¿qué vamos ver en esta clase?

crear, actualizar y borrar registros de una base de datos con sequelize usando el método **update()** y para borrarlo el método **destroy(),** y **créate()** para crear un registro.

Todo esto, como ya sabemos, recibe el nombre de **CRUD**, por sus siglas en inglés: **Create, Read, Update, Delete**.

* Implementar las tareas de un sistema CRUD con Sequelize.
* Entender que los métodos destroy() y update() necesitan previamente de un where.
* Capturar la información que viaja en el request para que la misma sea guardada en la base de datos.
* Implementar mecanismos de control en los escenarios en donde el registro no pudo ser creado o actualizado.

1. create() y bulkCreate()
2. update() y upsert()
3. Destroy/Soft deletes
4. Notas de clase

1.

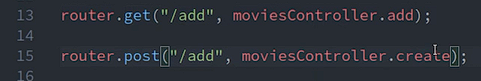
**create()**: Agrega nuevos registros a la tabla. **RECIVE UN SOLO PARAMETRO, UN OBJ LITERAL CON LOQ UE QUEREMOS AGREGAR**

Este método recibe un **obj literal** con todo lo que queramos agregar.

Cada atributo representa un nombre de una tabla: 

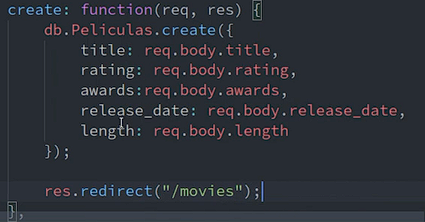
Ejemplo práctico:

Ya tenemos el modelo de la tabla de Pelicula (sequelize ya sabe como es la tabla, las columnas y los tipos de datos de cada una), en routs ya está la ruta por get a “/add” (con el método add que agrega) y la ruta por post con el método create (el que vamos a completar ahora) que crecive la info cuando el formulario ya este completo.



Completemos dentro del controlador el método create, que recivira la info y la almacenara el la DB.

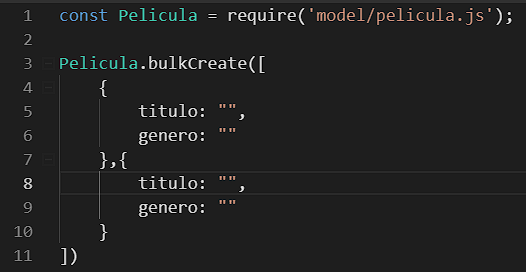
Lamamos a la DB, a la tabla de peliuclas, método create(), y dentro un objeto literal con todas la columnas. A cada columna le damos el valor que viene por el req.body y el nombre del label



\*no pone la columna id porque es autoincrementar y sequelize entiende eso y crea la columna automáticamente.

**bulkCreate()**:

Este método es igual a create, con la diferencia de que nos permite agregar varios registros a la DB. A este método le pasamos un array de objetos con el mismo formato y los agrega. Es como hacer varios create separados.



2.

Update(): actualiza un registro de la DB. **RECIVE DOS PARAMETROS: LO QUE VAMOS A ACTUALIZAR Y EL REGISTRO QUE QUEREMOS ACTUALIZAR**

Necesito **la info** que va a actualizar y **un filtro** a través del cual se actualizara la info de un determinado **registro existente**. Este filtro es un **where** para indicar que registro es el que debe actualizar



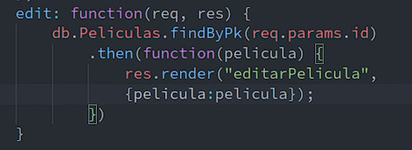
**IMPORTANTICIMO EL WHERE!, SI NO PUEDE SER QUE SE ACTUALICEN TODOS LOS REGISTROS DE LA DB.**

Ejemplo practico:

Tenemos una ruta por get, porque primero queremos que el usuario lea lo que va a modificar, que responde a /edit/:id, que utilice el método edit:

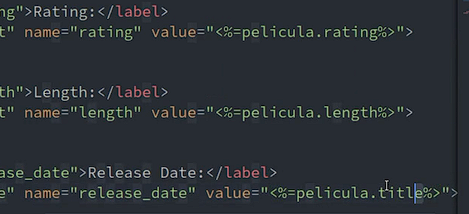


En el controles creamos el método edit, que busca por el id la película y la comparte en la vista



La vista será un formulario similar al de creación de películas, con la diferencia de que el action nos lleva /movies/edit, y que todos los campos deben aparecer completados con los datos registrados por lo que todos tengras la etiqueta value=”” con los datos registrados





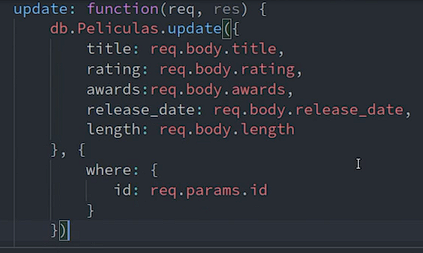
Ahora hacemos la ruta por post a /edit/:id, con el método update



En la vista cambiamos el action para que en la URL mantenga el id de lo que estemos editando



Ahora creamos el método update:



Y hacemos un res redirigiendo al usuario a donde queremos:



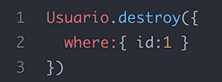
**upsert():** dos parámetros: los valores y las opciones.

3.

Destroy/Soft deletes: **RECIVE UN SOLO PARAMETRO, UN OBJ LITERAL CON UN WHERE**

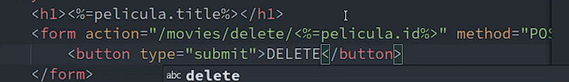
Elimina registros de las tablas de la DB. NO TE OLVIDES DE PONER EL WHERE EN EL DELEET DEL FROM.

Si queremos eliminar el usuario con id 1 sería así



Ejemplo práctico:

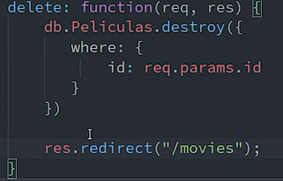
Creamos un formulario, el action responde a /delete/”el id de la película en cuestión”. Y el method es POST. El formulario es solo un botonm que dice delete.



La rtura por post a /delete/:id con el método delete.



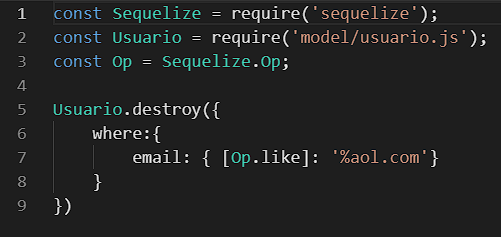
El método delete sería asi: redirige al listado le películas.



Hay que tener la cuenta si es que la tabla tiene relaciones. En esos casos hay que borrar primero las relaciones y luego la tabla en cuestión.

Ejemplo de playground:

Elimina los usuarios que su email termine con ”aol.com”



Sequelize en realidad no elimina los registros que le damos, si no que lo guarda en una columan llamada deletedAt. Este concepto es llamado **paranoid.**

**Documentación de sequelize paranoid:**

[**https://sequelize.org/docs/v6/core-concepts/paranoid/**](https://sequelize.org/docs/v6/core-concepts/paranoid/)

4.

Notas de clase:

Mapeador relacional de objetos: moongos si trabajáramos con mongoDB (competencia de Sequelize, MySQL)