Express

El framework de Node

¿Qué es Express?

Express es un framework para aplicaciones web de Node.js que proporciona un conjunto de características para aplicaciones web y móviles.

Es un framework de código abierto desarrollado y mantenido por la fundación Node.js.

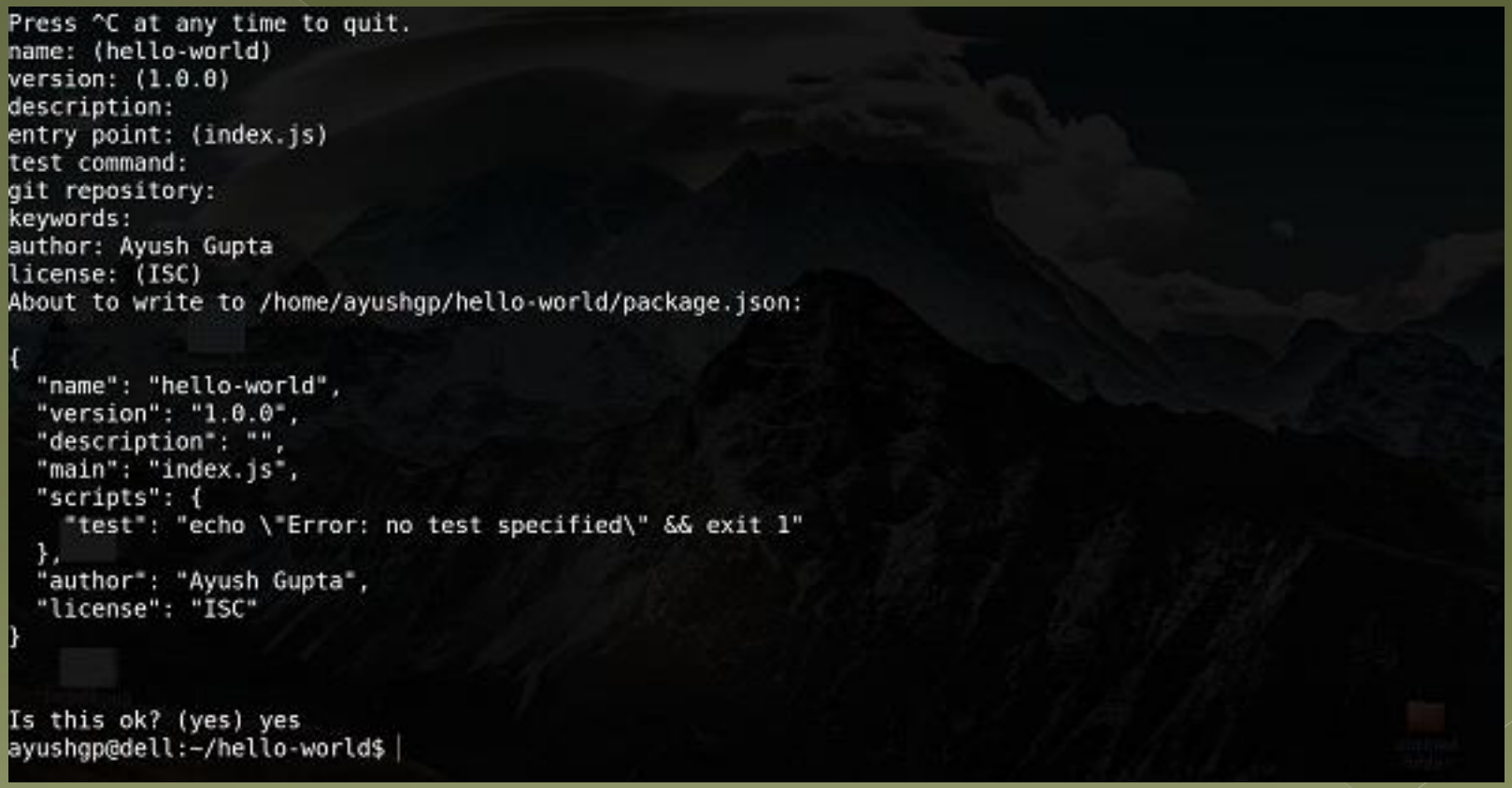
Instalación y configuración

La instalación se realiza a través de **npm.**

Primero - Crear el archivo package.json utilizando npm.
El archivo package.json, contiene todos los detalles del proyecto.

1) Crear la carpeta para el proyecto
2) A través de la terminal posicionarnos en la carpeta
3) Ejecutar: npm init

Package.json



Instalar Express

Ejecutar: npm install --save express

El flag **--save** puede ser reemplazada por **-S**. Este indicador asegura que Express se agrega como una dependencia al archivo **package.json.**

Nota: Esto tiene una ventaja, la próxima vez que necesitemos instalar todas las dependencias de nuestro proyecto, podemos ejecutar el comando **npm install** y encontrará las dependencias en este archivo e instalarlas para nosotros.

Hola Mundo

1 - Crear un archivo index.js
2 - Escribir el siguiente bloque de código:

```
var express = require('express');
var app = express();
app.get('/', function(req, res){
    res.send("Hello world!");
});
app.listen(3000);
```

app.get(route, callback)

Esta función le dice qué hacer cuando se ejecuta un ***request*** en la ruta determinada.
La función de ***callback*** tiene 2 parámetros, solicitud (req) y respuesta (res).
El objeto request (req) representa la solicitud HTTP y tiene propiedades del query string, parámetros, cuerpo, encabezados HTTP, etc.
El objeto ***response*** representa la respuesta HTTP que la aplicación Express envía cuando recibe una solicitud HTTP.

res.send()

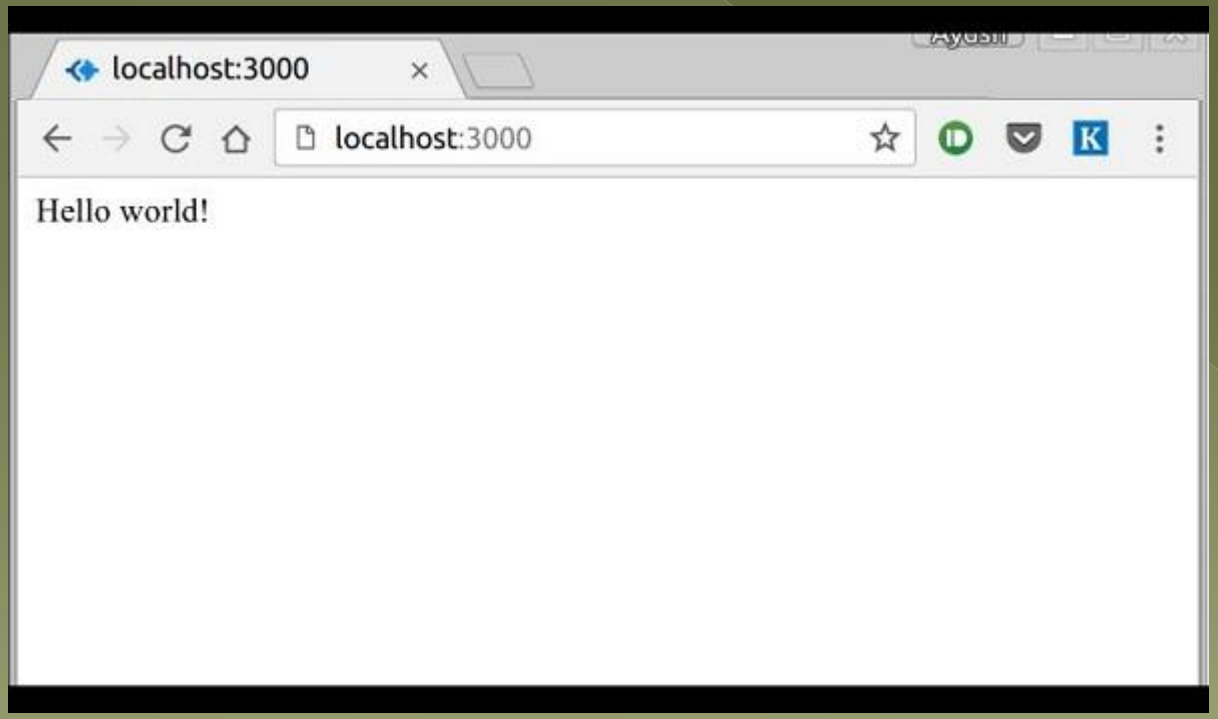
Esta función toma un objeto como entrada y lo envía al cliente solicitante. Aquí estamos enviando la cadena "¡Hello world!".

app.listen(port, [host], [backlog], [callback])

S.No.	Argumento y Descripción
1	**port** Número del puerto en el cual el servidor acepta request.
2	**host** Nombre de dominio. Es necesario cuando se despliega la app en la nube.
3	**backlog** Número máximo de conexiones pendientes. Default 511.
4	**callback** Función asincronica que es llamada cuando el servidor comienza a escuchar requests.

Hola Mundo

3 - Ejecutar en la consola: node index.js
4 - Abrir el browser



Dentro de la carpeta de mi proyecto.

Routing

Organizando la web app.

app.method(path, handler)

Se puede aplicar a cualquiera de las acciones HTTP - get, set, put, delete.

También existe un método alternativo, que se ejecuta independientemente del tipo de solicitud.

La ruta es la ruta en la que se ejecutará la solicitud.

Handler es una función de devolución de llamada que se ejecuta cuando se encuentra un tipo de solicitud coincidente en la ruta relevante.

Ejemplo de Routing

```
var express = require('express');
var app = express();
app.get('/hello', function(req, res){
    res.send("Hello World!");
});
app.listen(3000);
```

Ruta

Callback que retorna el objeto a mostrar.

Multirequest al mismo route

```
var express = require('express');
var app = express();

app.get('/hello', function(req, res){
        res.send("Hello World!");
});

app.post('/hello', function(req, res){
    res.send("Llamo al método post!");
 });

app.listen(3000);
```

Routers

Express.Router, nos permite separar las rutas de nuestros scripts de forma simple y eficiente.

Implementación

index.js

En customer.js, están las rutas.

customer.js

customer.js

```
var express = require('express');
var router = express.Router();
router.get('/', function(req, res){
    res.send('GET route on customer.');
});
router.post('/', function(req, res){
    res.send('POST route on customer.');
});
module.exports = router;
```

Index.js

```
var express = require('Express');
var app = express();
var customer = require('./customer.js');
app.use('/customer', customer);
app.listen(3000);
```

Conecta el **router** de customer
con la ruta especificada.