**Bases de datos desde el Backend**

\*“La necesidad de soluciones bien pensadas para la captura, el almacenamiento y la recuperación de datos solo tiene el propósito si existe una aceptación del valor intrínseco de los datos, más allá del uso ad hoc original...” Amalio Telenti — Jefe de Datos y Biología Computacional en Vir Biotecnología Inc.\*



Fuente: [Freepik](http://www.freepik.es/)

Manejar un nuevo lenguaje de programación lleva mucho tiempo y dedicación para poder dominarlo a la perfección. Recién estás comenzando a vislumbrar un nuevo universo: el de las bases de datos relacionales y SQL. Interesante, ¿verdad? En tan solo unas pocas semanas de este segundo sprint, ya sabes qué son las bases de datos relacionales, no relacionales y comenzaste a trabajar con el nuevo lenguaje SQL, ¡increíble! Ahora, es necesario dar unos pasos para atrás para ver cómo podemos diseñar un modelo para una base de datos relacional. A la hora de diseñar un modelo, es importante detenernos en ciertas cuestiones esenciales que tenemos que considerar antes de crearlas. Para ello los/as developers tenemos algunas buenas prácticas que deberíamos implementar para poder manejarnos con seguridad y confianza a la hora de trabajar con BDRs. Te las compartimos a continuación:

1. **Modelar las tablas antes de crear:**puedes empezar con un boceto en papel y lápiz o utilizar alguna herramienta como guía:
2. [Draw.io](https://app.diagrams.net/): online (open source)
3. [Lucidchart](https://www.lucidchart.com/): online (licencia limitada)
4. [Visio](https://www.microsoft.com/en/microsoft-365/visio/flowchart-software) parte del paquete office (licencia Microsoft)

2. Usar nombres acordes a las entidades: que sea coherente a lo que estamos desarrollando, recuerda que trabajamos en conjunto con otros/as developers.

3. Realizar la documentación: a partir de ella crear las entidades/tablas y manos a la obra con la creación.

4. Guardar queries importantes o de uso diario: **para evitar su reescritura.** Muchas veces, a la hora de hacer pruebas, vamos a necesitar tener nuestros queries para hacer consultas.

Estas son algunas de las buenas prácticas que puedes utilizar. Es importante que incorpores estos criterios y los tengas presentes a medida que avances en este aprendizaje, ya que sentarán las bases de tus habilidades como developer en el desarrollo de base de datos relacionales.

**Conectando datos desde NODE**

Ahora sí, continuemos entonces con las bases de datos relacionales. Según las palabras de [Cristian Moreno](https://medium.com/@khriztianmoreno), Full Stack Javascript Developer y Community builder, Co-Organizador de [MedellinJS](https://twitter.com/MedellinJS) y [Avanet](https://twitter.com/avanet):

\*“En el día a día de mi trabajo como Full Stack Developer es necesario trabajar con el acceso y persistencia de los datos, por lo general siempre que se usa una bases de datos NoSQL tenemos a MongoDB y Mongoose Js como la dupla perfecta, pero cuando debemos trabajar con bases de datos relacionales el equipo élite siempre será PostgreSQL y Sequelize.” \*

A continuación, nos enfocaremos en el aprendizaje de Sequelize, un [ORM (Object-Relational-Mapper).](https://es.wikipedia.org/wiki/Mapeo_objeto-relacional) ¡No te preocupes! Veremos este concepto en mayor detalle más adelante en el Sprint, por ahora solo utilizaremos Sequelize para ejecutar nuestras consultas SQL en la base de datos de NodeJS. Este es un paquete que debemos instalar con NPM y su función es comunicarnos con diferentes bases de datos relacionales, entre las cuales se encuentra María DB.



[Fuente](https://ed.team/blog/agiliza-tu-desarrollo-en-nodejs-con-el-orm-sequelize)

Por ejemplo, utilizar nuestros datos desde una aplicación web no es la mejor opción. Al ser una conexión directa, corremos el riesgo de exponer las credenciales de conexión a la DB y darle acceso a personas usuarias que no deberían. En este caso lo que hacemos los/as developers es crear **una API que se encarga de hacer consultas específicas de acuerdo a lo que la aplicación le solicite**para devolver la información finalmente a la aplicación.

Sequelize se destaca porque a través del método query**siempre devuelve una promesa** que debe ser procesada correctamente, y puede trabajar con las siguientes bases de datos: María DB, MYSQL, SQLite, POSTGRES y MSSQL. Contar con este mecanismo nos ahorra tiempo y esfuerzo durante el desarrollo, ya que con una sola librería puedes utilizar los mismos métodos para conectarte a diferentes bases de datos.

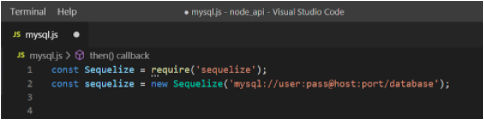
Lo más importante es que su aprendizaje será sencillo: ya cuentas con conocimientos sobre bases relacionales y JavaScript. ¡Despreocúpate y anímate a dar los primeros pasos!

**Primeros pasos con Sequelize**

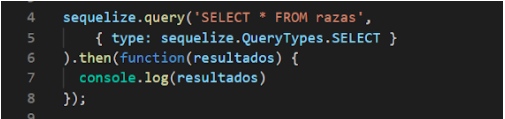
Como nosotros/as queremos interactuar con una base de datos SQL, tenemos que usar un drive específico: MySQL2.

PASO 1: Crear la conexión. Lo primero que hacemos es crear el archivo de conexión y le pasamos un string con la siguiente información de la conexión:

* **user:**Usuario de conexión
* **password:**Contraseña asignada
* **host**: Dominio o IP donde corre MySQL
* **port:**Puerto donde escucha MySQL
* **database:**Nombre de base de datos



**PASO 2:** Ejecutar la sentencia SQL. El método query retorna una promesa; entonces la capturamos con .then



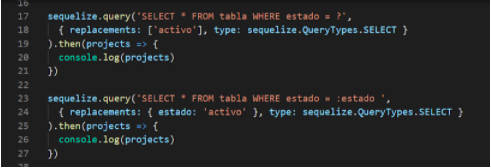
Su respuesta es un array con cada uno de los registros que provienen de la base de datos.

**Reemplazos**

Ya establecida la conexión con la base de datos, puedes empezar a realizar operaciones sobre ella. Utilizaremos los comandos mencionados en la toolbox anterior para manipular datos: Select, Insert, Update, Delete y Where.

Con Select vamos a realizar los reemplazos necesarios. Estos pueden realizarse de dos maneras: a través del signo ? o :campo

* **El signo ?** será reemplazado por el valor del array.
* **:campo** buscará la key que contenga el mismo nombre.



Por ejemplo, si necesitas extraer cierta porción de los registros de alguna tabla necesitarás añadir un WHERE a tu consulta. Imagina que tienes una tabla de eventos. Tu aplicación tendrá un Endpoint con express para administrar esta entidad. Seguramente tendrás un Endpoint que reciba un ID para devolver un evento en específico.

Es fundamental que valides que el ID que recibes sea un número para prevenir algún ataque de  [SQL Injection](https://www.w3schools.com/sql/sql_injection.asp) . Para facilitar esta operación, Sequelize te permite añadir un valor externo a la consulta securizando el valor y evitando que inyecten código a través de ese valor.

**Cierre**

En esta toolbox avanzaste en el trabajo de las bases de datos relacionales, y conocimos cuáles son las buenas prácticas que tienes que incorporar como developer para diseñarlas. También comenzaste a conectarlas entre sí a partir del paquete de Node: Sequelize. ¡Continúa ejercitando las conexiones en las bases de datos! Como dice el famoso refrán: “La práctica hace al maestro”.