**¿Qué es Express?**

Se usa para levantar un servidor web usando Express.js (un framework para Node.js). Este servidor está diseñado para ser la "espalda" (backend) de una aplicación, es decir, la parte que se encarga de recibir peticiones (como cuando un usuario se registra, inicia sesión, o pide información de artistas), procesarlas, y enviar respuestas.

app.use(express.json());//usa express.json() para que el servidor pueda recibir datos en formato JSON del frontend, es decir, que el servidor pueda recibir datos en formato JSON y los convierta a un objeto de JavaScript para manipularlos en los controladores del backend

// Rutas

app.use('/api/auth', authRoutes); // Rutas. Establece /api/auth como la ruta base o prefijo para todas las rutas que están definidas dentro del archivo authRoutes.js. Por ejemplo, si tienes una ruta de registro en authRoutes.js que es /register, la ruta completa para acceder a ella sería /api/auth/register.

app.listen(port, () => {// inicia tu servidor Express, haciendo que

empiece a escuchar peticiones en el puerto que le indicaste. Cuando procesa una petición, envía una respuesta de vuelta al cliente

  console.log(`Servidor corriendo en http://localhost:${port}`);

});

**¿Qué es Dotenv?**

**permite usar "variables de entorno". Las variables de entorno son como cajitas donde guardamos información secreta o que puede cambiar (como la contraseña de la base de datos o el puerto del servidor). No se sube a github o sitios publicos, queda guardado en nuestro proyecto personal.**

dotenv.config();

**Busca un archivo llamado .env en la carpeta principal de tu backend. Dentro de ese archivo .env es donde guardas información sensible o de configuración, como la contraseña de la base de datos, una clave secreta para los tokens JWT, o el puerto en el que quieres que corra el servidor. dotenv.config() lee ese archivo y hace que esas variables estén disponibles dentro de tu código**

**¿Qué es Cors?**

CORS es como un "permiso de comunicación". Cuando tu frontend (que corre, por ejemplo, en localhost:5500) quiere hablar con tu backend (que corre en localhost:3000), el navegador, por seguridad, necesita que el backend le dé permiso explícito para escuchar peticiones desde cualquier origen (direccion).

const cors = require('cors');//importamos cors para permitir el acceso a

la API desde el frontend, le dice al servidor que acepte peticiones de cualquier origen

* ARCHIVO DB.JS
* Trae las herramientas: Importa las librerías necesarias (mysql2 para MySQL y dotenv para secretos).
* Lee los secretos: Carga la información de conexión (usuario, contraseña, etc.) desde tu archivo .env.
* Prepara la conexión: Crea un objeto (db) que sabe cómo conectarse a tu base de datos MySQL usando esos secretos**. const db = mysql.createConnection({...}) crea la instancia db con toda la configuración necesaria (host, usuario, etc.), pero sin conectarse todavía.**
* **db.connect(...)**: Es la acción de **intentar** establecer la comunicación con tu base de datos MySQL. db.connect(...) inicia el intento de conexión.
* err => { ... } (El Callback): Es la instrucción que le das para después de que termine de marcar. Le dices: "Cuando sepas si te contestaron o no (si la conexión funcionó o falló), haz lo siguiente:".

if (err): "Si te dio error (err tiene algo), muéstrame el error por pantalla (console.error)".

else: "Si no hubo error (err está vacío), dime que todo fue bien (console.log)".

* module.exports = db; hace que tu objeto de conexión db esté disponible para ser

importado y usado por otros archivos de tu proyecto

db.connect(err => {

    if (err) {

        // Si hay un error, lo muestra en la consola

        console.error('Error de conexión a la base de datos:', err);

    } else {

        // Si todo va bien, avisa que se conectó

        console.log('Conectado a la base de datos');

    }

});

module.exports = db;

**¿Entender proceso de autenticación?**