. Agrega Markdown en las descripciones para mejorar la legibilidad.

1.3.3. Publicación de la Documentación

1. Haz clic en "Publish.en la vista de documentación.
2. Configura la visibilidad (pública o privada).
3. Personaliza la URL y añade una contraseña si es necesario.
4. Comparte la URL publicada con tu equipo o usuarios de la API.

1.3.4. Mantenimiento de la Documentación

1. La documentación se actualiza automáticamente al modificar la collection.
2. Utiliza versiones de la collection para mantener documentación de diferentes versiones de la API.
3. Revisa y actualiza regularmente la documentación para mantenerla precisa.

1.4. Configuración de Environments

1. Crea tres environments: "Local", "Staging" "Production".
2. Configura las variables `baseUrl` para cada environment:
   - Local: `http://localhost:3000`
   - Staging: `https://api-staging.example.com`
   - Production: `https://api.example.com`
3. Añade una variable `apiKey` en cada environment (usa valores ficticios para este ejercicio).

1.5. Actualización de la Collection

1. Modifica todas las URLs en tu collection ÇRUD API Users"para usar {{baseUrl}}.
2. Añade un header de autorización a cada solicitud:

```
Authorization: Bearer {{apiKey}}
```

3. Crea ejemplos de respuestas para cada solicitud.

1.6. Documentación de la API

1. Añade descripciones detalladas a cada solicitud en tu collection.
2. Genera la documentación automática para tu collection.
3. Personaliza la documentación con Markdown y ejemplos.
4. Publica la documentación (configúrala como privada para este ejercicio).

1.7. Pruebas Avanzadas

1. Añade scripts de prueba a cada solicitud. Por ejemplo, para la solicitud GET:

```
pm.test("Status code is 200", function () {
    pm.response.to.have.status(200);
});

pm.test("Response is an array", function () {
    var jsonData = pm.response.json();
    pm.expect(Array.isArray(jsonData)).to.be.true;
});
```

2. Configura una secuencia de ejecución en tu collection que pruebe todo el flujo CRUD.
3. Ejecuta la collection en diferentes environments y analiza los resultados.