Ficha de ayuda: Introducción a la API

Arquitectura de la aplicación

La arquitectura de la aplicación es la forma en la que se organizan los componentes de la aplicación. En una aplicación web, la arquitectura puede incluir el front-end, el back-end, la API y una base de datos.

API (de Application Programming Interface, interfaz de programación de aplicaciones) es la interfaz que ayuda a las aplicaciones a interactuar entre sí. A través de la API, intercambian datos mediante solicitudes y respuestas.

Como parte del back-end, el área de desarrollo "escribe" la API y, por lo general, lo hace de acuerdo con las especificaciones.

Una base de datos (DB, database) es un conjunto de datos al que accede el usuario o la usuaria con la ayuda de una computadora.

Las bases de datos se pueden organizar de diferentes maneras. El tipo más común de base de datos es **relacional**, donde la información se almacena en tablas.

El back-end interactúa con la base de datos enviando solicitudes y recibiendo datos como respuesta.

Al probar la API, es importante comprobar que los cambios (adición, modificación o eliminación de datos) se han guardado correctamente en la base de datos.

Una vez conoces cómo funciona la arquitectura de la aplicación, puedes hacer lo siguiente:

- Generar más opciones de prueba porque entiendes cómo funcionan los componentes de la aplicación.
- Identificar un error en cualquier componente más rápidamente.

- Identificar un error con mayor precisión para ahorrarle tiempo al equipo de desarrollo a la hora de corregirlo.

La arquitectura de la aplicación se puede construir de diferentes maneras. Las formas más comunes son SOAP y REST.

El estilo de la arquitectura de la API determina cómo intercambian mensajes los componentes de la arquitectura: sobre qué protocolo y en qué formato. Esto influye en la elección de las herramientas para probar una aplicación y cómo las emplearás.

REST (Representational state transfer), transferencia de estado representacional, es una forma de diseñar una aplicación, su principal característica es su sencillez para la visualización de datos. En otros estilos de arquitectura, los datos "se envuelven" en una capa adicional de etiquetado. No hay una capa adicional en REST, por lo que queda claro de inmediato a partir de las solicitudes y respuestas lo que le ocurre a los datos.

Protocolo HTTP

El cliente envía una **solicitud HTTP** al servidor y el servidor devuelve una **respuesta HTTP**.

El **método** le indica al servidor qué acción debe realizar. Aquí están los métodos más comunes:

- GET: solicita datos.
- POST: crea o recibe datos.
- DELETE: elimina datos.
- PUT: actualiza datos.

La **URL** es la dirección que utiliza el cliente para solicitar los datos y que envía el servidor.

El **código de estado** te indica si el servidor procesó la solicitud correctamente. Por ejemplo, el código 200 significa que todo funcionó correctamente, mientras que el

código 500 indica un error interno del servidor.

El **texto de estado** es el texto que acompaña al código de estado. Por ejemplo, el error 500 es un "error interno del servidor".

JSON

JSON (JavaScript Object Notation) es un formato de datos "ligero" que se utiliza con frecuencia para intercambiar datos entre aplicaciones. La principal ventaja de JSON consiste en que es fácil de leer y escribir. Puedes describir casi cualquier objeto estructuralmente en JSON para usarlo en el sistema.

Para esto tienes los pares clave:valor. La "clave" es el nombre del parámetro del objeto, por ejemplo, apellido, segundo nombre o nombre. El "valor" se especifica después de los dos puntos.

```
{
  "first_name": "John",
  "last_name": "Johnson",
  "age": 35,
  "is_married": true,
  "has_kids": false,
  "emails": ["johnson.j@tripleten.com", "john@tripleten-students.com"],
  "favorite_numbers": [0, 7, 42],
  "pets": [{"type": "perro", "name": "Buddy"}, {"type": "gato", "name": "Watson"}]
}
```

Los parámetros se enlistan separados por comas y se describen en un formato determinado. Por ejemplo, la clave siempre va entre comillas dobles, pero su uso para el valor depende del tipo de datos. Los datos de texto se encierran entre comillas. Los números no van entre comillas. Nota: un conjunto de pares se encierra entre llaves, lo que te permite indicar que se refieren al mismo objeto.

En "Estado civil" opera la lógica binaria condicional:

- `true`: significa que es verdadero
- `false`: significa que no es verdadero

Escribir un valor entre corchetes indica una lista o un array. Puede contener un valor, `["type": "perro"]`, varios valores, `["type": "perro", "type": "gato"]`, o ninguno. Por ejemplo, si una persona no tiene direcciones de correo, esos datos aún se transmiten. Se verá así: `"emails": []`.

Dentro de un array, los datos simplemente se separan por comas en una lista y el orden es importante. En cambio, dentro de los objetos donde los parámetros están definidos por pares `clave:valor`, el orden es irrelevante.

Un objeto puede estar vacío, al igual que un array. En este caso, el valor se escribirá simplemente entre llaves sin contenido entre ellas: {}. Si quieres especificar explícitamente que no se asigna ningún valor a un parámetro, utiliza la notación `"pets": null`: la palabra `null` indica que el valor de este parámetro no está definido.