Sequelize

|  |  |
| --- | --- |
| sequelize init | Posicionado en la raíz del proyecto. Genera la estructura de carpetas.  Raíz  |\_\_models  |\_\_controllers  |\_\_seeders  |\_\_migrations |
| sequelize model:generate –name NombreModelo –attributes attr:type | Generación de un modelo, notar que luego de correr el comando debería generar 2 archivos, uno en la carpeta modelos y otro en la carpeta migraciones. |
| sequelize db:migrate | Corre las migraciones a la base de datos |
| Sequelieze db:undo | p/volver atrás los cambios |
|  |  |

// REST – Apunte sobre verbos http ¿Cuándo usarlos?

Estas son algunas reglas que te servirán para saber cómo y cuándo debes usar los verbos Http en una arquitectura REST.

Los verbos Http involucrados en un sistema REST son GET, POST, PUT, PATCH y DELETE.

GET es el más simple de todos, es el que usamos para obtener un recurso. Las peticiones GET no deben causar efectos secundarios en un servidor, no deben producir nuevos registros, ni modificar los ya existentes. A esta cualidad la llamamos idempotencia, cuando una acción ejecutada un número indefinido de veces, produce siempre el mismo resultado.

Esto quiere decir, que no importa cuántas veces hagamos una petición GET, los resultados obtenidos serán los mismos.

Cuando ingresamos a la dirección usando GET <https://codigofacilito.com/cursos/backend-profesional/> estamos solicitando que se nos entregue el recurso identificado por /cursos/backend-profesional, este es un buen ejemplo de uso con GET.

Esta otra ruta: <https://codigofacilito.com/cursos/recomendar?selected_level=0&category_options=28> aunque más compleja, también es correcta, estamos solicitando los recursos identificados por /cursos con las opciones ahí indicadas. Sin importar cuantas veces hagamos esta solicitud, no modificará los resultados por sí misma.

Las peticiones con POST son para crear recursos nuevos, no para eliminarlos, ni para modificarlos. Cada llamada con POST debería producir un nuevo recurso.

Lo que es interesante acerca de POST no es el verbo en sí, queda muy claro que es para crear, más bien es los recursos a los que se dirige.

Por ejemplo, si en nuestra aplicación existe una colección de cursos, la solicitud para crear uno nuevo, debe ser con el verbo POST al recurso que identifica la colección, por ejemplo /cursos.

Si queremos crear un nuevo artículo, pudiéramos tener una URI /articulos. Lo que es importante en estos casos, es recordar que la URI no debe decir qué acción estamos ejecutando, nos olvidamos de /articulos/crear o de /cursos/agregar, etc. El verbo dice qué haremos, y la URI sobre qué recurso se harán las modificaciones.

Algunos escenarios más complejos para el uso de POST son los inicios de sesión, agregar a un carrito de compras, procesar un pago nuevo, etc.

Consideremos por ejemplo el inicio de sesión, normalmente al iniciar sesión, no producimos un nuevo registro en la base de datos, sin embargo, usamos POST porque estamos creando una sesión nueva. Esto nos da a entender que para saber si usaremos POST o no, no necesariamente tenemos que agregar registros en la base de datos, el recurso creado puede ser de otros tipos, como una sesión.

Los verbos PUT/PATCH van como el señor cara de papa y la señora cara de papa, siempre juntos. Los agrupamos porque ambos indican una modificación en el servidor.

En la teoría, PUT se diferencía de PATCH, en que el primero indica que vamos a sustituir por completo un recurso, mientras que PATCH habla de actualizar algunos elementos del recurso mismo, sin sustituirlo por completo.

Un escenario común para el uso de PUT sería una llamada para actualizar la información de un curso, por ejemplo:

PUT /cursos/backend-profesional

O también:

PATCH /cursos/backend-profesional

En la práctica, no conozco un framework que establezca una diferencia en funcionamiento para peticiones con PUT o con PATCH, ambos verbos son tratados como iguales.

Por último, DELETE es el verbo que usamos para eliminar registros, bien pudiera ser para eliminar un recurso con un mensaje Http como

DELETE /cursos/backend-profesional

O para eliminar una colección completa:

DELETE /cursos

Esta es la manera a través de la que usamos los verbos Http en una aplicación web. Estos en combinación con las URIs proveen la interfaz uniforme de la que hablamos cuando discutimos las características de un sistema REST.

Como podrás notar, el beneficio de estas es que se establece una guía para la construcción de la aplicación, las rutas siempre representan recursos, las acciones se representan con Http.