Bueno comenzaremos a trabajar con datos, digamos reale,s o por lo menos almacenados en una instancia real de acceso a la información una base de datos, pero claro, para trabajar con una base de datos, ustedes ya saben cómo crearlas, cómo manipularlas, pero cómo hacerlo desde dentro de nuestro sistema, eso es lo que vamos a aprender.
Vamos a ver cómo podemos este crear una base de datos y ponerla en conexión para que trabaje con nuestro proyecto, esto es lo interesante. Vamos a ver, además, la dependencia "MySQL" igual que nos permite precisamente trabajar con una base de datos de tipo SQL conectada con nuestra aplicación, vamos a crear la conexión a la base de datos. Lo haremos antes de que la base de datos realmente exista y esto nos va a servir para probar el error de conexión, por lo tanto, bueno, vamos a aprovechar esa circunstancia, induciremos a propósito un error, sólo para ver si hacemos en la lógica a un manejo de errores. Correcto, comenzaremos definiendo un archivo que, bueno, puede llamarse de varias maneras, podríamos hacer una carpeta independiente también. Vamos a ponerlo aquí, por esta vez va a estar bien. Vamos a poner menos "DB" por "data base" o base de datos y aquí vamos a escribir la configuración necesaria para conectarnos a la base de datos.
Para hacerlo, ya que usamos bases de datos con el administrador MySQL, vamos a requerir un paquete que se llama precisamente MySQL, lo vamos a instalar "npm" y "MySQL" o "MySQL".
Una vez instalado este paquete, revisamos el package "punto JSON", lo vemos aquí, correcto está todo bien, volvemos a nuestro archivo de configuración, de conexión, y vamos a crear una constante, que se llame "MySQL" y vamos a requerir ese paquete, "require MySQL", muy bien.
Ahora, también, vamos a definir aquí abajo los datos de la conexión. Crearemos un pool de conexiones, podemos crear una conexión individual, pero es preferible crear un pool de conexiones, porque esto va a permitir que cada consulta no bloquee a la siguiente, es decir, si tenemos una sola conexión y ingresan 10 peticiones de usuarios y cada una demora dos segundos, bueno, vamos a demorar 20 segundos en hacer todo el proceso. Se hará una cuando ésta termine, luego la otra y así sucesivamente. En cambio, con un pool de conexiones, lo que vamos a lograr es que se puedan hacer una determinada cantidad. Nosotros vamos a poner un límite de 10. Aquí puede ser un número mayor o menor y decíamos entonces que, en base a ese número, se pueden hacer tantas consultas simultáneas como sean necesarias. El pool de conexiones a través de "MySQL", directamente cuando detecta una nueva consulta, abre un nuevo puerto de conexión, bien, hasta el límite que nosotros le indiquemos. Esto permite hacer varias consultas en paralelo, okey, entonces aquí utilizaremos MySQL, punto, podríamos, como les dije recién, crear una conexión, pero crearemos un pool de conexiones, "ripoll" y aquí tenemos que pasar un objeto de configuración con, por ejemplo, el límite de conexiones. Podríamos indicar "conection limit" y aquí dos puntos. Vamos a poner un límite de 10, bien, okey.
Ahora tendremos que indicar el "host", es decir, dónde estará albergada nuestra base de datos. Recuerden como estamos en esta etapa de prueba, montamos un servidor local con Xampp o con Mampp, con el servicio que ustedes hayan instalado, por lo tanto, nuestra base de datos es servida en "localhost", recuerdan que íbamos al "localhost barra PHP MyAdmin", etcétera. Bien, ahí es donde está nuestra base de datos, en localhost. Vamos a darle un nombre a la base de datos, bueno, se llamará, a tono con nuestro proyecto, "kittens". Recuerden, esta base de datos aún no existe, el usuario, no vamos a crear un usuario ni una contraseña, por defecto el valor es "root", o raíz y si ustedes no incluyen una contraseña, cuando crean la base de datos, bueno, por defecto, viene sin contraseña, por lo tanto, lo

vamos a dejar así, será más rápido para probarlo, ¿okey? Y ahora vamos a intentar conectarnos a esta base de datos, pero me van a decir con justa lógica, “si aún no has creado la base de datos, esto se va a romper”. Exactamente. Lo que quiero probar es si la lógica de manejo de errores que vamos a construir, está bien implementada. Si está bien implementada, nuestro proceso va a seguir corriendo, no se va a romper. Vamos a hacer “pool, punto”, tenemos el método “get connection”, es decir, conectarnos con la base de datos a través del pool, esto recibe una función que puede tomar hasta dos parámetros, uno es el error y el otro es la conexión. Estos son nombres convencionales ¿verdad?, pero se recomienda usarlos, porque la documentación así lo indica. Y aquí abrimos nuestra función, recuerden, podemos usar funciones “flecha” para que esto sea un poco más conciso. Bien, y aquí dentro podríamos preguntar si hubo un error y si lo hubo, bueno, podríamos, podríamos hacer algo, ¿okey? En caso de que haya un error, podríamos indicar un mensaje, por ejemplo, “console, punto, log”. Bueno, usemos el objeto “warn” que nos permite enviar una advertencia un “warning” con dos campos, primero un mensaje, por ejemplo, “no se pudo establecer la conexión” y luego de la coma, le pasamos un objeto, en ese objeto podríamos enviar, por ejemplo, cuál es el error, es la clave del objeto y el valor, la propiedad, es precisamente el error que haya habido, si es que lo hubo cuando se ejecutó el método “get connection” que lo tenemos aquí, por lo tanto, aquí podríamos poner “err” y vamos a obtener del error que tienen muchas propiedades, mucha información, solamente el mensaje. Y aquí lo tenemos, “error punto message”, o “err punto message”, en este caso. Bueno, por el contrario, si no hubo un error, lo que hacemos es, podríamos indicar que la conexión se ha establecido exitosamente, algo así. “Conexión con la base de datos establecida”. Okey, muy bien, esto lo tenemos aquí abierto, cerrado, ¿qué nos falta? Bueno acá tengo que cerrar mi “get conexión” me falta, creo que una llave y una paréntesis, si no me equivoco. Muy bien, ahí lo tenemos. Entonces, ahora tendríamos que exportar nuestra, nuestro pool de conexiones, “module punto exports igual pool”. Antes de esto, vamos a convertir nuestra, nuestro método para hacer las queries, las consultas a la base de datos, al método asincrónico de promesas, es decir, esto nos va a permitir usar “async await”, las palabras reservadas “async await”, que nos permiten escribir un código un poco más legible. Cuando llegue el momento, esto es simplemente, repitan esto, aunque ahora, en este momento, en este preciso instante, no tenga ningún efecto y aquí vamos a indicar “util” punto, bueno, tengo que requerir este módulo “util”, lo vamos a requerir aquí, es el que me va a permitir, precisamente, convertir las consultas a la base de datos al modelo de “async await”, para poder escribir el código de esa manera. Esto simplemente es una utilidad ¿verdad? No hace algo de fondo con nuestro código. Vamos a requerir el módulo “util” hay, antes este módulo había que importarlo, pero ahora, si no me equivoco y creo que no me equivoco, ya es un módulo nuclear, es decir, si está en el core, en el núcleo de express, está incorporado, por lo tanto, no tenemos que importarlo en nuestro package punto JSON.

Bien, muy bien y aquí abajo entonces vamos a utilizarlo, vamos a decir “promisify pool query”, es decir, que cuando hagamos más adelante, en la próxima clase, una consulta a través de nuestro pool a la base de datos, podremos utilizar las palabras reservadas “async await”, es una forma más rápida de escribir código en promesas. Ya vimos promesas con ustedes en las clases de Javascript, por lo tanto, bueno, supongo que deben tener un recuerdo o alguna noción de esto.

Bien, vamos a probar qué ocurre ahora, si es que esto se conecta. Para que se conecte tenemos que venir a nuestro archivo del lanzamiento de la aplicación, que es a JS, en nuestro caso, y aquí, bueno, importar nuestra base de datos, es decir, requerir la conexión a

nuestra base de datos para que éstos ejecuten, que si no, nunca se va a ejecutar. Recuerden, este es el archivo js, en el cual comienza a existir, comienza a vivir toda nuestra aplicación, por lo tanto aquí, podríamos requerir el pool de conexiones, porque es lo que hemos exportado ¿verdad? Aquí nosotros exportamos el módulo con este nombre, podríamos haber elegido otro nombre de conexión, pero este nombre es descriptivo, porque no creamos una conexión única, sino, como les indique más temprano, un pool de conexiones, un grupo de conexiones, por eso usamos este nombre, es más descriptivo para alguien que venga y tome nuestro código por alguna razón, tenga que trabajar con nuestro código, algún compañero o compañera de trabajo, pues, no tenga una aproximación a qué es lo que viene en este objeto y dónde se encuentra esto, en un archivo y el archivo que está en la raíz y se llama debe, no hace falta poner js, porque no es necesario.
Muy bien, vamos a correr nuestro pool de conexiones. Esto, por supuesto, debería darnos un error, bueno y aquí tenemos el error, aparentemente tenemos un puerto en uso, ya es otro tipo de error, no lo había previsto, creo que tiene que ver con eso. Estábamos corriendo un proceso paralelo, vamos a cerrar aquí, no es un error de la base de datos, okey, es otra cuestión, vamos a limpiar y vamos a volver a correrlo y ahí tenemos perfectamente funcionando nuestro sistema. Me van a decir "¿Cómo perfectamente, si no se ha conectado?, bien, pero fíjense que aquí no tenemos un error, no aparece algo en rojo, no nos saca de juego. De hecho, si venimos a nuestra página de gatitos y aquí a nuestra página de gatitos, vemos que no tenemos un mensaje de error, en este caso y ahí se ha cargado a nuestra página, está corriendo perfectamente, pero no estamos conectados a la base de datos, claro, lo hicimos adrede para probar esta, esta secuencia de manejo de errores. Ahora tenemos que crear la base de datos en realidad no se conecta, porque la base de datos ni siquiera existe. Bueno, fíjense que se tiene que llamar "kittens", porque así hemos configurado la conexión ¿dónde hacemos esto?, recuerdan, vamos a terminar este proceso tenemos que abrir Xampp, nuestro servidor Apache, para poder trabajar en el navegador con una base de datos MySQL.
Ya está corriendo. Vamos al navegador, bueno, y aquí junto a los gatitos, pongamos "localhost", ¿recuerdan?, era para PHP MyAdmin y esto nos va a abrir el administrador de la base de datos. Vamos a crear una nueva base de datos, va a llamarse aquí "kittens". La creamos, muy bien, ahí está nuestra base de datos creada. Vamos a ir aquí, hagamos todo por línea de comandos, creemos una tabla "create table", la tabla debe llamarse, bueno, vamos a ponerle "usuarios", es una tabla de usuarios. Y aquí vamos a definir sus campos. Va a tener nuestra tabla un campo e-mail, recuerdan, esos son los datos que pedíamos en el formulario de contacto, que va a ser de tipo "bar chart" de 64, está bien y vamos a poner que sea de tipo "unique", no queremos que haya e-mails repetidos y luego podremos indicar "password" y nuestra contraseña va a ser de tipo "varchart" y aquí podemos pedir una contraseña de no más de 8, 10, 12 caracteres, 16, lo que ustedes quieran, pero, como la vamos a encriptar porque no es bueno guardar las contraseñas en una base de datos sin encriptarlas, por una cuestión de seguridad, vamos a ese es ese procedimiento va a ser que una contraseña relativamente corta sea mucho más larga porque se le agregan caracteres aleatorios de acuerdo a un algoritmo, por lo tanto, aquí vamos a prever eso y vamos a hacer que, bueno, permita 255 caracteres, con esto va a estar bien. Perfecto y ahora, si damos "center", tenemos nuestra tabla. Vamos a verla aquí vamos a hacer "describe usuarios", bien, y aquí tenemos nuestra nuestra tabla. La tabla usuarios, ¿okey? Podríamos indicar inclusive, aquí podríamos modificar la tabla, para que no admita valores nulos ¿verdad? Podríamos indicar "alter table", "alter table usuarios", recuerden, aquí estoy mezclando mayúsculas y

minúsculas, porque, bueno, ya saben ustedes que es exactamente lo mismo, pero para ser más prolijos podríamos indicarlo así, ¿no? Y aquí podríamos indicar “modify column” o también, miren, sería mucho más rápido porque es muy breve nuestra tabla, eliminarla y cambiar aquí los dos campos o también, esto nunca se los mostré, pero bueno está bien que lo vean, ¿recuerdan que dijimos que podíamos hacer todas estas operaciones directamente aquí, en el entorno gráfico? Bueno podemos, podemos hacerlo eso tranquilamente. Aquí podríamos entrar a nuestra tabla de usuarios, ir a la estructura de la tabla y aquí bueno, podemos poner cambiar y en el campo, por el valor predeterminado, podemos poner que no sea, que no sea nulo, o que no admita nulos, aquí, nulo no, lo clickeamos, estaba aplicado, lo quitamos, no puede ser nulo. Bien y guardamos este campo. Bien, no puede ser nulo este campo. Ahora vamos a cambiar el campo contraseña y vamos a decir que no puede ser un campo nulo, perfecto.
Muy bien, ya está actualizada nuestra estructura. Vamos a agregar un sólo campo, un sólo registro para poder hacer luego nuestras pruebas. Les voy a mostrar lo siguiente, si aquí intentamos insertar un registro, ustedes van a ver que en el campo e-mail aquí iría el valor ¿verdad? Aquí a la derecha, a la izquierda se pueden correr funciones antes del valor y tenemos funciones para encriptar, podemos usar por ejemplo, el método “MD5”, es un método un algoritmo, uno de muchos algoritmos que nos permite encriptar una contraseña o un campo, en realidad cualquier cosa. Si aquí escribo, por ejemplo, Pepe, si esa fuera mi contraseña, si, no es muy segura, lo sé, en realidad, luego de guardarla, no vería Pepe aquí, vería una cadena generada por el proceso de encriptamiento. Vamos a ver cómo lo hacemos. Volvamos a nuestra terminal de consultas. Vamos a insertar un registro “insert into”, la tabla se llama, debo escribir bien “insert”, sino no hará nada, “into usuarios”, usuarios, y vamos a poner que el e-mail sea “a, arroba, be, punto ce”, sólo para que luego sea rápido para nosotros trabajar con este, con este login ¿verdad? porque vamos a tener que escribir estos datos para intentar entrar al sistema y ahora si, acá pusiéramos, bueno, que la contraseña es 1, 2, 3, ese sería el valor almacenado, ¿bien?, pero lo que vamos a hacer es correr antes la función md5, entonces, vamos a hacer “md5” y entre paréntesis, el valor que queremos modificar, es decir, que queremos esconder, encriptar. Aquí le damos control enter, y lo ha agregado, si mostramos ahora, por ejemplo, aquí hacemos “select todo from usuarios”, veremos que el valor del campo contraseña no es 1, 2, 3, es ,bueno, esta cadena que ha generado el método “md5”. Tengan en cuenta lo siguiente, cuando trabajemos aquí para leer los datos que vengan de la base de datos, precisamente la contraseña estará encriptada, ¿cómo haremos desde nuestro lado, desde el backend para desencriptarla? Bueno, con el mismo paquete con el que fue encriptada, md5. ¿Cómo hacemos eso? Y con esto terminamos la clase, bueno, así, vamos a hacer una instalación de ese mismo paquete en “npm install md5”. Okey y ya estamos preparados para trabajar con nuestra base de datos. ¿Qué nos falta? Bueno, ya vimos que se mostraba correctamente el error, cuando no podía conectarse con la base de datos, pero que eso no nos tumbaba, no nos derribaba a nuestro proceso en la página, seguía cargándose bien, no hemos probado qué ocurre ahora, que efectivamente creamos la base de datos. Bueno vamos a ver si se conecta “mpm, running dev” y “conexión con la base de datos establecida”, hemos triunfado. No siempre ocurre, siempre hay muchos tropiezos en la vida del desarrollador, pero, a veces, las cosas nos salen y celebramos. Nos vemos en la próxima.