

Node JS

Robinson Andres Cortes Galindo



Creación de un proyecto en Node js npm init -y

El comando `-y` le dice a `npm init` que use los valores predeterminados para todas las preguntas. Esto significa que no se te pedirá que proporciones ninguna información.

Instalación de dependencias

- `express`
 - `mongodb`
 - `mongoose`
- 
- A thick green curved line starts from the bottom right corner and curves upwards and to the left, ending near the center of the right edge of the slide.

```
Express npm install express
```

- Express.js es el framework backend más popular para Node.js, y es una parte extensa del ecosistema JavaScript. Está diseñado para construir aplicaciones web de una sola página, multipágina e híbridas, también se ha convertido en el estándar para desarrollar aplicaciones backend con Node.

```
mongodb npm install mongodb
```

- MongoDB es una base de datos NoSQL orientada a documentos que se utiliza para almacenar grandes volúmenes de datos

```
mongoose npm install mongodb
```

- es una biblioteca de programación orientada a objetos de JavaScript que crea una conexión entre MongoDB y el entorno de tiempo de ejecución de JavaScript de Node.js

```
nodemon npm install --global nodemon
```

- Es una herramienta que nos ayuda a reiniciar nuestro servidor (o algún otro proceso de nuestra computadora) cada vez que hagamos un cambio en nuestro código

Agregar librerías

```
const express = require('express');
```

Importa el framework web Express.js, un componente esencial para construir aplicaciones y APIs web en Node.js. El framework proporciona un conjunto de funcionalidades y herramientas para:

- Gestionar rutas y solicitudes HTTP
- Renderizar contenido
- Manejar cookies y sesiones
- Trabajar con middleware
- Y mucho más

Variable 'express': Se crea una variable llamada 'express' para almacenar la referencia al framework, permitiendo su uso en el resto del código.

```
const mongoose = require( 'mongoose' ) ;
```

- la librería Mongoose, que sirve como un Object Document Mapper (ODM) para interactuar con bases de datos MongoDB desde Node.js. Mongoose facilita la definición y gestión de modelos de datos, el mapeo de estos modelos a documentos de MongoDB, y la realización de operaciones de consulta, inserción, actualización y eliminación de datos en la base de datos.
- **Variable 'mongoose':** Se crea una variable llamada 'mongoose' para almacenar la referencia a la librería, permitiendo su uso posterior para interactuar con MongoDB.



</Riwi>

```
mongoose.connect('cadena de conexion mongo', {  
  useNewUrlParser: true,  
  useUnifiedTopology: true,  
});
```

useNewUrlParser

Esta opción indica que se debe utilizar el nuevo analizador de URL para interpretar la cadena de conexión a la base de datos. El analizador moderno ofrece mejoras en la seguridad y el rendimiento.

useUnifiedTopology

Esta opción habilita el nuevo motor de descubrimiento y monitoreo del servidor de MongoDB. Este motor proporciona una forma más confiable y eficiente de descubrir y gestionar las conexiones a los servidores de la base de datos, especialmente en entornos con clústeres o replica sets.



</Riwi>

```
const db = mongoose.connection;
```

- **Al utilizar `console.error.bind(console)`, se asegura que el valor de `this` dentro de la nueva función creada sea siempre el objeto `console`.** Esto garantiza que cualquier propiedad o método que intente utilizar dentro de la función (por ejemplo, para formatear el mensaje de error) esté disponible y se comporte como se espera.



</Riwi>

```
db.on('error', console.error.bind(console, 'Error de conexión a la base de datos'));
```

</Riwi>

```
db.once( 'open', () => {  
  // Aqui toda la funcionalidad si la conexion esta funcionando  
});
```

db.once

escuchar una vez

db.on

escuchar continuamente



</Riwi>

```
const userSchema = new mongoose.Schema({  
  nombre: { type: String, required: true },  
  apellido: { type: String, required: true }  
});
```



</Riwi>

```
const user = mongoose.model('users', userSchema);
```



</Riwi>

```
const app = express();
```

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



</Riwi>

```
app.get('/api/users', async (req, res) => {  
  const users = await user.find();  
  res.json(users);  
});
```



</Riwi>

```
app.listen(3000, () => {  
  console.log('Servidor escuchando en el puerto 3000');  
});
```

A Escribir REST