🚀 EXPLICACIÓN COMPLETA (nivel principiante → experto)

🟦 1. Cargar la librería Express

const express = require('express');

¿Qué significa?

- require() sirve para **importar** módulos (librerías) en Node.js.
- Aquí estamos importando **Express**, un framework que te ayuda a construir servidores.

¿Por qué lo usamos?

Porque Express hace SUPER fácil:

- crear rutas,
- levantar servidores,
- manejar JSON,
- procesar peticiones.

Sin Express, tendrías que hacerlo con Node puro y es mucho más complicado.

🟦 2. Crear la aplicación Express

const app = express();

¿Qué significa?

- Estamos **creando una instancia** de Express.
- app es tu **servidor**.
- A partir de aquí, todas las rutas, configuraciones, middlewares y funciones se hacen a través de app.

Analógicamente:

Piensa que es como **crear un proyecto nuevo** en donde vas a agregar rutas, funciones, seguridad, etc.

🟦 3. Definir el puerto

const port = 3000;

¿Qué significa?

El puerto es como la **puerta de entrada** a tu servidor.

- 3000 es un número estándar para desarrollo.
- También podrías usar 5000, 8080, 4000, lo que sea.

Cuando entras a:

👉 http://localhost:3000

estás entrando a tu servidor por esa "puerta".

🟩 4. Middleware para interpretar JSON

```
app.use(express.json());
```

¿Qué es un middleware?

Es una función que **se ejecuta antes que las rutas**.

Sirve para procesar o modificar la información de la petición.

¿Qué hace este middleware EXACTAMENTE?

Permite que tu servidor entienda **JSON enviado desde el cliente**.

Ejemplo:

Si alguien manda:

```
{
  "nombre": "Angel"
}
```

Sin este middleware, Express **no sabría leer ese JSON**.

Con express.json():

- Express lo recibe
- Lo convierte a un objeto
- Lo pone en req.body

🟦 5. Ruta principal /

```
app.get('/', (req, res) => {
  res.send('Servidor Node.js funcionando correctamente 😎 WOWOWO');
});
```

¿Qué significa?

- app.get() crea una ruta para peticiones GET.
- '/' es la ruta raíz → es tu página principal.
- (req, res) => { ... } es una función que se ejecuta cuando alguien entra a esa ruta.

Detalles:

- req = información que el cliente envía
- res = lo que tú regresas al cliente

¿Qué hace esta ruta?

Cuando visitas:

👉 http://localhost:3000/

el servidor te responde:

Servidor Node.js funcionando correctamente 😎 WOWOWO

🟦 6. Ruta de API /api/status

```
app.get('/api/status', (req, res) => {
  res.json({
    ok: true,
    message: 'API lista y corriendo',
    hora: new Date().toISOString()
  });
});
```

¿Qué significa?

Otra ruta GET, pero esta vez dentro de una subcarpeta /api.

Es común que tus APIs vayan dentro de /api.

¿Qué hace?

Regresa JSON en lugar de texto:

```
{
  "ok": true,
  "message": "API lista y corriendo",
  "hora": "2025-12-02T..."
}
```

¿Por qué es útil?

- Para pruebas
- Para un frontend (React, Next.js)
- Para monitorear el servidor
- Para saber si tu API está viva

🟩 7. Levantar el servidor

```
app.listen(port, () => {
  console.log(`Servidor escuchando en http://localhost:${port}`);
});
```

¿Qué significa?

Esto **inicia el servidor** en el puerto que definiste arriba (3000).

Cuando corre, te imprime en consola:

Servidor escuchando en http://localhost:3000

¿Qué pasa si no pones esta línea?

👉 **Tu servidor no existe.**

Express ya está configurado pero NO está escuchando peticiones.

app.listen() es como **prender** el servidor.

🎯 **RESUMEN FINAL (explicado como si fuera para un niño inteligente)**

- **express** = herramienta para hacer servidores fácil.
- **app = express()** → creas tu servidor.
- **port = 3000** → eliges por dónde entran los clientes.
- **app.use(express.json())** → tu servidor ya entiende JSON.
- **app.get('/')** → ruta principal (regresa texto).
- **app.get('/api/status')** → ruta API (regresa JSON).
- **app.listen()** → prendes el servidor.