Vamos a repasar un poco los conceptos con respecto a la carga de archivos. Necesitamos una dependencia. Hay muchas, algunos utilizan "multer", en esta clase estamos utilizando "express file upload" funcionan de manera similar, no hay grandes diferencias, ustedes pueden elegir la que mejor les guste, la que mejor les acomode. Pero, el concepto es el siguiente: tomamos un archivo que viene en un objeto que se llama "req punto files" del formulario, recuerden los datos del formulario, esto lo vimos cuando nos lo grabamos en la primera parte, cuando hacíamos el "login", recogíamos los datos del formulario en un objeto que se llama "req punto body" y en el cuerpo de la request o petición. Para trabajar con imágenes, tenemos que hacer un pequeño cambio en nuestro formulario, cambiar el tipo de en coaching o codificación y luego, tenemos los campos. Ponganos normales, el objeto body, de la rico es sonia, en el cuerpo de la petición, pero los archivos serán del tipo que serán, van a venir en otro objeto, van a venir en "req punto file", o sea, en los archivos de la petición. Vamos a recoger eso y, no lo vamos a subir directamente a la base de datos, porque nuestra base de datos sería monumental, seguría, se haría muy grande. Haremos otra cosa, vamos a utilizar un servicio que se llama "Cloudinary" y subiremos la imagen que viene en el formulario a "Cloudinary" y le diremos a Cloudinary en ese paso, "cuando termines de subirla, dame su "public ID", su identificación pública", que es la manera de acceder a la imagen. Paso número 1. Una vez que tengamos ese dato, vamos a cargar el resto de la información en nuestra base de datos SQL y en el campo imagen, cargaremos el public ID, la identidad pública de la imagen que nos devolvió Cloudinary. Entonces, tendremos nuestra información repartida en dos servicios. Las imágenes en un servidor de imágenes y el resto de los datos, junto con el identificador que sirve para acceder a la imagen, en nuestra base de datos SQL. Este es un poco el círculo que queremos cerrar con este, con este concepto, en estas clases. Vamos a completar entonces.

Vamos a continuar ahora entonces con la lógica para agregar un nuevo registro, para hacer una de las operaciones básicas del "crud", que es "crud la create, read, update and delete". Vamos a crear un nuevo registro en la base de datos. ¿Recuerdan que en los modelos teníamos las operaciones sobre bases de datos? Como ahora vamos a hacer operaciones en una tabla distinta, en la tabla de razas, bien podemos crear podríamos hacerlo acá, pero es mucho más prolijo crear un nuevo modelo, un modelo para razas y que se llame de esta manera. Obviamente, vamos a requerir el pool de conexiones, como siempre. Está el pool de conexiones en nuestro archivo "DB", o "data base". Y vamos a hacer una función aquí, vamos a crear una función, vamos a hacerlo con la "arrow function agregar raza" esto va a ser una función de tipo asincrónico, ya a estas alturas tenemos que tener incorporada esta idea, cuando consultamos, cuando trabajamos contra una base de datos, es decir, contra un recurso fuera de nuestra computadora, siempre hay una demora, aunque sea mínima, por lo tanto, estamos hablando de código asincrónico. Debemos indicarle a Javascript cómo manejarlo exactamente. Y acá vamos a, bueno, vamos a hacer un bloque "try catch" para manejar un error, en el caso, en caso de que el error exista, que no se rompa nuestro, la comida, la ejecución de nuestro, de nuestro programa, sino, que podamos mostrar ese error, hacer algo con él. Por ahora, simplemente los mostramos, para tener una idea de que están allí, ¿okey? Y, ¿qué es lo que vamos a hacer dentro? ¿Qué causaremos dentro del bloque "try"? Bueno construyemos, una query, una consulta, lo que queremos preguntarle, lo que queremos lanzar a la base de datos y ¿cómo insertamos un nuevo registro? ¿recuerdan? Bueno, es con "insert into".