Paquetes:

1. primero instamos node

$ npm init -y

2. $ npm install express

Luego tenemos que crear el servidor de NODE JS

Direccionamiento básico

El direccionamiento hace referencia a la determinación de cómo responde una aplicación a una solicitud de cliente en un determinado punto final, que es un URI (o una vía de acceso) y un método de solicitud HTTP específico (GET, POST, etc.).

Donde:

* app es una instancia de express.
* METHOD es un [método de solicitud HTTP](http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol).
* PATH es una vía de acceso en el servidor.
* HANDLER es la función que se ejecuta cuando se correlaciona la ruta.

Rutas básicas:

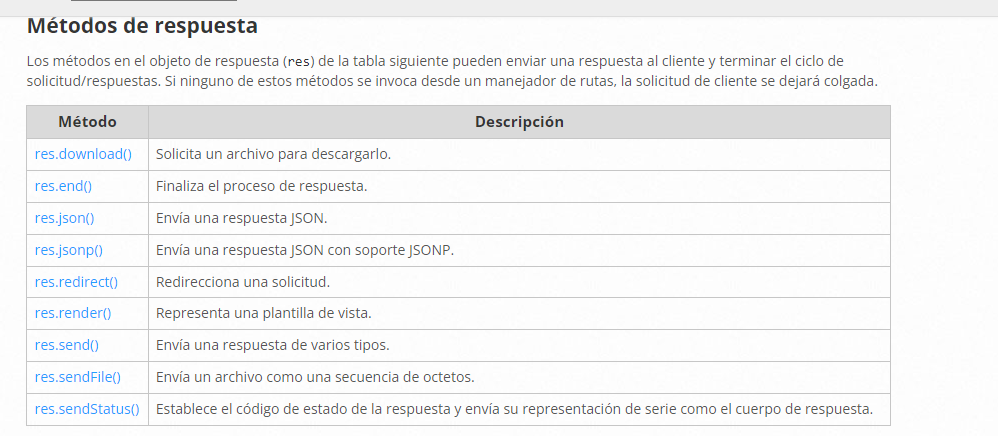


Para utilizar la ejecución dinámica



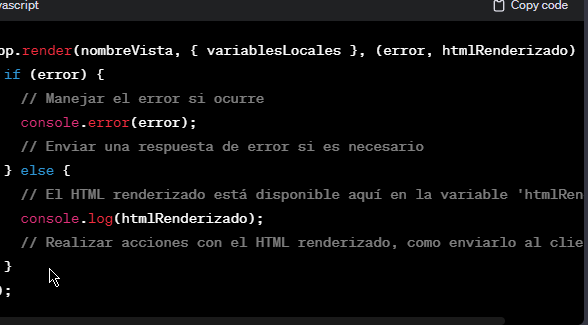
Y ejecutamos el servidor asi:  
**npm run dev**

Express da soporte a los siguientes métodos de direccionamiento que se corresponden con los métodos HTTP: get, post, put, head, delete, options, trace, copy, lock, mkcol, move, purge, propfind, proppatch, unlock, report, mkactivity, checkout, merge, m-search, notify, subscribe, unsubscribe, patch, search y connect.



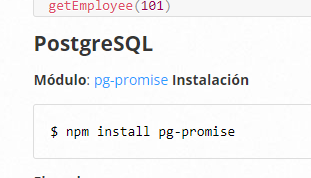
La función **app.render()** en Express se utiliza para renderizar una vista específica (template) de tu aplicación y obtener el HTML resultante, pero a diferencia de **res.render()**, esta función no envía directamente la vista renderizada al cliente, sino que devuelve el HTML generado a través de una función de callback.

Aquí está la estructura básica de cómo usar **app.render()**:



* **nombreVista**: Es el nombre de la vista (template) que deseas renderizar. Express buscará esta vista en el directorio de vistas configurado en tu aplicación.
* **{ variablesLocales }**: Es un objeto opcional que contiene las variables que deseas pasar a la vista para su renderizado. Estas variables pueden incluir datos dinámicos que la vista utilizará para generar el HTML.
* La función de callback recibe dos parámetros:
  + **error**: Si ocurriera algún error durante el proceso de renderizado, se capturaría aquí.
  + **htmlRenderizado**: Es el HTML resultante después de renderizar la vista con las variables proporcionadas. Esta variable contiene el HTML que puedes usar para realizar acciones, como enviarlo al cliente o realizar operaciones adicionales.

Usualmente, **app.render()** se utiliza cuando necesitas obtener el HTML de una vista renderizada dentro del servidor para realizar alguna operación adicional, como enviarlo en una respuesta personalizada al cliente o usarlo en alguna lógica del servidor.

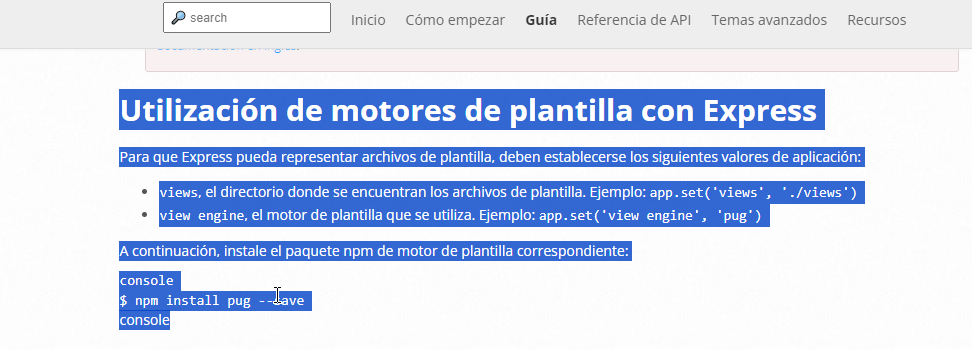
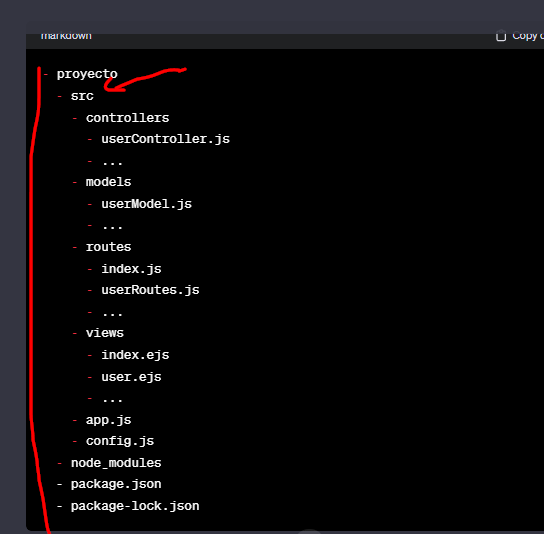


**Continuar:**

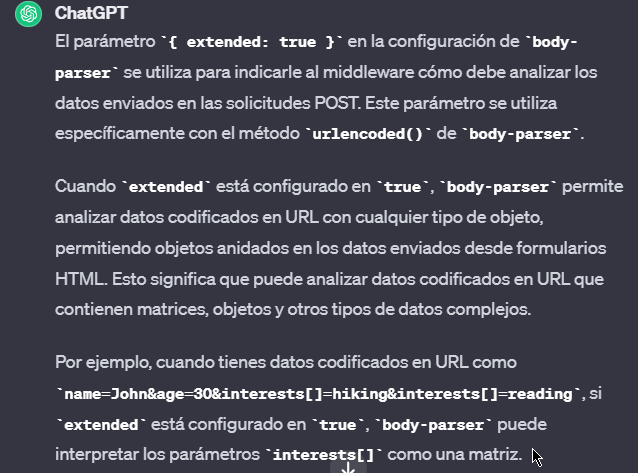
<https://www.w3schools.com/nodejs/nodejs_mysql_update.asp>

<https://nodejs.org/docs/latest-v18.x/api/>

<https://expressjs.com/es/4x/api.html#res.app>





El término "CORS" (Cross-Origin Resource Sharing) se refiere a una política de seguridad implementada por los navegadores web para controlar cómo los scripts de una página web pueden interactuar con recursos en un dominio diferente al de la propia página. Específicamente, CORS se utiliza para permitir o restringir las solicitudes HTTP entre diferentes dominios.

Cuando estás construyendo una API en Express u otro servidor, es posible que te encuentres con problemas de CORS si tu API está en un dominio diferente al de la aplicación cliente que intenta acceder a ella. En tales casos, es probable que las solicitudes entre los dominios sean bloqueadas por el navegador por razones de seguridad.

Para solucionar problemas de CORS en Express, puedes utilizar el middleware cors, que es una biblioteca que facilita la configuración de las cabeceras CORS para permitir solicitudes desde diferentes dominios.

Aquí tienes un ejemplo de cómo usar cors en tu aplicación Express:

Instala el paquete cors:

